Contents

Chapter 10

Iterative Methods for Linear Systems

10.1 Classical Iterative Methods

10.1.1 Introduction

The methods discussed so far for solving systems of linear equations $Ax = b$, have been direct methods based on matrix factorization. Disregarding rounding errors, direct methods yield the exact solutionin in a fixed finite number of operations. Iterative methods, on the other hand, start from an initial approximation, which is successively improved until a sufficiently accurate solution is obtained. The idea of solving systems of linear equations by iterative methods dates at least back to Gauss (1823). Before the advent of computers iterative methods used were usually noncyclic relaxation methods guided at each step by the sizes of the residuals of the current solution. When in the 1950s computers replaced desk calculators an intense development of cyclic iterative methods started.

Basic iterative methods work directly with the original matrix A and only need extra storage for a few vectors. Since A is involved only in terms of matrix-vector products there is usually no need even to store the matrix A. Such methods are particularly useful for sparse system, which typically arise in the solution of boundary value problems of partial differential equations by finite difference or finite element methods. The matrices involved can be huge, sometimes involving several million unknowns. The LU factors of matrices in such applications typically contain order of magnitudes more nonzero elements than A itself. Hence, because of the storage and number of arithmetic operations required, direct methods may become far too costly to use. This is true in particular for problems arising from three-dimensional simulations in e.g., reservoir modeling, mechanical engineering, electric circuit simulation. However, in some areas, e.g., structural engineering, which typically yield very ill-conditioned matrices, direct methods are still preferred.

In using an iterative methods for solving a linear system of equations $Ax = b$, where A is square and nonsingular it is assumed that an initial approximation $x^{(0)}$ is given (e.g., $x^{(0)} = 0$). A sequence of approximations $x^{(1)}$, $x^{(2)}, \ldots$, which

converges to the solution is then computed. The methods used before the age of high speed computers were usually rather unsophisticated **relaxation methods**. In **Richardson's method**,[1] the next approximation is computed as

$$x^{(k+1)} = x^{(k)} + \omega_k(b - Ax^{(k)}), \quad k = 0, 1, 2, \ldots, \tag{10.1.1}$$

where $\omega_k > 0$ are parameters to be chosen. It follows easily from (10.1.1) that the residual $r^{(k)} = b - Ax^{(k)}$ and error satisfy the recursions

$$r^{(k+1)} = (I - \omega_k A)r^{(k)}, \quad x^{(k+1)} - x = (I - \omega_k A)(x^{(k)} - x).$$

In the special case that $\omega = \omega_k$ we have

$$x^{(k)} - x = (I - \omega A)^k(x^{(0)} - x).$$

If A has a nonzero diagonal it can be scaled to have all diagonal elements equal to 1. In this case Richardson's method with $\omega = 1$ is equivalent to Jacobi's method. The convergence of the Richardson's method will be considered in Section 10.1.3.

10.1.2 A Model Problem

The biggest source of large linear systems is partial differential equations. An equation which is often encountered is Poisson's equation, where $u(x, y)$ satisfies

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f, \qquad (x, y) \in \Omega = (0, 1) \times (0, 1).$$

On the boundary Ω we assume $u(x, y)$ to be prescribed. We will frequently use this as a model problem.

To approximate the solution we impose a uniform square mesh of side $h = 1/n$ on Ω. Taking $f = 0$ (Laplace equation) and approximating the second derivatives by symmetric difference quotients gives a difference equation

$$\frac{1}{h^2}\big(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij}\big) = 0, \qquad 0 < i, j < n,$$

for the unknown values u_{ij} at interior mesh points; see Section 13.1.2. If the mesh points are enumerated line by line (the so called "natural ordering") and a vector u is formed of the function values, the difference equation can then be written in matrix form as

$$Au = h^2 b, \quad u = (u_1, u_2, \ldots, u_{n-1}),$$

where u_i is the vector of unknowns in the ith line and the matrix A is symmetric.

The linear system arising from Poisson's equation has several features common to boundary value problems for all linear partial differential equations. One of these is that there are at most 5 nonzero elements in each row of A, i.e. only a tiny fraction

[1] Lewis Fry Richardson (1881–1953) English mathematician, who was the first to use mathematical methods for weather prediction.

of the elements are nonzero. Such matrices are called **sparse**. Therefore a matrix-vector multiplication Ax requires only about $5 \cdot N^2$ multiplications or equivalently five multiplications per unknown. Using iterative methods which take advantage of the sparsity and other features does allow the efficient solution of such systems. This becomes even more essential for three-dimensional problems!

It can be verified that in block form the matrix can be written as

$$A = \begin{pmatrix} 2I+T & -I & & \\ -I & 2I+T & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & 2I+T \end{pmatrix} \in \mathbf{R}^{(n-1)^2 \times (n-1)^2}, \tag{10.1.2}$$

where T is symmetric tridiagonal,

$$T = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \in \mathbf{R}^{(n-1)\times(n-1)}. \tag{10.1.3}$$

Taking $n = 3$ we obtain the matrix used in Example **??**.

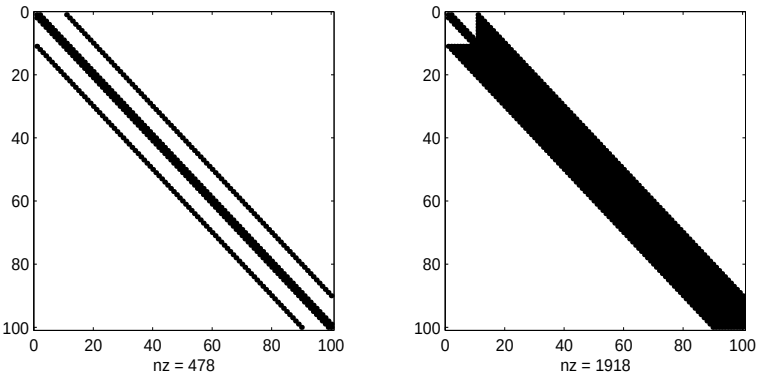

Figure 10.1.1. *Structure of A (left) and $L + U$ (right) for the Poisson problem, $N = 10$. (Row-wise ordering of the unknowns)*

In principle Gaussian elimination can be used to solve such systems. However, even taking symmetry and the banded structure into account this would require $\frac{1}{2} \cdot N^4$ multiplications, since in the LU factors the zero elements inside the outer diagonals will fill-in during the elimination Hence L contains about n^3 nonzero elements compared to only about $5n^2$ in A as shown in Figure 10.1.1 (right). To compute the Cholesky factorization of a symmetric band matrix of order n and (half) bandwidth w requires approximately $\frac{1}{2}nw^2$ flops (see Algorithm 6.4.6). For the matrix A in (10.1.3) the dimension is n^2 and the bandwidth equals n. Hence

about $\frac{1}{2}n^4$ flops are needed for the factorization. This can be compared to the $5n^2$ flops needed per iteration, e.g., with Jacobi's method.

The above shows that for the model problem direct methods use $O(n^2)$ flops and about $O(n)$ storage per grid point. This disadvantage becomes even more accentuated if a three dimensional problem is considered. For Laplace equation in the unit cube a similar study shows that for solving n^3 unknown we need $\frac{1}{2}n^7$ flops and about n^5 storage. When n growth this quickly becomes infeasible. However, basic iterative methods still require only about $7n^3$ flops per iteration.

We still have not discussed the number of iterations needed to get acceptable accuracy. It turns out that this will depend on the condition number of the matrix. We now show that for the Laplace equation considered above this condition number will be about πh^{-2}, independent of the dimension of the problem.

Lemma 10.1.1. *Let $T = \operatorname{trid}(c, a, b) \in \mathbf{R}^{n\times n}$ be a tridiagonal matrix with constant diagonals, and assume that a, b, c are real and $bc > 0$. Then the n eigenvalues of T are given by*

$$\lambda_i = a + 2\sqrt{bc}\cos\frac{i\pi}{n+1}, \quad i = 1, \ldots, n.$$

Further, the jth component of the eigenvector v_i corresponding to λ_i is

$$v_{ij} = \left(\frac{b}{c}\right)^{j/2} \sin\frac{ij\pi}{n+1}, \quad j = 1, \ldots, n.$$

From Lemma 10.1.1) it follows that the eigenvalues of T are $\lambda_i = 2(1 + \cos(i\pi/n))$, $i = 1, \ldots, n-1$. In particular we have that

$$\lambda_{\max} = 2(1 + \cos(\pi/n)) \approx 4, \qquad \lambda_{\min} = 2(1 - \cos(\pi/n)) \approx (\pi/n)^2.$$

Hence the condition number of T is approximately equal to $4n^2/\pi^2$.

The matrix $A = 4(I - L - U)$ in (10.1.2) can be written in terms of the Kronecker product (see Section 6.2.3) as

$$A = (I \otimes T) + (T \otimes I),$$

i.e., A is the Kronecker sum of T and T. It follows that the $(n-1)^2$ eigenvalues of A are $(\lambda_i + \lambda_j)$, $i, j = 1, \ldots, n-1$, and hence the condition number of A is the same as for T. The same conclusion can be shown to hold for a three dimensional problem.

10.1.3 Stationary Iterative Methods

Assume that A has nonzero diagonal entries, i.e, $a_{ii} \neq 0$, $i = 1, 2, \ldots, n$. If A is symmetric, positive definite this is necessarily the case. Otherwise, since A is non-singular, the equations can always be reordered so that this is true. In component form the system can then be written

$$x_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\Big), \qquad i = 1, 2 \ldots, n. \tag{10.1.4}$$

In a (minor) step of the iteration we pick one equation, say the ith, and then adjust the ith component of $x^{(k)}$ so that this equation becomes exactly satisfied. Hence, given $x^{(k)}$ we compute

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} r_i^{(k)}, \quad r_i^{(k)} = b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k)}. \tag{10.1.5}$$

In the days of "hand" computation one picked an equation with a large residual $|r_i$ and went through the equations in an irregular manner. This is less efficient when using a computer, and here one usually perform these adjustments for $i = 1, 2 \ldots, n$, in a cyclic fashion. The resulting iterative method is called the method of simultaneous displacements or **Jacobi's method**. Note that all components of x can be updated *simultaneously* and the result does not depend on the sequencing of the equations.

The method of successive displacements or **Gauss–Seidel's method**[2] differs from the Jacobi method by using new values $x_j^{(k+1)}$ as soon as they are available as follows:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} r_i^{(k)}, \quad r_i^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)}, \quad i = 1, 2 \ldots, n. \tag{10.1.6}$$

Here the components are *successively* updated and the sequencing of the equations will influence the result.

Since each new value $x_i^{(k+1)}$ can immediately replace $x_i^{(k)}$ in storage the Gauss–Seidel method storage for unknowns is halved compared to Jacobi's method. For both methods the amount of work required in each iteration step is comparable in complexity to the multiplication of A with a vector, i.e., proportional to the number of nonzero elements in A. By construction it follows that if $\lim_{k\to\infty} x^{(k)} = x$, then x satisfies (10.1.4) and therefore the system $Ax = b$.

The Jacobi, Gauss–Seidel, and Richardson methods are all special cases of a class of iterative methods, the general form of which is

$$Mx^{(k+1)} = Nx^{(k)} + b, \qquad k = 0, 1, \ldots. \tag{10.1.7}$$

Here

$$A = M - N \tag{10.1.8}$$

is a **splitting** of the matrix coefficient matrix A with M nonsingular. If the iteration (10.1.7) converges, i.e., $\lim_{k\to\infty} x^{(k)} = x$, then $Mx = Nx + b$ and it follows from (10.1.8) that the limit vector x solves the linear system $Ax = b$. For the iteration to be practical, it must be easy to solve linear systems with matrix M. This is the case, for example, if M is chosen to be triangular.

The iteration (10.1.7) is equivalent to

$$x^{(k+1)} = Bx^{(k)} + c, \qquad k = 0, 1, \ldots, \tag{10.1.9}$$

where

[2] It was noted by G. Forsythe that Gauss nowhere mentioned this method and Seidel never advocated using it!

$$B = M^{-1}N = I - M^{-1}A, \qquad c = M^{-1}b.$$

An iteration of the form of (10.1.3) is called a (one-step) **stationary iterative method**, and B the **iteration matrix**. (In a non-stationary method the iteration matrix B depends on k.) Subtracting the equation $x = Bx + c$ from (10.1.3) we obtain the recurrence formula

$$x^{(k+1)} - x = B(x^{(k)} - x) \tag{10.1.10}$$

for the errors in successive approximations.

Richardson's method (10.1.1) can, for fixed $\omega_k = \omega$, be written in the form (10.1.7) with

$$M = I, \qquad N = B = I - \omega A.$$

To write the Jacobi and Gauss–Seidel methods in the form of one-step stationary iterative methods we introduce the **standard splitting**

$$A = D_A - L_A - U_A, \tag{10.1.11}$$

where $D_A = \operatorname{diag}(a_{11}, \ldots, a_{nn})$,

$$L_A = -\begin{pmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \quad U_A = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & \ddots & \ddots & \vdots \\ & & 0 & a_{n-1,n} \\ & & & 0 \end{pmatrix},$$

and L_A and U_A are strictly lower and upper triangular, respectively. Assuming that $D_A > 0$, we can also write

$$D_A^{-1}A = I - L - U, \qquad L = D_A^{-1}L_A, \qquad U = D_A^{-1}U_A.$$

With these notations the Jacobi method, (10.1.5), can be written $D_A x^{(k+1)} = (L_A + U_A)x^{(k)} + b$ or

$$x^{(k+1)} = (L + U)x^{(k)} + c, \qquad c = D_A^{-1}b. \tag{10.1.12}$$

The Gauss–Seidel method, (10.1.6), becomes $(D_A - L_A)x^{(k+1)} = U_A x^{(k)} + b$, or equivalently

$$x^{(k+1)} = (I - L)^{-1}Ux^{(k)} + c, \quad c = (I - L)^{-1}D_A^{-1}b$$

Hence these methods are special cases of one-step stationary iterative methods, and correspond to the matrix splittings

$$\begin{array}{lll} \textit{Jacobi:} & M = D_A, & N = L_A + U_A, \\ \textit{Gauss–Seidel:} & M = D_A - L_A, & N = U_A, \end{array}$$

The iteration matrices for the Jacobi and Gauss–Seidel methods are

$$B_J = D_A^{-1}(L_A + U_A) = L + U,$$
$$B_{GS} = (D_A - L_A)^{-1}U_A = (I - L)^{-1}U.$$

10.1.4 Convergence Analysis

The iterative method (10.1.3) is called **convergent** if the sequence $\{x^{(k)}\}_{k=1,2,\ldots}$ converges for *all* initial vectors $x^{(0)}$. Of fundamental importance in the study of convergence of stationary iterative methods is conditions for a sequence of powers of a matrix to converge to the null matrix. For this we need some results from the theory of eigenvalues of matrices.

In Section 6.2.2 we introduced the spectral radius of a matrix A as the non-negative number

$$\rho(A) = \max_{1\le i\le n} |\lambda_i(A)|.$$

We have the following important result:

Theorem 10.1.2. *A given matrix $B \in \mathbf{R}^{n\times n}$ is said to be* **convergent** *if $\rho(B) < 1$. It holds that*

$$\lim_{k\to\infty} B^k = 0 \quad \Leftrightarrow \quad \rho(B) < 1. \tag{10.1.13}$$

Proof. We will show that the following four conditions are equivalent:

(i) $\lim_{k\to\infty} B^k = 0$,

(ii) $\lim_{k\to\infty} B^k x = 0, \quad \forall x \in \mathbf{C}^n$,

(iii) $\rho(B) < 1$,

(iv) $\|B\| < 1$ for at least one matrix norm.

For any vector x we have the inequality $\|B^k x\| \le \|B^k\|\,\|x\|$, which shows that (i) implies (ii).
If $\rho(B) \ge 1$, then there is an eigenvector $x \in \mathbf{C}^n$ such that $Bx = \lambda x$, with $|\lambda| \ge 1$. Then the sequence $B^k x = \lambda^k x$, $k = 1, 2, \ldots$, is not convergent when $k \to \infty$ and hence (ii) implies (iii).
By Theorem 10.2.9, (see Section 10.2.4) given a number $\epsilon > 0$ there exists a consistent matrix norm $\|\cdot\|$, depending on B and ϵ, such that

$$\|B\| < \rho(B) + \epsilon.$$

Therefore (iv) follows from (iii).
Finally, by applying the inequality $\|B^k\| \le \|B\|^k$, we see that (iv) implies (i). ∎

The following necessary and sufficient criterion for convergence of a stationary iterative method follows from Theorem 10.1.2.

Theorem 10.1.3. *The stationary iterative method $x^{(k+1)} = Bx^{(k)} + c$ is convergent for all initial vectors $x^{(0)}$ if and only if $\rho(B) < 1$, where $\rho(B)$ is the spectral radius of B.*

Proof. From the recurrence (10.1.10) it follows that

$$x^{(k)} - x = B^k(x^{(0)} - x). \tag{10.1.14}$$

Hence $x^{(k)}$ converges for *all* initial vectors $x^{(0)}$ if and only if $\lim_{k\to\infty} B^k = 0$. The theorem now follows from Theorem 10.1.2. □

The problem of obtaining the spectral radius of B is usually no less difficult than solving the linear system. Hence the following upper bound is useful when trying to prove convergence.

Lemma 10.1.4.
For any matrix $A \in \mathbf{R}^{n\times n}$ and for any consistent matrix norm we have

$$\rho(A) \le \|A\|.$$

Proof. Let λ be an eigenvalue of A such that $|\lambda| = \rho(A)$. Then $Ax = \lambda x$, $x \neq 0$, and taking norms

$$\|\lambda x\| = \rho(A)\|x\| = \|Ax\| \le \|A\|\,\|x\|.$$

Since $\|x\| > 0$, we can divide the inequality by $\|x\|$ and the theorem follows. □

From Lemma 10.1.4 it follows that a *sufficient* condition for convergence of the iterative method is that $\|B\| < 1$, for *some* matrix norm.

Usually, we are not only interested in convergence, but also in the **rate of convergence**. By (10.1.14) the error at step k, $e^{(k)} = x^{(k)} - x$, satisfies $e^{(k)} = B^k e^{(0)}$, we have for any consistent pair of norms

$$\|e^{(k)}\| \le \|B^k\|\,\|e^{(0)}\| \le \|B\|^k\|\,\|e^{(0)}\|.$$

Thus, to reduce the norm of the error by a given factor $\delta < 1$, it suffices to perform k iterations, where k is the smallest integer for which $\|B^k\| \le \delta$. Taking logarithms we obtain the condition

$$k \ge -\log\delta / R_k(B), \qquad R_k(B) = -\frac{1}{k}\log\|B^k\|.$$

An expression for the asymptotic rate follows from the relation

$$\rho(B) = \lim_{k\to\infty} (\|B^k\|)^{1/k},$$

which holds for any consistent matrix norm. This is a non-trivial result, but can be proved by using the Jordan normal form, see Problem 10.2.4.

This motivates the following definition:

Definition 10.1.5. *Assume that the iterative method* (10.1.3) *is convergent. For any matrix norm $\|\cdot\|$ we define the* **average rate** *of convergence by*

$$R_k(B) = -\frac{1}{k}\log\|B^k\|, \tag{10.1.15}$$

The corresponding **asymptotic rate** *of convergence is given by*

$$R_\infty(B) = \lim_{k\to\infty} R_k(B) = -\log \rho(B).$$

We now consider the convergence of some classical methods. We have seen that For fixed $\omega_k = \omega$ Richardson's method is a stationary iterative method with iteration matrix $B = I - \omega A \in \mathbf{R}^{n\times n}$.

Theorem 10.1.6.
Assume that the eigenvalues λ_i of A are all real and satisfy

$$0 < a \le \lambda_i \le b, \quad i = 1, \dots, n.$$

Then Richardson's method is convergent for $0 < \omega < 2/b$.

Proof. The eigenvalues of $B = I - \omega A$ are $\mu_i = 1 - \lambda_i$ and thus satisfy $1 - \omega b \le \mu_i \le 1 - \omega a$. If $1 - \omega a < 1$ and $1 - \omega b > -1$, then $|\mu_i| < 1$ for all i and the method is convergent. Since $a > 0$ the first condition is satisfied for all $\omega > 0$, while the second is true if $\omega < 2/b$. □

Assuming that $a = \lambda_{min}$ and $b = \lambda_{max}$, which value of ω will minimize the spectral radius

$$\rho(B) = \max\{|1 - \omega a|, |1 - \omega b|\}\,?$$

It is left as an exercise to show that the optimal ω satisfies $1 - \omega a = \omega b - 1$. (*Hint*: Plot the graphs of $|1 - \omega a|$ and $|1 - \omega b|$ for $\omega \in (0, 2/b)$.) Hence $\omega_{\text{opt}} = 2/(b + a)$, for which

$$\rho(B) = \frac{b-a}{b+a} = \frac{\kappa - 1}{\kappa + 1} = 1 - \frac{2}{\kappa + 1},$$

where $\kappa = b/a$ is the condition number of A. Note that for large values of κ the rate of convergence with ω_{opt} is proportional to κ^{-1}. This illustrates a typical fact for iterative methods: ill-conditioned systems require in general more work to achieve a certain accuracy!

Theorem 10.1.7. *The Jacobi method is convergent if A is strictly row-wise diagonally dominant, i.e.,*

$$|a_{ii}| > \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|, \quad i = 1, 2, \dots, n.$$

Proof. For the Jacobi method the iteration matrix $B_J = L + U$ has elements $b_{ij} = -a_{ij}/a_{ii}$, $i \neq j$, $b_{ij} = 0$, $i = j$. From the assumption it then follows that

$$\|B_J\|_\infty = \max_{1\le i\le n} \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|/|a_{ii}| < 1.$$

□

A similar result for strictly column-wise diagonally dominant matrices can be proved using $\|B_J\|_1$. A slightly stronger convergence result than in Theorem 10.1.7 is of importance in applications. (Note that, e.g., the matrix A in (10.1.4) is not strictly diagonal dominant!) For irreducible matrices (see Def. 10.1.1) the row sum criterion in Theorem 10.1.7 can be sharpened.

Theorem 10.1.8. *The Jacobi method is convergent if A is irreducible,*

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \quad i = 1, 2, \ldots, n,$$

and inequality holds for at least one row.

The column sum criterion can be similarly improved. The conditions in Theorem 10.1.7–10.1.8 are also sufficient for convergence of the Gauss–Seidel method for which $(I-L)B_{GS} = U$. Consider the strictly row-wise diagonally dominant and choose k so that

$$\|B_{GS}\|_\infty = \|B_{GS}^T\|_1 = \|B_{GS}^T e_k\|_1.$$

Then from $B_{GS}^T e_k = B_{GS}^T L^T e_k + U^T e_k$, we get

$$\|B_{GS}\|_\infty \leq \|B_{GS}\|_\infty \|L^T e_k\|_1 + \|U^T e_k\|_1.$$

Since A is strictly row-wise diagonally dominant we have $\|L^T e_k\|_1 + \|U^T e_k\|_1 \leq \|B_J\|_\infty < 1$, and it follows that

$$\|B_{GS}\|_\infty \leq \|U^T e_k\|_1 / \big(1 - \|L^T e_k\|_1\big) < 1.$$

Hence the Gauss–Seidel method is convergent. The proof for the strictly column-wise diagonally dominant case is similar but estimates $\|B_{GS}\|_1$.

Example 10.1.1. In Section 10.1.2 it was shown that the $(n-1)^2$ eigenvalues of the matrix $A = (I \otimes T) + (T \otimes I)$ arising from the model problem are $(\lambda_i + \lambda_j)$, $i, j = 1, \ldots, n-1$, where $\lambda_i = 2(1 + \cos(i\pi/n))$. It follows that the eigenvalues of the corresponding Jacobi iteration matrix $B_J = L + U = (1/4)(A - 4I)$ are

$$\mu_{ij} = \frac{1}{2}(\cos i\pi h + \cos j\pi h), \quad i, j = 1, 2, \ldots, n-1,$$

where $h = 1/n$ is the grid size. The spectral radius is obtained for $i = j = 1$,

$$\rho(B_J) = \cos(\pi h) \approx 1 - \tfrac{1}{2}(\pi h)^2.$$

This means that the low frequency modes of the error are damped most slowly, whereas the high frequency modes are damped much more quickly.[3] The same is true for the Gauss–Seidel method, for which

$$\rho(B_{GS}) = \cos^2(\pi h) \approx 1 - (\pi h)^2,$$

[3] This is one of the basic observations used in the multigrid method, which uses a sequence of different meshes to efficiently damp all frequencies.

The corresponding asymptotic rates of convergence are $R_\infty(B_J) \approx \pi^2 h^2/2$, and $R_\infty(B_{GS}) \approx \pi^2 h^2$. This explains the observation made in Example ex10.1.1 that Gauss–Seidel's method converged twice as fast as Jacobi's method. However, for both methods the number of iterations is proportional to $\kappa(A)$ for the model problem.

The rate of convergence of the Jacobi and Gauss–Seidel methods, as exhibited in the above example, is in general much too slow to make these methods of any practical use. In Section 10.2.1 we show how with a simple modification of the Gauss–Seidel method the rate of convergence can be improved by a factor of n for the model problem.

10.1.5 The Effect of Nonnormality and Finite Precision

While the spectral radius determines the *asymptotic* rate of growth of matrix powers, the norm will influence the *initial* behavior of the powers B^k. However, the norm of a convergent matrix can for a nonnormal matrix be arbitrarily large. By the Schur normal form any matrix A is unitarily equivalent to an upper triangular matrix. Therefore, in exact arithmetic, it suffices to consider the case of an upper triangular matrix.

Consider the 2×2 convergent matrix

$$B = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}, \quad 0 < \mu \le \lambda < 1, \quad \alpha \gg 1, \tag{10.1.16}$$

for which we have $\|B\|_2 \gg \rho(B)$. Therefore, even though $\|B^k\| \to 0$ as $k \to \infty$, the spectral norms $\|B^k\|_2$ will initially sharply increase! It is easily verified that

$$B^k = \begin{pmatrix} \lambda^k & \beta_k \\ 0 & \mu^k \end{pmatrix}, \quad \beta_k = \begin{cases} \alpha \dfrac{\lambda^k - \mu^k}{\lambda - \mu} & \text{if } \mu \neq \lambda; \\ \alpha k \lambda^{k-1} & \text{if } \mu = \lambda. \end{cases} \tag{10.1.17}$$

Clearly the element β_k will grow initially. In the case that $\lambda = \mu$ the maximum of $|\beta_k|$ will occur when $k \approx \lambda/(1-\lambda)$. (See also Computer Exercise 1.)

For matrices of larger dimension the initial increase of $\|B^k\|$ can be huge as shown by the following example:

Example 10.1.2. Consider the iteration $x^{(k+1)} = Bx^{(k)}$, where $B \in \mathbf{R}^{20\times 20}$ is the bidiagonal matrix

$$B = \begin{pmatrix} 0.5 & 1 & & & & \\ & 0.5 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & 0.5 & 1 & \\ & & & & 0.5 & \end{pmatrix}, \qquad x^{(0)} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Here $\rho(B) = 0.15$, and hence the iteration should converge to the exact solution of the equation $(I - B)x = 0$, which is $x = 0$. From Fig. 10.2.1 it is seen that $\|x^{(n)}\|_2$

increases by almost a factor 10^{15} until it starts to decrease after 25 iterations! Although in the long run the norm is reduced by about a factor of 0.5 at each iteration, large intermediate values of $x^{(n)}$ give rise to persistent rounding errors.

