CEMRACS 2016

Numerical challenges in parallel scientific computing
July 18th - August 26th

Krylov subspace solvers and preconditioners

Prof.dr.ir. C. Vuik

2016

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

Contents

Preface

In these lecture notes an introduction to Krylov subspace solvers and preconditioners is presented. After a discretization of partial differential equations large, sparse systems of linear equations has to be solved. Fast solution of these systems is very urgent nowadays. The size of the problems can be 10^{13} unknowns and 10^{13} equations. Iterative solution methods are the methods of choice for these large linear systems. We start with a short introduction of Basic Iterative Methods. Thereafter preconditioned Krylov subspace methods, which are state of the art, are describeed. A distinction is made between various classes of matrices.

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 [19] and Horn and Johnson [26] 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 [50] is good starting point to study the theory of basic iterative methods. Krylov subspace methods and multigrid are discussed in Saad [38] and Trottenberg, Oosterlee and Schüller [42]. Other books on Krylov subspace methods are [1, 34, 6, 21].

Delft, June 2016, Kees Vuik

1 Basic iterative solution methods for linear systems

1.1 Introduction and model problem

Problems coming from discretized partial differential equations lead in general to large sparse systems of equations. Direct solution methods can be impractical if A is large and sparse, because the factors L and U can be dense. This is especially the case for 3D problems. So a considerable amount of memory is required and even the solution of the triangular system costs many floating point operations.

In contrast to the direct methods are the iterative methods. These methods generate a sequence of approximate solutions $\left\{x^{(k)}\right\}$ and essentially involve the matrix A only in the context of matrix-vector multiplication. The evaluation of an iterative method invariably focuses on how quickly the iterates $x^{(k)}$ converge. The study of round off errors is in general not very well developed. A reason for this is that the iterates are only approximations of the exact solution, so round off errors in general only influence the speed of convergence but not the quality of the final approximation.

The use of iterative solution methods is very attractive in time dependent and nonlinear problems. For these problems the solution of the linear system is part of an outer loop: time stepping for a time dependent problem and Newton Raphson (or other nonlinear methods) for a nonlinear problem. So good start vectors are in general available: the solution of the preceding outer iteration, whereas the required accuracy is in general low for these problems. Both properties lead to the fact that only a small number of iterations is sufficient to obtain the approximate solution of the linear system. Before starting the description and analysis of iterative methods we describe a typical model problem obtained from a discretized partial differential equation. The properties and definitions of the given methods are illustrated by this problem.

Model problem
Consider the discrete approximation of the Poisson equation

$$\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} = G(x,y) \quad , \quad 0 < x < 1 \quad , \quad 0 < y < 1,$$

and boundary conditions

$$w(x,y) = g(x,y) \quad , \quad x \in \{0,1\} \quad , \quad \text{or} \quad y \in \{0,1\}\,.$$

In the sequel v_{ij} is an approximation of $w(x_i, y_j)$ where $x_i = ih$ and $y_j = jh$, $0 \leq i \leq m+1$, $0 \leq j \leq m+1$. The finite difference approximation may be written as:

$$4v_{i,j} - v_{i+1,j} - v_{i-1,j} - v_{i,j+1} - v_{i,j-1} = -h^2\, G(x_i, y_j).$$

The ordering of the nodal points is given in Figure 1 for $m = 3$. The k^{th} component u_k of the vector u is the unknown corresponding to the grid point labeled k. When all the boundary

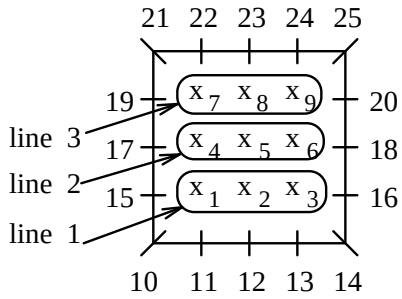

Figure 1: The natural (lexicographic) ordering of the nodal points

terms are moved to the right-hand side, the system of difference equations can be written as:

$$\begin{bmatrix} 4 & -1 & 0 & -1 & & & & & \\ -1 & 4 & -1 & 0 & -1 & & \oslash & & \\ 0 & -1 & 4 & 0 & 0 & -1 & & & \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & & \\ & -1 & 0 & -1 & 4 & -1 & 0 & -1 & \\ & & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ & & & -1 & 0 & 0 & 4 & -1 & 0 \\ & \oslash & & & -1 & 0 & -1 & 4 & -1 \\ & & & & & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix} = \begin{bmatrix} g_{11} + g_{15} - h^2G_1 \\ g_{12} - h^2G_2 \\ g_{13} + g_{16} - h^2G_3 \\ g_{17} - h^2G_4 \\ - h^2G_5 \\ g_{18} - h^2G_6 \\ g_{19} + g_{22} - h^2G_7 \\ g_{23} - h^2G_8 \\ g_{20} + g_{24} - h^2G_9 \end{bmatrix}$$

If the unknowns on lines parallel to the x-axis are grouped together the given system can be written as:

$$\begin{bmatrix} A_{1,1} & A_{1,2} & 0 \\ A_{2,1} & A_{2,2} & A_{2,3} \\ 0 & A_{3,2} & A_{3,3} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix},$$

where the unknowns on line k are denoted by U_k. The lines and grid points are given in Figure 2.1. The matrices $A_{i,j}$ are given by:

$$A_{k,k} = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix} \quad \text{and} \quad A_{k+1,k} = A_{k,k+1} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

The submatrix $A_{k,l}$ gives the coupling of the unknowns from line k to those on line l.

1.2 Basic iterative methods

The basic idea behind iterative methods for the solution of a linear system $Ax = b$ is: starting from a given $x^{(k)}$, obtain a better approximation $x^{(k+1)}$ of x in a cheap way. Note that $b - Ax^{(k)}$ is small if $x^{(k)}$ is close to x. This motivates the iteration process

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}) \tag{1}$$

One immediately verifies that if this process converges, x is a possible solution.
So if $\|b - Ax^{(k)}\|_2$ is large we get a large change of $x^{(k)}$ to $x^{(k+1)}$. The choice of M is crucial in order to obtain a fast converging iterative method. Rewriting of (1) leads to:

$$Mx^{(k+1)} = Nx^{(k)} + b \tag{2}$$

where the matrix N is given by $N = M - A$. The formula $A = M - N$ is also known as a splitting of A. It can easily be seen that if $x^{(k+1)} \to x$ the vector x should satisfy

$$Mx = Nx + b \quad \Leftrightarrow \quad Ax = (M - N)x = b.$$

As a first example we describe the point Gauss Jacobi method. First we define $D_A = diag\,(A)$ and L_A and U_A which are the strictly lower respectively the strictly upper part of A. So $A = D_A + L_A + U_A$.

Gauss Jacobi (point).
The choice $M = D_A$ and $N = -(L_A + U_A)$ leads to the point Gauss Jacobi iteration. This algorithm can be described by the following formula:

$$\begin{aligned} &\text{for } \; i = \; 1, \ldots, n \text{ do} \\ &\qquad x_i^{(k+1)} = \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right) / a_{ii} \qquad (3) \\ &\text{end for} \end{aligned}$$

One immediately sees that only memory is required to store the matrix A, the right-hand side vector b and the approximation vector $x^{(k)}$ which can be overwritten in the next step. For our model problem it is sufficient to store 7 vectors in memory. Furthermore, one iteration costs approximately as much work as a matrix vector product.

Whether or not the iterates obtained by formula (2) converge to $x = A^{-1}b$ depends upon the eigenvalues of $M^{-1}N$. The set of eigenvalues of A is denoted as the spectrum of A. The spectral radius of a matrix A is defined by

$$\rho(A) = \; \max \; \{|\lambda|, \text{ where } \lambda \in \text{ spectrum of } A\}\,.$$

The size of $\rho(M^{-1}N)$ is critical to the convergence behavior of (2).

Theorem 1.1 *Suppose $b \in \mathbb{R}^n$ and $A = M - N \in \mathbb{R}^{n \times n}$ is nonsingular. If M is nonsingular and the spectral radius of $M^{-1}N$ is less than 1, then the iterates $x^{(k)}$ defined by (2) converge to $x = A^{-1}b$ for any starting vector $x^{(0)}$.*

Proof: Let $e^{(k)} = x^{(k)} - x$ denote the error in the k^{th} iterate. Since $Mx = Nx + b$ it follows that

$$M(x^{(k+1)} - x) = N(x^{(k)} - x)$$

and thus the error in $x^{(k+1)}$ given by $e^{(k+1)}$ satisfies:

$$e^{(k+1)} = M^{-1}Ne^{(k)} = (M^{-1}N)^k e^{(0)} \qquad (4)$$

From [19], p. 336, Lemma 7.3.2 it follows that $(M^{-1}N)^k \to 0$ for $k \to \infty$ if $\rho(M^{-1}N) < 1$, so $e^{(k+1)} \to 0$ for $k \to \infty$. □

As an example we note that point Gauss Jacobi is convergent if the matrix A is strictly diagonal dominant. To show this we note that

$$\rho(M^{-1}N) \leq \|M^{-1}N\|_\infty = \|D_A^{-1}(L_A + U_A)\|_\infty = \max_{1 \leq i \leq n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|}\,.$$

Since a strictly diagonal dominant matrix has the property that $\sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| < |a_{ii}|$ it follows that $\rho(M^{-1}N) < 1$. In many problems an increase of the diagonal dominance leads to a decrease of the number of iterations.

Summarizing the results given above we see that a study of basic iterative methods proceeds along the following lines:

- a splitting $A = M - N$ is given where systems of the form $Mz = d$ (d given and z unknown) are cheaply solvable in order to obtain a practical iteration method by (2),
- classes of matrices are identified for which the iteration matrix $M^{-1}N$ satisfies $\rho(M^{-1}N) < 1$.
- further results about $\rho(M^{-1}N)$ are established to gain intuition about how the error $e^{(k)}$ tends to zero.

For Gauss Jacobi we note that M is a diagonal matrix so that systems of the form $Mz = d$ are easily solvable. We have specified a class of matrices such that convergence occurs. The final point given above is in general used to obtain new methods with a faster rate of convergence or a wider class of matrices such that $\rho(M^{-1}N) < 1$.

Below we first give a block variant of the Gauss Jacobi method. Thereafter other methods are specified. In the remainder of this section we suppose that $Ax = b$ is partitioned in the form

$$\begin{bmatrix} A_{1,1} & \dots & A_{1,q} \\ \vdots & & \vdots \\ A_{q,1} & \dots & A_{q,q} \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_q \end{bmatrix} = \begin{bmatrix} B_1 \\ \vdots \\ B_q \end{bmatrix},$$

where $A_{i,j}$ is an $n_i \times n_j$ submatrix and $n_1 + n_2 \ldots + n_q = n$. Here the X_i and B_i represent subvectors of order n_i. Furthermore, we define

$$D_A = \begin{bmatrix} A_{1,1} & & \oslash \\ & \ddots & \\ \oslash & & A_{q,q} \end{bmatrix}, \; L_A = \begin{bmatrix} \bigcirc & & \\ A_{2,1} & \oslash & \\ \vdots & \ddots & \\ A_{q,1} & A_{q,q-1} & \bigcirc \end{bmatrix} \text{ and } U_A = \begin{bmatrix} \bigcirc & A_{1,2} & A_{1,q} \\ & \ddots & \\ \oslash & & A_{q-1,q} \\ & & \bigcirc \end{bmatrix}$$

Gauss Jacobi (block)
For the given matrices D_A, L_A and U_A the Gauss Jacobi method is given by $M = D_A$ and $N = -(L_A + U_A)$. The iterates can be obtained by the following algorithm:

for $i = 1, \ldots, q$ do

$$X_i^{(k+1)} = A_{i,i}^{-1} \left(B_i - \sum_{\substack{j=1 \\ j\neq i}}^{q} A_{ij} X_j^{(k)} \right).$$

end for

For the special case $n_i = 1$, $i = 1, \ldots, n$ we get the point Gauss Jacobi back. In our example a natural block ordering is obtained if we take $n_i = 3$. In that case the diagonal block matrices $A_{i,i}$ are tridiagonal matrices for which cheap methods exist to solve systems of the form

$A_{i,i}z = d$.

Note that in the Gauss Jacobi iteration one does not use the most recently available information when computing $X_i^{(k+1)}$. For example $X_1^{(k)}$ is used in the calculation of $X_2^{(k+1)}$ even though component $X_1^{(k+1)}$ is already known. If we revise the Gauss Jacobi iteration so that we always use the most current estimate of the exact X_i then we obtain:

Gauss Seidel (block)
The Gauss Seidel method is given by $M = D_A + L_A$ and $N = -U_A$. The iterates can be obtained by

$$\begin{aligned} &\text{for } i = 1, \ldots, q \text{ do} \\ &\qquad X_i^{(k+1)} = A_{i,i}^{-1} \left(B_i - \sum_{j=1}^{i-1} A_{i,j} X_j^{(k+1)} - \sum_{j=i+1}^{q} A_{i,j} X_j^{(k)} \right). \\ &\text{end for} \end{aligned}$$

Note that again the solution of systems of the form $Mz = d$ are easily obtained since $M = D_A + L_A$ and L_A is a strictly lower triangular block matrix. As an example for the convergence results that can be proved for the point Gauss Seidel method ($A_{i,i} = a_{i,i}$), we give the following theorem:

Theorem 1.2 *If $A \in \mathbb{R}^{n \times n}$ is symmetric and positive definite, then the point Gauss Seidel iteration converges for any $x^{(0)}$.*

Proof: see [19], p. 512, Theorem 10.1.2.

