Scientific Computing (wi4201)

C. Vuik and D.J.P. Lahaye

2014

Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Preface

In these lecture notes an introduction to Scientific Computing is presented. We start by discussion the nature and properties of various sources of errors in scientific computing. Thereafter a summary is given on finite difference methods of boundary value problems. The result of this are large, sparse systems of linear equations. Fast solution of these systems is very urgent nowadays. The size of the problems can be 10^9 unknowns and 10^9 equations. For medium size problems, direct solution methods are the methods of choice. For large problems iterative solution methods are required. In order to obtain experience we start with basic iterative methods. After that we discuss Multi-grid and Preconditioned Krylov subspace methods, which are state of the art. A distinction is made between various classes of matrices. Besides fast computation of the solution of a system of linear equations, estimates for eigenvalues and eigenvectors are also of primary importance for stability analysis, wave phenomena and other physical quantities. In the final chapter we discuss iterative methods to approximate eigenvalues. These method range from the basic power method to the state of the art Lanczos and Arnoldi methods.

At the end of the lecture notes many references are given to state of the art Scientific Computing methods. Here, we will discuss a number of books which are nice to use for an overview of background material. First of all the books of Golub and Van Loan [25] and Horn and Johnson [34] are classical works on all aspects of numerical linear algebra. These books also contain most of the material, which is used for direct solvers. Varga [72] is good starting point to study the theory of basic iterative methods. Krylov subspace methods and multigrid are discussed in Saad [53] and Trottenberg, Oosterlee and Schüller [64]. Other books on Krylov subspace methods are [3, 5, 7, 29]. A classic reference for eigenvalue problems is the work of Wilkinson [75]. Good starting points for iterative solvers are the books by Parlett [45] and Chatelin [8, 9].

Delft, September 2014, C. Vuik

Contents

Chapter 1

Introduction

Computational Science and Engineering (CSE) is a rapidly growing multidisciplinary area with connections to the sciences, engineering, mathematics and computer science. CSE focuses on the development of problem-solving methodologies and robust tools for the solution of scientific and engineering problems. We believe that CSE will play an important if not dominating role for the future of the scientific discovery process and engineering design. Below we give a more detailed description of Computational Science and Engineering. For more information on CSE we refer to the Society for Industrial and Applied Mathematics (SIAM).

In CSE we deal with the simulation of processes, as they occur amongst others in the engineering, physical and economic sciences. Examples of application areas are fluid dynamics (the aerodynamics of cars and aircrafts, combustion processes, pollution spreading), semi-conductor technology (breeding of crystals, oxidation processes), weather and climate prediction (the growing and tracking of tornados, global warming), applied physics (many particle simulations, protein folding, drug design) but also financial mathematics (prediction of stock and option prices). Simulation is nowadays an equal and indispensable partner in the advance of scientific knowledge next to the theoretical and experimental research.

It is characteristic for CSE that practical relevant results are typically achieved by combining methods and research results from different scientific areas (see Figure 1.1).

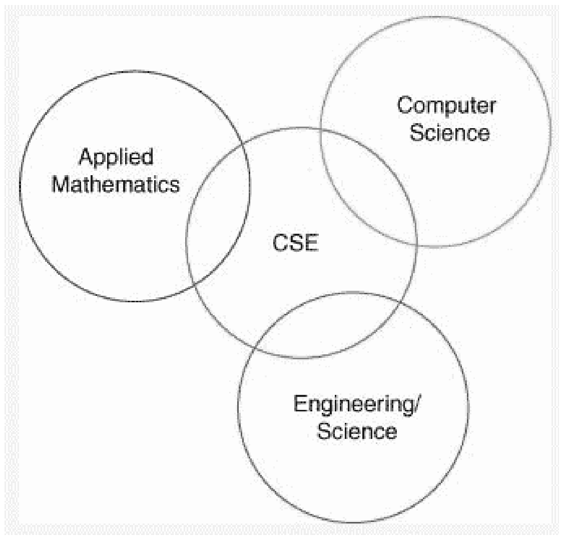


Figure 1.1: Embedding of computational science and engineering.

The application area brings the typical problem-dependent know-how, the knowledge to formulate the problem and model and to verify the computed results by means of real experiments. Applied mathematics deals with the definition of a mathematical model, existence and uniqueness results and develops efficient methods to solve the mathematical model with a computer. Computer science typically takes care for the usability, and programming of modern computers and designs powerful software packages. Especially the interplay of the disciplines is necessary for success. For instance: it is possible to parallelize a poor mathematical method and implement it on a supercomputer, but this does not help us much! We need much more a continuing development of mathematical models and numerical algorithms, the transfer of the algorithms into powerful and user-friendly software, running this on state-of-the-art computers that steadily increase their performance.

Basically, we can distinguish a number of important steps in simulation:

- Setting up a model.
  At the start of each simulation we have the development of a mathematical model of the process of interest. This must be a simplified image of the reality that contains many relevant phenomena. It must be formulated such that the model has a (unique) solution. Often we obtain a system of (not analytically solvable) differential equations.

- The analytical treatment.
  Analytical tools can be used to obtain properties of the solution: existence, uniqueness, maximum principle etc. Furtermore for simple problems an analytical solution can be found. In a number of cases approximate solutions can be derived using an asymptotical approach.

- The numerical treatment.
  Since a computer can only handle discrete numbers, we have to discretize the model (the equations and the solution domain), so that the computer can deal with them. For the solution of the resulting matrix from the discrete equations, mathematicians provide efficient methods.

- The implementation.
  Next to the choice of computer language and data structures, especially the distributed computation is of major importance.

- The embedding.
  The numerical simulation is just one step in the product development in industrial applications. Therefore, we need interfaces, so that we can link the simulation programme with CAD tools. Only in this way it is, for example, possible to use aerodynamics car simulation results on drag coefficients at an early stage into the design process.

- The visualization.
  Typically, after a simulation we have huge data sets (not only one drag coefficient as in the example above). Often we need the velocity in the complete flow domain, or one is interested for the optimal control of a robot in the path of the robot arm. We need to present such results with the help of computer graphics, so visualization is very important.

- The validation.
  After long computer times with many many computations, we have the result of a simulation. It is of a primary importance to verify the results obtained.

1.1 Errors in scientific computing

In scientific computing, we wish to compute an exact solution u, but we obtain an approximation $\widehat{u}_h$ on a numerical grid after a number of iterations of an iterative solution method implemented on a computer. An important aspect of scientific computing is therefore the understanding of the errors made during the computation of solutions. The aim is to find an approximation $\widehat{u}_h$ for the solution u *with a known accuracy*. One has to consider the possible errors occurring and their influence on the accuracy of the approximation. Obtaining an approximate solution without knowing anything about the approximation error is useless.

Error measures include the absolute and relative error in some norm $|\cdot|$:

$$|u - \widehat{u}_h| \leq \epsilon \quad \text{Estimate of the \underline{absolute} error,}$$

$$\frac{|u - \widehat{u}_h|}{|u|} \leq \epsilon \quad \text{Estimate of the \underline{relative} error.}$$

An obvious area of conflict is that estimates are only of interest, if they are more or less realistic. Especially for academic model problems it may often be possible to get sharp error estimates, but this is far less trivial, or even not yet reached in research for solutions of real-life, highly nonlinear systems of partial differential equations. An overview of errors that can be made in solving real-life problems with mathematical methods in scientific computing is presented in Figure 1.2.

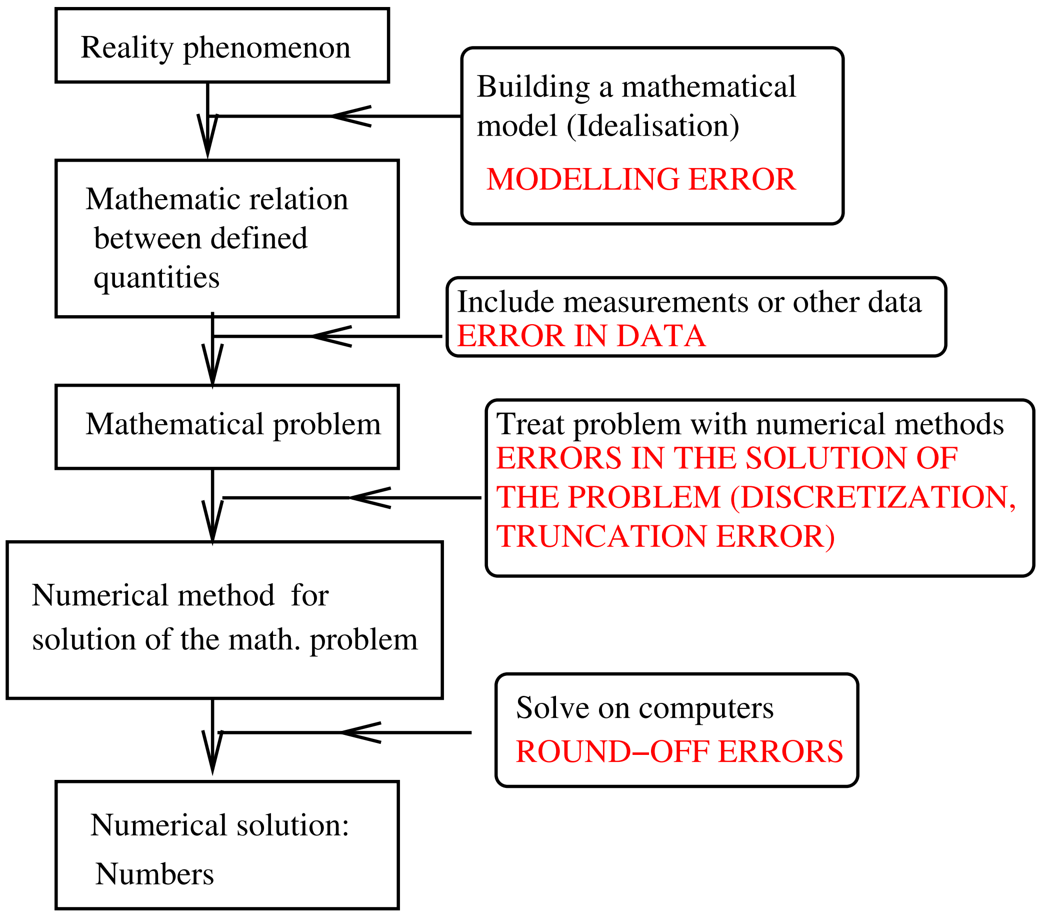

Reality phenomenon
Building a mathematical model (Idealisation)
MODELLING ERROR
Mathematic relation between defined quantities
Include measurements or other data
ERROR IN DATA
Mathematical problem
Treat problem with numerical methods
ERRORS IN THE SOLUTION OF THE PROBLEM (DISCRETIZATION, TRUNCATION ERROR)
Numerical method for solution of the math. problem
Solve on computers
ROUND–OFF ERRORS
Numerical solution: Numbers

Figure 1.2: Overview of errors made in scientific computing.

The notion of the *modeling error* is as follows. Practical problems are often physical, chemical, biological or economical applications. It is necessary to set up a mathematical model, which is typically a partial differential equation (pde) here. In order to analyze only certain features of a complicated problem, a linear simplification of a nonlinear process may be used, for simplicity. In fluid mechanics, for example, one distinguishes between laminar (steady, ordered) and turbulent (unsteady, nonlinear) flows. Whereas the solution of laminar flow problems is fully understood, for turbulent flows modeling is still state-of-research. Another distinction is between mathematical models involving friction, leading to the Navier-Stokes equations, versus models without friction, the Euler equations. Including friction in the mathematical model requires the use of more complicated numerical solution methods than the use of a frictionless model. The insight in the impact of the choice of mathematical model on the results obtained is indispensable.

Data errors may occur because, except for trivial cases, most data that enter a computation are afflicted with errors, for example with measurement errors. They can have a big or a small effect on a numerical solution. This typically depends on the condition of the problem. If a problem is well-posed small changes in data do not have a significant effect on the solution, whereas for an ill-posed problem the opposite is true.

Discretization errors occur because we need to solve a (continuous) problem under consideration on a numerical grid, as an analytic solution is not available. The approximation of a function on a discrete grid means that we try to recover a function by means of function values by a finite number of points. As only N values are obtained one ends up with an approximate solution curve (see Figure 1.3).

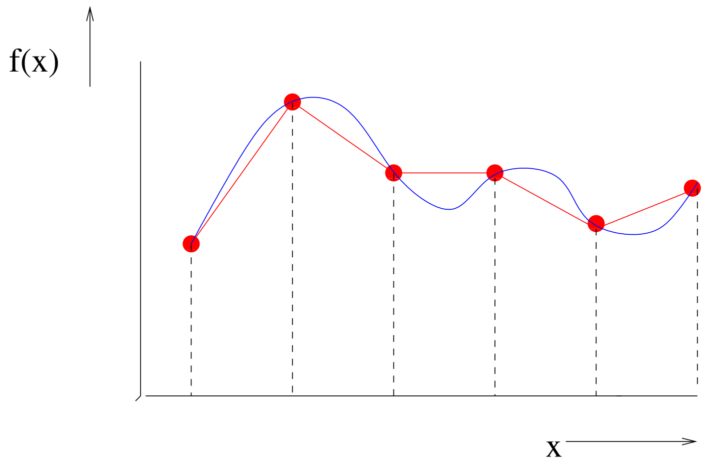


Figure 1.3: An approximate solution curve as only N values are obtained on a numerical grid.

Round off errors occur as only digital numbers of a finite length are used in computers. Practically, this means that already during multiplication and division numbers are rounded. Each separate round off error may be small, however, during long calculations many small errors can add up, to make a solution totally useless. In scientific computing one therefore distinguishes stable from unstable algorithms, the first class leading to reliable numerical solutions.

Chapter 2

Concepts from Numerical Linear Algebra

In this section we will introduce concepts from numerical linear algebra. Textbook references for this chapter include [38, 52].

2.1 Vectors and Matrices

In this chapter we will use bold lower-case Roman letters to denote real-valued vectors such as e.g. $\mathbf{u} \in \mathbb{R}^n$ having components u_i where $1 \leq i \leq n$. We will use the notation A and R to denote a real-valued square and rectangular matrix, respectively. More precisely, we will assume that $A \in \mathbb{R}^{n \times n}$ with components a_{ij} where $1 \leq i, j \leq n$ and that $R \in \mathbb{R}^{m \times n}$ with components r_{ij} where $1 \leq i \leq m$ and $1 \leq j \leq n$. We will also write $A = (a_{ij})$. We will use the Matlab-like notation $A(i, :)$ and $A(:, j)$ to denote the ith row and jth column of A, respectively. We will use the symbols $\emptyset_{n \times n}$ and $I_{n \times n}$ to denote the $n \times n$ zero and identity matrix, respectively.

We will denote by $|A|$ and $|R|$ the matrices obtained by taking the absolute value of A and R entry-wise, respectively. In what follows we will say that A is positive and write $A \geq \emptyset_{n \times n}$ iff for all components $a_{ij} \geq 0$. This notion induces a partial ordering on the set of $n \times n$ matrices by defining $A_1 \leq A_2$ to hold if and only if (iff) $A_2 - A_1 \geq \emptyset_{n \times n}$. It is easy to verify that the product of positive matrices is again positive. We will denote the (left and right) inverse of A by A^{-1}.

2.1.1 Rank of a Matrix

The rank of a matrix $R \in \mathbb{R}^{m \times n}$ is defined as the size of the largest collection of linear independent rows of columns of the matrix. It can be computed as the dimension of the largest square non-singular submatrix of R.

2.1.2 Kronecker Product of Matrices

In these lecture notes, matrices and vectors represent discretized differential operators and fields. In particular circumstances the discrete differential operator in two and higher dimensions can be represented as a Kronecker product of its one-dimensional counterpart.

Definition 2.1.1 *Given two rectangular matrices $R_1 \in \mathbb{R}^{m_1 \times n_1}$ and $R_2 \in \mathbb{R}^{m_2 \times n_2}$ its Kronecker product denoted by $R_1 \otimes R_2 \in \mathbb{R}^{m_1 m_2 \times n_1 n_2}$ is formed by replacing each entry $r_{1,ij}$ of R_1 by $r_{1,ij} \cdot R_2$.*

Example 2.1.1

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \otimes \begin{bmatrix}0 & 5\\ 6 & 7\end{bmatrix} = \begin{bmatrix}1\cdot 0 & 1\cdot 5 & 2\cdot 0 & 2\cdot 5\\ 1\cdot 6 & 1\cdot 7 & 2\cdot 6 & 2\cdot 7\\ 3\cdot 0 & 3\cdot 5 & 4\cdot 0 & 4\cdot 5\\ 3\cdot 6 & 3\cdot 7 & 4\cdot 6 & 4\cdot 7\end{bmatrix} = \begin{bmatrix}0 & 5 & 0 & 10\\ 6 & 7 & 12 & 14\\ 0 & 15 & 0 & 20\\ 18 & 21 & 24 & 28\end{bmatrix}$$

2.2 Eigenvalues, Eigenvectors and Spectrum

Definition 2.2.1 *The non-zero vector* $\mathbf{v}^{[k]} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ *is an* eigenvector *corresponding to the* eigenvalue $\lambda_k \in \mathbb{C}$ *iff* $A\,\mathbf{v}^{[k]} = \lambda_k \mathbf{v}^{[k]}$. *The algebraic multiplicity of* λ_k *is defined as the multiplicity of the root of* λ_k *of the characteristic equation det*$(A - \lambda I) = 0$. *The geometric multiplicity of* λ_k *is defined as the dimension of the space spanned by the corresponding eigenvectors. The set of all eigenvalues of A is called the* spectrum *of A and will be denoted as* $\sigma(A)$.

To explicitly link the eigenvalue to the matrix we will use the notation $\lambda_k(A)$.

Two $n \times n$ matrices A_1 and A_2 are called *similar* iff a non-singular $n \times n$ matrix V exists such that $A_2 = V^{-1}A_1V$. If A_1 and A_2 are similar, then they have the same spectrum. A matrix is called *diagonalizable* if it is similar to a diagonal matrix. If A is diagonalizable and can therefore be written as $A = V^{-1}DV$ for some matrices V and D, the k-th power of A can be computed as $A^k = V^{-1}D^kV$ [38].

The concept of singular values generalizes the concept of eigenvalues from square to rectangular matrices.

2.2.1 Eigenvalues of Tridiagonal Matrices

As tridiagonal matrices will often appear in this course, we give the following result on the eigenvalues of these matrices.

Theorem 2.2.1 *Assuming that a, b and c are real numbers such that* $bc < 0$ *and denoting* ι *the imaginary unit* ($\iota^2 = -1$)*, the eigenvalues of the tridiagonal Toeplitz matrix*

$$A = \begin{pmatrix} a & b & 0 & \dots & \dots & \dots & 0\\ c & a & b & 0 & \dots & \dots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & \dots & \dots & 0 & c & a & b\\ 0 & \dots & \dots & \dots & 0 & c & a \end{pmatrix} \in R^{n\times n}$$

are given by

$$\lambda_k(A) = a + 2\iota\sqrt{-bc}\cos(\frac{k\pi}{n+1})\,.$$

The proof of this theorem as well as the details of the frequently occurring case that $b = c$ is left as an exercise.

2.3 Symmetry

Definition 2.3.1 *The* transpose *of A, denoted by* A^T*, is an* $n \times n$ *matrix with components* $a^T_{ij} = a_{ji}$.

Example 2.3.1 *If* $R \in \mathbb{R}^{m\times n}$*, then* $R^T \in \mathbb{R}^{n\times m}$*,* $R^TR \in \mathbb{R}^{n\times n}$ *and* $RR^T \in \mathbb{R}^{m\times m}$.

Definition 2.3.2 *The matrix A is* symmetric *iff* $A^T = A$.

The theory of symmetric matrices is very rich as argued e.g. in Chapter 7 of [38]. We recall here some basic facts.

Theorem 2.3.1 *The eigenvalues of a symmetric matrix A are real, i.e.,* $A = A^T \Rightarrow \sigma(A) \subset \mathbb{R}$.

A proof of this theorem is given in e.g. [52] (Theorem 1.9). This theorem implies that the eigenvalues of a symmetric matrix can be ordered in increasing magnitude as follows

$$\sigma(A) = \{\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n\}\,, \tag{2.1}$$

where each eigenvalue is repeated according to its algebraic multiplicity.

An $n \times n$ matrix Q is called *orthogonal* iff its n columns are orthonormal, i.e., iff $Q^T Q = I$. The following theorem will be very helpful in computing the k-th power of a symmetric matrix

Theorem 2.3.2 *A symmetric matrix A is orthogonally diagonalizable, i.e., there exists an orthogonal matrix Q and a diagonal matrix D such that* $A = Q^T D Q$. *The entries of D are the eigenvalues of A and the columns of Q span the eigenspaces of A.*

2.4 Positive Definiteness

If $\mathbf{u}$ is a column (standing) vector, then $\mathbf{u}^T$ is a row (lying) vector, and the product $\mathbf{u}^T A\mathbf{u}$ is a scalar. This scalar is zero if $\mathbf{u} = \mathbf{0}$, independently of A. Other choices for $\mathbf{u}$ are considered in the following definition.

Definition 2.4.1 *The matrix A is called positive definite (*positive semi-definite*) iff*

$$\forall \mathbf{u} \in \mathbb{R}^N \setminus \{\mathbf{0}\} : \mathbf{u}^T A\mathbf{u} > 0 \ (\mathbf{u}^T A\mathbf{u} \geq 0)\,. \tag{2.2}$$

In the next theorem the spectrum of symmetric positive definite (SPD) and symmetric positive semi-definite (SPSD) is considered.

Theorem 2.4.1 *The spectrum of a symmetric positive definite (positive semi-definite) matrix A are strictly positive (positive), i.e., A SPD* $\Rightarrow \sigma(A) \subset \mathbb{R}_0^+$ *(A SPSD* $\Rightarrow \sigma(A) \subset \mathbb{R}^+$*).*

2.5 Vector and Matrix Norms

2.5.1 Inner Product

The inner product is a positive definite bilinear form on $\mathbb{R}^n$. More precisely, we have the following definition.

Definition 2.5.1 *Denoting* $c \in \mathbb{R}$ *a scalar and* $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ *vectors, the inner product* $(.,.) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *is a function that satisfies the following properties*

1. **symmetry**
$$(\mathbf{u}, \mathbf{v}) = (\mathbf{v}, \mathbf{u})$$

2. **linearity in the first argument**
$$\begin{aligned}(c\,\mathbf{u}, \mathbf{v}) &= c\,(\mathbf{u}, \mathbf{v}) \\ (\mathbf{u} + \mathbf{v}, \mathbf{w}) &= (\mathbf{u}, \mathbf{w}) + (\mathbf{v}, \mathbf{w})\end{aligned}$$

3. **positive definiteness**
$$(\mathbf{u}, \mathbf{u}) \geq 0 \textit{ with equality only for } \mathbf{u} = \mathbf{0}\,.$$

Example 2.5.1 *The scalar product* $\mathbf{u}^T\mathbf{v} = \sum_{i=1}^n u_i v_i$ *is an inner product on* $\mathbb{R}^n$.

Example 2.5.2 *Given an SPD matrix A, the function* $(\mathbf{u}, \mathbf{v})_A = \mathbf{u}^T A\mathbf{v}$ *is an inner product on* $\mathbb{R}^n$ *(verify this). Two non-zero vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *for which* $(\mathbf{u}, \mathbf{v})_A = 0$ *are called A-conjugate.*

A useful relation satisfied by any inner product is the so-called Cauchy-Schwartz inequality

$$|(x, y)|^2 \leq (x, x)\,(y, y)\,. \tag{2.3}$$

2.5.2 Vector Norms

Vector norms define a distance measure in $\mathbb{R}^n$. More precisely, we have the following definition.

Definition 2.5.2 *Denoting $c \in \mathbb{R}$ a scalar and $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ vectors, the vector norm $||.|| : \mathbb{R}^n \to \mathbb{R}$ is a function that satisfies the following properties*

1. **positivity**
$$||\mathbf{u}|| \geq 0 \textit{ with equality only for } \mathbf{u} = \mathbf{0}$$
2. **homogenity**
$$||c\,\mathbf{u}|| = |c|||\mathbf{u}||$$
3. **triangular inequality**
$$||\mathbf{u} + \mathbf{v}|| \leq ||\mathbf{u}|| + ||\mathbf{v}|| \,.$$

Example 2.5.3 *Given an inner product $(.,.)$, the function $\mathbf{u} \to \sqrt{(\mathbf{u},\mathbf{u})}$ defines a norm (verify). Taking in particular the scalar product as inner product, we arrive at the Euclidean norm denoted as $||\mathbf{u}||_2 = \sqrt{\mathbf{u}^T\mathbf{u}}$. Taking the inner product induced by an SPD matrix A, we arrive the A-norm denoted as $||\mathbf{u}||_A = \sqrt{\mathbf{u}^T A\mathbf{u}}$.*

The Euclidean norm is a special instance of a p-norm defined next.

Definition 2.5.3 *Given $1 \leq p < \infty$, the p-norm (Hölder norm) of a vector $\mathbf{u} \in \mathbb{R}^n$ denoted as $||\mathbf{u}||_p$ is defined by*

$$||\mathbf{u}||_p = \left[\sum_{i=1}^{n} |u_i|^p\right]^{1/p} \,. \tag{2.4}$$

We in particular have that

$$\begin{aligned} ||\mathbf{u}||_1 &= |u_1| + \ldots + |u_n| && (2.5)\\ ||\mathbf{u}||_2 &= \left[u_1^2 + \ldots + u_n^2\right]^{1/2} && (2.6)\\ ||\mathbf{u}||_\infty &= \max_{i=1,\ldots,n} |u_i| \,. && (2.7) \end{aligned}$$

An orthogonal transformation $Q \in \mathbb{R}^{n \times n}$ leaves the Euclidean ($p = 2$) norm of a vector invariant, i.e., if $Q^T Q = I$ then $||Q\mathbf{u}||_2 = ||\mathbf{u}||_2$.

2.5.3 Matrix Norms

The analysis of matrix algorithms frequently requires the use of matrix norms. For example, the quality of a linear system solver may be poor if the matrix of coefficients is *nearly singular*. To quantify the notion of near-singularity we need a measure of distance on the space of $m \times n$ matrices. The concepts of a vector norm and of an operator induced norm allow to define a such a metric.

Definition 2.5.4 *Given $1 \leq p < \infty$, the p-norm (Hölder norm) of a matrix $R \in \mathbb{R}^{m\times n}$ denoted as $||R||_p$ is defined by*

$$||R||_p = \sup_{\mathbf{u}\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \frac{||R\mathbf{u}||_p}{||\mathbf{u}||_p} \,. \tag{2.8}$$

The norm $||R||_p$ can be regarded as the maximum *amplification* factor of R.

Proposition 2.5.1 *It can be shown that the above definition is equivalent to*

$$||R||_p = \max_{\mathbf{u}\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \frac{||R\mathbf{u}||_p}{||\mathbf{u}||_p} = \max_{||\mathbf{u}||_p\leq 1} ||R\mathbf{u}||_p = \max_{||\mathbf{u}||_p=1} ||R\mathbf{u}||_p \,, \tag{2.9}$$

and that the matrix p-norm satisfies the usual properties of a norm, i.e.,

$$\begin{aligned} \|R\|_p &\geq 0 & (2.10)\\ \|c\,R\|_p &= |c|\|R\|_p & (2.11)\\ \|R_1 + R_2\|_p &\leq \|R_1\|_p + \|R_2\|_p & (2.12) \end{aligned}$$

as well as the so-called submultiplicative *property, i.e., that for any $R_1 \in \mathbb{R}^{m\times q}$ and $R_2 \in \mathbb{R}^{q\times n}$*

$$\|R_1 R_2\|_p \leq \|R_1\|_p \, \|R_2\|_p \qquad (2.13)$$

The proof of this proposition is left as an exercise.

As an orthogonal transformation Q leaves the Euclidean norm of a vector invariant, we have from the definition that $\|Q\|_2 = 1$. It appears that orthogonal transformations leaves the Euclidean norm of a matrix invariant. More precisely, we have the following proposition.

Proposition 2.5.2 *Given an $m \times n$ matrix R and orthogonal matrices Q and Z of appropriate size, we have that*

$$\|QRZ\|_2 = \|R\|_2 \,. \qquad (2.14)$$

This implies that in case of a symmetric $n \times n$ matrix A that can be diagonalized as $A = Q^T DQ$ that $\|A\|_2 = \|D\|_2$.

Example of a non-p-norm The Frobenius norm of a matrix B is defined as

$$\|R\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |r_{ij}|^2} \,. \qquad (2.15)$$

It can be viewed as the Euclidean norm of the vector obtained from all rows (or colums) of B. It is much easier to compute than the matrix p-norm and therefore very useful in numerical linear algebra.

Computing the 1, 2 and ∞-norm of a matrix In the case of $p = 1$, $p = 2$ and $p = \infty$, the following expressions exist that allow to the matrix p-norm in practice

$$\begin{aligned} \|R\|_1 &= \max_{1\leq j\leq n} \sum_{i=1}^{m} |r_{ij}| \quad \text{maximum absolute column sum} & (2.16)\\ \|R\|_2 &= \max_{1\leq i\leq n} \lambda_k(R^T R) = \lambda_{max}(R^T R) & (2.17)\\ \|R\|_\infty &= \max_{1\leq i\leq m} \sum_{j=1}^{n} |r_{ij}| \,. \quad \text{maximum absolute row sum} & (2.18) \end{aligned}$$

The proof of these expressions if left as an exercise. Expression (2.17) in particular implies that in case of an $n \times n$ matrix A that is SPD, $\|A\|_2 = \lambda_n(A)$.

Let $\mathbf{e}$ be the vector of appropriate size having only 1 as entry, that is $\mathbf{e} = (1, 1, \ldots, 1)^T$. Then the condition $|R|\mathbf{e} \leq C\mathbf{e}$ for some constant C states that all absolute row-sums of R are bounded by C. In this case, the maximum absolute row sum is bounded by C as well, i.e., $\|R\|_1 \leq C$, i.e., we have proven the following proposition

Proposition 2.5.3

$$|R|\mathbf{e} \leq C\mathbf{e} \Rightarrow \|R\|_1 \leq C \qquad (2.19)$$

2.6 Condition Number

Given the linear system $A\mathbf{u} = \mathbf{f}$, a small perturbation in the right-hand side vector $\mathbf{f} \rightarrow \mathbf{f} + \triangle\mathbf{f}$ will cause a (not necessarily small) perturbation in the solution $\mathbf{u} \rightarrow \mathbf{u} + \triangle\mathbf{u}$. We will see later that the the spectral condition number of the system matrix A denoted as $\kappa_p(A)$ allows to bound the magnitude of the perturbation $\triangle\mathbf{u}$ in terms of the magnitude in the perturbation $\triangle\mathbf{f}$.

Definition 2.6.1 *The spectral condition number measured in p-norm $\kappa_p(A)$ of an $n \times n$ matrix A is defined as*

$$\kappa_p(A) = \|A\|_p \|A^{-1}\|_p \,. \tag{2.20}$$

Observe that for any p, $\kappa_p(A) \in [1, \infty)$ (why?). Using the relation (2.17), the condition number in 2-norm can be expressed as

$$\kappa_2(A) = \frac{\sqrt{\lambda_{max}(A^T A)}}{\sqrt{\lambda_{min}(A^T A)}} \,. \tag{2.21}$$

In case that A is symmetric, the above expression reduces to

$$\kappa_2(A) = \frac{|\lambda_{max}(A)|}{|\lambda_{min}(A)|} \,, \tag{2.22}$$

and in case that A is SPD to

$$\kappa_2(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)} \,. \tag{2.23}$$

2.7 Spectral Radius

The spectral radius will be used in the study of the convergence of stationary iterative methods.

Definition 2.7.1 *The spectral radius $\rho(A)$ of a matrix $A \in \mathbb{R}^{n \times n}$ is defined as*

$$\rho(A) = \max_{i=1,\ldots,n} \{|\lambda_i| : \lambda_i \in \sigma(A)\} \,. \tag{2.24}$$

Note that in general $\rho(A) \notin \sigma(A)$. The computation of the spectral radius is typically not straightforward. The following theorem gives on upper bound on the spectral matrix that using e.g. (2.16), (2.17) or (2.18) is easier to compute.

Theorem 2.7.1 *Given $\|\cdot\|$ any submultiplicative matrix norm, then*

$$\rho(A) \leq \|A\| \,. \tag{2.25}$$

Proof. Assume $(\lambda, \mathbf{u})$ any eigenvalue-eigenvector pair of A. Then $A\mathbf{u} = \lambda\mathbf{u}$, and thus by virtue of the submultiplicative property

$$\|\lambda\mathbf{u}\| = |\lambda|\|\mathbf{u}\| = \|A\mathbf{u}\| \leq \|A\|\|\mathbf{u}\| \Rightarrow |\lambda| \leq \|A\| \,.$$

The result then follows from the fact that λ was chosen arbitrarily.

Power of a Matrix The following important theorem links the the spectral radius with the kth power of a matrix A denoted by A^k for $k \rightarrow \infty$.

Theorem 2.7.2

$$\rho(A) < 1 \Leftrightarrow \lim_{k \rightarrow \infty} A^k = \emptyset_{n \times n} \tag{2.26}$$

Proof. $\boxed{\Rightarrow}$ For the proof we assume A to be diagonalizable, i.e., we assume that a matrix P and a diagonal matrix D exist such that $A = PDP^{-1}$. The columns of P span the eigenspaces of A and a diagonal of D are the eigenvalues of A. (In case that A is not diagonalizable an argument similar to the one that follows can be made using the Jordan decomposition of A). This implies that $A^k = PD^kP^{-1}$, which in turn implies that if $\rho(A) < 1$ we have that $\lim_{k\to\infty} A^k = \emptyset_{n\times n}$.
$\boxed{\Leftarrow}$ Assume $\mathbf{u}_n$ to be a unit eigenvector corresponding to the eigenvalue with maximum modulus, i.e., $A\mathbf{u}_n = \lambda_n \mathbf{u}_n$ or

$$A^k\mathbf{u}_n = \lambda_n^k \mathbf{u}_n \tag{2.27}$$

which implies, by taking the 2-norm on both sides,

$$|\lambda_n|^k = \|A^k\mathbf{u}_n\|_2 \to 0\,. \tag{2.28}$$

This shows that $\rho(A) < 1$. □
The following results allows to quantify the *speed* of convergence of the limiting process in Theorem 2.7.2.

Theorem 2.7.3 (Gelfand's formula) *For any $\|\cdot\|$ matrix norm, we have that*

$$\lim_{k\to\infty} \sqrt[k]{\|A^k\|} = \rho(A)\,. \tag{2.29}$$

This shows that the spectral radius of A gives the asymptotic growth rate of the norm of A^k. The proof of this result can be found in literature.

Power Series of a Matrix For scalar arguments x we have that the power series

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \text{ converges if } |x| < 1\,. \tag{2.30}$$

The next theorem generalizes this result to $n \times n$ matrices.

Theorem 2.7.4

$$\rho(A) < 1 \Leftrightarrow (I - A) \textit{ is non-singular, and } \sum_{k=0}^{\infty} A^k = (I-A)^{-1} \tag{2.31}$$

Proof. $\boxed{\Rightarrow}$ If $\rho(A) < 1$, then $I - A$ has eigenvalues bounded away from zero, and is therefore non-singular. We furthermore have the equality

$$(I - A^{k+1}) = (I-A)(I + A + A^2 + \ldots + A^k)\,, \tag{2.32}$$

or equivalently

$$(I-A)^{-1}(I - A^{k+1}) = (I + A + A^2 + \ldots + A^k)\,. \tag{2.33}$$

Taking the limit as $k \to \infty$ and taking Theorem 2.7.2 into account then yields the desired result.
$\boxed{\Leftarrow}$ A necessary condition for the power series $\sum_{k=0}^{\infty} A^k$ to converge is that $\lim_{k\to\infty} \|A^k\|_p = 0$. This implies that $\lim_{k\to\infty} A^k = 0$ and again by Theorem 2.7.2 the desired result. □

2.8 Irreducibility and Diagonal Dominance

In case that diagonal entries in A are large compared to off-diagonal entries, useful estimates for the location of the eigenvalues of A can be obtained from Gershgorin's theorem. To formalize this idea we first introduce the concept of irreducibility and denote by P an $n \times n$ permutation matrix such that PA (AP) corresponds to a permutation of the rows (columns) of A. The matrix A is called reducible if a permutation matrix P exists such that PAP^T is block upper triangular. If in particular A is reducible and symmetric, then PAP^T is block diagonal, implying that A after permutation is made up of not-interconnected blocks. This motivates the following definition.

Definition 2.8.1 *The matrix A is called* irreducible *iff no permutation matrix P exists such that PAP^T is block upper triangular.*

Next we define the concept of diagonal dominance.

Definition 2.8.2 *The matrix A is called* row diagonal dominant *iff*

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad i = 1, \ldots, n \tag{2.34}$$

with strict inequality from at least one i.

In the following definition stronger requirements are imposed.

Definition 2.8.3 *The matrix A is called* row strictly diagonal dominant *iff*

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad i = 1, \ldots, n\,. \tag{2.35}$$

The concepts of *column (strictly) diagonal dominance* and *column strictly diagonal dominance* can be defined in a similar fashion. Note that in the above definitions the sign of the off-diagonal entries is irrelevant. The concept of irreducibility is linked to that of diagonal dominance in the following definition.

Definition 2.8.4 *The matrix A is called* irreducibly strictly diagonal dominant *iff A is irreducible and diagonally dominant.*

Diagonal dominance of A and its spectrum can be related through the following theorem that we state here (for matrices having possibly complex eigenvalues).

Theorem 2.8.1 (Gershgorin) *If $\lambda \in \sigma(A)$, then λ is located in one of the n closed disks in the complex plane that centered in A_{ii} and have as radius*

$$\rho_i = \sum_{j=1, j\neq i}^{n} |a_{ij}| \tag{2.36}$$

i.e.,

$$\lambda \in \sigma(A) \Rightarrow \exists i \textit{ such that } |a_{ii} - \lambda| \leq \rho_i = \sum_{j=1, j\neq i}^{n} |a_{ij}|\,. \tag{2.37}$$

This theorem is a powerful tool as it allows to certify that symmetric irreducibly diagonal dominant matrices are SPD without computing its eigenvalues explicitly. Details can be found in [52].

2.9 Positive Inverse

In the following definition the positivity of the inverse is the key feature.

Definition 2.9.1 *The matrix A is an M-matrix iff it satisfies the following four properties*

1. $a_{ii} > 0$ *for* $i = 1, \ldots, n$;
2. $a_{ij} \leq 0$ *for* $i \neq j$, $i, j = 1, \ldots, n$;
3. *A is non-singular;*
4. $A^{-1} \geq 0$.

In the context of the discretization of partial differential equations, the M-matrix property guarantees useful properties such as the absence of wiggles in convection-dominated flows and the positivity of concentrations in chemical engineering applications. In the context of iterative solution methods, the M-matrix property guarantees the convergence of the basic schemes. While the M-matrix property is therefore highly desirable, verifying whether or not a matrix satisfies the fourth of the above conditions is hard (if not impossible at all) in practice. Therefore the following result will have a large practical value.

Theorem 2.9.1 *If the matrix A satisfies the following three properties*

1. *$a_{ii} > 0$ for $i = 1, \dots, n$;*
2. *$a_{ij} \leq 0$ for $i \neq j$, $i, j = 1, \dots, n$;*
3. *A is irreducibly diagonally dominant*

then A is an M-matrix.

Note the absence of any condition on A^{-1} in the above conditions.

2.10 Perron-Frobenius Theorem

Converge results for stationary iterative methods critically hinge on the theorem of *Perron-Frobenius* (named after Oskar Perron (1880 - 1975) and Ferdinand Georg Frobenius (1849 - 1917)). Carl Meyer is quoted for saying that *In addition to saying something useful, the Perron-Frobenius theory is elegant. It is a testament to the fact that beautiful mathematics eventually tends to be useful, and useful mathematics eventually tends to be beautiful.*

Theorem 2.10.1 *(***Perron-Frobenius***) Let A be a real $n \times n$ nonnegative irreducible matrix. Then $\lambda = \rho(A)$, the spectral radius of A, is a simple eigenvalue of A. Moreover, there exists an eigenvector $\mathbf{u}$ with positive elements associated with this eigenvalue.*

We use this theorem to prove the following

Theorem 2.10.2 *Let A be a real $n \times n$ nonnegative irreducible matrix. Then*

$$\rho(A) < 1 \Leftrightarrow (I - A) \text{ is non-singular, and } (I - A)^{-1} \geq 0 \tag{2.38}$$

Proof. $\boxed{\Rightarrow}$ From Theorem 2.7.4 follows that if $\rho(A) < 1$ then $I - A$ is non-singular and that $\sum_{k=0}^{\infty} A^k = (I - A)^{-1}$. The fact that the product and sum of positive matrices is again positive then implies that $I - A$ is non-negative.
$\boxed{\Leftarrow}$ By Theorem 2.10.1, a nonnegative vector $\mathbf{u}$ exists such that $A\mathbf{u} = \rho(A)\mathbf{u}$ or that

$$(I - A)\mathbf{u} = [1 - \rho(A)]\mathbf{u} \Leftrightarrow \frac{1}{1 - \rho(A)}\mathbf{u} = (I - A)^{-1}\mathbf{u}\,. \tag{2.39}$$

As both $(I - A)^{-1}$ and $\mathbf{u}$ are non-negative and $(I - A)$ is non-singular, we have that $1 - \rho(A) > 0$. □

2.11 A collection of Matrices

In this section we introduce a collection of matrices that we will use as example in the remainder of these notes.

Example 2.11.1 *The* **Hilbert matrix** *$H \in \mathbb{R}^{n\times}$ is a square with unit fractions*

$$h_{ij} = \frac{1}{i + j - 1}\,. \tag{2.40}$$

Clearly H is symmetric. It can be shown that H is positive definite.

Example 2.11.2 *The* **Pei matrix** $P \in \mathbb{R}^{n\times}$ *is the square matrix than for* $\alpha \in \mathbb{R}$ *can be defined through its entries as*

$$p_{ij} = \begin{cases} \alpha \textit{ if } i = j \\ 1 \textit{ if } i \neq j \end{cases} \tag{2.41}$$

Example 2.11.3 *The* **Van der Monde matrix** $M \in \mathbb{R}^{n\times}$ *is for a given vector* $\mathbf{v} \in \mathbb{R}^n$ *can be defined through its entries as*

$$m_{ij} = v_i^{n-j} \, . \tag{2.42}$$

If $\mathbf{v}$ *holds the* n *complex roots of* 1*, then* M *represents the discrete Fourier transform.*

Example 2.11.4 *The* **graph Laplacian** *and* **weighted graph Laplacian** *are constructed on a graph. We therefore consider an oriented graph* $\mathcal{G} = \mathcal{G}(V, E)$ *with* n *vertices* $v \in V$ *and* m *edges* $e \in E$*. We consider the* $n \times m$ *integer matrix* G *to be the node-edge incidence matrix. Each column of* G *corresponds to an edge of* $\mathcal{G}$ *and has an entry equal to* $+1$ *in the row corresponding to the node in which the edge starts and an entry* -1 *in the row corresponding to the node in which the edge ends. All the other entries in the column are zero. The graph Laplacian is the* $n \times n$ *matrix*

$$L = G\, G^T \, . \tag{2.43}$$

Given an $m \times m$ *diagonal matrix* W *that attributes a strictly positive weight* w_j *to each edge* e_j *(maximum flow capacity for instance), the weighted graph Laplacian is the* $n \times n$ *matrix*

$$L_W = G\, W\, G^T \, . \tag{2.44}$$

As the column sum of G *equals zero,* G^T *times the vector having all* 1 *as components is zero. The constant vector thus lies in the null space of* L *and* L_W*. This implies that* L *and* L_W *are singular. We will therefore sometimes consider the submatrix obtained of* L *by deleting the first row and column of* L *and* L_W*. This submatrix is referred to as the first principal submatrix of* L *(or* L_W*) and is non-singular.*

As special case we consider the weighted graph Laplacian for the simple graph with 5 *nodes connected by* 4 *edges and unevenly weighted edges, i.e., the graph*

$$\begin{aligned} V &= \{1, 2, 3, 4, 5\} \\ E &= \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}\} \\ W &= \textit{diag}(1, 1, 10^4, 10^4) \, . \end{aligned}$$

2.12 Exercises

Exercise 2.12.1 Calculate the Euclidean inner products of the following vectors

a) $(1, -7, 12, 2)^T$, $(3, 2, 0, 5)^T$
b) $(\sin x, \cos x)^T$, $(\cos x, -\sin x)^T$.

Exercise 2.12.2 Calculate the Euclidean norm of the following vectors

a) $(-1, 5, 2)^T$
b) $(\sin x, \cos x)^T$.

Exercise 2.12.3 Calculate the uniform (infinity) norm of the vectors in Exercises 2.12.1, 2.12.2. In 2.12.2 b), determine the points x, $0 \leq x < 2\pi$, for which the vector $(\sin x, \cos x)^T$ has the largest infinity norm.

Exercise 2.12.4 Two vectors u_h and v_h are said to be orthogonal if $\langle u_h, v_h \rangle = 0$. In the following, take $\langle \cdot, \cdot \rangle$ to be the Euclidean inner product.

a) Find a vector which is orthogonal to $(a, b)^T$.
b) Use a) to find a vector which is orthogonal to $(\sin x, \cos x)^T$. Sketch the two vectors for $x = \pi/4$.

Exercise 2.12.5 Show that $\sup\limits_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \max\limits_{\|x\|_p = 1} \|Ax\|_p$ for $p \geq 1$.

Exercise 2.12.6 Show that $\|A\|_1 = \max\limits_{1 \leq j \leq n} \sum\limits_{i=1}^{m} |a_{ij}| \quad A \in \mathbb{R}^{m \times n}$ (the maximal absolute column sum).

Exercise 2.12.7 Show that $\|A\|_2 = \sqrt{\lambda_{max}(A^T A)}$.

Exercise 2.12.8 Let the matrix A be given by

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}.$$

$\|v\|_E = (\langle Av, v \rangle_2)^{1/2}$ is called the energy norm of v with respect to A.

a) Calculate the energy norm (with respect to A) of the vector $(v_1, v_2, v_3, v_4)^T$.
b) Show that the energy norm calculated in a) is positive, unless v is the zero vector.

Exercise 2.12.9 The matrix norm induced by a vector norm $\|\cdot\|$ is defined by

$$\|A\| = \sup_{v \neq 0} \frac{\|Av\|}{\|v\|} = \sup_{\|v\|=1} \|Av\|.$$

The spectral norm $\|A\|_S$ of a symmetric positive definite matrix A is given by

$$\|A\|_S = \max_{\lambda \text{ eigenvalue of } A} \lambda.$$

One can show that the matrix norm induced by the Euclidean norm is the spectral norm (see Exercise 5.9.1).
This exercise shows that the above statement is true for the matrix A given by

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}.$$

a) Show that A is symmetric, positive definite.
b) Compute the spectral norm of A by determining its eigenvalues.

c) Determine the norm of A induced by the Euclidean norm by using the method of Lagrange multipliers to calculate $\sup_{\|v\|=1} \|Av\|$. Compare your answer with the largest eigenvalue from b).

Exercise 2.12.10 What is the condition number of the identity matrix I in any p-norm?

Exercise 2.12.11 *Show the following properties of symmetric positive definite matrices:*

1. *A* principal submatrix *of a matrix $A \in \mathbb{R}^{n\times n}$ is obtained from A by deleting from A rows and columns corresponding to the same index are removed. A* principal minor *is the determinant of a principal submatrix. A* leading principal submatrix *of A consists of entries of A taken from the first k rows and columns of A, where $1 \leq k \leq n$. A* leading principal minor *is the determinant of a leading principle submatrix. Show that a symmetric matrix A is positive definite iff all its leading principal minors are positive.*

2. *If $A = (a_{ij})$ is symmetric positive definite, then $a_{ii} > 0$ for all i.*

3. *If $A = (a_{ij})$ is symmetric positive definite, then the largest element (in magnitude) of the whole matrix must lie on the diagonal.*

4. *The sum of two symmetric positive definite matrices is symmetric positive definite.*

Exercise 2.12.12 *Show the following properties on the condition number of a matrix [10]*

1. *$\kappa_p(A) \geq 1$ for any p-norm;*

2. *$\kappa(\alpha\, A) = \kappa(A)$, where α is any non-zero scalar, for any given norm;*

3. *$\kappa_p(A) = 1$ iff A is a non-zero scalar multiple of an orthogonal matrix, i.e., $A^T A = \alpha I$, where $\alpha \neq 0$. (Note that this property of an orthogonal matrix A makes the matrix very attractive for its use in numerical computations);*

4. *$\kappa_2(A^T A) = [\kappa_2(A)]^2$;*

5. *$\kappa_2(A) = \kappa_2(A^T)$; $\kappa_1(A) = \kappa_\infty(A^T)$;*

6. *for any given norm $\kappa(AB) \leq \kappa(A)\kappa(B)$ if the matrices A and B are compatible for matrix multiplication.*

Exercise 2.12.13 *In this exercise we compare the determinant and the condition number of a matrix A as a measure for the near-singularity of A.*

1. *consider the matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ with $a_{ii} = 1$ and $a_{ij} = -1$ if $j > 1$. Perform the following numerical experiment to verify that A is nearly singular for large n. Consider the linear system $A\mathbf{u} = \mathbf{f}$ to be such that the vector $\mathbf{u}$ with all components equal to 1 is its solution. This can easily be accomplished by, for a given values of n, setting the right-hand side vector $\mathbf{f}$ equal to $\mathbf{f} = A\mathbf{u}$ (by a matrix-vector multiplication). Given the right-hand side vector $\mathbf{f}$ and the system matrix A, solve the linear system for $\mathbf{u}$ for various values of n (using the backslash direct solver $\mathbf{u} = A\backslash\mathbf{f}$ in Matlab for instance). Let the computed solution be denoted by $\widehat{\mathbf{u}}$. List for various values of n the relative error $\|\mathbf{u} - \widehat{\mathbf{u}}\|_2/\|\mathbf{u}\|_2$ and the relative residual norm $\|\mathbf{f} - A\,\widehat{\mathbf{u}}\|_2/\|\mathbf{f}\|_2$ in the computed solution. You may use to this end the following Table 2.1.*

2. *consider again the matrix $A = (a_{ij})$ with $a_{ii} = 1$ and $a_{ij} = -1$ if $j > 1$. Show that A has determinant equal to 1. Show that A has condition number $\kappa_\infty(A) = n\,2^{n-1}$ by computing $\|A\|_\infty$, A^{-1} (using Gaussian elimination for instance) and $\|A^{-1}\|_\infty$. Does the determinant explain the effect observed in part (a) of the exercise? Does the condition number explain the observed effect in part (a) of the exercise?*

3. *repeat part (1) and part (2) of the exercise using as matrix the diagonal matrix of order n of the form $A = diag(0.1, 0.1, \ldots, 0.1) \in \mathbb{R}^{n\times n}$. You may use the 2-norm to measure the conditioning of the matrix.*

problem size n	relative error $\|\mathbf{u} - \widehat{\mathbf{u}}\|_2 / \|\mathbf{u}\|_2$	relative residual $\|\mathbf{f} - A\,\widehat{\mathbf{u}}\|_2 / \|\mathbf{f}\|_2$
10		
20		
30		
40		
50		

Table 2.1: Numerical study of matrix conditioning.

Exercise 2.12.14 *Demonstrate Theorem 2.2.1.*

Exercise 2.12.15 *Draw the unit sphere for the 1, 2 and ∞ norm in $\mathbb{R}^2$.*

Exercise 2.12.16 *Show that the 1, 2 and ∞ matrix norms are equivalent, i.e, show that for any $\alpha, \beta \in \{1, 2, \infty\}$ and for any $A \in \mathbb{R}^{n \times n}$ numbers r and s exist such that*

$$r\,\|A\|_\alpha \leq \|A\|_\beta \leq s\,\|A\|_\alpha\,.$$

Exercise 2.12.17 *Show that for any $A \in \mathbb{R}^{n \times n}$ holds that*

$$\|A\|_2 \leq \sqrt{\|A\|_1\,\|A\|_\infty}\,.$$

Exercise 2.12.18 *Proof Gelfand's formula and show it experimentally.*

Exercise 2.12.19 *Proof Gershgorin theorem and show it experimentally.*

Exercise 2.12.20 *Demonstrate the Perron-Frobenius theorem stated in Theorem 2.10.1.*

Chapter 3

The Model Problem and Its Finite Difference Discretization

`Many people drive, we provide the engine`

3.1 Introduction

We start this chapter by giving motivating examples from engineering practice and by classifying linear second order partial differential equations with constants coefficients. We subsequently define a sequence of elliptic problems of increasing complexity and introduce the second order finite difference discretization of the model problem on an uniform mesh. The properties of the coefficient matrix of the resulting linear systems will be discussed in detail. Finally we discuss which of these properties carry over to problems encountered in engineering practice. The linear systems introduced in this chapter will serve as example for the linear solvers discussed in subsequent chapters.

Study goals The study goals we aim at in this chapter are

- classify second order partial differential equations;
- define elliptic problems;
- discretize elliptic model problems using low order finite difference methods;
- name and demonstrate properties of the coefficient matrices of the resulting linear systems;
- argue which of the linear system properties carry over to more realistic problems.