The curve in Figure 10.2.1 shows a large hump. This is a typical phenomenon in several other matrix problems and occurs also, e.g., when computing the matrix exponential e^{Bt}, when $t \to \infty$.

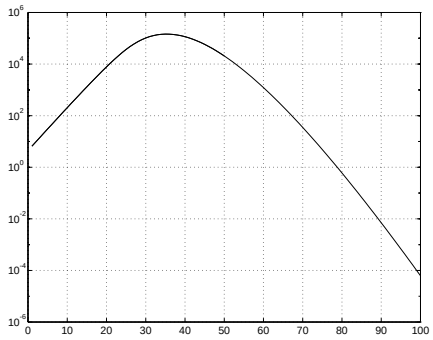

Figure 10.1.2. $\|x^{(n)}\|_2$, *where* $x^{(k+1)} = Bx^{(k)}$, *and* $x = (1, 1, \ldots, 1)^T$.

For the case when the iteration process is carried out in exact arithmetic we found a complete and simple mathematical theory of convergence for iterates $x^{(k)}$ of stationary iterative methods. According to Theorem 10.1.2 there is convergence for any $x^{(0)}$ if and only if $\rho(B) < 1$, where $\rho(B)$ is the spectral radius of B. The same condition is necessary and sufficient for $\lim_{k\to\infty} B^k = 0$ to hold. In finite precision arithmetic the convergence behavior turns out to be more complex and less easy to analyze.

It may be thought that iterative methods are less affected by rounding errors than direct solution methods, because in iterative methods one continues to work with the original matrix instead of modifying it. In Section 6.6.6 we showed that the total effect of rounding errors in Gaussian elimination with partial pivoting usually is equivalent to a perturbations in the elements of the original matrix of the order of machine roundoff. It is easy to verify that, in general, iterative methods cannot be expected to do much better than that!

Consider an iteration step with the Gauss–Seidel method performed in floating point arithmetic. Typically, in the first step an improved x_1 will be computed from previous $x_2, \ldots, x_n$ by

$$x_1 = fl\Big(\Big(b_1 - \sum_{j=1}^{n} a_{1j}x_j\Big)/a_{11}\Big) = \Big(b_1(1+\delta_1) - \sum_{j=1}^{n} a_{1j}x_j(1+\delta_j)\Big)/a_{11},$$

with the usual bounds for δ_i, cf. Section 2.4.1. This can be interpreted that we have performed an exact Gauss–Seidel step for a *perturbed problem* with elements

$b_1(1+\delta_1)$ and $a_{1i}(1+\delta_i)$, $i = 2, \ldots, n$. The bounds for these perturbations are of the same order of magnitude that for the perturbations in Gaussian elimination. The idea that we have worked with the original matrix is not correct.

Example 10.1.3. (J. H. Wilkinson) Consider the (ill-conditioned) system $Ax = b$, where

$$A = \begin{pmatrix} 0.96326 & 0.81321 \\ 0.81321 & 0.68654 \end{pmatrix}, \qquad b = \begin{pmatrix} 0.88824 \\ 0.74988 \end{pmatrix}.$$

The smallest singular value of A is $0.36 \cdot 10^{-5}$. This system is symmetric, positive definite and therefore the Gauss–Seidel method should converge, though slowly. Starting with $x_1 = 0.33116$, $x_2 = 0.70000$, the next approximation for x_1 is computed from the relation

$$x_1 = fl\big((0.88824 - 0.81321 \cdot 0.7)/0.96326\big) = 0.33116,$$

(working with five decimals). This would be an exact result if the element a_{11} was perturbed to be $0.963259\ldots$, but no progress is made towards the true solution $x_1 = 0.39473\ldots$, $x_2 = 0.62470\ldots$. The ill-conditioning has affected the computations adversely. Convergence is so slow that the modifications to be made in each step are less than $0.5 \cdot 10^{-5}$.

Stationary iterative methods may be badly affected by rounding errors for nonnormal matrices. We have seen that the "hump" phenomenon can make it possible for $\|x^{(k)}\|_2$ to increase substantially, even when the iteration matrix B is convergent; see Example 10.1.2. In such a case cancellation will occur in the computation of the final solution, and a rounding error of size $u \max_k \|x^{(k)}\|_2$ remains, where u is the machine unit.

Moreover, for as nonnormal matrix B asymptotic convergence in finite precision arithmetic is no longer guaranteed even if the condition $\rho(B) < 1$ is true in exact arithmetic. This phenomenon is related to the fact that for a matrix of a high degree of nonnormality the spectrum can be extremely sensitive to perturbations. As shown above the computed iterate $\bar{x}^{(k)}$ will at best be the exact iterate corresponding to a perturbed matrix $B + \Delta B$. Hence even though $\rho(B) < 1$ it may be that $\rho(B + \Delta B)$ is larger than one. To have convergence in finite precision arithmetic we need a stronger condition to hold, e.g.,

$$\max \rho(B + E) < 1, \quad \|E\|_2 < u\|B\|_2,$$

where u is the machine precision. (Compare the discussion of pseudospectra in Section 9.3.3.) The following rule of thumb has been suggested:

> *The iterative method with iteration matrix B can be expected to converge in finite precision arithmetic if the spectral radius computed via a backward stable eigensolver is less than 1.*

This is an instance when an inexact results is more useful than the exact result!

10.1.6 Termination Criteria

An iterative method solving a linear system $Ax = b$ is not completely specified unless clearly defined criteria are given for when to stop the iterations. Ideally such criteria should identify when the error $x - x^{(k)}$ is small enough and also detect if the error is no longer decreasing or decreasing too slowly.

Normally a user would like to specify an absolute (or a relative) tolerance ϵ for the error, and stop as soon as

$$\|x - x^{(k)}\| \leq \epsilon \tag{10.1.18}$$

is satisfied for some suitable vector norm $\|\cdot\|$. However, such a criterion cannot in general be implemented since x is unknown. Moreover, if the system to be solved is illconditioned, then because of roundoff the criterion (10.1.18) may never be satisfied.

Instead of (10.1.18) one can use a test on the residual vector $r^{(k)} = b - Ax^{(k)}$, which is computable, and stop when

$$\|r^{(k)}\| \leq \epsilon(\|A\|\,\|x^{(k)}\| + \|b\|).$$

This is often replaced by the stricter criterion

$$\|r^{(k)}\| \leq \epsilon\|b\|, \tag{10.1.19}$$

but this may be difficult to satisfy in case $\|b\| \ll \|A\|\|x\|$. Although such residual based criteria are frequently used, it should be remembered that if A is ill-conditioned a small residual does not guarantee a small relative error in the approximate solution. Since $x - x^{(k)} = A^{-1}r^{(k)}$, (10.1.19) only guarantees that $\|x - x^{(k)}\| \leq \epsilon\|A^{-1}\|\,\|b\|$, and this bound is attainable.

Another possibility is to base the stopping criterion on the Oettli–Prager backward error, see Theorem. 6.6.4. The idea is then to compute the quantity

$$\omega = \max_i \frac{|r_i^{(k)}|}{(E|x^{(k)}| + f)_i}, \tag{10.1.20}$$

where $E > 0$ and $f > 0$, and stop when $\omega \leq \epsilon$. It then follows from Theorem 6.6.4 that $x^{(k)}$ is the exact solution to a perturbed linear system

$$(A + \delta A)x = b + \delta b, \quad |\delta A| \leq \omega E, \quad |\delta b| \leq \omega f.$$

We could in (10.1.20) take $E = |A|$ and $f = |b|$, which corresponds to componentwise backward errors. However, it can be argued that for iterative methods a more suitable choice is to use a normwise backward error by setting

$$E = \|A\|_\infty e e^T, \quad f = \|b\|_\infty e, \quad e = (1, 1, \ldots, 1)^T.$$

This choice gives

$$\omega = \frac{\|r^{(k)}\|_\infty}{\|A\|_\infty \|x^{(k)}\|_1 + \|b\|_\infty}.$$

Review Questions

1. The standard discretization of Laplace equation on a square with Dirichlet boundary conditions leads to a certain matrix A. Give this matrix in its block triangular form.
2. What iterative method can be derived from the splitting $A = M - N$? How is a symmetrizable splitting defined?
3. Define the average and asymptotic rate of convergence for an iterative method $x^{(k+1)} = Bx^{(k)} + c$. Does the condition $\rho(B) < 1$ imply that the error norm $\|x - x^{(k)}\|_2$ is monotonically decreasing? If not, give a counterexample.
4. Give at least two different criteria which are suitable for terminating an iterative method.

Problems

1. Let $A \in \mathbf{R}^{n\times n}$ be a given nonsingular matrix, and $X^{(0)} \in \mathbf{R}^{n\times n}$ an arbitrary matrix. Define a sequence of matrices by
$$X^{(k+1)} = X^{(k)} + X^{(k)}(I - AX^{(k)}), \quad k = 0, 1, 2, \ldots.$$
(a) Prove that $\lim_{k\to\infty} X^{(k)} = A^{-1}$ if and only if $\rho(I - AX^{(0)}) < 1$.
Hint: First show that $I - AX^{(k+1)} = (I - AX^{(k)})^2$.
(b) Use the iterations to compute the inverse A^{-1}, where
$$A = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}, \qquad X^{(0)} = \begin{pmatrix} 1.9 & -0.9 \\ -0.9 & 0.9 \end{pmatrix}.$$
Verify that the rate of convergence is quadratic!
2. Let $A \in \mathbf{R}^{m\times n}$ be a given nonsingular matrix, Consider the stationary iterative method
$$x^{(k+1)} = x^{(k)} + \omega A^T(b - Ax^{(k)}),$$
where $A \in R^{m\times n}$ is a possibly rank deficient matrix.
(a) Show that if $\operatorname{rank}(A) = n$ and $0 < \omega < 2/\sigma^2_{\max}(A)$ then then the iteration converges to the unique solution to the normal equations $A^TAx = A^Tb$.
(b) If $\operatorname{rank}(A) < n$, then split the vector $x^{(k)}$ into orthogonal components,
$$x^{(k)} = x_1^{(k)} + x_2^{(k)}, \quad x_1^{(k)} \in \mathcal{R}(A^T), \quad x_2^{(k)} \in \mathcal{N}(A).$$
Show that the orthogonal projection of $x^{(k)} - x^{(0)}$ onto $\mathcal{N}(A)$ is zero. Conclude that in this case the iteration converges to the unique solution of the normal equations which minimizes $\|x - x^{(0)}\|_2$.
3. Show that if for a stationary iterative method $x^{(k+1)} = Bx^{(k)} + c$ it holds that $\|B\| \le \beta < 1$, and
$$\|x^{(k)} - x^{(k-1)}\| \le \epsilon(1 - \beta)/\beta,$$
then the error estimate $\|x - x^{(k)}\| \le \epsilon$ holds.

Computer Exercises

1. Let B be the 2×2 matrix in (10.1.17), and take $\lambda = \mu = 0.99$, $\alpha = 4$. Verify that $\|B^k\|_2 \geq 1$ for all $k < 805$!

2. Let $B \in \mathbf{R}^{20\times 20}$ be an upper bidiagonal matrix with diagonal elements equal to $0.025, 0.05, 0.075, \ldots, 0.5$ and elements in the superdiagonal all equal to 5.

(a) Compute and plot $\eta_k = \|x^{(k)}\|_2/\|x^{(0)}\|_2$, $k = 0:100$, where

$$x^{(k+1)} = Bx^{(k)}, \quad x^{(0)}0 = (1, 1, \ldots, 1)^T.$$

Show that $\eta_k > 10^{14}$ before it starts to decrease after 25 iterations. What is the smallest k for which $\|x^{(0)}k\|_2 < \|x^{(0)}\|_2$?

(b) Compute the eigendecomposition $B = X\Lambda X^{-1}$ and determine the condition number $\kappa = \|X\|_2\|X^{-1}\|_2$ of the transformation.

10.2 Successive Overrelaxation Methods

10.2.1 The SOR Method

It was noted early that great improvement in the rate of convergence could be obtained by the simple means of introducing a **relaxation parameter** ω in the Gauss–Seidel method

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}} r_i^{(k)},$$

with $\omega > 1$ (over-relaxation) or $\omega < 1$ under-relaxation). This lead to the famous **Successive Over Relaxation (SOR) method**, of Young [30], which remained for a long time the "workhorse" in scientific computing..

The SOR method can be written in matrix form as

$$x^{(k+1)} = x^{(k)} + \omega\Big(c + Lx^{(k+1)} - (I - U)x^{(k)}\Big),$$

where $c = D_A^{-1}b$, or after rearranging

$$(I - \omega L)x^{(k+1)} = [(1 - \omega)I + \omega U]x^{(k)} + \omega c.$$

The iteration matrix for SOR therefore is

$$B_\omega = (I - \omega L)^{-1}[(1 - \omega)I + \omega U]. \tag{10.2.1}$$

We now consider the convergence of the SOR method and first show that only values of ω, $0 < \omega < 2$ are of interest.

Theorem 10.2.1.

Let $B = L+U$ be any matrix with zero diagonal and let B_ω be the corresponding iteration matrix in the SOR method. Then we have

$$\rho(B_\omega) \geq |\omega - 1|, \tag{10.2.2}$$

with equality only if all the eigenvalues of B_ω are of modulus $|\omega - 1|$. Hence the SOR method can only converge for $0 < \omega < 2$.

Proof. Since the determinant of a triangular matrix equals the product of its diagonal elements we have

$$\det(B_\omega) = \det(I - \omega L)^{-1} \det[(1-\omega)I + \omega U] = (1-\omega)^n.$$

Also $\det(B_\omega) = \lambda_1 \lambda_2 \cdots \lambda_n$, where λ_i are the eigenvalues of B_ω. It follows that

$$\rho(B_\omega) = \max_{1 \le i \le n} |\lambda_i| \ge |1 - \omega|$$

with equality only if all the eigenvalues have modulus $|\omega - 1|$. □

The following theorem asserts that if A is a positive definite matrix, then the SOR method converges if $0 < \omega < 2$.

Theorem 10.2.2. *For a symmetric positive definite matrix A we have*

$$\rho(B_\omega) < 1, \quad \forall \omega, \quad 0 < \omega < 2.$$

Proof. We defer the proof to Theorem 10.3.4. □

For an important class of matrices an explicit expression for the optimal value of ω can be given. We first introduce the class of matrices with **property A**.

Definition 10.2.3. *The matrix A is said to have property A if there exists a permutation matrix P such that PAP^T has the form*

$$\begin{pmatrix} D_1 & U_1 \\ L_1 & D_2 \end{pmatrix}, \tag{10.2.3}$$

where D_1, D_2 are diagonal matrices.

Equivalently, the matrix $A \in \mathbf{R}^{n \times n}$ has property A if the set $\{1, 2, \ldots, n\}$ can be divided into two non-void complementary subsets S and T such that $a_{ij} = 0$ unless $i = j$ or $i \in S$, $j \in T$, or $i \in T$, $j \in S$.

Example 10.2.1. The tridiagonal matrix A below

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}, \qquad P^T A P = \begin{pmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 2 \end{pmatrix}$$

has property A, and we can choose $S = \{1, 3\}$, $T = \{2, 4\}$. Permutation of column 1 and 4 followed by a similar row permutation will give a matrix of the form above.

Definition 10.2.4. *A matrix A with the decomposition $A = D_A(I - L - U)$, D_A nonsingular, is said to be* **consistently ordered** *if the eigenvalues of*

$$J(\alpha) = \alpha L + \alpha^{-1} U, \qquad \alpha \neq 0,$$

are independent of α.

A matrix of the form of (10.2.3) is consistently ordered, since

$$J(\alpha) = \begin{pmatrix} 0 & -\alpha^{-1}D_1^{-1}U_1 \\ -\alpha D_2^{-1}L_1 & 0 \end{pmatrix} = -\begin{pmatrix} I & 0 \\ 0 & \alpha I \end{pmatrix} J(1) \begin{pmatrix} I & 0 \\ 0 & \alpha^{-1}I \end{pmatrix},$$

the matrices $J(\alpha)$ and $J(1)$ are similar and hence have the same eigenvalues. Similarly one can prove more generally that any block-tridiagonal matrix

$$A = \begin{pmatrix} D_1 & U_1 & & & \\ L_2 & D_2 & U_2 & & \\ & L_3 & \ddots & \ddots & \\ & & \ddots & \ddots & U_{n-1} \\ & & & L_n & D_n \end{pmatrix},$$

where D_i are nonsingular *diagonal* matrices is consistently ordered.

Theorem 10.2.5.

Let $A = D_A(I - L - U)$ *be a consistently ordered matrix. Then if* μ *is an eigenvalue of the Jacobi matrix so is* $-\mu$. *Further, to any eigenvalue* $\lambda \neq 0$ *of the SOR matrix* B_ω, $\omega \neq 0$, *there corresponds an eigenvalue* μ *of the Jacobi matrix, where*

$$\mu = \frac{\lambda + \omega - 1}{\omega \lambda^{1/2}} \tag{10.2.4}$$

Proof. Since A is consistently ordered the matrix $J(-1) = -L - U = -J(1)$ has the same eigenvalues as $J(1)$. Hence if μ is an eigenvalue so is $-\mu$. If λ is an eigenvalue of B_ω, then $\det(\lambda I - B_\omega) = 0$, or since $\det(I - \omega L) = 1$ for all ω, using (10.2.1)

$$\det[(I - \omega L)(\lambda I - B_\omega)] = \det[\lambda(I - \omega L) - (1 - \omega)I - \omega U] = 0.$$

If $\omega \neq 0$ and $\lambda \neq 0$ we can rewrite this in the form

$$\det\left(\frac{\lambda + \omega - 1}{\omega \lambda^{1/2}} I - (\lambda^{1/2}L + \lambda^{-1/2}U)\right) = 0$$

and since A is consistently ordered it follows that $\det\big(\mu I - (L + U)\big) = 0$, where μ given by (10.2.4). Hence μ is an eigenvalue of $L + U$. □

If we put $\omega = 1$ in (**??**) we get $\lambda = \mu^2$. Since $\omega = 1$ corresponds to the Gauss–Seidel method it follows that $\rho(B_{GS}) = \rho(B_J)^2$, which means that for consistently ordered matrices A the Gauss–Seidel method converges twice as fast as the Jacobi method.

We now state an important result due to Young [30].

Theorem 10.2.6.

Let A be a consistently ordered matrix, and assume that the eigenvalues μ of $B_J = L + U$ are real and $\rho_J = \rho(B_J) < 1$. Then the optimal relaxation parameter ω in SOR is given by

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho_J^2}}. \tag{10.2.5}$$

For this optimal value we have

$$\rho(B_{\omega_{opt}}) = \omega_{opt} - 1. \tag{10.2.6}$$

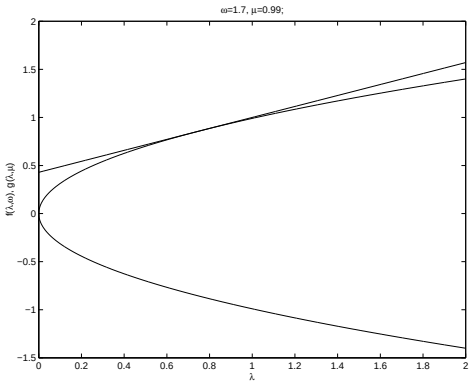

Figure 10.2.1. *$f_\omega(\lambda)$ and $g(\lambda, \mu)$ as functions of λ ($\mu = 0.99$, $\omega = \omega_b = 1.7527$).*

Proof. (See also Young [31, Section 6.2].) We consider, for a given value of μ in the range $0 < \mu \le \rho(L + U) < 1$, the two functions of λ,

$$f_\omega(\lambda) = \frac{\lambda + \omega - 1}{\omega}, \qquad g(\lambda, \mu) = \mu\lambda^{1/2}.$$

Here $f_\omega(\lambda)$ is a straight line passing through the points $(1, 1)$ and $(1 - \omega, 0)$, and $g(\lambda, \mu)$ a parabola. The relation (10.2.4) can now be interpreted as the intersection of these two curves. For given μ and ω we get for λ the quadratic equation

$$\lambda^2 + 2\Big((\omega - 1) - \frac{1}{2}\mu^2\omega^2\Big)\lambda + (\omega - 1)^2 = 0. \tag{10.2.7}$$

which has two roots

$$\lambda_{1,2} = \frac{1}{2}\mu^2\omega^2 - (\omega - 1) \pm \mu\omega\Big(\frac{1}{4}\mu^2\omega^2 - (\omega - 1)\Big)^{1/2}.$$

The larger of these roots decreases with increasing ω until eventually $f_\omega(\lambda)$ becomes a tangent to $g(\lambda, \mu)$, when $\mu^2\omega^2/4 - (\omega - 1) = 0$ (see Fig. 10.2.1) Solving for the root $\omega \le 2$ gives

$$\tilde{\omega} = \frac{1 - (1 - \mu^2)^{1/2}}{1/2\mu^2} = \frac{2}{1 + \sqrt{1 - \mu^2}}.$$

If $\omega > \tilde{\omega}$, we get two complex roots λ, which by the relation between roots and coefficients in (10.2.7) satisfy

$$\lambda_1 \lambda_2 = (\omega - 1)^2.$$

From this it follows that $|\lambda_1| = |\lambda_2| = \omega - 1$, $1 < \tilde{\omega} < \omega < 2$, and hence the minimum value of of $\max_{i=1,2} |\lambda_i|$ occurs for $\tilde{\omega}$. Since the parabola $g(\lambda, \rho(L+U))$ is the envelope of all the curves $g(\lambda, \mu)$ for $0 < \mu \leq \rho(L+U) < 1$ the theorem follows. □

Example 10.2.2. By (10.2.5) for SOR $\omega_{opt} = 2/(1 + \sin \pi h)$, giving

$$\rho(B_{\omega_{opt}}) = \omega_{opt} - 1 = \frac{1 - \sin \pi h}{1 + \sin \pi h} \approx 1 - 2\pi h. \tag{10.2.8}$$

Note that when $\lim_{n\to\infty} \omega_{opt} = 2$.

$$R_\infty(B_{\omega_{opt}}) \approx 2\pi h,$$

which shows that for the model problem the number of iterations is proportional to n for the SOR method

In Table 10.1.1 we give the number of iterations required to reduce the norm of the initial error by a factor of 10^{-3}.

Table 10.2.1. *Number of iterations needed to reduce the initial error by a factor of* 10^{-3} *for the model problem, as a function of* $n = 1/h$.

n	10	20	50	100	200
Gauss–Seidel	69	279	1,749	6,998	27,995
SOR	17	35	92	195	413

In practice, the number ρ_J is seldom known a priori, and its accurate determination would be prohibitively expensive. However, for some model problems the spectrum of the Jacobi iteration matrix is known. In the following we need the result:

A simple scheme for estimating ω_{opt} is to initially perform a fixed number of iterations using $\omega = 1$, i.e., with the Gauss–Seidel method, and attempt to measure the rate of convergence. The successive corrections satisfy

$$\delta^{(n+1)} = B_{GS}\delta^{(n)}, \qquad \delta^{(n)} = x^{(n+1)} - x^{(n)}.$$

Hence after a sufficient number of iterations we have

$$\rho(B_J)^2 = \rho(B_{GS}) \approx \theta_n, \qquad \theta_n = \|\delta^{(n+1)}\|_\infty / \|\delta^{(n)}\|_\infty,$$

An estimate of ω_{opt} is then obtained by substituting this value into (10.2.5). A closer analysis shows, however, that the number of iterations to obtain a good estimate of

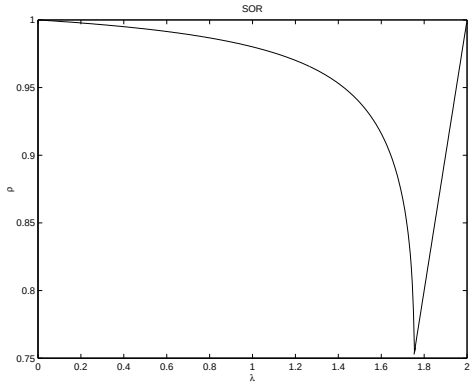

Figure 10.2.2. *The spectral radius $\rho(B_\omega)$ as a function of ω ($\rho = 0.99$, $\omega_b = 1.7527$).*

ω_{opt} is comparable to the number of iterations needed to solve the original problem by SOR. The scheme can still be practical if one wishes to solve a number of systems involving the same matrix A. Several variations of this scheme have been developed, see Young [31, p. 210].

In more complicated cases when ρ_J is not known, we have to estimate ω_{opt} in the SOR method. From Fig. 10.2.2 we conclude that it is much better to *overestimate* ω_{opt} than to underestimate it.

10.2.2 The SSOR Method

As remarked above the iteration matrix B_ω of the SOR-method is **not** symmetric and its eigenvalues are not real. In fact, in case ω is chosen slightly larger than optimal (as recommended when ρ_J is not known) the extreme eigenvalues of B_ω lie on a circle in the complex plane. However, a symmetric version of SOR, the (**SSOR**) method of Sheldon (1955), can be constructed as follows. One iteration consists of two half iterations The first half is the same as the SOR iteration. The second half iteration is the SOR method with the equations taken in reverse order. The SSOR method can be written in matrix form as

$$x^{(k+1/2)} = x^{(k)} + \omega\Big(c + Lx^{(k+1/2)} - (I-U)x^{(k)}\Big),$$
$$x^{(k+1)} = x^{(k+1/2)} + \omega\Big(c + Ux^{(k+1)} - (I-L)x^{(k+1/2)}\Big).$$

This method is due to Sheldon [1955]. The iteration matrix for SSOR is

$$S_\omega = (I-\omega U)^{-1}[(1-\omega)I + \omega L](I-\omega L)^{-1}[(1-\omega)I + \omega U].$$

It can be shown that SSOR corresponds to a splitting with the matrix

$$M_\omega = \frac{\omega}{2-\omega}\Big(\frac{1}{\omega}D_A - L_A\Big)D_A^{-1}\Big(\frac{1}{\omega}D_A - U_A\Big). \tag{10.2.9}$$

If A is symmetric, positive definite then so is M_ω. In this case the SSOR method is convergent for all $\omega \in (0, 2)$. A proof of this is obtained by a simple modification of the proof of Theorem 10.3.4.

In contrast to the SOR method, the rate of convergence of SSOR is not very sensitive to the choice of ω nor does it assume that A is consistently ordered. It can be shown that provided $\rho(LU) < 1/4$ a suitable value for ω is ω_b, where

$$\omega_b = \frac{2}{1 + \sqrt{2(1 - \rho_J)}}, \qquad \rho(S_{\omega_b}) \le \frac{1 - \sqrt{(1 - \rho_J)/2}}{1 + \sqrt{(1 - \rho_J)/2}}.$$

In particular, for the model problem in Section 10.1.2 it follows that

$$\rho(B_{\omega_b}) \le \frac{1 - \sin \pi h/2}{1 + \sin \pi h/2} \approx 1 - \pi h.$$

This is half the rate of convergence for SOR with ω_{opt}.