This result is frequently applicable because many of the matrices that arise from discretized elliptic partial differential equations are symmetric positive definite. In general Gauss Seidel converges faster than Gauss Jacobi. Furthermore, block versions converge faster than point versions. For these results and further detailed convergence proofs we refer to [51], [54], [24], and [3].

The Gauss Seidel iteration is in general a slowly converging process. Inspections of the iterates show that in general the approximations are monotonous increasing or decreasing. Hence we may expect an improvement of the rate of convergence if we use an extrapolation in the direction of the expected change. This leads to the so called successive over-relaxation (SOR) method:

SOR (block)
The successive over-relaxation method is obtained by choosing a constant $\omega \in \mathbb{R}$ and compute the iterates by

$$M_\omega x^{(k+1)} = N_\omega x^{(k)} + \omega b,$$

where $M_\omega = D_A + \omega L_A$ and $N_\omega = (1-\omega)D_A - \omega U_A$. Note that $\omega A = M_\omega - N_\omega$. The following algorithm can be used:

$$\begin{aligned} &\text{for } i = 1, \ldots, q \text{ do} \\ &\qquad X_i^{(k+1)} = \omega A_{i,i}^{-1} \left(B_i - \sum_{j=1}^{i-1} A_{i,j} X_j^{(k+1)} - \sum_{j=i+1}^{q} A_{i,j} X_j^{(k)} \right) + (1-\omega) X_i^{(k)}. \\ &\text{end for} \end{aligned}$$

It can be shown that for $0 < \omega < 2$ the SOR method converges if A is symmetric and positive definite. For $\omega < 1$ we have underrelaxation and for $\omega > 1$ we have overrelaxation. In most examples overrelaxation is used. For a few structured (but important) problems such as our model problem, the value of the relaxation parameter ω that minimizes $\rho(M_\omega^{-1}N_\omega)$ is known. Moreover, a significant reduction of $\rho(M_{\omega_{opt}}^{-1}N_{\omega_{opt}})$ with respect to $\rho(M_1^{-1}N_1)$ can be obtained. Note that for $\omega = 1$ we get the Gauss Seidel method. As an example it appears that for the model problem the number of Gauss Seidel iterations is proportional to $\frac{1}{h^2}$ whereas the number of SOR iterations with optimal ω is proportional to $\frac{1}{h}$. So for small values of h a considerable gain in work results. However, in more complicated problems it may be necessary to perform a fairly sophisticated eigenvalue analysis in order to determine an appropriate ω. A complete survey of "SOR theory" appeared in [54]. Some practical schemes for estimating the optimum ω are discussed in [5], [7], and [53].

Chebyshev

The SOR method is presented as an acceleration of the Gauss Seidel method. Another method to accelerate the convergence of an iterative method is the Chebyshev method. Suppose $x^{(1)}, \ldots, x^{(k)}$ have been obtained via (2), and we wish to determine coefficients $\gamma_j(k)$, $j = 0, \ldots, k$ such that

$$y^{(k)} = \sum_{j=0}^{k} \gamma_j(k) x^{(j)} \tag{5}$$

is an improvement of $x^{(k)}$. If $x^{(0)} = \ldots = x^{(k)} = x$, then it is reasonable to insist that $y^{(k)} = x$. Hence we require

$$\sum_{j=0}^{k} \gamma_j(k) = 1 \tag{6}$$

and consider how to choose the $\gamma_j(k)$ so that the error $y^{(k)} - x$ is minimized. It follows from the proof of Theorem 1.1 that $e^{(k+1)} = (M^{-1}N)^k e^{(0)}$ where $e^{(k)} = x^{(k)} - x$. This implies that

$$y^{(k)} - x = \sum_{j=0}^{k} \gamma_j(k)(x^{(j)} - x) = \sum_{j=0}^{k} \gamma_j(k)(M^{-1}N)^j e^{(0)}. \tag{7}$$

Using the 2-norm we look for $\gamma_j(k)$ such that $\|y^{(k)} - x\|_2$ is minimal. To simplify this minimization problem we use the following inequality:

$$\|y^{(k)} - x\|_2 \leq \|p_k(M^{-1}N)\|_2 \|x^{(0)} - x\|_2 \tag{8}$$

where $p_k(z) = \sum_{j=0}^{k} \gamma_j(k) z^j$ and $p_k(1) = 1$. We now try to minimize $\|p_k(M^{-1}N)\|_2$ for all polynomials satisfying $p_k(1) = 1$. Another simplification is the assumption that $M^{-1}N$ 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

$$\|p_k(M^{-1}N)\|_2 = \max_{\lambda_i} |p_k(\lambda_i)| \leq \max_{\alpha < \lambda < \beta} |p_k(\lambda)|.$$

So to make the norm of $p_k(M^{-1}N)$ 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

$$\begin{aligned} c_0(z) &= 1, \\ c_1(z) &= z, \\ c_j(z) &= 2zc_{j-1}(z) - c_{j-2}(z). \end{aligned}$$

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

$$-1 \leq -1 + 2\frac{z-\alpha}{\beta-\alpha} \leq 1 \qquad \text{for } \; z \in [\alpha, \beta] \;\; \text{so the}$$

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 (8) leads to

$$\|y^{(k)} - x\|_2 \leq \frac{\|x - x^{(0)}\|_2}{|c_k\left(1 + 2\frac{1-\beta}{\beta-\alpha}\right)|}. \tag{9}$$

Calculation of the approximation $y^{(k)}$ by formula (5) costs much time and memory, since all the vectors $x^{(0)}, \ldots, x^{(k)}$ 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:

$$y^{(0)} = x^{(0)}$$

solve $z^{(0)}$ from $Mz^{(0)} = b - Ay^{(0)}$ then $y^{(1)}$ is given by

$$y^{(1)} = y^{(0)} + \tfrac{2}{2-\alpha-\beta} z^{(0)}$$

solve $z^{(k)}$ from $Mz^{(k)} = b - Ay^{(k)}$ 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 $My^{(k+1)} = Ny^{(k)} + b$. 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 [51], [20] and [24].

In deriving the Chebyshev acceleration we assumed that the iteration matrix $M^{-1}N$ was symmetric. Thus our simple analysis does not apply to the SOR iteration matrix $M_\omega^{-1}N_\omega$ 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 [19], Section 10.1.5.
Finally we present some theoretical results for the Chebyshev method [1]. Suppose that the matrix $M^{-1}A$ 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 1.3 *If the Chebyshev method is applied and $M^{-1}A$ is symmetric positive definite then*

$$\|y^{(k)} - x\|_2 \leq 2\left(\frac{\sqrt{K_2(M^{-1}A)}-1}{\sqrt{K_2(M^{-1}A)}+1}\right)^k \|x^{(0)} - x\|_2.$$

Proof Since $M^{-1}A = M^{-1}(M-N) = I - M^{-1}N$ we see that the eigenvalues satisfy the following relation:

$$\mu_i = 1 - \lambda_i \qquad \text{or} \qquad \lambda_i = 1 - \mu_i.$$

This combined with (9) leads to the inequality:

$$\|y^{(k)} - x\|_2 \leq \frac{\|x - x^{(0)}\|_2}{|c_k\left(1 + 2\frac{(1-(1-\mu_1))}{(1-\mu_1)-(1-\mu_n)}\right)|}. \tag{10}$$

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

$$c_k(z) = \frac{1}{2}\left\{\left(z+\sqrt{z^2-1}\right)^k + \left(z-\sqrt{z^2-1}\right)^k\right\} \quad \text{[1], p. 180.}$$

This expression can be used to show that

$$\begin{aligned} c_k\left(\tfrac{1+\frac{\mu_1}{\mu_n}}{1-\frac{\mu_1}{\mu_n}}\right) &> \tfrac{1}{2}\left(\tfrac{1+\frac{\mu_1}{\mu_n}}{1-\frac{\mu_1}{\mu_n}} + \sqrt{\left(\tfrac{1+\frac{\mu_1}{\mu_n}}{1-\frac{\mu_1}{\mu_n}}\right)^2 - 1}\right)^k = \\ &= \tfrac{1}{2}\left(\tfrac{1+\frac{\mu_1}{\mu_n}+2\sqrt{\frac{\mu_1}{\mu_n}}}{1-\frac{\mu_1}{\mu_n}}\right)^k = \tfrac{1}{2}\left(\tfrac{1+\sqrt{\frac{\mu_1}{\mu_n}}}{1-\sqrt{\frac{\mu_1}{\mu_n}}}\right)^k. \end{aligned} \tag{11}$$

The condition number $K_2(M^{-1}A)$ is equal to $\frac{\mu_n}{\mu_1}$. Together with (10) and (11) this leads to

$$\|y^{(k)} - x\|_2 \leq 2\left(\frac{\sqrt{K_2(M^{-1}A)}-1}{\sqrt{K_2(M^{-1}A)}+1}\right)^k \|x^{(0)} - x\|_2.$$

□

Chebyshev type methods which are applicable to a wider range of matrices are given in the literature. In [28] 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 [8].

[1] These results are used in the following chapters to analyze the converge behavior of other iterative methods

1.3 Starting vectors and termination criteria

Starting vectors

All the given iterative solution methods used to solve $Ax = b$ start with a given vector $x^{(0)}$. In this subsection we shall give some ideas how to choose a good starting vector $x^{(0)}$. These choices depend on the problem to be solved. If no further information is available one always starts with $x^{(0)} = 0$. The solution of a nonlinear problem is in general approximated by the solution of a number of linear systems. In such a problem the final solution of the iterative method at a given outer iteration can be used as a starting solution for the iterative method used to solve the next linear system.

Suppose we have a time dependent problem. The solution of the continuous problem is denoted by $u^{(n)}$. In every time step this solution is approximated by a discrete solution $u_h^{(n)}$ satisfying the following linear system

$$A^{(n)} u_h^{(n)} = b^{(n)}.$$

These systems are approximately solved by an iterative method where the iterates are denoted by $x^{(n,k)}$. An obvious choice for the starting vector is $x^{(n+1,0)} = x^{(n,\boldsymbol{k}_n)}$ where $\boldsymbol{k}_n$ denotes the number of iterations in the n^{th} time step. A better initial estimate can be obtained by the following extrapolation:

$$u^{(n+1)} \cong u^{(n)} + \triangle t \frac{du^{(n)}}{dt},$$

where $\frac{du^{(n)}}{dt}$ is approximated by $\frac{x^{(n,\boldsymbol{k}_n)} - x^{(n-1,\boldsymbol{k}_{n-1})}}{\triangle t}$. This leads to the following starting vector

$$x^{(n+1,0)} = 2x^{(n,\boldsymbol{k}_n)} - x^{(n-1,\boldsymbol{k}_{n-1})} \ .$$

Finally starting vectors can sometimes be obtained by solution of related problems, e.g., analytic solution of a simplified problem, a solution computed by a coarser grid, a numerical solution obtained by a small change in one of the parameters etc.

Termination criteria

In Subsection 1.2 we have specified iterative methods to solve $Ax = b$. However, no criteria to stop the iterative process have been given. In general, the iterative method should be stopped if the approximate solution is accurate enough. A good termination criterion is very important for an iterative method, because if the criterion is too weak the approximate solution is useless, whereas if the criterion is too severe the iterative solution method never stops or costs too much work.

We start by giving a termination criterion for a linear convergent process. An iterative method is linear convergent if the iterates satisfy the following equation:

$$\|x^{(i)} - x^{(i-1)}\|_2 \approx r\|x^{(i-1)} - x^{(i-2)}\|_2, \quad r < 1 \tag{12}$$

and $x^{(i)} \to A^{-1}b$ for $i \to \infty$. Relation (12) is easily checked during the computation. In general initially (12) is not satisfied but after some iterations the quantity $\frac{\|x^{(i)} - x^{(i-1)}\|_2}{\|x^{(i-1)} - x^{(i-2)}\|_2}$ converges to r. The Gauss Jacobi, Gauss Seidel and SOR method all are linear convergent.

