4

Iterative Methods for Solving Linear Systems

Iterative methods formally yield the solution $\mathbf{x}$ of a linear system after an infinite number of steps. At each step they require the computation of the residual of the system. In the case of a full matrix, their computational cost is therefore of the order of n^2 operations for each iteration, to be compared with an overall cost of the order of $\frac{2}{3}n^3$ operations needed by direct methods. Iterative methods can therefore become competitive with direct methods provided the number of iterations that are required to converge (within a prescribed tolerance) is either independent of n or scales sublinearly with respect to n.

In the case of large sparse matrices, as discussed in Section 3.9, direct methods may be unconvenient due to the dramatic fill-in, although extremely efficient direct solvers can be devised on sparse matrices featuring special structures like, for example, those encountered in the approximation of partial differential equations (see Chapters 12 and 13).

Finally, we notice that, when A is ill-conditioned, a combined use of direct and iterative methods is made possible by preconditioning techniques that will be addressed in Section 4.3.2.

4.1 On the Convergence of Iterative Methods

The basic idea of iterative methods is to construct a sequence of vectors $\mathbf{x}^{(k)}$ that enjoy the property of *convergence*

$$\mathbf{x} = \lim_{k\to\infty} \mathbf{x}^{(k)}, \qquad (4.1)$$

where $\mathbf{x}$ is the solution to (3.2). In practice, the iterative process is stopped at the minimum value of n such that $\|\mathbf{x}^{(n)} - \mathbf{x}\| < \varepsilon$, where ε is a fixed tolerance and $\|\cdot\|$ is any convenient vector norm. However, since the exact solution is obviously not available, it is necessary to introduce suitable stopping criteria to monitor the convergence of the iteration (see Section 4.6).

To start with, we consider iterative methods of the form

$$\text{given } \mathbf{x}^{(0)},\ \mathbf{x}^{(k+1)} = \mathrm{B}\mathbf{x}^{(k)} + \mathbf{f}, \quad k \geq 0, \tag{4.2}$$

having denoted by B an $n \times n$ square matrix called the *iteration matrix* and by $\mathbf{f}$ a vector that is obtained from the right hand side $\mathbf{b}$.

Definition 4.1 An iterative method of the form (4.2) is said to be *consistent* with (3.2) if $\mathbf{f}$ and B are such that $\mathbf{x} = \mathrm{B}\mathbf{x} + \mathbf{f}$. Equivalently,

$$\mathbf{f} = (\mathrm{I} - \mathrm{B})\mathrm{A}^{-1}\mathbf{b}.$$

■

Having denoted by

$$\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x} \tag{4.3}$$

the error at the k-th step of the iteration, the condition for convergence (4.1) amounts to requiring that $\lim\limits_{k\to\infty} \mathbf{e}^{(k)} = \mathbf{0}$ for any choice of the initial datum $\mathbf{x}^{(0)}$ (often called the *initial guess*).

Consistency alone does not suffice to ensure the convergence of the iterative method (4.2), as shown in the following example.

Example 4.1 To solve the linear system $2\mathrm{I}\mathbf{x} = \mathbf{b}$, consider the iterative method

$$\mathbf{x}^{(k+1)} = -\mathbf{x}^{(k)} + \mathbf{b},$$

which is obviously consistent. This scheme is not convergent for any choice of the initial guess. If, for instance, $\mathbf{x}^{(0)} = \mathbf{0}$, the method generates the sequence $\mathbf{x}^{(2k)} = \mathbf{0}$, $\mathbf{x}^{(2k+1)} = \mathbf{b}$, $k = 0, 1, \ldots$.

On the other hand, if $\mathbf{x}^{(0)} = \frac{1}{2}\mathbf{b}$ the method is convergent. •

Theorem 4.1 *Let* (4.2) *be a consistent method. Then, the sequence of vectors* $\left\{\mathbf{x}^{(k)}\right\}$ *converges to the solution of* (3.2) *for any choice of* $\mathbf{x}^{(0)}$ *iff* $\rho(\mathrm{B}) < 1$.

Proof. From (4.3) and the consistency assumption, the recursive relation $\mathbf{e}^{(k+1)} = \mathrm{B}\mathbf{e}^{(k)}$ is obtained. Therefore,

$$\mathbf{e}^{(k)} = \mathrm{B}^k\mathbf{e}^{(0)}, \qquad \forall k = 0, 1, \ldots \tag{4.4}$$

Thus, thanks to Theorem 1.5, it follows that $\lim\limits_{k\to\infty} \mathrm{B}^k\mathbf{e}^{(0)} = \mathbf{0}$ for any $\mathbf{e}^{(0)}$ iff $\rho(\mathrm{B}) < 1$.