10.2.3 Block Iterative Methods

The basic iterative methods described so far can be generalized for block matrices A. Assume that

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \dots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix},$$

where the diagonal blocks are square and nonsingular. For this block matrix we consider the splitting

$$A = D_A - L_A - U_A, \qquad D_A = \operatorname{diag}(A_{11}, A_{22}, \dots, A_{nn}),$$

and where L_A and U_A are strictly lower and upper triangular. As before Jacobi's method can be written $D_A x^{(k+1)} = (L_A + U_A)x^{(k)} + b$, or with x is partitioned conformally

$$A_{ii}\left(x_i^{(k+1)} - x_i^{(k)}\right) = b - \sum_{j=1}^{n} A_{ij} x_j^{(k)}, \quad i = 1, \dots, n.$$

Hence it is important that linear systems in the diagonal blocks A_{ii} can be solved efficiently.

Example 10.2.3. For the model problem in Section 10.1.2 the matrix A can naturally be written in the block form where the diagonal blocks $A_{ii} = 2I + T$ are tridiagonal and nonsingular, see (10.1.4). The resulting systems can be solved with little overhead. Note that here the partitioning is such that x_i corresponds to the unknowns at the mesh points on the ith line. Hence block methods are in this context also known as "line" methods and the other methods as "point" methods.

Block versions of the Gauss–Seidel, SOR, and SSOR methods are developed similarly. For SOR we have

$$A_{ii}\Big(x_i^{(k+1)} - x_i^{(k)}\Big) = \omega\Big(b - \sum_{j=1}^{i-1} A_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} A_{ij}x_j^{(k)}\Big), \quad i = 1, \ldots, n.$$

(Taking $\omega = 1$ gives the Gauss–Seidel method.) Typically the rate of convergence is improved by a factor $\sqrt{2}$ compared to the point methods.

It can easily be verified that the SOR theory as developed in Theorems 10.2.2 and 10.2.5 are still valid in the block case. We have

$$B_\omega = (I - \omega L)^{-1}\big[(1-\omega)I + \omega U\big],$$

where $L = D_A^{-1}L_A$ and $U = D_A^{-1}U_A$. Let A be a consistently ordered matrix with nonsingular diagonal blocks A_{ii}, $1 \le i \le n$. Assume that the block Jacobi matrix B has spectral radius $\rho(B_J) < 1$. Then the optimal value of ω in the SOR method is given by (10.2.5). Note that with the block splitting any block-tridiagonal matrix

$$A = \begin{pmatrix} D_1 & U_1 & & & \\ L_2 & D_2 & U_2 & & \\ & L_3 & \ddots & \ddots & \\ & & \ddots & \ddots & U_{n-1} \\ & & & L_n & D_n \end{pmatrix},$$

is consistently ordered; for the point methods this was true only in case the block diagonal matrices D_i, $i = 1, \ldots, n$ were diagonal. In particular we conclude that with the block splitting the matrix A in (10.1.4) is consistently ordered.

10.2.4 Convergence Acceleration

Consider the stationary iterative method

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \qquad k = 0, 1, \ldots, \tag{10.2.10}$$

which corresponds to a matrix splitting $A = M - N$, and iteration matrix

$$B = M^{-1}N = I - M^{-1}A.$$

In this section we describe an important method for accelerating the convergence of the method (10.2.10) provided it is **symmetrizable**.

Definition 10.2.7. *The stationary iterative method* (10.2.10) *is said to be symmetrizable if there is a nonsingular matrix W such that the matrix $W(I - B)W^{-1}$ is symmetric and positive definite.*

For a symmetrizable method the matrix $I - B$ has real positive eigenvalues. A sufficient condition for a method to be symmetrizable is that both A and the

splitting matrix M are symmetric, positive definite., since then there is a matrix W such that $M = W^T W$, and

$$W(I - B)W^{-1} = WM^{-1}AW^{-1} = WW^{-1}W^{-T}AW^{-1} = W^{-T}AW^{-1},$$

which again is positive definite.

Example 10.2.4. If A is positive definite then in the standard splitting (10.1.11) $D_A > 0$, and hence the Jacobi method is symmetrizable with $W = D_A^{1/2}$. From (10.2.9) it follows that also the SSOR method is symmetrizable.

It is often possible to find an associated method which will converge faster than the given method by taking a weighted arithmetic mean of the first k approximations generated by the method (10.2.10),

$$\tilde{x}^{(k)} = \sum_{i=0}^{k} \gamma_{ki} x^{(i)}, \qquad \sum_{i=0}^{k} \gamma_{ki} = 1, \quad k = 0, 1, 2, \ldots, \tag{10.2.11}$$

Such methods, introduced by Varga (1957), are non-stationary and called **semi-iterative methods**. It follows from the error equation $x^{(k)} - x = B^k(x^{(0)} - x)$ that

$$\tilde{x}^{(k)} - x = \sum_{i=0}^{k} \gamma_{ki}(x^{(i)} - x) = \sum_{i=0}^{k} \gamma_{ki} B^i (x^{(0)} - x).$$

Introducing the generating polynomial we can write this

$$\tilde{x}^{(k)} - x = p_k(B)(x^{(0)} - x), \qquad p_k(\lambda) = \sum_{i=0}^{k} \gamma_{ki} \lambda^i, \tag{10.2.12}$$

where $p_k(\lambda)$ is a polynomial of degree k. Therefore this procedure is also known as **polynomial acceleration**. Note that from (10.2.11) it follows that $p_k(1) = 1$. In the special case that $M = I$ we obtain for the residual $\tilde{r}^{(k)} = b - A\tilde{x}^{(k)}$

$$\tilde{r}^{(k)} = A(x - \tilde{x}^{(k)}) = q_k(A) r^{(0)}, \qquad q_k(\lambda) = p_k(1 - \lambda). \tag{10.2.13}$$

where we have used that $A = I - B$. The polynomials $q_k(\lambda)$ have the property that $q_k(0) = 1$ and are known as **residual polynomials**.

Example 10.2.5. Consider the non-stationary Richardson iteration,

$$x^{(i+1)} = x^{(i)} + \omega_i (b - Ax^{(i)}), \quad i = 1, 2, \ldots,$$

(cf. (10.1.1)). It is easily seen that the residual vector $r^{(k)} = b - Ax^{(k)}$ satisfies

$$r^{(k)} = q_k(A) r^{(0)}, \qquad q_k(\lambda) = \prod_{i=0}^{k-1} (I - \omega_i \lambda). \tag{10.2.14}$$

Hence, by choosing a suitable sequence of parameters $\{\omega_i\}_{i=0}^{k-1}$ we can obtain any residual polynomial $q_k(\lambda)$. If the spectrum of A is real, $a < \lambda(A) < b$, we would like $|q_k(\lambda)|$ to be small in (a, b).

The most important case is Chebyshev acceleration, which we now develop. We assume that the eigenvalues $\{\lambda_i\}_{i=1}^n$ of $M^{-1}A$ are real and satisfy

$$0 < a \leq \lambda_i < b. \tag{10.2.15}$$

From (10.2.12) we get the error estimate

$$\|\tilde{x}^{(k)} - x\| = \|q_k(M^{-1}A)\| \, \|x^{(0)} - x\|,$$

If $M^{-1}A$ is Hermitian, then $\|q_k(M^{-1}A)\|_2 = \rho(q_k(M^{-1}A))$, and after k steps of the accelerated method the 2-norm of the error is reduced by at least a factor of

$$\rho(q_k(M^{-1}A)) = \max_i |q_k(\lambda_i)| \leq \max_{\lambda \in [a,b]} |q_k(\lambda)|.$$

Therefore a suitable polynomial q_k is obtained by solving the minimization problem

$$\min_{q \in \Pi_k^1} \max_{\lambda \in [a,b]} |q(\lambda)|,$$

where Π_k^1 denotes the set of residual polynomials q_k of degree $\leq k$ such that $q_k(0) = 1$. By a similar argument as used in the proof of the minimax property of Chebyshev polynomials, see Section 9.3.4, the solution to the above minimization problem is given by the shifted and normalized Chebyshev polynomials

$$q_k(\lambda) = T_k(z(\lambda))/T_k(z(0)), \tag{10.2.16}$$

where $T_k(z)$ is the Chebyshev polynomial of degree k and $z(\lambda)$ the linear transformation, which maps the interval $\lambda \in [a, b]$ onto $z \in [-1, 1]$. Hence

$$z(\lambda) = \frac{b + a - 2\lambda}{b - a} = \mu - \frac{2}{b - a}\lambda, \tag{10.2.17}$$

where

$$\mu = z(0) = \frac{b + a}{b - a} = \frac{\kappa + 1}{\kappa - 1} > 1, \qquad \kappa = \frac{b}{a}. \tag{10.2.18}$$

Note that if $M^{-1}A$ is symmetrizable κ is the condition number of $M^{-1}A$.

Since $|T_k(z)| \leq 1$, $z \in [-1, 1]$, and $\mu > 1$, we have

$$\rho(q_k(M^{-1}A)) \leq 1/T_k(\mu) < 1.$$

Hence k iterations will reduce the error norm by at least a factor

$$T_k(\mu) = \cosh(k\gamma) = \tfrac{1}{2}(e^{k\gamma} + e^{-k\gamma}) > \tfrac{1}{2}e^{k\gamma},$$

where $\mu = \cosh\gamma = (e^{\gamma} + e^{-\gamma})/2$ or $e^{\gamma} = \mu + \sqrt{\mu^2 - 1}$. We obtain using (10.2.18) after some simplification

$$\gamma = \log\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right) > \frac{2}{\sqrt{\kappa}}.$$

(Verify the last inequality! See Problem 2.) Hence $T_k(\mu) > \frac{1}{2}e^{2k/\sqrt{\kappa}}$, and to reduce the error norm at least by a factor of $\delta < 1$ it suffices to perform k iterations, where

$$k > \frac{1}{2}\sqrt{\kappa}\log\frac{2}{\delta}, \tag{10.2.19}$$

Thus the number of iterations required for a certain accuracy for the accelerated method is proportional to $\sqrt{\kappa}$ rather than κ. This is a great improvement! Note that the matrix M can be interpreted as a **preconditioner**. To increase the rate of convergence M should be chosen so that the conditioning of the matrix $M^{-1}A$ is improved.

Since the zeros of the Chebyshev polynomials $T_k(z)$ are known it is possible to implement Chebyshev acceleration as follows (cf. Example 10.2.5). To perform N steps we compute

$$x^{(k+1)} = x^{(k)} + \omega_k M^{-1}(b - Ax^{(k)}), \quad k = 0, 1, \ldots, N-1, \tag{10.2.20}$$

where

$$\omega_k = 2\big[(b+a) - (b-a)\cos\big((k+\tfrac{1}{2})/N\big)\big]^{-1}, \quad k = 0, 1, \ldots, N-1. \tag{10.2.21}$$

After N steps the iterations can be repeated in a cyclic fashion. (Note that for $N = 1$ we retrieve the optimal ω for the stationary Richardson's method derived in Section 10.1.3.) However, this scheme was shown by David Young to be very sensitive to rounding error effects unless N is small. The instability can be eliminated by using a certain reordering of the iteration parameters ω_k; see Computer exercise 1. However, one disadvantage remains, namely, the number N has to be fixed in advance.

The best way to compute the vectors $\tilde{x}^{(k)}$ is instead based on the three term recurrence relation for the Chebyshev polynomials. We have (see Section 9.3.4) $T_0(z) = 1$,

$$T_1(z) = zT_0, \qquad T_{k+1}(z) = 2zT_k(z) - T_{k-1}(z), \quad k \geq 1. \tag{10.2.22}$$

By (10.2.16) $T_k(z(\lambda)) = T_k(\mu)q_k(\lambda)$, and substituting $M^{-1}A$ for z in (10.2.22), we obtain

$$T_{k+1}(\mu)q_{k+1}(M^{-1}A) = 2z(M^{-1}A)T_k(\mu)q_k(M^{-1}A) - T_{k-1}(\mu)q_{k-1}(M^{-1}A).$$

Multiplying by $(\tilde{x}^{(0)} - x)$, using (10.2.12) and (10.2.17) we obtain

$$T_{k+1}(\mu)(\tilde{x}^{(k+1)} - x) = 2\Big(\mu I - \frac{2}{b-a}M^{-1}A\Big)T_k(\mu)(\tilde{x}^{(k)} - x) - T_{k-1}(\mu)(\tilde{x}^{(k-1)} - x).$$

From (10.2.22) with $z = \mu$ it then follows that for $k \geq 1$

$$T_{k+1}(\mu)\tilde{x}^{(k+1)} = 2\mu T_k(\mu)\tilde{x}^{(k)} - \frac{4T_k(\mu)}{b-a}M^{-1}A(\tilde{x}^{(k)} - x) - T_{k-1}(\mu)\tilde{x}^{(k-1)}.$$

Further, we have

$$M^{-1}A(\tilde{x}^{(k)} - x) = M^{-1}(A\tilde{x}^{(k)} - b) = -M^{-1}r^{(k)}.$$

Substituting $-T_{k-1}(\mu) = -2\mu T_k(\mu) + T_{k+1}(\mu)$ and dividing with $T_{k+1}(\mu)$ we obtain

$$\tilde{x}^{(k+1)} = \tilde{x}^{(k-1)} + \delta_k M^{-1}r^{(k)} + \omega_k(\tilde{x}^{(k)} - \tilde{x}^{(k-1)}), \qquad k = 1, 2, \ldots,$$

where $r^{(k)} = b - A\tilde{x}^{(k)}$, and with $\alpha = 2/(b+a)$,

$$\omega_k = 2\mu\frac{T_k(\mu)}{T_{k+1}(\mu)}, \qquad \delta_k = \alpha\omega_k, \quad k \geq 1.$$

A similar calculation for $k = 0$ gives $\tilde{x}^{(1)} = \tilde{x}^{(0)} + \alpha M^{-1}r^{(0)}$. Dropping the tilde this leads to the following algorithm:

Algorithm 10.2.1 The Chebyshev Semi-Iterative Method

Assume that the eigenvalues $\{\lambda_i\}_{i=1}^n$ of $M^{-1}A$ are real and satisfy $0 < a \leq \lambda_i < b$. Then

$$x^{(k+1)} = \begin{cases} x^{(0)} + \alpha M^{-1}r^{(0)}, & k = 0, \\ x^{(k-1)} + \omega_k\big(\alpha M^{-1}r^{(k)} + x^{(k)} - x^{(k-1)}\big), & k = 1, 2, \ldots, \end{cases} \tag{10.2.23}$$

where $\mu = (b+a)/(b-a)$, $\alpha = 2/(b+a)$, and

$$\omega_0 = 2, \quad \omega_k = \left(1 - \frac{\omega_{k-1}}{4\mu^2}\right)^{-1}, \quad k = 1, 2, \ldots.$$

A disadvantage of Chebyshev convergence acceleration is that it requires knowledge of an interval $[a, b]$ enclosing the (real) spectrum of $M^{-1}A$ is needed. If this the enclosure is too crude, then the process loses efficiency.

The eigenvalues of the iteration matrix of the SOR-method $B_{\omega_{opt}}$ are all complex and have modulus $|\omega_{opt}|$. Therefore in this case convergence acceleration is of no use. (A precise formulation is given in Young [31, p. 375].) However, Chebyshev acceleration can be applied to the Jacobi and SSOR methods, with

$$M_J = D_A, \qquad M_\omega = \frac{\omega}{2-\omega}\Big(\frac{1}{\omega}D_A - L_A\Big)D_A^{-1}\Big(\frac{1}{\omega}D_A - U_A\Big),$$

respectively, as well as block versions of these methods, often with a substantial gain in convergence rate.

Review Questions

1. When is the matrix A reducible? Illustrate this property using the directed graph of A.

2. Let $A = D_A(I - L - U)$, where $D_A > 0$. When is A said to have "property A". When is A consistently ordered? How are these properties related to the SOR method?

3. For the model problem the asymptotic rate of convergence for the classical iterative methods is proportional to h^p, where h is the mesh size. Give the value of p for Jacobi, Gauss–Seidel, SOR and SSOR. (For the last two methods it is assumed that the optimal ω is used.

4. Consider an iterative method based on the splitting $A = M - N$. Give conditions on the eigenvalues of $M^{-1}A$ which are sufficient for Chebyshev acceleration to be used. Express the asymptotic rate of convergence for the accelerated method in terms of eigenvalue bounds.

Problems

1. (a) Show that if A is reducible so is A^T. Which of the following matrices are irreducible?

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

(b) Is it true that a matrix A, in which the elements take the values 0 and 1 only, is irreducible if and only if the non-decreasing matrix sequence $(I + A)^k$, $k = 1, 2, 3, \ldots$ becomes a full matrix for some value of k?

2. The matrix A in (10.1.4) is block-tridiagonal, but its diagonal blocks are *not* diagonal matrices. Show that in spite of this the matrix is consistently ordered.

Hint: Perform a similarity transformation with the diagonal matrix

$$D(\alpha) = \operatorname{diag}(D_1(\alpha), D_2(\alpha), \ldots, D_n(\alpha)),$$

where $D_1(\alpha) = \operatorname{diag}(1, \alpha, \ldots, \alpha^{n-1})$, $D_{i+1}(\alpha) = \alpha D_i(\alpha)$, $i = 1, 2, \ldots, n-1$.

3. Verify the recursion for ω_k for the Chebyshev semi-iteration method.

4. Show that

$$\log\big((1+s)/(1-s)\big) = 2(s + s^3/3 + s^5/5 + \ldots), \quad 0 \le s < 1,$$

and use this result to prove (10.2.19).

5. Assume that A is symmetric indefinite with its eigenvalues contained in the union of two intervals of equal length,

$$\mathcal{S} = [a, b] \cup [c, d], \quad a < b < 0, \quad 0 < c < d,$$

where $d - c = b - a$. Then the Chebyshev semi-iterative method cannot be applied directly to the system $Ax = b$. Consider instead the equivalent system

$$Bx = c, \qquad B = A(A - \alpha I), \qquad c = Ab - \alpha b.$$

(a) Show that if $\alpha = d + a = b + c$, then the eigenvalues of B are positive and real and contained in the interval $[-bc, -ad]$.

(b) Show that the matrix B has condition number

$$\kappa(B) = \frac{d}{c} \cdot \frac{|a|}{|b|} = \frac{d}{c} \frac{d - c + |b|}{|b|}.$$

Use this to give estimates for the two special cases (i) Symmetric intervals with respect to the origin. (ii) The case when $|b| \gg c$.

Computer Exercises

1. Let A be a matrix with real eigenvalues $\{\lambda_i\}_{i=1}^n$, $0 < a \leq \lambda_i < b$. Then the Chebyshev semi-iterative method for solving $Ax = b$ can be implemented by the recursion (10.2.20)–(10.2.21). The instability of this scheme can be eliminated by using an ordering of the iteration parameters ω_k given by Lebedev and Finogenov. For $N = 2^p$ this permutation ordering κ is constructed by the following Matlab program:

```
N = 2^p;  int = 1;
kappa = ones(1, N);
for i = 1 : p
    int = 2 * int; ins = int + 1;
    for j = int/2 : -1 : 1
        kappa(2 * j) = ins - kappa(j);
        kappa(2 * j - 1) = kappa(j);
    end;
end;
```

Implement and test this method using the system $Ax = b$ from the Laplace equation on the unit square, with A block tridiagonal

$$A = \text{tridiag}(-I, T + 2I, -I) \in \mathbf{R}^{n^2 \times n^2}, \qquad T = \text{tridiag}(-1, 2 - 1) \in \mathbf{R}^{n \times n}.$$

Construct the right hand so that the exact solution becomes $x = (1, 1, \ldots, 1, 1)^T$. Let $x^{(0)} = 0$ as initial approximation and solve this problem using

- The implementation based on the three term recursion of Chebyshev polynomials
- Richardson implementation with natural ordering of the parameters
- Richardson implementation with the Lebedev–Finogenov ordering of the parameters

Take $n = 50$ and $N = 128$. Use the same number of iterations in all three implementations. List in each case the maximum norm of the error and the residual. Compare the results and draw conclusions!

10.3 Projection Methods

10.3.1 General Principles

Consider a linear system $Ax = b$, where $A \in \mathbf{R}^{n\times n}$. Suppose we want to find an approximate solution $\hat{x}$ in a subspace $\mathcal{K}$ of dimension m. Then m independent conditions are needed to determine $\hat{x}$. One way to obtain these is by requiring that the residual $b - A\hat{x}$ is orthogonal to a subspace $\mathcal{L}$ of dimension m, i.e.,

$$\hat{x} \in \mathcal{K}, \qquad b - A\hat{x} \perp \mathcal{L}. \tag{10.3.1}$$

Many important classes of iterative methods can be interpreted as being projection methods in this general sense. The conditions (10.3.1) are often known as **Petrov–Galerkin conditions**.

We can obtain a matrix form of (10.3.1) by introducing basis vectors in the two subspaces. If we let

$$\mathcal{K} = \mathcal{R}(U), \qquad \mathcal{L} = \mathcal{R}(V), \tag{10.3.2}$$

where $U = (u_1, \ldots, u_m)$, $V = (v_1, \ldots, v_m)$, then we can write (10.3.1) as $V^T(b - AUz) = 0$, for some $z \in \mathbf{R}^m$. Hence $\hat{x} = Uz$ is obtained by solving the reduced system

$$\hat{A}z = V^T b, \qquad \hat{A} = V^T AU. \tag{10.3.3}$$

We usually have $m \ll n$, and when m is small this system can be solved by a direct method.

Example 10.3.1. Even though A is nonsingular the matrix $\hat{A}$ may be singular. Take, e.g., $m = 1$, $U = V = e_1$, and

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

Then $\hat{A} = 0$. Note that the matrix A here is symmetric, but not positive definite.

There are two important special cases in which the matrix $\hat{A}$ can be guaranteed to be nonsingular.

1. Let A by symmetric, positive definite (s.p.d.) and $\mathcal{L} = \mathcal{K}$. Then we can take $V = U$, and have $\hat{A} = U^T AU$. Clearly $\hat{A}$ is s.p.d., and hence also nonsingular.
2. Let A be nonsingular and $\mathcal{L} = A\mathcal{K}$. Then we can take $V = AU$, and we get $\hat{A} = U^T A^T AU$. Here $A^T A$ is s.p.d. and hence $\hat{A}$ is nonsingular.

We now derive important optimality properties satisfied in these two special cases. For this purpose we first define a new inner product and norm related to a s.p.d. matrix A.

Definition 10.3.1. *For an s.p.d. matrix A we define a related A-inner product and A-norm by*

$$(u, v)_A = u^T Av, \qquad \|u\|_A = (u^T Au)^{1/2}, \tag{10.3.4}$$

It is easily verified that $\|u\|_A$ satisfies the conditions for a norm.

Lemma 10.3.2.

Let A by symmetric, positive definite (s.p.d.) and consider the case $\mathcal{L} = \mathcal{K}$, $(V = U)$. Then $\hat{x} = U(U^TAU)^{-1}U^Tb$ minimizes the A-norm of the error over all vectors $x \in \mathcal{K}$, i.e., $\hat{x}$ solves the problem

$$\min_{x\in\mathcal{K}} \|x - x^*\|_A, \quad x^* = A^{-1}b. \tag{10.3.5}$$

Proof. By (10.3.1) $\hat{x}$ satisfies $v^T(b - A\hat{x}) = 0$, $\forall v \in \mathcal{K}$. Let $\hat{e} = \hat{x} - x^*$ be the error in $\hat{x}$. Then for the error in $\hat{x} + v$, $v \in \mathcal{K}$ we have $e = \hat{e} + v$, and

$$\|e\|_A^2 = \hat{e}^TA\hat{e} + v^TAv + 2v^TA\hat{e}.$$

But here the last term is zero because $v^TA\hat{e} = v^T(A\hat{x} - b) = 0$. It follows that $\|e\|_A$ is minimum if $v = 0$. □

A related result is obtained for the second case.

Lemma 10.3.3.

Let A be nonsingular and consider the case $\mathcal{L} = A\mathcal{K}$, $(V = AU)$. Then $\hat{x} = U(U^TA^TAU)^{-1}U^TA^Tb$ minimizes the 2-norm of the residual over all vectors $x \in \mathcal{K}$, i.e.,

$$\min_{x\in\mathcal{K}} \|b - Ax\|_2. \tag{10.3.6}$$

Proof. Using $x = Uz$ we have $\|b - Ax\|_2 = \|b - AUz\|_2$, which is minimized when z satisfies the normal equations $U^TA^TAUz = U^TA^Tb$. This gives the desired result. □

In an iterative method often a sequence of projection steps of the above form is taken. Then we need to modify the above algorithms slightly so that they can start from a given approximation x_k.[4]

If we let $x = x_k + z$, then z satisfies the system $Az = r_k$, where $r_k = b - Ax_k$. In step k we now apply the above projection method to this system. Hence we require that $z \in \mathcal{K}$ and that $r_k - Az = b - A(x_k + z) \perp \mathcal{L}$. This gives the equations

$$r_k = b - Ax_k, \qquad \hat{z} = (V^TAU)^{-1}V^Tr_k, \qquad x_{k+1} = x_k + Uz. \tag{10.3.7}$$

for computing the new approximation x_{k+1}. A generic projection algorithm is obtained by starting from some x_0 (e.g., $x_0 = 0$), and repeatedly perform (10.3.7) for a sequence of subspaces $\mathcal{L} = \mathcal{L}_k$, $\mathcal{K} = \mathcal{K}_k$, $k = 1, 2, \ldots$.

[4] In the rest of this chapter we will use vector notations and x_k will denote the kth approximation and not the kth component of x.

10.3.2 The One-Dimensional Case

The simplest case of a projection method is when $m = 1$. Then in step k we take $\mathcal{L}_k = \operatorname{span}(v_k)$, and $\mathcal{K}_k = \operatorname{span}(u_k)$. Starting from some x_0, we update x_k in the kth step by

$$r_k = b - Ax_k, \quad \alpha_k = \frac{v_k^T r_k}{v_k^T A u_k}, \quad x_{k+1} = x_k + \alpha_k u_k, \tag{10.3.8}$$

where we have to require that $v_k^T A u_k \neq 0$. By construction the new residual r_{k+1} is orthogonal to v_k. Note that r_k can be computed recursively from

$$r_k = r_{k-1} - \alpha_{k-1} A u_{k-1}. \tag{10.3.9}$$

This expression is obtained by multiplying $x_k = x_{k-1} + \alpha_{k-1} u_{k-1}$ by A and using the definition $r_k = b - Ax_k$ of the residual. Since Au_{k-1} is needed for computing α_{k-1} using the recursive residual will save one matrix times vector multiplication.