Theorem 1.4 *For a linear convergent process we have the following inequality*

$$\|x - x^{(i)}\|_2 \leq \frac{r}{1-r}\|x^{(i)} - x^{(i-1)}\|_2.$$

Proof
Using (12) we obtain the following inequality for $k \geq i+1$.

$$\|x^{(k)} - x^{(i)}\|_2 \leq \sum_{j=i}^{k-1} \|x^{(j+1)} - x^{(j)}\|_2 \leq \sum_{j=1}^{k-i} r^j \|x^{(i)} - x^{(i-1)}\|_2$$

$$= r\frac{1-r^{k-i-1}}{1-r}\|x^{(i)} - x^{(i-1)}\|_2 \ .$$

Since $\lim_{k\to\infty} x^{(k)} = x$ this implies that

$$\|x - x^{(i)}\|_2 \leq \frac{r}{1-r}\|x^{(i)} - x^{(i-1)}\|_2.$$

□

The result of this theorem can be used to give a stopping criterion for linear convergent methods. Sometimes the iterations are stopped if $\|x^{(i)} - x^{(i-1)}\|_2$ is small enough. If r is close to one this may lead to inaccurate results since $\frac{r}{1-r}\|x^{(i)} - x^{(i-1)}\|_2$ and thus $\|x - x^{(i)}\|_2$ may be large. A safe stopping criterion is: stop if

$$\frac{r}{1-r}\frac{\|x^{(i)} - x^{(i-1)}\|_2}{\|x^{(i)}\|_2} \leq \epsilon.$$

If this condition holds then the relative error is less than ε:

$$\frac{\|x - x^{(i)}\|_2}{\|x\|_2} \cong \frac{\|x - x^{(i)}\|_2}{\|x^{(i)}\|_2} \leq \frac{r}{1-r}\frac{\|x^{(i)} - x^{(i-1)}\|_2}{\|x^{(i)}\|_2} \leq \epsilon.$$

Furthermore, Theorem 1.4 yields the following result:

$$\|x - x^{(i)}\|_2 \leq \frac{r^i}{1-r}\|x^{(1)} - x^{(0)}\|_2 \ . \tag{13}$$

So assuming that the expression (13) can be replaced by an equality

$$\log \ \|x - x^{(i)}\|_2 \ = \ i \log\,(r) + \log\,\left(\frac{\|x^{(1)} - x^{(0)}\|_2}{1-r}\right) \ . \tag{14}$$

This implies that the curve $\log \ \|x - x^{(i)}\|_2$ is a straight line as function of i. This was the motivation for the term linear convergent process. Given the quantity r, which is also known as the rate of convergence, or reduction factor, the required accuracy and $\|x^{(1)} - x^{(0)}\|_2$ it is possible to estimate the number of iterations to achieve this accuracy. In general r may be close to one and hence a small increase of r may lead to a large increase of the required number of iterations.

For iterative methods, which have another convergence behavior most stopping criteria are based on the norm of the residual. Below we shall give some of these criteria and give comment

on their properties.

Criterion 1 $\|b - Ax^{(i)}\|_2 \leq \epsilon$.

The main disadvantage of this criterion is that it is not scaling invariant. This implies that if $\|b - Ax^{(i)}\|_2 < \epsilon$ this does not hold for $\|100(b - Ax^{(i)})\|_2$. Although the accuracy of $x^{(i)}$ remains the same. So a correct choice of ϵ depends on properties of the matrix A.

The remaining criteria are all scaling invariant.

Criterion 2 $\frac{\|b-Ax^{(i)}\|_2}{\|b-Ax^{(0)}\|_2} \leq \epsilon$

The number of iterations is independent of the initial estimate $x^{(0)}$. This may be a drawback since a better initial estimate does not lead to a decrease of the number of iterations.

Criterion 3 $\frac{\|b-Ax^{(i)}\|_2}{\|b\|_2} \leq \epsilon$

This is a good termination criterion. The norm of the residual is small with respect to the norm of the right-hand side. Replacing ϵ by $\epsilon/K_2(A)$ we can show that the relative error in x is less than ϵ. It follows (compare Theorem **??**) that:

$$\frac{\|x - x^{(i)}\|_2}{\|x\|_2} \leq K_2(A)\frac{\|b - Ax^{(i)}\|_2}{\|b\|_2} \leq \epsilon.$$

In general $\|A\|_2$ and $\|A^{-1}\|_2$ are not known. Some iterative methods gives approximations of these quantities.

Criterion 4 $\frac{\|b-Ax^{(i)}\|_2}{\|x^{(i)}\|_2} \leq \epsilon/\|A^{-1}\|_2$

This criterion is closely related to Criterion 3. In many cases this criterion also implies that the relative error is less then ϵ:

$$\frac{\|x - x^{(i)}\|_2}{\|x\|_2} \cong \frac{\|x - x^{(i)}\|_2}{\|x^{(i)}\|_2} = \frac{\|A^{-1}(b - Ax^{(i)})\|_2}{\|x^{(i)}\|_2} \leq \frac{\|A^{-1}\|_2\|b - Ax^{(i)}\|_2}{\|x^{(i)}\|_2} \leq \epsilon$$

Sometimes physical relations lead to other termination criteria. This is the case if the residual has a physical meaning, for instance the residual is equal to some energy or the deviation from a divergence free vector field etc.

In Theorem **??** we have seen that due to rounding errors the solution x represented on a computer has a residual which may be of the magnitude of $u\|b\|_2$, where u is the machine precision. So we cannot expect that a computed solution by an iterative solution method has a smaller norm of the residual. In a good implementation of an iterative method, a warning is given if the required accuracy is too high. If for instance the termination criterion is $\|b-Ax^{(i)}\|_2 \leq \epsilon$ and ϵ is chosen less then $1000u\|b\|_2$ a warning should be given and ε should be replaced by $\epsilon = 1000u\|b\|_2$. The arbitrary constant 1000 is used for safety reasons.

1.4 Exercises

1. Suppose $\|E\| < 1$ for some matrix $E \in \mathbb{R}^{n\times n}$. Show that

$$(I-E)^{-1} = \sum_{k=0}^{\infty} E^k \text{ and } \|(I-E)^{-1}\| \leq \frac{1}{1-\|E\|}.$$

2. Show that if A is strictly diagonal dominant then the Gauss Seidel method converges.

3. Suppose that A is symmetric and positive definite.

   (a) Show that one can write $A = D_A - L_A - L_A^T$ where D_A is diagonal with $d_{ii} > 0$ for each $1 \leq i \leq n$ and L_A is strictly lower triangular. Further show that $D_A - L_A$ is nonsingular.

   (b) Let $T_g = (D_A - L_A)^{-1}L_A^T$ and $P = A - T_g^T A T_g$. Show that P is symmetric.

   (c) Show that T_g can also be written as $T_g = I - (D_A - L_A)^{-1}A$.

   (d) Let $Q = (D_A - L_A)^{-1}A$. Show that $T_g = I - Q$ and

$$P = Q^T(AQ^{-1} - A + (Q^T)^{-1}A)Q.$$

   (e) Show that $P = Q^T D_A Q$ and P is symmetric and positive definite.

   (f) Let λ be an eigenvalue of T_g with eigenvector x. Use part (b) to show that $x^T P x > 0$ implies that $|\lambda| < 1$.

   (g) Show that the Gauss Seidel method converges.

4. Extend the method of proof in Exercise 3 to the SOR method with $0 < \omega < 2$.

5. Suppose that $\tilde{\mu}_1$ is an estimate for μ_1 and $\tilde{\mu}_n$ for μ_n.

   (a) Show that in general the Chebyshev method converges slower if $0 < \tilde{\mu}_1 < \mu_1$ and $\tilde{\mu}_n > \mu_n$ if $\tilde{\mu}_1$ and $\tilde{\mu}_n$ are used in the Chebyshev method.

   (b) Show that divergence can occur if $\tilde{\mu}_n < \mu_n$.

6. (a) Do two iterations with Gauss Jacobi to the system:

$$\begin{pmatrix} 2 & 0 \\ -2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

   Note that the second iterate is equal to the exact solution.

   (b) Is the following claim correct?

   The Gauss Jacobi method converges in mostly n iterations if A is a lower triangular matrix

2 A Krylov subspace method for systems with a symmetric positive definite matrix

2.1 Introduction

In the basic iterative solution methods we compute the iterates by the following recursion:

$$x_{i+1} = x_i + M^{-1}(b - Ax_i) = x_i + M^{-1}r_i$$

Writing out the first steps of such a process we obtain:

$$\begin{aligned} x_0 &\quad , \\ x_1 &= x_0 + (M^{-1}r_0), \\ x_2 &= x_1 + (M^{-1}r_1) = x_0 + M^{-1}r_0 + M^{-1}(b - Ax_0 - AM^{-1}r_0) \\ &= x_0 + 2M^{-1}r_0 - M^{-1}AM^{-1}r_0, \\ &\vdots \end{aligned}$$

This implies that

$$x_i \in x_0 + \text{ span}\left\{M^{-1}r_0, M^{-1}A(M^{-1}r_0), \ldots, (M^{-1}A)^{i-1}(M^{-1}r_0)\right\}.$$

The subspace $K^i(A; r_0) := \text{span}\left\{r_0, Ar_0, \ldots, A^{i-1}r_0\right\}$ is called the Krylov-space of dimension i corresponding to matrix A and initial residual r_0. An x_i calculated by a basic iterative method is an element of $x_0 + K^i(M^{-1}A; M^{-1}r_0)$.

In the preceding chapter we tried to accelerate convergence by the Chebyshev method. In this method one approximates the solution x by a vector $x_i \in x_0 + K^i(M^{-1}A; M^{-1}r_0)$ such that $\|x - x_i\|_2$ is minimized in a certain way. One of the drawbacks of that method is that information on the eigenvalues of $M^{-1}A$ should be known. In this chapter we shall describe the Conjugate Gradient method. This method minimizes the error $x - x_i$ in an adapted norm, without having to know any information about the eigenvalues. In Section 2.3 we give theoretical results concerning the convergence behavior of the CG method.

2.2 The Conjugate Gradient (CG) method

In this section we assume that $M = I$, and $x_0 = 0$ 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 satisfies the following condition.

Condition 3.2.1
The matrix 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 could be to construct a vector $x_i \in K^i(A, r_0)$ such that $\|x - x_i\|_2$ is minimal. The first iterate x_1 can be written as $x_1 = \alpha_0 r_0$ where α_0 is a constant which has to be chosen such that $\|x - x_1\|_2$ is minimal. This leads to

$$\|x - x_1\|_2^2 = (x - \alpha_0 r_0)^T (x - \alpha_0 r_0) = x^T x - 2\alpha_0 r_0^T x + \alpha_0^2 r_0^T r_0 \ . \tag{15}$$

The norm given in (15) is minimized if $\alpha_0 = \frac{r_0^T x}{r_0^T r_0}$. Since x is unknown this choice cannot be determined, so this idea does not lead to a useful method. Note that $Ax = b$ 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 3.2.2
The A-inner product is defined by

$$(y, z)_A = y^T A z,$$

and the A-norm by $\|y\|_A = \sqrt{(y, y)_A} = \sqrt{y^T A y}$.

It is easy to show that if A satisfies Condition 3.2.1 $(.,.)_A$ and $\|.\|_A$ satisfy the rules for inner product and norm (see Section **??**) respectively. In order to obtain x_1 such that $\|x - x_1\|_A$ is minimal we note that

$$\|x - x_1\|_A^2 = x^T A x - 2\alpha_0 r_0^T A x + \alpha_0^2 \ r_0^T A r_0,$$

so $\alpha_0 = \frac{r_0^T A x}{r_0^T A r_0} = \frac{r_0^T b}{r_0^T A r_0}$. We see that this new inner product leads to a minimization problem, which can be easily solved. In the next iterations we compute x_i such that

$$\|x - x_i\|_A = \min_{y \in K^i(A; r_0)} \|x - y\|_A \tag{16}$$

The solution of this minimization problem leads to the conjugate gradient method. First we specify the CG method, thereafter we summarize some of its properties.

Conjugate Gradient method

$k = 0$;	$x_0 = 0$; $r_0 = b$	initialization
while	$r_k \neq 0$ do	termination criterion
	$k := k + 1$	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 x_{k-1} to x_k
	end if	
	$\alpha_k = \frac{r_{k-1}^T r_{k-1}}{p_k^T A p_k}$	
	$x_k = x_{k-1} + \alpha_k p_k$	update iterate
	$r_k = r_{k-1} - \alpha_k A p_k$	update residual
end while		

The first description of this algorithm is given in [25]. For recent results see [47]. Besides the two vectors x_k, r_k and matrix A only one extra vector p_k should be stored in memory. Note that the vectors from the 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 1.3. 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 $r_k = b - Ax_k$. In some applications the updated residual obtained from the CG algorithm can deviate much from the exact residual $b - Ax_k$ due to rounding errors. So it is strongly recommended to recompute $b - Ax_k$ 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 x_k as its starting vector.

The vectors defined in the CG method have the following properties:

Theorem 2.1

$$\text{1.} \qquad span\ \{p_1, \ldots, p_k\} = span\,\{r_0, \ldots, r_{k-1}\} = K^k(A; r_0), \tag{17}$$

$$\text{2.} \qquad r_j^T r_i = 0 \quad i = 0, \ldots, j-1 \quad ; \quad j = 1, \ldots, k \quad , \tag{18}$$

$$\text{3.} \qquad r_j^T p_i = 0 \quad i = 1, \ldots, j \quad ; \quad j = 1, \ldots, k \quad , \tag{19}$$

$$\text{4.} \qquad p_j^T A p_i = 0 \quad i = 1, \ldots, j-1 \quad ; \quad j = 2, \ldots, k \tag{20}$$

$$\text{5.} \qquad \|x - x_k\|_A = \min_{y \in K^k(A; r_0)} \|x - y\|_A. \tag{21}$$

Proof: see [19], Section 10.2.