Conversely, suppose that $\rho(\mathrm{B}) > 1$, then there exists at least one eigenvalue $\lambda(\mathrm{B})$ with module greater than 1. Let $\mathbf{e}^{(0)}$ be an eigenvector associated with λ; then $\mathrm{B}\mathbf{e}^{(0)} = \lambda\mathbf{e}^{(0)}$ and, therefore, $\mathbf{e}^{(k)} = \lambda^k\mathbf{e}^{(0)}$. As a consequence, $\mathbf{e}^{(k)}$ cannot tend to $\mathbf{0}$ as $k \to \infty$, since $|\lambda| > 1$. ◇

From (1.23) and Theorem 1.4 it follows that a sufficient condition for convergence to hold is that $\|\mathrm{B}\| < 1$, for any consistent matrix norm. It is reasonable to expect that the convergence is faster when $\rho(\mathrm{B})$ is smaller so that an estimate of $\rho(\mathrm{B})$ might provide a sound indication of the convergence of the algorithm. Other remarkable quantities in convergence analysis are contained in the following definition.

Definition 4.2 Let B be the iteration matrix. We call:

1. $\|\mathrm{B}^m\|$ the *convergence factor* after m steps of the iteration;
2. $\|\mathrm{B}^m\|^{1/m}$ the *average convergence factor* after m steps;
3. $R_m(\mathrm{B}) = -\frac{1}{m}\log\|\mathrm{B}^m\|$ the *average convergence rate* after m steps.

■

These quantities are too expensive to compute since they require evaluating B^m. Therefore, it is usually preferred to estimate the *asymptotic convergence rate*, which is defined as

$$R(\mathrm{B}) = \lim_{k\to\infty} R_k(\mathrm{B}) = -\log\rho(\mathrm{B}), \tag{4.5}$$

where Property 1.14 has been accounted for. In particular, if B were symmetric, we would have

$$R_m(\mathrm{B}) = -\frac{1}{m}\log\|\mathrm{B}^m\|_2 = -\log\rho(\mathrm{B}).$$

In the case of nonsymmetric matrices, $\rho(\mathrm{B})$ sometimes provides an overoptimistic estimate of $\|\mathrm{B}^m\|^{1/m}$ (see [Axe94], Section 5.1). Indeed, although $\rho(\mathrm{B}) < 1$, the convergence to zero of the sequence $\|\mathrm{B}^m\|$ might be nonmonotone (see Exercise 1). We finally notice that, due to (4.5), $\rho(\mathrm{B})$ is the *asymptotic convergence factor*. Criteria for estimating the quantities defined so far will be addressed in Section 4.6.

Remark 4.1 The iterations introduced in (4.2) are a special instance of iterative methods of the form

$$\mathbf{x}^{(0)} = \mathbf{f}_0(\mathrm{A}, \mathbf{b}),$$

$$\mathbf{x}^{(n+1)} = \mathbf{f}_{n+1}(\mathbf{x}^{(n)}, \mathbf{x}^{(n-1)}, \ldots, \mathbf{x}^{(n-m)}, \mathrm{A}, \mathbf{b}), \text{ for } n \geq m,$$

where $\mathbf{f}_i$ and $\mathbf{x}^{(m)}, \ldots, \mathbf{x}^{(1)}$ are given functions and vectors, respectively. The number of steps which the current iteration depends on is called the *order of the method.* If the functions $\mathbf{f}_i$ are independent of the step index i, the method is called *stationary*, otherwise it is *nonstationary.* Finally, if $\mathbf{f}_i$ depends linearly on $\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(m)}$, the method is called *linear*, otherwise it is *nonlinear.*

In the light of these definitions, the methods considered so far are therefore *stationary linear iterative methods of first order.* In Section 4.3, examples of nonstationary linear methods will be provided. ■

4.2 Linear Iterative Methods

A general technique to devise consistent linear iterative methods is based on an additive *splitting* of the matrix A of the form A=P−N, where P and N are two suitable matrices and P is nonsingular. For reasons that will be clear in the later sections, P is called *preconditioning matrix* or *preconditioner.*

Precisely, given $\mathbf{x}^{(0)}$, one can compute $\mathbf{x}^{(k)}$ for $k \geq 1$, solving the systems

$$\mathrm{P}\mathbf{x}^{(k+1)} = \mathrm{N}\mathbf{x}^{(k)} + \mathbf{b}, \quad k \geq 0. \tag{4.6}$$

The iteration matrix of method (4.6) is $\mathrm{B} = \mathrm{P}^{-1}\mathrm{N}$, while $\mathbf{f} = \mathrm{P}^{-1}\mathbf{b}$. Alternatively, (4.6) can be written in the form

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathrm{P}^{-1}\mathbf{r}^{(k)}, \tag{4.7}$$

where

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathrm{A}\mathbf{x}^{(k)} \tag{4.8}$$

denotes the *residual* vector at step k. Relation (4.7) outlines the fact that a linear system, with coefficient matrix P, must be solved to update the solution at step $k+1$. Thus P, besides being nonsingular, ought to be easily invertible, in order to keep the overall computational cost low. (Notice that, if P were equal to A and N=0, method (4.7) would converge in one iteration, but at the same cost of a direct method).