3.2 Motivating Examples

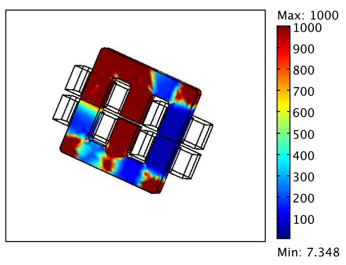
In this section we give two motivating examples, the first from computational electromagnetics and the second from computational fluid dynamics.

Fault Current Limiter Fault current limiters are expected to play an important role in the protection of future power grids. They are capable of preventing fault currents from reaching too high levels and, therefore enhance the life time expectancy all power system components. Figure 3.1 shows two examples of fault-current limiters along with some finite element simulation results.

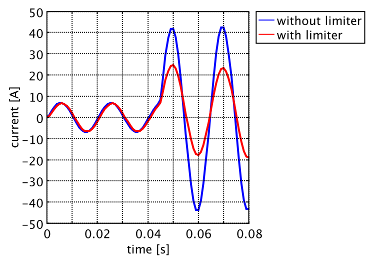
Industrial Furnace The simulation of flows through an enclosed surface is an often reoccuring problem in computational fluid dynamics. In Figure 3.2 a study case of the simulation of the flow and temperature profile inside an industrial furnace is presented.

(a) Open core configuration

(b) Three legged configuration



(c) Diffusion coefficient ν



(d) Current with and without limiter

Figure 3.1: Numerical simulation of fault current limiters.

3.3 Classification of Second Order Partial Differential Equations

In this section we classify second order linear partial differential equations (PDEs) with constant coefficients according to their elliptic, parabolic and hyperbolic nature and give examples of PDEs in each of these three classes.

Classification We consider an open two-dimensional domain $(x, y) \in \Omega \subset \mathbb{R}^2$ with boundary $\Gamma = \partial\Omega$ as the domain of the second order linear partial differential equation (PDE) for the unknown field $u = u(x, y)$ and the source function $f(x, y)$ that can be written as

$$\mathcal{L}(u) = f \text{ on } \Omega \tag{3.1}$$

where the operator $\mathcal{L}$ has constant coefficients a_{ij}, b_i and c

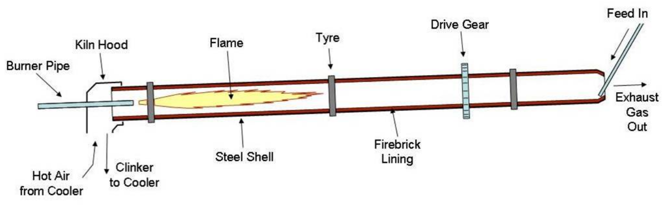
$$\mathcal{L}(u) = a_{11}\frac{\partial^2 u(x,y)}{\partial x^2} + 2a_{12}\frac{\partial^2 u(x,y)}{\partial x \partial y} + a_{22}\frac{\partial^2 u(x,y)}{\partial y^2} + b_1\frac{\partial u(x,y)}{\partial x} + b_2\frac{\partial u(x,y)}{\partial y} + c\,u(x,y)\,. \tag{3.2}$$

(Only in this section the coefficients a_{ij} denote functions in x and y.) We classify these equation based on the sign of the determinant

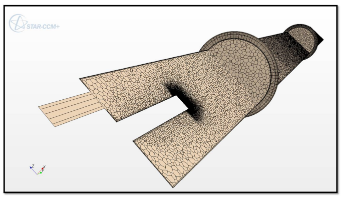
$$D = \begin{vmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}^2\,. \tag{3.3}$$

The differential operator $\mathcal{L}$ is called

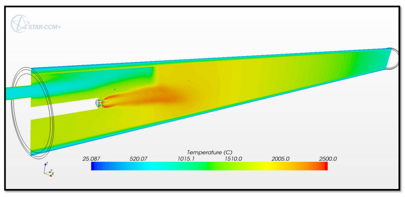
- *elliptic* if $D > 0$
- *parabolic* if $D = 0$
- *hyperbolic* if $D < 0$



(a) Schematic represention of an industrial furnace



(b) Mesh employed



(c) Simulated temperature profile.

Figure 3.2: Numerical simulation of industrial furnaces.

Prototypes in this classification are the elliptic Laplace equation $u_{xx} + u_{yy} = 0$, parabolic heat equation $u_{xx} - u_y = 0$ and the hyperbolic wave equation $u_{xx} - u_{yy} = 0$. In the case of xy-varying coefficients a_{11}, a_{22} and a_{12}, the sign of D and therefore the type of the PDE may change with the location in Ω.

As linear system solvers for discretized hyperbolic PDEs are far less developed, we will only consider elliptic and parabolic equations in this course. Elliptic equations are characterized by the fact that changes in the data imposed on the boundary Γ are felt instantaneously in the interior of Ω. Parabolic equations model the evolution of a system from initial to an stationary (time-independent) equilibrium state.

Examples of Elliptic Partial Differential Equations We will consider the Poisson equation

$$- \triangle u := -\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} = f \tag{3.4}$$

as it plays a central role in engineering applications in which the field u solved for plays the role of a (electric, magnetic or gravitational) potential, temperature or displacement. The reason for the minus sign in front of the Laplacian will be discussed together with the finite difference discretization. We will also consider two variants of (3.4). In the first variant we introduce a small positive parameter $0 < \epsilon \ll 1$ to arrive at the anisotropic variant

$$-\epsilon \frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} = f \tag{3.5}$$

that can be thought of as a simplified model to study the effect of local mesh refinement required to capture e.g. boundary layers or small geometrical details (such as for instance the air-gap in an electrical machine). This model will be used in the study of smoothers in a multigrid context. In the second variant we again incorporate more physical realism in (3.4) by allowing the parameter c in

$$-\frac{\partial}{\partial x}\left(c \frac{\partial u(x,y)}{\partial x}\right) - \frac{\partial}{\partial y}\left(c \frac{\partial u(x,y)}{\partial y}\right) = f \,, \tag{3.6}$$

to have a jump-discontinuity across the boundary of two subdomains (such as for instance the jump in electrical conductivity across the interface between copper and air). This model will be used both in the study of smoothers in a multigrid context as of Krylov subspace methods.

The Helmholtz and the convection-diffusion equation are other examples of elliptic partial differential equations. The former can be written as

$$- \triangle u - k^2 u = -\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} - k^2 u = f \,, \tag{3.7}$$

and models the propagation of an acoustical or electromagnetic wave with wave number k. The latter is a simplified model for the study of a flow with (dimensionless) velocity $\mathbf{v} = (1, 0)$ around a body and can be written as

$$-\epsilon \triangle u + \mathbf{v} \cdot \nabla u = -\epsilon \frac{\partial^2 u(x,y)}{\partial x^2} - \epsilon \frac{\partial^2 u(x,y)}{\partial y^2} + \frac{\partial u(x,y)}{\partial x} = f \,, \tag{3.8}$$

where ϵ is the inverse of the Peclet number (which plays the same role as the Reynolds number in the Navier-Stokes equations). The Helmholtz and the convection-diffusion equation will motivate the introduction of Krylov subspace methods for symmetric indefinite and non-symmetric matrices, respectively. The verification that the equations introduced above are indeed elliptic is left as an exercise.

The classification given above can be extended to systems of coupled partial differential equations. The vector-valued double curl equation for example relates the vector potential $\mathbf{U}(x,y,z) = (U_1(x,y,z), U_2(x,y,z), U_3(x,y,z))$ with the excitation vector field $\mathbf{F}(x,y,z)$ and can be written as

$$\nabla \times (\nu \nabla \times \mathbf{U}) = \mathbf{F}(x,y,z) \,, \tag{3.9}$$

where as in (3.6) ν represents a material parameter. This model can be derived from the Maxwell equations and plays a central role in the finite element modeling of electrical machines, wind turbines and power transformers. In this setting $\mathbf{F}$ and $\mathbf{U}$ play the role of the applied current density and the magnetic vector potential, respectively.

In order to guarantee uniqueness of the solution of the above elliptic partial differential equations, appropriate (Dirichlet, Neumann or other) boundary conditions on $\Gamma = \partial\Omega$ need to be supplied.

Example of a Parabolic Partial Differential Equation In order to give examples of parabolic partial differential equations, we change the notation y into t and consider Ω to be a rectangle $(x,t) \in \Omega = [0, L] \times [0, T]$. The parabolic differential equation with source term $f(x,t)$

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t) \tag{3.10}$$

models the diffusion of the quantity $u(x,t)$ in time that reaches a time-independent equilibrium state when $\frac{\partial u(x,t)}{\partial t}$ is zero. All of the elliptic models introduced above can be extended to a parabolic model by adding the allowing u to become time-dependent, that is $u = u(x,y,t)$, and adding the term $\frac{\partial u(x,t)}{\partial t}$. These models will be considered for instance when discussing the role of initial vectors of iterative solution methods.

In order to guarantee uniqueness of the solution of parabolic partial differential equations, both boundary and initial conditions need to be supplied.

3.4 The Model Problem

Problem Definition In the remainder of this chapter we consider the following one- and two-dimensional Poisson (elliptic) model problems (i.e., the partial differential equation supplied with boundary conditions):

- given the domain $\Omega = (0,1)$ with boundary $\Gamma = \partial\Omega$ and outward normal $\mathbf{n}$, and given the boundary data $b(x)$ or $c(x)$, solve for $u(x)$ the following ordinary differential equation

  $$-\frac{d^2 u(x)}{dx^2} = f(x) \text{ on } \Omega \tag{3.11}$$

  supplied with either Dirichlet boundary conditions

  $$u(x) = b(x) \text{ on } \Gamma \tag{3.12}$$

  or Neumann boundary conditions

  $$\frac{\partial u(x)}{\partial n} = \nabla u(x) \cdot \mathbf{n} = c(x) \text{ on } \Gamma \,. \tag{3.13}$$

  In case that $b(x) = 0$ in (3.12) ($c(x) = 0$ in (3.13)) the Dirichlet (Neumann) boundary conditions is called homogeneous.

- given the domain $\Omega = (0,1) \times (0,1)$ with boundary $\Gamma = \partial\Omega$ and outward normal $\mathbf{n}$, and given the boundary data $b(x,y)$ or $c(x,y)$, solve for $u(x,y)$ the following partial differential equation

$$- \triangle\, u := -\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y) \tag{3.14}$$

  supplied with either Dirichlet boundary conditions

$$u(x,y) = b(x,y) \text{ on } \Gamma \tag{3.15}$$

  or Neumann boundary conditions

$$\frac{\partial u(x,y)}{\partial n} = \nabla u(x,y) \cdot \mathbf{n} = c(x,y) \text{ on } \Gamma\,. \tag{3.16}$$

The extension of the concepts introduced in this chapter to problems with Robin, period or any combination of these boundary conditions is left as an exercise.

Continuous Spectral Properties In analyzing the convergence of iterative solution methods in subsequent chapters, we will make extensive use of spectral properties of the discrete Laplace matrices. To derive these properties we resort to the continuous counterpart and consider the related Sturm-Liouville problem. This means that in case of the one-dimensional problem we look for the eigenvalues λ and eigenvectors $u \neq 0$, such that

$$-\frac{d^2 u(x)}{dx^2} = \lambda\, u(x) \tag{3.17}$$

supplied with *homogeneous* Dirichlet or Neumann boundary conditions. *Why homogeneous boundary conditions?* In case of the two-dimensional problem the PDE becomes

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} = \lambda\, u(x,y) \tag{3.18}$$

To solve the one-dimensional problem, we make use of the characteristic equation $r^2 + \lambda = 0$. To solve the two-dimensional problem, we employ separation of variables and proceed subsequently as in the one-dimensional case. The results can be summarized as follows:

- the one-dimensional eigenvalues/eigenvectors problem (3.17) has the following solutions
  - in case of Dirichlet boundary conditions $u(x=0) = 0 = u(x=1)$

$$u^{[k]}(x) = \sin(k\pi x) \text{ corresponding to } \lambda_k = k^2\,\pi^2 \text{ for } k \in \mathbb{N}, k \neq 0 \tag{3.19}$$

    (imposing $k \neq 0$ is required to assure a non-trivial eigenfunction)
  - in case of Neumann boundary conditions $\frac{du}{dx}(x=0) = 0 = \frac{du}{dx}(x=1)$

$$u^{[k]}(x) = \cos(k\pi x) \text{ corresponding to } \lambda_k = k^2\,\pi^2 \text{ for } k \in \mathbb{N} \tag{3.20}$$

    ($k = 0$ gives the constant eigenvector)
  - in the mixed case $u(x=0) = 0$ and $\frac{du}{dx}(x=1) = 0$

$$u^{[k]}(x) = \sin(k\pi x) \text{ corresponding to } \lambda_k = k^2\frac{\pi^2}{4} \text{ for } k \in \mathbb{N}, k \neq 0 \tag{3.21}$$

- the two-dimensional problem (3.18) has the following solutions
  - in case of Dirichlet boundary conditions $u(x,y) = 0$ on Γ

$$u^{[k\ell]}(x,y) = \sin(k\pi x)\sin(\ell\pi y) \text{ corresponding to } \lambda_{k\ell} = \pi^2(k^2+\ell^2) \text{ for } k,\ell \in \mathbb{N}, k \neq 0 \neq \ell \tag{3.22}$$

– in case of Neumann boundary conditions $\frac{\partial u(x,y)}{\partial n} = 0$ on Γ

$$u^{[k\ell]}(x,y) = \cos(k\pi x)\cos(\ell\pi y) \text{ corresponding to } \lambda_{k\ell} = \pi^2(k^2+\ell^2) \text{ for } k,\ell \in \mathbb{N} \quad (3.23)$$

The fact that in the case of Neumann boundary conditions the differential operator has zero as eigenvalue corresponds to the fact that the solution is determined up to a constant. The consideration of other boundary conditions is again left as an exercise. The eigenfunctions can be normalized to form an orthonormal set of eigenfunctions for an infinite dimensional function space.

3.5 Finite Difference Discretization

Discretization of the Geometry For the discretization of the two-dimensional model problem we introduce a mesh size $h = \frac{1}{N}$ (N being the number of mesh elements) and an uniform grid G_h consisting of $N+1$ nodes (for the 1D problem) and consisting of $(N+1)^2$ nodes (for the 2D problem) including those on the boundary Γ

$$G_h = \{(x_i, y_j) | x_i = ih, y_j = jh; h = \frac{1}{N},\ 0 \le i,j \le N; N \in \mathbb{N}\}\,. \quad (3.24)$$

In this numbering the indices $i = 0$ and $i = N$ ($j = 0$ and $j = N$) correspond to grid points on the left and right (bottom and top) boundary, respectively, as in Figure 3.5.

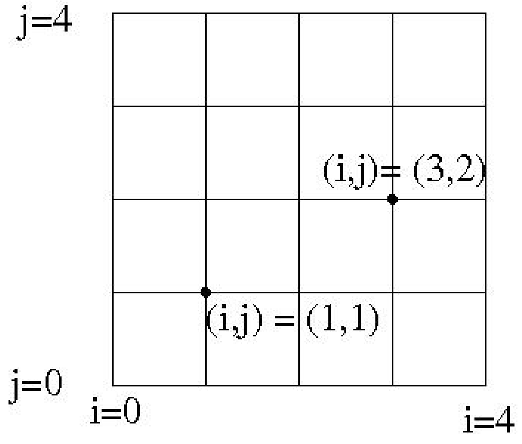


Figure 3.3: Uniform mesh on the square with $N = 4$.

Discretization of the Physics On G_h we introduce grid vectors approximating the source function $f(x,y)$, boundary data $b(x,y)$ and $c(x,y)$ and unknown $u(x,y)$ in the grid nodes with increasing accuracy as $h \to 0$, i.e.,

$$f^h_{i,j} \approx f(x_i, y_j) \text{ for } (x_i, y_j) \in G_h\,, \quad (3.25)$$

and similarly for $b^h_{i,j}$, $c^h_{i,j}$ and $u^h_{i,j}$.

Internal Nodes We enforce the model problem to hold in each grid node and approximate the continuous second order derivatives by central finite difference approximations. The use of finite difference here is generic, as low-order finite element or finite volume discretization result in the same linear system. Using nearest neighbours we have that for the internal nodes that

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \frac{u^h_{i-1,j} - 2u^h_{i,j} + u^h_{i+1,j}}{h^2} + \mathcal{O}(h^2) \text{ for } 1 \le i,j \le N-1 \quad (3.26)$$

(and similar for the y-derivative). The approximation to the partial differential equation (3.18) discretized on internal points of G_h can be written as

$$\frac{-u^h_{i,j-1} - u^h_{i-1,j} + 4u^h_{i,j} - u^h_{i+1,j} - u^h_{i,j+1}}{h^2} = f^h_{i,j} \text{ for } 1 \leq i,j \leq N-1\,. \tag{3.27}$$

The discrete Laplacian on these nodes can then be represented by a so-called *stencil* notation

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.28}$$

in which the middle row (column) represents the coupling of the unknown with its left and right (top and bottom) neighbours. This stencil is referred to as the *5-point* stencil.

In the treatment of the boundary conditions we distinguish between Dirichlet and Neumann boundary conditions.

Dirichlet Boundary Nodes Two options exist to enforce that the discrete problem satisfies the Dirichlet boundary conditions. One can either add an equation for each node on the Dirichtlet boundary by imposing for these nodes the stencil

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.29}$$

and overwriting $f^h_{i,j}$ on the boundary by $b^h_{i,j}$. Together with the equations on the internal nodes, this results in $(N+1)^2$ linear equations for the $(N+1)^2$ unknowns $\{u^h_{i,j} | 0 \leq i,j \leq N\}$. In order to preserve symmetry of the system matrix introduced later, it is required to bring the weight $-1/h^2$of a connection of an interior point (i,j) to a point on the boundary to the right-hand side vector. For an internal point $(1,j)$ connected to the left boundary $x=0$ for instance the stencil (3.28) is to be replaced by

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ 0 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.30}$$

and $f^h_{1,j}$ is to overwritten by $f^h_{1,j} + \frac{1}{h^2} b^h_{0,j}$. For the internal point $(1,1)$ with both a left and bottom neighbour on the boundary the stencil is to be replaced by

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ 0 & 4 & -1 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.31}$$

$f^h_{1,1}$ is to be overwritten by $f^h_{1,1} + \frac{1}{h^2} b^h_{0,1} + \frac{1}{h^2} b^h_{1,0}$. We will refer to the approach in which the boundary unknowns form part of the linear system as *without elimination* of the boundary conditions.

Alternatively one can take advantage of the fact that the equations for the nodes corresponding to the Dirichlet boundary are superfluous, and condense the linear system to a system for the internal points exclusively. For an interior point connected to the boundary the stencil and the right-hand vector are to be replaced as described above. This approach leads to a system of $(N-1)^2$ equations for the unknowns $\{u^h_{i,j} | 1 \leq i,j \leq N-1\}$ and will be referred to as *with elimination* of the boundary conditions.

Neumann Boundary Nodes In case that Neumann conditions are imposed, the stencil for nodes on the boundary needs to be defined. We distinguish between the four corners and the remaining boundary nodes. We start with the latter and consider a grid node $u^h_{0,j}$ with $1 \leq j \leq N-1$ on the left boundary where $x=0$. By introducing a ghost point $u^h_{-1,j}$ located at a distance h to the left of $u^h_{0,j}$, approximating the normal derivative using the central scheme

$$\frac{\partial u}{\partial n}(0,y_j) = -\frac{\partial u}{\partial x}(0,y_i) = -\frac{u^h_{1,j} - u^h_{-1,j}}{2h} + \mathcal{O}(h^2) \tag{3.32}$$

and using the boundary condition (3.16), one can eliminate the ghost point by writing

$$u^h_{-1,j} = 2\,h\,c^h_{0,j} + u^h_{1,j}\,. \tag{3.33}$$

The stencil for the nodes $u^h_{0,j}$ with $1 \leq j \leq N-1$ becomes

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ 0 & 4 & -2 \\ 0 & -1 & 0 \end{bmatrix}, \tag{3.34}$$

while in the right-hand vector $f^h_{0,j}$ is to be overwritten by $f^h_{0,j} + \frac{2}{h}c^h_{0,j}$. In order to obtain symmetry of the system matrix introduced later, the equation corresponding to this node is divided by two to obtain the stencil

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1/2 & 0 \\ 0 & 2 & -1 \\ 0 & -1/2 & 0 \end{bmatrix}, \tag{3.35}$$

and the right-hand side vector component $\frac{1}{2}f^h_{0,j} + \frac{1}{h}c^h_{0,j}$. The corner points are treated by repeating the above procedure in both x and y-direction. After dividing the equation corresponding to a corner by four, the resulting stencil for the node $u^h_{0,0}$ for instance is seen to be

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1/2 & 0 \\ 0 & 1 & -1/2 \\ 0 & 0 & 0 \end{bmatrix}, \tag{3.36}$$

while in the right-hand vector $f^h_{0,0}$ is to be overwritten by $\frac{1}{4}f^h_{0,0} + h\,c^h_{0,0}$. The stencils for interior and boundary points result in $(N+1)^2$ equations for $(N+1)^2$ unknowns.

Contrary to the case of Dirichlet boundary conditions, the discretization stencil in case of Neumann boundary conditions is adapted from the interior to the boundary without losing the property that the sum of the coefficients in the stencil equals zero. This implies that in the latter case the matrix resulting from the discretization has row sum equal to zero, and that therefore the constant vectors lie in the null space of the matrix. The discrete differential operator inherits this property from its continuous counterpart.

3.6 Linear System Formulation

To arrive at a linear system for the grid unknowns, we introduce a *global* ordering of the grid nodes. We distinguish the cases in which the boundary is and is not eliminated.

Without Elimination of the Boundary Conditions In case that the boundary conditions are not eliminated, we can introduce an x-line lexicografic ordering of the internal and boundary nodes in which

$$\text{node } (i,j) \text{ is assigned global index } I = i + 1 + j(N+1) \text{ for } 0 \leq i,j \leq N\,, \tag{3.37}$$

such that $1 \leq I \leq (N+1)^2$. This allows us to group the known and unknown grid values $f^h_{i,j}$ and $u^h_{i,j}$ into column vectors $\mathbf{u}^h$ and $\mathbf{f}^h$ of size $(N+1)^2$

$$\mathbf{u}^h = \begin{pmatrix} u^h_{0,0} \\ u^h_{0,1} \\ \vdots \\ u^h_{0,N} \\ u^h_{1,0} \\ u^h_{1,1} \\ \vdots \\ u^h_{1,N} \\ \vdots \\ u^h_{N,0} \\ u^h_{N,1} \\ \vdots \\ u^h_{N,N} \end{pmatrix} \in R^{(N+1)^2} \text{ and } \mathbf{f}^h = \begin{pmatrix} f^h_{0,0} \\ f^h_{0,1} \\ \vdots \\ f^h_{0,N} \\ f^h_{1,0} \\ f^h_{1,1} \\ \vdots \\ f^h_{1,N} \\ \vdots \\ f^h_{N,0} \\ f^h_{N,1} \\ \vdots \\ f^h_{N,N} \end{pmatrix} \in R^{(N+1)^2} . \tag{3.38}$$

The model problem then translates into a linear system of equations

$$A^h \, \mathbf{u}^h = \mathbf{f}^h , \tag{3.39}$$

in which the system matrix A^h represents the discretized differential operator. Its solution $\mathbf{u}^h$ is a second order approximation of the continuous solution u. If we measure the h-scaled Euclidean norm of the discretization error, we more precisely have that

$$\|u(x) - \mathbf{u}^h\|_{h,2} = \sqrt{h \sum_{I=0}^{N} [u(x_I) - u^h_I]^2} = \mathcal{O}(h^{-2}) , \tag{3.40}$$

and

$$\|u(x,y) - \mathbf{u}^h\|_{h,2} = \sqrt{h^2 \sum_{I=0}^{N^2} [u(x_I) - u^h_I]^2} = \mathcal{O}(h^{-2}) , \tag{3.41}$$

for the one- and two-dimensional problem, respectively. Details on the matrix A^h will be given for the one- and two-dimensional case separately.

- For the one-dimensional problem we have that
  - in case of Dirichlet boundary conditions

$$A^h = \frac{1}{h^2} \begin{pmatrix} h^2 & 0 & \dots & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -1 & 2 & -1 \\ 0 & \dots & \dots & \dots & 0 & h^2 \end{pmatrix} \in R^{(N+1)\times(N+1)} \tag{3.42}$$

This matrix can be made symmetric by translating the connections of the left-most and right-most interior points to the left and right boundary point into contributions to the right-hand side vector to obtain

$$A^h = \frac{1}{h^2} \begin{pmatrix} h^2 & 0 & \dots & \dots & \dots & 0 \\ 0 & 2 & -1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -1 & 2 & 0 \\ 0 & \dots & \dots & \dots & 0 & h^2 \end{pmatrix} \in R^{(N+1)\times(N+1)} \tag{3.43}$$

– in case of Neumann boundary conditions

$$A^h = \frac{1}{h^2}\begin{pmatrix} 1 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 1 \end{pmatrix} \in R^{(N+1)\times(N+1)} . \qquad (3.44)$$

To specify A^h for the two-dimensional problem, we denote by I^h_{N-1} and I^h_{N+1} the identity matrix on $R^{(N-1)\times(N-1)}$ and $R^{(N+1)\times(N+1)}$, respectively, by T^h the matrix

$$T^h = \begin{pmatrix} 4 & -1 & 0 & \dots & \dots & 0 \\ -1 & 4 & -1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -1 & 4 & -1 \\ 0 & \dots & \dots & 0 & -1 & 4 \end{pmatrix} \in R^{(N-1)\times(N-1)} \qquad (3.45)$$

(observe the value of 4 on the main diagonal and the absence of the scaling with h^2) and by $\widehat{I}^h$ and $\widehat{T}^h$ the matrices

$$\widehat{I}^h = \begin{pmatrix} 0 & 0 & 0 \\ 0 & I^h_{N-1} & 0 \\ 0 & 0 & 0 \end{pmatrix} \in R^{(N+1)\times(N+1)} \text{ and } \widehat{T}^h = \begin{pmatrix} h^2 & 0 & 0 \\ 0 & T^h & 0 \\ 0 & 0 & h^2 \end{pmatrix} \in R^{(N+1)\times(N+1)} . \qquad (3.46)$$

- For the two-dimensional problem we have with this notation that in case of Dirichlet boundary conditions

$$A^h = \frac{1}{h^2}\begin{pmatrix} h^2 I_{N+1} & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & \widehat{T}^h & -\widehat{I}^h & 0 & \dots & \dots & 0 \\ 0 & -\widehat{I}^h & \widehat{T}^h & -\widehat{I}^h & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -\widehat{I}^h & \widehat{T}^h & -\widehat{I}^h & 0 \\ 0 & \dots & \dots & 0 & -\widehat{I}^h & \widehat{T}^h & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 & h^2 I_{N+1} \end{pmatrix} \in R^{(N+1)^2\times(N+1)^2} . \qquad (3.47)$$

The case of Neumann boundary conditions is left as an exercise.

With Elimination of the Boundary Conditions In case that the boundary conditions are eliminated, we can introduce an x-line lexicografic ordering of the internal nodes *only* in which

$$\text{node } (i,j) \text{ is assigned global index } I = i + (j-1)(N-1) \text{ for } 1 \le i,j \le N-1 , \qquad (3.48)$$

such that $1 \le I \le (N-1)^2$. This allows us to again to group the know and unknown grid values $f^h_{i,j}$ and $u^h_{i,j}$ into column vectors u^h and f^h of size now $(N-1)^2$. The discretization again leads to a linear system of equations in which the coefficient matrix is

- for one-dimensional problems

$$A^h = \frac{1}{h^2}\text{tridiag}[\ -1 \quad 2 \quad -1\ ] = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 \end{pmatrix} \qquad (3.49)$$

- for two-dimensional problems

$$
\begin{aligned}
A^h &= \frac{1}{h^2}\text{tridiag}[\ -1 \quad 2 \quad -1\ ] \otimes I^h_{N-1} + \frac{1}{h^2} I^h_{N-1} \otimes \text{tridiag}[\ -1 \quad 2 \quad -1\ ] \\
&= \frac{1}{h^2}\text{tridiag}[\ -I^h_{N-1} \quad T^h \quad -I^h_{N-1}\ ] \\
&= \frac{1}{h^2}\begin{pmatrix}
T^h & -I^h_{N-1} & 0 & \dots & \dots & 0 \\
-I^h_{N-1} & T^h & -I^h_{N-1} & 0 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots \\
0 & \dots & 0 & -I^h_{N-1} & T^h & -I^h_{N-1} \\
0 & \dots & \dots & 0 & -I^h_{N-1} & T^h
\end{pmatrix}, \qquad (3.50)
\end{aligned}
$$

where $\otimes$ denotes the tensor product of two matrices. The tensor product notation is convenient for the implementation in MATLAB.

Alternative Grid Node Orderings For future reference we introduce two alternative grid node orderings: the diagonal and red-black ordering shown in Figure 3.4. The red-black ordering in particular induces a partitioning of the discrete Laplacian that can be written as

$$
A^h = \begin{pmatrix} A^h_{RR} & A^h_{RB} \\ A^h_{BR} & A^h_{BB} \end{pmatrix}, \qquad (3.51)
$$

where A^h_{RR} (A^h_{BB}) represents the coupling between the red (black) nodes and A^h_{RB} (A^h_{BR}) the coupling between the red and black (black and red) nodes. The submatrices A^h_{RR} and A^h_{BB} are diagonal. The submatrices A^h_{RB} and A^h_{BR} consist of four diagonal. This orderings will be further discussed in the context of basic iterative and multigrid methods.

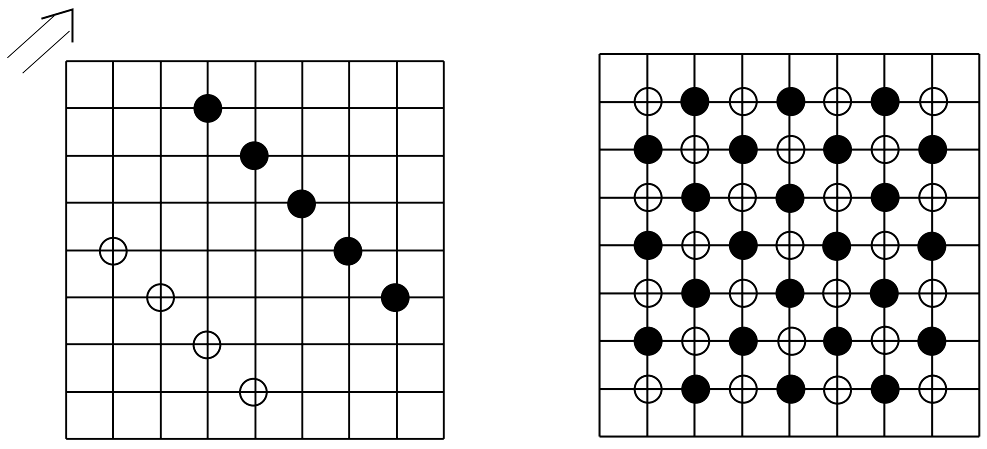

(a) Diagonal Ordering (b) Red black ordering

Figure 3.4: Diagonal and red-black ordering of the grid nodes.

3.7 Properties of the Discrete Linear System

In this section we list the properties of the linear system matrix A^h that will play an important role in subsequent chapters. For convenience, we will only consider the case with elimination of the boundary conditions from here on. The properties of A^h are:

- the matrix A^h is **sparse** with a regular tri-diagonal (in 1D) or penta-diagonal (in 2D) structure. This fact follows from the use of a compact finite difference stencil on a regular mesh. In subsequent chapters we will exploit the fact that the diagonal of A^h is constant;

- the matrix A^h is **symmetric**. By Theorem 2.3.1, the matrix A^h therefore has real eigenvalues, i.e., $\sigma(A) \subset \mathbb{R}$;
- the matrix A^h is **irreducible**. Indeed, the interconnection between nodes is such that *no* permutation matrix P exists such that PA^hP^T is block upper triangular;
- the matrix A^h is **irreducably diagonal dominant**. From the theorem of Gershgorin (Theorem 2.8.1) it therefore follows that the eigenvalues of A^h are positive. The symmetric matrix A^h is therefore **positive definite** and thus SPD;
- the matrix A^h satisfies all the conditions in Theorem 2.9.1 and is therefore an **M-matrix**.

All of the above properties can be traced back to properties of the continuous differential operator. For future analysis it will useful to have analytical expressions for the eigenvalues and the eigenvectors of A^h. These are provided in the next subsection.

3.7.1 Discrete Spectrum

The following two theorems state that the discrete Laplacian A^h inherits its spectral properties from its continuous counterpart. We start with the one-dimensional case.

Theorem 3.7.1 *In the one-dimensional problem and with elimination of the boundary conditions, the matrix $A^h \in \mathbb{R}^{(N-1)\times(N-1)}$ has the following eigenvalues and eigenvectors*

$$A^h \mathbf{v}^{h,[k]} = \lambda_k^h \mathbf{v}^{h,[k]} \tag{3.52}$$

where for $k = 1, \ldots, N-1$

$$\mathbf{v}^{h,[k]} = \frac{1}{\sqrt{N-1}} \begin{pmatrix} \sin(\pi k\, x_1) \\ \vdots \\ \sin(\pi k\, x_{N-1}) \end{pmatrix} = \frac{1}{\sqrt{N-1}} \begin{pmatrix} \sin(\pi k\, h) \\ \vdots \\ \sin(\pi k\, (N-1)\, h) \end{pmatrix} \tag{3.53}$$

and

$$\lambda_k^h(A^h) = \frac{2}{h^2}[1 - \cos(\pi\, h\, k)] = \frac{2}{h^2} 2 \sin^2(\frac{\pi\, h\, k}{2})\,. \tag{3.54}$$

Proof. The proof consist of performing a matrix-vector multiplication with the matrix A^h and the vector $\mathbf{v}^{h,[k]}$. We will denote the elements of the matrix A^h, the vector $\mathbf{v}^{h,[k]}$ and the resulting vector $A^h\, \mathbf{v}^{h,[k]}$ as $A_{i\alpha}$, $\mathbf{v}_\alpha^{h,[k]}$ and $[A^h\, \mathbf{v}^{[k]}]_i$, respectively, and make use of the trigonometric identity

$$\sin(\alpha + \beta) + \sin(\alpha - \beta) = 2 \sin(\alpha) \cos(\beta)\,. \tag{3.55}$$

For an index i corresponding to a node not connected to the boundary we have that

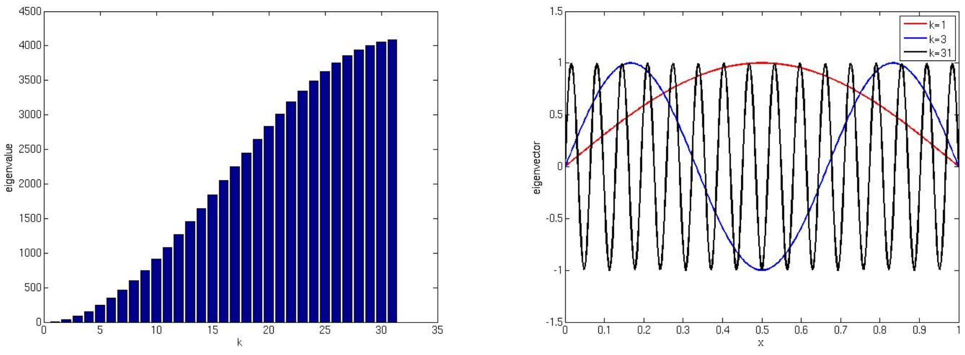
$$\begin{aligned}
[A^h\, \mathbf{v}^{h,[k]}]_i &= \sum_{\alpha=1}^{N-1} a_{i\alpha} v_\alpha^{h,[k]} \qquad (3.56) \\
&= \frac{1}{h^2} \frac{1}{\sqrt{N-1}} \left[-\sin(\pi(i-1)\, h\, k) + 2\sin(\pi i\, h\, k) - \sin(\pi(i+1)\, h\, k) \right] \\
&= \frac{1}{h^2} \frac{1}{\sqrt{N-1}} \left[2\sin(\pi i\, h\, k) - 2\sin(\pi i\, h\, k)\cos(\pi\, h\, k) \right] \\
&= \frac{2}{h^2} \frac{1}{\sqrt{N-1}} [1 - \cos(\pi\, h\, k)] \sin(\pi i\, h\, k) \\
&= \frac{2}{h^2} [1 - \cos(\pi\, h\, k)]\, v_i^{h,[k]} \\
&= \lambda_k^h\, v_i^{h,[k]} \text{ for } 2 \leq i \leq N-2\,.
\end{aligned}$$

For the left-most interior node instead we have that

$$\begin{aligned}
[A^h\,\mathbf{v}^{h,[k]}]_1 &= \frac{1}{h^2}\frac{1}{\sqrt{N-1}}\,[2\sin(\pi h\,k) - \sin(\pi\,2\,h\,k)] \\
&= \frac{1}{h^2}\frac{1}{\sqrt{N-1}}\,[2\sin(\pi h\,k) - 2\,\cos(\pi h\,k)\sin(\pi h\,k)] \\
&= \frac{2}{h^2}[1-\cos(\pi\,h\,k)]\,v_1^{h,[k]} \\
&= \lambda_k^h\,v_1^{h,[k]}\,,
\end{aligned} \tag{3.57}$$

which, given the fact that a similar computation holds for $i = N-1$ and given that k was chosen arbitrarily in its range, completes the proof. □

The verification that the eigenvalues derived above lie in the Gershgorin disks is left as an exercise. In Figure 3.7.1 the eigenvalues (as a function of k) and three eigenvectors (as a function of x) are shown for $N = 32$. Notice how the eigenvalues and the frequency of the eigenvectors increase with k.



(a) Eigenvalues (b) Eigenvectors

Figure 3.5: Eigenvectors and eigenvectors of the one-dimensional discrete Laplacian with elimination of the boundary for $N = 32$.

Corollary 3.7.1 *The set of $N-1$ eigenvectors $\mathbf{v}^{h,[k]}$ for $k = 1,\dots,N-1$ forms an orthonormal basis of $\mathbb{R}^{N-1}$. In this basis the matrix A^h can be diagonalized, i.e., given the matrix V^h whose columns are the vectors $\mathbf{v}^{h,[k]}$ and the diagonal matrix Λ^h whose diagonal entries are the eigenvalues are the eigenvalues λ_k^h, we have that*

$$A^h = V^h\Lambda^h(V^h)^T\,. \tag{3.58}$$

The next theorem extends the previous result to the two-dimensional case.

Theorem 3.7.2 *In the two-dimensional problem and with elimination of the boundary conditions, the matrix A^h has the following eigenvalues and eigenvectors*

$$A^h\mathbf{v}^{h,[k\ell]} = \lambda_{k\ell}^h\mathbf{v}^{h,[k\ell]} \tag{3.59}$$

where for $k, \ell = 1, \ldots, N-1$

$$\mathbf{v}^{h,[k\ell]} = \frac{1}{(N-1)} \begin{pmatrix} \sin(\pi k\, x_1)\sin(\pi\ell\, y_1) \\ \vdots \\ \sin(\pi k\, x_{N-1})\sin(\pi\ell\, y_1) \\ \sin(\pi k\, x_1)\sin(\pi\ell\, y_2) \\ \vdots \\ \sin(\pi k\, x_{N-1})\sin(\pi\ell\, y_2) \\ \vdots \\ \sin(\pi k\, x_1)\sin(\pi\ell\, y_{N-1}) \\ \vdots \\ \sin(\pi k\, x_{N-1})\sin(\pi\ell\, y_{N-1}) \end{pmatrix} = \frac{1}{(N-1)} \begin{pmatrix} \sin(\pi k\, h)\sin(\pi\ell\, h) \\ \vdots \\ \sin(\pi k\, (N-1)\, h)\sin(\pi\ell\, h) \\ \sin(\pi k\, h)\sin(\pi\ell\, 2\, h) \\ \vdots \\ \sin(\pi k\, (N-1)\, h)\sin(\pi\ell\, 2\, h) \\ \vdots \\ \sin(\pi k\, h)\sin(\pi\ell\, (N-1)\, h) \\ \vdots \\ \sin(\pi k\, (N-1)\, h)\sin(\pi\ell\, (N-1)\, h) \end{pmatrix}$$

and

$$\lambda_{k\ell}(A^h) = \frac{4}{h^2}[1 - \frac{1}{2}\cos(\pi\, h\, k) - \frac{1}{2}\cos(\pi\, h\, \ell)] = \frac{4}{h^2}[\sin^2(\frac{\pi\, h\, k}{2}) + \sin^2(\frac{\pi\, h\, \ell}{2})]\,. \tag{3.60}$$

Proof. The proof again consist of performing a matrix-vector multiplication. For indices i and j corresponding to a internal node not connected to the boundary, we have that

$$\begin{aligned} [A^h\, \mathbf{v}^{h,[k\ell]}]_{ij} &= \sum_{\alpha,\beta=1}^{(N-1)^2} a_{\alpha\beta,ij}\, v_{\alpha\beta}^{h,[k\ell]} \qquad (3.61) \\ &= \frac{1}{h^2}\frac{1}{N-1}\,[-\sin(\pi(i-1)\, h\, k)\sin(\pi j\, h\, \ell) - \sin(\pi(i+1)\, h\, k)\sin(\pi j\, h\, \ell) \\ &\quad +4\sin(\pi i\, h\, k)\sin(\pi j\, h\, \ell) \\ &\quad -\sin(\pi i\, h\, k)\sin(\pi(j-1)\, h\, \ell) - \sin(\pi i\, h\, k)\sin(\pi(j+1)\, h\, \ell)] \\ &= \frac{1}{h^2}\frac{1}{N-1}\,[4\sin(\pi i\, h\, k)\sin(\pi j\, h\, \ell) \\ &\quad -2\sin(\pi i\, h\, k)\sin(\pi j\, h\, \ell)\cos(\pi\, h\, k) - 2\sin(\pi i\, h\, k)\sin(\pi j\, h\, \ell)\cos(\pi\, h\, \ell)] \\ &= \frac{4}{h^2}\frac{1}{N-1}[1 - \frac{1}{2}\cos(\pi\, h\, k) - \frac{1}{2}\cos(\pi\, h\, \ell)]\sin(\pi i\, h\, k)\sin(\pi j\, h\, \ell) \\ &= \frac{4}{h^2}[1 - \frac{1}{2}\cos(\pi\, h\, k) - \frac{1}{2}\cos(\pi\, h\, \ell)]\, v_{ij}^{h,[k\ell]}\,. \end{aligned}$$

A similar computation can be done for the other nodes, completing the proof. □
The $(N-1)^2$ vectors $\mathbf{v}^{h,[k\ell]}$ can be numbered linearly using the same (e.g. lexicografic) ordering as the grid nodes.

Corollary 3.7.2 *The set of* $(N-1)^2$ *eigenvectors* $\mathbf{v}^{h,[k\ell]}$ *for* $k, \ell = 1, \ldots, N-1$ *forms an orthonormal basis of* $\mathbb{R}^{(N-1)^2}$. *In this basis the matrix* A^h *can be diagonalized, i.e., given the matrix* V^h *whose columns are the vectors* $\mathbf{v}^{h,[k\ell]}$ *and the diagonal matrix* Λ^h *whose diagonal entries are the eigenvalues are the eigenvalues* $\lambda_{k\ell}^h$*, we have that*

$$A^h = V^h \Lambda^h (V^h)^T\,. \tag{3.62}$$

Next we investigate asymptotic limits for the eigenvalues as $h \to 0$. Using the fact that

$$\cos(x) = 1 - \frac{1}{2}x^2 + \mathcal{O}(x^4) \text{ as } x \to 0 \tag{3.63}$$

we have that

$$\lambda_{k\ell} = \pi^2(k^2 + \ell^2) \text{ as } h \to 0\,, \tag{3.64}$$

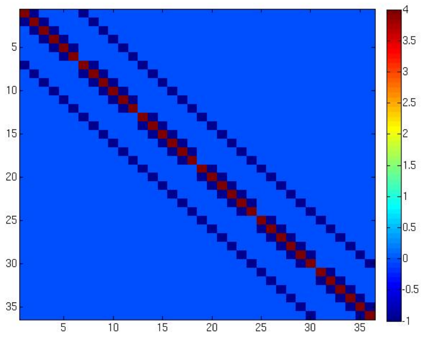
i.e., the eigenvalues of the continuous operator are recovered. We furthermore have for the quotient of the largest and smallest eigenvalue that

$$\begin{aligned}\frac{\lambda_{(N-1)(N-1)}}{\lambda_{11}} &= \frac{1-\frac{1}{2}\cos(\pi\,h\,(N-1))-\frac{1}{2}\cos(\pi\,h\,(N-1))}{1-\frac{1}{2}\cos(\pi\,h)-\frac{1}{2}\cos(\pi\,h)} \\ &= \frac{\pi^2\,h^2\,(N-1)^2+\mathcal{O}(h^4)}{\pi^2\,h^2+\mathcal{O}(h^4)} \\ &= (N-1)^2+\mathcal{O}(h^4) \\ &= \mathcal{O}(h^{-2}) \qquad (3.65)\end{aligned}$$

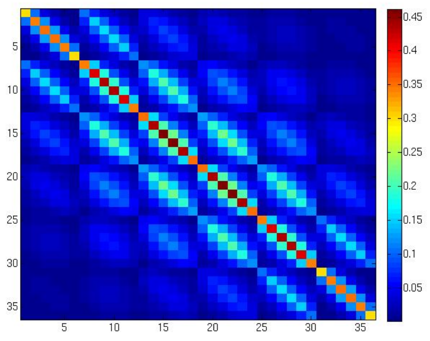
where we used the fact that $N = \frac{1}{h}$. This implies that A^h becomes worse conditioned on finer meshes. Summarizing this section we can state that

$$\boxed{\text{the discrete Laplacian } A^h \text{ is a sparse structured diag. dominant spd M-matrix with } \kappa_2(A) = \mathcal{O}(h^{-2})\,.} \qquad (3.66)$$

The properties derived in this section will play a *central* role in *all* subsequent chapters. To illustrate the M-matrix property, we plotted in Figure 3.7.1 the elements of A^h and of its inverse. The colourbar to the right of the graph of the latter shows that all its entries are positive confirming that A^h is indeed an M-matrix.



(a) A^h



(b) $(A^h)^{-1}$

Figure 3.6: Plot of entries of the discrete Laplacian A^h and its inverse $(A^h)^{-1}$.

3.8 Beyond the Model Problem

In this section we discuss to which extend the nice properties of the discrete Laplacian discussed above carry over to more realistic problems. Doing so, we will treat elliptic and parabolic problems seperately.

3.8.1 Elliptic Problems

Anisotropic Poisson Equation For the anisotropic Poisson equation (3.5) the 5-point finite difference stencil (3.28) needs to be modified into

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ -\epsilon & 2+2\epsilon & -\epsilon \\ 0 & -1 & 0 \end{bmatrix}\,. \qquad (3.67)$$

The resulting matrix is still an SPD M-matrix, but the presence of ϵ affects the eigenvalues and therefore the condition number.

Discontinuous Coefficients Poisson Equation For the discontinuous coefficients Poisson equation (3.6) the finite difference stencil for a point lying on the interface running vertically between two regions having diffusion coefficient μ^+ and μ^- is

$$\frac{1}{h^2}\begin{bmatrix} 0 & -(\mu^+ + \mu^-) & 0 \\ -\mu^- & 2(\mu^+ + \mu^-) & -\mu^+ \\ 0 & -(\mu^+ + \mu^-) & 0 \end{bmatrix}. \tag{3.68}$$

As in the anisotropic case, the resulting matrix is still an SPD M-matrix, but the discontinuity in μ affects the eigenvalues and the condition number.

Poisson Equation with Anti-Periodic Boundary Conditions The M-matrix property is lost if so-called *anti-periodic* period boundary conditions are imposed. These conditions state that

$$u_{\Gamma_1} = -u_{\Gamma_2} \tag{3.69}$$

where Γ_1 and Γ_2 are distinct parts of the boundary, and are frequently used in modeling in the presence of particular symmetry (as for instance in the modeling of rotary electrical machines). Suppose that Γ_1 and Γ_2 correspond to the left and right part of boundary of the unit square, respectively, and that the nodes on Γ_1 are eliminated from the linear system. An interior node with left neighbour on Γ_1 then gets connected to Γ_2 with a *positive* weight and has the stencil

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ 1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{3.70}$$

The resulting matrix is SPD, but no longer an M-matrix.

Poisson Equation Discretized by Higher Order Schemes The discretization of the Poisson equation using higher order compact or non-compact finite difference schemes (see e.g. [64]) results in matrices that are SPD, but not M-matrices due to positive off-diagonal entries. These matrices are denser than their lower order counterparts.

Poisson Equation Discretized on Complex Geometries The discretization of the Poisson equation on complex geometries typically requires the use of unstructured triangular meshes as shown if Figure 3.7. The resulting matrices are still SPD. If lower order methods are used and if the mesh meets certain mild requirements, the resulting matrices are even M-matrices! Their sparsity structure however is irregular.

Helmholtz Problem The low order finite difference discretization of the Helmholtz equation (3.7) results in the stencil

$$\frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 - k^2h^2 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{3.71}$$

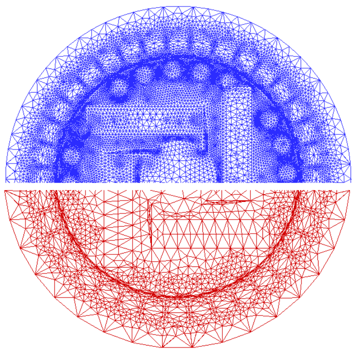
The resulting matrix is symmetric. It losses its diagonal dominance, positive definiteness and M-matrix property in case that the wave number k is sufficiently large.

Convection-Diffusion Problem leading to a Non-Symmetric Matrix The discretization of the convection-diffusion equation (3.8) using the 5-point finite difference stencil (3.28) for the diffusive terms and e.g. the following central scheme

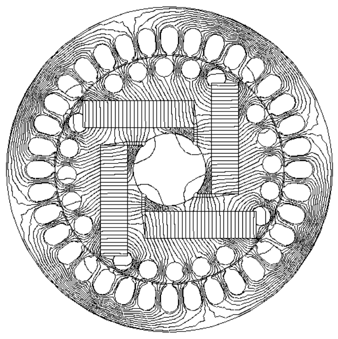
$$\frac{\partial u}{\partial x}(x_i, y_j) = \frac{u^h_{i+1,j} - u^h_{i-1,j}}{2h} + \mathcal{O}(h^2) \tag{3.72}$$

for the convective term, results in the stencil

$$\frac{\epsilon}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ -1 - \frac{h}{2\epsilon} & 4 & -1 + \frac{h}{2\epsilon} \\ 0 & -1 & 0 \end{bmatrix}. \tag{3.73}$$



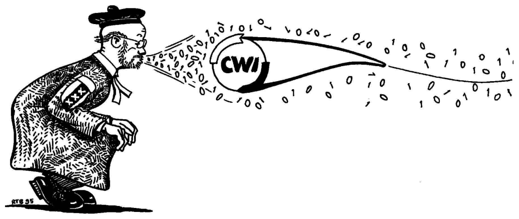
(a) Coarse and fine mesh



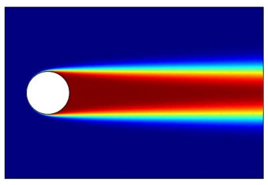
(b) Equipotentials of the computed solution

Figure 3.7: The Poisson equation solved on a complex geometry discretized by unstructured grids.

The resulting matrix has the same sparsity pattern as the discrete Laplacian, but its non-symmetry complicates its iterate solution considerably.



(a) Problem formulation.



(b) Computed solution

Figure 3.8: The convection-diffusion equation modeling the flow around an obstacle.

3.8.2 Parabolic Problem

The discretization of parabolic problems is typically performed by a two-stage procedure in which the system of coupled ordinary differential equations resulting from the spatial discretization, is discretized in time by a time-stepping procedure. This way the numerical solution procedure naturally follows the physics of the problem. The coefficient matrix of the system of linear equations to be solved at each time step has the same sparsity pattern as the discrete Laplacian. Due to the time stepping scheme, the matrices to be solved are more diagonally dominant than the discrete Laplacian, rendering their iterative solution easier.

3.8.3 Dense Matrices in PDE Problems

The discretization of PDEs does not necessarily lead to sparse matrices. The discretization of integro-differential equations for instance leads to linear systems with a dense coefficient matrix. As example we consider the Pocklington equation that models the current $u(x)$ induced in a thin wire antenna of length L exposed to an incident electrical wave f(x) with wavenumber k and that can be written as

$$\frac{d}{dx}\int_0^L dx' G(x-x')\frac{d\,u(x')}{dx'} + k^2 \int_0^L dx' G(x-x')u(x) = f(x) \tag{3.74}$$

with integration kernel

$$G(x-x') = \frac{\exp(-j\,k\,|x-x'|)}{4\pi|x-x'|}\,. \tag{3.75}$$

The discretization of this equation leads to symmetric Toeplitz matrices that are amenable to being solved by iterative solution methods.

3.8.4 Sparse Matrices in non-PDE Problems

Sparse matrices not only appear after the discretization of partial differential equations, but are central in the modeling of electrical, social and computer networks. The modeling of a power distribution in network with time independent generators and loads for instance gives rise to sparse SPD M-matrices.