Remarks on the properties given in Theorem 2.1

- It follows from (17) and (18) 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 $\|x - y\|_A$ is minimized over $K^n(A; r_0) = \mathbb{R}^n$ the norm is equal to zero and $x_n = x$. 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 do n iterations, secondly even if n is small, rounding errors can spoil the results such that the properties given in Theorem 2.1 do not hold for the computed vectors.
- The sequence $\|x - x_k\|_A$ is monotone decreasing, so

  $$\|x - x_{k+1}\|_A \leq \|x - x_k\|_A \ .$$

  This follows from (21) and the fact that $K^k(A; r_0) \subset K^{k+1}(A; r_0)$. In practice $\|x - x_k\|_A$ is not easy to compute since x is unknown. The norm of the residual is given by $\|r_k\|_2 = \|x - x_k\|_{A^T A}$. This sequence is not necessarily monotone 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

$$\|r_k\|_2 = \|Ax_k - b\|_2 \leq \sqrt{\|A\|_2}\|x - x_k\|_A$$

shows that $\|r_k\|_2$ is less than the monotone decreasing sequence $\sqrt{\|A\|_2}\|x - x_k\|_A$, 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 to calculate 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 $x_{k-2} = x$. The linear system is solved. The denominator of α_k is zero if $p_k^T A p_k = 0$ so $p_k = 0$. Using property (17) this implies that $r_{k-1} = 0$ so again the problem is already solved.
  Conclusion: If the matrix A satisfies Condition 3.2.1 then the CG method is robust.

In a following chapter we shall give CG type methods for general matrices A. But first we shall extend Condition 3.2.1 in such a way that also singular matrices are permitted. If the matrix A is symmetric and positive semi definite ($x^T Ax \geq 0$) the CG method can be used to solve the linear system $Ax = b$, 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 is no vector x such that $Ax = b$. For further details and references see [27].

2.3 The convergence behavior of the CG method

An important research 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 2.2 *The iterates x_k obtained from the CG algorithm satisfy the following inequality:*

$$\|x - x_k\|_A \leq 2\left(\frac{\sqrt{K_2(A)} - 1}{\sqrt{K_2(A)} + 1}\right)^k \|x - x_0\|_A.$$

Proof
We shall only give a sketch of the proof. It is easily seen that $x - x_k$ can be written as a polynomial, say $p_k(A)$ with $p_k(0) = 1$, times the initial residual (compare the Chebyshev method)

$$\|x - x_k\|_A = \|p_k(A)(x - x_0)\|_A.$$

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:

$$\|x - x_k\|_A \leq \|q_k(A)(x - x_0)\|_A.$$

The right-hand side can be written as

$$\|q_k(A)(x - x_0)\|_A = \|q_k(A)\sqrt{A}(x - x_0)\|_2 \leq \|q_k(A)\|_2\|\sqrt{A}(x - x_0)\|_2 = \|q_k(A)\|_2\|x - x_0\|_A$$

Taking $q_k(A)$ equal to the Chebyshev polynomial gives the desired result. $\square$

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 very popular. The reason for this is that the convergence can be slow for systems where the condition number $K_2(A)$ is very large. On the other hand the fact that the solution is found after n iteration is also not useful in practice. Firstly n may be very large, secondly the property does not hold in the presence of rounding errors. To illustrate this we consider the following classical example:

Example 1
The linear system $Ax = b$ should be solved where $n = 40$ and $b = (1, 0, \ldots, 0)^T$. 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 seen as a finite difference discretization of the bending beam equation: $u'''' = f$. The eigenvalues of this matrix are given by:

$$\lambda_k = 16\sin^4\frac{k\pi}{82} \quad k = 1, \ldots, 40.$$

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 $\left(\frac{82}{\pi}\right)^4$. The resulting rate of convergence given by

$$\frac{\sqrt{K_2(A)} - 1}{\sqrt{K_2(A)} + 1} \cong 0.997$$

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 found. In Figure 2 the convergence behavior is given where the rounding error is equal to 10^{-16}, [18]. This example suggests that CG has only a restricted range of applicability. These ideas however changed after the publication of [35]. Herein it is 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. These problems originate 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 2.2. This inequality is an upper bound for the error of the CG iterates, and suggests that the CG method is a linearly convergent process (see Figure 3). However, in practice the convergence behavior looks like the one given in Figure 4. This is called superlinear convergence behavior.

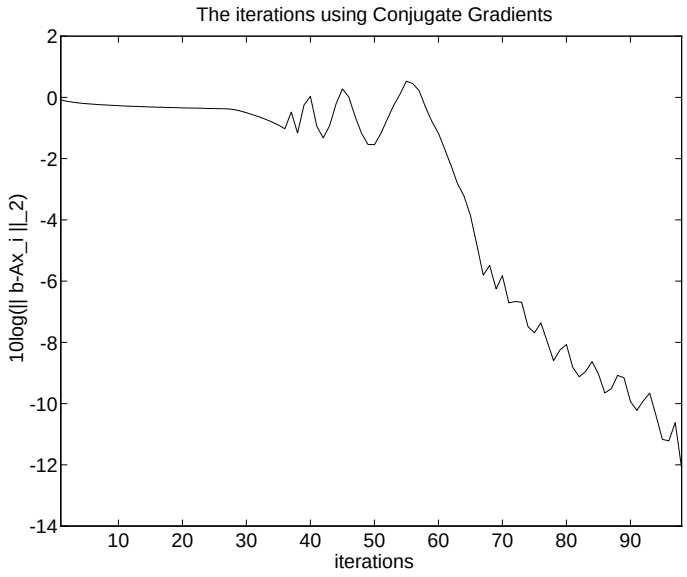

Figure 2: The convergence behavior of CG applied to Example 1.

So the upper bound is only sharp for the initial iterates. It seems that after some iterations the condition number in Theorem 2.2 is replaced by a smaller "effective" condition number. To illustrate this we give the following example:

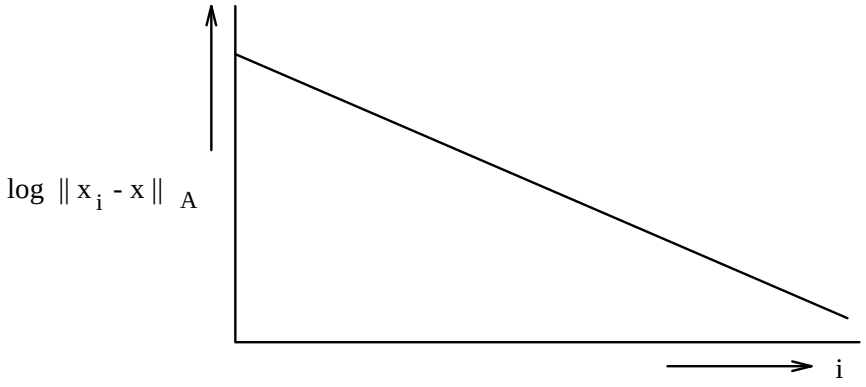

Figure 3: A linear convergent behavior

Example 2

The matrix A is the discretized Poisson operator. The physical domain is the 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

$$\lambda_{k,l} = 4 - 2\cos\frac{\pi k}{31} - 2\cos\frac{\pi l}{31} \quad , \quad 1 \leq k, l \leq 30.$$

Using Theorem 2.2 it appears that 280 iteration are necessary to ensure that

$$\frac{\|x - x_i\|_A}{\|x - x_0\|_A} \leq 10^{-12}.$$

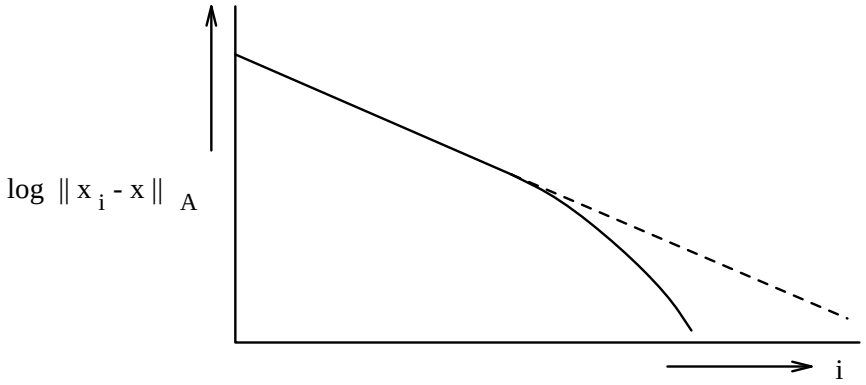

Figure 4: A super linear convergent behavior

Computing the solution it appears that CG iterates satisfy the given termination criterion after 120 iterations. So in this example the estimate given in Theorem 2.2 is not sharp.

To obtain a better idea of the convergence behavior we have a closer look to the CG method. We have seen that CG minimizes $\|x - x_i\|_A$ on 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

$$\|x - x_i\|_A = \|q_i(A)(x - x_0)\|_A = \min_{\substack{\tilde{q_i}, \\ \tilde{q}_i(0) = 1}} \|\tilde{q}_i(A)(x - x_0)\|_A \ .$$

Suppose that the orthonormal eigen system 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 $x - x_0 = \sum_{j=1}^{n} \gamma_j y_j$, which implies that

$$x - x_i = \sum_{j=1}^{n} \gamma_j q_i(\lambda_j) y_j \ . \tag{22}$$

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 in $\tilde{y}_1$ and $\tilde{y}_2$ such that $\tilde{\gamma}_1 \neq 0$ but $\tilde{\gamma}_2 = 0$. This combined with equation (22) implies that if $q_i(\lambda_j) = 0$ for all different λ_j then $x_i = x$. 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 2.2 can be sharpened.

Remark

For a given linear system $Ax = b$ and a given x_0 (note that $x - x_0 = \sum_{j=1}^{n} \gamma_j y_j$) 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:

$$\|x - x_i\|_A \leq 2 \left(\frac{\sqrt{\frac{\beta}{\alpha}} - 1}{\sqrt{\frac{\beta}{\alpha}} + 1} \right)^i \|x - x_0\|_A. \tag{23}$$

The ratio $\frac{\beta}{\alpha}$ is called the effective condition number of A.

It follows from Theorem 2.1 that $r_0, \ldots, r_{k-1}$ forms an orthogonal basis for $K^k(A; r_0)$. So 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

$$R_k \in I\!R^{n \times k} \text{ and the } j^{th} \text{ column of } R_k \text{ is } \tilde{r}_j,$$
$$T_k = R_k^T A R_k \text{ where } T_k \in I\!R^{k \times k}.$$

The Ritzmatrix T_k can be seen as the projection of A on $K^k(A; r_0)$. It follows from Theorem 2.1 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 eigenvalues and eigenvectors and play an important role in a better understanding of the convergence behavior of CG. The properties of the Ritzvalues are given in more detail in Chapter 6. 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 the 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 seems 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 2.1 it follows that $r_k = A(x - x_k)$ 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 two combined yields that $(A(x - x_k))^T y_j \cong 0$. The exact solution and the approximation can be written as

$$x = \sum_{i=1}^{n} (x^T y_i) y_i \quad \text{and} \quad x_k = \sum_{i=1}^{n} (x_k^T y_i) y_i.$$

From $(A(x - x_k))^T y_j = (x - x_k)^T \lambda_j y_j \cong 0$ it follows that $x^T y_j \cong x_k^T y_j$. So the error $x - x_k$ has a negligible component in the eigenvector y_j. This suggest 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 original process.

Definition
Let x_i be the i-th iterate of the CG process for $Ax = b$. For a given integer i let $\boldsymbol{x}_j$ denote the j-th iterate of the comparison CG process for this equation, starting with $\boldsymbol{x}_0$ such that $x - \boldsymbol{x}_0$ is the projection of $x - x_i$ on $\text{span}\{y_2, \ldots, y_n\}$.

Note that for the comparison process the initial error has no component in the y_1 eigenvector.

Theorem 2.3 *[44]*
Let x_i be the i-th iterate of CG, and $\boldsymbol{x}_j$ the j-th iterate of the comparison process. Then for any j there holds:

$$\|x - x_{i+j}\|_A \leq F_i \|x - \boldsymbol{x}_j\|_A \leq F_i \frac{\|x - \boldsymbol{x}_j\|_A}{\|x - \boldsymbol{x}_0\|_A} \|x - x_i\|_A$$

with $F_i = \frac{\theta_1^{(i)}}{\lambda_1} \max_{k \geq 2} \frac{|\lambda_k - \lambda_1|}{|\lambda_k - \theta_1^{(i)}|}$ *, where $\theta_1^{(i)}$ is the smallest Ritzvalue in the i-th step of the CG process.*

Proof: see [44], Theorem 2.1.

The theorem shows that from any stage i on for which $\theta_1^{(i)}$ does not coincide with an eigenvalue λ_k, the error reduction in the next j steps is at most the 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 $\theta_1^{(i)}$ 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 $\theta_1^{(i)}$ 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 [44].

2.4 Exercises

1. Show that $(y, z)_A = \sqrt{y^T A z}$ is an inner product if A is symmetric and positive definite.

2. Give the proof of inequality (23).

3. (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 $z = 0$.

4. Define
$$t_k = \frac{(v^{(k)}, b - Ax^{(k-1)})}{(v^{(k)}, Av^{(k)})}$$
and $x^{(k)} = x^{(k-1)} + t_k v^{(k)}$, 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 $(r^{(l+1)}, v^{(l+1)}) = 0$.

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 are going to solve $Ax = b$.

   (a) Show that Conjugate Gradients applied to this system should convergence in 1 or 2 iterations (using the convergence theory).

   (b) Choose $x^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ and do 2 iterations with the Conjugate Gradients method.

6. Suppose that A is nonsingular, symmetric, and indefinite. Give an example to show that the Conjugate Gradients method can break down.