If A is s.p.d. we can take $v_k = u_k$ and the above formulas become

$$r_k = b - Ax_k, \quad \alpha_k = \frac{u_k^T r_k}{u_k^T A u_k}, \quad x_{k+1} = x_k + \alpha_k u_k, \tag{10.3.10}$$

In this case x_{k+1} minimizes the quadratic functional

$$\phi(x) = \|x - x^*\|_A^2 = (x - x^*)^T A (x - x^*) \tag{10.3.11}$$

for all vectors of the form $x_k + \alpha_k u_k$.

The vectors u_k are often called **search directions**. Expanding the function $\phi(x_k + \alpha u_k)$ with respect to α, we obtain

$$\phi(x_k + \alpha u_k) = \phi(x_k) - \alpha u_k^T (b - Ax_k) + \frac{1}{2}\alpha^2 u_k^T A u_k. \tag{10.3.12}$$

Taking $\alpha = \omega \alpha_k$ where α_k is given by (10.3.10) we obtain

$$\phi(x_k + \omega\alpha_k u_k) = \phi(x_k) - \rho(\omega)\frac{(u_k^T r_k)^2}{u_k^T A u_k}, \quad \rho(\omega) = \frac{1}{2}\omega(2 - \omega), \tag{10.3.13}$$

which is a quadratic function of ω. In a projection step ($\omega = 1$) the line $x_k + \alpha u_k$ is tangent to the ellipsoidal level surface $\phi(x) = \phi(x_{k+1})$, and $\phi(x_k + \alpha_k u_k) < \phi(x_k)$ provided that $u_k^T r_k \neq 0$. More generally, if $u_k^T r_k \neq 0$ we have from symmetry that

$$\phi(x_k + \omega\alpha_k u_k) < \phi(x_k), \qquad 0 < \omega < 2.$$

For the error in $x_{k+1} = x_k + \omega\alpha_k u_k$ we have

$$\hat{x} - x_{k+1} = \hat{x} - x_k - \omega\frac{u_k^T r_k}{u_k^T A u_k} u_k = \left(I - \omega\frac{u_k u_k^T}{u_k^T A u_k} A\right)(\hat{x} - x_k).$$

This shows that the error in each step is transformed by a linear transformation.

Example 10.3.2. For the Gauss–Seidel method in the ith minor step the ith component of the current approximation x_k is changed so that the ith equation is satisfied, i.e., we take

$$x_k := x_k - \hat{\alpha} e_i, \qquad e_i^T(b - A(x_k - \hat{\alpha} e_i)) = 0,$$

where e_i is the ith unit vector. Hence the Gauss–Seidel method is equivalent to a sequence of one-dimensional modifications where the search directions are chosen equal to the unit vectors in cyclic order $e_1, \ldots, e_n, e_1, \ldots, e_n, \ldots$.

This interpretation can be used to prove convergence for the Gauss–Seidel (and more generally the SOR method) for the case when A is s.p.d..

Theorem 10.3.4.
If A is symmetric, positive definite then the SOR method converges for $0 < \omega < 2$, to the unique solution of $Ax = b$. In particular the Gauss–Seidel method, which corresponds to $\omega = 1$, converges.

Proof. In a minor step using search direction e_i the value of ϕ will decrease unless $e_i^T(b - Ax_k) = 0$, i.e., unless x_k satisfies the ith equation. A major step consists of a sequence of n minor steps using the search directions $e_1, \ldots, e_n$. Since each minor step effects a linear transformation of the error $y_k = \hat{x} - x_k$, in a major step it holds that $y_{k+1} = Ky_k$, for some matrix K. Here $\|Ky_k\|_A < \|y_k\|_A$ unless y_k is unchanged in *all* minor steps, $i = 1, \ldots, n$, which would imply that $y_k = 0$. Therefore if $y_k \neq 0$, then $\|Ky_k\|_A < \|y_k\|_A$, and thus $\|K\|_A < 1$. It follows that

$$\|K^n y_0\|_A \leq \|K\|_A^n \|y_0\|_A \to 0 \quad \text{when} \quad n \to \infty,$$

i.e., the iteration converges.

If we define the minor step as $x := \omega \hat{\alpha} e_i$, where ω is a fix relaxation factor, the convergence proof also holds. (We may even let ω vary with i, although the proof assumes that ω for the same i has the same value in all major steps.) This shows that the SOR method is convergent and by Theorem 10.1.3 this is equivalent to $\rho(B_\omega) < 1$, $0 < \omega < 2$. □

We make two remarks about the convergence proof. First, it also holds if for the basis vectors $\{e_i\}_{i=1}^n$ we substitute an arbitrary set of *linearly independent vectors* $\{p_j\}_{i=1}^n$. Second, if A is a positive diagonal matrix, then we obtain the *exact* solution by the Gauss–Seidel method after n minor steps. Similarly, if A assumes diagonal form after a coordinate transformation with, $P = (p_1, \ldots, p_n)$, i.e., if $P^T AP = D$, then the exact solution will be obtained in n steps using search directions $p_1, \ldots, p_n$. Note that this condition is equivalent to the requirement that the vectors $\{p_j\}_{i=1}^n$ should be A-orthogonal, $p_i^T A p_j = 0, \qquad i \neq j$.

10.3.3 The Method of Steepest Descent

From the expansion (10.3.12) it is clear that the negative gradient of $\phi(x)$ with respect to x equals $-\nabla\phi(x) = b - Ax$. Hence the direction in which the function

ϕ decreases most rapidly at the point x_k equals the residual $r_k = b - Ax_k$. The **method of steepest descent** is a one-dimensional projection method where we take $v_k = u_k = r_k$. Then

$$r_k = b - Ax_k, \quad \alpha_k = \frac{r_k^T r_k}{r_k^T A r_k}, \quad x_{k+1} = x_k + \alpha_k r_k. \tag{10.3.14}$$

and according to (10.3.13) when $r_k \neq 0$, we have $\phi(x_{k+1}) < \phi(x_k)$.

We now derive an expression for the rate of convergence of the steepest descent method. Denoting the error in x_k by $e_k = x_k - x^*$ we have

$$\begin{aligned}\|e_{k+1}\|_A^2 = e_{k+1}^T A e_{k+1} = -r_{k+1}^T e_{k+1} = -r_{k+1}^T (e_k + \alpha_k r_k) \\ = -(r_k - \alpha_k A r_k)^T e_k = e_k^T A e_k - \alpha_k r_k^T r_k,\end{aligned}$$

where we have used that $r_{k+1}^T r_k = 0$. Using the expression (10.3.14) for α_k we obtain

$$\|e_{k+1}\|_A^2 = \|e_k\|_A^2 \left(1 - \frac{r_k^T r_k}{r_k^T A r_k} \frac{r_k^T r_k}{r_k^T A^{-1} r_k}\right). \tag{10.3.15}$$

To estimate the right hand side we need the following result.

Lemma 10.3.5 (Kantorovich[5] inequality).

Let A be a real symmetric matrix with eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Then for any vector x it holds

$$\frac{(x^T A x)(x^T A^{-1} x)}{(x^T x)^2} \leq \frac{1}{4}\left(\kappa^{1/2} + \kappa^{-1/2}\right)^2, \tag{10.3.16}$$

where $\kappa = \lambda_n/\lambda_1$ is the condition number of A.

Proof. (After D. Braess) Let $\mu = (\lambda_1 \lambda_n)^{1/2}$ be the geometric mean of the eigenvalues and consider the symmetric matrix $B = \mu^{-1} A + \mu A^{-1}$. The eigenvalues of B satisfy

$$\lambda_i(B) = \mu^{-1}\lambda_i + \mu \lambda_i^{-1} \leq \kappa^{1/2} + \kappa^{-1/2}, \quad i = 1, \ldots, n.$$

Hence, by the Courant maximum principle, for any vector x it holds

$$x^T B x = \mu^{-1}(x^T A x) + \mu (x^T A^{-1} x) \leq (\kappa^{1/2} + \kappa^{-1/2})(x^T x).$$

The left hand can be bounded using the simple inequality

$$(ab)^{1/2} \leq \tfrac{1}{2}(\mu^{-1} a + \mu b), a, b > 0.$$

Squaring this and taking $a = x^T A x$ and $b = x^T A^{-1} x$ the lemma follows. □

From (10.3.15) and Kantorovich's inequality it follows for the method of steepest descent that

$$\|e_{k+1}\|_A^2 \leq \|e_k\|_A^2 \left(\frac{\kappa^{1/2} - \kappa^{-1/2}}{\kappa^{1/2} + \kappa^{-1/2}}\right)^2,$$

[5] Leonid V. Kantorovich, 1912–1986, Moscow, 1975 Nobel Lauerate in Economics.

and hence

$$\|x - x_k\|_A \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x - x_0\|_A. \tag{10.3.17}$$

It can also be shown that asymptotically this bound is sharp. Hence, the asymptotic rate of convergence only depends *on the extreme eigenvalues of A.*

If the matrix A is ill-conditioned the level curves of ϕ are very elongated hyper-ellipsoids. Then the successive iterates x_k, $k = 0, 1, 2, \ldots$ will zig-zag slowly towards the minimum $x = A^{-1}b$ as illustrated in Fig. 10.3.3 for a two dimensional case. Note that successive search directions are orthogonal.

Figure 10.3.1. *Convergence of the steepest descent method.*

We now consider the more general case when $u_0 = p_0 = r_0$ (i.e., the steepest descent direction) and the search direction $u_{k+1} = p_{k+1}$ is chosen as a linear combination of the negative gradient r_{k+1} and the previous search direction p_k, i.e.,

$$p_{k+1} = r_{k+1} + \beta_k p_k, \quad k = 0, 1, 2 \ldots. \tag{10.3.18}$$

Here the parameter β_k remains to be determined. (Note that $\beta_k = 0$ gives the method of steepest descent.) From (10.3.13) we know that to get $\phi(x_{k+1}) < \phi(x_k)$ we must have $p_k^T r_k \neq 0$. Replacing $(k+1)$ by k in (10.3.18) and multiplying by r_k^T, we obtain

$$r_k^T p_k = r_k^T r_k + \beta_{k-1} r_k^T p_{k-1} = r_k^T r_k, \tag{10.3.19}$$

since r_k is orthogonal to p_{k-1}. It follows that $r_k^T p_k = 0$ implies $r_k = 0$ and thus $x_k = A^{-1}b$. Hence unless x_k is the solution, the next iteration step is always defined, regardless of the value of the parameter β_k, and $\phi(x_{k+1}) < \phi(x_k)$. From (10.3.19) we also obtain the alternative expression

$$\alpha_k = (r_k^T r_k)/(p_k^T A p_k). \tag{10.3.20}$$

Review Questions

1. Let $\phi(x) = \frac{1}{2}x^TAx - x^Tb$, where A is symmetric positive definite, and consider the function $\varphi(\alpha) = \phi(x_k + \alpha p_k)$, where p_k is a search direction and x_k the current approximation to $x = A^{-1}b$. For what value $\alpha = \alpha_k$ is $\varphi(\alpha)$ minimized? Show that for $x_{k+1} = x_k + \alpha_k p_k$ it holds that $b - Ax_{k+1} \perp p_k$.
2. Show that minimizing the quadratic form $\frac{1}{2}x^TAx - x^Tb$ along the search directions $p_i = e_i$, $i = 1, 2, \ldots, n$ is equivalent to one step of the Gauss–Seidel method.
3. How are the search directions chosen in the method of steepest descent? What is the asymptotic rate of convergence of this method?

10.4 Krylov Subspace Methods

The Lanczos algorithm and the conjugate gradient algorithm of Hestenes and Stiefel are the most important examples of Krylov subspace methods. They were developed already in the early 1950s, but did not come into wide use until twenty years later. Now these methods, combined with preconditioning techniques, have been developed into sophisticated solvers for large scale linear systems. Today these are the standard method for solving linear systems involving large, sparse, symmetric (or Hermitian) systems.

10.4.1 The Conjugate Gradient Method

We consider now projection methods where the subspaces $\mathcal{K}$ and $\mathcal{L}$ are chosen to be the sequence of **Krylov subspaces** $\mathcal{K}_{k+1}(r_0, A)$, $k = 0, 1, 2, \ldots$, where $r_0 = b - Ax_0$, and

$$\mathcal{K}_m(r_0, A) = \text{span}\,\{r_0, Ar_0, \ldots, A^{m-1}r_0\}. \tag{10.4.1}$$

This choice leads to one of the most important iterative methods for solving large symmetric positive definite systems, the **conjugate gradient method**.

If $p_0 = r_0$, and the recurrence (10.3.18) is used to generate p_{k+1} then a simple induction argument shows that the vectors p_k and r_k both will lie in the $\mathcal{K}_{k+1}(r_0, A)$. In the conjugate gradient method the parameter β_k is chosen to make p_{k+1} A-orthogonal or conjugate to the previous search direction, i.e.,

$$p_{k+1}^TAp_k = 0. \tag{10.4.2}$$

(A motivation to this choice is given in the second remark to Theorem 10.3.4.) Multiplying (10.3.18) by p_k^TA and using (10.4.2) it follows that

$$\beta_k = -(p_k^TAr_{k+1})/(p_k^TAp_k). \tag{10.4.3}$$

We now prove the important result that this choice will in fact make p_{k+1} A-conjugate to *all* previous search directions!

Lemma 10.4.1.

In the conjugate gradient algorithm the residual vector r_k is orthogonal to all previous search directions and residual vectors

$$r_k^T p_j = 0, \quad j = 0, \ldots, k-1, \tag{10.4.4}$$

and the search directions are mutually A-conjugate

$$p_k^T A p_j = 0, \quad j = 0, \ldots, k-1. \tag{10.4.5}$$

Proof. We first prove the relations (10.4.4) and (10.4.5) jointly by induction. Clearly r_k is orthogonal to the previous search direction p_{k-1}, and (10.4.2) shows that also (10.4.5) holds for $j = k-1$, Hence these relations are certainly true for $k = 1$.

Assume now that the statements are true for some $k \geq 1$. From $p_k^T r_{k+1} = 0$, changing the index, and taking the scalar product with p_j, $0 \leq j < k$ we get

$$r_{k+1}^T p_j = r_k^T p_j - \alpha_k p_k^T A p_j.$$

From the induction hypothesis this is zero, and since $r_{k+1}^T p_k = 0$ it follows that (10.4.4) holds for $k := k+1$. Using equation (10.3.18), the induction hypothesis and equation (10.3.9) and then (10.3.18) again we find for $0 < j < k$

$$\begin{aligned} p_{k+1}^T A p_j &= r_{k+1}^T A p_j + \beta_k p_k^T A p_j = \alpha_j^{-1} r_{k+1}^T (r_j - r_{j+1}) \\ &= \alpha_j^{-1} r_{k+1}^T (p_j - \beta_{j-1} p_{j-1} - p_{j+1} + \beta_j p_j), \end{aligned}$$

which is zero by equation (10.4.4). For $j = 0$ we use $r_0 = p_0$ in forming the last line of the equation. For $j = k$ we use (10.4.2), which yields (10.4.5). □

Since the vectors $p_0, \ldots, p_{k-1}$ span the Krylov subspace $\mathcal{K}_k(r_0, A)$ the equation (10.4.4) shows that $r_k \perp \mathcal{K}_k(r_0, A)$. This relation shows that the conjugate gradient implements the projection method obtained by taking $\mathcal{K} = \mathcal{L} = \mathcal{K}_k(r_0, A)$. Hence from Lemma 10.3.2 we have the following *global minimization property.*

Theorem 10.4.2.

The vector x_k in the conjugate gradient method solves the minimization problem

$$\min_x \phi(x) = \frac{1}{2}\|x - x^*\|_A^2, \quad x - x_0 \in \mathcal{K}_k(r_0, A) \tag{10.4.6}$$

From this property it follows directly that the "energy" norm $\|x - x^*\|_A$ in the CG method is monotonically decreasing. It can also be shown that the error norm $\|x - x_k\|_2$ is monotonically decreased (see Hestenes and Stiefel [13]).

Since the vectors $r_0, \ldots, r_{k-1}$ span the Krylov subspace $\mathcal{K}_k(r_0, A)$ the following orthogonality relations also hold:

$$r_k^T r_j = 0, \quad j = 0, \ldots, k-1. \tag{10.4.7}$$

Equation (10.4.7) ensures that in exact arithmetic the conjugate gradient method will terminate after at most n steps. For suppose the contrary is true. Then $r_k \neq 0$, $k = 0, 1, \ldots, n$ and by (10.4.7) these $n+1$ nonzero vectors in $\mathbf{R}^n$ are mutually orthogonal and hence linearly independent, which is impossible. Hence the conjugate gradient method is in effect a direct method! However, as is now well know, round-off errors spoil the finite termination property and this aspect has little practical relevance.

From these relations, we can conclude that the residuals $r_0, r_1, \ldots, r_k$ are the same vectors as those obtained from the sequence $r_0, Ar_0, \ldots, A^k r_0$ by Gram-Schmidt orthogonalization. This gives a connection to the Lanczos process described in Section 10.8.4, which is further discussed below in Section 10.4.2. The vectors $p_0, p_1, p_2, \ldots$ may be constructed similarly from the conjugacy relation (10.4.5).

An alternative expression for β_k is obtained by multiplying the recursive expression for the residual $r_{k+1} = r_k - \alpha_k A p_k$ by r_{k+1}^T and using the orthogonality (10.4.7) to get $r_{k+1}^T r_{k+1} = -\alpha_k r_{k+1}^T A p_k$. Equations (10.3.20) and (10.4.3) then yield

$$\beta_k = r_{k+1}^T r_{k+1} / r_k^T r_k.$$

We observe that in this expression for β_k the matrix A is not needed. This property is important when the conjugate gradient method is extended to non-quadratic functionals.

We now summarize the conjugate gradient method. We have seen that there are alternative, mathematically equivalent formulas for computing r_k, α_k and β_k. However, these are not equivalent with respect to accuracy, storage and computational work. A comparison tends to favor the following version:

Algorithm 10.4.1 The Conjugate Gradient Method

$$\begin{aligned}
&r_0 = b - Ax_0; \quad p_0 = r_0;\\
&\textbf{for } k = 0, 1, 2, \ldots \textbf{ while } \|r_k\|_2 > \epsilon \textbf{ do}\\
&\quad \alpha_k = (r_k, r_k)/(p_k, Ap_k);\\
&\quad x_{k+1} = x_k + \alpha_k p_k;\\
&\quad r_{k+1} = r_k - \alpha_k A p_k;\\
&\quad \beta_k = (r_{k+1}, r_{k+1})/(r_k, r_k);\\
&\quad p_{k+1} = r_{k+1} + \beta_k p_k;\\
&\textbf{end}
\end{aligned}$$

Here the inner product used is $(p, q) = p^T q$. Four vectors x, r, p and Ap need to be stored. Each iteration step requires one matrix by vector product when forming Ap, two vector inner products and three scalar by vector products.

By instead taking the inner product in the above algorithm to be $(p, q) = p^T A q$ we obtain a related method that in each step minimizes the Euclidian norm of

the residual over the same Krylov subspace. In this algorithm the vectors Ap_i, $i = 0, 1, \ldots$ are orthogonal. In addition, the residual vectors are required to be A-orthogonal, i.e., conjugate. Consequently this method is called the **conjugate residual method**. This algorithm requires one more vector of storage and one more vector update than the conjugate gradient method. Therefore, when applicable the conjugate gradient method is usually preferred over the conjugate residual method.

10.4.2 CG and the Lanczos Process

To reveal the close connection between the conjugate gradient method and the Lanczos Process we define the following matrices:

$$R_k = \left(\frac{r_1}{\|r_1\|_2}, \ \frac{r_2}{\|r_2\|_2}, \ \cdots, \ \frac{r_k}{\|r_k\|_2} \right) \tag{10.4.8}$$

$$P_k = \left(\frac{p_1}{\|r_1\|_2}, \ \frac{p_2}{\|r_2\|_2}, \ \cdots, \ \frac{p_k}{\|r_k\|_2} \right) \tag{10.4.9}$$

$$L_k = \begin{pmatrix} 1 & & & & \\ -\sqrt{\beta_1} & 1 & & & \\ & -\sqrt{\beta_2} & 1 & & \\ & & \ddots & \ddots & \\ & & & -\sqrt{\beta_{k-1}} & 1 \end{pmatrix} \tag{10.4.10}$$

$$D_k = \text{diag}\,(\,\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_k\,) \tag{10.4.11}$$

Assuming that $r_k \neq 0$, $k = 1 : n$ R_n is an orthogonal matrix. Then we have the relations

$$AP_nD_n = R_nL_n, \qquad P_nL_n = R_n.$$

Eliminating P_n from the first relation we obtain

$$AR_n = R_n(L_mD_m^{-1}L_m) = R_nT_n,$$

Hence provides an orthogonal similarity transformation of A to symmetric tridiagonal form T_n.

In Section 9.9.4 we described the Lanczos process for a real symmetric matrix A. If this process can be carried out for k steps, starting with a vector v_1, it generates a symmetric tridiagonal matrix

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-1} & \alpha_{k-1} & \beta_k \\ & & & \beta_k & \alpha_k \end{pmatrix}.$$

and a matrix $V_k = (v_1, \ldots, v_k)$ with orthogonal columns spanning the Krylov subspace $\mathcal{K}_k(v_1, A)$ such that

$$AV_k = V_kT_k + \beta_{k+1}v_{k+1}e_k^T. \tag{10.4.12}$$

In the context of solving the linear system $Ax = b$, using Krylov subspaces the appropriate choice of starting vector is

$$\beta_1 v_1 = r_0 = b - Ax_0, \qquad \beta_1 = \|r_0\|_2.$$

We write the kth approximation as $x_k = x_0 + V_k y_k \in \mathcal{K}_k(r_0, A)$. Here y_k is determined by the condition that $r_k = b - Ax_k$ is orthogonal to $\mathcal{K}_k(r_0, A)$, i.e., $V_k^T r_k = 0$. Using (10.4.12) we have

$$r_k = r_0 - AV_k y_k = \beta_1 v_1 - V_k T_k y_k - \beta_{k+1}(e_k^T y_k) v_{k+1}. \tag{10.4.13}$$

Since $V_k^T v_{k+1} = 0$ and $V_k^T v_1 = e$, multiplying by V_k^T gives

$$V_k^T r_k = 0 = \beta_1 e_1 - T_k y_k.$$

Hence y_k is obtained by solving the tridiagonal system

$$T_k y_k = \beta_1 e_1, \tag{10.4.14}$$

and then $x_k = x_0 + V_k y_k$. Mathematically this gives the same sequence of approximations as generated by the CG method. Moreover, the columns of V_k equal the first k residual vectors in the conjugate gradient method, normalized to unit length.

The Lanczos process stops if $\beta_{k+1} = \|r_k\|_2 = 0$ since then v_{k+1} is not defined. However, then we have $AV_k = V_k T_k$ and using (10.4.13)

$$0 = r_k = \beta_1 v_1 - V_k T_k y_k = r_0 - AV_k y_k = r_0 - A(x_k - x_0).$$

It follows that $Ax_k = b$, i.e. x_k is an exact solution.

The recursion in the conjugate gradient method is obtained from (10.4.14) by computing the Cholesky factorization of $T_k = R_k^T R_k$. This is always possible. Suppose the Lanczos process stops for $k = l \leq n$. Then, since A is a positive definite matrix then $T_l = V_l^T A V_l$ is also positive definite. Thus T_k, $k \leq l$, which is a principal submatrix of T_l, is also positive definite and its Cholesky factorization must exist.

So far we have discussed the Lanczos process in exact arithmetic. In practice, roundoff will cause the generated vectors to lose orthogonality. A possible remedy is to reorthogonalize each generated vector v_{k+1} to all previous vectors $v_k, \ldots, v_1$. This is however very costly both in terms of storage and operations. The effect of finite precision on the Lanczos method is the same as for the CG method; it slows down convergence, but fortunately does not prevent accurate approximations to be found!

10.4.3 Convergence of the Conjugate Gradient Method

In a Krylov subspace method the approximations are of the form $x_k - x_0 \in \mathcal{K}_k(r_0, A)$, $k = 1, 2, \ldots$. With $r_k = b - Ax_k$ it follows that $r_k - r_0 \in A\mathcal{K}_k(r_0, A)$. Hence the residual vectors can be written

$$r_k = q_k(A) r_0,$$

where $q_k \in \tilde{\Pi}_k^1$, the set of polynomials q_k of degree k with $q_k(0) = 1$. Since

$$\phi(x) = \tfrac{1}{2}\|x - x^*\|_A^2 = \tfrac{1}{2} r^T A^{-1} r = \|r\|_{A^{-1}}^2,$$

the optimality property in Theorem 10.4.2 can alternatively be stated as

$$\|r_k\|_{A^{-1}}^2 = \min_{q_k \in \tilde{\Pi}_k^1} \|q_k(A) r_0\|_{A^{-1}}^2. \tag{10.4.15}$$

Denote by $\{\lambda_i, v_i\}$, $i = 1, \ldots, n$, the eigenvalues and eigenvectors of A. Since A is symmetric we can assume that the eigenvectors are orthonormal. Expanding the right hand side as

$$r_0 = \sum_{i=1}^{n} \gamma_i v_i, \tag{10.4.16}$$

we have for *any* $q_k \in \tilde{\Pi}_k^1$

$$\|r_k\|_{A^{-1}}^2 \le \|q_k(A) r_0\|_{A^{-1}}^2 = r_0^T q_k(A)^T A^{-1} q_k(A) r_0 = \sum_{i=1}^{n} \gamma_i^2 \lambda_i^{-1} q_k(\lambda_i)^2.$$

In particular, taking

$$q_n(\lambda) = \Big(1 - \frac{\lambda}{\lambda_1}\Big)\Big(1 - \frac{\lambda}{\lambda_2}\Big) \cdots \Big(1 - \frac{\lambda}{\lambda_n}\Big), \tag{10.4.17}$$

we get $\|r_n\|_{A^{-1}} = 0$. This is an alternative proof that the CG method terminates after at most n steps in exact arithmetic.

If the eigenvalues of A are distinct then q_n in (10.4.17) is the minimal polynomial of A (see Section 10.1.2). If A only has p distinct eigenvalues then the minimal polynomial is of degree p and CG converges in at most p steps for any vector r_0. Hence, CG is particularly effective when A has low rank! More generally, if the grade of r_0 with respect to A equals m then only m steps are needed to obtain the exact solution. This will be the case if, e.g., in the expansion (10.4.16) $\gamma_i \neq 0$ only for m different values of i.

We stress that the finite termination property of the CG method shown above is only valid in exact arithmetic. In practical applications we want to obtain a good approximate solution x_k in far less than n iterations. We now use the optimality property (10.4.16) to derive an upper bound for the rate of convergence of the CG method considered as an iterative method. Let the set S contain all the eigenvalues of A and assume that for some $\tilde{q}_k \in \tilde{\Pi}_k^1$ we have

$$\max_{\lambda \in S} |\tilde{q}_k(\lambda)| \le M_k.$$

Then it follows that

$$\|r_k\|_{A^{-1}}^2 \le M_k^2 \sum_{i=1}^{n} \gamma_i^2 \lambda_i^{-1} = M_k^2 \|r_0\|_{A^{-1}}^2$$

or

$$\|x - x_k\|_A \leq M_k \|x - x_0\|_A. \tag{10.4.18}$$

We now select a set S on the basis of some assumption regarding the eigenvalue distribution of A and seek a polynomial $\tilde{q}_k \in \tilde{\Pi}_k^1$ such that $M_k = \max_{\lambda \in S} |\tilde{q}_k(\lambda)|$ is small.