3.9 List of Model Problems

For future reference we give here an enumerated list of model problem

- **MP**-1: 1D or 2D Poisson equation with Dirichtlet boundary conditions, discretized by 5-point stencil;
- **MP**-2: 1D or 2D Poisson equation with Neumann boundary conditions, discretized by 5-point stencil;
- **MP**-3: 2D Poisson equation with anti-periodic boundary conditions, discretized by 5-point stencil;
- **MP**-4: 2D Poisson equation with Dirichtlet boundary conditions, discretized by a non-compact higher order scheme
- **MP**-5: convection diffusion equation discretized by a central scheme for the flux.

3.10 Computer Implementation

Storing sparse matrices by using two-dimensional arrays is memory inefficient. Sparse matrix storage schemes such as the compressed row format, that avoid having to store the zero elements, have therefore been developed.

3.11 Exercises

Exercise 3.11.1 Show that the operator $\mathcal{L}$ with mixed derivatives in the equation $-\Delta u + \tau u_{xy} = b$ is elliptic for $|\tau| < 2$, parabolic for $|\tau| = 2$ and hyperbolic for $|\tau| > 2$.

Exercise 3.11.2 Determine the type of each of the following PDEs for $u = u(x, y)$.

a) $u_{xx} + u_{yy} = b$
b) $-u_{xx} - u_{yy} + 3u_y = b$
c) $u_{xy} - u_x - u_y = b$
d) $3u_{xx} - u_x - u_y = b$
e) $-u_{xx} + 7u_{xy} + 2u_x + 3u = b$

Exercise 3.11.3 Determine the type of each of the following PDEs in dependence of the parameter ε for $u = u(x, y)$.

a) $-\varepsilon u_{xx} - u_{yy} = b$
b) $-\varepsilon \Delta u + a_1 u_x + a_2 u_y = b$

Exercise 3.11.4 In the case of nonconstant coefficients $a_{11} = a_{11}(x, y)$, $a_{12} = a_{12}(x, y)$, $a_{22} = a_{22}(x, y)$, it is possible to generalize the determination of the type of an equation (elliptic, hyperbolic, parabolic) in a straightforward way. In that case, the determinant

$$a_{11}a_{22} - a_{12}^2$$

depends on x, y. The equation type then also depends on x, y. With coefficients a_{ij} depending of solution u and/or derivatives u_x, u_y the differential equation is called quasilinear. It is also in that case possible to determine the equation type in a similar way.

An equation describing the flow of a stationary, rotation-free, inviscid ideal fluid is the so-called full potential equation. In 2D, it reads,

$$u_{xx} + u_{yy} - \frac{1}{C^2}(u_x^2 u_{xx} + 2u_x u_y u_{xy} + u_y^2 u_{yy}) = 0$$

where $C = C(u_x, u_y)$ is the local speed of sound. (C also depends on other physical quantities, such as density ρ.) Vector (u_x, u_y) represents the velocity, with components in x- and y-direction, respectively.

Determine the type of the equation in dependence on C.

Can you interpret your solution from a physical point of view?

Exercise 3.11.5 Show that, for sufficiently smooth functions u, the differential equation

$$\tilde{u}_{\xi\eta} = 0$$

can be transformed into the well-known wave equation

$$u_{xx} - u_{yy} = 0$$

by the coordinate transformation

$$x = (\xi + \eta)/2, \quad y = (\xi - \eta)/2.$$

Exercise 3.11.6 Show that the Black-Scholes equation

$$\frac{\partial u}{\partial t} + \frac{\sigma^2}{2}s^2\frac{\partial^2 u}{\partial s^2} + rs\frac{\partial u}{\partial s} - ru = 0$$

for $u(s, t)$ is equivalent to the equation

$$\frac{\partial y}{\partial \tau} = \frac{\partial^2 y}{\partial x^2}$$

for $y(x, \tau)$. To show this, proceed as follows:

a) For the transformation $s = Ee^x$ and an appropriate transformation $t \leftrightarrow \tau$, the Black-Scholes equation is equivalent to

$$-\dot{u} + u'' + \alpha u' + \beta u = 0$$

with $\dot{u} = \frac{\partial u}{\partial \tau},\ u' = \frac{\partial u}{\partial x}$, and α, β depending on r and σ.

b) The remaining part follows after a transformation of the type

$$u = Ee^{\gamma x + \delta \tau} y(x, \tau)$$

with appropriate $\gamma,\ \delta$.

Exercise 3.11.7 Let $[s_{\kappa_1 \kappa_2}]_h$ be the five-point stencil

$$[s_{\kappa_1 \kappa_2}]_h = \frac{1}{h^2} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}_h$$

and $\Omega_h = \{(ih, jh) \mid 0 < i, j < 5\}$.

a) Let w_h be the grid function

$$w_h(ih, jh) = 1 \qquad (ih, jh) \in \Omega_h\,.$$

Calculate $[s_{\kappa_1 \kappa_2}]_h w_h(3h, 2h)$ and $[s_{\kappa_1 \kappa_2}]_h w_h(3h, 3h)$.

b) Let w_h be the grid function

$$w_h(ih, jh) = jh \qquad (ih, jh) \in \Omega_h\,.$$

Calculate $[s_{\kappa_1 \kappa_2}]_h w_h(3h, 2h)$ and $[s_{\kappa_1 \kappa_2}]_h w_h(3h, 3h)$.

c) Let w_h be the grid function

$$w_h(ih, jh) = (jh)^2 \qquad (ih, jh) \in \Omega_h\,.$$

Calculate $[s_{\kappa_1 \kappa_2}]_h w_h(3h, 2h)$ and $[s_{\kappa_1 \kappa_2}]_h w_h(3h, 3h)$.

d) In Section 1.4, we will see that the five-point stencil is a discretization of the Laplacian $-\Delta = -\partial^2/\partial x^2 - \partial^2/\partial y^2$. In view of this fact, interpret the results of a) - c) in terms of their continuous analogs.

Exercise 3.11.8 We derive finite difference methods systematically for second order derivatives of functions of one variable. Therefore, we consider the central difference for a second derivative, which we write in the form

$$(u_i'')_h = \frac{1}{h^2}(a_1 u_{i-1} + a_2 u_i + a_3 u_{i+1})$$

where a_1, a_2 and a_3 represent parameters, that must be calculated so that the scheme has a high accuracy. By Taylor's expansion of u_{i-1} and u_{i+1} around u_i we find

$$\begin{aligned}(u_i'')_h =& \frac{1}{h^2}\Big(u_i(a_1 + a_2 + a_3) + hu_i'(-a_1 + a_3) + \frac{1}{2!}h^2 u_i''(a_1 + a_3) \\ &+ \frac{1}{3!}h^3 u_i'''(-a_1 + a_3) + \frac{1}{4!}h^4 u_i''''(a_1 + a_3) + \cdots\Big)\end{aligned}$$

Use this equation to compute the best possible approximation for the second derivative of u under the assumption that h is very small.

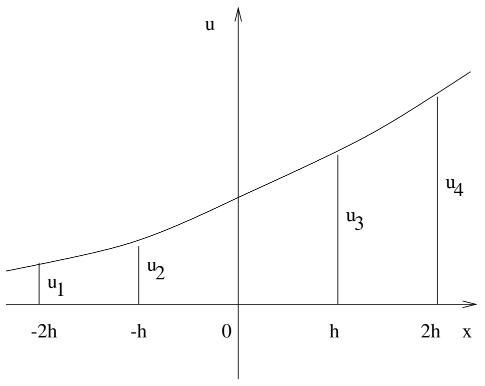


Figure 3.9: Grid points for a higher order central difference scheme.

Exercise 3.11.9 One can obtain a higher order central difference scheme for the approximation of a first derivative u', if one uses, for example, u-values at four grid points, $x = -2h,\ x = -h,\ x = h$ and $x = 2h$ (see Fig. 3.9).

The approximation for the first derivative at $x = 0$ is written in the form

$$(u_0')_h = a_1u_1 + a_2u_2 + a_3u_3 + a_4u_4,$$

where a_1, a_2, a_3 and a_4 are parameters that must be calculated so that the scheme has the required accuracy.

Calculate the parameters a_1, a_2, a_3 and a_4 so, that the approximation of u' is exact for the functions $u(x) = 1, u(x) = x, u(x) = x^2$ and $u(x) = x^3$.

Exercise 3.11.10 Let the following Sturm-Liouville boundary value problem be given:

$$-u'' + p(x)u' + q(x)u = r(x), \quad u(a) = \alpha, \quad u(b) = \beta$$

with $p(x) \geq p_0 > 0$ for $x \in [a, b]$.
We search for approximations $\tilde{u}_i$ of the exact values $u(x_i)$, $x_i = a + ih$, $i = 1, .., n$ and $h = \dfrac{b-a}{n+1}$. If one replaces $u'(x_i)$ by $\dfrac{\tilde{u}_{i+1} - \tilde{u}_{i-1}}{2h}$ and $u''(x_i)$ by $\dfrac{\tilde{u}_{i-1} - 2\tilde{u}_i + \tilde{u}_{i+1}}{h^2}$ for $i = 1, ..., n$ and further sets $\tilde{u}_0 = \alpha$ and $\tilde{u}_{n+1} = \beta$, a system of equations is obtained for the vector $\tilde{u} := (\tilde{u}_1, ..., \tilde{u}_n)^T$

$$A\tilde{u} = b \quad \text{with} \quad A \in \mathbb{R}^{n\times n},\ c \in \mathbb{R}^n.$$

a) Determine A and b.
b) For which $h > 0$ does A satisfy the requirement $a_{i,j} \leq 0$ for $i \neq j$?

Exercise 3.11.11

a) Set up three second-order accurate discretizations for the mixed derivative of the form u_{xy}, that can be represented in stencil notation by,

$$A_h = \frac{1}{h^2}\begin{bmatrix} a_2 & a_1 & 0 \\ a_1 & a_3 & a_1 \\ 0 & a_1 & a_2 \end{bmatrix}_h, \quad B_h = \frac{1}{h^2}\begin{bmatrix} 0 & b_1 & b_2 \\ b_1 & b_3 & b_1 \\ b_2 & b_1 & 0 \end{bmatrix}_h, \quad C_h = \frac{1}{h^2}\begin{bmatrix} c_1 & 0 & c_2 \\ 0 & 0 & 0 \\ c_3 & 0 & c_4 \end{bmatrix}_h;$$

b) Consider the equation

$$-\Delta u - \tau u_{xy} = 0 \quad (\Omega = (0,1)^2) \tag{3.76}$$
$$u = g \quad (\partial\Omega). \tag{3.77}$$

Write down a discretization in stencil notation for an interior grid point.

c) A Matrix A is called Z-matrix, if $a_{ij} \leq 0$ for $i \neq j$. The Z-matrix property is (for example, as a part of the definition of M-matrices) a basis for convergence proofs of certain iterative solution methods. For which τ-values and which discretization approaches discussed under (a), does the boundary value problem (3.77) result in a Z-matrix? Which discretization approach for (3.77) would give the Z-matrix property for general $-2 < \tau < 2$?

Exercise 3.11.12 Prove the relation

$$\mathcal{L}u - L_h u = O(h^2) \quad \text{for } h \to 0$$

for sufficiently smooth functions u for the standard five-point discretization

$$-\Delta_h = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_h$$

of the operator $\mathcal{L} = -\Delta$ using Taylor expansion.

Exercise 3.11.13 Consider the 1D problem $\mathcal{L}u(x) = -u''(x) = b(x)$ on the interval $\Omega = (0,1)$ with boundary conditions $u(0) = u_0$, $u(1) = u_1$. Set up the discrete problem (matrix and right-hand side) for the discretization

$$L_h = \frac{1}{h^2}\begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$$

and $h = 1/4$

a) without elimination of boundary conditions,
b) with elimination of boundary conditions.

Exercise 3.11.14 Consider $\mathcal{L}u(x) = -\Delta u(x,y) = b(x,y)$ on the domain $\Omega = (0,1)^2$ with boundary conditions $u(x,y) = g(x,y)$. Set up the discrete problem (matrix and right-hand side) for the standard five-point discretization of $\mathcal{L}$ (see Exercise 3.11.12) and $h = 1/3$

a) without elimination of boundary conditions,
b) with elimination of boundary conditions,

using a lexicographic ordering of grid points in both.

Exercise 3.11.15 Consider $\mathcal{L}u(x) = -\Delta u(x,y) = b(x,y)$ on the domain $\Omega = (0,1)^2$ with boundary conditions $u(x,y) = g(x,y)$. Set-up the discrete problem (matrix and right-hand side) for the standard five-point discretization of $\mathcal{L}$ (see Exercise 3.11.12) and $h = 1/3$

a) without elimination of boundary conditions,
b) with elimination of boundary conditions,

using a red–black ordering of grid points.

Exercise 3.11.16 Determine the sparsity structure of the five-point stencil if the grid points are ordered in alternating lines

$$\begin{array}{rrrr} 13 & 14 & 15 & 16 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 1 & 2 & 3 & 4 \end{array}$$

Chapter 4

Direct Solution Methods

`We learn from failures.`

4.1 Introduction

In this chapter we study direct solution methods for systems of linear equations. Given a non-singular coefficient matrix $A \in \mathbb{R}^{n \times n}$ and a right-hand vector $\mathbf{f} \in \mathbb{R}^n$, our goal is to solve the linear system

$$A\mathbf{u} = \mathbf{f}\,, \tag{4.1}$$

as efficiently and as reliably as possible. We will first discuss Gaussian elimination for A being a square, non-singular but otherwise general matrix. We will treat row permutations as a technique to keep pivot elements small in size and keep the error bounds of the Gaussian elimination sharp. We then extend this discussion for symmetric positive definite, banded and finally sparse matrices A. Direct solution methods are the methods of choice for solving problems that are moderate in size. The overhead in the use of these methods can be substantially reduced in solving problems with multiply right-hand sides that typically arise in time-stepping or parameter studies. Direct solution methods can also be combined with iterative solvers in various ways. Direct methods are for instance used as subdomain solver in domain-decomposition methods and as coarse grid solvers in multigrid methods. The former therefore remain an important component in current day solvers. Due to their requirement in computational resources however, direct methods are inadequate as stand-alone solvers the large scale problems that typically arise in scientific computing.

Study goals In this chapter we aim at

- introduce the concept of stability analysis for linear systems and give an intuitive interpretation to the condition number of a matrix;
- introducing the concept of the LU-factorization of the system matrix and of the forward and backward substitutions, discussing the computational complexity of these methods as well as their error bounds;
- explaining how partial pivoting allows to keep the pivoting elements small and therefore the algorithm numerically stable;
- discuss how to take advantage of system matrix properties such as diagonal dominance, symmetric positive definiteness, banded non-zero structures and sparsity to taylor the direct solutions methods to the problems being solved.

The monograph on numerical linear algebra by Golub and van Loan [23], of Jack Demmel [12], of Trefethen and Bau [63] and of Biswa Nath Datta [10] contain a chapter of direct solution methods. The direct solution of sparse linear systems is covered by the book of Duff, Erisman and Reid [13] and of Timothy

Davies [11]. Research into direct solution methods for sparse linear systems is actively being pursued giving rise to modern implementations such as MUMPS [1], PARDISO [55] and SUPERLU [39]. The report [27] compares the performance of different implementations using a collection of test matrices.

4.2 Floating Point Arithmetic

- t-digit floating point representation (subset of real numbers)

  $$x = \pm m \cdot \beta^e$$

  where m, β and e are the mantissa, base and exponent. m is a t-digit fraction. The exponent e varies between upper and lower bounds L and U, i.e., $L \leq e \leq U$. Include some example.

- Given a real number x, we denote by fl(x) its floating point representation obtained by rounding. The relative error in this representation is bounded by the machine precision denoted by μ, i.e.,

  $$\frac{|\text{fl}(x) - x|}{|x|} \leq \mu \text{ where } \mu = \frac{1}{2}\beta^{1-t}\,.$$

  This error bound is equivalent to

  $$\text{fl}(x) = x(1+\delta) \text{ where } |\delta| \leq \mu\,.$$

- elementary operations on real numbers

  $$\begin{aligned} \text{fl}(x \pm y) &= (x \pm y)(1+\delta) \text{ where } |\delta| \leq \mu \\ \text{fl}(xy) &= (xy)(1+\delta) \text{ where } |\delta| \leq \mu \\ \text{if } y \neq 0 \text{ then } \text{fl}(x/y) &= (x/y)(1+\delta) \text{ where } |\delta| \leq \mu \end{aligned}$$

- floating point arithmetic is **not** associative
- floating point arithmetic in the algebra on vectors and matrices
- the operations $+$, $-$, $*$, $/$ and $\sqrt{}$ have the same computational cost
- denoting by n the problem size, the assumption that $n\mu \leq 0.1$ is realistic (need to explain what base and number of digits is being used)

4.3 Perturbation Analysis

Prior to discussing the performance of *any* solution algorithm for the system of linear equations (4.1), it is crucially important to distinguish the following two types of difficulties

- those intrinsically related to the linear systems considered (and from which any solution algorithm will suffer);
- those related to a particular solution algorithm.

To be able to distinguish the above two types, we will perform in this section a perturbation analysis on the linear system (4.1). We will describe how small changes in the coefficient matrix A and the right-hand side vector $\mathbf{f}$ (the problem or input data) affect the solution vector $\mathbf{u}$ (the output). Such small changes can either be caused by measurement error or by round-off in the computation of A or $\mathbf{f}$ due to finite precision arithmetic. We will provide two results. In the first we limit ourselves to perturbations in $\mathbf{f}$ only. This will allow us to provide an intuitive explanation for the concept of the condition number of the matrix A and to show that a small perturbation in the data might result in a large perturbation in the computed solution if the problem is badly conditioned, i.e., if condition number is much larger than 1. In the second result the analysis is generalized to the case in which both the matrix A and the right-hand vector $\mathbf{f}$ are perturbed. The analysis in this section is by no means linked to a particular (direct or iterative) solution algorithm.

4.3.1 Perturbation in right-hand side only

Given the linear system (4.1), we aim at describing the effect of a perturbation in $\mathbf{f}$ on the solution $\mathbf{u}$. We assume $\mathbf{f} \neq \mathbf{0}$ and A non-singular such that $\mathbf{u} \neq \mathbf{0}$. Given a submultiplicative norm $\|\cdot\|$ and $\delta > 0$, we assume a perturbation of the form

$$\mathbf{f} \rightarrow \mathbf{f} + \Delta\mathbf{f} \text{ where } \|\Delta\mathbf{f}\| \leq \delta \|\mathbf{f}\| \,. \tag{4.2}$$

If we assume that A is not affected by the perturbation, the perturbed solution $\mathbf{u} + \Delta\mathbf{u}$ solves the system

$$A(\mathbf{u} + \Delta\mathbf{u}) = \mathbf{f} + \Delta\mathbf{f} \,. \tag{4.3}$$

Due to linearity, the perturbation $\Delta\mathbf{u}$ then solves the system

$$A\Delta\mathbf{u} = \Delta\mathbf{f} \,, \tag{4.4}$$

from which $\Delta\mathbf{u} = A^{-1}\Delta\mathbf{f}$ (formally!) and therefore $\|\Delta\mathbf{u}\| \leq \|A^{-1}\| \, \|\Delta\mathbf{f}\|$. From (4.1) follows that $\|\mathbf{f}\| \leq \|A\| \, \|\mathbf{u}\|$ and therefore

$$\frac{1}{\|\mathbf{u}\|} \leq \|A\| \, \frac{1}{\|\mathbf{f}\|} \tag{4.5}$$

Combining the inequalities we arrive at the following relative bound on the norm of the perturbed solution

$$\boxed{\frac{\|\Delta\mathbf{u}\|}{\|\mathbf{u}\|} \leq \|A^{-1}\| \, \|A\| \, \frac{\|\Delta\mathbf{f}\|}{\|\mathbf{f}\|} = \kappa(A) \frac{\|\Delta\mathbf{f}\|}{\|\mathbf{f}\|} \leq \delta \, \kappa(A) \,,} \tag{4.6}$$

where $\kappa(A)$ denotes the condition number of A measured in the norm $\|\cdot\|$. We thus have demonstrated the following theorem

Theorem 4.3.1 *A perturbation of the form* (4.2) *to the linear system* (4.1) *that does not affect A will result in a perturbation in the solution with relative error bound given by* (4.6).

Assume that e.g. $\delta = 10^{-6}$ and that the matrix is poorly conditioned with $\kappa(A) = 10^6$. Then the theorem says that in case that a perturbation $\Delta\mathbf{f}$ that is small in norm (as bounded by a small δ) results in a perturbation $\Delta\mathbf{u}$ that is possibly very large (as bounded by $\delta \, \kappa(A) = 1$). Examples showing that the bound given in the above theorem is attained and therefore sharp can easily be constructed.

Example 4.3.1 *Consider the following linear system*

$$\begin{pmatrix} 1 & 1 \\ 1 & 0.999 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2 \\ 1.999 \end{pmatrix} .$$

The two rows of the system matrix A are nearly equal and A is therefore nearly rank-deficient. The eigenvalues smallest in size is therefore nearly zero. The condition number of the system matrix measured in 2-norm (cfr. Subsection 2.5.3) is therefore large, i.e., the matrix is badly conditioned. The exact solution to this system is $\mathbf{u} = \begin{pmatrix} 1 & 1 \end{pmatrix}^T$. *Suppose now that the right-hand vector is slightly changed to* $\mathbf{f} = \begin{pmatrix} 2 & 2 \end{pmatrix}^T$ *either due to measurement or rounding errors. The solution of the perturbed problem is then given by* $\mathbf{u} = \begin{pmatrix} 2 & 0 \end{pmatrix}^T$ *which is totally different from the solution of the original (unperturbed) system.*

4.3.2 Perturbation in both matrix and right-hand side

The aim of this subsection is to generalize the analysis of the previous subsection to the more general and practically more relevant case in which a perturbation affects both $\mathbf{f}$ and A. Given as above a submultiplicative norm $\|\cdot\|$ and $\delta > 0$, we assume perturbations of the form

$$\mathbf{f} \rightarrow \mathbf{f} + \Delta\mathbf{f} \text{ where } \|\Delta\mathbf{f}\| \leq \delta \|\mathbf{f}\| \tag{4.7a}$$

$$A \rightarrow A + \Delta A \text{ where } \|\Delta A\| \leq \delta \|A\| \tag{4.7b}$$

The perturbed solution $\mathbf{v} = \mathbf{u} + \Delta\mathbf{u}$ then solves the system

$$(A + \Delta A)\mathbf{v} = \mathbf{f} + \Delta\mathbf{f}\,. \tag{4.8}$$

To avoid the perturbed problem from becoming singular, we impose the condition that

$$\delta\,\kappa(A) = r < 1\,. \tag{4.9}$$

To verify that this condition indeed prevents $A + \Delta A$ from becoming singular, we first write this matrix as

$$A + \Delta A = A\big[I - (-A^{-1}\,\Delta A)\big]\,. \tag{4.10}$$

Theorem 2.7 2.7.1 then implies that

$$\rho(-A^{-1}\,\Delta A) \le \|-A^{-1}\,\Delta A\| \le \|A^{-1}\|\,\|\Delta A\| \le \delta\|A^{-1}\|\,\|A\| = \delta\,\kappa(A) = r < 1\,. \tag{4.11}$$

By Theorem 2.7.4 we find that the matrix $I - (-A^{-1}\,\Delta A)$ is therefore non-singular. The matrix $A + \Delta A$ is therefore non-singular (as a product of two non-singular matrices) if the condition (4.9) holds. Theorem 2.7.4 also implies that

$$\Big[I - (-A^{-1}\,\Delta A)\Big]^{-1} = \sum_{k=0}^{\infty}(-A^{-1}\,\Delta A)^k\,, \tag{4.12}$$

and therefore given that $\|A^{-1}\|\,\|\Delta A\| \le r$ we have that

$$\left\|\big[I - (-A^{-1}\,\Delta A)\big]^{-1}\right\| \le \sum_{k=0}^{\infty}\|A^{-1}\,\Delta A\|^k \le \frac{1}{1-r}\,. \tag{4.13}$$

Next we rewrite the perturbed system (4.8) as

$$\begin{aligned} \mathbf{v} &= \big[I - (-A^{-1}\,\Delta A)\big]^{-1}A^{-1}\big[\mathbf{f} + \Delta\mathbf{f}\big] \qquad (4.14)\\ &= \big[I - (-A^{-1}\,\Delta A)\big]^{-1}\big[A^{-1}\mathbf{f} + A^{-1}\Delta\mathbf{f}\big]\\ &= \big[I - (-A^{-1}\,\Delta A)\big]^{-1}\big[\mathbf{u} + A^{-1}\Delta\mathbf{f}\big]\,. \end{aligned}$$

Therefore using (4.13)) we obtain

$$\begin{aligned} \|\mathbf{v}\| &\le \left\|\big[I - (-A^{-1}\,\Delta A)\big]^{-1}\right\|\Big[\|\mathbf{u}\| + \|A^{-1}\|\|\Delta\mathbf{f}\|\Big] \qquad (4.15)\\ &\le \frac{1}{1-r}\Big[\|\mathbf{u}\| + \delta\|A^{-1}\|\|\mathbf{f}\|\Big]\\ &\le \frac{1}{1-r}\Big[\|\mathbf{u}\| + \delta\|A^{-1}\|\|A\|\|\mathbf{u}\|\Big]\,. \end{aligned}$$

Scaling by $\|\mathbf{u}\|$ we obtain that

$$\boxed{\frac{\|\mathbf{v}\|}{\|\mathbf{u}\|} \le \frac{1}{1-r}\Big[1 + \delta\|A^{-1}\|\|A\|\|\Big] = \frac{1+r}{1-r}\,.} \tag{4.16}$$

We collect the results obtained in the following lemma.

Lemma 4.3.1 *A perturbation of the form* (4.7) *to the linear system* (4.1) *satisfying the condition* (4.9) *is such that $A + \Delta A$ is non-singular and the the bound* (4.16) *holds.*

This lemma is used to demonstrate the following theorem that gives a bound on the norm of the relative error.

Theorem 4.3.1 *A perturbation of the form* (4.7) *to linear system* (4.1) *satisfying condition* (4.9) *is such that* $A + \Delta A$ *is non-singular and*

$$\boxed{\frac{\|\mathbf{u} - \mathbf{v}\|}{\|\mathbf{u}\|} \leq \frac{2\delta}{1-r}\kappa(A)\,.} \tag{4.17}$$

Proof. The perturbed system (4.8) can be rewritten as

$$\begin{aligned} A\mathbf{v} + \Delta A\mathbf{v} = \mathbf{f} + \Delta\mathbf{f} \quad &\Leftrightarrow \quad \mathbf{v} + A^{-1}\Delta A\mathbf{v} = A^{-1}\mathbf{f} + A^{-1}\Delta\mathbf{f} \\ &\Leftrightarrow \quad \mathbf{v} + A^{-1}\Delta A\mathbf{v} = \mathbf{u} + A^{-1}\Delta\mathbf{f}\,. \end{aligned} \tag{4.18}$$

Therefore

$$\mathbf{u} - \mathbf{v} = A^{-1}\Delta A\mathbf{v} - A^{-1}\Delta\mathbf{f}\,, \tag{4.19}$$

and therefore

$$\|\mathbf{u} - \mathbf{v}\| \leq \delta\|A^{-1}\|\|A\|\|\mathbf{v}\| + \delta\|A^{-1}\|\|\mathbf{f}\| \leq \delta\|A^{-1}\|\|A\|\|\mathbf{v}\| + \delta\|A^{-1}\|\|A\|\|\mathbf{u}\| \tag{4.20}$$

and so

$$\frac{\|\mathbf{u} - \mathbf{v}\|}{\|\mathbf{u}\|} \leq \delta\,\kappa(A)\frac{\|\mathbf{v}\|}{\|\mathbf{u}\|} + \delta\,\kappa(A) \leq \delta\,\kappa(A)\frac{1+r}{1-r} + \delta\,\kappa(A) = \frac{2\delta}{1-r}\kappa(A)\,. \tag{4.21}$$

□

The theorem states that in case that A is badly conditioned, a perturbation in the input that is small in norm causes a perturbation in the output that is not necessarily small. Indeed, assume that $\delta = 10^6$ and that A is badly conditioned with $\kappa(A) = .5 \cdot 10^6$ such that $r = \delta\,\kappa(A) = .5$ to satisfy condition (4.9). Then the upper bound $2\,\delta\,\kappa(A)/(1-r) = 2$ is too large to guarantee that $\|\mathbf{u} - \mathbf{v}\|/\|\mathbf{u}\|$ remains small. Practical computations show that perturbations that are large in norm or likely to occur in the case that A is badly conditioned. Compared to the situation in which only the right-hand side is perturbed, the upper bound is magnified by a term $2/(1-r)$. Observe again that the above analysis is **independent** of the method used to solve the linear system.

4.3.3 Diagonal Scaling and Conditioning

In practical computations it may appear that the coefficient matrix $A \in \mathbb{R}^{n\times n}$ of the linear system (4.1) has entries whose order order of magnitude varies on a wide scale. The conditioning of badly scaled matrices can be improved by scaling the rows of the matrix, .i.e., by multiplying the matrix A on the left with a diagonal matrix $D = \text{diag}[d_1,\, d_2,\, \ldots, d_n]$. If D is chosen such that each row of $D^{-1}\,A$ has approximately the same ∞-norm, then $\kappa_\infty(D^{-1}\,A)$ is smaller than $\kappa_\infty(A)$. The problem is in other words better conditioned. In more advanced scaling techniques [23, 10] the columns of A are scaled as well.

Example 4.3.2 *Consider the following linear system [23]*

$$\begin{pmatrix} 10 & 100,000 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 100,000 \\ 2 \end{pmatrix} \tag{4.22}$$

and its equivalent row-scaled variant

$$\begin{pmatrix} 0.0001 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \tag{4.23}$$

each solved using $\beta = 10$, $t = 3$ *arithmetic. Then the solutions* $\hat{x} = (0.00,\, 1.00)^T$ *and* $\hat{x} = (1.00,\, 1.00)^T$ *are respectively computed. Note that* $x = (1.0001\ldots,\, .9999\ldots)^T$ *is the exact solution.*

4.4 The Gaussian Elimination Method

Direct solution methods for the system of linear equations (4.1) rely on some form of Gaussian elimination and typically consist of two stages. In the first stage the coefficient matrix is factored into a product of two matrices L and U such that $A \in \mathbb{R}^{n\times n}$ can be written as

$$A = L\,U\,. \tag{4.24}$$

The matrices L and U are lower and upper triangular and the diagonal elements of L are set equal to one. In the second stage the factored form (4.24) is substituted into the linear system (4.1) to obtain the system $L\,U\mathbf{u} = \mathbf{f}$. The latter system is solved by a forward and backward triangular substitution. In the forward step the system

$$L\mathbf{y} = \mathbf{f} \tag{4.25}$$

for an auxiliary vector $\mathbf{y}$. In the backward step the system

$$U\mathbf{u} = \mathbf{y} \tag{4.26}$$

is solved for the unknown vector $\mathbf{u}$. We will refer to U (and L) as the upper (and lower) triangular form of A. Some references further subdivide the factorization stage in a symbolic pre-processing stage and an actual computation stage. This section is subdivided into four subsection. We subsequently discuss the existence and uniqueness of the LU-factorization, its computation, the forward and backward triangular solves and the computational cost of the direct solution method.

4.4.1 Existence and Uniqueness of the LU-Factorization

The factorization (4.24) defines n^2 equations for the n^2 non-specified elements of L and U. The LU-factorization of A can be proven to exist if A and all its principal submatrices are non-singular. A principal submatrix of A is obtained by removing from A rows and columns that have the same index. If the LU-factorization of a non-singular matrix A exists, the factors L and U can be proven to be unique by an argument by contraction.

4.4.2 Computing the LU-Factorization

The coefficient matrix A is brought into the factored form (4.24) by a sequence of row operations that brings A to upper triangular (or echelon) form. These row operations correspond to linear combinations of the scalar equations that do not modify the solution of the linear system. The process that brings A to upper triangular form is known as Gaussian elimination. In this process the columns of A are consecutively visited. The row operations that brings the k-th column of A below the diagonal to zero are collected into a single Gaussian transformation M_k. To bring A to upper triangular form, $n-1$ Gaussian transformation are required. More precisely we have that given $A^{(0)} = A$, a sequence of $n-1$ matrices $\{A^{(1)}, A^{(2)}, \ldots, A^{(n-1)}\}$ is computed such that

$$A^{(k)} = M_k\,A^{(k-1)} = M_k\,M_{k-1}\ldots M_2\,M_1 A \text{ for } 1 \le k \le n-1\,, \tag{4.27}$$

and $A^{(n-1)} = U$, where U is the upper triangular form of A. The first $k-1$ columns of $A^{(k-1)}$ are zero below the main diagonal. It can therefore be partitioned into

$$A^{(k-1)} = \begin{array}{c c} & \begin{array}{cc} k-1 & n-k+1 \end{array} \\ \begin{array}{c} k-1 \\ n-k+1 \end{array} & \begin{pmatrix} A_{11}^{(k-1)} & A_{12}^{(k-1)} \\ 0 & A_{22}^{(k-1)} \end{pmatrix} \end{array} \text{ for } 1 \le k \le n-1\,, \tag{4.28}$$

where the $(1,1)$-block $A_{11}^{(k-1)}$ is of size $(k-1)\times(k-1)$ and upper triangular. The $(2,2)$-block $A_{22}^{(k-1)}$ is of size $(n-k+1)\times(n-k+1)$ and can further be partitioned into

$$A_{22}^{(k-1)} = \begin{array}{c c} & \begin{array}{cc} 1 & n-k \end{array} \\ \begin{array}{c} 1 \\ n-k \end{array} & \begin{pmatrix} a_{k,k}^{(k-1)} & \mathbf{c}_k^T \\ \mathbf{b}_k & B^{(k-1)} \end{pmatrix} \end{array} \text{ for } 1 \le k \le n-1\,. \tag{4.29}$$

The diagonal element $a_{k,k}^{(k-1)}$ is called the k-th pivot element. The zeroth pivot element is thus the $(1,1)$-element of the original matrix A.

To give details on how the k-th Gaussian transformation M_k is constructed using $A^{(k-1)}$ as input, we make the following assumptions

- we only use row replacement operations to bring A to echelon form. In a row replacement operation a given row is replaced by the sum of that row and a scalar multiple of another row [38]. No row interchanges and no row scaling operations will be used. The use of the latter two row operations will be covered in the forthcoming sections.
- we assume that each of the $n-1$ Gaussian elimination steps can be applied in such a way that all the pivot element remains non-zero, i.e., $a_{k,k}^{(k-1)} \neq 0$ for each $1 \leq k \leq n-1$. In finite precision arithmetic a bound away from zero will be required. Techniques to treat the occurrence of zero pivot elements will be covered in the next sections.

Example 4.4.1 *The linear system*

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

is trivially easy to solve. The straightforward application of Gaussian elimination requires due care and to avoid the division by zero caused by the selection of the (1,1)-element of the coefficient matrix as pivot element. We will later discuss techniques to avoid zero pivot elements.

The Gaussian transformations M_k for $1 \leq k \leq n-1$ can be expressed as a rank-one modification of the identity matrix. We therefore introduce such modifications first. Given two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$, its outer product $\mathbf{w}\,\mathbf{v}^T$ that is an $n \times n$ matrix that in column form can be written as

$$\mathbf{w}\,\mathbf{v}^T = [v_1\,\mathbf{w}, v_2\,\mathbf{w}, \ldots, v_n\,\mathbf{w}]\,. \tag{4.30}$$

If in particular $\mathbf{v} = \mathbf{e}_k$ where $\mathbf{e}_k$ is the k-th unit vector, the outer product $\mathbf{w}\,\mathbf{e}_k^T$ is the zero matrix with the k-th column replaced by the vector $\mathbf{w}$. Given the linear dependence of its columns, the matrix defined by (4.30) clearly has rank equal to one. The matrix $I + \mathbf{w}\,\mathbf{v}^T$ is therefore called a rank-one modification of the identity matrix.

The matrix equality $A^{(k)} = M_k A^{(k-1)}$ can be written as n equalities of columns. The k-th of these equalities reads

$$A^{(k)}(:,k) = M_k A^{(k-1)}(:,k)\,. \tag{4.31}$$

Using the block partitioning of $A^{(k-1)}$ defined in (4.28) and (4.29), the k-th column of $A^{(k-1)}$ can be expressed as

$$A^{(k-1)}(:,k) = \left[(\underbrace{A_{12}^{(k-1)}(:,1)}_{k-1}, \underbrace{a_{k,k}^{(k-1)}}_{1}, \underbrace{\mathbf{b}_k}_{n-k})\right]^T \tag{4.32}$$

where $A_{12}^{(k-1)}(:,1)$ denotes the first column of $A_{12}^{(k-1)}$. In the following we define the k-th Gaussian transformation M_k as the rank-one modification of I that is such that its action on $A^{(k-1)}$ reduces the components of the k-th column of $A^{(k-1)}$ below the diagonal to zero, i.e.,

$$A^{(k)}(:,k) = M_k A^{(k-1)}(:,k) = M_k \begin{pmatrix} A_{12}^{(k-1)}(:,1) \\ a_{k,k}^{(k-1)} \\ \mathbf{b}_k \end{pmatrix} = \begin{pmatrix} A_{12}^{(k-1)}(:,1) \\ a_{k,k}^{(k-1)} \\ 0 \end{pmatrix}\,. \tag{4.33}$$

The action of M_k on $A^{(k-1)}$ is equivalent to performing row operations on last $n-k$ rows of $A^{(k-1)}$ and leaves the first k rows of $A^{(k-1)}$ unchanged. Each of the last $n-k$ rows is more specifically replaced by this row minus a scalar multiply of the $(n-k-1)$-st row. The first $k-1$ entries of the rows that are replaced by the action of M_k were brought to zero by the application of previous Gauss transformations. The Gauss transformation M_k are thus defined as follows.

Definition 4.4.1 *Assume that $k-1$ Gaussian transformation steps of the matrix $A \in \mathbb{R}^{n \times n}$ result in the matrix $A^{(k-1)} \in \mathbb{R}^{n \times n}$ that can be decomposed into blocks according to (4.28) and (4.29) with a pivot element $a_{k,k}^{(k-1)} \neq 0$. The k-th Gauss-vector $\boldsymbol{\alpha}^{(k)} \in \mathbb{R}^n$ is defined as*

$$\boldsymbol{\alpha}^{(k)} = (\underbrace{0,\ldots,0}_{k}, \underbrace{\mathbf{b}_k / a_{k,k}^{(k-1)}}_{n-k}) \,. \tag{4.34}$$

The non-zero elements of $\boldsymbol{\alpha}^{(k)} \in \mathbb{R}^n$ are referred to as Gaussian multiplyers. The k-th Gaussian transformation $M_k \in \mathbb{R}^{n \times n}$ that satisfies (4.33) is defined as

$$M_k = I - \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T = \begin{pmatrix} & 0 & \\ & \vdots & \\ I(:,1:k-1) & 0 & I(:,k+1:n) \\ & 1 & \\ & -\mathbf{b}_k / a_{k,k}^{(k-1)} & \end{pmatrix} \,. \tag{4.35}$$

The application of M_k on $A^{(k-1)}$ amounts to $(n-k)$ row replacement operations in which the rows have length $(n-k+1)$. The action of M_k thus requires $2\,(n-k)(n-k+1)$ flops. The following gives the inverse of M_k.

Lemma 4.4.1 (Inverse of a Gauss Transformation) *The inverse of the Gauss transformation $M_k = I - \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T$ is the rank-one modification $M_k^{-1} = I + \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T$.*

Proof. A straightforward computation shows that $M_k\, M_k^{-1} = M_k^{-1}\, M_k = I$. □
The next lemma computes the product on the inverse of the $n-1$ Gaussian transformation matrices.

Lemma 4.4.2 (Product of Inverse of a Gauss Transformations) *We have that*

$$(M_{n-1} \cdot \ldots \cdot M_1)^{-1} = M_1^{-1} \ldots M_{n-1}^{-1} = \prod_{k=1}^{n-1} \left(I + \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T\right) = \sum_{k=1}^{n-1} I + \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T \,. \tag{4.36}$$

Proof. The result follows from the fact that for $1 \leq k \leq n-1$ we have that

$$\left(\boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T\right) \left(\boldsymbol{\alpha}^{(k+1)} \mathbf{e}_{k+1}^T\right) = \boldsymbol{\alpha}^{(k)} \left(\mathbf{e}_k^T\, \boldsymbol{\alpha}^{(k+1)}\right) \mathbf{e}_{k+1}^T = \boldsymbol{\alpha}^{(k)} \left(0\right) \mathbf{e}_{k+1}^T = 0 \,.$$

The ordering of the factors M_k thus matters. □
If we now define the matrix L as

$$L = (M_{n-1} \ldots M_1)^{-1} = \sum_{k=1}^{n-1} I + \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T \,, \tag{4.37}$$

then L is lower triangular with unit diagonal, i.e., diag$(L) = I$, and L contains all the Gaussian multiplyers below its main diagonal. In the above sum the matrix L is computed column-wise. We now have all the ingredients to formulate the main result of this section.

Theorem 4.4.1 *Assume that the coefficient matrix $A \in \mathbb{R}^{n \times n}$ of the linear system* (4.1) *is non-singular and that it can be brought to its upper triangular form U using $n-1$ row operations without scaling and without interchanges in such a way that pivot elements $a_{k,k}^{(k-1)}$ for $k = 1, \ldots, n-1$ remain non-zero. Then $n-1$ Gaussian transformation M_k for $k = 1, \ldots, n-1$ exist such that*

$$\begin{aligned} M_{n-1} M_{n-2} \ldots M_1 A = U \quad &\Leftrightarrow \quad A = (M_{n-1} \ldots M_1)^{-1}\, U \\ &\Leftrightarrow \quad A = L\, U \,. \end{aligned} \tag{4.38}$$

The theorem states that A can be factored into a lower triangular matrix L times a lower triangular matrix U. The fact that diag$(L) = I$ makes storing diag(L) redundant and allows to overwrite in a practical implementation the upper and lower triangular part of A by U and L, respectively. A prototype of such an implementation is by Algorithm 1.

After the application of $n-1$ Gaussian transformation, the $n-1$ pivot elements $a_{k,k}^{(k-1)}$ for $1 \leq k \leq n-1$ populate the $n-1$ first entries of the diagonal of U. The determinant of A can thus be computed as

$$\det(A) = \det(L\,U) = \det(L)\ \det(U) = 1\det(U) = u_{nn}\prod_{k=1}^{n-1} a_{k,k}^{(k-1)}\,, \tag{4.39}$$

where we used the fact that the determinant of a triangular matrix is given by the products of its diagonal elements. The matrix A can thus seen to be (close to) singular in case that one of the pivot elements $a_{k,k}^{(k-1)}$ is (close to) zero. The condition on the non-singularity of A thus guarantees that non of the pivot elements are zero in finite precision arithmetic. The LU-factorization of A can be modified to include information on the rank of A.

The factored form of A renders the linear system easy to solve. This is discussed in the next subsection.

Algorithm 1 Matrix LU Factorization Step

```
for $k = 1 \to n-1$ do
  if $A_{kk} \neq 0$ then
    quit {breakdown due to zero pivot}
  else
    for $i = k+1 \to n$ do
      $L(i,k) \leftarrow A(i,k)/A(k,k)$
      $A(i,k) \leftarrow L(i,k)$
      for $j = k+1 \to n$ do
        $A(i,j) \leftarrow A(i,j) - L(i,k)A(k,j)$
      end for
    end for
  end if
end for
```

4.4.3 Forward and Backward Substitution

- is dividing by the diagonal of L required?

Once the system matrix A is brought to factored form (4.24), the linear system (4.1) can easily be solved by a forward followed by a backward linear solve. The forward solve determines the auxiliary vector $\mathbf{y}$ from the linear system (4.25). The backward solve determines the solution vector $\mathbf{u}$ from the linear system (4.26). Prototype implementations are given in Algorithm 2 and Algorithm 3, respectively.

Algorithm 2 Forward Substitution Step

```
for $i = 1 \to n$ do
  $y(i) \leftarrow [f(i) - L(i, 1:i-1) \cdot f(1:i-1)]/L(i,i)$
end for
```

Algorithm 3 Backward Substitution Step

```
for $i = n \to i$ do
  $u(i) \leftarrow [y(i) - L(i, i+1:n) \cdot y(i+1:n)]/U(i,i)$
end for
```

4.4.4 Computational Cost

The computational cost of the LU direct solution method is given by the sum of the cost of the factorization and of the triangular solves.

Computational Cost of Factorization The computational cost of the LU-factorization can be computed by adding the computational cost of the $n-1$ Gaussian transformations. This results in a total computational cost equal to

$$2\sum_{k=1}^{n-1}(n-k)(n-k-1) = 2\sum_{\ell=1}^{n-1}\ell(\ell-1) = \frac{2}{3}n^3 + \mathcal{O}(n^2) \text{ flops}\,. \tag{4.40}$$

Computational Cost of Forward and Backward Substitution In the forward (or backward) substitution the computation of the i-th component of the solution vector requires the inner product of two vectors of length i (or $n-i$) and followed by a scaling (division by the diagonal element of L or U). This requires $2\,i+1$ (or $2(n-i)+1$) flops. The totals cost of both the forward and backward solve is thus given by

$$\sum_{i=1}^{n} 2i+1 = n^2 + \mathcal{O}(n) \text{ flops}\,. \tag{4.41}$$

The computational cost of the direct solution method is clearly dominated by the cost of the factorization process.

4.5 Gaussian Elimination in the Presence of Round-Off Errors

In this section we discuss the effect of finite precision arithmetic on both stages of the Gaussian elimination process. In the process we will reconsider the assumptions we made in Subsection 4.4.2 on the absence of row scaling and row interchange operations and on the boundedness away from zero of the pivot elements in the computation of the Gaussian transformations. This section is subdivided into two subsections. In the first and second subsection we discuss the effect of round-off errors in the solve and the factorization stage, respectively.

4.5.1 Solve Stage

The solve stage amounts to consequetively solving the linear systems (4.25) $L\mathbf{y} = \mathbf{f}$ for $\mathbf{y}$ and (4.26) $U\mathbf{u} = \mathbf{y}$ for $\mathbf{u}$. It can be shown (see e.g. [10, 23]) that in solving $L\mathbf{y} = \mathbf{f}$ the solution vector $\hat{\mathbf{y}}$ computed in the presence of round-off solves the system

$$(L+F)\hat{\mathbf{y}} = \mathbf{f} \tag{4.42}$$

where $F \in \mathbb{R}^{n\times n}$ is a perturbation of the matrix L that can be bounded element wise by

$$|F| \le n\,\mu|L| + \mathcal{O}(\mu^2) \tag{4.43}$$

where as before μ denotes the machine precision. On modern computing platform using double precision $\mu = .5\cdot 10^{-15}$ it is save to assume that $n\,\mu \le 0.1$. The above bound implies that each entry in F is small relative to the entries in L and that the forward triangular solve is numerically stable. Using a similar argument it can be shown that in solving $U\mathbf{u} = \mathbf{y}$ for $\mathbf{u}$ the computed solution solves the perturbed system

$$(U+F)\hat{\mathbf{u}} = \mathbf{y} \text{ where } |F| \le n\,\mu|U| + \mathcal{O}(\mu^2) \tag{4.44}$$

The perturbations are small in size and the algorithm is therefore stable.

4.5.2 Factorization Stage

The factorization stage amounts to computing the factors L and U such that $A = LU$. It can be shown (see again e.g. [10, 23]) that the factors $\widehat{L}$ and $\widehat{U}$ computed using the Gaussian elimanation procedure described in Subsection 4.4.2 satisfy the relation

$$\widehat{L}\,\widehat{U} = A + E \text{ where } \|E\| \leq n\,\mu \| \, |\widehat{L}| \, \| \, \| \, |\widehat{U}| \, \| \, . \tag{4.45}$$

The upper bound becomes large in the case that $\widehat{L}$ has elements that are large in size. In this case the norm of the perturbation E can become large without violating the upper bound. To illustrate what this means, we will consider the next example

Example 4.5.1 *Consider computing the LU-factorization of the matrix*

$$A = \begin{pmatrix} 0.0001 & 1 \\ 1 & 1 \end{pmatrix} . \tag{4.46}$$

The first pivot element $a_{1,1}^{(0)} = 0.0001$ *is small. The application of the first Gaussian transformation* M_1 *results in*

$$\widehat{U} = A^{(1)} = M_1 A^{(0)} = M_1 A = \begin{pmatrix} 1 & 0 \\ -10^4 & 1 \end{pmatrix} \begin{pmatrix} 0.0001 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 0.0001 & 1 \\ 0 & 1 - 10^4 \end{pmatrix} , \tag{4.47}$$

and therefore

$$\widehat{L} = \begin{pmatrix} 1 & 0 \\ 10^4 & 1 \end{pmatrix} \text{ and } \widehat{U} \approx \begin{pmatrix} 0.0001 & 1 \\ 0 & -10^4 \end{pmatrix} . \tag{4.48}$$

The product $\widehat{L}\widehat{U}$ *is equal to*

$$\widehat{L}\widehat{U} = \begin{pmatrix} 1 & 0 \\ 10^4 & 1 \end{pmatrix} \begin{pmatrix} 0.0001 & 1 \\ 0 & -10^4 \end{pmatrix} = \begin{pmatrix} 0.0001 & 1 \\ 1 & 0 \end{pmatrix} , \tag{4.49}$$

which is different from A *in the* $(2,2)$*-element. The 1-norm of the perturbation*

$$E = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \tag{4.50}$$

is equal to $\|E\|_1 = 1$ *which is of the same order of magnitude as* $\|A\|_1 = 2$*. The* $(1,2)$*-element of L is equal to* 10^4 *and its size results in a large upper bound for* $\|E\|$*.*

The bound for the size of the perturbation (4.45) allows for large perturbations E in case the elements in $\widehat{L}$ become large as in the previous example. The LU-factorization process in said to be *unstable*. The Gaussian elimination process can however be adapted to avoid small pivot elements and large entries in L by resorting to (partial) pivoting as will be discussed in the next section.

4.6 Gaussian Elimination with Pivoting

In the previous section we saw that the upper bound on the error of the LU-factorization of $A \in \mathbb{R}^{n \times n}$ becomes large due to large entries in L. In this section we will discuss how the growth in the elements of L can be avoided by resorting to a technique called pivoting. This section is subdivided into two subsections. In the first and second subsection we will partial and complete pivoting, respectively.

4.6.1 Gaussian Elimination with Partial Pivoting

Let as before the matrix $A^{(k-1)}$ and M_k denote the matrix obtained after $k-1$ stages of the Gaussian elimination process and the k-th Gaussian transformation, respectively. Let $A^{(k-1)}(:,k)$ be the k-th column of $A^{(k-1)}$ as in Equation (4.32). In Gaussian elimination with partial pivoting a row interchange is applied to $A^{(k-1)}$ prior to the application of M_k aiming at avoiding small pivot elements. More precisely, the following two steps are taken:

1. first identify the entry largest in size in the last $n-k+1$ rows of the k-th column of $A^{(k-1)}$, i.e., in the vector
$$A^{(k-1)}(k:n,k) \in \mathbb{R}^{n-k+1} \,. \tag{4.51}$$
Let ℓ denote the row index in which this largest element appears;
2. next interchange rows k and ℓ by multiplying $A^{(k-1)}$ to the left with a permutation matrix denoted by P_k. After this row interchange the element largest in size in the k-th column of $P_k A^{(k-1)}$ appears on the k-th position.

After this row interchange one proceeds as before by computing the Gauss vector $\boldsymbol{\alpha}^{(k)}$ and applying the Gaussian transformation M_k to $P_k A^{(k-1)}$. The difference however is that pivoting guarantees all the elements of $\boldsymbol{\alpha}^{(k)}$ to be bounded to 1. After $n-1$ Gaussian transformations all the elements of L are bounded by 1 avoiding the afore mentioned bound in the growth in the upper bound of the perturbation in the factorization. The algorithm is referred to as Gaussian Elimination with Partial Pivoting (GEPP).

It appears that Gaussian elimination with partial pivoting results in the LU-factorization of a row-permuted matrix A. This is formalized in the following theorem.

Theorem 4.6.1 *If Gaussian elimination with partial pivoting (GEPP) is used to compute the upper triangularization*
$$M_{n-1}P_{n-1}\dots M_1P_1A = U \,, \tag{4.52}$$
then
$$PA = LU \,, \tag{4.53}$$
where $P = P_{n-1}\dots P_1$ and L is a unit lower triangular matrix with $|l_{ij}| \le 1$.

Example 4.6.1 *Consider computing the LU-factorization with partial pivoting of the matrix*
$$A = \begin{pmatrix} 0.0001 & 1 \\ 1 & 1 \end{pmatrix} . \tag{4.54}$$
Permuting the first and second row of this matrix results in
$$P_1\,A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0.0001 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0.0001 & 1 \end{pmatrix} \tag{4.55}$$
The first pivot element equals 1. The application of the first Gaussian transformation M_1 results in
$$\widehat{U} = A^{(1)} = M_1P_1A = \begin{pmatrix} 1 & 0 \\ -0.0001 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0.0001 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 0.9999 \end{pmatrix} , \tag{4.56}$$
and therefore
$$\widehat{L} = \begin{pmatrix} 1 & 0 \\ 0.0001 & 1 \end{pmatrix} . \tag{4.57}$$
The product $\widehat{L}\widehat{U}$ is equal to
$$\widehat{L}\widehat{U} = \begin{pmatrix} 1 & 0 \\ 0.0001 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 0.9999 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0.0001 & 1 \end{pmatrix} , \tag{4.58}$$
which is exactly equal to $P_1\,A$. The use of partial pivoting prevented large elements to appear in L and the algorithm to become unstable.