3 Preconditioning of Krylov subspace methods

We have seen that the convergence behavior of Krylov subspace methods depends strongly 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

$$M^{-1}Ax = M^{-1}b\,,$$

and has the same solution as the original system $Ax = b$. 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 $M^{-1}y$ with low cost.

Most of this chapter contains preconditioners for symmetric positive definite systems (Section 3.1). For non-symmetric systems the ideas are analogously, so in Section 3.2 we give some details, which can be used only for non-symmetric systems.

3.1 The Preconditioned Conjugate Gradient (PCG) method

In Section 2.3 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 the $\frac{\lambda_n}{\lambda_1}$ is replaced by the effective condition number $\frac{\lambda_n}{\lambda_2}$ etc. So the question arises, is it possible to change the linear system $Ax = b$ 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 $Ax = b$. The idea behind Preconditioned Conjugate Gradients is to apply the "original" Conjugate Gradient method to the transformed system

$$\tilde{A}\tilde{x} = \tilde{b}\,,$$

where $\tilde{A} = P^{-1}AP^{-T}$, $x = P^{-T}\tilde{x}$ 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 $\tilde{}$ sign occurs.

Preconditioned Conjugate Gradient method

$k = 0$;	$x_0 = 0$;	$r_0 = b$;	initialization
while	$(r_k \neq 0)$ do		termination criterion
	$z_k = M^{-1} r_k$		preconditioning
	$k := k + 1$		
	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		
	$\alpha_k = \frac{r_{k-1}^T z_{k-1}}{p_k^T A p_k}$		
	$x_k = x_{k-1} + \alpha_k p_k$		update iterate
	$r_k = r_{k-1} - \alpha_k A p_k$		update residual
end while			

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 costlier 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 x_k obtained by PCG satisfies

$$x_k \in x_0 + K^k (P^{-T} P^{-1} A \; ; \; P^{-T} P^{-1} r_0), \text{ and} \tag{24}$$

$$\|x - x_k\|_A \leq 2 \left(\frac{\sqrt{K_2(P^{-1} A P^{-T})} - 1}{\sqrt{K_2(P^{-1} A P^{-T})} + 1} \right)^k \|x - x_0\|_A \; . \tag{25}$$

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 get the original CG method back, whereas if $P^T P = A$ the iterate x_1 is equal to x so PCG converges in one iteration. For a classical paper on the success of PCG we refer to [29]. 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 [43] it has been shown that this choice minimizes the condition number of $P^{-1}AP^{-T}$ if P is restricted to be a diagonal matrix. For this preconditioner it is advantageous to apply CG to $\tilde{A}\tilde{x} = \tilde{b}$. The reason is that $P^{-1}AP^{-T}$ is easily calculated. Furthermore, $diag\ (\tilde{A}) = 1$ which saves n multiplications in the matrix vector product.

Basic iterative method

The basic iterative methods described in Section 1.2 use a splitting of the matrix $A = M - N$. In the beginning of Section 2.2 we show that the k-th iterate y_k from a basic method is an element of $x_0 + K^k(M^{-1}A, M^{-1}r_0)$. Using this matrix M in the PCG method we see that the iterate x_k obtained by PCG satisfies the following inequality:

$$\|x - x_k\|_A = \min_{z \in K^k(M^{-1}A; M^{-1}r_0)} \|x - z\|_A \ .$$

This implies that $\|x - x_k\|_A \leq \|x - y_k\|_A$, so measured in the $\|\ .\ \|_A$ 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 $A = M - N$ 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 defined in Section 1.1. The coefficient matrix of this problem $A \in I\!R^{n \times n}$ 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.

$$A = \begin{bmatrix} a_1 & b_1 & & c_1 & & & & & \\ b_1 & a_2 & b_2 & & c_2 & & & & \\ \vdots & \ddots & \ddots & & & \ddots & & & \emptyset \\ c_1 & & b_m & a_{m+1} & b_{m+1} & & c_{m+1} & & \\ & \ddots & \emptyset & \ddots & \ddots & \ddots & \emptyset & \ddots & \\ & \emptyset & & & & & & & \end{bmatrix} \tag{26}$$

An optimal choice with respect to converge is take a lower triangular matrix L such that $A = L^T L$ 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. To specify this in a precise way we use the following definition:

Definition
The matrix $A = (a_{ij})$ is an M-matrix if $a_{ij} \leq 0$ for $i \neq j$, the inverse A^{-1} exists and has positive elements $(A^{-1})_{ij} \geq 0$.

The matrix of our model problem is an M-matrix. Furthermore, we give a notation for these elements of L which 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 3.1 *If A is a symmetric M-matrix, there exists for each $Q \subset Q_n$ having 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*

$$LL^T x_{i+1} = Rx_i + b \quad \textit{for every choice } x_0\ ,$$

where $x_i \to x = A^{-1}b$.

Proof (see [29]; 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
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:

$$L = \begin{bmatrix} \tilde{d}_1 & & & & & \\ \tilde{b}_1 & \tilde{d}_2 & & & & \\ & \ddots & \ddots & \emptyset & & \\ \tilde{c}_1 & & \tilde{b}_m & \tilde{d}_{m+1} & & \\ & \ddots & \emptyset & \ddots & \ddots & \\ \emptyset & & & & & \end{bmatrix} \tag{27}$$

it is easy to see that (using the notation as given in (26))

$$\left.\begin{array}{rcl} \tilde{d}_i &=& a_i - \frac{b_{i-1}^2}{\tilde{d}_{i-1}} - \frac{c_{i-m}^2}{\tilde{d}_{i-m}} \\ \tilde{b}_i &=& b_i \\ \tilde{c}_i &=& c_i \end{array}\right\} \quad i = 1, ..., n \ . \tag{28}$$

where elements that are not defined should be 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 ([29]; 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

$$Q^3 = \{(i,j)| \ \ |i-j| \neq 1, 2, m-2, m-1, m\}$$

For the formula's to obtain the decomposition we refer to ([29]; 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 [29]. As a first example we consider the model problem, where the boundary conditions are somewhat different:

$$\begin{array}{lll} \frac{\partial u}{\partial x}(x,y) = 0 & \text{for} & \left\{ \begin{array}{ll} x = 0 \ , & y \in [0,1] \\ x = 1 \ , & y \in [0,1] \end{array} \right. \ , \\ \frac{\partial u}{\partial y}(x,y) = 0 & \text{for} & y = 1 \ , \ x \in [0,1] \ , \\ u(x,y) = 1 & \text{for} & y = 0 \ , \ x \in [0,1] \ . \end{array}$$

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 5.

From inequality (25) it follows that the rate of convergence can be bounded by

$$r = \frac{\sqrt{K_2(P^{-1}AP^{-T})} - 1}{\sqrt{K_2(P^{-1}AP^{-T})} + 1}. \tag{29}$$

To obtain a better insight in the fast convergence of ICCG(0) and ICCG(3) the eigenvalues of $A, (L_0 L_0^T)^{-1}A$ and $(L_3 L_3^T)^{-1}A$ are computed and given in Figure 6. For this result given in [29] a small matrix of order $n = 36$ is used, so all eigenvalues can be calculated.
The eigenvalues as given in Figure 6 can be substituted in formula (29). We then obtain

$$\begin{array}{lll} r = 0.84 & \text{for} & \text{CG} \ , \\ r = 0.53 & \text{for} & \text{ICCG(0)} \ , \\ r = 0.23 & \text{for} & \text{ICCG(3)} \ , \end{array} \tag{30}$$

which explains the fast convergence of the PCG variants. In our explanation of the convergence behavior we have also used the notion of Ritz values. Applying these ideas to the given methods we note the following:

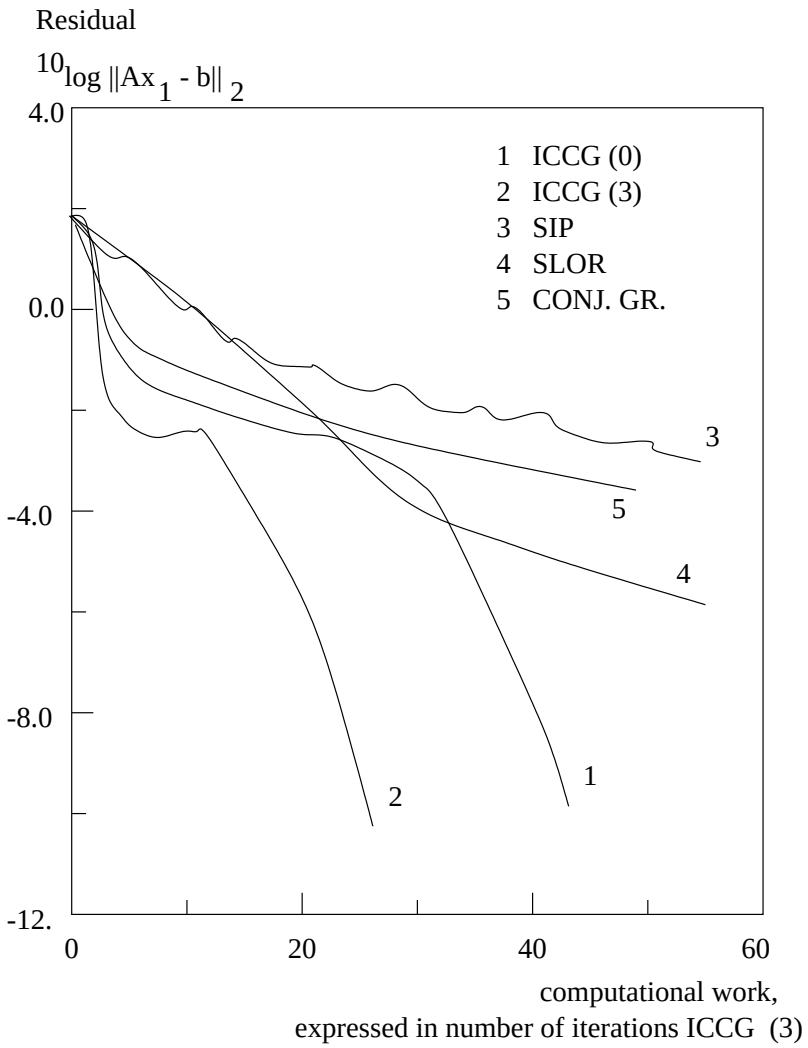

Figure 5: The results for the CG, ICCG(0) and ICCG(3) methods, compared with SIP (Strongly Implicit Procedure) and SLOR (Successive Line Over Relaxation method)

- 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 5, the CG method has a linear convergent behavior.

- For the PCG method the eigenvalue distribution is very different. Looking to the spectrum of $(L_3 L_3^T)^{-1} A$ 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 Ritz-value 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 super linear convergence is expected. This again agrees very well with the results given in Figure 5.

So the faster convergence of ICCG(3) comes from a smaller condition number and a more favorable distribution of the internal eigenvalues.

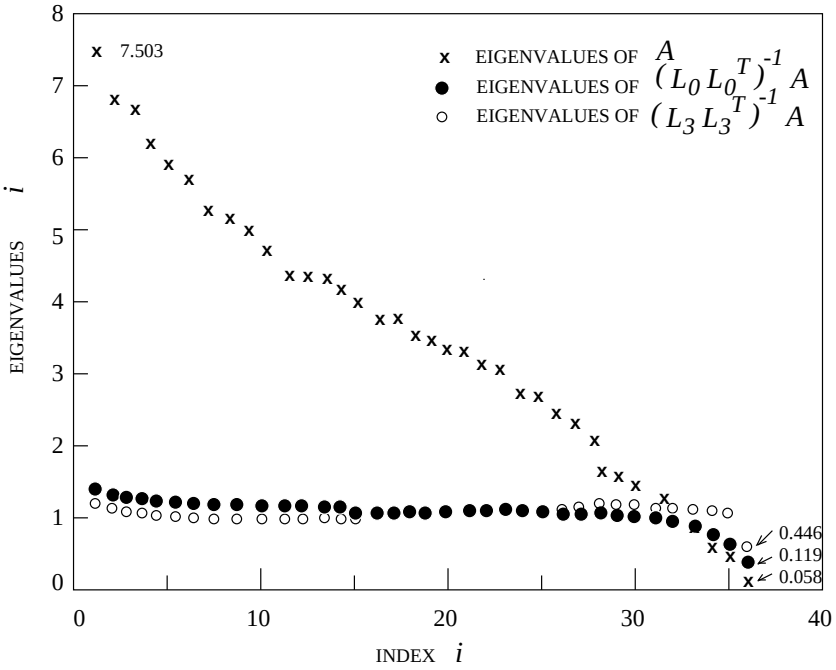

Figure 6: The eigenvalues of A, $(L_0L_0^T)^{-1}A$ and $(L_3L_3^T)^{-1}A$.

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, in order that the magnitude of each entry of the residual vector was below some fixed small number ε. In Figure 7 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 $K_2(A)$ 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 before $\frac{1}{h^2}$ is less for the ICCG(0) preconditioned system than for A and the distribution of the internal eigenvalues is much better so super linear 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 [23] (MICCG means Modified ICCG) and RICCG(0) given in [2] (RICCG means Relaxed ICCG).