Let us mention two results that ensure convergence of the iteration (4.7), provided suitable conditions on the splitting of A are fulfilled (for their proof, we refer to [Hac94]).

Property 4.1 *Let* $\mathrm{A} = \mathrm{P} - \mathrm{N}$*, with* A *and* P *symmetric and positive definite. If the matrix* $2\mathrm{P} - \mathrm{A}$ *is positive definite, then the iterative method defined in* (4.7) *is convergent for any choice of the initial datum* $\mathbf{x}^{(0)}$ *and*

$$\rho(\mathrm{B}) = \|\mathrm{B}\|_{\mathrm{A}} = \|\mathrm{B}\|_{\mathrm{P}} < 1.$$

Moreover, the convergence of the iteration is monotone with respect to the norms $\|\cdot\|_{\mathrm{P}}$ *and* $\|\cdot\|_{\mathrm{A}}$ *(i.e.,* $\|\mathbf{e}^{(k+1)}\|_{\mathrm{P}} < \|\mathbf{e}^{(k)}\|_{\mathrm{P}}$ *and* $\|\mathbf{e}^{(k+1)}\|_{\mathrm{A}} < \|\mathbf{e}^{(k)}\|_{\mathrm{A}}$ $k = 0, 1, \ldots$*).*

Property 4.2 *Let* $\mathrm{A} = \mathrm{P} - \mathrm{N}$ *with* A *being symmetric and positive definite. If the matrix* $\mathrm{P} + \mathrm{P}^T - \mathrm{A}$ *is positive definite, then* P *is invertible, the iterative method defined in* (4.7) *is monotonically convergent with respect to norm* $\|\cdot\|_{\mathrm{A}}$ *and* $\rho(\mathrm{B}) \leq \|\mathrm{B}\|_{\mathrm{A}} < 1$*.*

4.2.1 Jacobi, Gauss-Seidel and Relaxation Methods

In this section we consider some classical linear iterative methods.

If the diagonal entries of A are nonzero, we can single out in each equation the corresponding unknown, obtaining the equivalent linear system

$$x_i = \frac{1}{a_{ii}} \left[b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j \right], \qquad i = 1, \ldots, n. \tag{4.9}$$

In the Jacobi method, once an arbitrarily initial guess $\mathbf{x}^{(0)}$ has been chosen, $\mathbf{x}^{(k+1)}$ is computed by the formulae

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right], \; i = 1, \ldots, n. \tag{4.10}$$

This amounts to performing the following splitting for A

$$\mathrm{P} = \mathrm{D},\, \mathrm{N} = \mathrm{D} - \mathrm{A} = \mathrm{E} + \mathrm{F},$$

where D is the diagonal matrix of the diagonal entries of A, E is the lower triangular matrix of entries $e_{ij} = -a_{ij}$ if $i > j$, $e_{ij} = 0$ if $i \leq j$, and F is the upper triangular matrix of entries $f_{ij} = -a_{ij}$ if $j > i$, $f_{ij} = 0$ if $j \leq i$. As a consequence, $\mathrm{A} = \mathrm{D} - (\mathrm{E} + \mathrm{F})$.

The iteration matrix of the Jacobi method is thus given by

$$\mathrm{B}_J = \mathrm{D}^{-1}(\mathrm{E} + \mathrm{F}) = \mathrm{I} - \mathrm{D}^{-1}\mathrm{A}. \tag{4.11}$$

A generalization of the Jacobi method is the over-relaxation method (or JOR), in which, having introduced a relaxation parameter ω, (4.10) is replaced by

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left[b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right] + (1-\omega) x_i^{(k)}, \qquad i = 1, \ldots, n.$$

The corresponding iteration matrix is

$$\mathrm{B}_{J_\omega} = \omega \mathrm{B}_J + (1-\omega)\mathrm{I}. \tag{4.12}$$

In the form (4.7), the JOR method corresponds to

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega \mathrm{D}^{-1} \mathbf{r}^{(k)}.$$

This method is consistent for any $\omega \neq 0$ and for $\omega = 1$ it coincides with the Jacobi method.