To quantify the effect of round-off errors in finite arithmetic computations of GEPP, we will use the *growth factor* defined as follows

Definition 4.6.1 *The growth factor ρ of the Gaussian elimination of the matrix $A \in \mathbb{R}^{n\times n}$ is defined as the ratio*
$$\rho = \frac{\max\{\alpha, \alpha_1, \dots, \alpha_{n-1}\}}{\alpha} \,, \tag{4.59}$$
where $\alpha = \max_{i,j} |a_{ij}|$ and $\alpha_k = \max_{i,j} |a_{ij}^{(k)}|$.

The term *growth factor* derives from the fact that from this definition immediately follows that $|u_{ij}| = |a_{ij}^{n-1}| \leq \rho \max_{i,j} |a_{ij}|$. with partial pivoting we have that

$$\widehat{L}\,\widehat{U} = A + E \text{ where } \|E\|_\infty \leq n^3\,\mu\,\rho\|A\|_\infty\,. \tag{4.60}$$

How does the growth factor ρ increase with the problem size n? Numerical experience has shown that the growth factor ρ remains bounded and that GEPP is a numerically stable algorithm. The additional overhead to be paid for this stability is the comparison of $n-k$ floating point numbers at each stage k of the Gaussian elimination process.

4.6.2 Gaussian Elimination with Complete Pivoting

Examples of $n \times n$ matrices for which the growth factor equals $\rho = 2^{n-1}$ can be constructed. To compute the LU-factorization of such matrices, Gaussian elimination with complete pivoting was developed to obtain better bounds in the growth factor. In Gaussian elimination with complete pivoting both two rows and two columns of $A^{(k-1)}$ are interchanged prior to the application of M_k. More precisely, the following two steps are taken:

1. first identify the entry largest in size in the $(2,2)$-subblock of $A^{(k-1)}$ denoted by $A_{22}^{(k-1)}$ in (4.28). Denote by ℓ_1 and ℓ_2 the row and column index in which this largest element appears;
2. next interchange rows k and ℓ_1 by multiplying $A^{(k-1)}$ to the left with a permutation matrix denoted by P_k and interchange columns k and ℓ_2 by multiplying $A^{(k-1)}$ to the left with a permutation matrix denoted by Q_k. After this row and column interchange, the entry largest in size in the $(2,2)$-subblock of $P_k\,A^{(k-1)}\,Q_k$ appears in the (k,k)-position.

Gaussian elimination with complete pivoting (GECP) results in a growth factor that is a slowly function of the problem size n. The overhead to be paid is the comparison of $(n-k)^2$ comparison of floating point numbers at each stage k of the elimination process.

4.7 Diagonally dominant matrices

In case that the system matrix A is column diagonally dominant as defined in Definition 2.8.2, Gaussian elimination *without* pivoting can be applied. This will turn out to be a *very attractive property* in practical computations as we will point out in the remainder of this section. To argue that pivoting is redundant, one proceeds by induction. Assume that A is column diagonally dominant. Then no row interchanges in computing the first Gaussian transformation M_0. Assume next the matrix $A^{(k-1)}$ is column diagonally dominant in such a way that the k-th Gaussian transformation can be computed and applied without the need for row permutations. Then it can be shown that the resulting matrix $A^{(k)} = M_k\,A^{(k-1)}$ is again column diagonally. For a proof we refer to [23, 10]. It appears that the growth factor ρ for column diagonally matrices is bounded by 2 and that therefore Gaussian elimination is a very stable algorithm for diagonally matrices.

4.8 Cholesky Decomposition

In this section we focus on solving system with coefficient matrices that are symmetric and positive definite (SPD). Looking into this problem is important as SPD matrices appear in a large variety of applications such as for instance the discretization of elliptic partial differential equations or the modeling of networks in terms of weighted graph Laplacians. In case that A is SPD, it can be shown that the LU-decomposition of $A \in \mathbb{R}^{n\times n}$ reduces to its so-called *Cholesky decomposition* of A, i.e., a decomposition in the form

$$A = C\,C^T\,, \tag{4.61}$$

where $C \in \mathbb{R}^{n\times n}$ is a lower triangular matrix. The fact that only a single lower triangular matrix C needs to be be computed obviously results in savings both in computational cost and in memory storage over the LU-decomposition. It furthermore appears that the SPD property guarantees numerical stability of the factorization algorithm. This section is subdivided into four subsections. We subsequently discuss the following aspects of the Cholesky decomposition: the existence and the uniqueness, its computation, its computational cost and its algorithmic form.

4.8.1 Existence and uniqueness

The factorization (4.61) defines $1/2\, n\,(n+1)$ equations for the $1/2\, n\,(n+1)$ martrix elements of C. It can be shown that the principal submatrices of an SPD matrix are non-singular. The condition for the existence of an LU-decomposition is thus met and the decomposition $A = L\,U$ is guaranteed to exist. The non-singularity of A implies that none of the pivot elements stored in the diagonal of U is zero. We therefore can write the diagonal matrix $D = \text{diag}(U)$ and the lower triangular matrix M such that $M^T = D^{-1}\,U$. The LU-decomposition can thus be written as $A = L\,D\,M^T$. It can subsequently be shown that if A is symmetric, then necessarily $M = L$ and the factorization can be written as $A = L\,D\,L^T$. The positive definiteness of A implies that the elements of D are positive (at least in finite precision arithmetic) allowing to write that $D = \sqrt{D}\,\sqrt{D}$. We therefore have argued that if A is SPD, A can be written as $A = L\,\sqrt{D}\,\sqrt{D}\,L^T = C\,C^T$ where $C = L\,\sqrt{D}$.

4.8.2 Computing the Choleski Factor

An algorithm to compute the entries of the matrix C can be derived by carrying out the matrix-matrix product $C\,C^T$ and equating the results to the matrix A. This results in the algorithm listed in Algorithm 4. This algorithm computes the matrix C column-wise and stores the result in the lower triangular part of A. Other algorithmic variants exist.

Algorithm 4 Cholesky Factorization Step

```
for $k = 1 \to n$ do
  $A(k,k) \leftarrow C(k,k) = \sqrt{A(k,k) - \sum_{j=1}^{k-1} C(k,j)^2}$
  for $i = k+1 \to n$ do
    $A(i,k) \leftarrow C(i,k) = \frac{1}{C(k,k)} \Big( A(i,k) - \sum_{j=1}^{k-1} C(i,j)C(k,j) \Big)$
  end for
end for
```

4.8.3 Work

The computational cost of the Cholesky factorization equals half the cost of the LU-factorization as is given by

$$\frac{1}{3}n^3 + \mathcal{O}(n^2) \text{ flops}\,. \tag{4.62}$$

4.8.4 Stability

The Cholesky decomposition can be viewed as a special case of the LU-decomposition in which at stage k the k-th column of the factor C is computed. A growth factor can be defined similarly as in Definition 4.6.1 can be defined. Given however the fact that for $1 \leq j \leq i \leq n$ we have that

$$c_{ij}^2 \leq \sum_{k=1}^{i} c_{ik}^2 = a_{ii} \tag{4.63}$$

The growth factor is bounded by one, and the Cholesky factorization is a stable numerical algorithm. It can in fact be shown that the computed solution $\hat{\mathbf{u}}$ satisfies $(A+E)\hat{\mathbf{u}} = \mathbf{f}$ with

$$\|E\|_2 \leq c_n \mu \|A\|_2 \tag{4.64}$$

where c_n is a small constant independent of one. Moreover, if $q_n \mu \kappa_2(A) \leq 1$, where q_n is another small constant, then the algorithm runs to completion, i.e., no square roots of negative numbers appear.

4.9 Band Matrices

In our discussion on direct solution methods we thus far we made no assumptions on the sparsity pattern of the system matrix A. The discretization of partial differential equations using finite difference, finite volume or finite element methods and the modeling of networks results in *sparse* matrices, i..e., matrices with only a limited number of non-zero elements per row. In a first approach to analyze Gaussian elimination applied to such matrices we assume the matrix A to have a band structure. A band matrix is a matrix is a sparse matrix whose non-zero entries are confined to a diagonal band. More formally, we can give the following definition.

Definition 4.9.1 *Given a square matrix $A \in \mathbb{R}^{n \times n}$, and given non-negative integer numbers $p \geq 0$ and $q \geq 0$, the matrix A is said to have a* lower bandwidth *p iff p is the smallest number such that $a_{ij} = 0$ whenever $i > j + p$. Similarly, the matrix A is said to have a* upper bandwidth *q iff q is the smallest number such that $a_{ij} = 0$ whenever $j > i + q$. If $p = 0$ ($q = 0$) A is upper (lower) triangular.*

Observe that the above definition still allows for zero elements to appear inside the band. In a computer implementation one typically takes advantage of matrix storage data structures that avoid having to store the zero elements outside the band.

Example 4.9.1 *The discretization of the Poisson equation on the unit square with Dirichlet boundary conditions using N elements in both coordinate directions without the elimination of boundary conditions results in a discrete operator $A \in \mathbb{R}^{n \times n}$ where $n = (N+1)^2$ with lower and upper bandwidth $p = q = (N+1) = \sqrt{n}$.*

In the following three sections we discuss the Gaussian elimination of a band matrix without pivoting, the fill-in occurring during the factorization process and Gaussian elimination with partial pivoting, respectively.

4.9.1 Gaussian Elimination without Pivoting

The banded structure of A results in a substantial reduction in work and memory storage in the Gaussian elimination procedure. It can be shown that if A is banded, and if Gaussian elimination without pivoting can be applied to A, i.e., $A = L\,U$, then L and U inherit the lower and upper bandwidth of A, respectively. This is formalized in the next theorem.

Theorem 4.9.1 *Suppose that $A \in \mathbb{R}^{n \times n}$ has a lower bandwidth p, an upper bandwith q, and an LU-factorization that can be computed without partial pivoting, i.e., $A = L\,U$. Then L has a lower bandwidth p and U has an upper bandwidth q.*

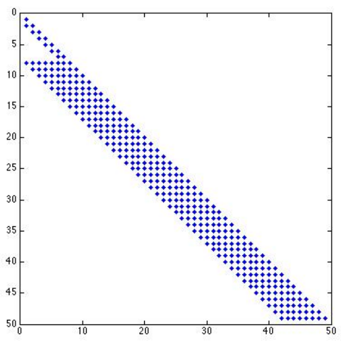
A proof of this theorem can be found in [23]. One can convince oneself of this result by writing down a few steps of the elimination process.

We recall here that if A is either SPD or diagonally dominant, the use of partial pivoting can be avoided. In discussing the required computational cost, we again distinguish between the factorization and solve stage. We furthermore assume that $n \gg p$ and $n \gg q$, allowing to neglect higher order terms in the asymptotic expressions of the flop count. It can be shown (see e.g. [23]) that the factorization stage requires $2\,p\,q\,n$ flops. The forward and backward triangular solve costs $2\,n\,p$ and $2\,n\,q$ flops, respectively.

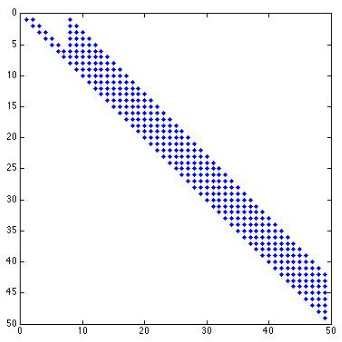
Example 4.9.2 *Consider again the discrete Laplace operator A as in Example 4.9.1. For this matrix Gaussian elimination without partial pivoting can be applied (explain why). The factorization stage costs $2\,p\,q\,n = 2\,n^2$ flops. Both the forward and backward triangular solve costs $2\,n^{3/2}$ flops.*

4.9.2 Occurrence of fill-in

The difficulty with Gaussian elimination or Cholesky factorization applied to the discrete Laplacian (or discretized elliptic partial differential equations in more general terms) is that they destroy zeros located inside the band. This is illustrated in Figure 4.1. In this figure we show the sparsity pattern of L (left) and U (right) obtained from the LU-factorization of the 2D discrete Laplacian A on a uniform mesh with $N = 8$ elements in both coordinate directions. The figure clearly shows that while A has zero diagonals inside its band, the lower band of L and the upper band of U are fully populated. These non-zero elements have to be computed and stored, causing a severe penalty in the deployment of Gaussian elimination or Cholesky factorization in large scale applications.



(a) Sparsity pattern of L



(b) Sparsity pattern of U

Figure 4.1: Sparsity pattern of the L and U factors resulting from the LU factorisation of the 5-point discrete Laplacian on a uniform mesh with 8 elements in both directions.

4.9.3 Gaussian elimination with partial pivoting

Gaussian elimination with partial pivoting can be tailored to exploit the band structure of A in the same way as Gaussian elimination without partial pivoting. However, if $P\,A = L\,U$, then the sparsity structure of L and U are not immediately obvious. The following theorem can be stated.

Theorem 4.9.2 *Suppose that $A \in \mathbb{R}^{n\times n}$ has a lower bandwidth p, an upper bandwith q and an LU-factorization that can be computed using partial pivoting, i.e., $P\,A = L\,U$. Then U has an upper bandwidth $p + q$. The k-th Gaussian transformation vector $\boldsymbol{\alpha}^{(k)}$ has at most p non-zero entries. More precisely, we have that $\boldsymbol{\alpha}_i^{(k)} = 0$ whenever $i \leq k$ or $k + p < i < n$.*

For a proof we again refer to [23].

Pivoting thus destroys the band of structure of A in the sense that U becomes larger than the upper triangle of A. Nothing at all can be said about the lower bandwidth of L. However, since the k-th column of L is a permutation of the k-th Gaussian transformation vector $\boldsymbol{\alpha}^{(k)}$, it follows that L has at most $p + 1$ non-zero entries per column. It is for this reason that partial pivoting is to be avoided as much ass possible in practical computations when solving banded systems .

4.10 General Sparse Matrices

In the case of a general sparse matrix A, substantial memory requirement reductions can be realized by storing only the non-zero entries. Storing the zero entries inside the band is thus avoided. Depending on the number and distribution of the non-zero entries, different data structures can be used and yield huge savings in memory when compared to the basic approach. The caveat is the accessing the individual elements becomes more complex and additional structures are needed to be able to recover the original matrix unambiguously.

In the following we discuss the Yale, the compressed sparse row and compressed sparse column matrix format. The former is the basis for the latter two.

4.10.1 Yale Format

The Yale sparse matrix format stores a given sparse $A \in \mathbb{R}^{n\times n}$ in row form using three (one-dimensional) arrays AA, IA and JA. Let nnz denote the number of nonzero entries in A. (Note that unlike in ordinary mathematics zero-based indices shall be used here.)

1. the real-valued array AA is of length nnz and holds all the nonzero entries of A in left-to-right top-to-bottom (*row-major*) order;

2. the integer array IA is of length $n+1$ and contains the index in A of the first element in each row, followed by the total number of nonzero elements $nnz+1$. $IA(i)$ contains the index in A of the first nonzero element of i-th row. Row i of the original matrix extends from $AA(IA(i))$ to $AA(IA(i+1)-1)$, i.e. from the start of one row to the last index before the start of the next. The last entry, $IA(n)$, must be the number of elements in AA plus one;

3. the third array, JA, contains the column index in A of each element of IA and hence is of length nnz.

Example 4.10.1 *Consider the* 4×4 *matrix*

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix} \tag{4.65}$$

with 4 *nonzero elements, hence*

$$\begin{aligned} AA &= \begin{pmatrix} 5 & 8 & 3 & 6 \end{pmatrix} \\ IA &= \begin{pmatrix} 0 & 1 & 3 & 4 & 5 \end{pmatrix} \\ JA &= \begin{pmatrix} 1 & 2 & 3 & 2 \end{pmatrix} . \end{aligned}$$

So, in array JA, the element "5" from A has column index 1*, "8" and "6" have index* 1*, and element "3" has index* 2*.*

Example 4.10.2 *Another example is the* 4×6 *matrix*

$$A = \begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix} \tag{4.66}$$

with 8 nonzero elements, so

$$\begin{aligned} AA &= \begin{pmatrix} 10 & 20 & 30 & 40 & 50 & 60 & 70 & 80 \end{pmatrix} \\ IA &= \begin{pmatrix} 1 & 3 & 5 & 8 & 9 \end{pmatrix} \\ JA &= \begin{pmatrix} 1 & 2 & 2 & 4 & 3 & 4 & 5 & 6 \end{pmatrix} . \end{aligned}$$

Here we have that

- *IA splits the array AA into rows:* $(10,20)(30,40)(50,60,70)(80)$*;*
- *JA aligns values in columns:* $(10,20,\ldots)(0,30,0,40,\ldots)(0,0,50,60,70,0)(0,0,0,0,0,80)$.

4.10.2 Compressed sparse row (CSR or CRS) format

CSR is effectively identical to the Yale Sparse Matrix format, except that the column array is normally stored ahead of the row index array. I.e., CSR is val, col_ind, row_ptr, where val is an array of the (left-to-right, then top-to-bottom) non-zero values of the matrix; col_ind is the column indices corresponding to the values; and, row_ptr is the list of value indexes where each row starts. This format is efficient for arithmetic operations, row slicing, and matrix-vector products.

4.10.3 Compressed sparse column (CSC or CCS) format

CSC is similar to CSR except that values are read first by column, a row index is stored for each value, and column pointers are stored. I.e. CSC is (val, row_ind, col_ptr), where val is an array of the (top-to-bottom, then left-to-right) non-zero values of the matrix; row_ind is the row indices corresponding to the values; and, col_ptr is the list of val indexes where each column starts. This format is efficient for arithmetic operations, column slicing, and matrix-vector products. This is the traditional format for specifying a sparse matrix in MATLAB (via the $sparse$ function).

4.10.4 Sparse Direct Solvers

The Gaussian elimination and Cholesky decomposition algorithms can be tailored to exploit sparsity to a larger extend than their counterpart for banded systems. This gives raise to sparse direct methods that are the subject of the monographs of e.g. Duff, Erisman and Reid [13] and of Timothy Davies [11]. Possibly these sparsity-exploiting methods need to be combined with reordering schemes that reduce the bandwidth of the matrix. These reordering schemes aim at reducing the amount of fill-in.

4.11 Tridiagonal Systems

Before closing this chapter, we discuss solving tridiagonal symmetric positive definite linear system. This discussion will serve as an ingredient in deriving the conjugate gradient method in a forthcoming chapter of Krylov subspace methods. We therefore assume $A = (a_{ij})$ to be a SPD tridiagonal matrix and derive an $L\,D\,L^T$ decomposition of this matrix. Assume $D = \text{diag}(d_1, \dots, d_n)$ and L to be of the form

$$L = \begin{pmatrix} 1 & & & \dots & 0 \\ e_1 & 1 & & & \vdots \\ & \ddots & \ddots & & \\ \vdots & & \ddots & & \\ 0 & \dots & & e_{n-1} & 1 \end{pmatrix} . \tag{4.67}$$

We deduce from the equation $A = L\,D\,L^T$ that

$$\begin{aligned} a_{11} &= d_{11} \\ a_{k,k-1} &= e_{k-1} d_{k-1} \text{ for } 2 \le k \le n \\ a_{kk} &= d_k + e_{k-1}^2 d_{k-1} = a_{kk} - e_{k-1} a_{kk-1} \text{ for } 2 \le k \le n\,. \end{aligned} \tag{4.68}$$

From these equations the number d_i and e_1 can be resolved.

4.12 Conclusions and Outlook

A summary of the important lessons to be drawn from this chapter can be found in Figure. The failure of direct solution methods applied to the sparse linear systems in shown in Figure 4.3. This failure motivates the study of iterative solution techniques in future chapters.

Direct solution of $A\,x = b$

Factorization stage	Solve stage
• $\mathcal{O}(n^3)$ operations • unstable without partial pivoting	• $\mathcal{O}(n^2)$ operations • stable

Effective for A small or dense.

Fails for A large and sparse.

Figure 4.2: A summary of direct solution methods.

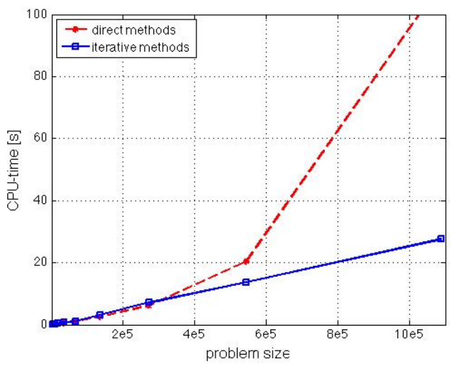


Figure 4.3: Performance in CPU time of direct and iterative solutions on a large scale problem.

4.13 Exercises

Exercise 4.13.1 Show that if $A \in \mathbb{R}^{n \times n}$ has an LU decomposition and is nonsingular, then L and U are unique.

Exercise 4.13.2 Show that for every nonsingular matrix A, partial pivoting leads to an LU decomposition of PA so: $PA = LU$.

Exercise 4.13.3 Show that if $A \in \mathbb{R}^{n \times n}$ has an LDM decomposition and is nonsingular, then L, D, and M are unique. Note that L is a lower triangular matrix with $l_{ii} = 1$, D is a diagonal matrix and M is an upper triangular matrix with $m_{ii} = 1$. (Remark: if you use the uniqueness of the LU decomposition, you should also show this)

Exercise 4.13.4 Suppose $A = LU$ with $L = (l_{ij}), U = (u_{ij})$ and $l_{ii} = 1$. Derive an algorithm to compute l_{ij} and u_{ij} by comparing the product LU with A.

Exercise 4.13.5 Suppose that A is symmetric and positive definite. Show that the matrix A_k consisting of the first k rows and columns of A is also symmetric and positive definite.

Exercise 4.13.6 Suppose that A is a symmetric and positive definite tridiagonal matrix. Give an algorithm to compute the LDL^T decomposition, where $l_{ii} = 1$.

Exercise 4.13.7 For a symmetric and positive definite matrix A, we define the numbers $f_i(A),\ i = 1, \ldots, n$ as follows:

$$f_i(A) = \min\{j | a_{ij} \neq 0\}.$$

Show that for the Cholesky decomposition $A = LL^T$ the equality $f_i(L) = f_i(A)$ holds for $i = 1, \ldots, n$.

Chapter 5

Basic Iterative Methods

`We iterate to exploit structure.`

5.1 Introduction

In this section we draw lessons from the pitfalls of direct solution methods and lay the basis for iterative solution methods (BIMs). In our presentation we follow the historical development of these methods. The methods introduced in this chapter are not efficient, but serve as a building block for more advanced methods (preconditioners in a Krylov subspace context and smoothers in a multigrid context).

Study Goals In this chapter we aim at

- introducing the concept of iteratively solving a system of linear equations
- giving the Richardson, (damped) Jacobi, Gauss-Seidel, SOR(ω) and symmetric SOR(ω) methods as examples of BIMs derived from splitting the coefficient matrix
- arguing how diagonal dominance, the M-matrix and consistent ordering property guarantees the convergence of the prototype BIMs introduced
- quantifying the speed of convergence of the BIMs if applied to the model problem and reveal the twice bad news story
- studying the convergence properties of SOR(ω) in more detail
- linking the concept of splitting to that of defect correction and of preconditioning

Classical references on this topic are the monographs of David Young and [76] and of Richard Varga [72]. Modern discussion can be found in among others the monographs of Stoer and Burlish [61], of Anne Greebaum [28], of Yousef Saad [50] and Jack Demmel [12]. A reference focussing on implementational aspects is [5].

5.2 Why Iterating?

The idea of iterating for solving a linear equations

$$A\mathbf{u} = \mathbf{f} \tag{5.1}$$

can be compared to shooting on a target with a crooked gun. After a first shot, the deviation from the target is measured and an improved shot is attempted for. The process is repeated untill the target is deemed to have been reached with sufficient precision. We denote the sequence of iterands by

$$\{\mathbf{u}^k\}_{k\geq 0} \text{ where } \mathbf{u}^k \to \mathbf{u}^* \text{ for } k \to \infty\,, \tag{5.2}$$

where $\mathbf{u}^0$ and $\mathbf{u}^*$ denotes the initial guess (first shot) and exact solution (target) of the linear system (5.1), respectively. With each iterand we associate the distance to the solution or *error* vector

$$\mathbf{e}^k = \mathbf{u}^* - \mathbf{u}^k \,. \qquad \text{[error]} \tag{5.3}$$

Knowing $\mathbf{e}^k$ at step k is equivalent to knowning $\mathbf{u}^*$ as $\mathbf{u}^k + \mathbf{e}^k = \mathbf{u}^*$. Computing the error is therefore as challenging as computing the exact solution. The *residual* vector at step k defined as

$$\mathbf{r}^k = \mathbf{f} - A\,\mathbf{u}^k \,. \qquad \text{[residual]} \tag{5.4}$$

and is a computable measure of the quality of the approximation at step k. By approximating the error by the residual we mean that the following *residual equation* holds

$$A\mathbf{e}^k = \mathbf{r}^k \,. \tag{5.5}$$

To construct an iterative scheme, we assume that a non-singular matrix M exists and define the matrix N as $N = M - A$. We can then write

$$A = M - N \,. \tag{5.6}$$

The linear system $A\mathbf{u} = \mathbf{f}$ can then be written as $M\mathbf{u} = N\mathbf{u} + \mathbf{f}$. By multiplying to the left and right by M^{-1} we can define an iterative scheme

$$\begin{aligned}
\mathbf{u}^{k+1} &= M^{-1}N\mathbf{u}^k + M^{-1}\mathbf{f} \\
&= M^{-1}(M-A)\mathbf{u}^k + M^{-1}\mathbf{f} \\
&= \mathbf{u}^k + M^{-1}(\mathbf{f} - A\mathbf{u}^k) \\
&= \mathbf{u}^k + M^{-1}\mathbf{r}^k
\end{aligned} \tag{5.7}$$

Each update $k \rightarrow k+1$ of the iterative scheme can be seen to requires

- a matrix-vector multiplication with the matrix A. If A is sparse (with on average c number of non-zeros per row), this operation requires $\mathcal{O}(cN)$ instead of $\mathcal{O}(N^2)$ flops for dense matrices. If A is Toeplitz, a fast matrix-vector multiplication can be performed via a discrete fast-Fourier transform. *Matrix-structure* motivates the use of iterative solution techniques.
- a linear system solve with the matrix M. It is therefore of paramount importance to make this operation as cheap as possible. From our study of direct solution techniques we know this to be the case if M is diagonal or triangular.

We will see in the next section that the matrix-vector multiplication with A and the linear system solve with M can be interleaved in such a way that the computational complexity (number of floating point operations) of the above iteration bringing $\mathbf{u}^k$ to $\mathbf{u}^{k+1}$ is roughly equivalent to the computational complexity of a matrix vector multiplication with the matrix A. The computational complexity of this matrix-vector multiplication is therefore typically used as unit (called work-unit and abbreviated as WU) to measure the cost of these schemes.

The recursion for the error vector is given by

$$\begin{aligned}
\mathbf{e}^{k+1} &= \mathbf{u}^* - \mathbf{u}^{k+1} \qquad (5.8) \\
&= \mathbf{u}^* - \mathbf{u}^k - M^{-1}\mathbf{r}^k \\
&= \mathbf{e}^k - M^{-1}A\,\mathbf{e}^k \\
&= (I - M^{-1}A)\mathbf{e}^k
\end{aligned}$$

and for the residual vector by

$$\begin{aligned}
\mathbf{r}^{k+1} &= \mathbf{f} - A\mathbf{u}^{k+1} \qquad (5.9) \\
&= \mathbf{f} - A\mathbf{u}^k - AM^{-1}\mathbf{r}^k \\
&= \mathbf{r}^k - AM^{-1}\,\mathbf{r}^k \\
&= (I - AM^{-1})\mathbf{r}^k
\end{aligned}$$

These recurrence relations imply that the vectors $\mathbf{e}^{k+1}$ and $\mathbf{r}^{k+1}$ satisfy (5.5) if $\mathbf{e}^k$ and $\mathbf{r}^k$ do. The matrices $I - M^{-1}A$ and $I - AM^{-1}$ are called the error and residual propagation matrix, respectively. They are related by

$$I - M^{-1}A = A^{-1}(I - AM^{-1})A \tag{5.10}$$

are therefore similar and have the same spectrum. This justifies using the (computable) residual norm to develop a *stopping criterium* for the iterative scheme. We will call the matrix (as is conventional practise)

$$B = I - M^{-1}A\,. \tag{5.11}$$

the *iteration matrix*. Iterative solution methods which can be completely characterized by one single matrix as above are called *stationary iterative* methods. Various families of stationary iterative solution methods correspond to choices for M. Examples include splittings, approximate factorizations and approximate inverses. Splittings will be introduced in the next section. The discussion on approximate factorizations is postponed to the study of preconditioners in subsequent chapters. Approximate inverse fall outside the scope of this lecture notes and are discussed in e.g. Section 10.5 of [52].

5.3 Prototypes of BIMs

To give specific examples of methods that fit into the abstract framework introduced above, we write the system matrix A as

$$A = D - E - F \in \mathbb{R}^{n\times n} \tag{5.12}$$

where D, $-E$ and $-F$ denote the diagonal, the strictly lower and the strictly upper triangular part of A, respectively, as shown in Figure . We also introduce the notation

$$L = D^{-1}E \text{ and } U = D^{-1}F\,. \tag{5.13}$$

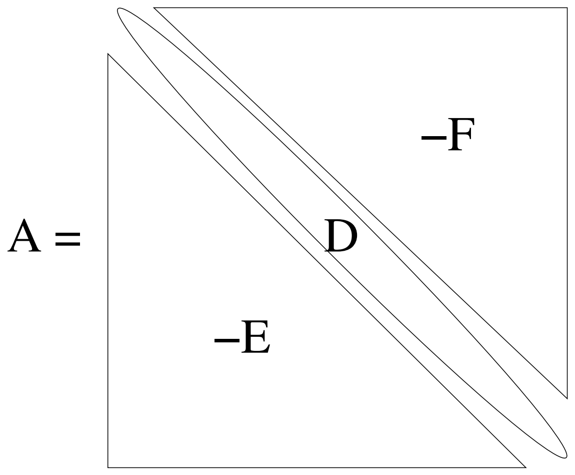


Figure 5.1: Splitting of the coefficient matrix A as $A = D - E - F$.

5.3.1 The Method of Jacobi and Gauss-Seidel

The method of Jacobi (named after Carl Gustav Jacob Jacobi 1804-1851) is obtained by setting

$$M_{JAC} = D \tag{5.14}$$

(and thus $N = E + F$) while the method of Gauss-Seidel (named after Carl Friedrich Gauss 1777-1855 and Philipp Ludwig von Seidel 1821-1896) is obtained by setting

$$M_{GS} = D - E \tag{5.15}$$

(and thus $N = F$). With these choices M is either diagonal or triangular and therefore easy to invert.

It is instructive to write the update $\mathbf{u}^{k+1} = M^{-1}(N\mathbf{u}^k + \mathbf{f})$ for both methods component wise. For the method of Jacobi we have that $\mathbf{u}^{k+1} = D^{-1}[(E+F)\mathbf{u}^k + \mathbf{f}]$ and therefore

$$u_i^{k+1} = \big[f_i - \sum_{j=1,j\neq i}^{n} a_{ij} u_j^k\big]/a_{ii} \quad \forall i = 1,\ldots,n \quad \text{[Jacobi]}\,. \tag{5.16}$$

For the method of Gauss-Seidel we have that $\mathbf{u}^{k+1} = (D-E)^{-1}(F\mathbf{u}^k + \mathbf{f})$ or $(D-E)\mathbf{u}^{k+1} = F\mathbf{u}^k + \mathbf{f}$ yielding $D\mathbf{u}^{k+1} = E\mathbf{u}^{k+1} + F\mathbf{u}^k + \mathbf{f}$ and therefore

$$u_i^{k+1} = \big[f_i - \sum_{j=1}^{i-1} a_{ij} u_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} u_j^k\big]/a_{ii} \quad \forall i = 1,\ldots,n \quad \text{[Gauss-Seidel]}\,. \tag{5.17}$$

From the above two formulas we observe that

- the Jacobi iteration allows to update all components of the iterand $\mathbf{u}^k$ independently from each other and is therefore inherently parallel. The result of the Gauss-Seidel iteration depends on the order in which the components of $\mathbf{u}^k$ are visited and is inherently sequential.
- the Gauss-Seidel iteration uses recent information as soon as it becomes available. We can therefore expect the the Gauss-Seidel iteration will converge faster the Jacobi. We will see later that for a specific class of problems Gauss-Seidel is twice as fast to converge than Jacobi.

The pseudo code for a single step of a Jacobi and Gauss-Seidel iteration is given in Algorithm 5 and Algorithm 6. These pseudo-codes show that the computational complexity of these algorithms is the same and equal to the computational complexity of one matrix-vector multiplication with the matrix A.

Algorithm 5 Single Step of a Jacobi Iteration

```
$z \leftarrow u$ { $z$ is an auxilary vector }
for $i = 1 \to n$ do
    $z(i) \leftarrow [f(i) - A(i,1:i-1) \cdot u(1:i-1) - A(i,i+1:n) \cdot u(i+1:n)]/A(i,i)$
end for
$u \leftarrow z$
```

Algorithm 6 Single Step of a Gauss-Seidel Iteration

```
for $i = 1 \to n$ do
    $u(i) \leftarrow [f(i) - A(i,1:i-1) \cdot u(1:i-1) - A(i,i+1:n) \cdot u(i+1:n)]/A(i,i)$
end for
```

Error propagation In the convergence analysis of these methods, we will use the expressions for the error propagation matrix $B = I - M^{-1}A$. For the method of Jacobi we have that

$$\begin{aligned} B_{JAC} &= I - D^{-1}(D - E - F) \\ &= I - I + D^{-1}E + D^{-1}F \\ &= L + U\,, \end{aligned} \tag{5.18}$$

while for the method of Gauss-Seidel we have that

$$\begin{aligned} B_{GS} &= I - (D-E)^{-1}(D-E-F) \\ &= I - I + (D-E)^{-1}F \\ &= (D-E)^{-1}DD^{-1}F \\ &= (I-L)^{-1}U\,. \end{aligned} \tag{5.19}$$

The following lemma will be useful in demonstrating that the Gauss-Seidel method converges for a wide class of matrices.

Lemma 5.3.1 (Elementwise bound on the Gauss-Seidel error propagation matrix) *Assume $A \in \mathbb{R}^{n\times n}$ and assume B_{GS} to be the Gauss-Seidel error propagation matrix defined by (5.19). Then*

$$|B_{GS}| \leq (I - |L|)^{-1}|U|\,. \tag{5.20}$$

Proof. Given that the lower triangular matrix L has a zero diagonal, its n-th (and all higher) power(s) is (are) equal to the zero matrix. The power series expansion of $(I-L)^{-1}$ given in Theorem 2.7.4 in Chapter 2 reduces to the finite sum

$$(I-L)^{-1} = I + L + L^2 + \ldots + L^{n-1}\,.$$

The same argument yields that

$$(I-|L|)^{-1} = I + |L| + |L|^2 + \ldots + |L|^{n-1}\,.$$

Therefore

$$|(I-L)^{-1}| \leq I + |L| + |L|^2 + \ldots + |L|^{1-1} = (I-|L|)^{-1}\,.$$

and thus

$$|B_{GS}| \leq |(I-L)^{-1}||U| \leq (I-|L|)^{-1}|U|$$

concluding the proof. □

Application to the Model Problem The above expressions can be made more explicit in case that the system matrix $A = A^h$ results from the discretization on a grid of mesh size h of the model problem **MP**-1 introduced in a previous chapter. In this case the Jacobi error propagation matrix $B = B^h = I^h - (D^h)^{-1}A^h$ can be represented by a stencil notation.

- For the one-dimensional problem we have that

$$B^h_{JAC} = [\ 1/2 \quad 0 \quad 1/2\ ]\,. \tag{5.21}$$

- For the two-dimensional problem we have that

$$B^h_{JAC} = \begin{bmatrix} 0 & 1/4 & 0 \\ 1/4 & 0 & 1/4 \\ 0 & 1/4 & 0 \end{bmatrix}\,. \tag{5.22}$$

The verification of these stencils is left as an exercise. As the error $\mathbf{e}^k$ satisfies the recursion $\mathbf{e}^{k+1} = B\mathbf{e}^k$, these stencils show that in a Jacobi iteration the error in each node is replaced by an average of its neighbours. We will return to this issue when explaining that the Jacobi (and Gauss-Seidel) method is slow to converge and motivating its multigrid acceleration.

Parallel Complexity and Orderings The fact that BIMs use the matrix-vector multiplication as a building block renders their implementation on parallel computing architectures particularly attractive. The parallel complexity of a method is defined as the number components of the iterand $\mathbf{u}^k$ that can be updated independently from each other. The parallel complexity of the Jacobi method is n (and thus optimal). So-called *block* variants of the Jacobi method exist in which groups of components of the iterand $\mathbf{u}^k$ are treated as aggregates. In the model problem **MP**-1, blocks can be formed by grouping components of $\mathbf{u}^k$ lying e.g. on a (vertical or horizontal) grid line. Alternatively, the computational domain can be subdivided into patches and aggregates formed according to these patches as shown in Figure 5.3.1. This gives raise to so-called domain decomposition methods studied in the monographs [57, 48, 62].

The Gauss-Seidel method introduced by (5.15) is referred to as the *forward* lexicografic Gauss-Seidel method to contrast is with the *backward* lexicografic Gauss-Seidel method defined by the splitting

$$M_{GS} = D - F \,. \tag{5.23}$$

Both have parallel complexity equal to 1. This can be improved by changing orderings in which the unknowns are visited without increasing the computational cost per iteration. In the model problem **MP**-1 the red-black and diagonal ordering of the unknowns results in a Gauss-Seidel method with parallel complex equal to $n/2$ and $\sqrt{n}$, respectively (explain!). Block variant of the Gauss-Seidel method have been used as well.

Block Jacobi and block Gauss-Seidel Method The Jacobi and Gauss-Seidel method discussed above act on single components of the iterand $\mathbf{u}^k$ at a time and are therefore also referred to as pointwise methods. It however possible to aggregate components into groups and to update members of a group simultaneously. This gives raise to the block variant of the Jacobi and Gauss-Seidel method. Assume that the system $A\,\mathbf{u} = \mathbf{f}$ is partitioned into the form

$$\begin{pmatrix} A_{1,1} & \dots & A_{1,q} \\ \vdots & & \vdots \\ A_{q,1} & \dots & A_{q,q} \end{pmatrix} \begin{pmatrix} U_1 \\ \vdots \\ U_q \end{pmatrix} = \begin{pmatrix} F_1 \\ \vdots \\ F_q \end{pmatrix} \tag{5.24}$$

where U_i and F_i are vectors of size n_i, where $A_{i,j}$ is a submatrix of size $n_i \times n_j$ and where $n_1 + \ldots + n_q = n$. This partitioning into blocks allows to define the splitting of the matrix

$$A = D_A - E_A - F_A \,, \tag{5.25}$$

where D_A, E_A and F_A are block diagonal, the block lower triangle part and block upper triangular part of A, respectively. With this splitting a block Jacobi and block Gauss-Seidel method can be defined.

The **block Jacobi method** updates the iterand $\mathbf{U}^k = (U_1^k, \ldots, U_q^k)$ using the formula

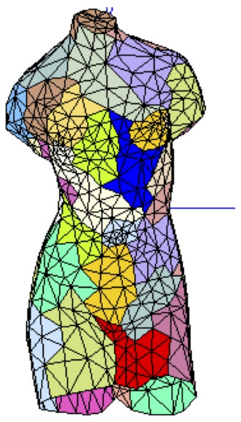
$$U_i^{k+1} = A_{i,i}^{-1} \left[F_i - \sum_{j=1, j\neq i}^{q} A_{i,j} U_j^k \right] \qquad \forall i = 1, \ldots, q \,. \tag{5.26}$$

The forward **block Gauss-Seidel method** updates the iterand $\mathbf{U}^k = (U_1^k, \ldots, U_q^k)$ according to

$$U_i^{k+1} = A_{i,i}^{-1} \left[F_i - \sum_{j=1}^{i-1} A_{i,j} U_j^{k+1} - \sum_{j=i+1}^{q} A_{i,j} U_j^k \right] \qquad \forall i = 1, \ldots, q \,. \tag{5.27}$$

Each iteration k of both methods requires a linear system solves with the matrices $A_{i,i}$ for $1 \leq i \leq q$. The partitioning is ideally chosen such that these operations are cheap to perform. If for example in the model problem **MP**-1 the nodes are numbered according to a x-lexicografic ordening and if the different blocks are chosen to correspond to a grid line parallel to the x-axis, then the diagonal blocks $A_{i,i}$ are tridiagonal. For tridiagonal systems cheap solution algorithms are available.

The rate of convergence of block methods is typically higher than that of their pointwise counterparts. In choosing the size of the blocks one typically takes into account the computational cost of one iteration and the overall rate of convergence.



Venus of Milo decomposed in subdomains
(from DD solvers on plates/shells, D.J. Rixen an P.Gosselet,
Gamni meeting, Paris, Mai 10 2004)

Figure 5.2: Examples of the use of block variants of BIMs.

5.3.2 The Richardson and damped Jacobi Method

Given a real-valued, non-zero parameter $\tau \neq 0$, the method of Richardson (named after Lewis Fry Richardson 1881-1953) is defined by the splitting $A = \tau I - (\tau I - A)$ and setting $M_{RICH} = \tau I$. We will show in the next section that the optimal value τ^* is the argument that minimizes the spectral radius of the error propagation matrix $B_{RICH} = I - \tau^{-1} A$, i.e.,

$$\tau^* = \text{argmin}_{\tau \in \mathbb{R}} \rho(B_{RICH(\tau)})\,. \tag{5.28}$$

In the *damped* Jacobi method a weighted average of the current iterand $\mathbf{u}^k$ and the full Jacobi step $\bar{\mathbf{u}}^{k+1,JAC}$ is computed. We denote the damping parameter by ω, and define the iterand resulting from the damped Jacobi method as

$$\mathbf{u}^{k+1} = (1-\omega)\,\mathbf{u}^k + \omega\,\bar{\mathbf{u}}^{k+1,JAC}\,. \tag{5.29}$$

In this definition $\omega = 1$ corresponds to the undamped Jacobi iteration. Substituting the expression (5.7) with $M = D$ for $\bar{\mathbf{u}}^{k+1}$, we obtain that

$$\begin{aligned} \mathbf{u}^{k+1} &= (1-\omega)\,\mathbf{u}^k + \omega\,\mathbf{u}^k + \omega\,D^{-1}\mathbf{r}^k \\ &= \mathbf{u}^k + \omega\,D^{-1}\mathbf{r}^k \end{aligned} \tag{5.30}$$

showing that the ω-damped Jacobi method is defined by

$$M_{JAC(\omega)} = \frac{1}{\omega} D \text{ and } B_{JAC(\omega)} = I - \omega D^{-1} A\,. \tag{5.31}$$

It is an easy exercise to show that the ω-damped Jacobi method corresponds to the Richardson method with parameter $\tau = \omega^{-1}$ applied to the diagonally scaled system $D^{-1}A = D^{-1}\mathbf{f}$.

Application to the Model Problem When applied to the model problem **MP**-1, the choice $\omega = 1$ appears to be optimal in terms of convergence. Other values of values of ω will become meaningful in a multigrid context.

5.3.3 The Successive Overrelaxation Method

Damping the Gauss-Seidel results in the method of Successive Overrelaxation (SOR). As the Gauss-Seidel updates the components of the iterand with the most recently available information, the damping is performed component wise as well. Denoting the components of components of the iterand computed using a full Gauss-Seidel step as $\bar{u}_i^{k+1,GS}$, damping results in

$$\begin{aligned} u_i^{k+1} &= (1-\omega)u_i^k + \omega \bar{u}_i^{k+1,GS} \\ &= (1-\omega)u_i^k + \omega\big[f_i - \sum_{j=1}^{i-1} a_{ij}u_j^{k+1} - \sum_{j=i+1}^{N} a_{ij}u_j^k\big]/a_{ii} \end{aligned} \tag{5.32}$$

or equivalently

$$a_{ii}u_i^{k+1} + \omega\sum_{j=1}^{i-1} a_{ij}u_j^{k+1} = (1-\omega)a_{ii}u_i^k - \omega\sum_{j=i+1}^{N} a_{ij}u_j^k + \omega f_i \tag{5.33}$$

which in matrix-vector form can be expressed as

$$(D-\omega E)\mathbf{u}^{k+1} = (1-\omega)D\mathbf{u}^k + \omega F\mathbf{u}^k + \omega\mathbf{f}\,. \tag{5.34}$$

This iterative scheme corresponds to the following splitting of ωA

$$\omega A = (D-\omega E) - ((1-\omega)D + \omega F) \tag{5.35}$$

showing that the SOR(ω) method is defined by

$$M_{SOR(\omega)} = \frac{1}{\omega}D - E \tag{5.36}$$

and

$$\begin{aligned} B_{SOR(\omega)} &= I - M_{SOR(\omega)}^{-1}A \\ &= I - \omega(D-\omega E)^{-1}A \\ &= I - \omega(I-\omega L)^{-1}D^{-1}(D-E-F) \\ &= I - (I-\omega L)^{-1}(\omega I - \omega L - \omega U) \\ &= I - (I-\omega L)^{-1}(I - \omega L + (\omega-1)I - \omega U) \\ &= (I-\omega L)^{-1}((1-\omega)I + \omega U) \end{aligned} \tag{5.37}$$

Observe that for $\omega = 1$ as expected, $M_{SOR(\omega)}$ and $B_{SOR(\omega)}$ coincide with M_{GS} and B_{GS}, respectively. A pseudo code for SOR(ω) is given in Algorithm 7, showing that the computational cost of one SOR(ω) iteration equals that of one matrix-vector product plus one vector update.

Algorithm 7 Single Step of a SOR(ω) Iteration

```
for $i = 1 \to n$ do
  $\sigma \leftarrow u(i)$ { $\sigma$ is an auxilary variable }
  $u(i) \leftarrow [f(i) - A(i,1:i-1)\cdot u(1:i-1) - A(i,i+1:n)\cdot u(i+1:n)]/A(i,i)$
  $u(i) \leftarrow (1-\omega)\sigma + \omega u(i)$
end for
```

The following lemma will be useful in analyzing the convergence properties of SOR(ω).

Lemma 5.3.2 (Characteristic Polynomial of SOR error propagation matrix) *Assume $A \in \mathbb{R}^{n\times n}$. Then the characteristic polynomials of the SOR(ω) error propagation matrix defined by (5.37) can be written as*

$$\varphi_{SOR(\omega)}(\lambda) = det\big[(1-\lambda-\omega)I + \omega U + \lambda\omega L\big]\,. \tag{5.38}$$

Proof. We have that

$$\begin{aligned}\varphi_{SOR(\omega)}(\lambda) &= \det\big[B_{SOR(\omega)} - \lambda I\big] \\ &= \det\big[(I-\omega L)^{-1}((1-\omega)I + \omega U) - \lambda I\big] \\ &= \det\big[(I-\omega L)^{-1}\big]\det\big[(1-\omega)I + \omega U - \lambda(I-\omega L)\big] \\ &= \frac{1}{\det\big[(I-\omega L)\big]}\det\big[(1-\lambda-\omega)I + \omega U + \lambda\omega L\big]\end{aligned}$$

The proof then follows from the fact that $I - \omega L$ is a lower-triangular matrix with ones on the diagonal. □
The following lemma immediately follows from the previous one and from the fact that the matrix U has a zero diagonal.

Lemma 5.3.3 (Characteristic Polynomial of SOR error propagation matrix) *Assume $A \in \mathbb{R}^{n\times n}$. Then zero is an eigenvalue of the Gauss-Seidel error propagation matrix, i.e., $0 \in \sigma(B_{GS})$.*

Symmetric SOR In a Krylov subspace context is will be advantageous to have the matrix defining the splitting to be SPD in case that A is SPD. It is an easy exercise to show that the Jacobi splitting results in such an M-matrix. The SOR method can be made to result in an SPD splitting by combining a M_1-forward and M_2-backward sweep by setting

$$M_1 = \frac{1}{\omega}D - L \text{ and } M_2 = \frac{1}{\omega}D - U\,. \tag{5.39}$$

and defining the composite step for the error vector $\mathbf{e}^k$

$$\begin{aligned}\mathbf{e}^{k+1} &= [I - M_2^{-1}A]\,[I - M_1^{-1}A]\,\mathbf{e}^k \\ &= [I - (M_1^{-1} + M_2^{-1} - M_2^{-1}A\,M_1^{-1})\,A]\,\mathbf{e}^k\end{aligned} \tag{5.40}$$

This iteration can be identified with (5.8) with the inverse of the splitting matrix defined by

$$\begin{aligned}M^{-1} &= M_1^{-1} + M_2^{-1} - M_2^{-1}A\,M_1^{-1} \\ &= M_2^{-1}[M_2 + M_1 - A]M_1^{-1} \\ &= \frac{2-\omega}{\omega}M_2^{-1}DM_1^{-1} \\ &= \omega(2-\omega)(D-\omega U)^{-1}D(D-\omega L)^{-1}\,.\end{aligned} \tag{5.41}$$

The resulting iteration is referred to as Symmetric SOR(ω) (SSOR(ω)) and has as iteration matrix

$$M_{SSOR(\omega)} = \frac{1}{\omega(2-\omega)}(D-\omega L)D^{-1}(D-\omega U)\,. \tag{5.42}$$

A pseudo code for a single SSOR(ω) iteration in given in Algorithm 8 showing that the computational cost of a single iteration equals that of two matrix vector multiplication plus two vector updates.

5.4 To Convergence or Not To Converge

In this section we discuss qualitative (yes/no) results on the convergence of BIMs. We start with a general result.

Algorithm 8 Single Step of a SSOR(ω) Iteration

```
for $i = 1 \to n$ do
    $\sigma \leftarrow u(i)$ { $\sigma$ is an auxilary variable }
    $u(i) \leftarrow [f(i) - A(i,1:i-1) \cdot u(1:i-1) - A(i,i+1:n) \cdot u(i+1:n)]/A(i,i)$
    $u(i) \leftarrow (1-\omega)\sigma + \omega u(i)$
end for
for $i = n \to 1$ do
    $\sigma \leftarrow u(i)$ { $\sigma$ is an auxilary variable }
    $u(i) \leftarrow [f(i) - A(i,1:i-1) \cdot u(1:i-1) - A(i,i+1:n) \cdot u(i+1:n)]/A(i,i)$
    $u(i) \leftarrow (1-\omega)\sigma + \omega u(i)$
end for
```

5.4.1 A General Result

Applying the expression (5.8) repeatedly, we obtain

$$\begin{aligned} \mathbf{e}^k &= (I - M^{-1}A)\mathbf{e}^{k-1} \\ &= (I - M^{-1}A)^2\mathbf{e}^{k-2} \\ &= \ldots \\ &= (I - M^{-1}A)^k\mathbf{e}^0 \\ &= B^k\mathbf{e}^0 \end{aligned} \tag{5.43}$$

stating that the error after k steps can be obtained by k application of the error propagation matrix. A single matrix thus fully characterizes the iterative scheme and the scheme is therefore called *stationary*. The recursion is similar to that of a discrete dynamical system. Necessary and sufficient conditions for the convergence of the scheme are given in the following theorem

Theorem 5.4.1

$$\boxed{\rho(B) = \rho(I - M^{-1}A) < 1 \Leftrightarrow \{\mathbf{u}^k\}_{k=1}^{\infty} \textit{ converges}}$$

Proof. This theorem is an immediate consequence of the Theorem 2.7.2. □

Apart from giving a condition on the convergence, the above theorem also quantifies the speed of convergence (number of iterations required to reach sufficient accuracy). Indeed, if $\rho(I - M^{-1}A) \ll 1$ (M is good approximation of A) then we can expect a fast convergence, while if $\rho(I - M^{-1}A) \approx 1$ (M is bad approximation if A) be can expect a slow convergence.

5.4.2 Matrix Norm Bounds

Theorem 2.7.1 states that any matrix norm is an upper bound on the spectral radius. For p-norm with $p = 1$, $p = 2$ or $p = \infty$, the matrix norms can be computed using (2.16), (2.17) or (2.18), respectively. For the Jacobi iteration, the $p = 1$, $p = \infty$ and Frobenius matrix norm bound translates into the following conditions for convergence

$$\|B_{JAC}\|_1 = \|L+U\|_1 = \max_{1 \le j \le n} \sum_{i=1, i \ne j}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1 \text{ [row sum criterium]} \tag{5.44}$$

$$\|B_{JAC}\|_\infty = \|L+U\|_\infty = \max_{1 \le i \le n} \sum_{j=1, j \ne i}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1 \text{ [column sum criterium]}\,. \tag{5.45}$$

$$\|B_{JAC}\|_F = \|L+U\|_F = \sum_{i,j=1, i \ne j}^{n} \big(\frac{a_{ij}}{a_{ii}}\big)^2 < 1 \text{ [square sum criterium]}\,. \tag{5.46}$$

These results are of practical importance as they show that the Jacobi method is faster to converge if more weight is placed on the diagonal. This occurs for instance when as parabolic PDE in advanced in time using an implicit time integration method.