A simple choice is to take $S = [\lambda_1, \lambda_n]$ and seek the polynomial $\tilde{q}_k \in \tilde{\Pi}_k^1$ which minimizes $\max_{\lambda_1 \leq \lambda \leq \lambda_n} |q_k(\lambda)|$. The solution to this problem is known to be a shifted and scaled Chebyshev polynomial of degree k, see the analysis in Section **??**. It follows that

$$\|x - x_k\|_A < 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \|x - x_0\|_A. \tag{10.4.19}$$

where $\kappa = \lambda_n(A)/\lambda_1(A)$. This estimate is the same as (10.2.19) for Chebyshev semi-iteration.

We note that the convergence of the conjugate residual method can be analyzed using a similar technique.

Example 10.4.1. For the model problem in Section 10.1.2 the extreme eigenvalues of $\frac{1}{4}A$ are $\lambda_{max} = 1 + \cos \pi h$, $\lambda_{min} = 1 - \cos \pi h$. It follows that

$$\kappa = \frac{1 + \cos \pi h}{1 - \cos \pi h} \approx \frac{1}{\sin^2 \pi h/2} \approx \frac{4}{(\pi h)^2}.$$

For $h = 1/100$ the number of iterations needed to reduce the initial error by a factor of 10^{-3} is then bounded by

$$k \approx \frac{1}{2} \log 2 \cdot 10^3 \sqrt{\kappa} \approx 242.$$

This is about the same number of iterations as needed with SOR using ω_{opt} to reduce the L_2-norm by the same factor. However, the conjugate gradient method is more general in that it does not require the matrix A to have "property A".

The error estimate above tends to be pessimistic asymptotically. One often observes, in practice, a *superlinear convergence* for the conjugate gradient method. This can be theoretically explained for the case when there are gaps in the spectrum of A. Then, as the iterations proceeds, the effect of the smallest and largest eigenvalues of A are eliminated and the convergence then behaves according to a smaller "effective" condition number. This behavior, called *superlinear convergence*, is in contrast to the Chebyshev semi-iterative method, which only takes the extreme eigenvalues of the spectrum into account and for which the error estimate in Section **??** tends to be sharp asymptotically.

We have seen that, in exact arithmetic, the conjugate gradient algorithm will produce the exact solution to a linear system $Ax = b$ in at most n steps. However, in the presence of rounding errors, the orthogonality relations in Theorem 10.3.4 will no longer be satisfied exactly. Indeed, orthogonality between residuals r_i and r_j, for $|i-j|$ is large, will usually be completely lost. Because of this, the finite termination

property does not hold in practice. Its main use is instead as an *iterative method for solving large, sparse, well-conditioned linear systems*, using *far fewer than n iterations*.

The behavior of the conjugate gradient algorithm in finite precision is much more complex than in exact arithmetic. It has been observed that the bound (10.4.19) still holds to good approximation in finite precision. On the other hand a good approximate solution may not be obtained after n iterations, even though a large drop in the error sometimes occurs after step n. It has been observed that the conjugate gradient algorithm in finite precision behaves like the exact algorithm applied to a larger linear system $\hat{A}\hat{x} = \hat{b}$, where the matrix $\hat{A}$ has many eigenvalues distributed in tiny intervals about the eigenvalues of A. This means that $\kappa(\hat{A}) \approx \kappa(A)$, which explains why the bound (10.4.19) still applies. It can also be shown that even in finite precision $\|r_k\|_2 \to 0$, where r_k is the recursively computed residual in the algorithm. (Note that the norm of true residual $\|b - Ax_k\|_2$ cannot be expected to approach zero.) This means that a termination criterion $\|r_k\|_2 \leq \epsilon$ will eventually always be satisfied even if $\epsilon \approx u$, where u is the machine precision.

Table 10.4.1. *Maximum error for Example 10.4.2 using Chebyshev iteration with optimal parameters and the conjugate gradient algorithm.*

Iteration	Chebyshev	Conj. gradient
1	$1.6 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$
2	$7.1 \cdot 10^{-4}$	$6.5 \cdot 10^{-4}$
3	$1.1 \cdot 10^{-5}$	$1.0 \cdot 10^{-5}$
4	$2.7 \cdot 10^{-7}$	$1.0 \cdot 10^{-7}$
5	$4.3 \cdot 10^{-9}$	$8.1 \cdot 10^{-10}$
6	$1.2 \cdot 10^{-10}$	$5.7 \cdot 10^{-12}$

Example 10.4.2. Consider the elliptic equation

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + p(x,y)u = f(x,y), \qquad p = \frac{6(x^2+y^2)}{1 + \frac{1}{2}(x^4+y^4)},$$

$0 < x, y < 1$, and let the boundary conditions be determined so that

$$u(x,y) = 2\Big((x-1/2)^2 + (y-1/2)^2\Big).$$

The Laplacian operator is approximated with 32 mesh points in each direction. In Table 10.4.3 we compare the maximum error using Chebyshev iteration with optimal parameters and the conjugate gradient algorithm. An initial estimate identically equal to zero is used.

It is seen that Algorithm 10.4.1 yields a smaller error without the need to estimate the parameters.

10.4.4 Symmetric Indefinite Systems

For symmetric positive definite matrices A the conjugate gradient method computes iterates x_k that satisfy the minimization property

$$\min_{x \in S_k} \|\hat{x} - x\|_A, \qquad S_k = x_0 + \mathcal{K}_k(r_0, A).$$

In case A is symmetric but indefinite $\|\cdot\|_A$ is no longer a norm. Hence the standard conjugate gradient method may break down. This is also true for the conjugate residual method.

A Krylov subspace method for symmetric indefinite systems was given by Paige and Saunders [22, 1975]. Using the Lanczos basis V_k they seek approximations $x_k = V_k y_k \in \mathcal{K}_k(b, A)$, which are stationary values of $\|\hat{x} - x_k\|_A^2$. These are given by the Galerkin condition

$$V_k^T(b - AV_k y_k) = 0.$$

This leads again to the tridiagonal system (10.4.14). However, when A is indefinite, although the Lanczos process is still well defined, the Cholesky factorization of T_k may not exist. Moreover, it may happen that T_k is singular at certain steps, and then y_k is not defined.

If the Lanczos process stops for some $k \leq n$ then $AV_k = V_k T_k$. It follows that the eigenvalues of T_k are a subset of the eigenvalues of A, and thus if A is nonsingular so is T_k. Hence the problem with a singular T_k can only occur at intermediate steps.

To solve the tridiagonal system (10.4.14) Paige and Saunders suggest computing the LQ factorization

$$T_k = \bar{L}_k Q_k, \qquad Q_k^T Q_k = I,$$

where $\bar{L}_k$ is lower triangular and Q_k orthogonal. Such a factorization always exists and can be computed by multiplying T_k with a sequence of plane rotations from the right

$$T_k G_{12} \cdots G_{k-1,k} = \bar{L}_k = \begin{pmatrix} \gamma_1 & & & & \\ \delta_2 & \gamma_2 & & & \\ \epsilon_3 & \delta_3 & \gamma_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \epsilon_k & \delta_k & \bar{\gamma}_k \end{pmatrix}.$$

The rotation $G_{k-1,k}$ is defined by elements c_{k-1} and s_{k-1}. The bar on the element $\bar{\gamma}_k$ is used to indicate that $\bar{L}_k$ differs from L_k, the $k \times k$ leading part of $\bar{L}_{k+1}$, in the (k, k) element only. In the next step the elements in $G_{k,k+1}$ are given by

$$\gamma_k = (\bar{\gamma}_k^2 + \beta_{k+1}^2)^{1/2}, \qquad c_k = \bar{\gamma}_k / \gamma_k, \qquad s_k = \beta_{k+1} / \gamma_k.$$

Since the solution y_k of $T_k y_k = \beta_1 e_1$ will change fully with each increase in k we write

$$x_k = V_k y_k = (V_k Q_k^T)\bar{z}_k = \bar{W}_k \bar{z}_k,$$

and let

$$\bar{W}_k = (w_1, \ldots, w_{k-1}, \bar{w}_k),$$
$$\bar{z}_k = (\zeta_1, \ldots, \zeta_{k-1}, \bar{\zeta}_k) = Q_k y_k.$$

Here quantities without bars will be unchanged when k increases, and $\bar{W}_k$ can be updated with $\bar{T}_k$. The system (10.4.14) now becomes

$$\bar{L}_k \bar{z}_k = \beta_1 e_1, \qquad x_k^c = \bar{W}_k \bar{z}_k.$$

This formulation allows the v_i and w_i to be formed and discarded one by one.

In implementing the algorithm we should note that x_k^c need not be updated at each step, and that if $\bar{\gamma}_k = 0$, then $\bar{z}_k$ is not defined. Instead we update

$$x_k^L = W_k z_k = x_{k-1}^L + \zeta_k w_k,$$

where L_k is used rather than $\bar{L}_k$. We can then always obtain x_{k+1}^c when needed from

$$x_{k+1}^c = x_k^L + \bar{\zeta}_{k+1} \bar{w}_{k+1}.$$

This defines the SYMMLQ algorithm. In theory the algorithm will stop with $\beta_{k+1} = 0$ and then $x_k^c = x_k^L = x$. In practice it has been observed that β_{k+1} will rarely be small and some other stopping criterion based on the size of the residual must be used.

Paige and Saunders also derived an algorithm called MINRES, which is based on minimizing the Euclidian norm of the residual r_k. It should be noted that MINRES suffers more from poorly conditioned systems than SYMMLQ does.

Review Questions

1. Define the Krylov space $\mathcal{K}_j(b, A)$. Show that it is invariant under (i) scaling τA. (ii) translation $A - sI$. How is it affected by an orthogonal similarity transformation $\Lambda = V^T A V$, $c = V^T b$?
2. What minimization problems are solved by the conjugate gradient method? How can this property be used to derive an upper bound for the rate of convergence of the conjugate gradient method.
3. Let the symmetric matrix A have eigenvalues λ_i and orthonormal eigenvectors v_i, $i = 1, \ldots, n$. If only $d < n$ eigenvalues are distinct, what is the maximum dimension of the Krylov space $\mathcal{K}_j(b, A)$?

Problems

1. Let λ_i, v_i be an eigenvalue and eigenvector of the symmetric matrix A.

   (a) Show that if $v_i \perp b$, then also $v_i \perp \mathcal{K}_j(b, A)$, for all $j > 1$.

   (b) Show that if b is orthogonal against p eigenvectors, then the maximum dimension of $\mathcal{K}_j(b, A)$ is at most $n-p$. Deduce that the the conjugate gradient method converges in at most $n - p$ iterations.

2. Let $A = I + BB^T \in \mathbf{R}^{n\times n}$, where B is of rank p. In exact arithmetic, how many iterations are at most needed to solve a system $Ax = b$ with the conjugate gradient method?

3. Write down explicitly the conjugate residual method. Show that in this algorithm one needs to store the vectors x, r, Ar, p and Ap.

4. SYMMLQ is based on solving the tridiagonal system (10.4.12) using an LQ factorization of T_k. Derive an alternative algorithm, which solves this system with Gaussian elimination with partial pivoting.

10.5 Nonsymmetric Problems

An ideal conjugate gradient-like method for nonsymmetric systems would be characterized by one of the properties (10.4.6) or (10.4.13). We would also like to be able to base the implementation on a short vector recursion. Unfortunately, it turns out that such an ideal method essentially can only exist for matrices of very special form. In particular, a two term recursion like in the CG method is only possible in case A either has a minimal polynomial of degree ≤ 1, or is Hermitian, or is of the form

$$A = e^{i\theta}(B + \rho I), \qquad B = -B^H,$$

where θ and ρ are real. Hence the class essentially consists of shifted and rotated Hermitian matrices.

There are several possibilities for generalizing the conjugate gradient method to nonsymmetric systems. One simple approach is to apply the conjugate gradient method to the symmetrize system of normal equations. Since these usually have much higher spectral condition convergence of these methods can be very slow.

We can maintain the three-term relation by

10.5.1 Arnoldi's Method and GMRES

A serious drawback with using methods based on the normal equations is that they often converge very slowly, which is related to the fact that the singular values of A^TA are the square of the singular values of A. There also are applications where it is not possible to compute matrix-vector products A^Tx—note that A may only exist as subroutine for computing Ax.

We now consider a method for solving a general nonsymmetric system $Ax = b$ based on the Arnoldi process (see Section 9.9.6) with the starting vector

$$v_1 = r_0/\beta_1, \quad r_0 = b - Ax_0, \quad \beta_1 = \|r_0\|_2,$$

In the following implementation of the Arnoldi process we perform the orthogonalization by the modified Gram-Schmidt method.

Algorithm 10.5.1 The Arnoldi Process.

$$\beta_1 = \|r_0\|_2; \quad v_1 = r_0/\beta_1;$$

for $k = 1 : n$ **do**
  $z_k = Av_k$;
  for $i = 1 : k$ **do**
    $h_{ik} = z_k^T v_i$;
    $z_k = z_k - h_{ik} v_i$;
  end
  $h_{k+1,k} = \|z_k\|_2$;
  if $|h_{k+1,k}| < \epsilon$, *break* **end**
  $v_{k+1} = z_k / h_{k+1,k}$;
end

In exact arithmetic the result after k steps is a matrix $V_k = (v_1, \ldots, v_k)$, that (in exact arithmetic) gives an orthogonal basis for the Krylov subspace

$$\mathcal{K}_k(r_0, A) = \text{span}\,(r_0, Ar_0, \ldots, A^{k-1} r_0),$$

and a related square Hessenberg matrix $H_k = (h_{ij}) \in \mathbf{R}^{k\times k}$. Further we have

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T = V_{k+1} \bar{H}_k, \tag{10.5.1}$$

where

$$\bar{H}_k = \begin{pmatrix} H_k \\ h_{k+1,k} e_k^T \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2k} \\ & \ddots & \ddots & \vdots \\ & & h_{k,k-1} & h_{kk} \\ & & & h_{k+1,k} \end{pmatrix} \in \mathbf{R}^{(k+1)\times k}. \tag{10.5.2}$$

We seek at step k an approximate solution of the form

$$x_k = x_0 + V_k y_k \in x_0 + \mathcal{K}_k(r_0, A), \tag{10.5.3}$$

There are two different ways to choose the approximation x_k. In the **full orthogonalization method (FOM)** x_k is determined by the Galerkin condition

$$r_k \perp \mathcal{K}_k(r_0, A), \quad r_k = b - Ax_k.$$

Using (10.5.1) the residual r_k can be expressed as

$$r_k = r_0 - AV_k y_k = \beta_1 v_1 - V_k H_k y_k - h_{k+1,k} v_{k+1} e_k^T . = V_{k+1}(\beta_1 e_1 - \bar{H}_k y_k). \tag{10.5.4}$$

Using this the Galerkin condition gives $V_k^T r_k = \beta_1 e_1 - H_k y_k = 0$. Hence $y_k = \beta_1 H_k^{-1} e_1$ is obtained as the solution of a linear system of Hessenberg form. (Note that H_k is nonsingular if $\mathcal{K}_k(r_0, A)$ has full rank).

In the **generalized minimum residual (GMRES) method** y_k is chosen so that $\|b - Ax_k\|_2$ is minimized. Notice that this ensures that $\|r_k\|_2$ is monotonically decreasing as the iteration proceeds. Since (in exact arithmetic) V_{k+1} has orthogonal columns, $\|r_k\|_2$ is minimized by taking y_k to be the solution of the least squares problem

$$\min_{y_k} \|\beta_1 e_1 - \bar{H}_k y_k\|_2. \tag{10.5.5}$$

The Arnoldi process breaks down at step k if and only if $A^k r_0 \in \mathcal{K}_k(r_0, A)$. Then z_k vanishes, $h_{k+1,k} = 0$ and $AV_k = V_k H_k$. Since $\operatorname{rank}(AV_k) = \operatorname{rank}(V_k) = k$ the matrix H_k is nonsingular. Then

$$r_k = V_k(\beta_1 e_1 - H_k y_k) = 0, \quad y_k = \beta_1 H_k^{-1} e_1,$$

and $x_k = x_0 + V_k y_k$ is the solution of $Ax = b$. This shows the important property (in exact arithmetic) that *GMRES does not break down before the exact solution is found.* It follows that GMRES terminates in at most n steps.

We now discuss the implementation of GMRES. To solve (10.5.5) we compute the QR factorization of the Hessenberg matrix $\bar{H}_k$. This can be done by using a sequence of k plane rotations. Let

$$Q_k^T(\bar{H}_k \; e_1) = \begin{pmatrix} R_k & d_k \\ 0 & \rho_k \end{pmatrix}, \quad Q_k^T = G_{k,k+1} G_{k-1,k} \cdots G_{12}, \tag{10.5.6}$$

where $G_{j+1,j}$ is chosen to zero the subdiagonal element $h_{j+1,j}$. Then the solution to (10.5.5) and its residual is given by

$$R_k y_k = \beta_1 d_k, \qquad \|r_k\|_2 = \beta_1 |\rho_k|. \tag{10.5.7}$$

The iterations can be stopped as soon as $|\rho_k|$ is maller than a prescribed tolerance.

Since $\bar{H}_{k-1}$ determines the first $k-1$ Givens rotations and $\bar{H}_k$ is obtained from $\bar{H}_{k-1}$ by adding the kth column, it is possible to save work by *updating the QR factorization* (10.5.6) at each step of the Arnoldi process. To derive the updating formulas for step $j = k$ we write

$$Q_k^T \bar{H}_k = G_{k,k+1} \begin{pmatrix} Q_{k-1}^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{H}_{k-1} & h_k \\ 0 & h_{k+1,k} \end{pmatrix} = \begin{pmatrix} R_{k-1} & c_{k-1} \\ 0 & \gamma_k \\ 0 & 0 \end{pmatrix},$$

We first apply the previous rotations to h_k giving

$$Q_{k-1}^T h_k = G_{k-1,k} \cdots G_{12} h_k = \begin{pmatrix} c_{k-1} \\ \delta_k \end{pmatrix}, \tag{10.5.8}$$

The rotation $G_{k,k+1}$ is determined by

$$G_{k,k+1} \begin{pmatrix} \delta_k \\ h_{k+1,k} \end{pmatrix} = \begin{pmatrix} \gamma_k \\ 0 \end{pmatrix}. \tag{10.5.9}$$

and gives the last element in the kth column in R_k.

Proceeding similarly with the right hand side, we have

$$Q_k^T e_1 = G_{k,k+1} \begin{pmatrix} Q_{k-1}^T e_1 \\ 0 \end{pmatrix} = G_{k,k+1} \begin{pmatrix} d_{k-1} \\ \rho_{k-1} \\ 0 \end{pmatrix} = \begin{pmatrix} d_{k-1} \\ \tau_k \\ \rho_k \end{pmatrix} \equiv \begin{pmatrix} d_k \\ \rho_k \end{pmatrix}. \quad (10.5.10)$$

(Note that the different dimensions of the unit vectors e_1 above is not indicated in the notation.) The first $k-1$ elements in $Q_k^T e_1$ are not changed.

The approximate solution can be obtained from $x_k = x_0 + V_k y_k$. Note that the whole vector y_k differs from y_{k-1} and therefore all the vectors $v_1, \ldots, v_k$ needs to be saved. Since $\|r_k\|_2 = |\rho_k|$ is available without forming x_k, this expensive operation can be delayed until until GMRES has converged, i.e., when ρ_k is small enough.

Alternatively, an updating formula for x_k can be derived using the same trick as in LSQR. Set $W_k R_k = V_k$, which can be written

$$(W_{k-1},\ w_k) \begin{pmatrix} R_{k-1} & c_{k-1} \\ 0 & \gamma_k \end{pmatrix} = (V_{k-1},\ v_k).$$

Equating the first block columns gives $W_{k-1}R_{k-1} = V_{k-1}$, which shows that the first $k-1$ columns of W_k equal W_{k-1}. Equating the last columns and solving for w_k we get

$$w_k = (v_k - W_{k-1} r_{k-1})/\gamma_k \quad (10.5.11)$$

Then from (10.6.26) $x_k = x_{k-1} + \beta_1 \tau_k w_k$. Note that if this formula is used we only need the last column of the matrix R_k. (We now need to save W_k but not R_k.)

The steps in the resulting GMRES algorithm can now be summarized as follows:

1. Obtain last column of $\bar{H}_k$ from the Arnoldi process and apply old rotations $g_k = G_{k-1,k} \cdots G_{12} h_k$.
2. Determine rotation $G_{k,k+1}$ and new column in R_k, i.e., c_{k-1} and γ_k according to (10.5.9). This also determines τ_k and $|\rho_k| = \|r_k\|_2$.
3. If x_{k-1} is recursively updated, then compute w_k using (10.5.10) and x_k from (10.6.26).

Suppose that the matrix A is diagonalizable,

$$A = X \Lambda X^{-1}, \quad \Lambda = \mathrm{diag}(\lambda_i).$$

Then, using the property that the GMRES approximations minimize the Euclidian norm of the residual $r_k = b - Ax_k$ in the Krylov subspace $\mathcal{K}_k(r_0, A)$, it can be shown that

$$\frac{||r_k||_2}{||r_0||_2} \leq \kappa_2(X) \min_{q_k} \max_{i=1,2,\ldots,n} |q_k(\lambda_i)|, \quad (10.5.12)$$

where q_k is a polynomial of degree $\leq k$ and $q_k(0) = 1$. The proof is similar to the convergence proof for the conjugate gradient method in Section 10.4.3. This results shows that if A has $p \leq n$ distinct eigenvalues then, as for CG in the symmetric

case, GMRES converges in at most p steps. If the spectrum is clustered in p clusters of sufficiently small diameters, then we can also expect GMRES to provide accurate approximations after about p iterations.

Because of the factor $\kappa_2(X)$ in (10.5.12) an upper bound for the rate of convergence can no longer be deduced from the spectrum $\{\lambda_i\}$ of A alone. In the special case that A is normal we have $\kappa_2(X) = 1$, and the convergence is related to the complex approximation problem

$$\min_{q_k} \max_{i=1,2,\ldots,n} |q_k(\lambda_i)|, \quad q_k(0) = 1.$$

Because complex approximation problems are harder than real ones, no simple results are available even for this special case.

In practice it is often observed that GMRES (like the CG method) has a so-called superlinera convergence. By this we mean that the rate of convergence improves as the iteration proceeds. It has been proved that this is related to the convergence of Ritz values to exterior eigenvalues of A. When this happens GMRES converges from then on as fast as for a related system in which these eigenvalues and their eigenvector components are missing.

The memory requirement of GMRES increases linearly with the number of steps k and the cost for orthogonalizing the vector Av_k is proportional to k^2. In practice the number of steps taken by GMRES must therefore often be limited. by **restarting** GMRES after each m iterations, where in practice typically m is between 10 and 30. We denote the corresponding algorithm GMRES(m). GMRES(m) cannot break down (in exact arithmetic) before the true solution has been produced, but for $m < n$ GMRES may never converge.

Since restarting destroys the accumulated information about the eigenvalues of A the superlinear convergence is usually lost. This loss can be compensated for by extracting form the computed Arnoldi factorization an approximate invariant subspace of A associated with the small eigenvalues. This is then used to precondition the restarted iteration.

If GMRES is applied to a real symmetric indefinite system, it can be implemented with a three-term recurrence, which avoids the necessity to store all basis vectors v_j. This leads to the method MINRES by Paige and Saunders mentioned in Section 10.5.1.

10.5.2 Lanczos Bi-Orthogonalization

The GMRES method is related to the reduction of a nonsymmetric matrix $A \in \mathbf{R}^{n\times n}$ to Hessenberg form by an orthogonal similarity $H = Q^T AQ$. It gives up the short recurrences of the CG method. Another possible generalization proposed by Lanczos [17] is related to the reduction of to tridiagonal form by a general similarity transformation.

Hence the stability of this process cannot be guaranteed, and this reduction is in general not advisable.

Assume that A can be reduced to tridiagonal form

$$W^TAV = T_n = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \gamma_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_n \\ & & & \gamma_n & \alpha_n \end{pmatrix},$$

where $V = (v_1, \ldots, v_n)$ and $W = (w_1, \ldots, w)$, are nonsingular and $W^TV = I$. The two vector sequences $\{v_1, \ldots, v_n\}$ and $\{w_1, \ldots, w\}$, then are **bi-orthogonal**, i.e.,

$$w_i^T v_j = \begin{cases} 1, & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{10.5.13}$$

Comparing columns in $AV = VT$ and $A^TW = WT^T$ we find (with $v_0 = w_0 = 0$) the recurrence relations

$$\gamma_{k+1}v_{k+1} = \tilde{v}_{k+1} = (A - \alpha_k I)v_k - \beta_k v_{k-1}, \tag{10.5.14}$$
$$\beta_{k+1}w_{k+1} = \tilde{w}_{k+1} = (A^T - \alpha_k I)w_k - \gamma_k w_{k-1}, \tag{10.5.15}$$

Multiplying equation (10.5.14) by w_k^T, and using the bi-orthogonality we have

$$\alpha_k = w_k^T A v_k.$$

To satisfy the bi-orthogonality relation (10.5.15) for $i = j = k + 1$ it suffices to choose γ_{k+1} and β_{k+1} so that.

$$\gamma_{k+1}\beta_{k+1} = \tilde{w}_{k+1}^T \tilde{v}_{k+1}.$$

Hence there is some freedom in choosing these scale factors.

If we denote

$$V_k = (v_1, \ldots, v_k), \qquad W_k = (w_1, \ldots, w_k),$$

then we have $W_k^T AV_k = T_k$, and the recurrences in this process can be written in matrix form as

$$AV_k = V_k T_k + \gamma_{k+1} v_{k+1} e_k^T, \tag{10.5.16}$$
$$A^T W_k = W_k T_k^T + \beta_{k+1} w_{k+1} e_k^T. \tag{10.5.17}$$

By construction these vector sequences form basis vectors for the two Krylov spaces

$$\mathcal{R}(V_k) = \mathcal{K}_k(v_1, A), \qquad \mathcal{R}(W_k) = \mathcal{K}_k(w_1, A^T). \tag{10.5.18}$$

We summarize the algorithm for generating the two sequences of vectors $v_1, v_2, \ldots$ and $w_1, w_2, \ldots$:

Algorithm 10.5.2 The Lanczos Bi-orthogonalization Process.