The Gauss-Seidel method differs from the Jacobi method in the fact that at the $k+1$-th step the available values of $x_i^{(k+1)}$ are being used to update the solution, so that, instead of (4.10), one has

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right], \; i = 1, \ldots, n. \quad (4.13)$$

This method amounts to performing the following splitting for A

$$\mathrm{P} = \mathrm{D} - \mathrm{E}, \; \mathrm{N} = \mathrm{F},$$

and the associated iteration matrix is

$$\mathrm{B}_{GS} = (\mathrm{D} - \mathrm{E})^{-1}\mathrm{F}. \quad (4.14)$$

Starting from Gauss-Seidel method, in analogy to what was done for Jacobi iterations, we introduce the successive over-relaxation method (or SOR method)

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right] + (1-\omega) x_i^{(k)}, \quad (4.15)$$

for $i = 1, \ldots, n$. The method (4.15) can be written in vector form as

$$(\mathrm{I} - \omega \mathrm{D}^{-1}\mathrm{E})\mathbf{x}^{(k+1)} = [(1-\omega)\mathrm{I} + \omega \mathrm{D}^{-1}\mathrm{F}]\mathbf{x}^{(k)} + \omega \mathrm{D}^{-1}\mathbf{b}, \quad (4.16)$$

from which the iteration matrix is

$$\mathrm{B}(\omega) = (\mathrm{I} - \omega \mathrm{D}^{-1}\mathrm{E})^{-1}[(1-\omega)\mathrm{I} + \omega \mathrm{D}^{-1}\mathrm{F}]. \quad (4.17)$$

Multiplying by D both sides of (4.16) and recalling that $\mathrm{A} = \mathrm{D} - (\mathrm{E} + \mathrm{F})$ yields the following form (4.7) of the SOR method

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \left(\frac{1}{\omega}\mathrm{D} - \mathrm{E} \right)^{-1} \mathbf{r}^{(k)}.$$

It is consistent for any $\omega \neq 0$ and for $\omega = 1$ it coincides with Gauss-Seidel method. In particular, if $\omega \in (0, 1)$ the method is called under-relaxation, while if $\omega > 1$ it is called over-relaxation.

4.2.2 Convergence Results for Jacobi and Gauss-Seidel Methods

There exist special classes of matrices for which it is possible to state a priori some convergence results for the methods examined in the previous section. The first result in this direction is the following.

Theorem 4.2 *If* A *is a strictly diagonally dominant matrix by rows, the Jacobi and Gauss-Seidel methods are convergent.*

Proof. Let us prove the part of the theorem concerning the Jacobi method, while for the Gauss-Seidel method we refer to [Axe94]. Since A is strictly diagonally dominant by rows, $|a_{ii}| > \sum_{j=1}^{n} |a_{ij}|$ for $j \neq i$ and $i = 1, \ldots, n$. As a consequence, $\|\mathrm{B}_J\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1, j\neq i}^{n} |a_{ij}|/|a_{ii}| < 1$, so that the Jacobi method is convergent. ◇

Theorem 4.3 *If* A *and* 2D − A *are symmetric and positive definite matrices, then the Jacobi method is convergent and* $\rho(\mathrm{B}_J) = \|\mathrm{B}_J\|_\mathrm{A} = \|\mathrm{B}_J\|_\mathrm{D}$.

Proof. The theorem follows from Property 4.1 taking P=D. ◇

In the case of the JOR method, the assumption on 2D − A can be removed, yielding the following result.

Theorem 4.4 *If* A *is symmetric positive definite, then the JOR method is convergent if* $0 < \omega < 2/\rho(\mathrm{D}^{-1}\mathrm{A})$.

Proof. The result immediately follows from (4.12) and noting that A has real positive eigenvalues. ◇

Concerning the Gauss-Seidel method, the following result holds.

Theorem 4.5 *If* A *is symmetric positive definite, the Gauss-Seidel method is monotonically convergent with respect to the norm* $\|\cdot\|_\mathrm{A}$.

Proof. We can apply Property 4.2 to the matrix P=D−E, upon checking that $\mathrm{P} + \mathrm{P}^T - \mathrm{A}$ is positive definite. Indeed

$$\mathrm{P} + \mathrm{P}^T - \mathrm{A} = 2\mathrm{D} - \mathrm{E} - \mathrm{F} - \mathrm{A} = \mathrm{D},$$

having observed that $(\mathrm{D} - \mathrm{E})^T = \mathrm{D} - \mathrm{F}$. We conclude by noticing that D is positive definite, since it is the diagonal of A. ◇

Finally, if A is tridiagonal (or block tridiagonal), it can be shown that

$$\rho(\mathrm{B}_{GS}) = \rho^2(\mathrm{B}_J) \tag{4.18}$$

(see [You71] for the proof). From (4.18) we can conclude that both methods converge or fail to converge at the same time. In the former case, the Gauss-Seidel method is more rapidly convergent than the Jacobi method, and the asymptotic convergence rate of the Gauss-Seidel method is twice than that of the Jacobi method. In particular, if A is tridiagonal and symmetric positive definite, Theorem 4.5 implies convergence of the Gauss-Seidel method, and (4.18) ensures convergence also for the Jacobi method.