5.4.3 Regular Splittings

In the following definition the notion of positivity plays a central role.

Definition 5.4.1 *The splitting $A = M - N$ is called* regular *iff M is non-singular, $M^{-1} \geq 0$ and $N \geq 0$.*

Example 5.4.1 *The Jacobi splitting of A^h in model problem* **MP**-*1 and* **MP**-*2 is obviously regular. Theorem 2.9.1 allows to establish that the matrix M in the Gauss-Seidel splitting of A^h in model problem* **MP**-*1 and* **MP**-*2 satisfies $M^{-1} \geq 0$ and that it is therefore defines a regular splitting. In model problem* **MP**-*3 and* **MP**-*4 nor the Jacobi neither the Gauss-Seidel define a regular splitting due to the negative entries in N.* `What happens in the case of SOR, IC and ILU?.`

The following theorem gives necessary conditions on A for a regular splitting to converge. The notion of positivity again plays a central role.

Theorem 5.4.1 *Assume $A = M - N$ to be a regular splitting. Then*

$$\rho(M^{-1}N) < 1 \Leftrightarrow A \textit{ is non-singular and } A^{-1} \textit{ is non-negative} \tag{5.47}$$

Proof. $\boxed{\Rightarrow}$ Let as before B denote the matrix $B = I - M^{-1}A = M^{-1}N$. Then $B \geq 0$ (as a product of non-negative matrices). By virtue of Theorem 2.10.2, the matrix $(I - B)$ is then non-singular and $(I - B)^{-1} \geq 0$. As

$$A = M - N = M(I - M^{-1}N) = M(I - B)\,, \tag{5.48}$$

we have that A is non-singular (as a product of non-singular matrices). Furthermore, as $A^{-1} = (I - M^{-1}N)^{-1}M^{-1} = (I - B)^{-1}M^{-1}$, we have that A^{-1} is non-negative (as a product of non-negative matrices).
$\boxed{\Leftarrow}$ From (5.48) follows that $(I - B)$ is non-singular. Both matrices $B = M^{-1}N$ and $A^{-1}N = (I - B)^{-1}B$ are positive (as a product of positive matrices). By virtue of the Perron-Frobenius Theorem 2.10.1, there exists a positive vector $\mathbf{u}$ such that $B\mathbf{u} = \rho(B)\mathbf{u}$. As $(I - B)^{-1} \geq 0$, we have that $(I - B)^{-1}B\mathbf{u} = \rho(B)(I - B)^{-1}\mathbf{u} = \frac{\rho(B)}{1-\rho(B)}\mathbf{u} \geq 0$. As $\mathbf{u} \geq 0$, the former can only be true is $\frac{\rho(B)}{1-\rho(B)} \geq 0$, which can only be true if $0 \leq \rho(B) \leq 1$. Since $I - B$ is non-singular, then $\rho(B) \neq 1$, which implies that $\rho(B) < 1$. $\square$

Example 5.4.2 *The convergence of Jacobi and Gauss-Seidel for in model problem* **MP**-*1 follows immediately from this theorem (why?). In model problem* **MP**-*2 the matrix A^h is singular, and the theorem does not apply. The theorem does not guarantee the convergence of Jacobi and Gauss-Seidel for* **MP**-*3 and* **MP**-*4. This issue is settled in the next section.* `What happens in the case of SOR, IC and ILU?`

5.4.4 Diagonal Dominance

Whereas in the previous subsection we discussed regular splittings of the coefficient matrix, in this subsection we will only consider the Jacobi and Gauss-Seidel methods. It appears that for these methods the condition of regularity of the splitting for convergence can be weakened to that of strong diagonal dominance of the coefficient matrix.

Theorem 5.4.1 *Assume $A \in \mathbb{R}^{n \times n}$ to be strongly row diagonally dominant. Then the Jacobi and Gauss-Seidel method applied to A converge, i.e.,*

$$\sum_{j=1, j\neq i}^{n} |a_{ij}| < |a_{ii}| \quad \forall i = 1, \ldots, n \Rightarrow \|B_{GS}\|_1 \leq \|B_{JAC}\|_1 < 1 \tag{5.49}$$

Proof. For the Jacobi method, the result follows from (5.44). For the Gauss-Seidel method we denote by $\mathbf{e} \in \mathbb{R}^n$ the vector having only 1 as components. Using Lemma 5.3.1, we have that

$$|B_{GS}|\mathbf{e} \leq (I - |L|)^{-1}|U|\mathbf{e}\,. \tag{5.50}$$

As the matrix $|B_{JAC}|$ is computed element-wise, we have that

$$|B_{JAC}| = |L| + |U|$$

and therefore $|U|\mathbf{e} = |B_{JAC}|\mathbf{e} + |L|\mathbf{e}$. The term $|B_{JAC}|\mathbf{e}$ can be bounded by the strict diagonal dominance of A. We indeed have that $|B_{JAC}|\mathbf{e} \leq \|B_{JAC}\|_1\mathbf{e}$, from which follows $|U|\mathbf{e} \leq [\|B_{JAC}\|_1 I - |L|]\,\mathbf{e}$. Substituting this in Equation (5.50), we obtain

$$\begin{aligned} |B_{GS}|\mathbf{e} &\leq (I - |L|)^{-1}\big[\|B_{JAC}\|_1 I - |L|\big]\mathbf{e} \\ &= (I - |L|)^{-1}\big[I - |L| + (\|B_{JAC}\|_1 - 1)I\big]\mathbf{e} \\ &= \big[I + (\|B_{JAC}\|_1 - 1)(I - |L|)^{-1}\big]\,\mathbf{e} \end{aligned}$$

We have already proven that $\|B_{JAC}\|_1 < 1$ and therefore $\|B_{JAC}\|_1 - 1 < 0$. Moreover, from $|L| \geq 0$ follows that $I - |L| \leq I$ and by taking the inverse of both sides that $(I - |L|)^{-1} \geq I$. Combining these two results we have that $I + (\|B_{JAC}\|_1 - 1)(I - |L|)^{-1} \leq I + (\|B_{JAC}\|_1 - 1)I = \|B_{JAC}\|_1 I$, from which

$$|B_{GS}|\mathbf{e} \leq \|B_{JAC}\|_1\mathbf{e}\,. \tag{5.51}$$

Proposition 2.5.3 then gives the desired result that $\|B_{GS}\|_1 \leq \|B_{JAC}\|_1$. $\square$

Example 5.4.3 *The matrix A^h in model problems* **MP**-*1,* **MP**-*3 and* **MP**-*4 is irreducably diagonally dominant, while is not in model problem* **MP**-*2 (as a strict inequality is missing) and nor in model problem* **MP**-*5 (due to the weight in the off-diagonals). The above theorem thus extends the proof of convergence of Jacobi and Gauss-Seidel from* **MP**-*1 to* **MP**-*3 and* **MP**-*4.*

5.4.5 Converge of SOR(ω)

In this subsection we discuss the convergence of the successive overrelaxation method. The following theorem states that the method does not converge unless the parameter ω lies in the range $0 < \omega < 2$.

Theorem 5.4.2 *For an arbitrary $n \times n$ matrix A holds that*

$$\rho(B_{SOR(\omega)}) \geq |1 - \omega|$$

Proof. The constant term $\varphi(0)$ in the characteristic polynomial $\varphi(\lambda)$ of $B_{SOR(\omega)}$ is the product of its eigenvalues, i.e,. by Lemma 5.3.2

$$\Pi_{i=1}^{n}\lambda_i(B_{SOR(\omega)}) = \varphi(0) = \det[(1-\omega)I + \omega U] = (1-\omega)^n\,. \tag{5.52}$$

Hence the result that $\rho(B_{SOR(\omega)}) = \max_i |\lambda_i(B_{SOR(\omega)})| \geq |1 - \omega|\,.$ $\square$
The convergence of SOR(ω) for the model problems **MP**-1 is a colorary of the following theorem.

Theorem 5.4.3 *Assume A to be SPD. Then SOR(ω) for $0 < \omega < 2$ applied to A converges.*

As proof of this theorem can be found in [61].

5.5 Speed Convergence of BIMs

The aim of this section is to derive quantitative results on the *speed* of convergence of BIMs applied to model problem **MP**-1. The term speed here refers to the required number of iterations to reach a prescribed accuracy. We in particular relate the speed of convergence to the meshwidth h used in the finite difference discretization.

5.5.1 A General Result

It appears that the asymptotic speed of convergence of the BIMs is *linear*. This type of speed of convergence is defined next.

Definition 5.5.1 *The sequence* $\{\mathbf{u}^k\}_{k\geq 0}$ *converges* linearly *to* $\mathbf{u}^*$ *in the norm* $\|\cdot\|$ *if there exists a number* $\mu \in (0,1)$ *such that for all* k

$$\|\mathbf{u}^k - \mathbf{u}^{k-1}\| \leq \mu \|\mathbf{u}^{k-1} - \mathbf{u}^{k-2}\| \tag{5.53}$$

The number μ *is called the* rate of convergence *or* reduction factor.

Applying this definition recursively yields that if the sequence $\{\mathbf{u}^k\}_{k\geq 0}$ converges linearly

$$\|\mathbf{u}^k - \mathbf{u}^{k-1}\| \leq \mu \|\mathbf{u}^{k-1} - \mathbf{u}^{k-2}\| \leq \ldots \leq \mu^{k-1}\|\mathbf{u}^1 - \mathbf{u}^0\|\,. \tag{5.54}$$

This allows to proof the following estimate for the error after k steps

Theorem 5.5.1 *If the sequence* $\{\mathbf{u}^k\}_{k\geq 0}$ *converges* linearly *to* $\mathbf{u}^*$ *in the norm* $\|\cdot\|$ *with rate of convergence* μ*, then the following estimate holds*

$$\|\mathbf{e}^k\|_2 = \|\mathbf{u}^k - \mathbf{u}^*\| \leq \frac{\mu^k}{1-\mu}\|\mathbf{u}^1 - \mathbf{u}^0\| = \frac{\mu^k}{1-\mu}\|\mathbf{e}^0\|\,. \tag{5.55}$$

Proof. If $\ell \geq k+1$ then

$$\begin{aligned}
\|\mathbf{u}^\ell - \mathbf{u}^k\| &= \|\mathbf{u}^\ell - \mathbf{u}^{\ell-1} + \mathbf{u}^{\ell-1} - \mathbf{u}^{\ell-2} + \ldots + \mathbf{u}^{k+1} - \mathbf{u}^k\| \\
&\leq \|\mathbf{u}^\ell - \mathbf{u}^{\ell-1}\| + \|\mathbf{u}^{\ell-1} - \mathbf{u}^{\ell-2}\| + \ldots + \|\mathbf{u}^{k+1} - \mathbf{u}^k\| \\
&\leq (\mu^{\ell-k} + \mu^{\ell-k-1} + \ldots + \mu)\|\mathbf{u}^k - \mathbf{u}^{k-1}\| \\
&= \mu(\mu^{\ell-k-1} + \mu^{\ell-k-2} + \ldots + 1)\|\mathbf{u}^k - \mathbf{u}^{k-1}\| \\
&= \mu\frac{1-\mu^{\ell-k}}{1-\mu}\|\mathbf{u}^k - \mathbf{u}^{k-1}\|\,.
\end{aligned}$$

Taking the limit of both sides of this inequality as $\ell \to \infty$ using the fact that the norm $\|\cdot\|$ is a continuous functional, we obtain

$$\begin{aligned}
\lim_{\ell\to\infty}\|\mathbf{u}^\ell - \mathbf{u}^k\| &= \|\lim_{\ell\to\infty}\mathbf{u}^\ell - \mathbf{u}^k\| \\
&= \|\mathbf{u}^* - \mathbf{u}^k\| \\
&\leq \lim_{\ell\to\infty}\mu\frac{1-\mu^{\ell-k}}{1-\mu}\|\mathbf{u}^k - \mathbf{u}^{k-1}\| \\
&= \frac{\mu}{1-\mu}\|\mathbf{u}^k - \mathbf{u}^{k-1}\| \\
&\leq \frac{\mu^k}{1-\mu}\|\mathbf{u}^1 - \mathbf{u}^0\|\,,
\end{aligned}$$

concluding the proof. □

Assuming that the equality holds in the estimate (5.55), and taking the $\log_{10}$ of both sides yields

$$\log_{10}\|\mathbf{e}^k\| = \log_{10}(\mu)k + \log_{10}(\frac{\|\mathbf{e}^0\|}{1-\mu}) \tag{5.56}$$

This implies that on a log-log scale the curve $\|\mathbf{e}^k\|$ as a function of k is a straight line with a negative slope given by $\log_{10}(\mu) < 0$. As small value of μ therefore give a fast convergence. Given $\|\mathbf{e}^0\|$, this result can be used to estimate the required number of iterations required to reach a particular accuracy.

To link this general result to the convergence of a BIM, we will make the following assumptions on the convergent error propagation matrix $B \in \mathbb{R}^{n\times n}$ ($\rho(B) < 1$) that fully characterizes the BIM. These assumptions are met by the Jacobi, Gauss-Seidel and SOR(ω) methods when applied to the discretized elliptic problems introduced earlier, as we will see for the Jacobi method later in this section. We assume that

- the spectral radius $\rho(B) < 1$ is large compared with the modulus of the $n-1$ remaining eigenvalues of B. We denote $\sigma(B) = \{\mu_k | k = 1, \ldots, n\}$ and number the eigenvalues such that $\rho(B) = |\mu_n|$. We thus have that $1 > |\mu_n| \gg |\mu_k|$ for $k = 1, \ldots, n-1$;
- B has n eigenvectors $\{\mathbf{v}^{[k]} | k = 1, \ldots, n\}$ that form an orthonormal basis of $\mathbb{R}^n$.

The argument that now follows will revisited when discussing the power method for eigenvalue computations and the smoothing property of BIMs. Assume that the initial error $\mathbf{e}_0$ corresponding to the initial guess $\mathbf{u}^0$ can be decomposed in the eigenbasis of B according to

$$\mathbf{e}^0 = \gamma_1 \mathbf{v}^{[1]} + \ldots + \gamma_n \mathbf{v}^{[n]} , \tag{5.57}$$

where $\gamma_\ell =< \mathbf{e}^0, \mathbf{u}^{[\ell]} > / < \mathbf{u}^{[\ell]}, \mathbf{u}^{[\ell]} >$. Substituting this initial error into (5.43) yields that after k steps

$$\begin{aligned} \mathbf{e}^k &= B^k \mathbf{e}^0 \\ &= \gamma_1 B^k \mathbf{v}^{[1]} + \ldots + \gamma_n B^k \mathbf{v}^{[n]} \\ &= \gamma_1 \mu_1^k \mathbf{v}^{[1]} + \ldots + \gamma_n \mu_n^k \mathbf{v}^{[n]} . \end{aligned}$$

Each of the n error contributions in span($\mathbf{v}^{[\ell]}$) is therefore damped with different factor μ_ℓ^k, the slowest one to converge being $\ell = n$. For k sufficiently large, as good approximation is therefore

$$\mathbf{e}^k = \gamma_n \mu_n^k \mathbf{v}^{[n]}$$

(as other contributions to this sum are negligible) and given that $\|\mathbf{v}^{[n]}\| = 1$, we have

$$\|\mathbf{e}^k\| = |\gamma_n| |\mu_n|^k = |\gamma_n| \rho(B)^k \Leftrightarrow \boxed{\log_{10} \|\mathbf{e}^k\| = \rho(B) k + \log_{10} \gamma_n}$$

which shows that $\rho(B)$ can be identified as the convergence factor of the linearly converging process. We can now state that

- while the condition $\rho(B) < 1$ assures convergence, the condition $\rho(B) \ll 1$ assures *fast* convergence;
- in practical problems $\rho(B)$ is close to 1 and increasing towards 1 with the problem size; this will be illustrated further in this section and implies that BIM as stand-alone methods are too slow to solve realistic engineering problem;
- the above observations do not change when monitoring the residual instead of the error norm.

5.5.2 Speed Convergence of Damped Jacobi

Next we quantify the speed of convergence of the damped Jacobi method.

Theorem 5.5.1 *Assume A^h to results from the discretization of the two-dimensional model problem* **MP-***1 on a grid of mesh size $h = \frac{1}{N}$ afterv elimination of the boundary conditions implying that A^h can be diagonalized according to $A^h = V^h \Lambda^h (V^h)^T$. Assume $B^h_{JAC(\omega)}$ to be the error propagation matrix resulting from a damped Jacobi splitting of A^h. Then $B^h_{JAC(\omega)} = V^h \Lambda^h_{JAC(\omega)} (V^h)^T$ where the diagonal entries of $\Lambda^h_{JAC(\omega)}$ are the eigenvalues of $B^h_{JAC(\omega)}$ given by*

$$\lambda_{k\ell}(B^h_{JAC(\omega)}) = 1 - \omega \left[1 - \frac{1}{2} \cos(\pi h k) - \frac{1}{2} \cos(\pi h \ell) \right] . \tag{5.58}$$

In the proof we exploit the fact that A^h has a constant diagonal.
Proof. If $v^{h,[k\ell]}$ is an eigenvector of A^h corresponding to the eigenvalue $\lambda_{k\ell}(A^h)$, the same $v^{h,[k\ell]}$ is a eigenvector of $B^h_{JAC(\omega)}$ corresponding to the eigenvalue $\lambda_{k\ell}(B^h_{JAC(\omega)}) = 1 - \omega \frac{h^2}{4} \lambda_{k\ell}(A^h)$. The result then follows for Theorem 3.7.2. □

The theorem can be used to investigate the speed convergence of Jacobi ($\omega = 1$) as a function of the meshwidth h. Indeed, we have have that the eigenvalues of the Jacobi method are given by

$$\lambda_{k\ell}(B^h_{JAC}) = \frac{1}{2}\cos(\pi h k) + \frac{1}{2}\cos(\pi h \ell)\,, \tag{5.59}$$

and thus the spectral radius by

$$\rho(B_{JAC}) = \lambda_{11}(B^h_{JAC}) = \cos(\pi h) = 1 - \frac{\pi^2}{2}h^2 + \mathcal{O}(h^4)\,. \tag{5.60}$$

This result gives us the first instance of the *twice bad news* story, namely that

- for moderate values of h, $\rho(B_{JAC}) \approx 1$, implying slow convergence. For $h = 1/64$ for instance, we that $\rho(B_{JAC}) = 0.998$;
- $\rho(B_{JAC}) \to 1$ as $h \to 0$, implying that convergence slows down as the mesh is refined to obtain more accurate discretizations.

The convergence of the Jacobi method on the model problem **MP**-1 is shown in Figure 5.3. This figure clearly shows how the convergence slows down on finer meshes. The fact that the eigenvectors of $B^h_{JAC(\omega)}$ form a basis for $\mathbb{R}^{n\times n}$ will be used in a more refined analysis of the convergence of damped Jacobi in the chapter on multigrid methods.

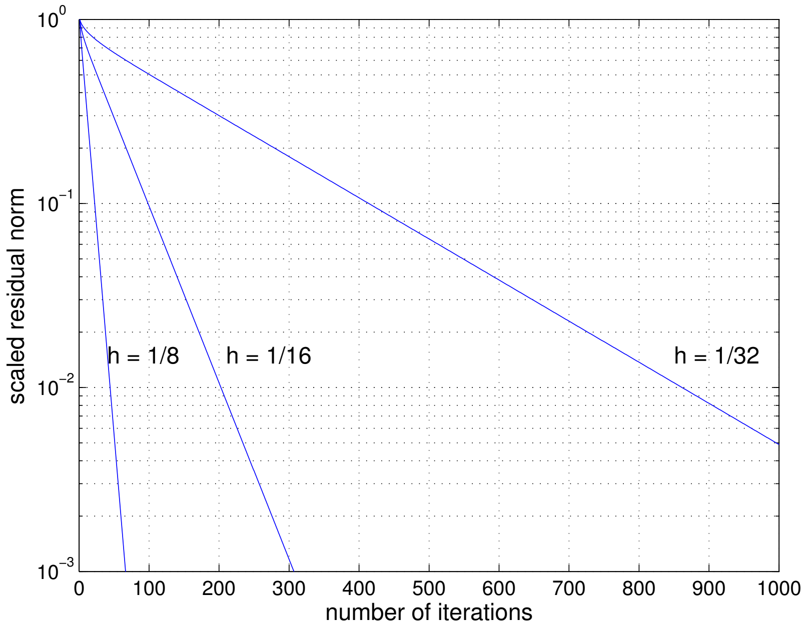


Figure 5.3: Convergence of the Jacobi method for model problem **MP**-1 on various mesh sizes.

5.5.3 Speed of Convergence of SOR(ω)

In this section we look into the convergence of SOR(ω) motivated by the following goals:

- clarifying under which conditions the Gauss-Seidel method ($\omega = 1$ in SOR(ω) converges twice as fast the Jacobi method;
- illustrating how the the convergence of SOR(ω) can be adapted by tuning a parameter;

- giving an example of an iterative method that has a better computational asymptotic complexity for $h \to 0$ than the method of Jacobi and Gauss-Seidel.

To be able to make quantitative statements on the convergence of SOR(ω), we need to make additional assumptions on the matrix A as in the following definition.

Definition 5.5.2 *The matrix $A \in \mathbb{R}^{n\times n}$ has the A-*property *if a permutation matrix P exists such that the matrix $\overline{A} = PAP^T$ has the following form*

$$\overline{A} = PAP^T = \begin{bmatrix} D_1 & M_1 \\ M_2 & D_2 \end{bmatrix} \text{ with } D_1 \text{ and } D_2 \text{ diagonal matrices}\,. \tag{5.61}$$

Example 5.5.1 *The system matrix in model problem* **MP**-*1 has the A-property as the red-black reordering of the unknowns (3.51) results in a non-zero structure required by (5.61).*

The A-matrix property is important as it allows to:

- to relate the eigenvalues and thus the speed of convergence of the Jacobi and SOR(ω) iteration matrices (and thus relate the speed of convergence of Jacobi and Gauss-Seidel method as $B_{GS} = B_{SOR(\omega=1)}$);
- to find the values of ω that is optimal in terms of convergence for SOR(ω).

The following theorem is an intermediate results towards the above statement.

Theorem 5.5.2 *Assume the $n \times n$ matrix A to have the A-property and that $a_{ii} \neq 0$ for $i = 1, \ldots, n$. Let P be a permutation such that the matrix $\overline{A} = PAP^T$ is of the form (5.61) and let $\overline{A} = D^{-1}(I - L - U)$ denote a splitting of $\overline{A}$ of the form* (5.12)-(5.13). *Then for $\alpha \in \mathbb{C}$, $\alpha \neq 0$, the eigenvalues of the matrix*

$$B_{JAC}(\alpha) = \alpha L + \alpha^{-1} U \tag{5.62}$$

are independent of α

Proof. We show that the matrices $J(\alpha)$ and $J(1)$ are similar, and therefore have the same spectrum. Indeed, by definition there exists an permutation matrix P such that

$$\overline{A} = \begin{pmatrix} D_1 & M_1 \\ M_2 & D_2 \end{pmatrix} = D^{-1}(I - L - U)$$

where as D_1 and D_2 are invertible, and

$$D = \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix}\,,\; L = -\begin{pmatrix} 0 & 0 \\ D_2^{-1}M_2 & 0 \end{pmatrix} \text{ and } U = -\begin{pmatrix} 0 & D_1^{-1}M_1 \\ 0 & 0 \end{pmatrix}\,.$$

We now have that

$$\begin{aligned} B_{JAC}(\alpha) = \begin{pmatrix} 0 & \alpha^{-1}D_1^{-1}M_1 \\ \alpha D_2^{-1}M_2 & 0 \end{pmatrix} &= -S_\alpha \begin{pmatrix} 0 & D_1^{-1}M_1 \\ D_2^{-1}M_2 & 0 \end{pmatrix} S_\alpha^{-1} \\ &= -S_\alpha B_{JAC}(1) S_\alpha^{-1}\,, \end{aligned}$$

where S_α is the diagonal matrix

$$S_\alpha = \begin{pmatrix} I_1 & 0 \\ 0 & \alpha I_2 \end{pmatrix} \text{ and } I_1, I_2 \text{ are identity matrices}\,.$$

This concludes the proof. □

This result motivates the following definition.

Definition 5.5.3 *An matrix $A \in \mathbb{R}^n \times n$ is called* consistently ordered *iff the eigenvalues of the matrix $B_{JAC}(\alpha) = \alpha L + \alpha^{-1} U$ are independent of α.*

The concept of A-property was introduced by David Young (1923-2008) in his seminal PhD thesis and later generalized to consistent orderings by Richard Varga (1928). We have now all element to state the main result on the convergence of SOR(ω). Observe that Lemma 5.3.3 for $\omega = 1$ implies that $0 \in \sigma(B_{GS})$ always holds. The zero eigenvalue is therefore separately in part c. in the next theorem.

Theorem 5.5.3 *Assume A to be consistently ordered and let $\omega \neq 0$. Then the following propositions hold*

a. if $\mu \in \sigma(B_{JAC})$ then $-\mu \in \sigma(B_{JAC})$;

b. if $\mu \in \sigma(B_{JAC})$ and if

$$(1 - \lambda - \omega)^2 = \lambda \omega^2 \mu^2 \,, \tag{5.63}$$

then $\lambda \in \sigma(B_{SOR(\omega)})$;

c. if $\lambda \neq 0$, $\lambda \in \sigma(B_{SOR(\omega)})$ and if (5.63) holds, then $\mu \in \sigma(B_{JAC})$.

Proof. $\boxed{\text{a.}}$ If A is consistently ordered, then $J(1) = L + U$ and $J(-1) = -L - U$ have the same eigenvalues.
$\boxed{\text{b.}}$ If $\lambda = 0$, then (5.63) implies that $\omega = 1$. In this case $\lambda \in \sigma(B_{GS})$ as follows from Lemma 5.3.3. We can therefore assume that $\lambda \neq 0$ in demonstrating that

$$\mu \in \sigma(B_{JAC}) \Leftrightarrow \det[B_{JAC} - \mu I] = 0 \Leftrightarrow \det[L + U - \mu I] = 0$$

implies that $\lambda \in \sigma(B_{SOR(\omega)})$. From (5.5.3), follows that either $1-\lambda-\omega = \sqrt{\lambda}\omega\mu$ or $1-\lambda-\omega = -\sqrt{\lambda}\omega\mu$. Without loss of generality, we choose the latter. Using Lemma 5.3.2, we then have that

$$\begin{aligned}
\lambda \in \sigma(B_{SOR(\omega)}) &\Leftrightarrow \varphi_{SOR(\omega)}(\lambda) = 0 \\
&\Leftrightarrow \det\big[(1 - \lambda - \omega)I + \omega U + \lambda \omega L\big] = 0 \\
&\Leftrightarrow \det\big[-\omega\mu\sqrt{\lambda} I + \omega U + \lambda \omega L\big] = 0 \\
&\Leftrightarrow \det\big[-\omega\mu\sqrt{\lambda} I + \omega\sqrt{\lambda}(\sqrt{\lambda} L + \frac{1}{\sqrt{\lambda}} U)\big] = 0 \\
&\Leftrightarrow (\omega\sqrt{\lambda})^N \det\big[\sqrt{\lambda} L - \frac{1}{\sqrt{\lambda}} U - \mu I\big] = 0
\end{aligned}$$

By virtue of Theorem 5.5.2, the eigenvalues of the matrices $B_{JAC}(\sqrt{\lambda}) = \sqrt{\lambda} L + \frac{1}{\sqrt{\lambda}} U$ and $B_{JAC}(1) = L + U$ are the same, and therefore we have that $\mu \in \sigma(B_{SOR(\omega)})$.
$\boxed{\text{c.}}$ The above reasoning can be reversed to proof this part of the theorem. □

Example 5.5.2 *Theorem 5.5.2 shows that Jacobi iteration matrix B_{JAC} in model problem* **MP**-*1 lie symmetric around the origin.*

Corollary 5.5.1 *If A is consistently ordered, then*

$$\rho(B_{GS}) = [\rho(B_{JAC})]^2 \,. \tag{5.64}$$

This corollary quantities the heuristic that Gauss-Seidel is *twice as fast* to converge as Jacobi. In model problem **MP**-1 we have that

$$\begin{aligned}
\rho(B_{GS}) = \mu_{11}^2 &= \left[1 - \frac{\pi^2}{2} h^2 + \mathcal{O}(h^4)\right]^2 &(5.65)\\
&= 1 - \pi^2 h^2 + \mathcal{O}(h^4) \,, &(5.66)
\end{aligned}$$

showing that the twice bad news story holds for the Gauss-Seidel method as well. For $h = 1/64$ for instance, we that $\rho(B_{JAC}) = 0.9976$. The convergence of the Gauss-Seidel method is compared to that of the Jacobi method in Figure 5.4. This figure clearly shows that Gauss-Seidel requires half the number of Jacobi iterations to converge to the same accuracy.

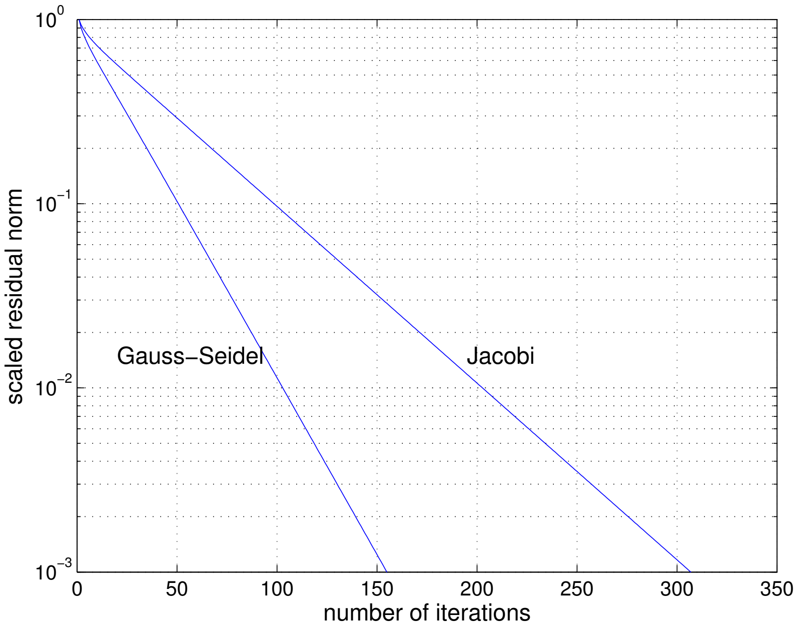


Figure 5.4: Convergence of the Jacobi and Gauss-Seidel method for model problem **MP**-1 for mesh size $h = 1/16$.

The above theorem gives the eigenvalues of $B_{SOR(\omega)}$ as the roots of a quadratic equation that depend on both ω and the eigenvalues of B_{JAC}. We will denote these roots as $\lambda_i^{(1)}(\omega, \mu_i)$ and $\lambda_i^{(2)}(\omega, \mu_i)$, where the numbering in i is in ascending order of magnitude. This quadratic relation allows to compute the optimal relaxation parameter denoted as ω^* and defined by

$$\omega^* = \text{argmin}_{\omega \in \mathbb{R}} \rho(B_{SOR(\omega)}) = \text{argmin}_{0<\omega<2} \rho(B_{SOR(\omega)}) \,, \tag{5.67}$$

where we've used Theorem 5.4.2. In the next theorem the optimal value of ω is determined.

Theorem 5.5.4 *Assume that $A \in \mathbb{R}^{n \times n}$ is consistently ordered and that the eigenvalues of B_{JAC} are real values with $\rho(B_{JAC}) < 1$. Then the optimal relaxation parameter and the corresponding optimal spectral radius are given by*

$$\omega^* = \frac{2}{1 + \sqrt{1 - [\rho(B_{JAC})]^2}} \textit{ and } \rho(B_{SOR(\omega=\omega^*)}) = \omega^* - 1 = \left(\frac{\rho(B_{JAC})}{1 + \sqrt{1 - [\rho(B_{JAC})]^2}} \right)^2 \,, \tag{5.68}$$

respectively. More generally, we have that

$$\rho(B_{SOR(\omega)}) = \begin{cases} \lambda_n^{(2)}(\omega, \rho(B_{JAC})) & \text{if } 0 < \omega \leq \omega^* \\ \omega - 1 & \text{if } \omega^* \leq \omega < 2 \end{cases} \tag{5.69}$$

where

$$\lambda_n^{(2)}(\omega, \rho(B_{JAC})) = 1 - \omega + \frac{1}{2}\omega^2[\rho(B_{JAC})]^2 + \omega\rho(B_{JAC})\sqrt{1 - \omega + \frac{1}{4}\omega^2[\rho(B_{JAC})]^2} \tag{5.70}$$

as illustrated in Figure 5.5.

Proof. We start the proof by a geometric interpretation of the relation (5.63) that can be equivalently rewritten as

$$(\lambda + \omega - 1)/\omega = \pm\mu_i\sqrt{\lambda} \,,$$

separating the dependency in ω and μ_i in the left and right-hand side, respectively. This shows that the roots $\lambda_i^{(1)}(\omega, \mu_i)$ and $\lambda_i^{(2)}(\omega, \mu_i)$ are abscissa of the points of intersection of the line $L_\omega(\lambda) = (\lambda + \omega - 1)/\omega$ and the parabola $P_{\mu_i}(\lambda) = \pm\mu_i\sqrt{\lambda}$ as shown in Figure 5.5(a) . For all $\omega \in (0, 2)$, the line $L_\omega(\lambda)$ passes through the points $(\lambda = 1 - \omega, L_\omega = 0)$ and $(\lambda = 1, L_\omega = 1)$ as shown in Figure 5.5(b). Increasing the value of $\mu_i \in (0, 1)$ decreases the curvature of P_{μ_i} as shown in Figure 5.5(c). To find the largest abscissa of intersection of $L_\omega(\lambda)$ and P_{μ_i}, it suffices to consider $\mu_i = \rho(B_{JAC})$ only. There exists a critical value ω_c for ω such that the graphs of $L_\omega(\lambda)$ and $P_{\rho(B_{JAC})}$ have two points of intersection for $\omega \in (0, \omega_c)$, one single point of intersection for $\omega = \omega_c$ and no such points for $\omega > \omega_c$. For $\omega = 1$, the points of intersection are $\lambda^{(1)} = 0$ and $\lambda^{(2)} = [\rho(B_{JAC})]^2$. To find ω_c, we adopt an algebraic argument and write (5.63) as the following quadratic equation in λ

$$\lambda^2 + [2(\omega - 1) - \omega^2(\rho(B_{JAC}))^2]\lambda + (\omega - 1)^2 = 0\,. \tag{5.71}$$

Depending on the sign of the discriminant

$$D_\lambda = \omega^2(\rho(B_{JAC}))^2[(\rho(B_{JAC}))^2\omega^2 - 4(\omega - 1)]\,,$$

this equation has two distinct real-valued ($D_\lambda > 0$), one single real-valued ($D_\lambda = 0$) or two complex conjugate $D_\lambda < 0$ solutions. The roots of D_λ in turn are given by $\omega = 0$, and the roots of the quadratic equation

$$(\rho(B_{JAC}))^2\omega^2 - 4\omega + 4 = 0\,.$$

given by $\omega_\pm = (2 \pm 2\sqrt{1 - [\rho(B_{JAC})]^2})/[\rho(B_{JAC})]^2$, where $1 < \omega_- < 2$ and $\omega_+ > 2$. We furthermore have that

- for $0 < \omega \leq \omega_-$, $D_\lambda > 0$ and the two real roots of (5.71) are

$$\lambda_n^{(1),(2)}(\omega, \rho(B_{JAC})) = 1 - \omega + \frac{1}{2}\omega^2[\rho(B_{JAC})]^2 \pm \omega\rho(B_{JAC})\sqrt{1 - \omega + \frac{1}{4}\omega^2[\rho(B_{JAC})]^2}\,,$$

- for $\omega = \omega_-$, $D_\lambda = 0$ and the single real root of (5.71) is

$$\lambda_n^{(1)}(\omega, \rho(B_{JAC})) = \lambda_n^{(2)}(\omega, \rho(B_{JAC})) = 1 - \omega + \frac{1}{2}\omega^2[\rho(B_{JAC})]^2\,;$$

- for $\omega > \omega_-$, $D_\lambda < 0$ and the modulus of the complex conjugate roots of (5.71) in λ is

$$|\lambda_n^{(1)}(\omega, \rho(B_{JAC}))| = |\lambda_n^{(2)}(\omega, \rho(B_{JAC}))| = |1 - \omega| = \omega - 1\,;$$

and therefore we have that

- for $0 < \omega \leq \omega_-$, $\rho(B_{SOR(\omega)}) = \lambda_n^{(2)}(\omega, \rho(B_{JAC}))$;
- for $\omega_- \leq \omega \leq 2$, $\rho(B_{SOR(\omega)}) = \omega - 1$.

The graph in Figure 5.5(d) shows that $\rho(B_{SOR(\omega)})$ reaches its minimum at

$$\omega^* = \omega_- = \frac{2 - 2\sqrt{1 - [\rho(B_{JAC})]^2}}{[\rho(B_{JAC})]^2} = \frac{2}{1 + \sqrt{1 - [\rho(B_{JAC})]^2}} \tag{5.72}$$

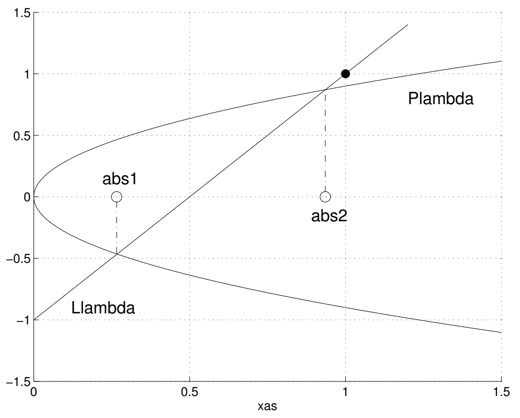
with a minimum value equal given by (5.68), completing the proof. □

In the model problem **MP**-1 we have that the optimal relaxation parameter is given by

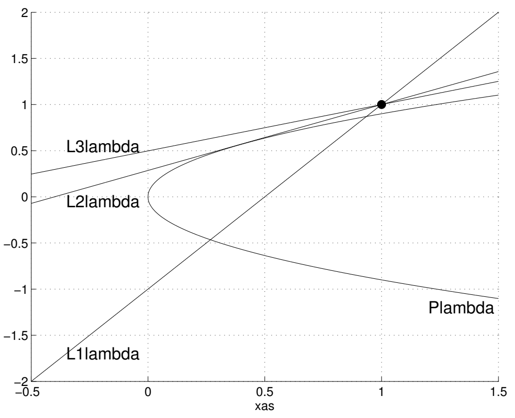
$$\omega^* = \frac{2}{1 + \sin(\pi h)} \tag{5.73}$$

and that the corresponding spectral radius is given by

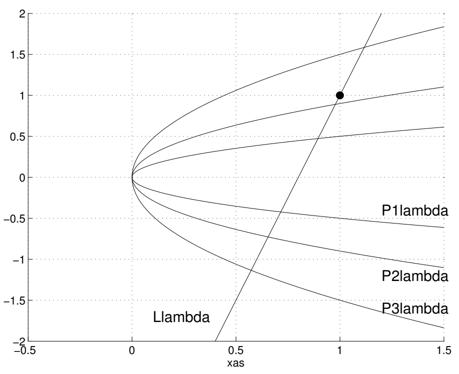
$$\rho(B_{SOR(\omega=\omega^*)}) = \omega^* - 1 = \frac{1 - \sin(\pi h)}{1 + \sin(\pi h)} = 1 - 2\pi h + \mathcal{O}(h^2) \tag{5.74}$$



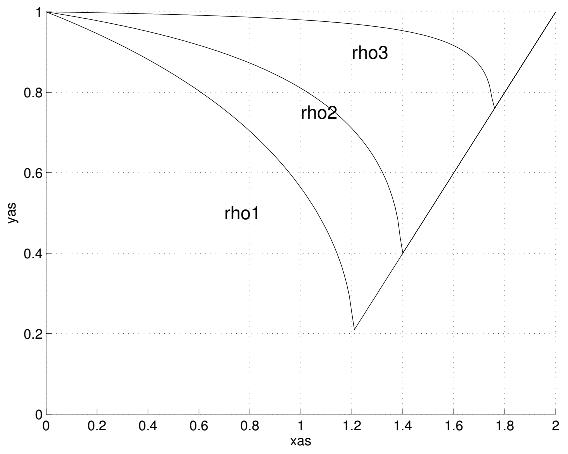
(a) $\lambda_i^{(1)}(\omega,\mu)$ and $\lambda_i^{(2)}(\omega,\mu)$.



(b) $L_\omega(\lambda)$ vs. λ for various ω.



(c) $P_\mu(\lambda)$ vs. λ for various μ.



(d) $\rho(B_{SOR(\omega)})$ vs. ω for various $\rho(B_{JAC})$.

Figure 5.5: Convergence analysis of SOR(ω).

which for $h = 1/64$ yields $\rho(B_{SOR(\omega=\omega^*)}) = 0.90$. The convergence of the SOR(ω) method applied to model problem **MP**-1 for the mesh size $h = 1/16$ and for various values of ω is shown in Figure 5.6. This figure clearly shows reduction in number of iterations that can be achieved by choosing close-to-optimal values of ω.

The fact that using the optimal relaxation parameter $\rho(B_{SOR(\omega=\omega^*)}) = 1 - \mathcal{O}(h)$ is a substantial improvement over Gauss-Seidel (and Jacobi) for which $\rho(B_{GS}) = 1 - \mathcal{O}(h^2)$. Determining the optimal ω-value does however require the computation of $\rho(B_{JAC})$, which is not a straightforward task.

To summarize this section, we recall the three main results

$$\rho(B_{JAC}) = 1 - \frac{\pi^2}{2}h^2 + \mathcal{O}(h^4) \tag{5.75}$$

$$\rho(B_{GS}) = 1 - \pi^2 h^2 + \mathcal{O}(h^4) \tag{5.76}$$

$$\rho(B_{SOR(\omega=\omega^*)}) = 1 - 2\pi h + \mathcal{O}(h^2) \tag{5.77}$$

A more refined analysis of matrices with the A-property allows to derive expressions for the eigenvectors of the GS and SOR(ω) iteration matrices. This in turn allows to analyze the convergence of these methods in more details.

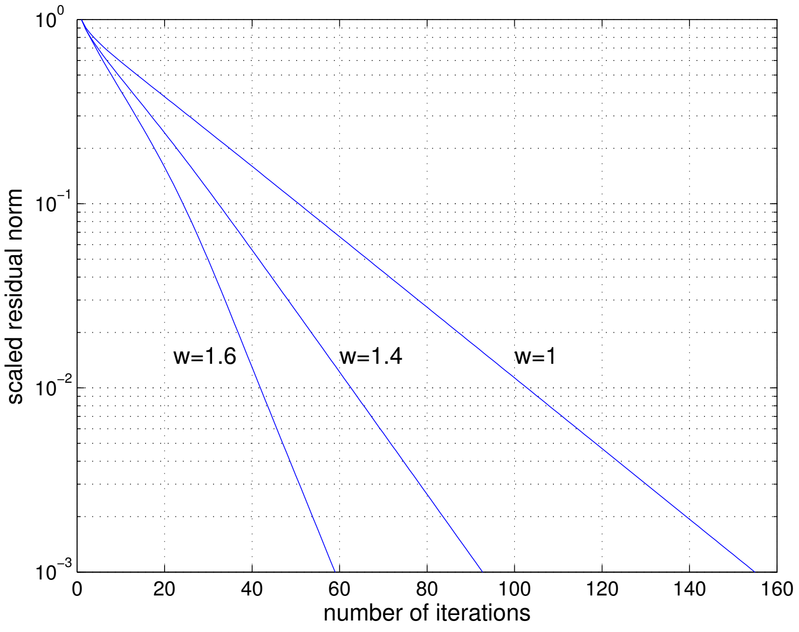


Figure 5.6: Convergence of the SOR(ω) method for model problem **MP**-1 for mesh sizes $h = 1/16$ for various values of ω.

5.6 Starting, Monitoring and Stopping

The practical implementations of (basic) iterative methods requires an initial guess, a procedure monitoring the converge (i.e., the distance to the solution) and a stopping criterium. These issues will be looked into in this section.

5.6.1 Choice of Starting Guess

The choice of the initial guess is usually given by the context of the problem. In solving a discretized linear elliptic partial differential equation for example, an initial guess can be obtained by for instance solving the same problem approximately by a simplified pen-and-paper model or on a coarser mesh. The latter option requires interpolating the solution obtained from the coarse to the fine mdesh. In solving a non-linear, time op parameter-dependent partial differential equation, an initial guess can be obtained from a previous iterations or by an extrapolation of a sequence of such solutions.

5.6.2 Monitoring the Convergence Process

To judge the quality of the solution $\mathbf{u}^k$ of a given iterand at iteration k, the 2-norm of the residual vector is usually used. Bounding the norm of the error by the norm of the residual however is not without difficulties. Indeed, as the residual equations (5.5) hold, we have the following bound on the relative error of the error

$$\frac{\|\mathbf{e}^k\|}{\|\mathbf{e}^0\|} \leq \kappa_2(A) \frac{\|\mathbf{r}^k\|}{\|\mathbf{r}^0\|} , \tag{5.78}$$

and this bound is unpractical in case that $\kappa_2(A)$ is large.

Application-specific contexts might motivate the use of other norms or other quantitaties like an energy or deviation from a divergence-free vector field.

5.6.3 Choice of Stopping Criterium

In the choice of a good stopping criterium one aims at balancing the quality imposed on the solution with the required computational cost. The above discussion motivates iterating untill the residual norm $\|\mathbf{r}^k\|$ becomes smaller than a small number ϵ. In the following we make this statement more precise by discussing various stopping criteria. One can decide to stop iterating if

1. criterium 1: $\|\mathbf{r}^k\| \leq \epsilon$:
   the main disadvantage of this stopping criterium is that it is not scaling invariant. This means that $\|\mathbf{r}^k\| = \|\mathbf{f} - A\mathbf{u}^k\| \leq \epsilon$ does *not* imply that e.g. $\|100(\mathbf{f} - A\mathbf{u}^k)\| \leq \epsilon$, although the accuracy of the iterand $\mathbf{u}^k$ remains the same. Some kind of scaling therefore needs to be introduced.

2. criterium 2: $\|\mathbf{r}^k\|/\|\mathbf{r}^0\| \leq \epsilon$:
   the main disadvantage of this stopping criterium is that the required accuracy increases with the quality of the initial guess, i.e., a good initial guess does *not* imply a reduction in the number of iterations.

3. criterium 3: $\|\mathbf{r}^k\|/\|\mathbf{f}\| \leq \epsilon$:
   this is a good stopping criterium.

5.7 The Triumvir: Splitting, Defect Correction and Preconditioning

BIMs have been shown to be slow to converge. They do however serve as *building blocks* for more efficient iterative solution methods. In this section we introduce three equivalent points of view on writing the matrix A as in (5.6) that will allow to build these more efficient methods.

Splitting Writing the matrix A as in equation (5.6) is referred to as a *splitting* of A. Observe the contrast between factoring the matrix A in $A = LU$ performed by direct soution methods and a splitting of A. In direct solution methods, computing the factors L and U determines the computational complexity of the algorithm. In BIMs, the term M is immediately available. The choice of M is motivated by the desire to make the linear system solve involving M as cheap as possible (either diagonal or triangular).

Defect Correction We already mentioned the fact that knowning the error $\mathbf{e}^k$ is equivalent to knowing the exact solution $\mathbf{u}^*$. Assume an approximation $\mathbf{u}^k$ and some approximation $\widehat{A}$ to A such that the $\widehat{A}$ linear system is easy to solve to be given. Then we can define an iterative scheme by solving the residual equations (5.5) approximately by the following sequence of three steps

- compute the *defect*: $\mathbf{r}^k = \mathbf{f} - A\mathbf{u}^k$;
- compute the approximate *correction* by solving the approximate residual equations: $\widehat{A}\widehat{\mathbf{e}}^k = \mathbf{r}^k$;
- add the correction to the previous iterand $\mathbf{u}^{k+1} = \mathbf{u}^k + \widehat{\mathbf{e}}^k$.

We will refer to this scheme as the *defect-correction* scheme. Combining the last two steps we arrive at

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \widehat{\mathbf{e}}^k = \mathbf{u}^k + (\widehat{A})^{-1}\mathbf{r}^k\,, \tag{5.79}$$

which upon identifying terms with (5.7) shows that $\widehat{A}$ in a defect-correction scheme plays the same role as a BIM defined by M. The concept of a defect-correction iteration will link BIM with multigrid methods in subsequent chapters. In the latter, the matrix $\widehat{A}$ is obtained by discretizing the PDE on a coarser mesh. In other contexts, the matrices A and $\widehat{A}$ may correspond to the discretization of the same partial differential equation using higher and lower order techniques, respectively.

Preconditioning The term preconditioning refers to replacing the linear system to solve by another one that it easier to solve while keeping the same solution. Given a matrix M, possibly in factored form $M = M_1\,M_2$ such that the M (M_1 and M_2) linear system is easy to solve, the term *left*, *right* and *split* preconditioning refer to solving

$$M^{-1}\,A\mathbf{u} = M^{-1}\mathbf{f} \tag{5.80}$$

$$A\,M^{-1}\mathbf{z} = \mathbf{f} \qquad \mathbf{z} = M\mathbf{u} \tag{5.81}$$

$$M_1^{-1}A\,M_2^{-1}\mathbf{z} = M_1^{-1}\mathbf{f} \qquad \mathbf{z} = M_2\mathbf{u}\,, \tag{5.82}$$

respectively. Richardson with parameter $\tau = 1$ applied to the left-(right-)preconditioned system has iteration matrix $B = I - M^{-1}A$ ($B = I - AM^{-1}$). Solving the left preconditioned system with Richardson ($\tau = 1$) therefore corresponds to solving the original unpreconditioned system with a BIM defined by M. The goal of preconditioning is to enhance the speed of Richardson by replacing asymptotic speed of convergence $\rho(I - A)$ on the unpreconditioned system by the superior rate $\rho(I - M^{-1}A) \ll \rho(I - A)$ of the preconditioned system. The concept of preconditioning will link BIMs with Krylov subspace methods in subsequent chapters.

5.8 What did we learn?

In this chapter we introduced basic iterative methods for solving linear systems. Quantitative and qualitative convergence properties were discussed and illustrated on model problems.

5.9 Exercises

Exercise 5.9.1

a) Let $A \in \mathbb{R}^{n\times n}$ be a positive definite symmetric $(n \times n)$-matrix. Show, that

$$||x|| := \sqrt{x^T A x}$$

defines a vector norm in $\mathbb{R}^n$.

b) Show that the spectral norm for square matrices is the matrix norm induced by the Euclidean norm (see Exercise 2.12.9 for the definition of induced matrix norms).
Hint: The spectral norm is defined as follows:

$$\| A \|_2 := \sqrt{\rho(\overline{A}^T A)}$$

where ρ is the spectral radius.

c) Let $D \in \mathbb{R}^{n\times n}$ be a diagonal $(n \times n)$-matrix with diagonal elements $d_i > 0$ and let the $(n \times n)$-matrix $C \in \mathbb{R}^{n\times n}$ be symmetric.
Show that, the following inequality is valid for the spectral norm:

$$\| D^{-1} C \|_2 \leq (\min\{d_i : 1 \leq i \leq n\})^{-1} \rho(C)$$

d) Determine the unit circle $E := \{x \in \mathbb{R}^2 | x^T A x \leq 1\}$ for

$$A = \begin{pmatrix} 9/5 & -8/5 \\ -8/5 & 21/5 \end{pmatrix}$$

and sketch E.

Exercise 5.9.2 Show that the iteration given by

$$r^i = b - Au^i, \;\; \hat{A}\hat{e}^i = r^i, \;\; u^{i+1} = u^i + \hat{e}^i \quad (i = 0, 1, 2 \ldots)$$

can be written in the form $u^{i+1} = Qu^i + s$, with $Q = I - (\hat{A})^{-1} A$.

Exercise 5.9.3 Show that the iteration

$$\hat{A} u^{i+1} = R u^i + b \; .$$

(which is based on a splitting $A = \hat{A} - R$ leads to an iteration of the form $u^{i+1} = Qu^i + s$, with $Q = I - (\hat{A})^{-1} A$.

Exercise 5.9.4 Let A be a symmetric, positive definite matrix. Show that the Richardson iteration applied to A converges in the Euclidean norm for

$$0 < \tau < 2\|A\|_S^{-1} \; .$$

Exercise 5.9.5 Let A be a symmetric, positive definite matrix with largest and smallest eigenvalues λ_{max} and λ_{min} respectively. Show that the value of τ for which $\rho(I - \tau A)$ becomes minimum, and thus the optimal value for the Richardson iteration, is

$$\tau = \frac{2}{\lambda_{\text{max}} + \lambda_{\text{min}}} \; .$$

Exercise 5.9.6 For the solution of the system $Au = b$, an iterative method

$$u^{i+1} = Qu^i + s$$

with $u^0 = 0$ is chosen. Prove that for u^i

$$u^i = (I - Q^i)A^{-1}f$$

is valid. Indicate the assumptions and determine s.