Let v_1 and w_1 be two vectors such that $w_1^T v_1 = 1$. The following algorithm computes in exact arithmetic after k steps a symmetric tridiagonal matrix $T_k = \text{trid}\,(\gamma_j, \alpha_j, \beta_{j+1})$ and two matrices W_k and V_k with bi-orthogonal columns spanning the Krylov subspaces $\mathcal{K}_k(v_1, A)$ and $\mathcal{K}_k(w_1, A^T)$:

$$
\begin{aligned}
&w_0 = v_0 = 0;\\
&\beta_1 = \gamma_1 = 0;\\
&\textbf{for } j = 1, 2, \ldots\\
&\quad \alpha_j = w_j^T A v_j;\\
&\quad v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1};\\
&\quad w_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1};\\
&\quad \delta_{j+1} = |w_{j+1}^T v_{j+1}|^{1/2};\\
&\quad \textbf{if } \delta_{j+1} = 0 \textbf{ then } exit;\\
&\quad \beta_{j+1} = (w_{j+1}^T v_{j+1})/\delta_{j+1};\\
&\quad v_{j+1} = v_{j+1}/\delta_{j+1};\\
&\quad w_{j+1} = w_{j+1}/\beta_{j+1};\\
&\textbf{end}
\end{aligned}
$$

Note that if $A = A^T$, $w_1 = v_1$, and we take $\beta_k = \gamma_k$, then the two sequences generated will be identical. The process then is equivalent to the symmetric Lanczos process.

There are two cases when the above algorithm breaks down. The first occurs when either $\tilde{v}_{k+1}$ or $\tilde{w}_{k+1}$ (or both) is null. In this case it follows from (10.5.16)–(10.5.17) that an invariant subspace has been found; if $v_{k+1} = 0$, then $AV_k = V_k T_k$ and $\mathcal{R}(V_k)$ is an A-invariant subspace. If $w_{k+1} = 0$, then $A^T W_k = W_k T_k^T$ and $\mathcal{R}(W_k)$ is an A^T-invariant subspace. This is called regular termination. The second case, called serious breakdown, occurs when $\tilde{w}_k^T \tilde{v}_k = 0$, with neither $\tilde{v}_{k+1}$ nor $\tilde{w}_{k+1}$ null.

10.5.3 Bi-Conjugate Gradient Method and QMR

We now consider the use of the nonsymmetric Lanczos process for solving a linear system $Ax = b$. Let x_0 be an initial approximation. Take $\beta_1 v_1 = r_0$, $\beta_1 = \|r_0\|_2$, and $w_1 = v_1$. We seek an approximate solution x_k such that

$$x_k - x_0 = V_k y_k \in \mathcal{K}_k(r_0, A).$$

For the residual we then have

$$r_k = b - A x_k = \beta_1 v_1 - A V_k y_k,$$

Here y_k is determined so that the Galerkin condition $r_k \perp \mathcal{K}_k(w_1, A^T)$ is satisfied, or equivalently $W_k^T r_k = 0$. Using (10.5.16) and the bi-orthogonality conditions

$W_k^T V_k = 0$ this gives

$$W_k^T(\beta_1 v_1 - AV_k y_k) = \beta_1 e_1 - T_k y_k = 0. \tag{10.5.19}$$

Hence, if the matrix T_k is nonsingular x_k is determined by solving the tridiagonal system $T_k y_k = \beta_1 e_1$ and setting $x_k = x_0 + V_k y_k$.

If A is symmetric, this method becomes the SYMMLQ method, see Sec. 10.5.1. We remark again that in the nonsymmetric case this method can break down without producing a good approximate solution to $Ax = b$. In case of a serious breakdown, it is necessary to restart from the beginning with a new starting vector r_0. As in SYMMLQ the matrix T_k may be singular for some k and this is an additional cause for breakdown.

The Lanczos bi-orthogonalization algorithm is the basis for several iterative methods for nonsymmetric systems. The method can be written in a form more like the conjugate gradient algorithm, which is called the **bi-conjugate gradient** or Bi-CG method. The algorithm was first proposed by Lannczos [18] and later in conjugate gradient form by Fletcher [7]. The Bi-CG algorithm can be derived from Algorithm 10.5.4 in exactly the same way as the CG method was derived from the Lanczos algorithm. The algorithm solves not only the original system $Ax = b$ but also a dual linear system $A^T\tilde{x} = \tilde{b}$, although the dual system usually is ignored in the derivation of the algorithm.

To derive th Bi-CG algorithm from the Lanczos bi-orthogonalization we introduce the LU decomposition

$$T_k = L_k U_k,$$

and write

$$x_k = x_0 + V_k T_k^{-1}(\beta e_1) = x_0 + P_k L_k^{-1}(\beta e_1),$$

where $P_k = V_k U_k^{-1}$. Notice that x_k can be obtained by updating x_{k-1} as in the CG method.

Define similarly the matrix $\tilde{P}_k = W_k L_k^{-T}$. Then the columns of P_k and $\tilde{P}_k$ are A-conjugate, since

$$\tilde{P}_k^T A P_k = L_k^{-1} W_k^T A V_k U_k^{-1} = L_k^{-1} T_k U_k^{-1} = I.$$

Algorithm 10.5.3 Bi-conjugate Gradient Algorithm
Set $r_0 = b - Ax_0$ and choose $\tilde{r}_0$ so that $(r_=0, \tilde{r}_0) \neq 0$.

$p_0 = r_0$; $\tilde{p}_0 = \tilde{r}_0$;
$\rho_0 = (\tilde{r}_0, r_0)$;
for $j = 0, 1, 2, \ldots$
  $v_j = Ap_j$; $\alpha_j = \rho_j/(\tilde{p}_j, v_j)$;
  $x_{j+1} = x_j + \alpha_j p_j$;
  $r_{j+1} = r_j - \alpha_j v_j$;
  $\tilde{r}_{j+1} = \tilde{r}_j - \alpha_j (A^T \tilde{p}_j)$;

$\rho_{j+1} = (\tilde{r}_{j+1}, r_{j+1});$
$\beta_j = \rho_{j+1}/\rho_j;$
$p_{j+1} = r_{j+1} + \beta_j p_j;$
$\tilde{p}_{j+1} = \tilde{r}_{j+1} + \beta_j \tilde{p}_j;$
end

The vectors r_j and $\tilde{r}_j$ are in the same direction as v_{j+1} and w_{j+1}, respectively. Hence they form a biorthogonal sequence. Note that Bi-CG has the computational advantage over CGNE that the most time-consuming operationss Ap_j and $A^T\tilde{p}_j$ can be carried out in parallel.

One can encounter convergence problems with Bi-CG, sinve for general matrices the bilinear form

$$[x, y] = (\psi(A^T)x, \psi(A)y)$$

used to define bi-orthogonality, does not define an inner product. Therefore if $\tilde{r}_0$ is chosen unfavorably, it may occur that ρ_j or $(\tilde{p}_j, v_j)$ is zero (or very small), without convergence having taken place.

Nothing is minmized in the Bi-CG and related methods, and for a general unsymmetric matrix A there is no gurantee that the algorithm will not break down or be unstable. On the contrary, it has been observed that sometimes convergence can be as fast as for GMRES. However, the convergence behavior can be very irregular, and as remarked above, breakdown occurs. Sometimes, breakdown can be avoided by a restart at the iteration step immediately before the breakdown step.

A related method called the **Quasi-Minimal Residual (QMR) method** can be developed as follows. After k steps of the nonsymmetric Lanczos process we have from the relation (10.5.16) that

$$AV_k = V_{k+1}\hat{T}_k, \quad \hat{T}_k = \begin{pmatrix} T_k \\ \gamma_{k+1}e_k^T \end{pmatrix},$$

where $\hat{T}_k$ is an $(k+1) \times k$ tridiagonal matrix. We can now proceed as was done in developing GMRES. If we take $v_1 = \beta r_0$, the the residual associated with with an approximate solution of the form $x_k = x_0 + V_k y$ is given by

$$\begin{aligned} b - Ax_k &= b - A(x_0 + V_k y) = r_0 - AV_k y \\ &= \beta v_1 - V_{k+1}\hat{T}_k y = V_{k+1}(\beta e_1 - \hat{T}_k y). \end{aligned} \tag{10.5.20}$$

Hence the norm of the residual vector is

$$\|b - Ax_k\|_2 = \|V_{k+1}(\beta e_1 - \hat{T}_k y)\|_2.$$

If the matrix V_{k+1} had orthonormal columns then the residual norm would become $\|(\beta e_1 - \hat{T}_k y)\|_2$, as in GMRES, and a least squares solution in the Krylov subspace could be obtained by solving

$$\min_y \|\beta e_1 - \hat{T}_k y\|_2.$$

for y_k and taking $x_k = x_0 + V_k y_k$.

Recent surveys on progress in iterative methods for non-symmetric systems are given by Freund, Golub and Nachtigal [8, 1991] and Golub and van der Vorst [11]. There is a huge variety of methods to choose from. Unfortunately in many practical situations it is not clear what method to select. In general there is no best method. In [21] examples are given which show that, depending on the linear system to be solved, each method can be clear winner or clear loser! Hence insight into the characteristics of the linear system is needed in order to discriminate between methods. This is different from the symmetric case, where the rate of convergence can be deduced from the spectral properties of the matrix alone.

10.5.4 Transpose-free Methods

A disadvantage of the methods previously described for solving non-symmetric linear systems is that they require subroutines for the calculation of both Ax and $A^T y$ for arbitrary vectors x and y. If the data structure favors the calculation of Ax then it is often less favorable for the calculation of $A^T y$. Moreover, for some problems deriving from ordinary differential equations the rows of A arise natuarrly from a finite difference approximation and the matrix product Ax may be much more easily computed than $A^T y$. These consideration has lead to the development of "transpose-free" methods

The first of the transpose-free iterative methods **Bi-CGS**, due to Sonneveld [26], is a modification of the Bi-CG algorithm.. Here CGS stands for "conjugate gradient squared". The key observation behind this algorithm is the following property of the vectors generated in the Bi-CG algorithm. Taking into account that $p_0 = r_0$, it is easily showed that there are polynomials $\phi_j(x)$ and $psi_j(x)$ of degree such that for $j = 1, 2, 3, \ldots,$

$$\begin{aligned} r_j &= \phi_j(A)\, r_0, \qquad \tilde{r}_j = \phi_j(A^T)\, \tilde{r}_0, \\ p_j &= \psi_j(A)\, r_0, \qquad \tilde{p}_j = \psi_j(A^T)\, \tilde{r}_0. \end{aligned}$$

That is r_j and $\tilde{r}_j$ are obtained by premultiplication by *the same polynomial* $\phi(t)$ in A and A^T, respectively. The same is true for p_j and $\tilde{p}_j$ for the polynomial $\psi(t)$. Using the fact that the polynomial of a transpose matrix is the transpose of the polynomial, it follows that the quantities needed in the Bi-CG algorithm can be expressed as

$$(\tilde{r}_j, r_j) = (\tilde{r}_0, \phi_j^2(A)\, r_0), \qquad (\tilde{p}_j, Ap_j) = (\tilde{p}_0, \psi_j^2(A)\, r_0).$$

Therefore, if we somehow could generate the vectors $\phi_j(A)]^2 r_0$ and $\psi_j(A)]^2 p_0$ directly, then no products with A^T would be required. To achieve this we note that from the Bi-CG algorithm we have the relations, $\phi_0(A) = \psi_0(A) = I$,

$$\phi_{j+1}(A) = \phi_j(A) - \alpha_j A\, \psi_j(A), \tag{10.5.21}$$

$$\psi_{j+1}(A) = \phi_{j+1}(A) + \beta_j \psi_j(A), \tag{10.5.22}$$

Squaring these relations we obtain

$$\phi_{j+1}^2 = \phi_j^2 - 2\alpha_j A\, \phi_j \psi_j + \alpha_j^2 A^2 \psi_j^2,$$

$$\psi_{j+1}^2 = \phi_{j+1}^2 + 2\beta_j\phi_{j+1}\psi_j + \beta_j^2\psi_j^2.$$

where we have omitted the argument A. For the first cross product term we have using (10.5.22)

$$\phi_j\psi_j = \phi_j(\phi_j + \beta_{j-1}\psi_{j-1}) = \phi_j^2 + \beta_{j-1}\phi_j\psi_{j-1}.$$

From this and (10.5.21) we get for the other cross product term

$$\phi_{j+1}\psi_j = (\phi_j - \alpha_j A\,\psi_j)\psi_j = \phi_j\psi_j - \alpha_j A\,\psi_j^2 = \phi_j^2 + \beta_{j-1}\phi_j\psi_{j-1} - \alpha_j A\,\psi_j^2.$$

Summarizing, we now have the three reccurence relations, which are the basis of the Bi-CGS algorithm:

$$\begin{aligned}
\phi_{j+1}^2 &= \phi_j^2 - \alpha_j A\,(2\phi_j^2 + 2\beta_{j-1}\phi_j\psi_{j-1} - \alpha_j A\,\psi_j^2),\\
\phi_{j+1}\psi_j &= \phi_j^2 + \beta_{j-1}\phi_j\psi_{j-1} - \alpha_j A\,\psi_j^2\\
\psi_{j+1}^2 &= \phi_{j+1}^2 + 2\beta_j\phi_{j+1}\psi_j + \beta_j^2\psi_j^2.
\end{aligned}$$

If we now define

$$r_j = \phi_j^2(A)r_0, \qquad q_j = \phi_{j+1}(A)\psi_j(A)r_0, \qquad p_j = \psi_j^2(A)r_0. \tag{10.5.23}$$

we get

$$r_{j+1} = r_j - \alpha_j A\,(2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j A\,p_j), \tag{10.5.24}$$
$$q_j = r_j + \beta_{j-1}q_{j-1} - \alpha_j A\,p_j, \tag{10.5.25}$$
$$p_{j+1} = r_{j+1} + 2\beta_j q_j + \beta_j^2 p_j. \tag{10.5.26}$$

These recurrences can be simplified by introducing the auxiliary vectors

$$u_j = r_j + \beta_{j-1}q_{j-1}, \qquad d_j = u_j + q_j. \tag{10.5.27}$$

The resulting algorithm is given below.

Algorithm 10.5.4 Bi-CGS Algorithm
Set $r_0 = b - Ax_0$ and choose $\tilde{r}_0$ so that $(r_0, \tilde{r}_0) \neq 0$.

$p_0 = u_0 = r_0;\quad \rho_0 = (\tilde{r}_0, r_0);$
for $j = 0, 1, 2, \ldots$
  $v_j = Ap_j;\quad \alpha_j = \rho_j/(\tilde{r}_0, v_j);$
  $q_j = u_j - \alpha_j v_j;$
  $d_j = u_j + q_j;$
  $x_{j+1} = x_j + \alpha_j d_j;$
  $r_{j+1} = r_j - \alpha_j A\,d_j;$
  $\rho_{j+1} = (\tilde{r}_0, r_{j+1});$
  $\beta_j = \rho_{j+1}/\rho_j;$
  $u_{j+1} = r_{j+1} + \beta_j q_j;$
  $p_{j+1} = u_{j+1} + \beta_j(q_j + \beta_j p_j);$
end

There are now two matrix-vector multiplications with A in each step. When Bi-CG converges well we can expect Bi-CGS to converge about twice as fast.

Although the Bi-CGS algorithm often is competetive with other methods such as GMRES, a weak point of Bi-CGS is that the residual norms may behave very erratically, in particular when the iteration is started close to the solution. For example, although the norm of of the vector $\psi_j(A)r_0$ is small it may happen that $\|\psi_j^2(A)r_0\|$ is much bigger than $\|r_0\|$. This may even lead to such severe cancellation that the accuracy of the computed solution is spoilt.

This problem motivated the developemnt a stabilized version called Bi-CGSTAB by van der Vorst [27]), which is more smoothly converging. Instead of computing the residuals $\psi_j^2(A)r_0$, this algorithm uses

$$r_j = \chi_j(A)\psi_j(A)r_0, \quad \chi_j(t) = (1-\omega_1 t)(1-\omega_2 t)\cdots(1-\omega_j t), \tag{10.5.28}$$

where the constants ω_j are determined so that $\|r_j\|_2$ is minmized as a function of ω_j.

From the ortogonality property $(\psi_i(A)r_0, \chi_j(A)r_0) = 0$, for $j < i$, it follows that Bi-CGSTAB is a finite method, i.e. in exact arithmetic it will convergen in at most n steps.

Algorithm 10.5.5 Bi-CGSTAB Algorithm
Let x_0 be an initial guess, $r_0 = b - Ax_0$ and choose $\tilde{r}_0$ so that $(\tilde{r}_0, r_0) \neq 0$.

$p_0 = u_0 = r_0; \quad \rho_0 = (\tilde{r}_0, r_0);$
for $j = 0, 1, 2, \ldots$
  $v_j = Ap_j; \quad \alpha_j = \rho_j/(\tilde{r}_0, v_j);$
  $s_j = r_j - \alpha_j v_j;$
  $t_j = Ap_j;$
  $\omega_j = (t_j, s_j)/(t_j, t_j);$
  $q_j = u_j - \alpha_j v_j;$
  $d_j = u_j + q_j;$
  $x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j;$
  $r_{j+1} = s_j - \omega_j t_j;$
  $\rho_{j+1} = (\tilde{r}_0, r_{j+1});$
  $\beta_j = (\rho_{j+1}/\rho_j)(\alpha_j/\omega_j);$
  $p_{j+1} = r_{j+1} + \beta_j(p_j - \omega_j v_j);$
end

As for Bi-CGS this algorithm requires two matrix-vector products with A.

Review Questions

1. What optimality property does the residual vectors $r_k = b - Ax_k$ in the GMRES method satisfy. In what subspace does the vector $r_k - r_0$ lie?
2. In Lanczos bi-orthogonalization bases for two different Krylov subspaces are computed. Which subspaces and what property has these bases?
3. (a) The bi-conjugate gradient (Bi-CG) method is based on the reduction of $A \in \mathbf{C}^{n\times n}$ to tridiagonal form by a general similarity transformation.

   (b) What are the main advantages and drawbacks of the Bi-CG method compared to GMRES.

   c) How are the approximations x_k defined in QMR?

Problems

1. Derive Algorithms CGLS and CGNE by applying the conjugate gradient algorithm to the normal equations $A^TAx = A^Tb$ and $AA^Ty = b$, $x = A^Ty$, respectively.
2. Consider using GMRES to solve the system $Ax = b$, where

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

   using $x_0 = 0$. Show that $x_1 = 0$, and that therefore GMRES(1) will never produce a solution.

10.6 Methods Related to Normal Equations.

10.6.1 Landweber's method

When $A \in \mathbf{R}^{m\times n}$, rank $(A) = n$, the normal equations of the first kind

$$A^TAx = A^Tb \tag{10.6.1}$$

give the conditions for the solution of the linear least squares problems

$$\min_x \|Ax - b\|_2, \tag{10.6.2}$$

Similarly, When rank $(A) = m$, the normal equations of the second kind

$$AA^Ty = b, \qquad x = A^Ty, \tag{10.6.3}$$

give the conditions for the solution of the minimum norm problem

$$\min \|x\|_2, \qquad Ax = b. \tag{10.6.4}$$

In particular, if A is nonsingular, then both systems are symmetric positive definite with solution $x = A^{-1}b$. Hence a natural extension of iterative methods for symmetric positive definite systems to general nonsingular, non-symmetric linear systems $Ax = b$ is to to apply them to the normal equations of first or second kind.

Because of the relation $\kappa(A^TA) = \kappa(AA^T) = \kappa^2(A)$, the condition number is squared compared to the original system $Ax = b$. From the estimate (10.4.19) we note that this can lead to a substantial decrease in the rate of convergence.

The non-stationary Richardson iteration applied to the normal equations $A^TAx = A^Tb$ can be written in the form

$$x^{(k+1)} = x^{(k)} + \omega_k A^T(b - Ax^{(k)}), \quad k = 1, 2, \ldots,$$

This method is often referred to as **Landweber's method**. It can be shown that this methods is convergent provided that for some $\epsilon > 0$ it holds

$$0 < \epsilon < \omega_k < (2 - \epsilon)/\sigma_{\max}(A), \quad \forall k.$$

An important things to notice in the implementation is that to avoid numerical instability and fill-in, the matrix A^TA should not be *explicitly* computed.

The eigenvalues of the iteration matrix $G = I - \omega A^TA$ equal

$$\lambda_k(G) = 1 - \alpha\sigma_k^2, \quad k = 1, \ldots, n,$$

where σ_k are the singular values of A. From this it can be shown that Richardson's method converges to the least squares solution $x = A^\dagger b$ if

$$x^{(0)} \in \mathcal{R}(A^T), \qquad 0 < \alpha < 2/\sigma_1^2(A).$$

Cimmino ([4]) introduced a method that is related the Landweber's method. Consider the nonsingular system $Ax = b$, where the rows of A are $a_1^T, \ldots, a_m^T$. The solution $x = A^{-1}b$ is then equal to the intersection of the n hyperplanes

$$a_i^Tx = b_i, \quad i = 1 : n.$$

In **Cimmino's method** one considers the reflections of an initial approximation $x^{(0)}$, with respect to these hyperplanes

$$x_i^{(0)} = x^{(0)} + 2\frac{b_i - a_i^Tx^{(0)}}{\|a_i\|_2}a_i, \quad i = 1 : n. \tag{10.6.5}$$

The next approximation is then taken to be

$$x^{(1)} = \sum_{i=1}^{n} m_i x_i^{(0)}/\mu, \quad \mu = \sum_{i=1}^{n} m_i.$$

This can be interpreted as the center of gravity of n masses m_i placed at the points $x_i^{(0)}$. Cimmino noted that the initial point $x^{(0)}$ and its reflections with respect to the n hyperplanes (10.6.5) all lie on a hypersphere the center of which is the solution

of the linear system. Because the center of gravity of the system of masses m_i must fall inside this hypersphere it follows that

$$\|x^{(1)} - x\|_2 < \|x^{(0)} - x\|_2,$$

that is, the error is reduced. Therefore Cimmino's method converges.

In matrix form Cimmino's method can be written as

$$x^{(k+1)} = x^{(k)} + \frac{2}{\mu} A^T D(b - Ax^{(k)}), \tag{10.6.6}$$

where $D = \operatorname{diag}(d_1, \ldots, d_n)$, where $d_i = m_i/\|a_i\|_2^2$. In particular, with $m_i = \|a_i\|_2$ we get Landwebers method with $\omega = 2/\mu$. It follows that Cimmino's method also converges for singular and inconsistent linear systems provided that $\operatorname{rank}(A) \geq 2$.

10.6.2 Jacobi's and Gauss–Seidel's Methods

Assume that all columns in A are nonzero, and let

$$A = (a_1, \ldots, a_n) \in \mathbf{R}^{m\times n}, \quad d_j = a_j^T a_j = \|a_j\|_2^2 > 0. \tag{10.6.7}$$

In **Jacobi's** method a sequence of approximations

$$x^{(k)} = \left(x_1^{(k)}, \ldots, x_n^{(k)} \right)^T, \qquad k = 1, 2, \ldots,$$

is computed from

$$x_j^{(k+1)} = x_j^{(k)} + a_j^T (b - Ax^{(k)})/d_j, \quad j = 1, 2, \ldots, n. \tag{10.6.8}$$

Jacobi's method can be written (10.6.8) in matrix form as

$$x^{(k+1)} = x^{(k)} + D_A^{-1} A^T (b - Ax^{(k)}), \tag{10.6.9}$$

where $D_A = \operatorname{diag}(d_1, \ldots, d_n) = \operatorname{diag}(A^T A)$. Jacobi's method is symmetrizable since

$$D_A^{1/2}(I - D_A^{-1} A^T A) D_A^{-1/2} = I - D_A^{-1/2} A^T A D_A^{-1/2}.$$

Jacobi's method can also be used to solve the normal equations of second type (10.6.3). This method can be written in the form

$$x^{(k+1)} = x^{(k)} + A^T D_A^{-1} (b - Ax^{(k)}), \quad D_A = \operatorname{diag}(AA^T). \tag{10.6.10}$$

The Gauss–Seidel method is a special case of the following class of **residual reducing** methods. Let $p_j \notin \mathcal{N}(A)$, $j = 1, 2, \ldots$, be a sequence of nonzero n-vectors and compute a sequence of approximations of the form

$$x^{(j+1)} = x^{(j)} + \alpha_j p_j, \qquad \alpha_j = p_j^T A^T (b - Ax^{(j)})/\|Ap_j\|_2^2. \tag{10.6.11}$$

It is easily verified that $r^{(j+1)} \perp Ap_j = 0$, where $r_j = b - Ax^{(j)}$, and hence

$$\|r^{(j+1)}\|_2^2 = \|r^{(j)}\|_2^2 - |\alpha_j|^2 \|Ap_j\|_2^2 \leq \|r^{(j)}\|_2^2,$$

which shows that this class of methods (10.6.11) is residual reducing. For a square matrix A method (10.6.12) was developed by de la Garza [6, 1951]. This class of residual reducing projection methods was studied by Householder and Bauer [14, 1960].

If A has linearly independent columns we obtain the Gauss–Seidel method for the normal equations by taking p_j in (10.6.11) equal to the unit vectors e_j in cyclic order. Then if $A = (a_1, a_2, \ldots, a_n)$, we have $Ap_j = Ae_j = a_j$. An iteration step in the Gauss–Seidel method consists of n minor steps where we put $z^{(1)} = x^{(k)}$, and $x^{(k+1)} = z^{(n+1)}$ is computed by

$$z^{(j+1)} = z^{(j)} + e_j a_j^T r^{(j)}/d_j, \qquad r^{(j)} = b - Az^{(j)}, \tag{10.6.12}$$

$j = 1, 2, \ldots, n$. In the jth minor step only the jth component of $z^{(j)}$ is changed, and hence the residual $r^{(j)}$ can be cheaply updated. With $r^{(1)} = b - Ax^{(k)}$ we obtain the recursions

$$\begin{aligned} &z^{(j+1)} = z^{(j)} + \delta_j e_j, \quad r^{(j+1)} = r^{(j)} - \delta_j a_j, \\ &\delta_j = a_j^T r^{(j)}/d_j, \quad j = 1, \ldots, n. \end{aligned} \tag{10.6.13}$$

Note that in the jth minor step only the jth column of A is accessed. and that it can be implemented without forming the matrix A^TA explicitly. In contrast to the Jacobi method the Gauss–Seidel method is not symmetrizable and the ordering of the columns of A will influence the convergence.

The Jacobi method has the advantage over Gauss–Seidel's method that it is more easily adapted to parallel computation, since (10.6.10) just requires a matrix-vector multiplication. Further, it does not require A to be stored (or generated) columnwise, since products of the form Ax and A^Tr can conveniently be computed also if A can only be accessed by rows. In this case, if $a_1^T, \ldots, a_m^T$ are the rows of A, then we have

$$(Ax)_i = a_i^T x, \quad i = 1, \ldots, n, \quad A^T r = \sum_{i=1}^{m} a_i r_i.$$

That is, for Ax we use an inner product formulation, and for A^Tr, an outer product formulation.

The **successive overrelaxation (SOR) method** for the normal equations $A^TAx = A^Tb$ is obtained by introducing an **relaxation parameter** ω in the Gauss–Seidel method (10.6.14),

$$\begin{aligned} &z^{(j+1)} = z^{(j)} + \delta_j e_j, \quad r^{(j+1)} = r^{(j)} - \delta_j a_j, \\ &\delta_j = \omega a_j^T r^{(j)}/d_j, \quad j = 1, \ldots, n. \end{aligned} \tag{10.6.14}$$

The SOR method always converges when A^TA is positive definite and ω satisfies $0 < \omega < 2$. The SOR shares with the Gauss–Seidel the advantage of simplicity and small storage requirements.