Relation (4.18) holds even if A enjoys the following *A-property.*

Definition 4.3 A *consistently ordered* matrix $\mathrm{M} \in \mathbb{R}^{n\times n}$ (that is, a matrix such that $\alpha\mathrm{D}^{-1}\mathrm{E} + \alpha^{-1}\mathrm{D}^{-1}\mathrm{F}$, for $\alpha \neq 0$, has eigenvalues that do not depend on α, where $\mathrm{M} = \mathrm{D} - \mathrm{E} - \mathrm{F}$, $\mathrm{D} = \mathrm{diag}(m_{11}, \ldots, m_{nn})$, E and F are strictly lower and upper triangular matrices, respectively) enjoys the *A*-property if it can be partitioned in the 2×2 block form

$$\mathrm{M} = \begin{bmatrix} \tilde{\mathrm{D}}_1 & \mathrm{M}_{12} \\ \mathrm{M}_{21} & \tilde{\mathrm{D}}_2 \end{bmatrix},$$

where $\tilde{\mathrm{D}}_1$ and $\tilde{\mathrm{D}}_2$ are diagonal matrices. ■

When dealing with general matrices, no a priori conclusions on the convergence properties of the Jacobi and Gauss-Seidel methods can be drawn, as shown in Example 4.2.

Example 4.2 Consider the 3×3 linear systems of the form $\mathrm{A}_i\mathbf{x} = \mathbf{b}_i$, where $\mathbf{b}_i$ is always taken in such a way that the solution of the system is the unit vector, and the matrices A_i are

$$\mathrm{A}_1 = \begin{bmatrix} 3 & 0 & 4 \\ 7 & 4 & 2 \\ -1 & 1 & 2 \end{bmatrix}, \qquad \mathrm{A}_2 = \begin{bmatrix} -3 & 3 & -6 \\ -4 & 7 & -8 \\ 5 & 7 & -9 \end{bmatrix},$$

$$\mathrm{A}_3 = \begin{bmatrix} 4 & 1 & 1 \\ 2 & -9 & 0 \\ 0 & -8 & -6 \end{bmatrix}, \qquad \mathrm{A}_4 = \begin{bmatrix} 7 & 6 & 9 \\ 4 & 5 & -4 \\ -7 & -3 & 8 \end{bmatrix}.$$

It can be checked that the Jacobi method does fail to converge for A_1 ($\rho(\mathrm{B}_J) = 1.33$), while the Gauss-Seidel scheme is convergent. Conversely, in the case of A_2, the Jacobi method is convergent, while the Gauss-Seidel method fails to converge ($\rho(\mathrm{B}_{GS}) = 1.\bar{1}$). In the remaining two cases, the Jacobi method is more slowly convergent than the Gauss-Seidel method for matrix A_3 ($\rho(\mathrm{B}_J) = 0.44$ against $\rho(\mathrm{B}_{GS}) = 0.018$), and the converse is true for A_4 ($\rho(\mathrm{B}_J) = 0.64$ while $\rho(\mathrm{B}_{GS}) = 0.77$). •

We conclude the section with the following result.

Theorem 4.6 *If the Jacobi method is convergent, then the JOR method converges if* $0 < \omega \leq 1$.

Proof. From (4.12) we obtain that the eigenvalues of B_{J_ω} are

$$\mu_k = \omega\lambda_k + 1 - \omega, \qquad k = 1, \ldots, n,$$

where λ_k are the eigenvalues of B_J. Then, recalling the Euler formula for the representation of a complex number, we let $\lambda_k = r_k e^{i\theta_k}$ and get

$$|\mu_k|^2 = \omega^2 r_k^2 + 2\omega r_k \cos(\theta_k)(1 - \omega) + (1 - \omega)^2 \leq (\omega r_k + 1 - \omega)^2,$$

which is less than 1 if $0 < \omega \leq 1$. ◇

4.2.3 Convergence Results for the Relaxation Method

The following result provides a necessary condition on ω in order the SOR method to be convergent.

Theorem 4.7 *For any $\omega \in \mathbb{R}$ we have $\rho(\mathrm{B}(\omega)) \geq |\omega - 1|$; therefore, the SOR method fails to converge if $\omega \leq 0$ or $\omega \geq 2$.*

Proof. If $\{\lambda_i\}$ denote the eigenvalues of the SOR iteration matrix, then

$$\left|\prod_{i=1}^{n} \lambda_i\right| = \left|\det\left[(1-\omega)\mathrm{I} + \omega \mathrm{D}^{-1}\mathrm{F}\right]\right| = |1-\omega|^n.$$

Therefore, at least one eigenvalue λ_i must exist such that $|\lambda_i| \geq |1-\omega|$ and thus, in order for convergence to hold, we must have $|1-\omega| < 1$, that is $0 < \omega < 2$. $\diamond$

Assuming that A is symmetric and positive definite, the condition $0 < \omega < 2$, besides being necessary, becomes also sufficient for convergence. Indeed the following result holds (for the proof, see [Hac94]).

Property 4.3 (Ostrowski) *If* A *is symmetric and positive definite, then the SOR method is convergent iff $0 < \omega < 2$. Moreover, its convergence is monotone with respect to $\|\cdot\|_{\mathrm{A}}$.*

Finally, if A is *strictly diagonally dominant* by rows, the SOR method converges if $0 < \omega \leq 1$.