Exercise 5.9.7 Consider the problem

$$\begin{aligned} 10u_1 + u_2 &= 1 \\ u_1 + 10u_2 &= 10 \end{aligned}$$

with the solution $(u_1, u_2)^T = (0, 1)^T$. For a general system of equations

$$Au = b$$

with an $n \times n$ matrix A, which consists of a lower triangular submatrix L, the diagonal D and the upper triangular submatrix U ($A = L + D + U$), Gauss–Seidel iteration is defined by

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

and Jacobi iteration is given by

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

Perform

a) four Gauss–Seidel iterations,
b) four Jacobi iterations,

starting with the initial approximation $(u_1, u_2) = (0, 0)$.

Exercise 5.9.8 Perform

a) four Gauss–Seidel iterations,
b) four Jacobi iterations,

starting with the initial approximation $(u_1, u_2) = (0, 0)$, for the problem

$$\begin{aligned} u_1 + u_2 &= 2 \\ 4u_1 + 5u_2 &= 9 \end{aligned}$$

The exact solution is $(u_1, u_2) = (1, 1)$. Compare the convergence speed of both iterations with that in Exercise 5.9.7.

Exercise 5.9.9 Let the matrix A be as in Exercise 2.12.8. This matrix is a discretization of $\mathcal{L} = -d^2/dx^2$ with homogeneous boundary conditions.

a) Compute the eigenvalues and eigenvectors of L_h. What is the spectral norm of L_h?
b) Sketch the eigenvectors belonging to the largest and smallest eigenvalues as grid functions on the grid $[0, 1, 2, 3, 4, 5]$, appending zeros before and after them.

Exercise 5.9.10 Apply the lexicographic Gauss-Seidel method for the five-point stencil on a grid of mesh size h one time at the point (x_5, y_6) given that

a) $b_h(x_m, y_n) = 0$ for all m, n, $u_h^{i+1}(x_5 - h, y_6) = 1$, $u_h^{i+1}(x_5, y_6 - h) = 1$, $u_h^i(x_5 + h, y_6) = 1$, $u_h^i(x_5, y_6 + h) = 1$.

b) $b_h(x_m, y_n) = 0$ for all m, n, $u_h^{i+1}(x_5 - h, y_6) = -1$, $u_h^{i+1}(x_5, y_6 - h) = -1$, $u_h^i(x_5 + h, y_6) = 1$, $u_h^i(x_5, y_6 + h) = 1$.

Exercise 5.9.11

a) Show that the Jacobi iterative method, but not the Gauss-Seidel method, converges for

$$A = \begin{pmatrix} 1 & -2 & 2 \\ -1 & 1 & -1 \\ -2 & -2 & 1 \end{pmatrix}.$$

b) Show that the Gauss-Seidel method, but not the Jacobi method, converges for

$$A = \frac{1}{2}\begin{pmatrix} 2 & 1 & 1 \\ -2 & 2 & -2 \\ -1 & 1 & 2 \end{pmatrix}.$$

Exercise 5.9.12 Suppose that

$$A_1 = \begin{pmatrix} 1 & -1/2 \\ -1/2 & 1 \end{pmatrix} \text{ and } A_2 = \begin{pmatrix} 1 & -3/4 \\ -1/12 & 1 \end{pmatrix}$$

Let $B_{JAC,1}$ and $B_{JAC,2}$ be the associated Jacobi iteration matrices. Show that

$$\rho(B_{JAC,1}) > \rho(B_{JAC,2})$$

thereby refuting the claim that greater diagonal dominance implies more rapid Jacobi convergence.

Chapter 6

Multigrid Methods

`Towards Optimality.`

6.1 Introduction

In this chapter we introduce multigrid methods that are considered to be among the most efficient solution techniques for systems of linear equations resulting from the discretization of PDEs. In these solution methods the PDE background giving raise to the linear algebra problem is explicitly exploited. Their computational efficiency stems from a divide and conquer approach that decomposes the iteration error into frequency components and subsequently treats each component on its most appropriate scale of discretization.

Study Goals In this chapter we aim at

- explaining that while basic iterative methods (BIMs) are effective in reducing the high frequency component of the iteration error, the low frequency components impede the fast convergence of these methods. These low frequency error components are slowly varying grid vectors;
- explaining that slowly varying error grid vectors can be transferred to a coarser grid without essential loss of information and that this allows to construct defect correction (DC) type iteration that effectively reduces low frequency error components;
- explaining how a basic iterative method can be combined with defect correction to arrive at a two grid (TG) cycle that effectively reduces all frequency components of the error, i.e., explaning that TG = BIM + DC;
- explaining that computational efficiency requires to call the the two grid cycle recursively giving raise to different types of genuine multigrid cycles (MG);
- gives evidence of the distinct characteristic of the convergence of a multigrid cycle, namely its h-independent convergence and how this characteristic translates into computational efficiency.

6.2 Basic Iterative Methods Act as Smoothers

In this section we revisit the convergence of BIMs applied to the discrete Poisson equation and reveal what causes their slow convergence. We recall some facts from previous chapters. To this end we consider solving the linear system

$$A^h \mathbf{u}^h = \mathbf{f}^h \tag{6.1}$$

resulting from the finite difference discretization of model problem **MP**-1 in either one or two dimensions. The eigenvalues and eigenvectors of the system matrix A^h were derived in Chapter 3. The eigenmodes can be subdivided into *low* and *high frequency* modes. The low frequency modes are slowly varying grid vectors that correspond to the small eigenvalues of $A^h \in \mathbb{R}^{n \times n}$. We more precisely have that in one and two dimension the low frequency modes are the eigenvectors $\mathbf{v}^{h,[k]}$ for $1 \leq k \leq n/2$ and $\mathbf{v}^{h,[k\ell]}$ for $1 \leq k, \ell \leq n/2$, respectively. The remaining eigenmodes are called the high frequency modes and correspond to oscillatory grid vectors.

We consider solving (6.1) using Jacobi's method. Let us denote the matrices $D^h = \text{diag}(A^h)$, $N^h = M^h - A^h$, $B^h_{JAC} = I - (M^h)^{-1} A^h$ and the vector $\mathbf{s}^h = (M^h)^{-1} \mathbf{f}_j$. Jacobi's method produces a sequence of iterands $\mathbf{u}^{h,k+1}$ given by

$$\mathbf{u}^{h,k+1} = B^h_{JAC} \mathbf{u}^{h,k} + \mathbf{s}^h \, . \tag{6.2}$$

In our notation we will drop the upper index h in case there in The stencil representation of the error propagation matrix is given by (5.21) in one dimension and by (5.22) in two dimensions. These stencils show that a single iteration $\mathbf{e}^{k+1} = B_{JAC} \mathbf{e}^k$ amounts to replacing a single component of the error by an arithmetic average of its neighbors. We indeed have that $e^{k+1}_i = (e^k_{i-1} + e^k_{i+1})/2$ in 1D and $e^{k+1}_{ij} = (e^k_{(i-1)j} + e^k_{(i+1)j} + e^k_{i(j-1)} + e^k_{i(j+1)})/4$ in 2D. If therefore the error $\mathbf{e}^k$ is only a slowly varying grid vector (constant for instance), applying B to it will hardy change its shape or amplitude. Only if $\mathbf{e}^k$ is an oscillatory grid vector, the above averaging procedure will have an apparent effect.

To formalize these ideas, we consider solving (6.1) in 1D using a damped Jacobi method with corresponding error propagation matrix $B^h_{JAC(\omega)}$ defined by (5.31). Theorem 5.5.2 then show that the matrices A^h and $B^h_{JAC(\omega)}$ share their eigenvectors and that the eigenvalues of $B^h_{JAC(\omega)}$ are given by

$$\lambda_k(B^h_{JAC(\omega)}) = 1 - \omega \frac{h^2}{2} \lambda_k(A^h) = 1 - \omega \left[1 - \cos(\pi h k)\right] \, . \tag{6.3}$$

These eigenvalues are plotted for $n = 32$ as a function of k in Figure 6.1 for three values of ω. This figure shows that for all values of ω the spectral radius $\rho(B^h_{JAC(\omega)})$ is given by the eigenvalue corresponding to the lowest frequency mode $k = 1$. It also shows that for $\omega = 2/3$:

- the eigenvalues $\lambda_k(B^h_{JAC(2/3)})$ for $1 \leq k \leq n/2$ corresponding to *low* frequency modes are close to 1;
- the eigenvalues $\lambda_k(B^h_{JAC(2/3)})$ for $n/2 < k \leq n$ corresponding to *high* frequency modes are close to 0.

We use this information to reconsider the argument made in Subsection 5.5.1 on how the iteration error evolves according to $\mathbf{e}^k = B^k_{JAC} \mathbf{e}^0$. To this end we set first the initial error $\mathbf{e}^0$ equal to particular eigenmodes $\mathbf{v}^{h,[k]}$ of B_{JAC}. The analysis of Subsection 5.5.1 shows that the low frequency modes have a damping factor close to 1 and are thus slow to converge while high frequency modes have a small damping factor and are thus fast to converge. This is illustrated in Figure 6.2 in which three iterations of damped Jacobi are applied to different eigenmodes. Subsequently we set the initial error $\mathbf{e}^0$ equal to a grid vector made up from both low and high frequency components. In this case the analysis of Subsection 5.5.1 shows that damped Jacobi acts as a low-pass filter that effectively reduces the high frequency components of the error while leaving the low frequency error components virtually unchanged. Figure 6.3 indeed shows that the error becomes slowly varying while its amplitude is hardly affected. Stated differently, damped Jacobi acts as a *smoother*.

The argument given above for the damped Jacobi method for a one-dimensional problem carries over to higher dimensions and to other BIMs such as the Gauss-Seidel and SOR(ω) methods. The smoothing property of the lexicografic Gauss-Seidel method in two-dimensions e.g. is illustrated in Figure 6.4. The smoothing property of BIMs is the first of two ingredients of multigrid methods.

6.2.1 Smoothing Analysis

The above discussion does not address the issue of optimality in recuding the high-frequency error components. This optimality can be achieved either by choosing the most suitable damping parameter in the damped Jacobi method or by switching to another BIM. Understanding these alternatives in order to find the *best* possible smoother is the subject of the so-called *smoothing* analysis of multigrid methods.

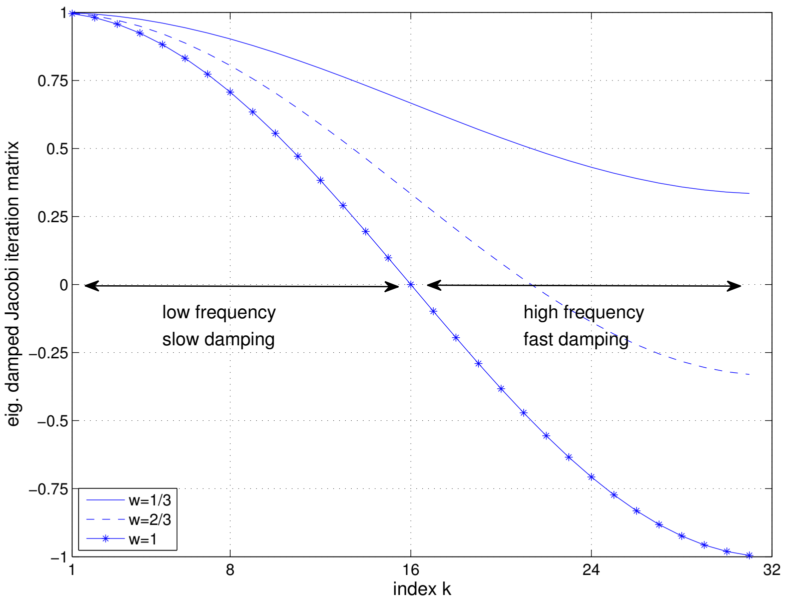


Figure 6.1: Smoothing behaviour of the damped Jacobi method for $n = 32$ and various values of ω.

6.3 Using Coarser Grids

The above reasoning motivates the use of coarser grids in solving the linear system (6.1). Indeed

- as a coarser grid contains less degrees of freedom, a BIM is cheaper to apply. On a mesh with a wider grid spacing, a BIM is furthermore faster to converge asymptotically. It might therefore be advantageous to use coarser grids either to generate good initial guesses or to use them in intermediate stages of computations on the fine grid;
- as smooth grid vectors can be transfered to a coarser grid without essential loss of information, a cheap to compute approximation of the error on the coarse grid can be attempted for;
- as smooth grid vectors appear as again oscillatory on coarser grids, a BIM will be effective in removing the high frequency components of the error.

The use of coarser grids requires both the transfer on grid vectors between grids of different mesh widths as well as the equivalent the coarse grid equivalent A^H of the fine grid operator A^h. These two topics will be treated in the next two sections.

6.4 Intergrid Transfer Operators

Intergrid transfer operators assure the transfer of vectors between grids of different size. In the discussing that follows we will consider the domain $\Omega = (0,1) \times (0,1)$ to be discretized by two uniform grids of meshwidth h and $2h$. We denote the set of fine and coarse grid by G_h and G_H and their number of elements by n_h and n_{2h}, respectively. As $G_H \subset G_h$, we can partition G_h according to $G_h = G_H \cup G_H^c$, where G_H^c is the complement of G_H in G_h.

Restriction The restriction operator denoted by I_h^{2h} (symbols to be read from bottom to top) transfers grid vectors from the fine (G_h) to coarse (G_H) grids, i.e.,

$$I_h^{2h} : \mathbf{u}^h \in \mathbb{R}^{n_h} \to \mathbf{u}^{2h} = I_h^{2h}\mathbf{u}^h \in \mathbb{R}^{n_{2h}} . \tag{6.4}$$

We distinguish several variants. The injection operator has a stencil given by

$$I_h^{2h} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}_h^{2h} . \tag{6.5}$$

The half injection operator has a stencil given by

$$I_h^{2h} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 0 \end{bmatrix}_h^{2h} . \tag{6.6}$$

The full weighting restriction operator performs a weighted averaging of the fine grid nodes. Its stencil is given by

$$I_h^{2h} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^{2h} . \tag{6.7}$$

In this notation the entry in the center corresponds to a coarse grid node that *receives* weight from itself and from its nearest neighbors on the fine grid. Using a lexicografic ordering of the grid nodes as before, the restriction operator can be assembled into an rectangular $n_{2h} \times n_h$ matrix with full row rank.

Interpolation The interpolation operator I_{2h}^h transfers grid vectors from the coarse to the fine grid, i.e.,

$$I_{2h}^h : \mathbf{u}^{2h} \in \mathbb{R}^{n_{2h}} \to \mathbf{u}^h = I_{2h}^h \mathbf{u}^{2h} \in \mathbb{R}^{n_h} . \tag{6.8}$$

In the bilinear interpolation data is transformed using the identity operator from G_H to G_h and from G_H to G_H^c using averaging from its (two or four) nearest neighbours. Its stencil is given by

$$I_{2h}^h = \frac{1}{4} \left] \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} \right[_{2h}^{h} . \tag{6.9}$$

in this notation the entry in the center corresponds to a coarse grid node that *gives* its weight to itself and from its nearest neighbors on the fine grid. Using a lexicografic ordering of the grid nodes, the interpolation operator can be represented by a rectangular $n_h \times n_{2h}$ matrix with full column rank. The bilinear interpolation and the full weighting restriction are related by the variational condition that says that

$$I_{2h}^h = 4(I_h^{2h})^T , \tag{6.10}$$

which is useful in theoretical considerations.

6.5 Coarse grid operator

The coarse grid equivalent A^{2h} of the fine grid matrix A^h can be build in two ways.

Rediscretization In the rediscretization approach the matrix A^{2h} is obtained on the coarse mesh in the same way that the matrix A^h is obtained on the fine mesh. In this way A^{2h} is a five-point discrete Laplacian with all the properties given in Chapter 3.

Galerkin Coarsening In the Galerkin approach the matrix A^{2h} is constructed algebraically using the relation

$$A^{2h} = I_h^{2h} A^h I_{2h}^h \, , \tag{6.11}$$

that is motivated by the residual equations on the coarse grid

$$A^{2h} e_{2h} = r_{2h} = I_h^{2h} r_h = I_h^{2h} A^h e_h = I_h^{2h} A^h I_{2h}^h e_{2h} \, . \tag{6.12}$$

Using Galerkin coarsening, the matrix A^{2h} is SPD whenever A^h is (explain why). If A^h is the five-point Laplacian and if I_h^{2h} and I_{2h}^h are the full weighting restriction and bilinear interpolation operators, then A^{2h} has the *nine* point stencil

$$A^{2h} = \frac{1}{4h^2} \begin{bmatrix} -1/3 & -1/3 & -1/3 \\ -1/3 & 8/3 & -1/3 \\ -1/3 & -1/3 & -1/3 \end{bmatrix} . \tag{6.13}$$

The matrix A^{2h} is then an irriducable K-matrix, and therefore an M-matrix.

6.6 Two Grid Method

We now have all ingredient to define an iterative two-grid method (TGM). We proceed as in the construction of the Symmetric SOR(ω) method and define a composite iteration step for the error $\mathbf{e}^k$ in which this time the action of a BIM acting as a smoothing in complemented with a defect-correction type iteration as detailed in Section 5.7. In the latter the coarser grid operator A^{2h} acts as the approximation $\widehat{A}$ to A^h. To formalize this idea, the use of of the intergrid transfer operator is required. The resulting algorithm is called the coarse grid correction (CGC) iteration and is described in Step 2 up to Step 6 in Algorithm 9. It can be written as a stationary iterative scheme with error propagation matrix B_{CGC} given by

$$\mathbf{e}^{k+1} = B_{CGC}\mathbf{e}^k \text{ where } B_{CGC} = I_{n_h \times n_h} - I_{2h}^h \, (A^{2h})^{-1} \, I_h^{2h} \, A^h \, . \tag{6.14}$$

This iteration dampens the low frequency components of the error that are hardly affected by the smoother. Let us denote the error propagation matrix corresponding to the (damped Jacobi, Gauss-Seidel or other) smoother and the number of times this matrix is applied by S_h and ν, respectively, i.e.,

$$\mathbf{e}^{k+\nu} = (S_h)^\nu \mathbf{e}^k \text{ where } S_h = I_{n_h \times n_h} - (M_h)^{-1} \, A^h \, . \tag{6.15}$$

This iteration can be written equivalently for the iterand $\mathbf{u}^{k+\nu}$. The divide-and-conquer strategy in frequency space is implemented by defining the error propagation matrix corresponding to the two-grid method as

$$\mathbf{e}^{k+1} = B_{TGM}\mathbf{e}^k \text{ where } B_{TGM}(\nu_1, \, \nu_2) = (S_h)^{\nu_2} B_{CGC} (S_h)^{\nu_1} \, , \tag{6.16}$$

where ν_1 and ν_2 represent the number of so-called pre and post-smoothing steps. The post-smoothing is introduced as adding the correction term e_h to the existing approximation $u_h^{i+1/3}$ (cfr. Algorithm 9) may introduce high frequency error components, a few post-smoothing steps are usually performed. It may also assure that B_{TGM} is symmetric is case that A^h is. The speed of convergence of this method is governed by the spectral radius $\rho(B_{TGM}(\nu_1, \, \nu_2))$.

6.7 Multigrid Method

The defect correction iteration replaces the fine grid linear system solve by the multiplication of a smoother and a coarse grid solve with A^{2h} as system matrix. This can somehow be compared with, given a natural number n, replacing the computation of n factorial by the multiplication of n (the smoother) and $n-1$ factorial (the coarse grid solve), i.e., $n! = n \cdot (n-1)!$. For large dimensions, solving the linear system with A^H as coefficient matrix can still be computationally expensive in the same way that computing $n-1$ factorial can be hard for large n. The idea therefore is to apply the two-grid idea to the A^{2h} linear system

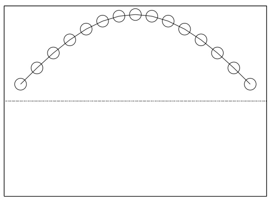
Algorithm 9 A two-grid cycle

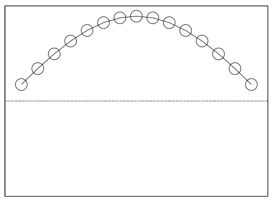
```
1  $u_h^{i+1/3} = S_h^{\nu_1}\, u_h^i + s_h$           $\nu_1$ presmoothing sweeps
2  $r_h = b_h - A_h u_h^{i+1/3}$                       residual computation
3  $r_{2h} = I_h^{2h}\, r_h$                            restriction of the residual to $G_{2h}$
4  $e_{2h} = (A^{2h})^{-1} r_{2h}$                      exact determination of the error on $G_{2h}$
5  $e_h = I_{2h}^h\, e_{2h}$                            prolongation of the error to $G_h$
6  $u_h^{i+2/3} = u_h^{i+1/3} + e_h$                   correction of the last solution iterate
7  $u_h^{i+1} = S_h^{\nu_2}\, u_h^{i+2/3} + s_h$       $\nu_2$ postsmoothing sweeps
```

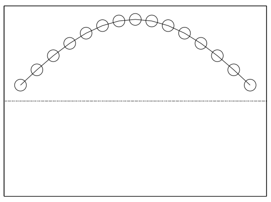
and to continue this recursion until the coarse linear can be solved with negligible computational cost. This can be viewed as the equivalent of computing n by $n! = n \cdot (n-1) \dots 3 \cdot 2 \cdot 1$ and gives raise to a genuine multigrid method.

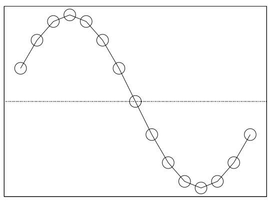
In a genuine multigrid cycle with error propagation matrix B_{MGM} a hierarchy of (more than two) grids of different levels of accuracy is visited during a *single* iteration $\mathbf{e}^{k+1} = B_{MGM}\mathbf{e}^k$. Depending on the order in which the grids are visited during one multigrid iteration, one distinguishes different *cycles* types. Calling the two-grid once or twice on each of the coarse grid results in the V-cycle and W-cycle, respectively. In the F-cycle the sequence of grids is traversed as shown in Figure 6.5.

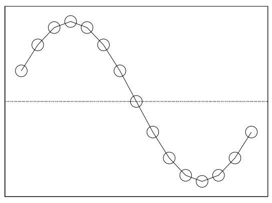
Multigrid methods are known to be optimal in the sense that the iteration matrix (6.16) can be bounded in some norm by a constant smaller than one *independent* of the mesh size h. This implies that the multigrid convergence is mesh size independent.

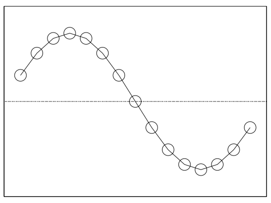

(a) $\mathbf{v}^{h,[1]}$

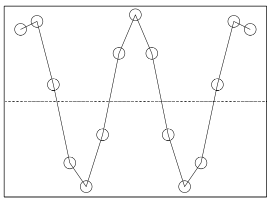

(b) $B\mathbf{v}^{h,[1]}$

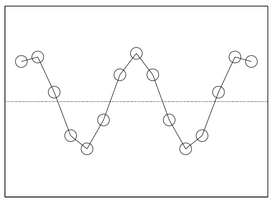

(c) $B^3\mathbf{v}^{h,[1]}$

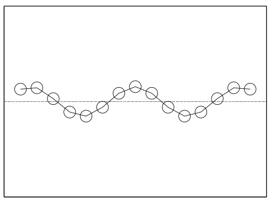

(d) $\mathbf{v}^{h,[2]}$

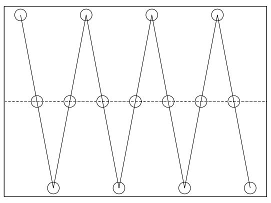

(e) $B\mathbf{v}^{h,[2]}$

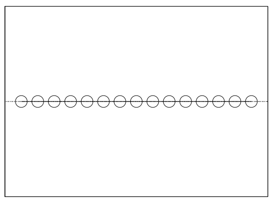

(f) $B^3\mathbf{v}^{h,[2]}$


(g) $\mathbf{v}^{h,[5]}$


(h) $B\mathbf{v}^{h,[5]}$


(i) $B^3\mathbf{v}^{h,[5]}$


(j) $\mathbf{v}^{h,[8]}$


(k) $B\mathbf{v}^{h,[8]}$

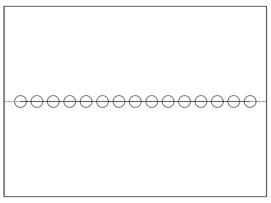
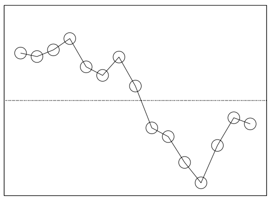
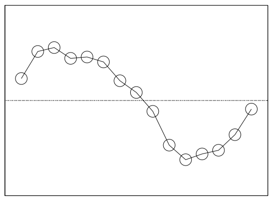

(l) $B^3\mathbf{v}^{h,[8]}$

Figure 6.2: Effect of doing 1 (middle column) and 3 (third column) damped Jacobi iterations on two low (first and second row) and two high frequency (third and fourth row) eigenvectors.


(a) $\mathbf{e}^h$


(b) $B\mathbf{e}^h$

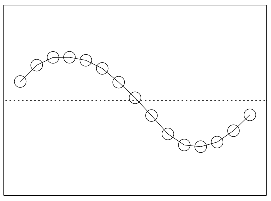
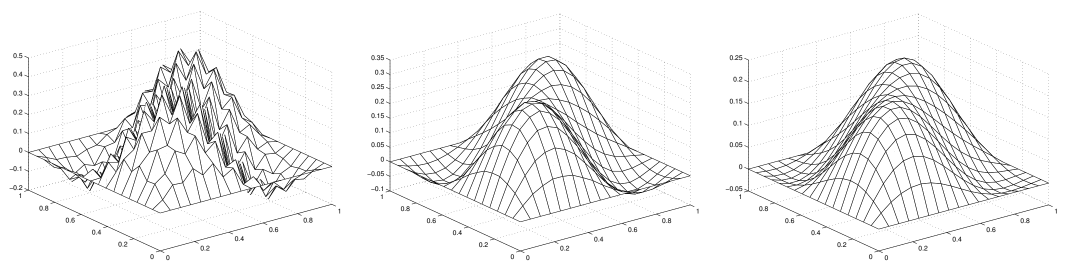
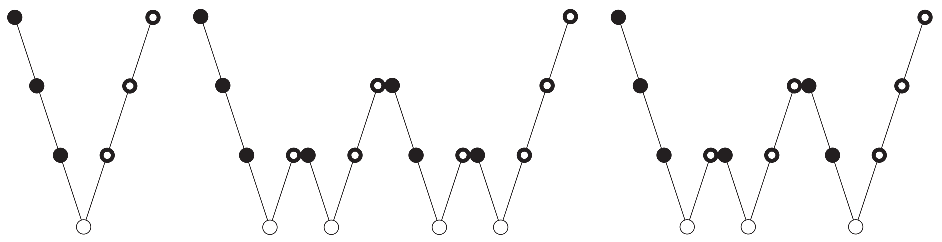

(c) $B^3\mathbf{e}^h$

Figure 6.3: Effect of doing 1 (middle column) and 3 (third column) damped Jacobi iterations on linear combination of low and high frequency error components.



(a) Initial error. (b) Error after 5 iterations. (c) Error after 10 iterations.

Figure 6.4: Illustration of the smoothing property of the Gauss-Seidel iteration. Figure 6.4(a) shows the initial error while Figure 6.4(b) and Figure 6.4(c) show the error after 5 and 10 iterations respectively.



(a) 4-grid V-cycle (b) 4-grid W-cycle (c) 4-grid F-cycle

Figure 6.5: Different multigrid cycles for a 4-grid method. ● = presmoothing, ⭘ = postsmoothing, ○ = exact solution, \ = Restriction, / = Prolongation

Chapter 7

Krylov Subspace Methods

7.1 Method for Systems with a Symmetric Positive Definite Matrix

7.1.1 Introduction

In the basic iterative solution methods we compute the iterates by the following recursion:

$$u^{i+1} = u^i + B^{-1}(b - Au^i) = u^i + B^{-1}r^i$$

Writing out the first steps of such a process we obtain:

$$\begin{array}{lcl} u^0, & & \\ u^1 & = & u^0 + (B^{-1}r^0), \\ u^2 & = & u^1 + (B^{-1}r^1) = u^0 + B^{-1}r^0 + B^{-1}(b - Au^0 - AB^{-1}r^0), \\ & = & u^0 + 2B^{-1}r^0 - B^{-1}AB^{-1}r^0, \\ \vdots & & \end{array}$$

This implies that

$$u^i \in u^0 + \text{ span} \left\{ B^{-1}r^0, B^{-1}A(B^{-1}r^0), \dots, (B^{-1}A)^{i-1}(B^{-1}r^0) \right\}.$$

The subspace $K^i(A; r^0) := \text{span} \left\{ r^0, Ar^0, \dots, A^{i-1}r^0 \right\}$ is called the Krylov-space of dimension i corresponding to matrix A and initial residual r^0. A u^i calculated by a basic iterative method is an element of $u^0 + K^i(B^{-1}A; B^{-1}r^0)$.

In this chapter we shall describe the Conjugate Gradient method. This method minimizes the error $u - u^i$ in an adapted norm, without any information about the eigenvalues. In Section 7.1.4 we give theoretical results concerning the convergence behavior of the CG method. Since part of the convergence theory is based on the Chebyshev method, we describe this method in Section 7.1.2

7.1.2 The Chebyshev method

A method to accelerate the convergence of a basic iterative method is the Chebyshev method. Suppose $u^1, \dots, u^k$ have been obtained via a basic iterative method, and we wish to determine coefficients $\gamma_j(k),\ j = 0, \dots, k$ such that

$$y^k = \sum_{j=0}^{k} \gamma_j(k) u^j \tag{7.1}$$

is an improvement of u^k. If $u^0 = \dots = u^k = u$, then it is reasonable to insist that $y^k = u$. Hence we require

$$\sum_{j=0}^{k} \gamma_j(k) = 1, \tag{7.2}$$

and consider how to choose the $\gamma_j(k)$ so that the error $y^k - u$ is minimized. It follows that $e^{(k+1)} = Q^k e^0$ where $e^k = u^k - u$. This implies that

$$y^k - u = \sum_{j=0}^{k} \gamma_j(k)(u^j - u) = \sum_{j=0}^{k} \gamma_j(k) Q^j e^0. \tag{7.3}$$

Using the 2-norm we look for $\gamma_j(k)$ such that $\|y^k - u\|_2$ is minimal. To simplify this minimization problem we use the following inequality:

$$\|y^k - u\|_2 \leq \|p_k(Q)\|_2 \|u^0 - u\|_2 \tag{7.4}$$

where $p_k(z) = \sum\limits_{j=0}^{k} \gamma_j(k) z^j$ and $p_k(1) = 1$. We now try to minimize $\|p_k(Q)\|_2$ for all polynomials satisfying $p_k(1) = 1$. Another simplification is the assumption that Q is symmetric with eigenvalues λ_i that satisfy $\alpha \leq \lambda_n \ldots \leq \lambda_1 \leq \beta < 1$. Using these assumptions we see that

$$\|p_k(Q)\|_2 = \max_{\lambda_i} |p_k(\lambda_i)| \leq \max_{\alpha < \lambda < \beta} |p_k(\lambda)|.$$

So to make the norm of $p_k(Q)$ small we need a polynomial $p_k(z)$ that is small on $[\alpha, \beta]$ subject to the constraint that $p_k(1) = 1$. This is a minimization problem of polynomials on the real axis. The solution of this problem is obtained by Chebyshev polynomials. These polynomials $c_j(z)$ can be generated by the following recursion

$$\begin{aligned} &c_0(z) = 1, \\ &c_1(z) = z, \\ &c_j(z) = 2z c_{j-1}(z) - c_{j-2}(z). \end{aligned}$$

These polynomials satisfy $|c_j(z)| \leq 1$ on $[-1, 1]$ but grow rapidly in magnitude outside this interval. As a consequence the polynomial

$$p_k(z) = \frac{c_k\left(-1 + 2\frac{z-\alpha}{\beta-\alpha}\right)}{c_k\left(1 + 2\frac{1-\beta}{\beta-\alpha}\right)}$$

satisfies $p_k(1) = 1$, since $-1 + 2\frac{1-\alpha}{\beta-\alpha} = 1 + 2\frac{1-\beta}{\beta-\alpha}$, and tends to be small on $[\alpha, \beta]$. The last property can be explained by the fact that

$$-1 \leq -1 + 2\frac{z-\alpha}{\beta-\alpha} \leq 1 \quad \text{for } z \in [\alpha, \beta] \text{ so the}$$

numerator is less than 1 in absolute value, whereas the denominator is large in absolute value since $1 + 2\frac{1-\beta}{\beta-\alpha} > 1$. This polynomial combined with (7.4) leads to

$$\|y^k - u\|_2 \leq \frac{\|u - u^0\|_2}{|c_k\left(1 + 2\frac{1-\beta}{\beta-\alpha}\right)|}. \tag{7.5}$$

Calculation of the approximation y^k by formula (7.1) costs much time and memory, since all the vectors $u^0, \ldots, u^k$ should be kept in memory. Furthermore, to calculate y^k one needs to add $k+1$ vectors, which for the model problem costs for $k \geq 5$ more work than one matrix vector product. Using the recursion of the Chebyshev polynomials it is possible to derive a three term recurrence among the y^k. It can be shown that the vectors y^k can be calculated as follows:

$$y^0 = u^0$$

solve z^0 from $Bz^0 = b - Ay^0$ then y^1 is given by

$$y^1 = y^0 + \tfrac{2}{2-\alpha-\beta} z^0$$

solve z^k from $Bz^k = b - Ay^k$ then $y^{(k+1)}$ is given by

$$y^{(k+1)} = \frac{4-2\beta-2\alpha}{\beta-\alpha} \frac{c_k\left(1+2\frac{1-\beta}{\beta-\alpha}\right)}{c_{k+1}\left(1+2\frac{1-\beta}{\beta-\alpha}\right)} \left(y^k - y^{(k-1)} + \frac{2}{2-\alpha-\beta} z^k\right) + y^{(k-1)} .$$

We refer to this scheme as the Chebyshev semi-iterative method associated with $By^{(k+1)} = (B-A)y^k + b$. Note that only 4 vectors are needed in memory and the extra work consists of the addition of 4 vectors. In order that the acceleration is effective it is necessary to have good lower and upper bounds of α and β. These parameters may be difficult to obtain. Chebyshev semi-iterative methods are extensively analyzed in [72], [26] and [32].

In deriving the Chebyshev acceleration we assumed that the iteration matrix $B^{-1}(B-A)$ was symmetric. Thus our simple analysis does not apply to the SOR iteration matrix $B_\omega^{-1}(B_\omega - A)$ because this matrix is not symmetric. To repair this Symmetric SOR (SSOR) is proposed. In SSOR one SOR step is followed by a backward SOR step. In this backward step the unknowns are updated in reversed order. For further details see [25], Section 10.1.5.

Finally we present some theoretical results for the Chebyshev method [1]. Suppose that the matrix $B^{-1}A$ is symmetric and positive definite and that the eigenvalues μ_i are ordered as follows $0 < \mu_1 \leq \mu_2 \ldots \leq \mu_n$. It is then possible to prove the following theorem:

Theorem 7.1.1 *If the Chebyshev method is applied and $B^{-1}A$ is symmetric positive definite then*

$$\|y^k - u\|_2 \leq 2\left(\frac{\sqrt{K_2(B^{-1}A)}-1}{\sqrt{K_2(B^{-1}A)}+1}\right)^k \|u^0 - u\|_2.$$

Proof Since $B^{-1}A = B^{-1}(B-(B-A)) = I - B^{-1}(B-A) = I - Q$ we see that the eigenvalues satisfy the following relation:

$$\mu_i = 1 - \lambda_i \quad \text{or} \quad \lambda_i = 1 - \mu_i.$$

This combined with (7.5) leads to the inequality:

$$\|y^k - u\|_2 \leq \frac{\|u - u^0\|_2}{\left|c_k\left(1+2\frac{(1-(1-\mu_1))}{(1-\mu_1)-(1-\mu_n)}\right)\right|}. \tag{7.6}$$

So it remains to estimate the denominator. Note that

$$c_k\left(1+\frac{2(1-(1-\mu_1))}{(1-\mu_1)-(1-\mu_n)}\right) = c_k\left(\frac{\mu_n+\mu_1}{\mu_n-\mu_1}\right) = c_k\left(\frac{1+\frac{\mu_1}{\mu_n}}{1-\frac{\mu_1}{\mu_n}}\right).$$

The Chebyshev polynomial can also be given by

$$c_k(z) = \frac{1}{2}\left\{\left(z+\sqrt{z^2-1}\right)^k + \left(z-\sqrt{z^2-1}\right)^k\right\} \quad \text{[3], p. 180.}$$

This expression can be used to show that

$$\begin{aligned} c_k\left(\frac{1+\frac{\mu_1}{\mu_n}}{1-\frac{\mu_1}{\mu_n}}\right) &> \frac{1}{2}\left(\frac{1+\frac{\mu_1}{\mu_n}}{1-\frac{\mu_1}{\mu_n}} + \sqrt{\left(\frac{1+\frac{\mu_1}{\mu_n}}{1-\frac{\mu_1}{\mu_n}}\right)^2 - 1}\right)^k = \\ &= \frac{1}{2}\left(\frac{1+\frac{\mu_1}{\mu_n}+2\sqrt{\frac{\mu_1}{\mu_n}}}{1-\frac{\mu_1}{\mu_n}}\right)^k = \frac{1}{2}\left(\frac{1+\sqrt{\frac{\mu_1}{\mu_n}}}{1-\sqrt{\frac{\mu_1}{\mu_n}}}\right)^k. \end{aligned} \tag{7.7}$$

The condition number $K_2(B^{-1}A)$ is equal to $\frac{\mu_n}{\mu_1}$. Together with (7.6) and (7.7) this leads to

$$\|y^k - u\|_2 \leq 2\left(\frac{\sqrt{K_2(M^{-1}A)}-1}{\sqrt{K_2(M^{-1}A)}+1}\right)^k \|u^0 - u\|_2.$$

[1]These results are used to analyze the converge behavior of other iterative methods

$\square$

Chebyshev type methods which are applicable to a wider range of matrices are given in the literature. In [40] a Chebyshev method is given for matrices with the property that their eigenvalues are contained in an ellipse in the complex plane, and the origin is no element of this ellipse. For a general theory of semi-iterative methods of Chebyshev type we refer to [14].

7.1.3 The Conjugate Gradient (CG) method

In this section we assume that $B = I$, and $u^0 = 0$ so $r^0 = b$. These assumptions are only needed to facilitate the formula's. They are not necessary for the CG method itself. Furthermore we assume that A is symmetric $(A = A^T)$ and positive definite $(x^T Ax > 0$ for $x \neq 0)$. This condition is crucial for the derivation and success of the CG method. Later on we shall derive extensions to non-symmetric matrices.

The first idea would be to construct a vector $u^i \in K^i(A, r^0)$ such that $\|u - u^i\|_2$ is minimal. The first iterate u^1 can be written as $u^1 = \alpha_0 r^0$ where α_0 is a constant which has to be chosen such that $\|u - u^1\|_2$ is minimal. This leads to

$$\|u - u^1\|_2^2 = (u - \alpha_0 r^0)^T (u - \alpha_0 r^0) = u^T u - 2\alpha_0 (r^0)^T u + \alpha_0^2 (r^0)^T r_0 \; . \tag{7.8}$$

The norm given in (7.8) is minimized if $\alpha_0 = \frac{(r^0)^T u}{(r^0)^T r^0}$. Since u is unknown this choice cannot be determined, so this idea does not lead to a useful method. Note that $Au = b$ is known so using an adapted inner product implying A could lead to an α_0 which is easy to calculate. To follow this idea we define the following inner product and related norm.

Definition

The A-inner product is defined by $(y, z)_A = y^T Az$, and the A-norm by

$$\|y\|_A = \sqrt{(y, y)_A} = \sqrt{y^T Ay}.$$

It is easy to show that if A is an SPD matrix, $(.,.)_A$ and $\|.\|_A$ satisfy the rules for inner product and norm, respectively. In order to obtain u^1 such that $\|u - u^1\|_A$ is minimal we note that

$$\|u - u^1\|_A^2 = u^T Au - 2\alpha_0 (r^0)^T Au + \alpha_0^2 \, (r^0)^T Ar^0,$$

so $\alpha_0 = \frac{(r^0)^T Au}{(r^0)^T Ar^0} = \frac{(r^0)^T b}{(r^0)^T Ar^0}$. We see that this new inner product leads to a minimization problem, which can easily be solved. In the following iterations, we compute u^i such that

$$\|u - u^i\|_A = \min_{y \in K^i(A; r^0)} \|u - y\|_A \tag{7.9}$$

The solution of this minimization problem leads to the Conjugate Gradient method [2]. First we specify the CG method, thereafter we summarize some of its properties.

[2]for another explanation see: http://www.cs.cmu.edu/∼quake-papers/painless-conjugate-gradient.pdf

Conjugate Gradient method

```
$u^0 = 0$ ;   $r^0 = b$                                   initialization
for           $k = 1, 2, \ldots$ do                        $k$ is the iteration number
              if $k = 1$ do
                 $p^1 = r^0$
              else
                 $\beta_k = \frac{(r^{k-1})^T r^{k-1}}{(r^{k-2})^T r^{k-2}}$    $p^k$ is the search direction vector
                 $p^k = r^{k-1} + \beta_k p^{k-1}$         to update $u^{k-1}$ to $u^k$
              end if
              $\alpha_k = \frac{(r^{k-1})^T r^{k-1}}{(p^k)^T A p^k}$
              $u^k = u^{k-1} + \alpha_k p^k$               update iterate
              $r^k = r^{k-1} - \alpha_k A p^k$             update residual
end for
```

The first description of this algorithm is given in [33]. Besides the two vectors u^k, r^k and matrix A, only one extra vector p^k should be stored in memory. Note that the vectors from a previous iteration can be overwritten. One iteration of CG costs one matrix vector product and 10 N flops for vector calculations. If the CG algorithm is used in a practical application the termination criterion should be replaced by one of the criteria given in Section 5.6.3. In this algorithm r^k is computed from r^{k-1} by the equation $r^k = r^{k-1} - \alpha_k A p^k$. This is done in order to save one matrix vector product for the original calculation $r^k = b - Au^k$ per iteration. In some applications the updated residual obtained from the CG algorithm can deviate significantly from the exact residual $b - Au^k$ due to rounding errors. It is therefore strongly recommended to recompute $b - Au^k$ after the termination criterion is satisfied for the updated residual and compare the norm of the exact and updated residual. If the exact residual does no satisfy the termination criterion the CG method should be restarted with u^k as its starting vector.

The vectors defined in the CG method have the following properties:

Theorem 7.1.2

1. $$\textit{span}\,\{p^1, \ldots, p^k\} = \textit{span}\,\{r^0, \ldots, r^{k-1}\} = K^k(A; r^0), \tag{7.10}$$
2. $$(r^j)^T r^i = 0 \quad i = 0, \ldots, j-1 \ ; \ j = 1, \ldots, k \ , \tag{7.11}$$
3. $$(r^j)^T p^i = 0 \quad i = 1, \ldots, j \ ; \ j = 1, \ldots, k \ , \tag{7.12}$$
4. $$(p^j)^T A p^i = 0 \quad i = 1, \ldots, j-1 \ ; \ j = 2, \ldots, k \tag{7.13}$$
5. $$\|u - u^k\|_A = \min_{y \in K^k(A; r^0)} \|u - y\|_A. \tag{7.14}$$

Proof: see [25], Section 10.2.

Some remarks on the properties given in Theorem 7.1.2 are:

- It follows from (7.10) and (7.11) that the vectors $r^0, \ldots, r^{k-1}$ form an *orthogonal basis* of $K^k(A; r^0)$.
- In theory the CG method is a finite method. After N iterations the Krylov subspace is identical to $\mathbb{R}^N$. Since $\|u - y\|_A$ is minimized over $K^N(A; r^0) = \mathbb{R}^N$ the norm should be equal to zero and $u^N = u$. However, in practice this property is never utilized for two reasons: firstly in many applications N is very large so that it is not feasible to perform N iterations; secondly even if N is small, rounding errors can spoil the results such that the properties given in Theorem 7.1.2 do not hold for the computed vectors.
- The sequence $\|u - u^k\|_A$ is theoretically monotonically decreasing, so

$$\|u - u^{k+1}\|_A \leq \|u - u^k\|_A \ .$$

This follows from (7.14) and the fact that $K^k(A;r^0) \subset K^{k+1}(A;r^0)$. In practice, $\|u-u^k\|_A$ is not easily computed since u is unknown. The norm of the residual is given by $\|r^k\|_2 = \|u-u^k\|_{A^TA}$. This sequence is not necessarily monotonically decreasing. In applications it may occur that $\|r^{k+1}\|_2$ is larger than $\|r^k\|_2$. This does not mean that the CG process becomes divergent. The inequality

$$\|r^k\|_2 = \|Au^k - b\|_2 \leq \sqrt{\|A\|_2}\|u-u^k\|_A$$

shows that $\|r^k\|_2$ is less than the monotonically decreasing sequence $\sqrt{\|A\|_2}\|u-u^k\|_A$, so after some iterations the norm of the residual decreases again.

- The direction vector p^j is *A-orthogonal or A-conjugate* to all p^i with index i less than j. This is the motivation for the name of the method: the directions or gradients of the updates are mutually conjugate.

- In the algorithm we see two ratios, one in calculating β_k and the other one for α_k. If the denominator is equal to zero, the CG method breaks down. With respect to β_k this implies that $(r^{k-2})^T r^{k-2} = 0$, which implies $r^{k-2} = 0$ and thus $u = u^{k-2}$. The linear system is solved. The denominator of α_k is zero if $(p^k)^T Ap^k = 0$, so if $p^k = 0$. Using property (7.10) this implies that $r^{k-1} = 0$ so, again, the problem is already solved.
  The conclusion is that if the matrix A satisfies Condition 2.1.2 then the CG method is robust.

In the following chapter we shall give CG type methods for more general matrices A, but first we shall extend the property SPD in such a way that also singular matrices are permitted. If the matrix A is symmetric and positive *semi* definite $(x^TAx \geq 0)$ (SPSD) the CG method can be used to solve the linear system $Au = b$, provided b is an element of the column space of A (range(A)). This is a natural condition because if it does not hold there would be no vector u such that $Au = b$. For further details and references see [36].

7.1.4 The Convergence Behavior of the CG Method

An important topic is the rate of convergence of the CG method. The optimality property enables one to obtain easy to calculate upper bounds of the distance between the k^{th} iterate and the exact solution.

Theorem 7.1.3 *The iterates u^k obtained from the CG algorithm satisfy the following inequality:*

$$\|u-u^k\|_A \leq 2\left(\frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1}\right)^k \|u-u^0\|_A. \tag{7.15}$$

Proof

We shall only give a sketch of the proof. It is easily seen that $u - u^k$ can be written as a polynomial, say $p_k(A)$ with $p_k(0) = 1$, times the initial residual

$$\|u-u^k\|_A = \|p_k(A)(u-u^0)\|_A.$$

Due to the minimization property every other polynomial $q_k(A)$ with $q_k(0) = 1$ does not decrease the error measured in the A-norm:

$$\|u-u^k\|_A \leq \|q_k(A)(u-u^0)\|_A.$$

The right-hand side can be written as

$$\|q_k(A)(u-u^0)\|_A = \|q_k(A)\sqrt{A}(u-u^0)\|_2 \leq \|q_k(A)\|_2\|\sqrt{A}(u-u^0)\|_2 = \|q_k(A)\|_2\|u-u^0\|_A$$

Taking $q_k(A)$ equal to the Chebyshev polynomial (see, for example, [25]) gives the desired result. □

Note that the rate of convergence of CG is comparable to that of the Chebyshev method, however it is not necessary to estimate or calculate eigenvalues of the matrix A. Furthermore increasing diagonal dominance leads to a better rate of convergence.

Initially, the CG method was not popular. The reason for this is that the convergence can be slow for systems where the condition number $\kappa_2(A)$ is very large. On the other hand the fact that the solution is found in N iterations is also not useful in practice, as N may be very large, and the property does not hold in the presence of rounding errors. To illustrate this we consider the following classical example:

Example 7.1.4.1 *The linear system $Au = b$ is to be solved where $N = 40$ and $b = (1, 0, \dots, 0)^T$. The matrix A is given by*

$$A = \begin{bmatrix} 5 & -4 & 1 & & & & & & \\ -4 & 6 & -4 & 1 & & & & \oslash & \\ 1 & -4 & 6 & -4 & 1 & & & & \\ & \ddots & \ddots & \ddots & \ddots & & & & \\ & & \ddots & \ddots & \ddots & \ddots & & & \\ & & & & 1 & -4 & 6 & -4 & 1 \\ & \oslash & & & & 1 & -4 & 6 & -4 \\ & & & & & & 1 & -4 & 5 \end{bmatrix}.$$

This can be interpreted as a finite difference discretization of the bending beam equation: $u'''' = f$. The eigenvalues of this matrix are given by:

$$\lambda_k = 16 sin^4 \frac{k\pi}{82} \quad k = 1, \dots, 40.$$

The matrix A is symmetric positive definite, so the CG method can be used to solve the linear system. The condition number of A is approximately equal to $(82/\pi)^4$. The resulting rate of convergence given by

$$\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1} \cong 0.997$$

is close to one. This explains a slow convergence of the CG method for the first iterations. However after 40 iterations the solution should be obtained. In Figure 7.1 the convergence behavior is given where the rounding error is equal to 10^{-16}, [22].

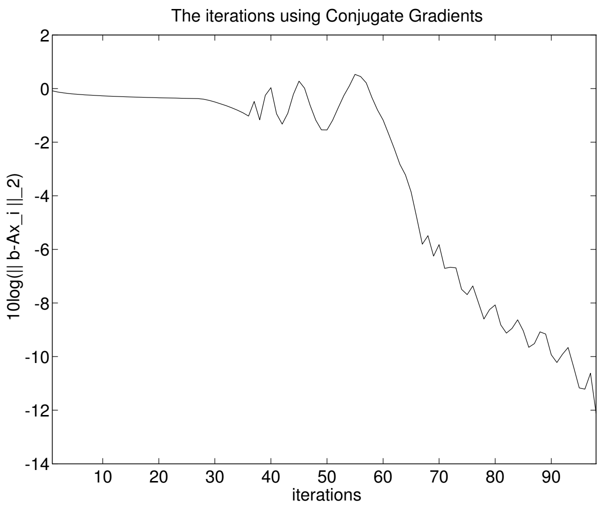


Figure 7.1: The convergence behavior of CG.

This example suggests that CG has only a restricted range of applicability. This idea changed, however, completely after the publication of [49]. Herein it has been shown that the CG method can be very useful for a class of linear systems, not as a direct method, but as an iterative method. The problem class originates from discretized partial differential equations. It appears that not the size of the matrix is important for convergence but the extreme eigenvalues of A.

One of the results which is based on the extreme eigenvalues is given in Theorem 7.1.3. Inequality (7.15) is an upper bound for the error of the CG iterates, and suggests that the CG method is a linearly convergent process (see Figure 7.2). However, in practice the convergence behavior looks like the one given in Figure 7.3. This is called *superlinear convergence behavior*. So, the upper bound is only sharp for the initial iterates. It seems that after some iterations the condition number in Theorem 7.1.3 is replaced by a smaller condition number. To illustrate this we give the following example:

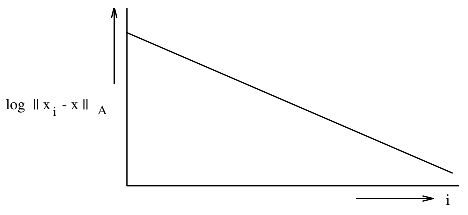


Figure 7.2: A linear convergent behavior

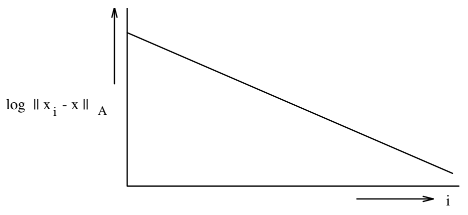


Figure 7.3: A superlinear convergent behavior

Example 7.1.4.2 *The matrix A is from the discretized Poisson operator. The physical domain is a two-dimensional unit square. The grid used consists of an equidistant distribution of* 30×30 *grid points. The dimension of A is equal to* 900 *and the eigenvalues are given by*

$$\lambda_{k,\ell} = 4 - 2cos\frac{\pi k}{31} - 2cos\frac{\pi \ell}{31} \ , \ \ 1 \leq k, \ell \leq 30.$$

Using Theorem 7.1.3 it appears that 280 *iterations are necessary to ensure that*

$$\frac{\|u - u^i\|_A}{\|u - u^0\|_A} \leq 10^{-12}.$$

Computing the solution it appears that the CG iterates satisfy the given termination criterion after 120 *iterations. So, for this example the estimate given in Theorem 7.1.3 is not sharp.*

To obtain a better insight in the convergence behavior we have a closer look into the CG method. We have seen that CG minimizes $\|u - u^i\|_A$ in the Krylov subspace. This can also be seen as the construction of a polynomial q_i of degree i and $q_i(0) = 1$ such that

$$\|u - u^i\|_A = \|q_i(A)(u - u^0)\|_A = \min_{\substack{\tilde{q_i}, \\ \tilde{q}_i(0) = 1}} \|\tilde{q}_i(A)(u - u^0)\|_A \ .$$

Suppose that the orthonormal eigensystem of A is given by: $\{\lambda_j, y^j\}_{j=1,\ldots,n}$ where $Ay^j = \lambda_j y^j$, $\lambda_j \in \mathbb{R}$, $\|y^j\|_2 = 1$, $(y^j)^T y^i = 0, j \neq i$, and $0 < \lambda_1 \leq \lambda_2 \ldots \leq \lambda_N$. The initial errors can be written as $u - u^0 = \sum\limits_{j=1}^{N} \gamma_j y^j$, which implies that

$$u - u^i = \sum_{j=1}^{N} \gamma_j q_i(\lambda_j) y^j \ . \tag{7.16}$$

If for instance $\lambda_1 = \lambda_2$ and $\gamma_1 \neq 0$ and $\gamma_2 \neq 0$ it is always possible to change y^1 and y^2 to $\tilde{y}_1$ and $\tilde{y}_2$ such that $\tilde{\gamma}_1 \neq 0$ but $\tilde{\gamma}_2 = 0$. This, in combination with equation (7.16), implies that if $q_i(\lambda_j) = 0$ for all different λ_j then $u^i = u$. So if there are only $m < N$ different eigenvalues the CG method stops at least after m iterations. Furthermore, the upper bound given in Theorem 7.1.3 can be sharpened.