MICCG

In the MICCG method the MIC preconditioner is constructed by slightly adapted rules. Again A is splitted 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 $a_{ij} = 0 \;\; i > j$,

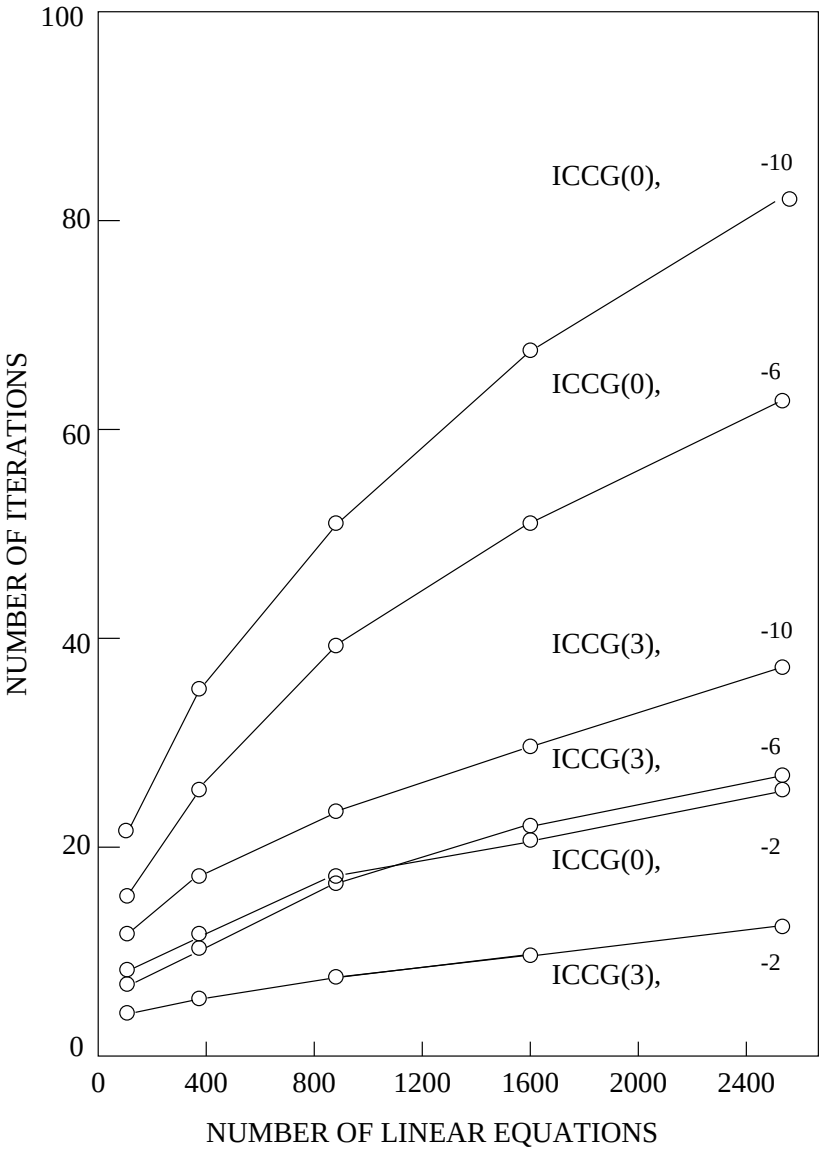

Figure 7: Effect of number of equations on the rate of convergence

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 $a_{ij} \neq 0 \;\; i > j$.

A consequence of c) is that $LD^{-1}L^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = A \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ 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 approximation for the inverse of A with respect to x and/or b. Using the same notation of L as given in (27) we obtain

$$\left.\begin{array}{rcl} \tilde{d}_i & = & a_i - (b_{i-1} + c_{i-1})\frac{b_{i-1}}{\tilde{d}_{i-1}} - (b_{i-m} + c_{i-m})\frac{c_{i-m}}{\tilde{d}_{i-m}} \\ \tilde{b}_i & = & b_i \\ \tilde{c}_i & = & c_i \end{array}\right\} \quad i = 1, .., n \tag{31}$$

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 super linear convergence sets in. Using MICCG the largest Ritz values 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 advantage of both methods the RIC preconditioner is proposed in [2], which is an average of the IC and MIC preconditioner. For the details we refer to [2]. 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:

$$\left.\begin{array}{rcl} \tilde{d}_i & = & a_i - (b_{i-1} + \alpha c_{i-1})\frac{b_{i-1}}{\tilde{d}_{i-1}} - (\alpha b_{i-m} + c_{i-m})\frac{c_{i-m}}{\tilde{d}_{i-m}} \\ \tilde{b}_i & = & b_i \\ \tilde{c}_i & = & c_i \end{array}\right\} \quad i = 1, ..., n \tag{32}$$

However the question remains: how to choose α? In Figure 8 which is copied from [45] 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.

Diagonal scaling

The above given preconditioners IC, MIC and RIC can be optimized with respect to work. One way to do this is to look at the explicitly preconditioned system:

$$D^{1/2}L^{-1}AL^{-T}D^{1/2}y = D^{1/2}L^{-1}b \tag{33}$$

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 this in such a way that the main diagonal is equal to the identity matrix. One idea could be to replace (33) by

$$D^{1/2}L^{-1}D^{1/2}D^{-1/2}AD^{-1/2}D^{1/2}L^{-T}D^{1/2}y = D^{1/2}L^{-1}D^{1/2}D^{-1/2}b\,.$$

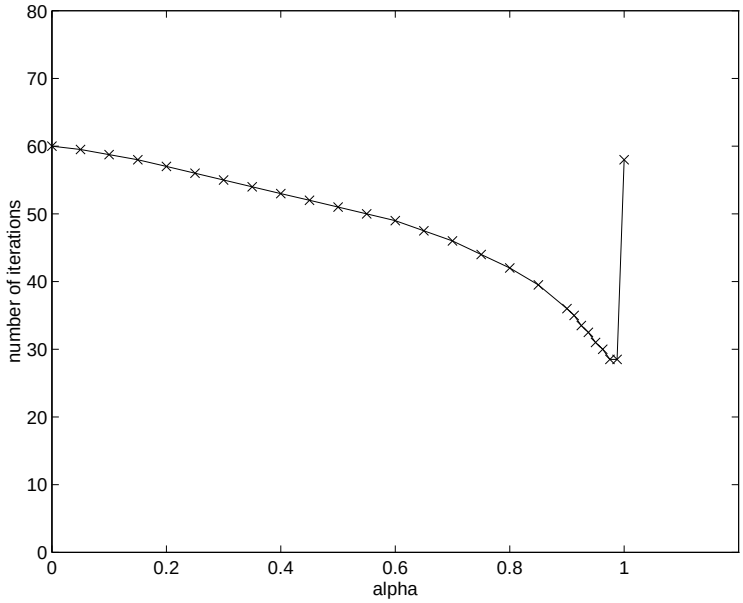

Figure 8: Convergence in relation with α

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

$$\tilde{L}^{-1}\tilde{A}\tilde{L}^{-T}y = \tilde{L}\tilde{b}\,. \tag{34}$$

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 than a CG iteration. In [9] it is shown that much of the extra work can be avoided. If CG is applied to (34) 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:

$$\begin{aligned} v_{j+1} &= \tilde{L}^{-1}\tilde{A}\tilde{L}^{-T}v_j = \tilde{L}^{-1}(\tilde{L} + \tilde{A} - \tilde{L} - \tilde{L}^T + \tilde{L}^T)\tilde{L}^{-T}v_j \\ &= \tilde{L}^{-T}v_j + \tilde{L}^{-1}(v_j + (diag\ (\tilde{A}) - 2I)\tilde{L}^{-T}v_j) \end{aligned} \tag{35}$$

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 irregular distributed than for 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 $a_{ij} \neq 0 \;\; i > j$
$(LD^{-1}L^T)_{ii} = a_{ii} \quad i = 1, ..., n.$

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.

3.2 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 much more difficult to prove that the decomposition does not break down or that the resulting preconditioned system has a spectrum which leads to a fast convergence. Since symmetry is no longer important the number of possible preconditioners is much larger. Furthermore, if we have an incomplete LU decomposition of A, we can apply the iterative methods from 4.3.3 to the following three equivalent systems of equations:

$$U^{-1}L^{-1}Ax = U^{-1}L^{-1}b\,, \tag{36}$$

$$L^{-1}AU^{-1}y = L^{-1}b\,, \quad x = U^{-1}y\,, \tag{37}$$

or

$$AU^{-1}L^{-1}y = b\,, \quad x = U^{-1}L^{-1}y\,. \tag{38}$$

The rate of convergence is approximately the same for all variants. When the Eisenstat implementation is applied one should use (37). Otherwise we prefer (38) because then the stopping criterion is based on $\|r\|_2 = \|b - Ax_k\|_2$ whereas for (36) it is based on $\|U^{-1}L^{-1}r_k\|_2$, and for (37) it is based on $\|L^{-1}r_k\|_2$.

3.3 Exercises

1. Derive the preconditioned CG method using the CG method applied to $\tilde{A}\tilde{x} = \tilde{b}$.

2. (a) Show that the formula's given in (28) are correct.
   (b) Show that the formula's given in (31) are correct.

3. (a) Suppose that $a_i = 4$ and $b_i = -1$. Show that $\lim_{i\to\infty} \tilde{d}_i = 2 + \sqrt{3}$, where $\tilde{d}_i$ is as defined in (28).
   (b) Do the same for $a_i = 4$, $b_i = -1$ and $c_i = -1$ with $m = 10$, and show that $\lim_{i\to\infty} \tilde{d}_i = 2 + \sqrt{2}$.
   (c) Prove that the $LD^{-1}L^T$ decomposition (28) exists if $a_i = a, b_i = b, c_i = c$ and A is diagonally dominant.

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 (28) 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.

4 Krylov subspace methods for general matrices

4.1 Introduction

In the preceding chapter we discuss the Conjugate Gradient method. This Krylov subspace method can only be used if the coefficient matrix is symmetric and positive definite. In this chapter we discuss Krylov subspace methods for an increasing class of matrices. For these we give different iterative methods, and at this moment there is no method which is the best for all cases. This is in contrast with the symmetric positive definite case. In Subsection 4.3.4 we give some guidelines for choosing an appropriate method for a given system of equations. In Section 4.2 we consider symmetric indefinite systems. General real matrices are the subject of Section 4.3. We end this chapter with a section containing iterative methods for complex linear systems.

4.2 Indefinite symmetric matrices

In this section we relax the condition that A should be positive definite (Chapter 2), and only assume that A is symmetric. This means that $x^T Ax > 0$ for some x and possibly $y^T Ay < 0$ for some y. For the real eigenvalues this implies that A has positive and negative eigenvalues. For this type of matrices $\|.\|_A$ defines no longer a norm. Furthermore, CG may have a serious break down since $p_k^T Ap_k$ may be zero whereas $\|p_k\|_2$ is not zero. In this section we give two different (but related) methods to overcome these difficulties. These methods are defined in [31].

SYMMLQ

In the CG method we have defined the orthogonal matrix $R_k \in I\!R^{n\times k}$ where the j^{th} column of R_k is equal to the normalized residual $r_j/\|r_j\|_2$. It appears that the following relation holds

$$AR_k = R_{k+1}\bar{T}_k, \tag{39}$$

where $\bar{T}_k \in I\!R^{k+1\times k}$ is a tridiagonal matrix. This decomposition is also known as the Lanczos algorithm for the tridiagonalisation of A ([19]; p.477). This decomposition is always possible for symmetric matrices, also for indefinite ones. The CG iterates are computed as follows:

$$\begin{aligned} x_k &= R_k y_k, \quad \text{where} && (40)\\ R_k^T AR_k y_k &= T_k y_k = R_k^T b. && (41) \end{aligned}$$

Note that T_k consists of the first k rows of $\bar{T}_k$. If A is positive definite then T_k is positive definite. It appears further that in the CG process an LDL^T factorization of T_k is used to solve (41). This can lead to break down because T_k may be indefinite in this section (compare [19]; Section 9.3.1, 9.3.2, 10.2.6). In the SYMMLQ method [31] problem (41) is solved in a stable way by using an LQ decomposition. So T_k is factorized in the following way:

$$T_k = \bar{L}_k Q_k \quad \text{where} \quad Q_k^T Q_k = I \tag{42}$$

with $\bar{L}_k$ lower triangular. For more details to obtain an efficient code we refer to [31] section 5.

MINRES
In SYMMLQ we have solved (41) in a stable way and obtain the "CG" iteration. However since $\|.\|_A$ is no longer a correct norm, the optimality properties of CG are lost. To get these back we can try to use the decomposition

$$AR_k = R_{k+1}\bar{T}_k$$

and minimize the error in the $\|.\|_{A^TA}$ norm. This is a norm if A is nonsingular. This leads to the following approximation:

$$x_k = R_k y_k, \tag{43}$$

where y_k is such that

$$\|Ax_k - b\|_2 = \min_{y \in I\!R^k} \|AR_k y - b\|_2. \tag{44}$$

Note that $\|AR_k y - b\|_2 = \|R_{k+1}\bar{T}_k y - b\|_2$ using (39).