The results above show that the SOR method is more or less rapidly convergent, depending on the choice of the relaxation parameter ω. The question of how to determine the value ω_{opt} for which the convergence rate is the highest possible can be given a satisfactory answer only in special cases (see, for instance, [Axe94], [You71], [Var62] or [Wac66]). Here we limit ourselves to quoting the following result (whose proof is in [Axe94]).

Property 4.4 *If the matrix* A *enjoys the A-property and if* B_J *has real eigenvalues, then the SOR method converges for any choice of* $\mathbf{x}^{(0)}$ *iff* $\rho(\mathrm{B}_J) < 1$ *and* $0 < \omega < 2$. *Moreover,*

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho^2(\mathrm{B}_J)}} \tag{4.19}$$

and the corresponding asymptotic convergence factor is

$$\rho(\mathrm{B}(\omega_{opt})) = \frac{1 - \sqrt{1 - \rho^2(\mathrm{B}_J)}}{1 + \sqrt{1 - \rho^2(\mathrm{B}_J)}}.$$

4.2.4 A priori Forward Analysis

In the previous analysis we have neglected the rounding errors. However, as shown in the following example (taken from [HW76]), they can dramatically affect the convergence rate of the iterative method.

Example 4.3 Let A be a lower bidiagonal matrix of order 100 with entries $a_{ii} = 1.5$ and $a_{i,i-1} = 1$, and let $\mathbf{b} \in \mathbb{R}^{100}$ be the right-side with $b_i = 2.5$. The exact solution of the system $\mathrm{A}\mathbf{x} = \mathbf{b}$ has components $x_i = 1-(-2/3)^i$. The SOR method with $\omega = 1.5$ should be convergent, working in exact arithmetic, since $\rho(\mathrm{B}(1.5)) = 0.5$ (far below one). However, running Program 16 with $\mathbf{x}^{(0)} = fl(\mathbf{x}) + \epsilon_M$, which is extremely close to the exact value, the sequence $\mathbf{x}^{(k)}$ diverges and after 100 iterations the algorithm yields a solution with $\|\mathbf{x}^{(100)}\|_\infty = 10^{13}$. The flaw is due to rounding error propagation and must not be ascribed to a possible ill-conditioning of the matrix since $K_\infty(\mathrm{A}) \simeq 5$. •

To account for rounding errors, let us denote by $\widehat{\mathbf{x}}^{(k+1)}$ the solution (in finite arithmetic) generated by an iterative method of the form (4.6) after k steps. Due to rounding errors, $\widehat{\mathbf{x}}^{(k+1)}$ can be regarded as the exact solution to the problem

$$\mathrm{P}\widehat{\mathbf{x}}^{(k+1)} = \mathrm{N}\widehat{\mathbf{x}}^{(k)} + \mathbf{b} - \boldsymbol{\zeta}_k, \tag{4.20}$$

with

$$\boldsymbol{\zeta}_k = \delta\mathrm{P}_{k+1}\widehat{\mathbf{x}}^{(k+1)} - \mathbf{g}_k.$$

The matrix $\delta\mathrm{P}_{k+1}$ accounts for the rounding errors in the solution of (4.6), while the vector $\mathbf{g}_k$ includes the errors made in the evaluation of $\mathrm{N}\widehat{\mathbf{x}}^{(k)} + \mathbf{b}$. From (4.20), we obtain

$$\widehat{\mathbf{x}}^{(k+1)} = \mathrm{B}^{k+1}\mathbf{x}^{(0)} + \sum_{j=0}^{k} \mathrm{B}^j\mathrm{P}^{-1}(\mathbf{b} - \boldsymbol{\zeta}_{k-j})$$

and for the absolute error $\widehat{\mathbf{e}}^{(k+1)} = \mathbf{x} - \widehat{\mathbf{x}}^{(k+1)}$

$$\widehat{\mathbf{e}}^{(k+1)} = \mathrm{B}^{k+1}\mathbf{e}^{(0)} + \sum_{j=0}^{k} \mathrm{B}^j\mathrm{P}^{-1}\boldsymbol{\zeta}_{k-j}.$$

The first term represents the error that is made by the iterative method in exact arithmetic; if the method is convergent, this error is negligible for sufficiently large values of k. The second term refers instead to rounding error propagation; its analysis is quite technical and is carried out, for instance, in [Hig88] in the case of Jacobi, Gauss-Seidel and SOR methods.

4.2.5 Block Matrices

The methods of the previous sections are also referred to as *point* (or *line*) iterative methods, since they act on single entries of matrix A. It is possible to devise *block* versions of the algorithms, provided that D denotes the block diagonal matrix whose entries are the $m \times m$ diagonal blocks of matrix A (see Section 1.6).