The Gauss–Seidel method for solving the normal equations of second kind can also be implemented without forming AA^T. We define a class of **error reducing**

methods as follows: Let $p_i \notin \mathcal{N}(A)$, $i = 1, 2, \ldots$, be a sequence of nonzero m-vectors and compute approximations of the form

$$x^{(j+1)} = x^{(j)} + \alpha_j A^T p_j, \qquad \alpha_j = p_j^T (b - Ax^{(j)})/\|A^T p_j\|_2^2. \tag{10.6.15}$$

If the system $Ax = b$ is consistent there is a unique solution x of minimum norm. If we denote the error by $d^{(j)} = x - x^{(j)}$, then by construction $d^{(j+1)} \perp A^T p_i$. Thus

$$\|d^{(j+1)}\|_2^2 = \|d^{(j)}\|_2^2 - |\alpha_j|^2 \|A^T p_j\|_2^2 \leq \|d^{(j)}\|_2^2,$$

i.e. this class of methods is error reducing.

We obtain the Gauss–Seidel method by taking p_j to be the unit vectors e_j in cyclic order. Then $A^T p_j = a_{j\cdot}$, where $a_{j\cdot}^T$ is the jth row of A^T and the iterative method (10.6.15) takes the form

$$x^{(j+1)} = x^{(j)} + a_{j\cdot}(b_j - a_{j\cdot}^T x^{(j)})/d_j, \quad j = 1, \ldots, n. \tag{10.6.16}$$

the approximation $y^{(j)}$ is updated by

$$\Delta y^{(j)} = e_j (b_j - a_{j\cdot}^T A^T y^{(i)})/d_j,$$

and with $x^{(j)} = A^T y^{(j)}$ and $x^{(j+1)} = x^{(j)} + A^T \Delta y^{(j)}$ we recover (10.6.16). This shows that if we take $x^{(0)} = Ay^{(0)}$, then for an arbitrary $y^{(0)}$ (10.6.16) is equivalent to the Gauss–Seidel method for (10.6.3). For the case of a square matrix A this method was originally devised by Kaczmarz [15, 1937].

The SOR method applied to the normal equations of the second kind can be obtained by introducing an acceleration parameter ω, i.e.

$$x^{(j+1)} = x^{(j)} + \omega a_{j\cdot}(c_j - a_{j\cdot}^T x^{(j)})/d_j, \quad j = 1, \ldots, n. \tag{10.6.17}$$

10.6.3 The Conjugate Gradient Method

The implementation of CG applied to the normal equations of the first kind becomes as follows:

Algorithm 10.6.1 CGLS

$$\begin{aligned}
&r_0 = b - Ax_0; \quad p_0 = s_0 = A^T r_0;\\
&\textbf{for } k = 0, 1, \ldots \textbf{ while } \|r_k\|_2 > \epsilon \textbf{ do}\\
&\quad q_k = Ap_k;\\
&\quad \alpha_k = \|s_k\|_2^2/\|q_k\|_2^2;\\
&\quad x_{k+1} = x_k + \alpha_k p_k;\\
&\quad r_{k+1} = r_k - \alpha_k q_k;\\
&\quad s_{k+1} = A^T r_{k+1};
\end{aligned}$$

$$\beta_k = \|s_{k+1}\|_2^2/\|s_k\|_2^2;$$
$$p_{k+1} = s_{k+1} + \beta_k p_k;$$
end

Note that it is important for the stability that the residuals $r_k = b - Ax_k$ and *not* the residuals $s_k = A^T(b - Ax_k)$ are recurred. The method obtained by applying CG to the normal equations of the second kind is also known as **Craig's Method**. This method can only be used for consistent problems, i.e., when $b \in \mathcal{R}(A)$. It can also be used to compute the (unique) minimum norm solution of an underdetermined system, $\min \|x\|_2$, subject to $Ax = b$, where $A \in \mathbf{R}^{m\times n}$, $m < n$.

Craig's method (CGNE) can be implemented as follows:

Algorithm 10.6.2 CGNE

$$r_0 = b - Ax_0; \quad p_0 = A^T r_0;$$
for $k = 0, 1, \ldots$ **while** $\|r_k\|_2 > \epsilon$ **do**
$$\alpha_k = \|r_k\|_2^2/\|p_k\|_2^2;$$
$$x_{k+1} = x_k + \alpha_k p_k;$$
$$r_{k+1} = r_k - \alpha_k A p_k;$$
$$\beta_k = \|r_{k+1}\|_2^2/\|r_k\|_2^2;$$
$$p_{k+1} = A^T r_{k+1} + \beta_k p_k;$$
end

Both CGLS and CGNE will generate iterates in the shifted Krylov subspace,

$$x_k \in x_0 + \mathcal{K}_k(A^T r_0, A^T A).$$

From the minimization property we have for the iterates in CGLS

$$\|x - x_k\|_{A^TA} = \|r - r_k\|_2 < 2\Big(\frac{\kappa - 1}{\kappa + 1}\Big)^k \|r_0\|_2,$$

where $\kappa = \kappa(A)$. Similarly for CGNE we have

$$\|y - y_k\|_{AA^T} = \|x - x_k\|_2 < 2\Big(\frac{\kappa - 1}{\kappa + 1}\Big)^k \|x - x_0\|_2.$$

For consistent problems the method CGNE should in general be preferred.

The main drawback with the two above methods is that they often converge very slowly, which is related to the fact that $\kappa(A^TA) = \kappa^2(A)$. Note, however, that in some special cases both CGLS and CGNE may converge much faster than alternative methods. For example, when A is orthogonal then $A^TA = AA^T = I$ and both methods converge in one step!

10.6.4 Least Squares and LSQR.

As shown by Paige and Saunders the Golub–Kahan bidiagonalization process developed in Section 9.9.4 can be used for developing methods related to CGLS and CGNE for solving the linear least squares problem (10.6.1) and the minimum norm problem (10.6.1), respectively.

To compute a sequence of approximate solutions to the least squares problem we start the recursion (9.9.19)–(9.19.20) by

$$\beta_1 u_1 = b - Ax_0, \qquad \alpha_1 v_1 = A^T u_1, \tag{10.6.18}$$

and for $j = 1, 2, \ldots$ compute

$$\begin{aligned} \beta_{j+1} u_{j+1} &= Av_j - \alpha_j u_j, \\ \alpha_{j+1} v_{j+1} &= A^T u_{j+1} - \beta_{j+1} v_j, \end{aligned} \tag{10.6.19}$$

where $\beta_j \geq 0$ and $\alpha_j \geq 0$, $j \geq 1$, are determined so that $\|u_j\|_2 = \|v_j\|_2 = 1$.

After k steps we have computed orthogonal matrices

$$V_k = (v_1, \ldots, v_k), \qquad U_{k+1} = (u_1, \ldots, u_{k+1})$$

and a rectangular lower bidiagonal matrix

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} \in \mathbf{R}^{(k+1)\times k}. \tag{10.6.20}$$

The recurrence relations (10.6.18)–(10.6.19) can be written in matrix form as

$$U_{k+1}\beta_1 e_1 = r_0, \quad r_0 = b - Ax_0, \tag{10.6.21}$$

where e_1 denotes the first unit vector, and

$$AV_k = U_{k+1}B_k, \qquad A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. \tag{10.6.22}$$

We now seek an approximate solution $x_k \in \mathcal{K}_k = \mathcal{K}_k(A^T r_0, A^T A)$. From the recursions (10.6.18)–(10.6.19) it follows that $\mathcal{K}_k = \text{span}(V_k)$ and so we write

$$x_k = x_0 + V_k y_k. \tag{10.6.23}$$

Multiplying the first equation in (10.6.22) by y_k we obtain $Ax_k = AV_k y_k = U_{k+1}B_k y_k$, and then from (10.6.21)

$$b - Ax_k = U_{k+1} t_{k+1}, \qquad t_{k+1} = \beta_1 e_1 - B_k y_k. \tag{10.6.24}$$

Using the orthogonality of U_{k+1} and V_k, which holds in exact arithmetic, it follows that $\|b - Ax_k\|_2$ is minimized over all $x_k \in \text{span}(V_k)$ by taking y_k to be the solution to the least squares problem

$$\min_{y_k} \|B_k y_k - \beta_1 e_1\|_2. \tag{10.6.25}$$

This forms the basis for the algorithm LSQR. Note the special form of the right-hand side, which holds because the starting vector was taken as b. Now $x_k = V_k y_k$ solves $\min_{x_k \in \mathcal{K}_k} \|Ax - b\|_2$, where $\mathcal{K}_k = \mathcal{K}_k(A^T b, A^T A)$. Thus *mathematically* LSQR generates the same sequence of approximations as Algorithm 10.6.3 CGLS.

To solve (10.6.25) stably we need the QR factorization $Q_k^T B_k = R_k$. This can be computed by premultiplying B_k by a sequence Givens transformations, which are also applied to the right hand side e_1,

$$G_{k,k+1} G_{k-1,k} \cdots G_{12} (B_k \ e_1) = \begin{pmatrix} R_k & d_k \\ 0 & \rho_k \end{pmatrix}.$$

Here the rotation $G_{j,j+1}$ is used to zero the element β_{j+1}. It is easily verified that R_k is an upper bidiagonal matrix. The least squares solution y_k and the norm of the corresponding residual are then obtained from

$$R_k y_k = \beta e_1, \qquad \|b - Ax_k\|_2 = |\rho_k|.$$

Note that the whole vector y_k differs from y_{k-1}. An updating formula for x_k can be derived using an idea due to Paige and Saunders. With $W_k = V_k R_k^{-1}$ we can write

$$\begin{aligned} x_k &= x_0 + V_k y_k = x_0 + \beta_1 V_k R_k^{-1} d_k = x_0 + \beta_1 W_k d_k \\ &= x_0 + \beta_1 (W_{k-1}, \ w_k) \begin{pmatrix} d_{k-1} \\ \tau_k \end{pmatrix} = x_{k-1} + \beta_1 \tau_k w_k. \end{aligned} \tag{10.6.26}$$

Consider now the minimum norm problem for a consistent system $Ax = b$. Let L_k be the lower bidiagonal matrix formed by the first k rows of B_k

$$L_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{pmatrix} \in \mathbf{R}^{k \times k}. \tag{10.6.27}$$

The relations (10.6.22) can now be rewritten as

$$AV_k = U_k L_k + \beta_{k+1} u_{k+1} e_k^T, \qquad A^T U_k = V_k L_k^T. \tag{10.6.28}$$

The iterates x_k in Craig's method can be computed as

$$L_k y_k = \beta_1 e_1, \qquad x_k = V_k z_k. \tag{10.6.29}$$

Using (10.6.28) and (10.6.18) it follows that the residual vector satisfies

$$r_k = b - AV_k z_k = -\beta_{k+1} u_{k+1} (e_k^T z_k) = -\beta_{k+1} \eta_k u_{k+1},$$

and hence $U_k^T r_k = 0$. It can be shown that if $r_{k-1} \neq 0$ then $\alpha_k \neq 0$. Hence the vectors y_k and x_k can recursively be formed using

$$\eta_k = -\frac{\beta_k}{\alpha_k} \eta_{k-1}, \qquad x_k = x_{k-1} + \eta_k v_k.$$

10.7 Preconditioned Iterative Methods

Preconditioned iterative methods can be viewed as a compromise between a direct and iterative solution method. General purpose techniques for constructing preconditioners have made iterative methods successful in many industrial applications.

The term "preconditioning" dates back to Turing in 1948, and is in general taken to mean the transformation of a problem to a form that can more efficiently be solved. In order to be effective iterative methods must usually be combined with a (nonsingular) preconditioning matrix M, which in some sense is an approximation to A. The original linear system $Ax = b$ is then transformed by considering the **left-preconditioned system**

$$M^{-1}Ax = M^{-1}b. \tag{10.7.1}$$

or **right-preconditioned system**

$$AM^{-1}u = b, \qquad u = Mx. \tag{10.7.2}$$

The idea is to choose M so that the rate of convergence of the iterative method is improved. Note that the product $M^{-1}A$ (or AM^{-1}) should never be formed. The preconditioned iteration is instead implemented by forming matrix vector products with A and M^{-1} separately. Since forming $u = M^{-1}v$ for an arbitrary vector v is equivalent to solving a linear system $Mu = v$, the inverse M^{-1} is not needed either.

Often the rate of convergence depends on the spectrum of the transformed matrix. Since the eigenvalues of $M^{-1}A$ and AM^{-1} are the same, we see that the main difference between these two approaches is that the actual residual norm is available in the right-preconditioned case.

If A is symmetric, positive definite, the preconditioned system should also have this property. In this case, it is natural to consider a **split preconditioner**. Let $M = LL^T$ where L is the Cholesky factor of M. Then we consider the preconditioned linear system

$$\tilde{A} = L^{-1}AL^{-T}, \qquad \tilde{x} = L^T x, \qquad \tilde{b} = L^{-1}b, \tag{10.7.3}$$

and the spectrum of $\tilde{A}$ is real. Note that the spectrum of $A = C^{-1}AC^{-T}$ is the same as for $L^{-T}L^{-1}A = M^{-1}A$.

10.7.1 The Preconditioned CG Method

The conjugate gradient algorithm 10.4.1 can be applied to linear systems $Ax = b$, where A is symmetric positive definite. In this case it is natural to use a split preconditioner and consider the system (10.7.3).

In implementing the preconditioned conjugate gradient method we need to form matrix-vector products of the form $t = \tilde{A}p = L^{-1}(A(L^{-T}p))$. These can be calculated by solving two linear systems and performing one matrix multiplication with A as

$$L^T q = p, \qquad s = Aq, \qquad Lt = s.$$

Thus, the extra work per step in using the preconditioner essentially is to solve two linear systems with matrix L^T and L respectively.

The preconditioned algorithm will have recursions for the transformed variables and residuals vectors $\tilde{x} = L^T x$ and $\tilde{r} = L^{-1}(b - Ax)$. It can be simplified by reformulating it in terms of the original variables x and residual $r = b - Ax$. It is left as an exercise to show that if we let $p_k = L^{-T}\tilde{p}_k$, $z_k = L^{-T}\tilde{r}_k$, and

$$M = LL^T, \tag{10.7.4}$$

we can obtain the following implementation of the **preconditioned conjugate gradient method**:

Algorithm 10.7.1
Preconditioned Conjugate Gradient Method

$r_0 = b - Ax_0; \quad p_0 = z_0 = M^{-1}r_0;$
for $k = 0, 1, 2, \ldots,$ **while** $\|r_k\|_2 > \epsilon$ **do**
  $w = Ap_k;$
  $\beta_k = (z_k, r_k)/(p_k, Ap_k);$
  $x_{k+1} = x_k + \beta_k p_k;$
  $r_{k+1} = r_k - \beta_k Ap_k;$
  $z_{k+1} = M^{-1}r_{k+1};$
  $\beta_k = (z_{k+1}, r_{k+1})/(z_k, r_k);$
  $p_{k+1} = z_{k+1} + \beta_k p_k;$
end

A surprising and important feature of this version is that it depends only on the symmetric positive definite matrix $M = LL^T$.

The rate of convergence in the $\tilde{A}$-norm depends on $\kappa(\tilde{A})$, see (10.4.19). Note, however, that

$$\|\tilde{x} - \tilde{x}_k\|_{\tilde{A}}^2 = (\tilde{x} - \tilde{x}_k)^T L^{-1} A L^{-T} (\tilde{x} - \tilde{x}_k) = \|x - x_k\|_A^2,$$

so the rate of convergence in A-norm of the error in x also depends on $\kappa(\tilde{A})$. The preconditioned conjugate gradient method will have rapid convergence if one or both of the following conditions are satisfied:

i. $M^{-1}A$ to have small condition number, or

ii. $M^{-1}A$ to have only few distinct eigenvalues.

For symmetric indefinite systems SYMMLQ can be combined with a positive definite preconditioner M. To solve the symmetric indefinite system $Ax = b$ the preconditioner is regarded to have the form $M = LL^T$ and SYMMLQ *implicitly* applied to the system

$$L^{-1}AL^{-T}w = L^{-1}b.$$

The algorithm accumulates approximations to the solution $x = L^{-T}w$, without approximating w. A MATLAB implementation of this algorithm, which only requires solves with M, is given by Gill et al. [9].

10.7.2 Preconditioned CGLS and CGNE.

To precondition CGLS Algorithm (10.6.3) it is natural to use a right preconditioner $S \in \mathbf{R}^{n\times n}$, i.e., perform the transformation of variables

$$\min_y \|(AS^{-1})y - b\|_2, \quad Sx = y.$$

(Note that for a nonconsistent system $Ax = b$ a left preconditioner would change the problem.) If we apply CGLS to this problem and formulate the algorithm in terms of the original variables x, we obtain the following algorithm:

Algorithm 10.7.2 Preconditioned CGLS.

$r_0 = b - Ax_0; \quad p_0 = s_0 = S^{-T}(A^T r_0);$
for $k = 0, 1, \ldots$ **while** $\|r_k\|_2 > \epsilon$ **do**
  $t_k = S^{-1}p_k;$
  $q_k = At_k;$
  $\alpha_k = \|s_k\|_2^2/\|q_k\|_2^2;$
  $x_{k+1} = x_k + \alpha_k t_k;$
  $r_{k+1} = r_k - \alpha_k q_k;$
  $s_{k+1} = S^{-T}(A^T r_{k+1});$
  $\beta_k = \|s_{k+1}\|_2^2/\|s_k\|_2^2;$
  $p_{k+1} = s_{k+1} + \beta_k p_k;$
end

For solving a consistent underdetermined systems we can derive a preconditioned version of CGNE. Here it is natural to use a left preconditioner S, and apply CGNE to the problem

$$\min \|x\|_2, \qquad S^{-1}Ax = S^{-1}b,$$

i.e., the residual vectors are transformed. If the algorithm is formulated in terms of the original residuals, the following algorithm results:

Algorithm 10.7.3 Preconditioned CGNE

$$
\begin{aligned}
&r_0 = b - Ax_0; \quad z_0 = S^{-1}r_0; \quad p_0 = A^T(S^{-T}z_0);\\
&\textbf{for } k = 0, 1, \ldots \textbf{ while } \|r_k\|_2 > \epsilon \textbf{ do}\\
&\quad \alpha_k = \|z_k\|_2^2/\|p_k\|_2^2;\\
&\quad x_{k+1} = x_k + \alpha_k p_k;\\
&\quad r_{k+1} = r_k - \alpha_k A p_k;\\
&\quad z_{k+1} = S^{-1}r_{k+1};\\
&\quad \beta_k = \|z_{k+1}\|_2^2/\|z_k\|_2^2;\\
&\quad p_{k+1} = A^T(S^{-T}z_{k+1}) + \beta_k p_k;\\
&\textbf{end}
\end{aligned}
$$

Algorithm PCCGLS still minimizes the error functional $\|\hat{r} - r^{(k)}\|_2$, where $r = b - Ax$, but over a different Krylov subspace

$$x^{(k)} = x^{(k)} + \mathcal{K}_k, \qquad \mathcal{K}_k = (S^{-1}S^{-T}A^TA, S^{-1}S^{-T}A^Tr_0).$$

Algorithm PCCGNE minimizes the error functional $\|\hat{x} - x^{(k)}\|_2$, over the Krylov subspace

$$x^{(k)} = x^{(k)} + \mathcal{K}_k, \qquad \mathcal{K}_k = (A^TS^{-T}S^{-1}A, A^TS^{-T}S^{-1}r_0).$$

The rate of convergence for PCGTLS depends on $\kappa(AS^{-1})$, and for PCCGNE on $\kappa(S^{-1}A) = \kappa(A^TS^{-T})$.

10.7.3 Preconditioned GMRES

For nonsymmetric linear systems there are two options for applying the preconditioner. We can use the left preconditioned system (10.7.1) or the right preconditioned system (10.7.2). (If A is almost symmetric positive definite, then a split preconditioner might also be considered.) The changes to the GMRES algorithm are small.

In the case of using a left preconditioner M only the following changes in the Arnoldi algorithm are needed. We start the recursion with

$$r_0 = M^{-1}(b - Ax_0), \quad \beta_1 = \|r_0\|_2; \quad v_1 = r_0/\beta_1,$$

and define

$$z_j = M^{-1}Av_j, \quad j = 1, 2, \ldots, k.$$

All computed residual vectors will be preconditioned residuals $M^{-1}(b-Ax_m)$. This is a disadvantage since most stopping criteria depend are based on the actual residuals $r_m = b - Ax_m$. In this left preconditioned version the transformed residual norm $\|M^{-1}(b-Ax)\|_2$ will be minimized among all vectors of the form

$$x_0 + \mathcal{K}_m(M^{-1}r_0, M^{-1}A). \tag{10.7.5}$$

In the right preconditioned version of GMRES the actual residual vectors are used, but the variables are transformed according to $u = Mx$ ($x = M^{-1}u$). The right preconditioned algorithm can easily be modified to give the untransformed solution. We have

$$z_j = AM^{-1}v_j, \quad j = 1, 2, \ldots, k.$$

The kth approximation is $x_k = x_0 + M^{-1}V_k y_k$, where y_k solves

$$\min_{y_k} \|\beta_1 e_1 - \bar{H}_k y_k\|_2.$$

As before this can be written as

$$x_k = x_{k-1} + \beta_1 \tau_k M_k^{-1} w_k, \quad w_k = R_k y_k,$$

see (10.6.26).

In the right preconditioned version the residual norm $\|b - AM^{-1}u\|_2$ will be minimized among all vectors of the form $u_0 + \mathcal{K}_m(r_0, AM^{-1})$. However, this is equivalent to minimizing $\|b - AM^{-1}u\|_2$ among all vectors of the form

$$x_0 + M^{-1}\mathcal{K}_m(r_0, AM^{-1}). \tag{10.7.6}$$

Somewhat surprisingly the two affine subspaces (10.7.5) and (10.7.6) are the same! The jth vector in the two Krylov subspaces are $w_j = (M^{-1}A)^j M^{-1} r_0$ and $\tilde{w}_j = M^{-1}(AM^{-1})^j r_0$. By a simple induction proof it can be shown that $M^{-1}(AM^{-1})^j = (M^{-1}A)^j M^{-1}$ and so $\tilde{w}_j = w_j$, $j \geq 0$. Hence the left and right preconditioned versions generate approximations in the same Krylov subspaces, and they differ only with respect to which error norm is minimized.

For the case when A is diagonalizable, $A = X\Lambda X^{-1}$, where $\Lambda = \operatorname{diag}(\lambda_i)$ we proved the error estimate

$$\frac{||r_k||_2}{||r_0||_2} \leq \kappa_2(X) \min_{q_k} \max_{i=1,2,\ldots,n} |q_k(\lambda_i)|, \tag{10.7.7}$$

where q_k is a polynomial of degree $\leq k$ and $q_k(0) = 1$. Because of the factor $\kappa_2(X)$ in (10.7.7) the rate of convergence can no longer be deduced from the spectrum $\{\lambda_i\}$ of the matrix A alone. Since the spectrum of $M^{-1}A$ and AM^{-1} are the same we can expect that the convergence behavior will be similar when if A is close to normal.

10.8 Preconditioners

A preconditioner should typically satisfy the following conditions:

(i) $M^{-1}A = I + R$, where $\|R\|$ is small.

(ii) Linear systems of the form $Mu = v$ should be easy to solve.

(iii) $\text{nz}(M) \approx \text{nz}(A)$.

Condition (i) implies fast convergence, (ii) that the arithmetic cost of preconditioning is reasonable, and (iii) that the storage overhead is not too large. Obviously these conditions are contradictory and a compromise must be sought. For example, taking $M = A$ is optimal in the sense of (i), but obviously this choice is ruled out by (ii).

The choice of preconditioner is strongly problem dependent and possibly the most crucial component in the success of an iterative method! A preconditioner which is expensive to compute may become viable if it is to be used many times, as may be the case, e.g., when dealing with time-dependent or nonlinear problems. It is also dependent on the architecture of the computing system. Preconditioners that are efficient in a scalar computing environment may show poor performance on vector and parallel machines.

10.8.1 Preconditioners from Matrix Splittings

The stationary iterative method

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \qquad k = 0, 1, \ldots, \tag{10.8.1}$$

corresponds to a matrix splitting $A = M - N$, and the iteration matrix

$$B = M^{-1}N = I - M^{-1}A.$$

The iteration (10.8.1) can be considered as a fixed point iteration applied to the preconditioned system $M^{-1}Ax = M^{-1}b$. Hence, the basic iterative methods considered in Sections 10.1.3 give simple examples of preconditioners.

The Jacobi and Gauss–Seidel methods are both special cases of one-step stationary iterative methods. Using the standard splitting $A = D_A - L_A - U_A$, where D_A is diagonal, L_A and U_A are strictly lower and upper triangular, these methods correspond to the matrix splittings

$$M_J = D_A, \quad \text{and} \quad M_{GS} = D_A - L_A.$$

If A is symmetric positive definite then $M_J = D_A > 0$ and symmetric. However, M_{GS} is lower triangular and unsymmetric.

The simplest choice related to this splitting is to take $M = D_A$. This corresponds to a diagonal scaling of the rows of A, such that the scaled matrix $M^{-1}A = D_A^{-1}A$ has a unit diagonal. For s.p.d. matrices symmetry can be preserved by using a split preconditioner with $L = L^T = D_A^{1/2}$. In this case it can be shown that this is close to the optimal diagonal preconditioning.

Lemma 10.8.1. Van der Sluis [1969]

Let $A = D_A - L_A - L_A^T$ be a symmetric positive definite matrix. Then if A has at most q nonzero elements in any row it holds that

$$\kappa(D_A^{-1/2} A D_A^{-1/2}) = \min_{D>0} \kappa(DAD).$$

Although diagonal scaling may give only a modest improvement in the rate of convergence it is trivial to implement and therefore recommended even if no other preconditioning is carried out.

In Section 10.2.2 it was shown that for a symmetric matrix A the SSOR iteration method corresponds to a splitting with the matrix

$$M_{SSOR} = \frac{1}{\omega(2-\omega)} \big(D_A - \omega L_A\big) D_A^{-1} \big(D_A - \omega U_A\big).$$

Since M_{SSOR} is given in the form of an LDL^T factorization it is easy to solve linear systems involving this preconditioner. It also has the same sparsity as the original matrix A. For $0 < \omega < 2$, if A is s.p.d. so is M_{SSOR}.

The performance of the SSOR splitting turns out to be fairly insensitive to the choice of ω. For systems arising from second order boundary values problems, e.g., the model problem studied previously, the original condition number $\kappa(A) = \mathcal{O}(h^{-2})$ can be reduced to $\kappa(M^{-1}A) = \mathcal{O}(h^{-1})$. Taking $\omega = 1$ is often close to optimal. This corresponds to the symmetric Gauss–Seidel (SGS) preconditioner

$$M_{SGS} = \big(D_A - L_A\big) D_A^{-1} \big(D_A - U_A\big).$$

All the above preconditioners satisfy conditions (ii) and (iii). The application of the preconditioners involves only triangular solves and multiplication with a diagonal matrix. They are all defined in terms of elements of the original matrix A, and hence do not require extra storage. However, they may not be very effective with respect to the condition (i).