Remark 1.4.1 For a given linear system $Au = b$ and a given u^0 (note that $u-u^0 = \sum\limits_{j=1}^{N} \gamma_j y^j$) the quantities α and β are defined by:

$$\alpha = \min \{\lambda_j | \gamma_j \neq 0\} ,$$
$$\beta = \max \{\lambda_j | \gamma_j \neq 0\} .$$

It is easy to show that the following inequality holds:

$$\|u - u^i\|_A \leq 2 \left(\frac{\sqrt{\frac{\beta}{\alpha}} - 1}{\sqrt{\frac{\beta}{\alpha}} + 1} \right)^i \|u - u^0\|_A. \tag{7.17}$$

The ratio $\frac{\beta}{\alpha}$ is called the *effective condition number* of A.

It follows from Theorem 7.1.2 that $r^0, \ldots, r^{k-1}$ form an orthogonal basis for $K^k(A; r^0)$. Then, the vectors $\tilde{r}^i = r^i / \|r^i\|_2$ form an orthonormal basis for $K^k(A; r^0)$. We define the following matrices

$$R_k \in \mathbb{R}^{N \times k} \text{ and the } j^{th} \text{ column of } R_k \text{ is } \tilde{r}^j,$$
$$T_k = R_k^T A R_k \text{ where } T_k \in \mathbb{R}^{k \times k}.$$

Ritzmatrix T_k can be seen as the projection of A on $K^k(A; r^0)$. It follows from Theorem 7.1.2 that T_k is a tridiagonal symmetric matrix. The coefficients of T_k can be calculated from the α_i's and β_i's of the CG process. The eigenvalues θ_i of the matrix T_k are called *Ritzvalues* of A with respect to $K^k(A; r^0)$. If z^i is an eigenvector of T_k so that $T_k z^i = \theta_i z^i$ and $\|z^i\|_2 = 1$ then $R_k \, z^i$ is called a *Ritzvector* of A. Ritzvalues and Ritzvectors are approximations of the eigenvalues and eigenvectors of A and play an important role in a better understanding of the convergence behavior of CG. Some important properties are:

- the rate of convergence of a Ritzvalue to its limit eigenvalue depends on the distance of this eigenvalue to the rest of the spectrum.
- in general the extreme Ritzvalues converge fastest and their limits are α and β.

In practical experiments we see that, if Ritzvalues approximate the extreme eigenvalues of A, then the rate of convergence appears to be based on a smaller effective condition number (the extreme eigenvalues seem to be absent). We first give an heuristic explanation. Thereafter an exact result from the literature is cited.

From Theorem 7.1.2 it follows that $r^k = A(u - u^k)$ is perpendicular to the Krylov subspace $K^k(A; r^0)$. If a Ritzvector is a good approximation of an eigenvector y^j of A this eigenvector is "nearly" contained in the subspace $K^k(A; r^0)$. These observations combined yield that $(A(u - u^k))^T y^j \cong 0$. The exact solution and the approximation can be written as

$$u = \sum_{i=1}^{N} (u^T y^i) y^i \ \text{ and } \ u^k = \sum_{i=1}^{N} ((u^k)^T y^i) y^i.$$

From $(A(u-u^k))^T y^j = (u-u^k)^T \lambda_j y^j \cong 0$ it follows that $x^T y^j \cong (u^k)^T y^j$. So the error $u - u^k$ has a negligible component in the eigenvector y^j. This suggests that λ_j does no longer influence the convergence of the CG process.

For a more precise result we define a *comparison process*. The iterates of this process are comparable to that of the original process, but its condition number is less than that of the original process.

Definition
Let u^i be the i-th iterate of the CG process for $Au = b$. For a given integer i let $\overline{u}^j$ denote the j-th iterate of the comparison CG process for this equation, starting with $\overline{u}^0$ such that $u - \overline{u}^0$ is the projection of $u - u^i$ on span$\{y^2, \ldots, y^N\}$.

Note that for the comparison process the initial error has no component in the y^1 eigenvector.

Theorem 7.1.4 *Let u^i be the i-th iterate of CG, and $\overline{u}^j$ the j-th iterate of the comparison process. Then for any j there holds:*

$$\|u - u^{i+j}\|_A \leq F_i \|u - \overline{u}^j\|_A \leq F_i \frac{\|u - \overline{u}^j\|_A}{\|u - \overline{u}^0\|_A} \|u - u^i\|_A$$

with $F_i = \dfrac{\theta_1^i}{\lambda_1} \max\limits_{k \geq 2} \dfrac{|\lambda_k - \lambda_1|}{|\lambda_k - \theta_1^i|}$, *where θ_1^i is the smallest Ritzvalue in the i-th step of the CG process.*

Proof: see [66], Theorem 7.1.2.

The theorem shows that from any stage i for which θ_1^i does not coincide with an eigenvalue λ_k, the error reduction in the next j steps is at most a fixed factor F_i worse than the error reduction in the first j steps of the comparison process in which the error vector has no y_1-component. As an example we consider the case that $\lambda_1 < \theta_1^i < \lambda_2$ we then have

$$F_i = \frac{\theta_1^i}{\lambda_1} \frac{\lambda_2 - \lambda_1}{\lambda_2 - \theta_1^i},$$

which is a kind of *relative convergence measure* for θ_1^i relative to λ_1 and $\lambda_2 - \lambda_1$. If $\frac{\theta_1^i - \lambda_1}{\lambda_1} < 0.1$ and $\frac{\theta_1^i - \lambda_1}{\lambda_2 - \lambda_1} < 0.1$ then we have $F_i < 1.25$. Hence, already for this modest degree of convergence of θ_1^i the process virtually converges as well as the comparison process (as if the y_1-component was not present). For more general results and experiments we refer to [66].

7.1.5 Exercises

Exercise 7.1.1 Show that $(y, z)_A = y^T A z$ is an inner product if A is symmetric and positive definite.

Exercise 7.1.2 Give the proof of inequality (7.17).

Exercise 7.1.3 Two properties of A-orthogonal sets are proved.

a) Show that an A-orthogonal set of nonzero vectors associated with a symmetric and positive definite matrix is linearly independent.
b) Show that if $\{v^1, v^2, \ldots, v^N\}$ is a set of A-orthogonal vectors in $\mathbb{R}^N$ and $z^T v^i = 0$ for $i = 1, \ldots, N$ then $z = 0$.

Exercise 7.1.4 Define

$$t_k = \frac{(v^k, b - Au^{k-1})}{(v^k, Av^k)}$$

and $u^k = u^{k-1} + t_k v^k$, then $(r^k, v^j) = 0$ for $j = 1, \ldots, k$, if the vectors v^j form an A-orthogonal set. To prove this, use the following steps using mathematical induction:

a) Show that $(r^1, v^1) = 0$.
b) Assume that $(r^k, v^j) = 0$ for each $k \leq l$ and $j = 1, \ldots, k$ and show that this implies that

$$(r^{l+1}, v^j) = 0 \text{ for each } j = 1, \ldots, l.$$

c) Show that $(r^{l+1}, v^{l+1}) = 0$.

Exercise 7.1.5 Take $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}$ and $b = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix}$. We solve $Au = b$.

a) Show that Conjugate Gradients applied to this system should convergence in 1 or 2 iterations (using the convergence theory).
b) Choose $u^0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ and do 2 iterations with the Conjugate Gradient method.

Exercise 7.1.6 Suppose that A is symmetric and indefinite. Give an example that shows that the Conjugate Gradient method can break down.

7.2 Preconditioning of Krylov Subspace Methods

We have seen that the convergence behavior of Krylov subspace methods strongly depends on the eigenvalue distribution of the coefficient matrix. A preconditioner is a matrix that transforms the linear system such that the transformed system has the same solution but the transformed coefficient matrix has a *more favorable spectrum*. As an example we consider a matrix M which resembles the matrix A. The transformed system is given by

$$M^{-1}Au = M^{-1}b \; ,$$

and has the same solution as the original system $Au = b$. The requirements on the matrix M are the following:

- the eigenvalues of $M^{-1}A$ should be clustered around 1,
- it should be possible to obtain $M^{-1}y$ at low cost.

Most of this chapter contains preconditioners for symmetric positive definite systems (Section 7.2.1). For non-symmetric systems the ideas are analogous. So, in Section 7.3.7 we give some details, which can be used only for non-symmetric systems.

7.2.1 The Preconditioned Conjugate Gradient (PCG) method

In Section 7.1.4 we observed that the rate of convergence of CG depends on the eigenvalues of A. Initially the condition number $\frac{\lambda_N}{\lambda_1}$ determines the decrease of the error. After a number of iterations $\frac{\lambda_N}{\lambda_1}$ is replaced by the effective condition number $\frac{\lambda_N}{\lambda_2}$ etc. So the question arises, whether it is possible to change the linear system $Au = b$ in such a way that the eigenvalue distribution becomes more favorable with respect to the CG convergence? This is indeed possible and the approach is known as: *the preconditioning of a linear system*. Consider the $N \times N$ symmetric positive definite linear system $Au = b$. The idea behind Preconditioned Conjugate Gradients is to apply the "original" Conjugate Gradient method to the transformed system

$$\tilde{A}\tilde{u} = \tilde{b} \; ,$$

where $\tilde{A} = P^{-1}AP^{-T}, \; u = P^{-T}\tilde{u}$ and $\tilde{b} = P^{-1}b$, and P is a nonsingular matrix. The matrix M defined by $M = PP^T$ is called the preconditioner. The resulting algorithm can be rewritten in such a way that only quantities without a ˜ tilde appear.

```
Preconditioned Conjugate Gradient method
    $u^0 = 0$ ;   $r^0 = b$                                        initialization
    for           $k = 1, 2, \ldots$ do                            $k$ is the iteration number
                  $z^{k-1} = M^{-1} r^{k-1}$                       preconditioning
                  if $k = 1$ do
                    $p^1 = z^0$
                  else
                    $\beta_k = \frac{(r^{k-1})^T z^{k-1}}{(r^{k-2})^T z^{k-2}}$    update of $p^k$
                    $p^k = z^{k-1} + \beta_k p^{k-1}$
                  end if
                  $\alpha_k = \frac{(r^{k-1})^T z^{k-1}}{(p^k)^T A p^k}$
                  $u^k = u^{k-1} + \alpha_k p^k$                   update iterate
                  $r^k = r^{k-1} - \alpha_k A p^k$                 update residual
    end for
```

Observations and properties for this algorithm are:

- it can be shown that the residuals and search directions satisfy:

$$\begin{aligned} (r^j)^T M^{-1} r^i &= 0 \quad , \quad i \neq j \ , \\ (p^j)^T (P^{-1} A P^{-T}) p^i &= 0 \quad , \quad i \neq j \ . \end{aligned}$$

- The denominators $(r^{k-2})^T z^{k-2} = (z^{k-2})^T M z^{k-2}$ never vanish for $r^{k-2} \neq 0$ because M is a positive definite matrix.

With respect to the matrix P we have the following requirements:

- the multiplication of $P^{-T} P^{-1}$ by a vector should be cheap (comparable with a matrix vector product using A). Otherwise one iteration of PCG is much more expensive than one iteration of CG and hence preconditioning leads to a more expensive algorithm.
- The matrix $P^{-1} A P^{-T}$ should have a favorable distribution of the eigenvalues. It is easy to show that the eigenvalues of $P^{-1} A P^{-T}$ are the same as for $P^{-T} P^{-1} A$ and $A P^{-T} P^{-1}$. So we can choose one of these matrices to study the spectrum.

In order to give more details on the last requirement we note that the iterate u^k obtained by PCG satisfies

$$u^k \in u^0 + K^k(P^{-T} P^{-1} A \ ; \ P^{-T} P^{-1} r^0), \text{ and} \tag{7.18}$$

$$\|u - u^k\|_A \leq 2 \left(\frac{\sqrt{\kappa_2(P^{-1} A P^{-T})} - 1}{\sqrt{\kappa_2(P^{-1} A P^{-T})} + 1} \right)^k \|u - u^0\|_A \ . \tag{7.19}$$

So a small condition number of $P^{-1} A P^{-T}$ leads to fast convergence. Two extreme choices of P show the possibilities of PCG. Choosing $P = I$ we have the original CG method, whereas if $P^T P = A$ the iterate u^1 is equal to u so PCG converges in one iteration. For a classical paper on the success of PCG we refer to [41]. In the following pages some typical preconditioners are discussed.

Diagonal scaling A simple choice for P is a diagonal matrix with diagonal elements $p_{ii} = \sqrt{a_{ii}}$. In [65] it has been shown that this choice minimizes the condition number of $P^{-1} A P^{-T}$ if P is a diagonal matrix. For this preconditioner it is advantageous to apply CG to $\tilde{A} \tilde{u} = \tilde{b}$. The reason is that $P^{-1} A P^{-T}$ is easily calculated. Furthermore $diag\ (\tilde{A}) = 1$ which saves n multiplications in the matrix vector product.

Basic iterative method The basic iterative methods use a splitting of the matrix $A = M - R$. In the beginning of Section 7.1.3 we showed that the i-th iterate y^i from a basic method is an element of $u^0 + K^i(M^{-1}A, M^{-1}r^0)$. Using this matrix M in the PCG method we see that the iterate u^i obtained by PCG satisfies the following inequality:

$$\|u - u^i\|_A = \min_{z \in K^i(M^{-1}A; M^{-1}r^0)} \|u - z\|_A \ .$$

This implies that $\|u - u^i\|_{P^{-1}AP^{-T}} \leq \|u - y^i\|_{P^{-1}AP^{-T}}$, so measured in the $\| \ . \ \|_{P^{-1}AP^{-T}}$ norm the error of a PCG iterate is less than the error of a corresponding result of a basic iterative method. The extra costs to compute a PCG iterate with respect to the basic iterate are in general negligible. This leads to the notion that any basic iterative method based on the splitting $A = M - R$ can be accelerated by the Conjugate Gradient method so long as M (the preconditioner) is symmetric and positive definite.

Incomplete decomposition This type of preconditioner is a combination of an iterative method and an approximate direct method. As illustration we use the model problem. The coefficient matrix of this problem $A \in \mathbb{R}^{N \times N}$ is a matrix with at most 5 nonzero elements per row. Furthermore the matrix is symmetric and positive definite. The nonzero diagonals are numbered as follows: m is number of grid points in the x-direction.

$$A = \begin{bmatrix} a_1 & b_1 & & c_1 & & & & \\ b_1 & a_2 & b_2 & & c_2 & & & \\ \vdots & \ddots & \ddots & & & \ddots & & \emptyset \\ c_1 & & b_m & a_{m+1} & b_{m+1} & & c_{m+1} & \\ & \ddots & \emptyset & \ddots & \ddots & \ddots & \emptyset & \ddots \\ & \emptyset & & & & & & \end{bmatrix} \qquad (7.20)$$

An optimal choice with respect to convergence is to take a lower triangular matrix L such that $A = LL^T$ and $P = L$ (L is the Cholesky factor). However it is well known that the zero elements in the band of A become non zero elements in the band of L. So the amount of work to construct L is large. With respect to memory we note that A can be stored in $3N$ memory positions, whereas L needs $m \ . \ N$ memory positions. For large problems the memory requirements are not easily fulfilled.

If the Cholesky factor L is calculated one observes that the absolute value of the elements in the band of L decreases considerably if the "distance" to the non zero elements of A increases. The non zero elements of L on positions where the elements of A are zero are called fill-in (elements). The observation of the decrease of fill-in motivates to discard fill-in elements entirely, which leads to an incomplete Cholesky decomposition of A. Since the Cholesky decomposition is very stable this is possible without break down for a large class of matrices. The matrix of our model problem is an M-matrix. Furthermore, we give a notation for the elements of L that should be kept to zero. The set of all pairs of indices of off-diagonal matrix entries is denoted by

$$Q_N = \{(i,j) | \ i \neq j \ , \ 1 \leq i \leq N \ , \ 1 \leq j \leq N \ \} \ .$$

The subset Q of Q_N are the places (i,j) where L should be zero. Now the following theorem can be proved:

Theorem 7.2.1 *If A is a symmetric M-matrix, there exists for each $Q \subset Q_N$ (with the property that $(i,j) \in Q$ implies $(j,i) \in Q$), a uniquely defined lower triangular matrix L and a symmetric nonnegative matrix R with $l_{ij} = 0$ if $(i,j) \in Q$ and $r_{ij} = 0$ if $(i,j) \notin Q$, such that the splitting $A = LL^T - R$ leads to a convergent iterative process*

$$LL^T u^{i+1} = Ru^i + b \quad \textit{for each choice} \ \ u^0 \ ,$$

where $u^i \to u = A^{-1}b$.

Proof (see [41]; p.151.)

After the matrix L is constructed it is used in the PCG algorithm. Note that in this algorithm multiplications by L^{-1} and L^{-T} are necessary. This is never done by forming L^{-1} or L^{-T}. It is easy to see that L^{-1} is a full matrix. If for instance one wants to calculate $z = L^{-1}r$ we compute z by solving the linear system $Lz = r$. This is cheap since L is a lower triangular matrix so the forward substitution algorithm can be used.

Example 7.2.1.1 *We consider the model problem and compute a slightly adapted incomplete Cholesky decomposition:* $A = LD^{-1}L^T - R$ *where the elements of the lower triangular matrix L and diagonal matrix D satisfy the following rules:*

a) $l_{ij} = 0$ *for all* (i,j) *where* $a_{ij} = 0 \quad i > j$,

b) $l_{ii} = d_{ii}$,

c) $(LD^{-1}L^T)_{ij} = a_{ij}$ *for all* (i,j) *where* $a_{ij} \neq 0 \quad i \geq j$.

In this example $Q_0 = \{(i,j)| \ |i-j| \neq 0, 1, m\}$. *If the elements of L are given as follows:*

$$L = \begin{bmatrix} \tilde{d}_1 & & & & & \\ \tilde{b}_1 & \tilde{d}_2 & & & & \\ & \ddots & \ddots & \emptyset & & \\ \tilde{c}_1 & & \tilde{b}_m & \tilde{d}_{m+1} & & \\ & \ddots & \emptyset & \ddots & \ddots & \\ \emptyset & & & & & \end{bmatrix} \tag{7.21}$$

it is easy to see that (using the notation as given in (7.20))

$$\left.\begin{array}{rcl} \tilde{d}_i & = & a_i - \dfrac{b_{i-1}^2}{\tilde{d}_{i-1}} - \dfrac{c_{i-m}^2}{\tilde{d}_{i-m}} \\ \tilde{b}_i & = & b_i \\ \tilde{c}_i & = & c_i \end{array}\right\} \quad i = 1, ..., N \ . \tag{7.22}$$

where elements that are not defined are replaced by zeros. For this example the amount of work for $P^{-T}P^{-1}$ *times a vector is comparable to the work to compute A times a vector. The combination of this incomplete Cholesky decomposition process with Conjugate Gradients is called the ICCG(0) method ([41]; p. 156). The 0 means that no extra diagonals are used for fill in. Note that this variant is very cheap with respect to memory: only one extra vector to store D is needed.*

Another successfull variant is obtained by a smaller set Q. In this variant the matrix L has three more diagonals than the lower triangular part of the original matrix A. This preconditioning is obtained for the choice

$$Q^3 = \{(i,j)| \ |i-j| \neq 0, 1, 2, m-2, m-1, m\}$$

For the formula's to obtain the decomposition we refer to ([41]; p. 156). This preconditioner combined with PCG is known as the ICCG(3) method. A drawback is that all the elements of L are different from the corresponding elements of A so 6 extra vectors are needed to store L in memory.

To give an idea of the power of the ICCG methods we have copied some results from [41].

Example 7.2.1.2 *As a first example we consider the model problem, where the boundary conditions are somewhat different:*

$$\begin{array}{lll} \dfrac{\partial u}{\partial x}(x,y) = 0 & \textit{for} & \left\{ \begin{array}{ll} x = 0 \ , & y \in [0,1] \\ x = 1 \ , & y \in [0,1] \end{array} \right. \ , \\ \dfrac{\partial u}{\partial y}(x,y) = 0 & \textit{for} & y = 1 \ , \ x \in [0,1] \ , \\ u(x,y) = 1 & \textit{for} & y = 0 \ , \ x \in [0,1] \ . \end{array}$$

The distribution of the grid points is equidistant with $h = \frac{1}{31}$. The results for CG, ICCG(0) and ICCG(3) are plotted in Figure 7.4.

From inequality (7.19) it follows that the rate of convergence can be bounded by

$$\overline{\rho} = \frac{\sqrt{\kappa_2(P^{-1}AP^{-T})} - 1}{\sqrt{\kappa_2(P^{-1}AP^{-T})} + 1}. \tag{7.23}$$

To obtain a better insight in the fast convergence of ICCG(0) and ICCG(3) the eigenvalues of A, $(L_0L_0^T)^{-1}A$ and $(L_3L_3^T)^{-1}A$ are computed and given in Figure 7.5. For this result given in [41] a small matrix of order $n = 36$ is used, so all eigenvalues can be calculated.

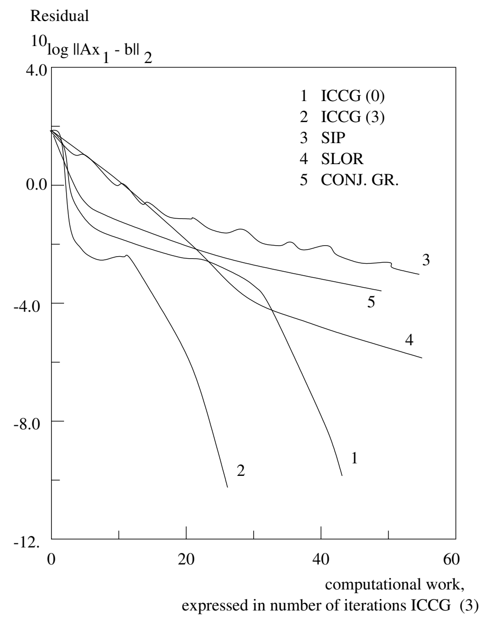


Figure 7.4: The results for the CG, ICCG(0) and ICCG(3) methods, compared with SIP (Strongly Implicit Procedure) and SLOR (Successive Line Over Relaxation method)

The eigenvalues as given in Figure 7.5 can be substituted in formula (7.23). We then obtain

$$\begin{array}{lll} \overline{\rho} = 0.84 & \textit{for} & \textit{CG}\ , \\ \overline{\rho} = 0.53 & \textit{for} & \textit{ICCG(0)}\ , \\ \overline{\rho} = 0.23 & \textit{for} & \textit{ICCG(3)}\ , \end{array} \tag{7.24}$$

which explains the fast convergence of the PCG variants.

In our explanation of the convergence behavior we have also used the notion of Ritzvalues. Applying these ideas to the given methods we note the following:

- For CG the eigenvalues of A are more or less equidistantly distributed. So if a Ritzvalue has converged we only expect a small decrease in the rate of convergence. This agrees with the results given in Figure 7.4, the CG method has a linear convergent behavior.
- For the PCG method the eigenvalue distribution is completely different. Looking to the spectrum of $(L_3L_3^T)^{-1}A$ we see that $\lambda_{36} = 0.446$ is the smallest eigenvalue. The distance between λ_{36} and the other eigenvalues is relatively large which implies that there is a fast convergence of the smallest Ritzvalue to λ_{36}. Furthermore, if the smallest Ritzvalue is a reasonable approximation of λ_{36} the effective condition number is much less than the original condition number. Thus superlinear convergence is expected. This again agrees very well with the results given in Figure 7.4.

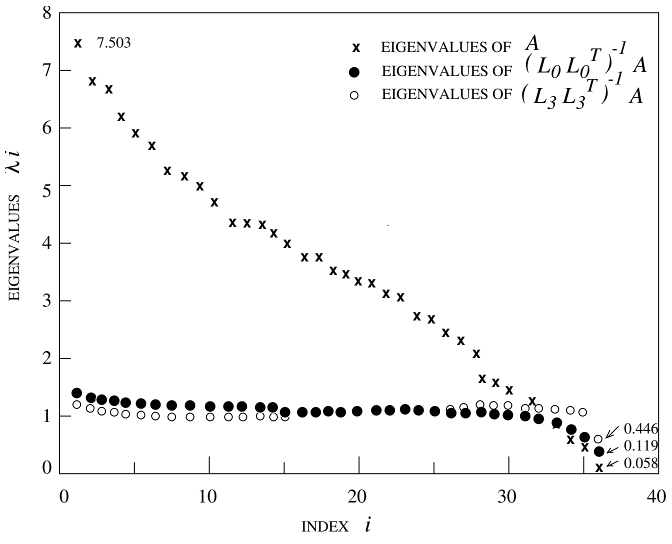


Figure 7.5: The eigenvalues of A, $(L_0L_0^T)^{-1}A$ and $(L_3L_3^T)^{-1}A$.

So, the faster convergence of ICCG(3) comes from a smaller condition number and a more favorable distribution of the internal eigenvalues.

Finally, the influence of the order of the matrix on the number of iterations required to reach a certain precision was checked for both ICCG(0) and ICCG(3). Therefore several uniform rectangular meshes have been chosen, with mesh spacings varying from $\sim 1/10$ up to $\sim 1/50$. This resulted in linear systems with matrices of order 100 up to about 2500. In each case it was determined how many iterations were necessary, to reduce the ∞ norm of the residual vector below some fixed small number ε. In Figure 7.6 the number of iterations are plotted against the order of the matrices for $\varepsilon = 10^{-2}$, $\varepsilon = 10^{-6}$ and $\varepsilon = 10^{-10}$. It can be seen that the number of iterations, necessary to get the residual vector sufficiently small, increases only slowly for increasing order of the matrix. The dependence of $\kappa_2(A)$ for this problem is $O(\frac{1}{h^2})$. For ICCG preconditioning it can be shown that there is a cluster of large eigenvalues of $(L_0L_0^T)^{-1}A$ in the vicinity of 1, whereas the small eigenvalues are of order $O(h^2)$ and the gaps between them are relatively large. So also for ICCG(0) the condition number is $O(\frac{1}{h^2})$. Faster convergence can be explained by the fact that the constant in front of $\frac{1}{h^2}$ is less for the ICCG(0) preconditioned system than for A and the distribution of the internal eigenvalues is much better. Therefore, superlinear convergence sets in after a small number of iterations.

The success of the ICCG method has led to many variants. In the following we describe two of them MICCG(0) given in [31] (MICCG means Modified ICCG) and RICCG(0) given in [4] (RICCG means Relaxed ICCG).

MICCG In the MICCG method the MIC preconditioner is constructed by slightly adapted rules. Again A is split as follows $A = LD^{-1}L^T - R$, where L and D satisfy the following rules:

a) $l_{ij} = 0$ for all (i,j) where $a_{ij} = 0 \;\; i > j$,

b) $l_{ii} = d_{ii}$,

c) rowsum $(LD^{-1}L^T)$=rowsum(A) for all rows and $(LD^{-1}L^T)_{ij} = a_{ij}$ for all (i,j) where $a_{ij} \neq 0 \;\; i > j$.

Consequence of c):$LD^{-1}L^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = A \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ so $(LD^{-1}L^T)^{-1}A \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$. This means that if $Ax = b$ and x and/or b are slowly varying vectors, this incomplete Cholesky decomposition is a very good

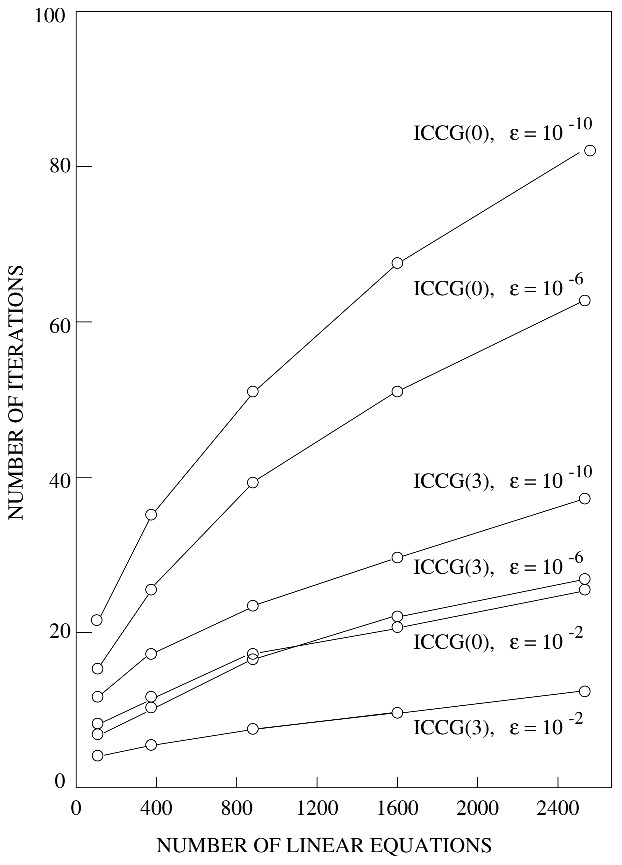


Figure 7.6: Effect of number of equations on the rate of convergence

approximation for the inverse of A with respect to x and/or b.
Using the same notation of L as given in (7.21) we obtain

$$\left.\begin{array}{rcl} \tilde{d}_i &=& a_i - (b_{i-1}+c_{i-1})\dfrac{b_{i-1}}{\tilde{d}_{i-1}} - (b_{i-m}+c_{i-m})\dfrac{c_{i-m}}{\tilde{d}_{i-m}} \\ \tilde{b}_i &=& b_i \\ \tilde{c}_i &=& c_i \end{array}\right\} \quad i = 1,..,N \tag{7.25}$$

It can be proved that for this preconditioning there is a cluster of small eigenvalues in the vicinity of 1 and the largest eigenvalues are of order $\frac{1}{h}$ and have large gap ratio's. So the condition number is $O(1/h)$.

In many problems the initial iterations of MICCG(0) converge faster than ICCG(0). Thereafter for both methods superlinear convergence sets in. Using MICCG the largest Ritzvalues are good approximations of the largest eigenvalues of the preconditioned matrix. A drawback of MICCG is that due to rounding errors components in eigenvectors related to large eigenvalues return after some iterations. This deteriorates the rate of convergence. So if many iterations are needed ICCG can be better than MICCG.

In order to combine the advantages of both methods the RIC preconditioner is proposed in [4], which is an average of the IC and MIC preconditioner. For the details we refer to [4]. Only the algorithm is given: choose the average parameter $\alpha \in [0,1]$ then $\tilde{d}_i, \tilde{b}_i$ and $\tilde{c}_i$ are given by:

$$\left.\begin{array}{rcl} \tilde{d}_i &=& a_i - (b_{i-1}+\alpha c_{i-1})\frac{b_{i-1}}{\tilde{d}_{i-1}} - (\alpha b_{i-m}+c_{i-m})\frac{c_{i-m}}{\tilde{d}_{i-m}} \\ \tilde{b}_i &=& b_i \\ \tilde{c}_i &=& c_i \end{array}\right\} \quad i = 1,...,N \tag{7.26}$$

However, the question how to choose α remains? In Figure 7.7 which is copied from [68] a typical convergence behavior as function of α is given. This motivates the choice $\alpha = 0.95$, which leads to a very good rate of convergence on a wide range of problems.

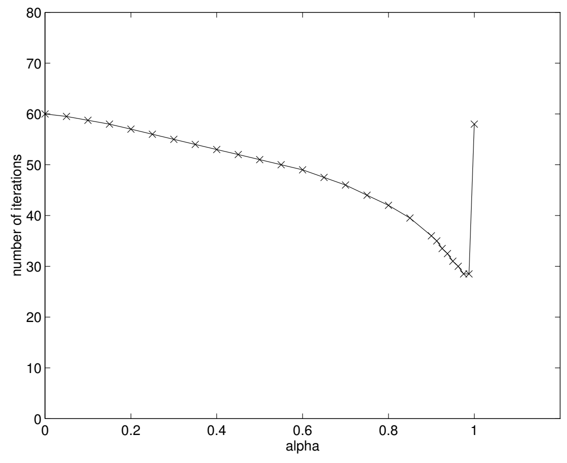


Figure 7.7: Convergence in relation to α

Diagonal scaling The above given preconditioners IC, MIC and RIC can be optimized with respect to work. One way is to look at the explicitly preconditioned system:

$$D^{1/2}L^{-1}AL^{-T}D^{1/2}y = D^{1/2}L^{-1}b \tag{7.27}$$

Applying CG to this system one has to solve lower triangular systems of equations with matrix $LD^{-1/2}$. The main diagonal of this matrix is equal to $D^{1/2}$. It saves time if we can change the system in such a way that the main diagonal is equal to the identity matrix. One idea is to replace (7.27) by

$$D^{1/2}L^{-1}D^{1/2}D^{-1/2}AD^{-1/2}D^{1/2}L^{-T}D^{1/2}y = D^{1/2}L^{-1}D^{1/2}D^{-1/2}b\ .$$

with $\tilde{A} = D^{-1/2}AD^{-1/2}$, $\tilde{L} = D^{-1/2}LD^{-1/2}$ and $\tilde{b} = D^{-1/2}b$ we obtain

$$\tilde{L}^{-1}\tilde{A}\tilde{L}^{-T}y = \tilde{L}\tilde{b} \, . \tag{7.28}$$

Note that $\tilde{L}_{ii} = 1$ for $i = 1, ..., N$. PCG now is the application of CG to this preconditioned system.

Eisenstat implementation In this section we restrict ourselves to the IC(0), MIC(0) and RIC(0) preconditioner. We have already noted that the amount of work of one PCG iteration is approximately 2 times as large as for a CG iteration. In [15] it is shown that much of the extra work can be avoided. If CG is applied to (7.28) products of the following form are calculated: $v^{j+1} = \tilde{L}^{-1}\tilde{A}\tilde{L}^{-T}v^j$. For the preconditioners used, the off-diagonal part of $\tilde{L}$ is equal to the off-diagonal part of $\tilde{A}$. Using this we obtain:

$$\begin{aligned} v^{j+1} &= \tilde{L}^{-1}\tilde{A}\tilde{L}^{-T}v^j = \tilde{L}^{-1}(\tilde{L} + \tilde{A} - \tilde{L} - \tilde{L}^T + \tilde{L}^T)\tilde{L}^{-T}v^j \\ &= \tilde{L}^{-T}v^j + \tilde{L}^{-1}(v^j + (diag\ (\tilde{A}) - 2I)\tilde{L}^{-T}v^j) \end{aligned} \tag{7.29}$$

So v^{j+1} can be calculated by a multiplication by $\tilde{L}^{-T}$ and $\tilde{L}^{-1}$ and some vector operations. The saving in CPU time is the time to calculate the product of A times a vector. Now one iteration of PCG costs approximately the same as one iteration of CG.

General stencils In practical problems the stencils in finite element methods may be larger or more irregularly distributed than in finite difference methods. The same type of preconditioners can be used. However, there are some differences. We restrict ourselves to the IC(0) preconditioner. For the five point stencil we see that the off-diagonal part of L is equal to the strictly lower triangular part of A. For general stencils this property does not hold. Drawbacks are: All the elements of L should be stored, so the memory requirements of PCG are two times as large as for CG. Furthermore, the Eisenstat implementation can no longer be used. This motivates another preconditioner constructed by the following rules:

ICD (Incomplete Cholesky restricted to the Diagonal).
A is again splitted as $A = LD^{-1}L^T - R$ and L and D satisfy the following rules:

a) $l_{ij} = 0$ for all (i, j) where $a_{ij} = 0 \quad i > j$

b) $l_{ii} = d_{ii}, \quad i = 1, ..., N$

c) $l_{ij} = a_{ij}$ for all (i, j) where $a_{ij} \neq 0 \quad i > j$
$(LD^{-1}L^T)_{ii} = a_{ii} \quad i = 1, ..., N.$

This enables us to save memory (only the diagonal D should be stored) and CPU time (since now Eisenstat implementation can be used) per iteration. For large problems the rate of convergence of ICD is worse than for IC. Also MICD and RICD preconditioners can be given.

7.2.2 Exercises

Exercise 7.2.1 Derive the preconditioned CG method using the CG method applied to $\tilde{A}\tilde{u} = \tilde{b}$.

Exercise 7.2.2 a) Show that the formula's given in (7.22) are correct.

b) Show that the formula's given in (7.25) are correct.

Exercise 7.2.3 a) Suppose that $a_i = 4$ and $b_i = -1$. Show that $\lim\limits_{i \to \infty} \tilde{d}_i = 2 + \sqrt{3}$, where $\tilde{d}_i$ is as defined in (7.22).

b) Do the same for $a_i = 4$, $b_i = -1$ and $c_i = -1$ with $m = 10$, and show that $\lim\limits_{i \to \infty} \tilde{d}_i = 2 + \sqrt{2}$.

c) Prove that the $LD^{-1}L^T$ decomposition (7.22) exists if $a_i = a, b_i = b, c_i = c$ and A is diagonally dominant.

Exercise 7.2.4 A practical exercise
Use as test matrices:

$$[a, f] = poisson(30, 30, 0, 0,' central')$$

a) Adapt the matlab cg algorithm such that preconditioning is included. Use a diagonal preconditioner and compare the number of iterations with cg without preconditioner.

b) Use the formula's given in (7.22) to obtain an incomplete $LD^{-1}L^T$ decomposition of A. Make a plot of the diagonal elements of D. Can you understand this plot?

c) Use the $LD^{-1}L^T$ preconditioner in the method given in (a) and compare the convergence behavior with that of the diagonal preconditioner.

7.3 Krylov Subspace Methods for General Matrices

In Section 7.1 we have discussed the Conjugate Gradient method. This Krylov subspace method can only be used if the coefficient matrix is symmetric and positive definite (SPD). In this section we discuss Krylov subspace methods for general matrices. For these matrices we consider various iterative methods. It appears that there is no method which is the best for all cases. This is in contrast with the symmetric positive definite case. In Section 7.3.6 we give some guidelines for choosing an appropriate method for a given system of equations.

7.3.1 Iterative Methods for General Matrices

In this section we consider iterative methods to solve $Au = b$ where the only requirement is that $A \in \mathbb{R}^{N \times N}$ is nonsingular. In the SPD case we have seen that CG has the following three nice properties:

- the approximation u^i is an element of $K^i(A; r^0)$,
- optimality, the A-norm of the error is minimal,
- short recurrences, only the results of one foregoing step are necessary; work and memory do not increase for an increasing number of iterations.

It has been shown in [18] that it is impossible to obtain a Krylov method, which has these properties for general matrices. So either the method has an optimality property but long recurrences, or no optimality and short recurrences, or it is not based on $K^i(A; r^0)$. Some surveys on general iteration methods have been published: [7], [25] Section 10.4, [20], [53], [29], and [5].

It appears that there are essentially three different classes of methods to solve non-symmetric linear systems, while maintaining some kind of orthogonality between the residuals:

1. Solve the normal equations $A^T Au = A^T b$ with Conjugate Gradients.
2. Construct a basis for the Krylov subspace by a 3-term bi-orthogonality relation.
3. Make all the residuals explicitly orthogonal in order to have an orthogonal basis for the Krylov subspace.

These classes form the subject of the following sections. An introduction and comparison of these classes is given in [42].

7.3.2 CG Applied to the Normal Equations

The first idea is to apply CG to the normal equations

$$A^T Au = A^T b, \tag{7.30}$$

or

$$AA^T y = b \quad \text{with} \quad u = A^T y. \tag{7.31}$$

When A is nonsingular $A^T A$ is symmetric positive definite and CG can be used. This method is denoted by the CGNR (Conjugate Gradient Normal Residual) method. All properties and theoretical results for CG can be used. There are, however, some drawbacks. First of all, the rate of convergence now depends on $\kappa_2(A^T A) = \kappa_2(A)^2$. In many applications $\kappa_2(A)^2$ is extremely large. Hence, the convergence of CG applied to (7.30) is very slow. Another difference is that the convergence of CG applied to $Au = b$ depends on the eigenvalues of A whereas the convergence of CG applied to (7.30) depends on the eigenvalues of $A^T A$, which are equal to the singular values of A squared.

Per iteration a multiplication with A and A^T are necessary, so the amount of work per iteration is approximately two times as much as for the CG method. Furthermore, in several (FEM, parallel) applications Av

is easily obtained but $A^T v$ is not due to an unstructured grid and the corresponding data structure. Finally, not only the convergence depends on $\kappa_2(A)^2$ but also the behavior due to rounding errors. To improve the numerical stability it is suggested in [6] to replace inner products like $p^T A^T Ap$ by $(Ap)^T Ap$. Another improvement is the LSQR (Least Squares QR) method proposed in [44]. This method is based on the application of the Lanczos method to the auxiliary system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ u \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} .$$

LSQR is a very robust algorithm. It uses reliable stopping criteria and estimates for the error and the condition number of A.

7.3.3 Bi-CG Type Methods

In this type of method we have that u^i is an element of $K^i(A; r^0)$, short recurrences but we do not have an optimality property. We have seen that CG is based on the Lanczos algorithm. The Lanczos algorithm for non-symmetric matrices is called the Bi-Lanczos algorithm. Bi-CG type methods are based on Bi-Lanczos. In the Lanczos method we look for a matrix Q such that $Q^T Q = I$ and

$$Q^T AQ = T \quad \text{tridiagonal} .$$

In the Bi-Lanczos algorithm we construct a similarity transformation X such that

$$X^{-1}AX = T \quad \text{tridiagonal} .$$

To obtain this matrix we construct a basis $r^0, ..., r^{i-1}$, which are the residuals, for $K^i(A; r^0)$ such that $r^j \perp K^j(A^T; s^0)$ and $s^0, ..., s^{i-1}$ form a basis for $K^i(A^T; s^0)$ such that $s_j \perp K^j(A; r^0)$, so the sequences $\{r^i\}$ and $\{s^i\}$ are bi-orthogonal. Using these properties and the definitions $R_k = [r^0 ... r^{k-1}]$, $S_k = [s^0 ... s^{k-1}]$ the following relations can be proved [69]:

$$AR_k = R_k T_k + \alpha_k r^k e_k^T \ , \tag{7.32}$$

and

$$S_k^T (Au^k - b) = 0 \ .$$

Using (7.32), $r^0 = b$ and $u^k = R_k y$ we obtain

$$S_k^T R_k T_k y = s_0 r_0^T e_1. \tag{7.33}$$

Since $S_k^T R_k$ is a diagonal matrix with diagonal elements $(r^j)^T s^j$, we find if all these diagonal elements are nonzero,

$$T_k y = e_1 \ , \quad u^k = R_k y \ .$$

This algorithm fails when a diagonal element of $S_k^T R_k$ becomes (nearly) zero, because these elements are used to normalize the vectors s_j (compare [25] §9.3.6). This is called a serious (near) break down. The way to get around this difficulty is the so-called look-ahead strategy. For details on look-ahead we refer to [47], and [21]. Another way to avoid break down is to restart as soon as a diagonal element gets small. This strategy is very simple, but one should realize that at a restart the Krylov subspace that has been built up, is thrown away, which destroys possibilities for faster (superlinear) convergence.

Bi-CG (Bi-Conjugate Gradient) As has been shown for Conjugate Gradients, the LU decomposition of the tridiagonal system T_k can be updated from iteration to iteration. It leads to a recursive update of the solution vector, which avoids saving all intermediate r and s vectors. This variant of Bi-Lanczos is usually called Bi-Conjugate Gradients, or shortly Bi-CG [19]. Of course, one can in general not be sure that an LU decomposition (without pivoting) of the tridiagonal matrix T_k exists, and if it does not exist a serious break down of the Bi-CG algorithm will occur. This break down can be avoided in the Bi-Lanczos formulation of this iterative solution scheme. The algorithm is given below.

```
Bi-CG method
    $u^0$ is an initial guess; $r^0 = b - Au^0$;
    $\hat{r}^0$ is an arbitrary vector, such that $(r^0, \hat{r}^0) \neq 0$ ,
    e.g., $\hat{r}^0 = r^0$
    $\rho_0 = 1$ $\hat{p}^0 = p^0 = 0$
    for $i = 1, 2, ...$
        $\rho_i = (\hat{r}^{i-1}, r^{i-1})$ ; $\beta_i = (\rho_i / \rho_{i-1})$ ;
        $p^i = r^{i-1} + \beta_i p^{i-1}$ ;
        $\hat{p}^i = \hat{r}^{i-1} + \beta_i \hat{p}^{i-1}$ ;
        $v^i = Ap^i$
        $\alpha_i = \rho_i / (\hat{p}^i, v^i)$;
        $u^i = u^{i-1} + \alpha_i p^i$
        $r^i = r^{i-1} - \alpha_i v^i$
        $\hat{r}^i = \hat{r}^{i-1} - \alpha_i A^T \hat{p}^i$
    end for
```

Note that for symmetric matrices Bi-Lanczos generates the same solution as Lanczos, provided that $s^0 = r^0$, and under the same condition, Bi-CG delivers the same iterates as CG, for symmetric positive definite matrices. However, the Bi-orthogonal variants do so at the cost of two matrix vector operations per iteration. The method has been superseded by CGS and Bi-CGSTAB.

CGS (Conjugate Gradient Squared) The bi-conjugate gradient residual vectors can be written as $r^j = P_j(A)r^0$ and $\hat{r}^j = P_j(A^T)\hat{r}^0$, where P_j is a polynomial of degree j such that $P_j(0) = 1$. Due to the bi-orthogonality relation it follows that

$$(r^j, \hat{r}^i) = (P_j(A)r^0, P_i(A^T)\hat{r}^0) = (P_i(A)P_j(A)r^0, \hat{r}^0) = 0 \text{ , for } i < j.$$

The iteration parameters for bi-conjugate gradients are computed from innerproducts like above. Sonneveld observed in [58] that one can also construct the vectors $r^j = P_j^2(A)r^0$, using only the latter form of the innerproduct for recovering the bi-conjugate gradients parameters (that implicitly define the polynomial P_j). By doing so, neither the vectors $\hat{r}^j$ have to be formed, nor is it necessary to multiply by the matrix A^T. The resulting CGS [58] method works well in general for many non-symmetric linear problems. It often converges faster than Bi-CG (about twice as fast in some cases). However, CGS usually shows a very irregular convergence behavior. This behavior can even lead to cancellation and a spoiled solution [68].

The following scheme carries out the CGS process for the solution of $Au = b$, with a given preconditioner M (to be discussed next):

```
Conjugate Gradient Squared method
    $u^0$ is an initial guess; $r^0 = b - Au^0$;
    $\tilde{r}^0$ is an arbitrary vector, such that
    $(r^0, \tilde{r}^0) \neq 0$ ,
    e.g., $\tilde{r}^0 = r^0$ ; $\rho_0 = (r^0, \tilde{r}^0)$ ;
    $\beta_{-1} = \rho_0$ ; $p_{-1} = q_0 = 0$ ;
    for $i = 0, 1, 2, ...$ do
        $w^i = r^i + \beta_{i-1} q^i$ ;
        $p^i = w^i + \beta_{i-1}(q^i + \beta_{i-1} p^{i-1})$ ;
        $\hat{p} = M^{-1} p^i$ ;
        $\hat{v} = A\hat{p}$ ;
        $\alpha_i = \frac{\rho_i}{(\tilde{r}^0, \hat{v})}$ ;
        $q^{i+1} = w^i - \alpha_i \hat{v}$ ;
        $\hat{w} = M^{-1}(w^i + q^{i+1})$
        $u^{i+1} = u^i + \alpha_i \hat{w}$ ;
        if $u^{i+1}$ is accurate enough then quit;
        $r^{i+1} = r^i - \alpha_i A\hat{w}$ ;
        $\rho_{i+1} = (\tilde{r}^0, r^{i+1})$ ;
        if $\rho_{i+1} = 0$ then method fails to converge!;
        $\beta_i = \frac{\rho_{i+1}}{\rho_i}$ ;
    end for
```

In exact arithmetic, the α_j and β_j are the same as those generated by Bi-CG. Therefore, they can be used to compute the Petrov-Galerkin approximations for eigenvalues of A.

Bi-CGSTAB (Bi-CG Stabilized) Bi-CGSTAB [69] is based on the following observation. Instead of squaring the Bi-CG polynomial, we can construct another iteration method, by which iterates u^i are generated so that $r^i = \tilde{P}_i(A)P_i(A)r^0$ with another i^{th} degree polynomial $\tilde{P}_i$. An obvious possibility is to take for $\tilde{P}_j$ a polynomial of the form

$$Q_i(x) = (1 - \omega_1 x)(1 - \omega_2 x)...(1 - \omega_i x) \ ,$$

and to select suitable constants $\omega_j \in \mathbb{R}$. This expression leads to an almost trivial recurrence relation for the Q_i. In Bi-CGSTAB, ω_j in the j^{th} iteration step is chosen as to minimize r^j, with respect to ω_j, for residuals that can be written as $r^j = Q_j(A)P_j(A)r^0$.
The preconditioned Bi-CGSTAB algorithm for solving the linear system $Au = b$, with preconditioning M reads as follows:

Bi-CGSTAB method

```
u^0 is an initial guess; $r^0 = b - Au^0$;
$\bar{r}^0$ is an arbitrary vector, such that $(\bar{r}^0, r^0) \neq 0$, e.g., $\bar{r}^0 = r^0$ ;
$\rho_{-1} = \alpha_{-1} = \omega_{-1} = 1$ ;
$v^{-1} = p^{-1} = 0$ ;
for $i = 0, 1, 2, ...$ do
    $\rho_i = (\bar{r}^0, r^i)$ ; $\beta_{i-1} = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1})$ ;
    $p^i = r^i + \beta_{i-1}(p^{i-1} - \omega_{i-1}v^{i-1})$ ;
    $\hat{p} = M^{-1}p^i$ ;
    $v^i = A\hat{p}$ ;
    $\alpha_i = \rho_i/(\bar{r}^0, v^i)$ ;
    $s = r^i - \alpha_i v^i$ ;
    if $\|s\|$ small enough then
        $u^{i+1} = u^i + \alpha_i\hat{p}$ ; quit;
    $z = M^{-1}s$ ;
    $t = Az$ ;
    $\omega_i = (t, s)/(t, t)$ ;
    $u^{i+1} = u^i + \alpha_i\hat{p} + \omega_i z$ ;
    if $u^{i+1}$ is accurate enough then quit;
    $r^{i+1} = s - \omega_i t$ ;
end for
```

The matrix M in this scheme represents the preconditioning matrix and the way of preconditioning [69]. The above scheme in fact carries out the Bi-CGSTAB procedure for the explicitly postconditioned linear system

$$AM^{-1}y = b \ ,$$

but the vectors y^i has been transformed back to the vectors u^i corresponding to the original system $Au = b$. Compared to CGS two extra innerproducts need to be calculated.

In exact arithmetic, the α_j and β_j have the same values as those generated by Bi-CG and CGS. Hence, they can be used to extract eigenvalue approximations for the eigenvalues of A.

An advantage of these methods is that they use *short recurrences*. A disadvantage is that there is only a *semi-optimality property*. As a result of this, more matrix vector products are needed and no convergence properties have been proved. In experiments we see that the convergence behavior looks like CG for a large class of problems. However, the influence of rounding errors is much more significant than for CG. Small changes in the algorithm can lead to instabilities. Finally, it is always necessary to compare the norm of the updated residual to the exact residual $\|b - Au^k\|_2$. If "near" break down had occurred in the algorithm these quantities may differ by several orders of magnitude. In such a case the method should be restarted.

7.3.4 GMRES Type Methods

These methods are based on u^i is an element of $K^i(A; r^0)$, *long recurrences*, but have certain *optimality properties*. The long recurrences imply that the amount of work per iteration and the required memory grow for an increasing number of iterations. Consequently, in practice one cannot afford to run the full algorithm, and it becomes necessary to use *restarts* or to *truncate* vector recursions. In this section we describe GMRES and GCR.