The *block Jacobi method* is obtained taking again P=D and N=D-A. The method is well-defined only if the diagonal blocks of D are nonsingular. If A is decomposed in $p \times p$ square blocks, the block Jacobi method is

$$\mathrm{A}_{ii}\mathbf{x}_i^{(k+1)} = \mathbf{b}_i - \sum_{\substack{j=1 \\ j\neq i}}^{p} \mathrm{A}_{ij}\mathbf{x}_j^{(k)},\ i = 1, \ldots, p,$$

having also decomposed the solution vector and the right side in blocks of size p, denoted by $\mathbf{x}_i$ and $\mathbf{b}_i$, respectively. As a result, at each step, the block Jacobi method requires solving p linear systems of matrices A_{ii}. Theorem 4.3 is still valid, provided that D is substituted by the corresponding block diagonal matrix.

In a similar manner, the block Gauss-Seidel and block SOR methods can be introduced.

4.2.6 Symmetric Form of the Gauss-Seidel and SOR Methods

Even if A is a symmetric matrix, the Gauss-Seidel and SOR methods generate iteration matrices that are not necessarily symmetric. For that, we introduce in this section a technique that allows for symmetrizing these schemes. The final aim is to provide an approach for generating symmetric preconditioners (see Section 4.3.2).

Firstly, let us remark that an analogue of the Gauss-Seidel method can be constructed, by simply exchanging E with F. The following iteration can thus be defined, called the *backward Gauss-Seidel method*

$$(\mathrm{D} - \mathrm{F})\mathbf{x}^{(k+1)} = \mathrm{E}\mathbf{x}^{(k)} + \mathbf{b},$$

with iteration matrix given by $\mathrm{B}_{GSb} = (\mathrm{D} - \mathrm{F})^{-1}\mathrm{E}$.
The *symmetric Gauss-Seidel method* is obtained by combining an iteration of Gauss-Seidel method with an iteration of backward Gauss-Seidel method. Precisely, the k-th iteration of the symmetric Gauss-Seidel method is

$$(\mathrm{D} - \mathrm{E})\mathbf{x}^{(k+1/2)} = \mathrm{F}\mathbf{x}^{(k)} + \mathbf{b}, \qquad (\mathrm{D} - \mathrm{F})\mathbf{x}^{(k+1)} = \mathrm{E}\mathbf{x}^{(k+1/2)} + \mathbf{b}.$$

Eliminating $\mathbf{x}^{(k+1/2)}$, the following scheme is obtained

$$\begin{aligned}
\mathbf{x}^{(k+1)} &= \mathrm{B}_{SGS}\mathbf{x}^{(k)} + \mathbf{b}_{SGS},\\
\mathrm{B}_{SGS} &= (\mathrm{D} - \mathrm{F})^{-1}\mathrm{E}(\mathrm{D} - \mathrm{E})^{-1}\mathrm{F},\\
\mathbf{b}_{SGS} &= (\mathrm{D} - \mathrm{F})^{-1}[\mathrm{E}(\mathrm{D} - \mathrm{E})^{-1} + \mathrm{I}]\mathbf{b}.
\end{aligned} \tag{4.21}$$

The preconditioning matrix associated with (4.21) is

$$\mathrm{P}_{SGS} = (\mathrm{D} - \mathrm{E})\mathrm{D}^{-1}(\mathrm{D} - \mathrm{F}).$$

The following result can be proved (see [Hac94]).

Property 4.5 *If* A *is a symmetric positive definite matrix, the symmetric Gauss-Seidel method is convergent, and, moreover,* B_{SGS} *is symmetric positive definite.*

In a similar manner, defining the backward SOR method

$$(\mathrm{D} - \omega\mathrm{F})\mathbf{x}^{(k+1)} = \left[\omega\mathrm{E} + (1-\omega)\mathrm{D}\right]\mathbf{x}^{(k)} + \omega\mathbf{b},$$

and combining it with a step of SOR method, the following *symmetric SOR method* or *SSOR*, is obtained

$$\mathbf{x}^{(k+1)} = \mathrm{B}_s(\omega)\mathbf{x}^{(k)} + \mathbf{b}_\omega,$$

where

$$\mathrm{B}_s(\omega) = (\mathrm{D} - \omega\mathrm{F})^{-1}(\omega\mathrm{E} + (1-\omega)\mathrm{D})(\mathrm{D} - \omega\mathrm{E})^{-1}(\omega\mathrm{F} + (1-\omega)\mathrm{D}),$$

$$\mathbf{b}_\omega = \omega(2-\omega)(\mathrm{D} - \omega\mathrm{F})^{-1}\mathrm{D}(\mathrm{D} - \omega\mathrm{E})^{-1}\mathbf{b}.$$

The preconditioning matrix of this scheme is

$$\mathrm{P}_{SSOR}(\omega) = \left(\frac{1}{\omega}\mathrm{D} - \mathrm{E}\right)\frac{\omega}{2-\omega}\mathrm{D}^{-1}\left(\frac{1}{\omega}\mathrm{D} - \mathrm{F}\right). \tag{4.22}$$