Starting with $x_0 = 0$ implies that $r_0 = b$. Since R_{k+1} is an orthogonal matrix and $b = R_{k+1}\|r_0\|_2 e_1$, where $e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in I\!R^{k+1}$, we have to solve the following least squares problem

$$\min_{y \in I\!R^k} \|\bar{T}_k y - \|r_0\|_2 e_1\|_2 \ . \tag{45}$$

Again for more details see [31]; Section 6.7. A comparison of both methods is given in [31]. In general the rate of convergence of SYMMLQ or MINRES for indefinite symmetric systems of equations is much worse than of CG for definite systems. Preconditioning techniques for these methods are specified in [36].

4.3 Iterative methods for general matrices

In this section we consider iterative methods to solve $Ax = b$ where the only requirement is that $A \in I\!R^{n \times n}$ is nonsingular. In the symmetric case we have seen that CG has the following three nice properties:

- the approximation x_i is an element of $K^i(A; r_0)$,
- optimality, the error is minimal measured in a certain norm,
- short recurrences, only the results of one foregoing step is necessary so work and memory do not increase for an increasing number of iterations.

It is shown in [13] that it is impossible to obtain a Krylov method based on $K^i(A; r_0)$, 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 $K^i(A; r_0)$. Recently some surveys on general iteration methods have been published: [6], [19] Section 10.4, [15], [39], [21], [3].

It appears that there are essentially three different ways to solve non-symmetric linear systems, while maintaining some kind of orthogonality between the residuals:

1. Solve the normal equations $A^TAx = A^Tb$ 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 subsections. An introduction and comparison of these classes is given in [30].

4.3.1 CG applied to the normal equations

The first idea to apply CG to the normal equations

$$A^TAx = A^Tb, \tag{46}$$

or

$$AA^Ty = b \quad \text{with} \quad x = A^Ty \tag{47}$$

is obvious. The first method is known as the CGNR method, whereas the second method is known as the CGNE method. When A is nonsingular A^TA is symmetric and positive definite. So all the properties and theoretical results for CG can be used. There are however some drawbacks first the rate of convergence now depends on $K_2(A^TA) = K_2(A)^2$. In many applications $K_2(A)^2$ is very large so the convergence of CG applied to (46) is very slow. Another difference is that CG applied to $Ax = b$ depends on the eigenvalues of A whereas CG applied to (46) depends on the eigenvalues of A^TA, which are equal to the singular values of A squared.

Per iteration a multiplication with A and A^T is necessary, so the amount of work is approximately two times as much as for the CG method. Furthermore, in several (FEM, parallel) applications Av is easily obtained but A^Tv not due to the unstructured grid and the data structure used.
Finally not only the convergence depends on $K_2(A)^2$ but also the error due to rounding errors. To improve the numerical stability it is suggested in [4] to replace inner products like p^TA^TAp by $(Ap)^TAp$. Another improvement is the method LSQR proposed by [32]. This method is based on the application of the Lanczos method to the auxiliary system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} .$$

This is a very reliable algorithm. It uses reliable stopping criteria and estimates of standard errors for x and the condition number of A.

4.3.2 Bi-CG type methods

In this type of methods we have short recurrences but no 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 try to find a matrix Q such that $Q^TQ = I$ and

$$Q^TAQ = T \quad \text{tridiagonal} .$$

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 relation can be proved [46]:

$$AR_k = R_k T_k + \alpha_k r_k e_k^T , \tag{48}$$

and

$$S_k^T (Ax_k - b) = 0 .$$

Using (48), $r_0 = b$ and $x_k = R_k y$ we obtain

$$S_k^T R_k T_k y = s_0 r_0^T e_1. \tag{49}$$

Since $S_k^T R_k$ is a diagonal matrix with diagonal elements $r_j^T s_j$, we find, that if all these diagonal elements are nonzero,

$$T_k y = e_1 , \quad x_k = R_k y .$$

We see that 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 [19] §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 [33], and [16]. 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 so far, is thrown away, which destroys possibilities for faster (superlinear) convergence. (The description of the methods given below is based on those given in [46].)

Bi-CG

As has been shown for Conjugate Gradients, the LU decomposition of the tridiagonal system T_k can be updated from iteration to iteration and this leads to a recursive update of the solution vector. This avoids to save all intermediate r and s vectors. This variant of Bi-Lanczos is usually called Bi-Conjugate Gradients, or shortly Bi-CG [14]. 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 then a serious break-down of the Bi-CG algorithm occurs. This break-down can be avoided in the Bi-Lanczos formulation of this iterative solution scheme. The algorithm is given as follows:

Bi-CG
x_0 is given; $r_0 = b - Ax_0$;
$\hat{r}_0$ is an arbitrary vector $(\hat{r}_0, r_0) \neq 0$
possible choice $\hat{r}_0 = r_0$;
$\rho_0 = 1$
$\hat{p}_0 = p_0 = 0$
for $i = 1, 2, ...$
$\rho_i = (\hat{r}_{i-1}, r_{i-1})$; $\beta_i = (\rho_i / \rho_{i-1})$;
$p_i = r_{i-1} + \beta_i p_{i-1}$;

$\hat{p}_i = \hat{r}_{i-1} + \beta_i \hat{p}_{i-1}$;
$v_i = Ap_i$
$\alpha_i = \rho_i/(\hat{p}_i, v_i)$;
$x_i = x_{i-1} + \alpha_i p_i$
$r_i = r_{i-1} - \alpha_i v_i$
$\hat{r}_i = \hat{r}_{i-1} - \alpha_i A^T \hat{p}_i$
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 step.

QMR
The QMR method [17] relates to Bi-CG in a similar way as MINRES relates to CG. For stability reasons the basis vectors r_j and s_j are normalized (as is usual in the underlying Bi-Lanczos algorithm).

If we group the residual vectors r_j, for $j = 0, ..., i-1$ in a matrix R_i, then we can write the recurrence relations as

$$AR_i = R_{i+1}\bar{T}_i \ ,$$

with

$$\bar{T}_i = \begin{pmatrix} \ddots & \ddots & & & \\ \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \\ & & & \ddots & \end{pmatrix} \begin{matrix} \uparrow \\ \\ i+1 \\ \\ \downarrow \end{matrix} \quad \begin{matrix} \longleftarrow & i & \longrightarrow \end{matrix} \ .$$

Similar as for MINRES we would like to construct the x_i, with

$$x_i \in span\ \{r_0, Ar_0, ..., A^{i-1}r_0\} \ , \quad x_i = R_i\bar{y},$$

for which

$$\begin{aligned} \|Ax_i - b\|_2 &= \|AR_i\bar{y} - b\|_2 \\ &= \|R_{i+1}\bar{T}_i y - b\|_2 \\ &= \|R_{i+1}\{\bar{T}_i y - \|r_0\|_2 e_1\}\|_2 \end{aligned}$$

is minimal. However, in this case that would be quite an amount of work since the columns of R_{i+1} are not necessarily orthogonal. In [17] it is suggested to solve the minimum norm least squares problem

$$\min_{y \in I\!R^i} \|barT_i y - \|r_0\|_2 e_1\|_2 \ . \tag{50}$$

This leads to the simplest form of the QMR method. A more general form arises if the least squares problem (50) is replaced by a weighted least squares problem. No strategies are yet

known for optimal weights, however. In [17] the QMR method is carried out on top of a look-ahead variant of the bi-orthogonal Lanczos method, which makes the method more robust. Experiments suggest that QMR has a much smoother convergence behavior than Bi-CG, but it is not essentially faster than Bi-CG. For the algorithm we refer to [3] page 24.

CGS

For the bi-conjugate gradient residual vectors it is well-known that they 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 $P_j(0) = 1$. From the bi-orthogonality relation we have 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 [41] that we 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 (which implicitly define the polynomial P_j). By doing so, it can be avoided that the vectors $\hat{r}_j$ have to be formed, nor is there any multiplication with the matrix A^T. The resulting CGS [41] method works in general very well for many non-symmetric linear problems. It converges often 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 [45].

The following scheme carries out the CGS process for the solution of $Ax = b$, with a given preconditioner K:

Conjugate Gradient Squared method

x_0 is an initial guess; $r_0 = b - Ax_0$;
$\tilde{r}_0$ 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
$u_i = r_i + \beta_{i-1} q_i$;
$p_i = u_i + \beta_{i-1}(q_i + \beta_{i-1} p_{i-1})$;
$\hat{p} = K^{-1} p_i$;
$\hat{v} = A\hat{p}$;
$\alpha_i = \frac{\rho_i}{(\tilde{r}_0, \hat{v})}$;
$q_{i+1} = u_i - \alpha_i \hat{v}$;
$\hat{u} = K^{-1}(u_i + q_{i+1})$
$x_{i+1} = x_i + \alpha_i \hat{u}$;
if x_{i+1} is accurate enough then quit;
$r_{i+1} = r_i - \alpha_i A\hat{u}$;
$\rho_{i+1} = (\tilde{r}_0, r_{i+1})$;
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 constants as those generated by Bi-CG. Therefore, they can be used to compute the Petrov-Galerkin approximations for eigenvalues

of A.

Bi-CGSTAB

Bi-CGSTAB [46] is based on the following observation. Instead of squaring the Bi-CG polynomial, we can construct other iteration methods, by which x_i are generated so that $r_i = \tilde{P}_i(A)P_i(A)r_0$ with other i^{th} degree polynomials $\tilde{P}_i$. 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 $\omega_j \in I\!R$. 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 $Ax = b$, with preconditioning K reads as follows:

Bi-CGSTAB method

x_0 is an initial guess; $r_0 = b - Ax_0$;
$\bar{r}_0$ 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
  $\rho_i = (\bar{r}_0, r_i)$; $\beta_{i-1} = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1})$;
  $p_i = r_i + \beta_{i-1}(p_{i-1} - \omega_{i-1}v_{i-1})$;
  $\hat{p} = K^{-1}p_i$;
  $v_i = A\hat{p}$;
  $\alpha_i = \rho_i/(\bar{r}_0, v_i)$;
  $s = r_i - \alpha_i v_i$;
  if $\|s\|$ small enough then
    $x_{i+1} = x_i + \alpha_i\hat{p}$; quit;
  $z = K^{-1}s$;
  $t = Az$;
  $\omega_i = (t, s)/(t, t)$;
  $x_{i+1} = x_i + \alpha_i\hat{p} + \omega_i z$;
  if x_{i+1} is accurate enough then quit;
  $r_{i+1} = s - \omega_i t$;
end for

The matrix K in this scheme represents the preconditioning matrix and the way of preconditioning [46]. The above scheme in fact carries out the Bi-CGSTAB procedure for the explicitly postconditioned linear system

$$AK^{-1}y = b ,$$

but the vector y_i has been transformed back to the vector x_i corresponding to the original system $Ax = b$. 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 (see Bi-CG).

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 important 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 $\|b - Ax_k\|_2$. If "near" break down had occurred these quantities may be different by several orders of magnitude. In such a case the method should be restarted.

4.3.3 GMRES-type methods

These methods are based on long recurrences, and have certain optimality properties. The long recurrences imply that the amount of work per iteration and required memory grow for 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, GCR and a combination of both GMRESR.

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 [40]:

1. Start: choose x_0 and compute $r_0 = b - Ax_0$ and $v_1 = r_0/\|r_0\|_2$,

2. Iterate: 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 the upper $k + 1 \times k$ Hessenberg matrix $\bar{H}_k$ are the scalars h_{ij}.

In GMRES (General Minimal RESidual method) the approximate solution $x_k = x_0 + z_k$ with $z_k \in K^k(A; r_0)$ is such that

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \min_{z \in K^k(A; r_0)} \|r_0 - Az\|_2 \tag{51}$$

As a consequence of (51) 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 as described in [31]. Using 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 [40] that $x_k = x_0 + V_k y_k$ where y_k is the solution of the following least squares problem:

$$\|\beta e_1 - \bar{H}_k y_k\|_2 = \min_{y \in I\!R^k} \|\beta e_1 - \bar{H}_k y\|_2 \tag{52}$$

with $\beta = \|r_0\|_2$ and e_1 is the first unit vector in $I\!R^{k+1}$. GMRES is a stable method and no break down occurs, if $h_{j+1,j} = 0$ than $x_j = x$ so this is a "lucky" break down (see [40];

Section 3.4).

Due to the optimality (see inequality (51)) convergence proofs are possible [40]. 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 σ represent the spectrum of A.

Theorem 4.1 *Suppose that A is diagonalizable so that $A = XDX^{-1}$, $\sigma = \{\lambda_1, ..., \lambda_n\}$, 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:

$$\|r_m\|_2 \leq K(X)\varepsilon^{(m)}\|r_0\|_2 \tag{53}$$

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

$$\varepsilon^{(m)} \leq \left(\frac{R}{C}\right)^m . \tag{54}$$

Proof: see [40]; p. 866.

For GMRES we see in many cases a super linear convergence behavior comparable to CG. The same type of results are proved for GMRES [48]. 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, form the approximate solution and use 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 [11, 22].

GCR

Slightly earlier than GMRES, [10] proposed the GCR method (Generalized Conjugate Residual method). The algorithm is given as follows:

GCR algorithm
choose x_0, compute $r_0 = b - Ax_0$

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

$s_i := s_i/\|v_i\|_2$, $v_i := v_i/\|v_i\|_2$
$x_i := x_{i-1} + (v_i, r_{i-1})s_i$;
$r_i := r_{i-1} - (v_i, r_{i-1})v_i$;
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 are comparable. However there are examples where GCR breaks down. So comparing full GMRES and full GCR the first one is preferred in many applications.
When the required memory is not available GCR can be restarted. Furthermore, another strategy is possible which is known as truncation. An example of this is to replace the j-loop by

for $j = i - m, ..., i - 1$ do

Now $2m$ vectors are needed in memory. Other truncation variants to discard search direction are possible. In general we see that truncated methods have a better convergence behavior especially if super linear convergence plays an important role. So if restarting or truncation is necessary truncated GCR is in general better than restarted GMRES. For convergence results and other properties we refer to [10].