GMRES In this method, Arnoldi's method is used for computing an orthonormal basis $\{v^1, ..., v^k\}$ of the Krylov subspace $K^k(A; r^0)$. The modified Gram-Schmidt version of Arnoldi's method can be described as follows [54]:

```
GMRES method
Choose $u^0$ and compute $r^0 = b - Au^0$ and $v^1 = r^0/\|r^0\|_2$,
        for $j = 1, ..., k$ do:
          $v^{j+1} = Av^j$
          for $i = 1, .., j$ do:
            $h_{ij} := (v^{j+1})^T v^i$ , $v^{j+1} := v^{j+1} - h_{ij} v^i$ ,
          end for
          $h_{j+1,j} := \|v^{j+1}\|_2$ , $v^{j+1} := v^{j+1}/h_{j+1,j}$
        end for
The entries of upper $k + 1 \times k$ Hessenberg matrix $\bar{H}_k$ are the scalars $h_{ij}$.
```

In GMRES (General Minimal RESidual method) the approximate solution $u^k = u^0 + z^k$ with $z^k \in K^k(A; r^0)$ is such that

$$\|r^k\|_2 = \|b - Au^k\|_2 = \min_{z \in K^k(A;r^0)} \|r^0 - Az\|_2. \tag{7.34}$$

As a consequence of (7.34) it appears that r^k is orthogonal to $AK^k(A; r^0)$, so $r^k \perp K^k(A; Ar^0)$. If A is symmetric the GMRES method is equivalent to the MINRES method (described in [43]). For the matrix $\bar{H}_k$ it follows that $AV_k = V_{k+1}\bar{H}_k$ where the $N \times k$ matrix V_k is defined by $V_k = [v^1, ..., v^k]$. With this equation it is shown in [54] that $u^k = u^0 + V_k y^k$ where y^k is the solution of the following least squares problem:

$$\|\beta e_1 - \bar{H}_k y^k\|_2 = \min_{y \in \mathbb{R}^k} \|\beta e_1 - \bar{H}_k y\|_2, \tag{7.35}$$

with $\beta = \|r^0\|_2$ and e_1 is the first unit vector in $\mathbb{R}^{k+1}$. GMRES is a stable method and break down does not occur. If $h_{j+1,j} = 0$ then $u^j = u$ so this is a "lucky" break down (see [54]; Section 3.4).

Due to the optimality (see inequality (7.34)) convergence proofs exist [54]. If the eigenvalues of A are real the same bounds on the norm of the residual can be proved as for the CG method. For a more general eigenvalue distribution we shall give one result in the following theorem. Let P_m be the space of all polynomials of degree less than m and let $\sigma = \{\lambda_1, ..., \lambda_N\}$ represent the spectrum of A.

Theorem 7.3.1 *Suppose that A is diagonalizable so that $A = XDX^{-1}$ and let*

$$\varepsilon^{(m)} = \min_{\substack{p \in P_m \\ p(0)=1}} \max_{\lambda_i \in \sigma} |p(\lambda_i)|$$

Then the residual norm of the m-th iterate satisfies:

$$\|r^m\|_2 \leq K(X)\varepsilon^{(m)}\|r^0\|_2 \tag{7.36}$$

where $K(X) = \|X\|_2\|X^{-1}\|_2$. If furthermore all eigenvalues are enclosed in a circle centered at $C \in \mathbb{R}$ with $C > 0$ and having radius R with $C > R$, then

$$\varepsilon^{(m)} \leq \left(\frac{R}{C}\right)^m . \tag{7.37}$$

Proof: see [54]; p. 866.

Note that $K(X)$ can be very large. In such a case bound (7.36) is not usefull [30]. For GMRES we see in many cases a superlinear convergence behavior comparable to CG. The same type of results have been proved for GMRES [70]. As we have already noted in the beginning, work per iteration and memory requirements increase for an increasing number of iterations. In this algorithm the Arnoldi process requires k vectors in memory in the k-th iteration. Furthermore $2k^2 \cdot N$ flops are needed for the total Gram Schmidt process. To restrict work and memory requirements one stops GMRES after m iterations, forms the approximate solution and uses this as a starting vector for a following application of GMRES. This is denoted by the GMRES(m) procedure (not restarted GMRES is denoted by *full* GMRES). However, restarting destroys many of the nice properties of full GMRES, for instance the optimality property is only valid inside a GMRES(m) step and the superlinear convergence behavior is lost. This is a severe drawback of the GMRES(m) method [17].

GCR Slightly earlier than GMRES, the GCR method was proposed in [16] (Generalized Conjugate Residual method). The algorithm is given as follows:

```
GCR algorithm
Choose $u^0$, compute $r^0 = b - Au^0$

    for $i = 1, 2, ...$ do
        $s^i = r^{i-1}$ ,
        $v^i = As^i$ ,
        for $j = 1, ..., i-1$ do
            $\alpha = (v^j, v^i)$ ,
            $s^i := s^i - \alpha s^j$ ,  $v^i := v^i - \alpha v^j$ ,
        end for
        $s^i := s^i / \|v^i\|_2$ ,  $v^i := v^i / \|v^i\|_2$
        $\beta = (v^i, r^{i-1})$ ;
        $u^i := u^{i-1} + \beta s^i$ ;
        $r^i := r^{i-1} - \beta v^i$ ;
    end for
```

The storage of s^i and v^i costs two times as much memory as for GMRES. The rate of convergence of GCR and GMRES is comparable. However, there are examples where GCR may break down. So, when comparing full GMRES and full GCR the first one is to be preferred in many applications.
When the required memory is not available GCR can be restarted. However, another strategy is possible which is known as *truncation*. An example of this is to replace the j-loop by

```
for $j = i - m, ..., i$ do
```

Now $2m$ vectors are needed in memory. Other truncation variants to discard search directions are possible. In general, we see that truncated methods have a better convergence behavior, especially if superlinear convergence plays an important role. If restarting or truncation is necessary truncated GCR is in general better than restarted GMRES. For convergence results and other properties we refer to [16].

7.3.5 Hybrid Methods

The most popular methods are the the Bi-CG-type methods and the GMRES-type methods. However, both classes have their drawbacks. In this section we consider two hybrid methods, combinations of Krylov methods, which have an optimality property and which recurrences are reasonably short. We consider the methods: IDR(s) from the Bi-CG-type class and GMRESR from the GMRES-type class.

IDR(s) (Induced Dimension Reduction) The IDR method has already been proposed in 1980 [74]. Analysis of IDR reveals a close relation with Bi-CG. It has been shown that the iteration polynomial constructed by IDR is the product of the Bi-CG polynomial with another, locally minimizing polynomial. Sonneveld's observation that the Bi-CG polynomial could be combined with another polynomial without transpose-matrix-vector multiplications led to the development first of CGS and later of Bi-CGSTAB.

Over the years, CGS and Bi-CGSTAB have completely overshadowed IDR, which is now practically forgotten, except perhaps as the predecessor of CGS. This is unfortunate since, although there is a clear relation between CG-type methods and the original IDR method, the underlying ideas are completely different. This suggests that by exploiting the differences new methods may be developed.

Bi-CG, CGS, and Bi-CGSTAB are essentially based on the computation of two mutually biorthogonal bases for the Krylov subspaces based on A and A^T. The "S"-part in CGS and the "STAB"-part in Bi-CGSTAB are different ways of making more efficient use of the A^T-related information. The finiteness of these methods (in exact arithmetic) comes from the finiteness of any basis for finite dimensional space.

The IDR method, on the other hand, generates residuals that are forced to be in subspaces of decreasing dimension and at the end a 0-dimensional space remains, which imply that the residual is the zero vector.

IDR(s) describes a recently developed class of methods [60], which generalizes the Induced Dimension Reduction idea further. Where in IDR only a subspace of dimension 1 is used to reduce the space which contains the residual vector, IDR(s) uses an s-dimensional subspace. For modestly large choices of s, $s \leq 10$ a robust and efficient method is obtained. The method is at least as fast as Bi-CGSTAB but for hard problems it can be 4-10 times faster. Furthermore, it has been shown that the convergence of IDR(s) is only slightly slower than that of full GMRES [59].

GMRESR (GMRES Recursive) Another hybrid method is the GMRESR proposed in [71] and further investigated in [73]. This method consists of an outer and inner loop. In the inner loop one approximates the solution of a linear system by GMRES to find a good search direction. Thereafter in the outer loop the minimal residual approximation using these search directions is calculated by a GCR approach.

```
GMRESR algorithm
Choose $u^0$ and $m$, compute $r^0 = b - Au^0$

    for $i = 1, 2, ...$ do
        $s^i = P_{m,i-1}(A)r^{i-1}$ ,
        $v^i = As^i$ ,
        for $j = 1, ..., i-1$ do
            $\alpha = (v^j, v^i)$ ,
            $s^i := s^i - \alpha s^j$ ,  $v^i := v^i - \alpha v^j$ ,
        end for
        $s^i := s^i/\|v^i\|_2$ ,  $v^i := v^i/\|v^i\|_2$
        $\beta = (v^i, r^{i-1})$ ;
        $u^i := u^{i-1} + \beta s^i$ ;
        $r^i := r^{i-1} - \beta v^i$ ;
    end for
```

The notation $s^i = P_{m,i-1}(A)r^{i-1}$ indicates that one applies one iteration of GMRES(m) to the system $As = r^{i-1}$. The result of this operation is s^i. For $m = 0$ we have just GCR, whereas for $m \to \infty$ one outer iteration is sufficient and GMRESR reduces to GMRES. For the amount of work we refer to [71], where optimal choices of m are also given. In many problems the rate of convergence of GMRESR is comparable to full GMRES, whereas the amount of work and memory is much less. In the following picture we visualize the strong point of GMRESR in comparison with GMRES(m) and truncated GCR. A search direction is indicated by v^i. We see for GMRES(3) that after 3 iterations all information is thrown

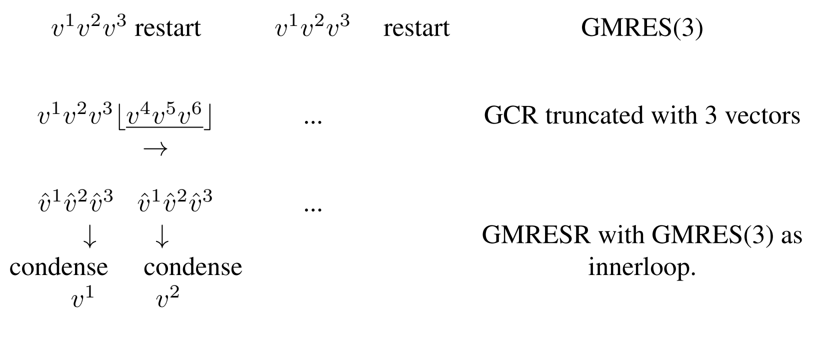

$v^1v^2v^3$ restart
$v^1v^2v^3$ restart
GMRES(3)
$v^1v^2v^3\lfloor\underline{v^4v^5v^6}\rfloor$
...
GCR truncated with 3 vectors
$\hat{v}^1\hat{v}^2\hat{v}^3$
$\hat{v}^1\hat{v}^2\hat{v}^3$
condense
v^1
condense
v^2
GMRESR with GMRES(3) as innerloop.

Figure 7.8: The use of search directions for restarted GMRES, truncated GCR and full GMRESR.

away. For GCR(3) a window of the last 3 vectors moves from left to right. For GMRESR the information after 3 inner iterations is condensed into one search direction so information does not get lost.

Also for GMRESR restart and truncation are possible [73]. In the inner loop other iterative methods can be used. Several of these choices lead to good iterative methods. In theory, we can call the same loop again, which motivates the name GMRES Recursive. A comparable approach is the FGMRES method given in [51]. Herein the outer loop consists of a slightly adapted GMRES algorithm. FGMRES and GMRESR are comparable in work and memory. However, FGMRES can not be truncated. Therefore, we prefer the GMRESR method.

7.3.6 Choice of Iterative Method

For non-symmetric matrices it is difficult to decide which iterative method should be used. All the methods treated here have their own type of problems for which they are winners. Furthermore the choice depends on the computer used and the availability of memory. In general CGS and Bi-CGSTAB are easy to implement and reasonably fast for a large class of problems. If break down or bad convergence occurs, GMRES like methods may be preferred. Finally LSQR always converges but can take a large number of iterations.

In [73] some easy to obtain parameters are specified to facilitate a choice. Firstly, one should have a crude idea of the total number of iterations (mg) using full GMRES. Secondly, one should measure the ratio f

$$f = \frac{\text{the CPU time used for one preconditioned matrix vector product}}{\text{the CPU time used for a vector update}}$$

Note that f also depends on the hardware used. Under certain assumptions, given in [73], Figure 7.9 is obtained. This figure gives only qualitative information. It illustrates the dependence of the choice on f and mg. If mg is large and f is small, Bi-CGSTAB is the best method. For large values of f and small values of mg the GMRES method is optimal and for intermediate values GMRESR is the best method. In [5] a flowchart is given with suggestions for the selection of a suitable iterative method.

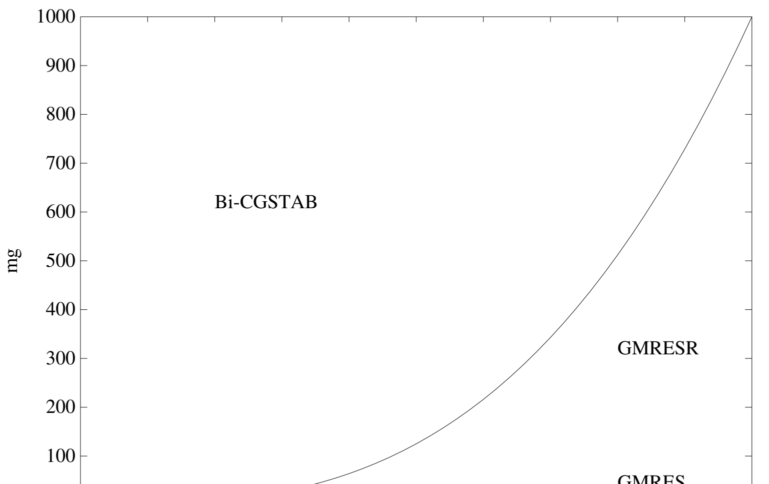


Figure 7.9: Regions of feasibility of Bi-CGSTAB, GMRES, and GMRESR.

7.3.7 Preconditioning for General Matrices

The preconditioning for non-symmetric matrices goes along the same lines as for symmetric matrices. There is a large amount of literature for generalization of the incomplete Cholesky decompositions. In general, it is more difficult to prove that the decomposition does not break down or that the resulting preconditioned system has a spectrum which leads to fast convergence. Since symmetry is no longer an issue the number of possible preconditioners is larger. Furthermore, if we have an incomplete LU decomposition of A, we can apply the iterative methods from 7.3.4 to the following three equivalent systems of equations:

$$U^{-1}L^{-1}Au = U^{-1}L^{-1}b \; , \tag{7.38}$$

$$L^{-1}AU^{-1}y = L^{-1}b \; , \quad u = U^{-1}y \; , \tag{7.39}$$

or

$$AU^{-1}L^{-1}y = b \; , \quad u = U^{-1}L^{-1}y \; . \tag{7.40}$$

The rate of convergence is approximately the same for all variants. When the Eisenstat implementation is applied one should use (7.39). Otherwise, we prefer (7.40) because in that case the stopping criterion is based on $\|r\|_2 = \|b - Au^k\|_2$ whereas for (7.38) it is based on $\|U^{-1}L^{-1}r^k\|_2$, and for (7.39) it is based on $\|L^{-1}r^k\|_2$.

7.3.8 Exercises

Exercise 7.3.1 Show that the solution $\begin{pmatrix} y \\ u \end{pmatrix}$ of the augmented system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ u \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

is such that u satisfies $A^T Au = A^T b$.

Exercise 7.3.2 Take the following matrix

$$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}.$$

a) Suppose that GCR is applied to the system $Au = b$. Show that GCR converges in 1 iteration if $u - u^0 = cr^0$, where $c \neq 0$ is a scalar and $r^0 = b - Au^0$.
b) Apply GCR for the choices $b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $u^0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.
c) Do the same for $u^0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

Exercise 7.3.3 In the GCR algorithm the vector r^i is obtained from vector updates. Show that the relation $r^i = b - Au^i$ is valid.

Exercise 7.3.4 Prove the following properties for the GMRES method:

- $AV_k = V_{k+1}\hat{H}_k$,

- $u^k = u^0 + V_k y^k$, where y^k is obtained from (7.35).

Exercise 7.3.5 Figure 7.9 can give an indication which solution method should be used. Give advice in the following situations:

- Without preconditioning Bi-CGSTAB is the best method. What happens if preconditioning is added?

- We use GMRESR for a stationary problem. Switching to an instationary problem, what are good methods?

- We use GMRES. After optimizing the matrix vector product, which method is optimal?

Exercise 7.3.6 A practical exercise
For the methods mentioned below we use as test matrices:

$$[a, f] = poisson(30, 30, 100, 0, 'central')$$

and

$$[a, f] = poisson(30, 30, 100, 0, 'upwind')$$

a) Adapt the matlab cg algorithm such that it solves the normal equations. Apply the algorithm to both matrices.
b) Implement Bi-CGSTAB from the lecture notes. Take $K = I$ (identity matrix). Apply the algorithm to both matrices.
c) Implement the GCR algorithm from the lecture notes. Apply the algorithm to both matrices.
d) Compare the convergence behavior of the three methods.

Chapter 8

Iterative methods for eigenvalue problems

8.1 Introduction

In many technical problems eigenvalues play an important role. For example eigenvalues give information of physical properties like eigenmodes, or eigenvalues are used to analyze and/or enhance mathematical methods for the solution of a physical problem.
As examples of the first kind we mention the following:

- eigenvalues are important to obtain eigenfrequences of a construction,
- characteristic properties of a fluid flow problem are defined using eigenvalues,
- if a bifurcation occurs eigenvalues and eigenvectors can be used to calculate a solution after the bifurcation point.

Examples of the second kind are:

- estimation of the 2-norm of a matrix (or its inverse),
- to predict and understand the convergence behavior of an iterative method,
- as a check of a discretization method. In general the matrices are so large that it is not easy to check their contents. However a small number of extreme eigenvalues can give sufficient information to decide wether the obtained discretization is correct or not,
- the choice of the time step for stable time integration methods.

In the remainder of this section we give some general information about eigenvalue problems.

The mathematical eigenvalue problem for a linear system of equations can be defined as follows: find $\lambda \in \mathbb{C}$ and $x \in \mathbb{C}^n$ such that $Ax = \lambda x$ and $x \neq 0$.
Some references for the theory on this type of problems are [25]; Chapter 7, 8, 9, [75], [8], [9] and [45]. Again the symmetric eigenvalue problem is much easier than the unsymmetric eigenvalue problem (compare the situation for linear systems). This observation not only holds from a computational point of view but also for the theory of the eigenvalue problem. All methods to solve the eigenvalue problem are of an iterative nature. We distinguish between two different classes of methods. In the first class of methods the matrix A is transformed to a condensed form (computational costs $O(n^3)$) and the iteration process is applied to the condensed matrix (costs $O(n^2)$). As an example of these methods we mention the QR method. This class of methods is used in the public domain linear algebra software library LAPACK and is described in [25]; Chapter 7, 8. Drawback of these methods are that the matrix A should be given explicitly, and in general a large amount of memory is required. It is advised to use these methods for matrices

with relatively small dimensions (say $n < 200$). In the second class the iteration is applied to the original matrix. A clear advantage is that the matrix A does not have to be available. The only requirement is that one is able to calculate matrix vector-products. It is advised to use this type of methods only if a small number of eigenvalues is wanted or the matrix A can not be easily formed.
In Section 8.2 we consider the classical power method. Krylov subspace methods are described in Section 8.3 for symmetric matrices and Section 8.4 for unsymmetric matrices.

8.2 The Power method

The Power method is the classical method to compute the largest few eigenvalues of a matrix. The method is motivated by the property that if we multiply a vector by a matrix, the contribution of the eigenvector corresponding to the largest eigenvalue (in absolute value sense) increased more than the contribution of the other eigenvectors. If the vector is multiplied a large number of times by the matrix, the contribution of this eigenvector will dominate, so the resulting iteration vector will approximate this eigenvector. So we arrive at the following algorithm.

```
The Power method
$q_0 \in \mathbb{C}^n$ is given
for $k = 1, 2, ...$
        $z_k = Aq_{k-1}$
        $q_k = z_k / \|z_k\|_2$
        $\lambda^{(k)} = \bar{q}_{k-1}^T z_k$
endfor
```

It is easy to see that if q_{k-1} is an eigenvector corresponding to λ_j then

$$\lambda^{(k)} = q_{k-1}^T A q_{k-1} = \lambda_j q_{k-1}^T q_{k-1} = \lambda_j \|q_{k-1}\|_2^2 = \lambda_j .$$

In order to derive the convergence behavior of the Power method we assume that the n eigenvalues are ordered such that $|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_n|$ and the eigenvectors by $x_1, ..., x_n$ so $Ax_i = \lambda_i x_i$. Each arbitrary start vector q_0 can be written as:

$$q_0 = a_1 x_1 + a_2 x_2 + ... + a_n x_n$$

and if $a_1 \neq 0$ if follows that

$$A^k q_0 = a_1 \lambda_1^k (x_1 + \sum_{j=2}^{n} \frac{a_j}{a_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k x_j) \ . \tag{8.1}$$

Using this equality we conclude that

$$|\lambda_1 - \lambda^{(k)}| = O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \ , \quad \text{and also} \tag{8.2}$$

the angle between $span\ \{q_k\}$ and $span\ \{x_1\}$ is of order $|\frac{\lambda_2}{\lambda_1}|^k$.
These formula's (8.1) and (8.2) can be used to obtain the following observations. First it is important that $a_1 \neq 0$ so the starting vector should have a non-zero component in the x_1-vector. Due to rounding errors this is in general no problem because if q_0 has no component in the x_1 direction such a component is created during the computation. However, a large component in the start vector leads to a faster convergence. Secondly we see that the convergence depends on $|\frac{\lambda_2}{\lambda_1}|$. So applying the Power method to $A - cI$ the rate of convergence is equal to $|\frac{\lambda_2 - c}{\lambda_1 - c}|$. This property shows that the Power method is not shift invariant.

Furthermore, it can be used to increase the convergence speed. Finally if c is chosen carefully we can compute other eigenvalues. For example: suppose $\lambda_i \in \mathbb{R}\ \ i = 1, ..., n$ then the choice $c \cong \lambda_1$ leads to the fact that the in norm largest eigenvalue of $A - \lambda_1 I$ is equal to λ_n. So also the smallest eigenvalue can be computed by the Power method. Thirdly we see that the Power method is a linearly converging method. This implies that the following stopping criterion can be used:

$$\text{estimate } r \text{ from } \tilde{r} = \frac{|\lambda^{(k+1)} - \lambda^{(k)}|}{|\lambda^{(k)} - \lambda^{(k-1)}|} , \tag{8.3}$$

and stop if $\frac{\tilde{r}}{1-\tilde{r}} \frac{|\lambda^{(k+1)} - \lambda^{(k)}|}{|\lambda^{(k+1)}|} \leq \varepsilon$.

This stopping criterion leads to $|\lambda_1 - \lambda^{(k+1)}| \leq \varepsilon$.

Note that there is a problem if $|\lambda_1| = |\lambda_2|$, which is the case for instance if $\lambda_1 = \bar{\lambda}_2$. A vector q_0 which has a nonzero component in x_1 and x_2 can be written as

$$q_0 = a_1 x_1 + a_2 x_2 + \sum_{j=3}^{n} a_j x_j .$$

The component in the direction of $x_3, ..., x_n$ will vanish in the Power method, but q_k will not tend to a limit. In [75], pp. 579-582 a method is given to obtain the eigenvalues λ_1 and λ_2 from the last three iterates. However when the imaginary part of λ_1 is small the obtained results have a poor accuracy.

The inverse Power method

We have seen that small eigenvalues can be computed by a correct shift of the matrix. However, in general the differences between small eigenvalues are much less then the differences between the large eigenvalues. So convergence to the smallest eigenvalue is very slow. A remedy for this is to apply the Power method to the inverse matrix A^{-1}. It is easily seen that the eigenvalues of A^{-1} are $\frac{1}{\lambda_i}$. So the smallest eigenvalue of A is the largest eigenvalue of A^{-1}. This leads to a much faster rate of convergence. As an example suppose

$$\lambda_1 = 1000 , \quad \lambda_{n-1} = 1.1 \text{ and } \lambda_n = 1 .$$

The rate of convergence of the Power method applied to

$$A - 1000I \text{ is equal to } \frac{|1.1 - 1000|}{|1 - 1000|} = 0.99989$$

whereas application to

$$A^{-1} \text{ leads to } \frac{\frac{1}{1.1}}{\frac{1}{1}} = 0.909 .$$

In order to compute $z_k = A^{-1} q_{k-1}$ one solves the z_k from the linear system

$$A z_k = q_{k-1},$$

by Gaussian elimination, or an iterative solver. In general the inverse Power method costs less work than the Power method applied to the shifted matrix.

Orthogonal iteration

A straightforward generalization of the power method is "orthogonal iteration" which can be used to compute more than one eigenvalue. Let p be an integer less than n, and $Q_0 \in \mathbb{C}^{n \times p}$ an orthogonal matrix. Compute a sequence of matrices $\{Q_k\}$ where $Q_k \in \mathbb{C}^{n \times p}$ as follows:

```
for $k = 1, 2, ...$
    $Z_k = AQ_{k-1}$
    orthonormalize the columns of $Z_k$ such that
    $Q_k R_k = Z_k$ ,   $R_k \in \mathbb{R}^{k\times k}$ is an upper triangular matrix,
        and $\bar{Q}_k^T Q_k = I$.
endfor
```

This can be used to approximate the p largest eigenvalues. For more details we refer to [25]; Section 7.3.2.

8.3 A Krylov subspace method for symmetric matrices

Symmetry simplifies the real eigenvalue problem $Ax = \lambda x$ in two ways. It implies that all eigenvalues are real and that there is an orthogonal basis of eigenvectors. It can be shown that if A is a real $n \times n$ symmetric matrix then there exists a real orthogonal matrix S such that

$$S^T A S = diag\ (\lambda_1, ..., \lambda_n)\ .$$

The iterative method considered in this section is known as the Lanczos method [37], [25]; Chapter 9. The relation between the Power method and the Lanczos method is comparable to the relation between basic iterative methods for linear systems and the CG method. We have seen that in the Power method one calculates $q_0, Aq_0, A^2q_0, ...$ and sees that the vector $A^k q_0$ tends to the eigenvector corresponding to the largest eigenvalue. In the Power method only one vector is used. To explain the properties of the Lanczos method we first define the Rayleigh quotient

$$r(x) = \frac{x^T A x}{x^T x} \quad , \quad x \neq 0\ .$$

It is easily seen that $\min\limits_{x\in\mathbb{R}^n} r(x) = \lambda_n$ the smallest eigenvalue and $\max\limits_{x\in\mathbb{R}^n} r(x) = \lambda_1$ the largest eigenvalue.
In the Lanczos method the approximations after k iterations are $\theta_1^{(k)}$ of λ_1 and $\theta_k^{(k)}$ of λ_n. They satisfy the following (in)equalities

$$\theta_1^{(k)} = \max_{y\in K^k(A;q_0)} r(y) \leq \lambda_1$$

and

$$\theta_k^{(k)} = \min_{y\in K^k(A;q_0)} r(y) \geq \lambda_n\ .$$

These (in)equalities imply that $\theta_1^{(k)}$ is always closer to λ_1 than the approximation of the Power method. Furthermore, Lanczos gives an approximation of the smallest eigenvalue. The rate of convergence of $\theta_1^{(k)}$ to λ_1 and $\theta_k^{(k)}$ to λ_n is comparable. The Lanczos method involves partial tridiagonalizations of the matrix A. Information of A's extremal eigenvalues tends to emerge long before the tridiagonalization is complete. This makes the Lanczos algorithm particularly useful in situations where a few of A's largest or smallest eigenvalues are desired. Unfortunately, roundoff errors make the Lanczos method somewhat difficult to use in practice. The central problem is a loss of orthogonality among the Lanczos vectors that the iteration produces. Some ideas are given to repair orthogonality. We start by the specification of the Lanczos algorithm:
Choose a starting vector q_1 where $\|q_1\|_2 = 1$

```
Lanczos method
  $r_0 = q_1$ ;   $\beta_0 = 1$ ; $q_0 = 0$              initialization

  for       $j = 1, 2, \ldots$ do                        iteration
            $q_j = r_{j-1}/\beta_{j-1}$                  normalization of $q$
            $\alpha_j = q_j^T A q_j$
            $r_j = (A - \alpha_j I)q_j - \beta_{j-1}q_{j-1}$   new direction
                                                         orthogonal to
            $\beta_j = \|r_j\|_2$                        previous $q$.
  end for
```

Thereafter we form the tridiagonal symmetric matrix T_j as follows

$$T_j = \begin{bmatrix} \alpha_1 & \beta_1 & & & 0 \\ \beta_1 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & \beta_{j-1} \\ & & & \beta_{j-1} & \alpha_j \end{bmatrix} .$$

This matrix is called the Ritzmatrix. The eigenvalues of T_j : $\theta_1^{(j)}, ..., \theta_j^{(j)}$ are called Ritzvalues and are approximations of the eigenvalues of A.
With respect to work we note that the Lanczos method costs one matrix vectorproduct per iteration and 5 vector operations. The memory requirements are 5 vectors in memory. Therafter the eigenvalues of T_j have to be calculated. Note that T_j is in general much smaller than A and has only three non zero elements per row. So this eigenvalue problem is always solved by a QR like method ([25]; Section 8.2) for instance by a call to a LAPACK subroutine. The Lanczos vector q_j has several nice properties. In the following theorem it is proved that the vectors $q_1, ..., q_j$ form an orthonormal basis for $K^j(A; q_1)$.

Theorem 8.3.1 *Let $A \in \mathbb{R}^{n \times n}$ be symmetric and assume $q_1 \in \mathbb{R}^n$ satisfies $\|q_1\|_2 = 1$. Then the Lanczos algorithm runs until $j = m$ where m is the number of independent vectors in $K^n(A; q_1)$. Moreover for $j \in [1, m]$ we have*

$$AQ_j = Q_j T_j + r_j e_j^T \ , \tag{8.4}$$

where $Q_j = [q_1, ..., q_j]$ has orthonormal columns that span $K^j(A; q_1)$.

Proof: see [25]; Section 9.1.3.

The Lanczos results can also be used to obtain an approximation of the eigenvectors of A. In order to do this all Lanczos vectors should be kept in memory. Suppose that $\theta_i^{(j)}$ is an eigenvalue of T_j and its corresponding eigenvector is denoted by s_i where $\|s_i\|_2 = 1$. The vector $y_i = Q_j s_i$ is called the Ritzvector and is an approximation of the eigenvector of A belonging to the eigenvalue approximated by $\theta_i^{(j)}$. Heuristically this can be seen as follows: suppose $\|r_j e_j^T\|_2$ in (8.4) is small then

$$AQ_j \cong Q_j T_j$$

so

$$Ay_i = AQ_j s_i \cong Q_j T_j s_i = Q_j \theta_i^{(j)} s_i = \theta_i^{(j)} y_i \ .$$

It can be shown that $\|Ay_i - \theta_i^{(j)} y_i\|_2 = |\beta_j| \, |(s_i)_j|$ where $(s_i)_j$ denotes the final element of the vector s_i ([25]; Section 9.13). This equation can be used to obtain the following error bound:

$$\min_{\mu \in \lambda(A)} |\theta_i^{(j)} - \mu| \leq |\beta_j| \, |(s_i)_j| \quad i = 1, ..., j \ . \tag{8.5}$$

It is much cheaper to check this bound than forming y_i and compute $\|Ay_i - \theta_i^{(j)} y_i\|_2$. So (8.5) can be used as a cheap stopping criterion.

In [25]; Section 9.1.4 some theoretical results are given on the convergence behavior of the extremal Ritzvalues. Suppose λ_1 is the largest eigenvalue of A than it is proved that the largest Ritzvalue θ_1 converges to λ_1. The speed of convergence depends on the so called gap-ratio

$$\rho_1 = \frac{(\lambda_1 - \lambda_2)}{(\lambda_2 - \lambda_n)} \ . \tag{8.6}$$

The value of ρ_1 measures the distance of λ_1 to the rest of the spectrum divided by the distance of λ_2 to λ_n. A large gap ratio leads to a fast convergence of θ_1 to λ_1.
It can be shown that the Lanczos algorithm is shift invariant. If it is applied to

$$\tilde{A} = A - cI \quad \text{the new matrix} \quad \tilde{T}_j \quad \text{is equal to} \quad \tilde{T}_j = T_j - cI \ .$$

So all the results are only shifted and the convergence speed remains the same. This is a clear difference with the Power method. This is in agreement with the fact that the gap ratio is shift invariant:

$$\tilde{\rho}_1 = \frac{\tilde{\lambda}_1 - \tilde{\lambda}_2}{\tilde{\lambda}_2 - \tilde{\lambda}_n} = \frac{\lambda_1 - c - (\lambda_2 - c)}{\lambda_2 - c - (\lambda_n - c)} = \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n} = \rho_1 \ .$$

If the smallest eigenvalues of a matrix A are wanted it is a good idea to apply the Lanczos method to the inverse system

$$A^{-1}x = \mu x \ .$$

This can lead to a much better gap ratio. Note that $\mu = \frac{1}{\lambda}$. Suppose we have an example where $\lambda_1 = 1000,\ \lambda_{n-1} = 1.1$ and $\lambda_n = 1$. The gap ratio for the smallest eigenvalue is equal to

$$\rho_n = \frac{|\lambda_n - \lambda_{n-1}|}{|\lambda_{n-1} - \lambda_1|} = \frac{0.1}{1000} = 10^{-4}$$

so the iteration takes very long to obtain a good approximation. For the inverse problem we want to calculate the largest eigenvalue

$$\mu_1 = 1 \ , \ \mu_2 = \frac{1}{1.1} \ , \ ..., \ \mu_n = \frac{1}{1000}$$

the gap ratio is now equal to

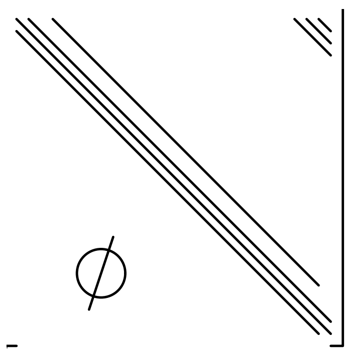
$$\tilde{\rho}_1 = \frac{|\mu_1 - \mu_2|}{|\mu_2 - \mu_n|} = \frac{1 - \frac{1}{1.1}}{\frac{1}{1.1} - \frac{1}{1000}} \simeq 0.1$$

which is much larger than for the original matrix. A drawback is again that one has to solve a linear system of equations in every iteration.

The convergence of Ritzvalues to interior eigenvalues is not so good. Moreover theoretical results for this convergence are not sharp. In general the same behavior as the for CG method applied to linear systems is observed. So if the Ritzvalue θ_1 is close to λ_1, the method behaves as if the eigenvalue λ_1 is absent. So once λ_1 has been approximated, θ_2 converges faster to λ_2.
With respect to rounding errors we note that equation (8.4) holds to working precision. However loss of orthogonality of the computed vectors q_j appears if one of the Ritzvalues converges to an eigenvalue. One remedy is to orthogonalize each newly computed Lanczos vector against its predecessors. This leads to the complete reorthogonalization Lanczos method. However, such an orthogonalization requires many vector operations. This makes the method unpractical if many iterations are necessary. To decrease the costs, a selective orthogonalization procedure is proposed [25]; Section 9.2.4. In this algorithm the new Lanczos vector is not orthogonalized against all its predecessors, but only against the much smaller set of converged Ritzvectors. For details we refer to [46], [56] and [35].

8.4 Krylov subspace methods for unsymmetric matrices

Arnoldi
A generalization of the Lanczos method to unsymmetric matrices is the Arnoldi method [2] and [25]; p.499. In this method the matrix A is transformed to an upper Hessenberg matrix by an orthogonal transformation. An upper Hessenberg matrix has the following nonzero pattern:



The relation between Lanczos and Arnoldi is comparable to the relation between CG and GMRES for linear systems. The Arnoldi method has the same nice properties with respect to convergence as the Lanczos method. A drawback is that for Lanczos only 5 vector operations are necessary during computation, whereas for Arnoldi the number of vector operations is proportional to the number of iterations.
The Arnoldi algorithm is given by: choose a starting vector q_1 where $\|q_1\|_2 = 1$.

```
Arnoldi method

$r = q_1$ ;   $\beta = 1$                initialization
for           $j = 1, 2, \dots$          iteration
              $q_j = r/\beta$ ;          normalization
              $r = Aq_j$ ;
              for $i = 1, ..., j$        modified Gram Schmidt orthogonalization
                  $h_{ij} = q_i^T r$
                  $r = r - h_{ij} q_i$
              end for
              $\beta = \|r\|_2$
              if $j < n$
                  $h_{j+1,j} = \beta$
              end if
end for
```

After the iteration is stopped one can form the Hessenberg matrix H_j as follows

$$H_j = \begin{bmatrix} h_{11} & \dots & \dots & h_{1j} \\ h_{21} & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ O & & h_{jj-1} & h_{jj} \end{bmatrix} .$$

H_j is the Ritzmatrix, θ_i the Ritzvalue. In the same way as for Lanczos we have if $H_j s_i = \theta_i s_i$ then $Q_j s_i$ is an approximation of the corresponding eigenvector.

If the matrix A is symmetric the matrix H_j becomes tridiagonal so we get the same results as using the Lanczos method. Many of the properties of the Lanczos method can be generalized to the Arnoldi method. In general it is a stable method with respect to rounding errors. There is no break down possible, only the case $\beta = 0$ can occur, but then an invariant subspace is obtained and all eigenvalues can be calculated (Assuming that q_0 has nonzero components in all eigenvector directions). A drawback of this method is the fact that due to the modified Gram Schmidt orthogonalization the amount of work increases quadratically. Restarting the Arnoldi method prevents this, however in such a case the good convergence properties are lost. The rate of convergence can be much different from Lanczos, because complex eigenvalues can occur. If all the eigenvalues are real than the convergence behavior of Arnoldi is comparable to Lanczos. In the general case of complex eigenvalues we see again that the Ritzvalues converging to the extreme eigenvalues are converging much faster than the interior ones.

Bi-Lanczos

To get rid of the Gram Schmidt process another generalization is proposed: Bi-Lanczos. It is possible to reduce A to tridiagonal form using a general similarity transformation. However, this leads to an unstable procedure ([75]; pp. 388-405). In the Bi-Lanczos procedure two vector sequences are produced x_j and y_j, which have the property that they are bi-orthogonal:
if $X_j = [x_1, ..., x_j]$, $Y_j = [y_1, ..., y_j]$ we have $X^T Y = I$. The Bi-Lanczos method runs as follows: choose starting vectors x_1, y_1 such that $x_1^T y_1 = 1$.

```
Bi-Lanczos
$j = 0$ , $\beta_0 = 1$ , $x_0 = 0$ , $r_0 = x_1$ , $y_0 = 0$ , $p_0 = y_1$
while $\beta_j \neq 0 \bigwedge r_j^T p_j \neq 0$
    $\gamma_j = r_j^T p_j / \beta_j$
    $x_{j+1} = r_j / \beta_j$ ; $y_{j+1} = p_j / \gamma_j$
    $j := j + 1$
    $\alpha_j = y_j^T A x_j$ ; $r_j = (A - \alpha_j I) x_j - \gamma_{j-1} x_{j-1}$ ,
    $\beta_j = \|r_j\|_2$ ; $p_j = (A - \alpha_j I)^T y_j - \beta_{j-1} y_{j-1}$ ,
end while
```

The tridiagonal Ritzmatrix T_j is formed by:

$$T_j = \begin{bmatrix} \alpha_1 & \gamma_1 & & & \\ \beta_1 & \alpha_2 & \gamma_2 & O & \\ & \ddots & \ddots & \ddots & \\ & O & \ddots & \ddots & \gamma_{j-1} \\ & & & \beta_{j-1} & \alpha_j \end{bmatrix} .$$

The amount of work per iteration is equal to two matrix vector products one with A and the other with A^T. Furthermore, 12 vector operations are needed. To circumvent stability problems the look ahead Lanczos procedures are developed per iteration [21]. However, many open questions remain as there are: how to implement complete or selective orthogonalization, which stop criterion can be used, multiple eigenvalues etc.

8.5 The generalized eigenvalue problem

In practical finite element eigenvalue problems one also wants to solve generalized eigenvalue problems. In such a problem one has to solve the following problem: for $A, B \in \mathbb{R}^{n \times n}$ given, compute $\lambda \in \mathbb{C}$ and $x \in \mathbb{C}^n$ where $x \neq 0$ such that

$$Ax = \lambda Bx . \tag{8.7}$$

In finite element problems A may be the stiffness matrix and B the mass matrix. For theoretical properties and QR like methods we refer to [25]; Section 7.7. If A and B are symmetric and B also positive definite we refer to [25]; Section 8.7.2. If B^{-1} exists (8.7) can be transformed to

$$B^{-1} A x = \lambda x \tag{8.8}$$

so iteration methods can be applied to (8.8).
There are also iterative methods which are suited to be applied to (8.7) directly. For these methods we refer to: [24] and [67].

8.6 Exercises

Exercise 8.6.7 The Power method can be used to approximate the largest eigenvalue λ_1. In this exercise two methods are given to estimate the eigenvalue λ_2 if λ_1 and eigenvector x_1 are known.

a) Take $q_0 = (A - \lambda_1 I)q$, where q is an arbitrary vector. Show that the Power method applied to this starting vector leads to an approximation of λ_2 (Annihilation Technique).
b) Take A is symmetric and show that if the Power method is applied to the matrix

$$B = A - \frac{\lambda_1}{x_1^T x_1} x_1 x_1^T$$

one gets an approximation of λ_2. What is the amount of work per iteration using B (Hotelling Deflation).

Exercise 8.6.8 Suppose that $A \in \mathbb{R}^{n\times n}$ is skew-symmetric.

a) Derive a Lanczos-like algorithm for computing a skew-symmetric tridiagonal matrix T_m such that

$$AQ_m = Q_m T_m,$$

where $m = dim\{K(A; q_1, n)\}$ and $Q_m^T Q_m = I_m$.
b) Show that if m is equal to the dimension of the smallest invariant subspace for A that contains q_1.

Exercise 8.6.9 Suppose $A \in \mathbb{R}^{n\times n}$ is symmetric and that we wish to compute its largest eigenvalue. Let η be an approximate eigenvector and set

$$\alpha = \frac{\eta^T A \eta}{\eta^T \eta}, \quad z = A\eta - \alpha\eta.$$

a) Show that there is an eigenvalue of A in the interval $[\alpha - \delta, \alpha + \delta]$, where $\delta = \|z\|_2 / \|\eta\|_2$.
b) Consider $\bar{\eta} = a\eta + bz$ and show how to determine a and b such that $\bar{a} = \bar{\eta}^T A \bar{\eta} / \bar{\eta}^T \bar{\eta}$ is maximal.
c) Compare this with the first two iterations of Lanczos.

Exercise 8.6.10 A practical exercise
A bending beam with a force P on top (see Figure 8.1) can be described by the following equation:

$$EI\frac{d^2 w}{dx^2} = Pw, \ w(0) = w(L) = 0.$$

The solution of this equation is $w(0) = 0$. For certain values of P, there is also a non-trivial solution. The smallest value of such a P is: $P = \frac{EI\pi^2}{L^2}$.

We can also approximate the smallest value of P by the smallest eigenvalue of

$$A = \frac{EI}{h^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & \cdots \\ 0 & -1 & 2 & -1 & 0 & \cdots \\ & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix},$$

where $A \in \mathbb{R}^{n\times n}$ and $h = \frac{L}{n+1}$. Take $n = 100$, $EI = 10$ and $L = 2$.

a) Compute an approximation of P by doing 50 iterations of the inverse Power method applied to A.
b) Do 10 iterations with the Lanczos method and form T_{10}.
c) Compute the eigenvalues of T_j using the 'eig' command of Matlab.
d) Compare the convergence of the inverse Power method and the Lanczos method.

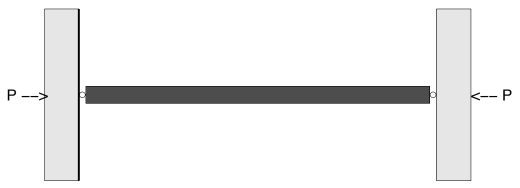


Figure 8.1: Bending beam configuration

Bibliography

[1] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[2] W.E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[3] O. Axelsson. *Iterative solution methods*. Cambridge University Press, Cambridge, 1994.

[4] O. Axelsson and G. Lindskog. On the eigenvalue distribution of a class of preconditioning methods. *Numer. Math.*, 48:479–498, 1986.

[5] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. 1994.

[6] Å. Björck and T. Elving. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. *BIT*, 19:145–163, 1979.

[7] A.M. Bruaset. *A survey of preconditioned iterative methods*. Pitman research notes in mathematics series 328. Longman Scientific and Technical, Harlow, 1995.

[8] F. Chatelin. *Spectral approximation of linear operators*. Academic Press, New York, 1983.

[9] F. Chatelin. *Eigenvalues of matrices*. Wiley, Chichester, 1993.

[10] Biswa Nath Datta. *Numerical Linear Algebra and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2010.

[11] Timothy A. Davies. *Direct Methods for Sparse Linear Systems*. Number 2 in Fundamentals of Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

[12] J.W. Demmel. *Applied Numerical Linear Algebra*. Miscellaneous Bks. Society for Industrial and Applied Mathematics, 1997.

[13] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Solution Methods for Sparse Linear Systems*. Clarendon Press, Oxford, UK, 1986.

[14] M. Eiermann, W. Niethammer, and R.S. Varga. A study of semiiterative methods for nonsymmetric systems of linear equations. *Numer. Math.*, 47:505–533, 1985.

[15] S.C. Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM J. Sci. Stat. Comput.*, 2:1–4, 1981.

[16] S.C. Eisenstat, H.C. Elman, and M.H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Num. Anal.*, 20:345–357, 1983.

[17] M. Embree. The Tortoise and the Hare restart GMRES. *SAIM Review*, 45:259–266, 2003.

[18] V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Num. Anal.*, 21:356–362, 1984.

[19] R. Fletcher. Factorizing symmetric indefinite matrices. *Lin. Alg. and its Appl.*, 14:257–277, 1976.

[20] R.W. Freund, G.H. Golub, and N.M. Nachtigal. Iterative solution of linear systems. In A. Iserles, editor, *Acta Numerica*, pages 57–100. Cambridge University Press, Cambridge, 1992.

[21] R.W. Freund, M.H. Gutknecht, and N.M. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM J. Sci. Comp.*, 14:137–156, 1993.

[22] T. Ginsburg. The conjugate gradient method. In J.H. Wilkinson and C. Reinsch, editors, *Handbook for Automatic Computation, 2, Linear Algebra*, pages 57–69, Berlin, 1971. Springer.

[23] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, Maryland, fourth edition, 2013.

[24] G.H. Golub, R. Underwood, and J.H. Wilkinson. The Lanczos algorithm for the symmetric $Ax = \lambda Bx$ problem. Report STAN-CS-72-270, Department of Computer Science, Stanford University, Stanford, California, 1972.

[25] G.H. Golub and C.F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1996. Third edition.

[26] G.H. Golub and R.S. Varga. Chebychev semi-iterative methods, successive over-relaxation iterative methods and second order Richardson iterative methods. Part I and II. *Numer. Math.*, 3:147–156, 157–168, 1961.

[27] Nicholas I. M. Gould, Jennifer A. Scott, and Yifan Hu. A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations. *ACM Trans. Math. Softw.*, 33(2), June 2007.

[28] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1987.

[29] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. Frontiers in applied mathmatics 17. SIAM, Philadelphia, 1997.

[30] A. Greenbaum, V. Ptak, and Z. Strakos. Any nonincreasing convergence curve is possible for GMRES. *SIAM J. Matrix Anal. Appl.*, 17:465–469, 1996.

[31] I. Gustafsson. A class of first order factorization methods. *BIT*, 18:142–156, 1978.

[32] L.A. Hageman and D.M. Young. *Applied iterative methods*. Academic Press, New York, 1981.

[33] M.R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *J. Res. Nat. Bur. Stand.*, 49:409–436, 1952.

[34] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambrige, 1985.

[35] T.J.R. Hughes. *The Finite Element Method, Linear Static and Dynamic Finite Element Analysis*. Prentice Hall Inc., Englewood Cliffs, New Yersey, 1987.

[36] E.F. Kaasschieter. Preconditioned conjugate gradients for solving singular systems. *J. Comp. Appl. Math.*, 24:265–275, 1988.

[37] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Stand.*, 45:255–282, 1950.

[38] David C. Lay. *Linear Algebra and Its Applications*. Pearson Education, third edition, 2006.

[39] Xiaoye S. Li. An overview of SuperLU: Algorithms, implementation and user interface. *ACM Transactions on Mathematical Software*, 31(3):302–325, September 2005.

[40] T.A. Manteuffel. The Tchebychew iteration for nonsymmetric linear systems. *Num. Math.*, 28:307–327, 1977.

[41] J.A. Meijerink and H.A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31:148–162, 1977.

[42] N.M. Nachtigal, S.C. Reddy, and L.N. Trefethen. How fast are non symmetric matrix iterations. *SIAM J. Matrix Anal. Appl.*, 13:778–795, 1992.

[43] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[44] C.C. Paige and M.A. Saunders. LSQR : An algorithm for sparse linear equations and sparse least squares. *A.C.M. Trans. Math. Softw.*, 8:43–71, 1982.

[45] B.N. Parlett. *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics 20. SIAM, Philadelphia, 1998.

[46] B.N. Parlett and D.S. Scott. The Lanczos algorithm with selective orthogonalization. *Math. Comp.*, 33:217–238, 1979.

[47] B.N. Parlett, D.R. Taylor, and Z.A. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comp.*, 44:105–124, 1985.

[48] Q. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford Science Publications, 1999.

[49] J.K. Reid. The use of conjugate for systems of linear equations posessing property A. *SIAM J. Num. Anal.*, 9:325–332, 1972.

[50] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, Oxford Road, Manchester M13 9PL, UK, 1992.

[51] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Stat. Comp.*, 14:461–469, 1993.

[52] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, MA, 1996.

[53] Y. Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. SIAM, Philadelphia, 2003.

[54] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[55] Olaf Schenk, Matthias Bollhöfer, and Rudolf A. Römer. On large-scale diagonalization techniques for the anderson model of localization. *SIAM Rev.*, 50(1):91–112, February 2008.

[56] H.D. Simon. The Lanczos algorithm with partial reorthogonalization. *Math. Comp.*, 42:115–142, 1984.

[57] B. F. Smith, P. O. Bjørstad, and W. D. Gropp. *Domain Decompostion: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[58] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 10:36–52, 1989.

[59] P. Sonneveld. On the convergence behaviour of IDR(s). Report 10-08, Delft University of Technology, Department of Applied Mathematical Analysis, Delft, 2010.

[60] P. Sonneveld and M.B. van Gijzen. IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM J. Scientific Computing*, 31:1035–1062, 2008.

[61] J. Stoer and R. Burlish. *Introduction to Numerical Analysis*. Springer-Verlag, 2002.

[62] A. Toselli and O. Widlund, editors. *Domain Decompposition Methods - Algorithms and Theory*. Number 34 in Springer Series in Computational Mathematics. Springer, 2004.

[63] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. SIAM, 1997.

[64] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, San Diego, 2001.

[65] A. van der Sluis. Conditioning, equilibration, and pivoting in linear algebraic systems. *Numer. Math.*, 15:74–86, 1970.

[66] A. van der Sluis and H.A. van der Vorst. The rate of convergence of conjugate gradients. *Numer. Math.*, 48:543–560, 1986.

[67] H.A. van der Vorst. A vectorizable variant of some ICCG methods. *SIAM J. Sci. Stat. Comp.*, 3:350–356, 1982.

[68] H.A. van der Vorst. High performance preconditioning. *SIAM J. Sci. Stat. Comput.*, 10:1174–1185, 1989.

[69] H.A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 13:631–644, 1992.

[70] H.A. van der Vorst and C. Vuik. The superlinear convergence behaviour of GMRES. *J. Comput. Appl. Math.*, 48:327–341, 1993.

[71] H.A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Num. Lin. Alg. Appl.*, 1:369–386, 1994.

[72] Richard S. Varga. *Matrix Iterative Analysis*. Springer, Berlin, New York, second edition, 2000.

[73] C. Vuik. Solution of the discretized incompressible Navier-Stokes equations with the GMRES method. *Int. J. for Num. Meth. Fluids*, 16:507–523, 1993.

[74] P. Wesseling and P. Sonneveld. Numerical experiments with a multiple grid and a preconditioned Lanczos type method. In R. Rautmann, editor, *Approximation methods for Navier-Stokes problems*, pages 543–562, Berlin, 1980. Springer-Verlag. Lecture Notes in Math. 771.

[75] J.H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, England, 1965.

[76] David M. Jr. Young. *Iterative Solution of Large Linear Systems*. Dover, Mineola, NY, USA, 2003.

Index