10.8.2 Incomplete LU Factorizations

The SGS preconditioner has the form $M_{SGS} = LU$ where $L = \big(I - L_A^T D_A^{-1}\big)$ is lower triangular and $U = \big(D_A - L_A^T\big)$ upper triangular. To find out how well M_{SGS} approximates A we form the **defect matrix** ixdefect matrix

$$A - LU = D_A - L_A - U_A - \big(I - L_A D_A^{-1}\big)\big(D_A - U_A\big) = -L_A D_A^{-1} U_A.$$

An interesting question is whether we can find matrices L and U with the same nonzero structure as above, but with a smaller defect matrix $R = LU - A$.

We now develop an important class of preconditioners obtained from so called **incomplete LU-factorizations** of A. The idea is to compute a lower triangular matrix L and an upper triangular matrix U with a *prescribed* sparsity structure such that

$$A = LU - R,$$

with R small. Such incomplete LU-factorizations can be realized by performing a modified Gaussian elimination on A, in which elements are allowed only in in specified places in the L and U matrices. We assume that these places (i, j) are given by the index set

$$\mathcal{P} \subset \mathcal{P}_n \equiv \{(i, j) \,|\, 1 \le i, j \le n\},$$

where the diagonal positions always are included in $\mathcal{P}$. For example, we could take $\mathcal{P} = \mathcal{P}_A$, the set of nonzero elements in A.

Algorithm 10.8.1 Incomplete LU Factorization

```
for k = 1, ..., n − 1
    for i = k + 1, ..., n
        if (i, k) ∈ P  l_ik = a_ik/a_kk;
        for j = k + 1, ..., n
            if (k, j) ∈ P  a_ij = a_ij − l_ik a_kj;
        end
    end
end
```

The elimination consists of $n-1$ steps. In the kth step we first subtract from the current matrix elements with indices (i, k) and $(k, i) \notin \mathcal{P}$ and place in a defect matrix R_k. We then carry out the $k$the step of Gaussian elimination on the so modified matrix. This process can be expressed as follows. Let $A_0 = A$ and

$$\widetilde{A}_k = A_{k-1} + R_k, \qquad A_k = L_k \widetilde{A_k}, \quad k = 1, \ldots, n-1.$$

Applying this relation recursively we obtain

$$\begin{aligned} A_{n-1} &= L_{n-1}\widetilde{A}_{n-1} = L_{n-1}A_{n-2} + L_{n-1}R_{n-1} \\ &= L_{n-1}L_{n-2}A_{n-3} + L_{n-1}L_{n-2}R_{n-2} + L_{n-1}R_{n-1} \\ &= L_{n-1}L_{n-2}\cdots L_1 A + L_{n-1}L_{n-2}\cdots L_1 R_1 \\ &\quad + \cdots + L_{n-1}L_{n-2}R_{n-2} + L_{n-1}R_{n-1}. \end{aligned}$$

We further notice that since the first $m-1$ rows of R_m are zero it follows that $L_k R_m = R_m$, if $k < m$. Then by combining the above equations we find $LU = A+R$, where

$$U = A_{n-1}, \qquad L = (L_{n-1}L_{n-2}\cdots L_1)^{-1}, \qquad R = R_1 + R_2 + \cdots R_{n-1}.$$

Algorithm 10.8.2 can be improved by noting that any elements in the resulting $(n-k) \times (n-k)$ lower part of the reduced matrix not in $\mathcal{P}$ need not be carried

along and can also be included in the defect matrix R_k. This is achieved simply by changing line five in the algorithm to

$$\textbf{if } (k,j) \in \mathcal{P} \textbf{ and } (i,j) \in \mathcal{P} \quad a_{ij} = a_{ij} - l_{ik}a_{kj};$$

In practice the matrix A is sparse and the algorithm should be specialized to take this into account. In particular, a version where A is processed a row at a time is more convenient for general sparse matrices. Such an algorithm can be derived by interchanging the k and i loops in Algorithm 10.8.2.

Example 10.8.1. For the model problem using a five-point approximation the non-zero structure of the resulting matrix is given by

$$\mathcal{P}_{\mathcal{A}} = \{(i,j) \,|\, |i-j| = -n, -1, 0, 1, n\}.$$

Let us write $A = LU + R$, where

$$L = L_{-n} + L_{-1} + L_0, \qquad U = U_0 + U_1 + U_n,$$

where L_{-k} (and U_k) denote matrices with nonzero elements only in the k-th lower (upper) diagonal. By the rule for multiplication by diagonals (see Problem 6.1.6),

$$A_k B_l = C_{k+l}, \text{ if } k + l \le n - 1,$$

we can form the product

$$\begin{aligned} LU = (L_{-n} + L_{-1} + L_0)(U_0 + U_1 + U_n) = (L_{-n}U_n + L_{-1}U_1 + L_0U_0) \\ + L_{-n}U_0 + L_{-1}U_0 + L_0U_n + L_0U_1 + R, \end{aligned}$$

where $R = L_{-n}U_1 + L_{-1}U_n$. Hence the defect matrix R has nonzero elements only in two extra diagonals.

A family of preconditioners can be derived by different choices of the set $\mathcal{P}$. The simplest choice is to take $\mathcal{P}$ equal to the sparsity pattern of A. This is called a level 0 incomplete factorization. A level 1 incomplete factorization is obtained by using the union of $\mathcal{P}$ and the pattern of the defect matrix$R = LL^T - A$. Higher level incomplete factorizations are defined in a similar way, and so on.

An incomplete LU factorization may not exist even if A is nonsingular and has an LU factorization. However, for some more restricted classes of matrices the existence of incomplete factorizations can be guaranteed. Many matrices arising from the discretization of partial differential equations have the following property.

Definition 10.8.2. *A matrix $A = (a_{ij})$ is an M-matrix if $a_{ij} \le 0$ for $i \ne j$, A is nonsingular and $A^{-1} \ge 0$.*

The following result was proved by Meijerink and van der Vorst [20, 1977].

Theorem 10.8.3.

If A is an M-matrix, there exists for every set $\mathcal{P}$ such that $(i,j) \in \mathcal{P}$ for $i = j$, uniquely defined lower and upper triangular matrices L and U with $l_{ij} = 0$ or $u_{ij} = 0$ if $(i,j) \notin \mathcal{P}$, such that the splitting $A = LU - R$ is regular.

In case A is s.p.d., we define similarly an **incomplete Cholesky factorization**. Here the nonzero set $\mathcal{P}$ is assumed to be symmetric, i.e., if $(i,j) \in \mathcal{P}$ then also $(j,i) \in \mathcal{P}$. Positive definiteness of A alone is not sufficient to guarantee the existence of an incomplete Cholesky factorization. This is because zero elements may occur on the diagonal during the factorization.

For the case when A is a symmetric M-matrix, a variant of the above theorem guarantees the existence for each symmetric set $\mathcal{P}$ such that $(i,j) \in \mathcal{P}$ for $i = j$, a uniquely defined lower triangular matrix L, with $l_{ij} = 0$ if $(i,j) \notin \mathcal{P}$ such that the splitting $A = LL^T - R$ is regular. In particular the matrix arising from the model problem is a symmetric M-matrix. A symmetric M-matrix is also called a **Stieltjes** matrix.

An implementation of the incomplete Cholesky factorization in the general case is given below.

Algorithm 10.8.2 Incomplete Cholesky Factorization

for $j = 1, 2, \ldots, n$

$$l_{jj} = \Big(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2\Big)^{1/2};$$

for $i = j+1, \ldots, n$

if $(i,j) \notin \mathcal{P}$ **then** $l_{ij} = 0$ **else**

$$l_{ij} = \Big(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}\Big)$$

end

end

10.8.3 Block Incomplete Factorizations

Many matrices arising from the discretization of multidimensional problems have a block structure. For such matrices one can generalize the above idea and develop **block incomplete factorizations**. In particular, we consider here a symmetric positive definite block tridiagonal matrices of the form

$$A = \begin{pmatrix} D_1 & A_2^T & & & \\ A_2 & D_2 & A_3^T & & \\ & A_3 & \ddots & \ddots & \\ & & \ddots & \ddots & A_N^T \\ & & & A_N & D_N \end{pmatrix} = D_A - L_A - L_A^T, \tag{10.8.2}$$

with square diagonal blocks D_i. For the model problem with the natural ordering of mesh points we obtain this form with $A_i = -I$, $D_i = \text{tridiag}(-1 \quad 4 \quad -1)$. If systems with D_i can be solved efficiently a simple choice of preconditioner is the block diagonal preconditioner

$$M = \text{diag}\,(D_1, D_2, \ldots, D_N).$$

The case $N = 2$ is of special interest. For the system

$$\begin{pmatrix} D_1 & A_2^T \\ A_2 & D_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \tag{10.8.3}$$

the **block diagonal preconditioner** gives a preconditioned matrix of the form

$$M^{-1}A = \begin{pmatrix} I & D_1^{-1}A_2^T \\ D_2^{-1}A_2 & I \end{pmatrix}.$$

Note that this matrix is of the form (10.2.3) and therefore has property A. Suppose the conjugate gradient method is used with this preconditioner and a starting approximation $x_1^{(0)}$. If we set

$$x_2^{(0)} = D_2^{-1}(b_2 - A_2 x_1^{(0)}),$$

then the corresponding residual $r_2^{(0)} = b_2 - D_2 x_1^{(0)} A_2 x_2^{(0)} = 0$. It can be shown that in the following steps of the conjugate gradient method we will alternately have

$$r_2^{(2k)} = 0, \qquad r_1^{(2k+1)} = 0, \qquad k = 0, 1, 2, \ldots.$$

This can be used to save about half the work.

If we eliminate x_1 in the system (10.8.3) then we obtain

$$Sx_2 = b_2 - A_2 D_1^{-1} b_1, \qquad S = D_2 - A_2 D_1^{-1} A_2^T, \tag{10.8.4}$$

where S is the Schur complement of D_1 in A. If A is s.p.d., then S is also s.p.d., and hence the conjugate gradient can be used to solve the system (10.8.4). This process is called **Schur complement preconditioning**. Here it is not necessary to form the Schur complement S, since we only need the effect of S on vectors. We can save some computations by writing the residual of the system (10.8.4) as

$$r_2 = (b_2 - D_2 x_2) - A_2 D_1^{-1}(b_1 - A_2^T x_2).$$

Note here that $x_1 = D_1^{-1}(b_1 - A_2^T x_2)$ is available as an intermediate result. The solution of the system $D_1 x_1 = b_1 - A_2^T x_2$ is cheap when, e.g., D_1 is tridiagonal. In other cases this system may be solved in each step by an iterative method in an **inner iteration**.

We now describe a **block incomplete Cholesky factorization** due to Concus, Golub and Meurant [5], which has proved to be very useful. We assume in the following that in (10.8.2) D_i is tridiagonal and A_i is diagonal, as in the model problem. First recall from Section 6.4.6 that the exact block Cholesky factorization of a symmetric positive definite block-tridiagonal matrix can be written as

$$A = (\Sigma + L_A)\Sigma^{-1}(\Sigma + L_A^T),$$

where L_A is the lower block triangular part of A, and $\Sigma = \operatorname{diag}(\Sigma_1, \ldots, \Sigma_n)$, is obtained from the recursion

$$\Sigma_1 = D_1, \qquad \Sigma_i = D_i - A_i \Sigma_{i-1}^{-1} A_i^T, \quad i = 2, \ldots, N.$$

For the model problem, although D_1 is tridiagonal, Σ^{-1} and hence Σ_i, $i \geq 2$, are dense. Because of this the exact block Cholesky factorization is not useful.

Instead we consider computing an incomplete block factorization from

$$\Delta_1 = D_1, \qquad \Delta_i = D_i - A_i \Lambda_{i-1}^{-1} A_i^T, \quad i = 2, \ldots, N. \tag{10.8.5}$$

Here, for each i, Λ_{i-1} is a sparse approximation to Δ_{i-1}. The incomplete block Cholesky factorization is then

$$M = (\Delta + L_A)\Delta^{-1}(\Delta + L_A^T), \qquad \Delta = \operatorname{diag}(\Delta_1, \ldots, \Delta_n).$$

The corresponding defect matrix is $R = M - A = \operatorname{diag}(R_1, \ldots, R_n)$, where $R_1 = \Delta_1 - D_1 = 0$,

$$R_i = \Delta_i - D_i - A_i \Delta_{i-1}^{-1} A_i^T, \qquad i = 2, \ldots, n.$$

We have assumed that the diagonal blocks D_i are diagonally dominant symmetric tridiagonal matrices. We now discuss the construction of an approximate inverse of such a matrix

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{pmatrix},$$

where $\alpha_i > 0$, $i = 1, \ldots, n$ and $\beta_i < 0$, $i = 1, \ldots, m-1$. A sparse approximation of D_i^{-1} can be obtained as follows. First compute the Cholesky factorization $T = LL^T$, where

$$L = \begin{pmatrix} \gamma_1 & & & & \\ \delta_1 & \gamma_2 & & & \\ & \delta_2 & \ddots & & \\ & & \ddots & \gamma_{n-1} & \\ & & & \delta_{n-1} & \gamma_n \end{pmatrix},$$

It can be shown that the elements of the inverse $T^{-1} = L^{-T}L$ decrease strictly away from the diagonal. This suggests that the matrix L^{-1}, which is lower triangular and dense, is approximated by a banded lower triangular matrix $L^{-1}(p)$, taking only the first $p+1$ lower diagonals of the exact L^{-1}. Note that elements of the matrix L^{-1}

$$L^{-1} = \begin{pmatrix} 1/\gamma_1 & & & & \\ \zeta_1 & 1/\gamma_2 & & & \\ \eta_1 & \zeta_2 & \ddots & & \\ \vdots & \ddots & \ddots & 1/\gamma_{n-1} & \\ & \cdots & \eta_{n-2} & \zeta_{n-1} & 1/\gamma_n \end{pmatrix},$$

can be computed diagonal by diagonal. For example, we have

$$\zeta_i = \frac{\delta_i}{\gamma_i \gamma_{i+1}}, \qquad i = 2, \ldots, n-1.$$

For $p = 0$ we get a diagonal approximate inverse. For $p = 1$ the approximate Cholesky factor $L^{-1}(1)$ is lower bidiagonal, and the approximate inverse $L^{-T}(1)L^{-1}(1)$ a tridiagonal matrix. Since we have assumed that A_i are diagonal matrices, the approximations Δ_i generated by (10.8.5) will in this case be tridiagonal.

10.8.4 Fast Direct Methods

For the solution of discretizations of some elliptic problems on a rectangular domain fast direct methods can be developed. For this approach to be valid we needed to make strong assumptions about the regularity of the system. It applies only to discretizations of problems with constant coefficients on a rectangular domain. However, if we have variable coefficients the fast solver may be used as a preconditioner in the conjugate gradient method. Similarly, problems on an irregular domain may be embedded in a rectangular domain, and again we may use a preconditioner based on a fast solver.

Consider a linear system $Ax = b$ where A has the block-tridiagonal form

$$A = \begin{pmatrix} B & T & & & \\ T & B & T & & \\ & T & B & \ddots & \\ & & \ddots & \ddots & T \\ & & & T & B \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{pmatrix}, \tag{10.8.6}$$

where $B, T \in \mathbf{R}^{p\times p}$. We assume that the matrices B and T commute, i.e., $BT = TB$. Then it follows that B and T have a common eigensystem, and we let

$$Q^T B Q = \Lambda = \operatorname{diag}(\lambda_i), \qquad Q^T T Q = \Omega = \operatorname{diag}(\omega_i).$$

The system $Ax = b$ can then be written $Cz = y$, where

$$C = \begin{pmatrix} \Lambda & \Omega & & & \\ \Omega & \Lambda & \Omega & & \\ & \Omega & \Lambda & \ddots & \\ & & \ddots & \ddots & \Omega \\ & & & \Omega & \Lambda \end{pmatrix},$$

and $x_j = Qz_j$, $y_j = Q^T b_j$, $j = 1, \ldots, q$.

For the model problem with Dirichlet boundary conditions the eigenvalues and eigenvectors are known. Furthermore, the multiplication of a vector by Q and Q^T is efficiently obtained by the Fast Fourier Transform algorithm (see Section 9.6.3). One fast algorithm then is as follows:

1. Compute
$$y_j = Q^T b_j, \qquad j = 1, \ldots, q.$$
2. Rearrange taking one element from each vectors y_j,
$$\hat{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iq})^T, \quad i = 1, \ldots, p,$$
and solve by elimination the p systems
$$\Gamma_i \hat{z}_i = \hat{y}_i, \qquad i = 1, \ldots, p,$$
where
$$\Gamma_i = \begin{pmatrix} \lambda_i & \omega_i & & & \\ \omega_i & \lambda_i & \omega_i & & \\ & \omega_i & \lambda_i & \ddots & \\ & & \ddots & \ddots & \omega_i \\ & & & \omega_i & \lambda_i \end{pmatrix}, \quad i = 1, \ldots, p.$$
3. Rearrange (inversely to step 2) taking one element from each $\hat{z}_i$,
$$z_j = (\hat{z}_{j1}, \hat{z}_{j2}, \ldots, \hat{z}_{jq})^T, \quad j = 1, \ldots, q,$$
and compute
$$x_j = Qz_j, \qquad j = 1, \ldots, q.$$

The fast Fourier transforms in steps 1 and 3 takes $\mathcal{O}(n^2 \log n)$ operations. Solving the tridiagonal systems in step 2 only takes $\mathcal{O}(n^2)$ operations, and hence for this step Gaussian elimination is superior to FFT. In all this algorithm uses only $\mathcal{O}(n^2 \log n)$ operations to compute the n^2 unknown components of x.

Review Questions

1. What is meant by preconditioning of a linear system $Ax = b$. What are the properties that a good preconditioner should have?

2. What is meant by an incomplete factorization of a matrix? Describe how incomplete factorizations can be used as preconditioners.

3. Describe two different transformations of a general nonsymmetric linear system $Ax = b$, that allows the transformed system to be solved by the standard conjugate gradient method.

Problems

1. Consider square matrices of order n, with nonzero elements only in the k-th upper diagonal, i.e., of the form

$$T_k = \begin{pmatrix} & t_1 & & \\ & & \ddots & \\ & & & t_{n-k} \\ & & & \end{pmatrix}, \qquad k \geq 0.$$

Show the following rule for *multiplication by diagonals*:

$$A_k B_l = \begin{cases} C_{k+l}, & \text{if } k+l \leq n-1; \\ 0, & \text{otherwise,} \end{cases}$$

where the elements in C_{k+l} are $a_1 b_{k+1}, \ldots, a_{n-k-l} b_{n-l}$.

2. Let B be a symmetric positive definite M-matrix of the form

$$B = \begin{pmatrix} B_1 & -C^T \\ -C & B_2 \end{pmatrix}$$

with B_1 and B_2 square. Show that the Schur complement $S = B_2 - C B_1^{-1} C^T$ of B_1 in B is a symmetric positive definite M-matrix.

3. The penta-diagonal matrix of the model problem has nonzero elements in positions $\mathcal{P}_A = \{(i,j) \,|\, |i-j| = 0, 1, n\}$, which defines a level 0 incomplete factorization. Show that the level 1 incomplete factorization has two extra diagonals corresponding to $|i-j| = n-1$.

4. The triangular solves needed when using an incomplete Cholesky factorizations as a preconditioner are inherently sequential and difficult to vectorize. If the factors are normalized to be unit triangular, then the solution can be computed making use of one of the following expansions

$$(I-L)^{-1} = \begin{cases} I + L + L^2 + L^3 + \cdots & \text{(Neumann expansion)} \\ (I+L)(I+L^2)(I+L^4)\cdots & \text{(Euler expansion)} \end{cases}$$

Verify these expansions and prove that they are finite.

Computer Exercises

1. Let A be a symmetric positive definite matrix. An incomplete Cholesky preconditioner for A is obtained by neglecting elements in places (i,j) prescribed by a symmetric set

$$\mathcal{P} \subset \mathcal{P}_n \equiv \{(i,j) \,|\, 1 \le i,j \le n\},$$

where $(i,j) \in \mathcal{P}$, if $i = j$.

(a) The simplest choice is to take $\mathcal{P}$ equal to the sparsity pattern of A, which for the model problem is $\mathcal{P}_A = \{(i,j) \,|\, |i-j| = 0, 1, n\}$. This is called a level 0 incomplete factorization. A level 1 incomplete factorization is obtained by using the union of $\mathcal{P}_0$ and the pattern of the defect matrix $R = LL^T - A$. Higher level incomplete factorizations are defined in a similar way.

(b) Consider the model problem, where A is block tridiagonal

$$A = \text{tridiag}(-I, T + 2I, -I) \in \mathbf{R}^{n^2 \times n^2}, \qquad T = \text{tridiag}(-1, 2 - 1) \in \mathbf{R}^{n \times n}.$$

Show that A is an M-matrix and hence that an incomplete Cholesky factorizations of A exists?

(c) Write a MATLAB function, which computes the level 0 incomplete Cholesky factor L_0 of A. (You should *NOT* write a general routine like that in the textbook, but an efficient routine using the special five diagonal structure of A!) Implement also the preconditioned conjugate gradient method in MATLAB, and a function which solves $L_0 L_0^T z = r$ by forward and backward substitution. Solve the model problem for $n = 10$, and 20 with and without preconditioning, plot the error norm $\|x - x_k\|_2$, and compare the rate of convergence. Stop the iterations when the recursive residual is of the level of machine precision. Discuss your results!

(d) Take the exact solution to be $x = (1, 1, \ldots, 1, 1)^T$. To investigate the influence of the preconditioner $M = LL^T$ on the spectrum of $M^{-1}A$ do the following. For $n = 10$ plot the eigenvalues of A and of $M^{-1}A$ for level 0 and level 1 preconditioner. You may use, e.g., the built-in MATLAB functions to compute the eigenvalues, and efficiency is not a premium here. (To handle the level 1 preconditioner you need to generalize your incomplete Cholesky routine.)

Notes

We remark that Kaczmarz's method has been rediscovered and used successfully in image reconstruction. In this context the method is known as the unconstrained ART algorithm (algebraic reconstruction technique).

The book by Barret et al. [2, 1994] gives a compact survey of iterative methods and their implementation.

Krylov subspace iteration, which originated with the conjugate gradient method has been named one of the Top 10 Algorithms of the 20th century. The conjugate gradient method was developed independently by E. Stiefel and M. R. Hestenes. Further work was done at the Institute for Numerical Analysis, on the campus of the University of California, in Los Angeles (UCLA). This work was published in 1952 in the seminal paper [13], which has had a tremendous impact in scientific computing. In this paper the author acknowledge cooperation with J. B. Rosser, G. E. Forsythe, and L. Paige, who were working at the institute during this period. They also mention that C. Lanczos has developed a closely related method (see Chapter 9).

Two excellent classical texts on iterative methods for linear systems are Varga [29, 1962] and Young [31, 1971]. Axelsson [1, 1994] is an excellent source of more modern material. For an up to date treatment of iterative methods and preconditioners we recommend Greenbaum [12, 1997], Saad [24, 2003] and H. van der Vorst [28, 2003]. Barret et al. [2, 1994] gives a compact survey of iterative methods and their implementation. (See also available software listed in Chapter 15.) The early history of the conjugate gradient and Lanczos algorithms are are detailed in [10, 1989] and a historical view of iterative methods in general given in [25, 2000]. Krylov solvers are studied by Broyden and Vespucci [3]

Bibliography

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

[2] R. Barret, M. W. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, editors. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, 1993.

[3] Charles Broyden and Maria Theresa Vespucci. *Krylov Solvers for Linear Algebraic Systems*. Studies in Computational Mathematics 11. Elsevier, Amsterdam, 2004.

[4] Gianfranco Cimmino. Calcolo aprossimato per le soluzioni dei sistemi di equazioni lineari. *Ricerca Sci. II*, 9:326–333, 1938.

[5] P. Concus, G. H. Golub, and G. Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Statist. Comput.*, 6:220–252, 1985.

[6] A. de la Garza. An iterative method for solving systems of linear equations. Report K-731, Oak Ridge Gaseous Diffusion Plant, Oak Ridge, TN, 1951.

[7] Roger Fletcher. Conjugate gradient methods for indefinite systems. In G. A. Watson, editor, *Numerical Analysis 1975: Proceedings of the Dundee Conference on Numerical Analysis 1975*, Lecture Notes in Mathematics Vol. 506, pages 73–89, Berlin, 1976. Springer Verlag.

[8] Roland W. Freund, Gene H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1991.

[9] PHilip E. Gill, Walter Murray, D. B. Ponceleón, and Michael A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM J. Matrix. Anal. Appl.*, 13:292–311, 1992.

[10] Gene H. Golub and Dianne O'Leary. Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Review*, 31:50–102, 1989.

[11] Gene H. Golub and Henk A. van der Vorst. Closer to the solution: iterative linear solvers. In I. S. Duff and G. A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 63–92. Clarendon Press, Oxford, 1997.

[12] Anne Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM, Philadelphia, PA, 1997.

[13] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear system. *J. Res. Nat. Bur. Standards, Sect. B*, 49:409–436, 1952.

[14] A. S. Householder and F. L. Bauer. On certain iterative methods for solving linear systems. *Numer. Math.*, 2:55–59, 1960.

[15] S. Kaczmarz. Angenäherte auflösung von systemen linearen gleichungen. *Acad. Polon. Sciences et Lettres*, pages 355–357, 1937.

[16] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards, Sect. B*, 45:255–282, 1950.

[17] Cornelius Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Standards, Sect. B*, 49:33–53, 1952.

[18] L. Landweber. An iterative formula for fredholm integral equations of the first kind. *Amer. J. Math.*, 73:615–624, 1951.

[19] J. A. Meijerink and Henk 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.

[20] N. M. Nachtigal, S. C. Reddy, and Lloyd N. Trefethen. How fast are nonsymmetric matrix iterations? *SIAM J. Matrix. Anal. Appl.*, 13:778–795, 1992.

[21] Christopher C. Paige and Michael Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[22] John K. Reid. A note on the stability of Gaussian elimination. *J. Inst. Maths. Applics.*, 8:374–375, 1971.

[23] Yosef Saad. *Iterative Methods for Sparse Linear Systems.* SIAM Publications., Philadelphia, PA, second edition, 2003.

[24] Yosef Saad and Henk van der Vorst. Iterative solution of linear systems in the 20th century. *J. Comput. Appl. Math.*, 123:1–33, 2000.

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

[26] Henk van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comput..*, 12:631–644, 1992.

[27] Henk van der Vorst. *Iterative Krylov Methods for Large Linear Systems.* Cambridge University Press, Cambridge, UK, 2003.

[28] Richard S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, 1962.

[29] David M. Young. *Iterative methods for solving partial differential equations of elliptic type*. PhD thesis, Harward University, 1950.

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

[31] David M. Young. A historical overview of iterative methods. *Computer Phys. Comm.*, 53:1–17, 1989.

Index