GMRESR
A number of methods are proposed to diminish the disadvantages of restarting and or truncation. One of these methods is GMRESR proposed in [49] and further investigated in [52]. This method consists of an outer- and an inner loop. In the inner loop we approximate the solution of a linear system with 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 x_0 and m, compute $r_0 = b - Ax_0$

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
$s_i := s_i/\|v_i\|_2$, $v_i := v_i/\|v_i\|_2$
$x_i := x_{i-1} + (v_i, r_{i-1})s_i$;
$r_i := r_{i-1} - (v_i, r_{i-1})v_i$;
end for

The notation $s_i = P_{m,i-1}(A)r_{i-1}$ denotes 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 get GCR, whereas for $m \to \infty$ one outer iteration is sufficient and we get GMRES. For the amount of work we refer to [49], where also optimal choices of m are 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 have tried to visualize the strong point of GMRESR in comparison with GMRES(m) and truncated GCR. A search directions is indicated by the symbol v_i. We see for GMRES(3) that after 3 iterations all information is thrown away. For GCR(3)

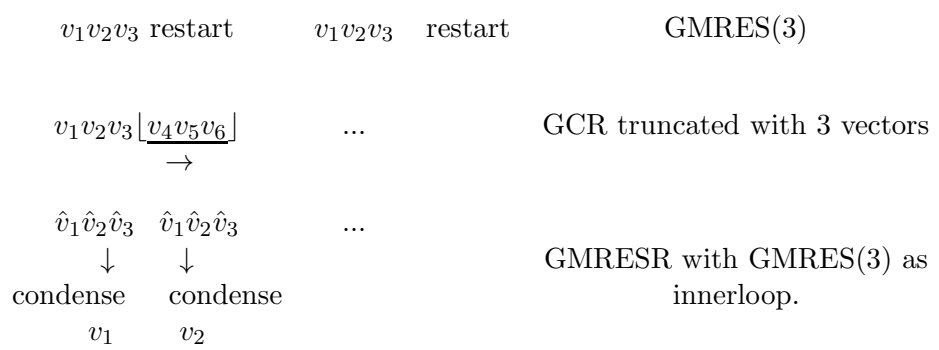

Figure 9: The use of search directions for restarted GMRES, truncated GCR and full GMRESR.

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 "no" information gets lost.
Also for GMRESR restarts and truncation are possible [52]. In the inner loop other iterative methods can be used. Several of these choices lead to a good iterative method. In theory we can call the same loop again, which motivates the name GMRES Recursive. A comparable approach is the FGMRES method give in [37]. Herein the outer loop consists of a slightly adapted GMRES algorithm. Since FGMRES and GMRESR are comparable in work and memory but FGMRES can not be truncated we prefer the GMRESR method.

4.3.4 Choice of an iterative method

For non-symmetric matrices it is very 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 appear, GMRES like methods are better. Finally LSQR always converges but can take a large number of iterations.

In [52] we have tried to specify some easy to obtain parameters to facilitate a choice. First one should have an (crude) idea of the total number of iterations (mg) using full GMRES, secondly one should measure the ratio f which is defined as

$$f = \frac{\text{the CPU time used for one preconditioned matrix vector product}}{\text{the CPU time used for a vector update}}$$

Note that f depends on the used computer. Under certain assumptions given in [52] we obtain Figure 10. 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 [3] a flowchart is given with suggestions for the selection of a suitable iterative method.

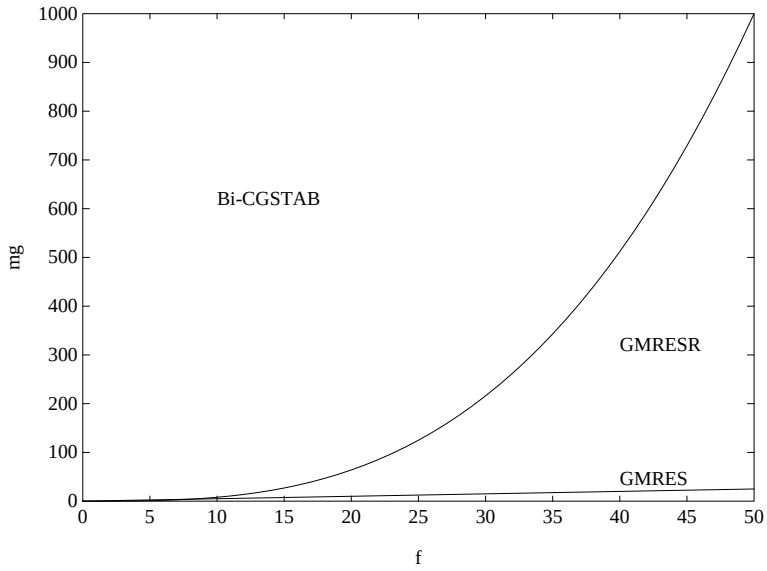

Figure 10: Regions of feasibility of Bi-CGSTAB, GMRES, and GMRESR.

4.3.5 Iterative methods for complex matrices

There are in practice, important applications that lead to linear systems where the coefficient matrix has complex valued entries. Examples are: complex Helmholtz equations, Schrödinger's equation, under water acoustics etc [12]. If the resulting system is Hermitian the methods of Chapter 2 can be used. In these algorithms the inner product $x^T y$ should be replaced by the complex inner product $\bar{x}^T y$. For non Hermitian matrices iterative methods as given in Sections 4.3.1 to 4.3.3 can be used. Again they should be adapted to use the correct inner product.
In many applications the resulting complex linear systems have additional structure that can be exploited. For instance matrices of the following form arise:

$$A = e^{i\Theta}(T + \sigma I) \quad \text{where} \quad T = \bar{T}^T \,, \quad \Theta \in I\!R \,, \quad \sigma \in \mathbb{C}$$

Another special case that arises frequently in applications are complex symmetric matrices

$$A = A^T$$

For example, the complex Helmholtz equations leads to complex symmetric systems. For methods to solve these systems we refer to [16]; section 2.2, 2.3, 6.

4.4 Exercises

1. Show that the solution $\begin{pmatrix} y \\ x \end{pmatrix}$ of the augmented system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

is such that x satisfies $A^T Ax = A^T b$.

2. Take the following matrix

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

   (a) Suppose that GCR is applied to the system $Ax = b$. Show that GCR converges in 1 iteration if $x - x_0 = cr_0$, where $c \neq 0$ is a scalar and $r_0 = b - Ax_0$.

   (b) Apply GCR for the choices $b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

   (c) Do the same for $x_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

3. In the GCR algorithm the vector r_i is obtained from vectorupdates. Show that the relation $r_i = b - Ax_i$ is valid.

4. Prove the following properties for the GMRES method:

   - $AV_k = V_{k+1}\bar{H}_k$,
   - $x_k = x_0 + V_k y_k$, where y_k is obtained from (52).

5. Figure 10 can give an indication which solution method should be used. Give an 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?

6. **A practical exercise**
   For the methods mentioned below we use as test matrices:

$$[a, f] = poisson(5, 5, 100, 0, 'central')$$

   and

$$[a, f] = poisson(5, 5, 100, 0, 'upwind')$$

   (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.

References

[1] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, Cambridge, UK, 1994.

[2] O. Axelsson and G. Lindskog. On the eigenvalue distribution of a class of preconditioning methods. *Numer. Math.*, 48:479–498, 1986.

[3] 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*. SIAM, Philadelphia, 1994.

[4] A. Björck and T. Elfving. Accelerated projection methods for computing pseudo-inverse solution of systems of linear equations. *BIT*, 19:145–163, 1979.

[5] E.K. Blum. *Numerical Analysis and Computation, Theory and Practice*. Addison-Wesley, Reading, 1972.

[6] A.M. Bruaset. *A Survey of Preconditioned Iterative Methods*. Pitman research notes in mathematics series 328. Longman Scientific and Technical, Harlow, 1995.

[7] B.A. Carré. The determination of the optimum accelerating factor for successive over-relaxation. *Computer Journal*, 4:73–78, 1961.

[8] 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.

[9] S.C. Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM J. Sci. Stat. Comput.*, 2:1–4, 1981.

[10] S.C. Eisenstat, H.C. Elman, and M.H. Schultz. Variable iterative methods for nonsymmetric systems of linear equations. *SIAM J. Num. Anal.*, 20:345–357, 1983.

[11] M. Embree. The Tortoise and the Hare restart GMRES. *SAIM Review*, 45:259–266, 2003.

[12] Y.A. Erlangga, C.W. Oosterlee, and C. Vuik. A novel multigrid based preconditioner for heterogeneous Helmholtz problems. *SIAM J. Sci. Comput.*, 27:1471–1492, 2006.

[13] 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.

[14] R. Fletcher. Factorizing symmetric indefinite matrices. *Lin. Alg. and its Appl.*, 14:257–277, 1976.

[15] 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, UK, 1992.

[16] R.W. Freund, M.H. Gutknecht, and N.M. Nachtigal. An implimentation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM J. Sci. Comp.*, 14:137–156, 1993.

[17] R.W. Freund and N.M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[18] 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.

[19] G.H. Golub and C.F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1996. Third edition.

[20] 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.

[21] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. Frontiers in applied mathmatics 17. SIAM, Philadelphia, 1997.

[22] A. Greenbaum, V. Ptak, and Z. Strakos. Any nonincreasing convergence curve is possible for GMRES. *SIAM J. Matrix Anal. Appl.*, 17:465–469, 1996.

[23] I.A. Gustafsson. A class of first order factorization methods. *BIT*, 18:142–156, 1978.

[24] L.A. Hageman and D.M. Young. *Applied Iterative Methods*. Academic Press, New York, 1981.

[25] M.R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *J. Res. Nat. Bur. Stand.*, 49:409–436, 1952.

[26] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambrige, 1985.

[27] E.F. Kaasschieter. Preconditioned conjugate gradients for solving singular systems. *J. of Comp. Appl. Math.*, 24:265–275, 1988.

[28] T.A. Manteuffel. The Tchebychev iteration for nonsymmetric linear systems. *Numer. Math.*, 28:307–327, 1977.

[29] 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.

[30] 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.

[31] C.C. Paige and M.A. Saunders. Solution of sparse indefinite system of linear equations. *SIAM J. Num. Anal.*, 12:617–629, 1975.

[32] C.C. Paige and M.A. Saunders. LSQR: an algorithm for sparse linear equations and sparse least square problem. *ACM Trans. Math. Softw.*, 8:44–71, 1982.

[33] B.N. Parlett, D.R. Taylor, and Z.A. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comp.*, 44:105–124, 1985.

[34] T. F. Chan J. Demmel J. Donato J. Dongarra V. Eijkhout R. Pozo C. Romine R. Barrett, M. Berry and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, 1994.

[35] J.K. Reid. The use of conjugate for systems of linear equations posessing property A. *SIAM J. Num. Anal.*, 9:325–332, 1972.

[36] Y. Saad. Preconditioning techniques for non symmetric and indefinite linear system. *J. Comp. Appl. Math.*, 24:89–105, 1988.

[37] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Stat. Comput.*, 14:461–469, 1993.

[38] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing, Boston, 1996.

[39] Y. Saad. *Iterative methods for sparse linear systems, Second Edition*. SIAM, Philadelphia, 2003.

[40] Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 7:856–869, 1986.

[41] P. Sonneveld. CGS: a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989.

[42] C. Oosterlee U. Trottenberg and A. Schüller. *Multigrid*. Academic Press, San Diego, 2001.

[43] A. van der Sluis. Conditioning, equilibration, and pivoting in linear algebraic systems. *Numer. Math.*, 15:74–86, 1970.

[44] A. van der Sluis and H.A. van der Vorst. The rate of convergence of conjugate gradients. *Numer. Math.*, 48:543–560, 1986.

[45] H.A. van der Vorst. High performance preconditioning. *SIAM J. Sci. Stat. Comp.*, 10:1174–1185, 1989.

[46] 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.

[47] H.A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge Monographs on Applied and Computational Mathematics, 13. Cambridge University Press, Cambridge, 2003.

[48] H.A. van der Vorst and C. Vuik. The superlinear convergence behaviour of GMRES. *J. Comput. Appl. Math.*, 48:327–341, 1993.

[49] H.A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Num. Lin. Alg. Appl.*, 1:369–386, 1994.

[50] R. S. Varga. *Matrix Iterative Analysis*. Springer, Berlin, 2000.

[51] R.S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, N.J., 1962.

[52] C. Vuik. Solution of the discretized incompressible Navier-Stokes equations with the GMRES method. *Int. J. Num. Meth. in Fluids*, 16:507–523, 1993.

[53] E.L. Wachspress. *Iterative Solution of Elliptic Systems*. Prentice-Hall, Englewood Cliffs, 1966.

[54] D.M. Young. *Iterative Solution of Large Linear Systems*. Academic Press, New York, 1971.

Index