If A is symmetric and positive definite, the SSOR method is convergent if $0 < \omega < 2$ (see [Hac94] for the proof). Typically, the SSOR method with an optimal choice of the relaxation parameter converges more slowly than the corresponding SOR method. However, the value of $\rho(\mathrm{B}_s(\omega))$ is less sensitive to a choice of ω around the optimal value (in this respect, see the behavior of the spectral radii of the two iteration matrices in Figure 4.1). For this reason, the optimal value of ω that is chosen in the case of SSOR method is usually the same used for the SOR method (for further details, we refer to [You71]).

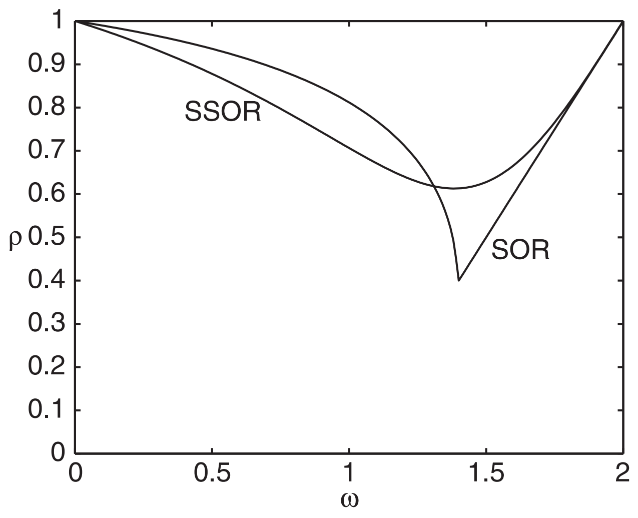

Fig. 4.1. Spectral radius of the iteration matrix of SOR and SSOR methods, as a function of the relaxation parameter ω for the matrix $\mathrm{tridiag}_{10}(-1, 2, -1)$

4.2.7 Implementation Issues

We provide the programs implementing the Jacobi and Gauss-Seidel methods in their point form and with relaxation.

In Program 15 the JOR method is implemented (the Jacobi method is obtained as a special case setting `omega` = 1). The stopping test monitors the Euclidean norm of the residual at each iteration, normalized to the value of the initial residual.
Notice that each component `x(i)` of the solution vector can be computed independently; this method can thus be easily parallelized.

Program 15 - jor : JOR method

```
function [x,iter]=jor(A,b,x0,nmax,tol,omega)
%JOR JOR method
%  [X,ITER]=JOR(A,B,X0,NMAX,TOL,OMEGA) attempts to solve the system
%  A*X=B with the JOR method. TOL specifies the tolerance of the method.
%  NMAX specifies the maximum number of iterations. X0 specifies the initial
%  guess. OMEGA is the relaxation parameter. ITER is the iteration number at
%  which X is computed.
[n,m]=size(A);
if n ~= m, error('Only square systems'); end
iter=0;
r = b-A*x0; r0=norm(r); err=norm(r); x=x0;
while err > tol & iter < nmax
   iter = iter + 1;
   for i=1:n
      s = 0;
      for j = 1:i-1, s=s+A(i,j)*x(j);   end
      for j = i+1:n, s=s+A(i,j)*x(j);   end
      xnew(i,1)=omega*(b(i)-s)/A(i,i)+(1-omega)*x(i);
   end
   x=xnew;  r=b-A*x;  err=norm(r)/r0;
end
return
```

Program 16 implements the SOR method. Taking `omega=1` yields the Gauss-Seidel method.
Unlike the Jacobi method, this scheme is fully sequential. However, it can be efficiently implemented without storing the solution of the previous step, with a saving of memory storage.

Program 16 - sor : SOR method

```
function [x,iter]=sor(A,b,x0,nmax,tol,omega)
%SOR SOR method
%  [X,ITER]=SOR(A,B,X0,NMAX,TOL,OMEGA) attempts to solve the system
%  A*X=B with the SOR method. TOL specifies the tolerance of the method.
% NMAX specifies the maximum number of iterations. X0 specifies the initial
% guess. OMEGA is the relaxation parameter. ITER is the iteration number at
% which X is computed.
[n,m]=size(A);
if n ~= m, error('Only square systems'); end
iter=0; r=b-A*x0; r0=norm(r); err=norm(r); xold=x0;
while err > tol & iter < nmax
   iter = iter + 1;
   for i=1:n
      s=0;
      for j = 1:i-1, s=s+A(i,j)*x(j); end
      for j = i+1:n, s=s+A(i,j)*xold(j); end
      x(i,1)=omega*(b(i)-s)/A(i,i)+(1-omega)*xold(i);
   end
   xold=x;  r=b-A*x; err=norm(r)/r0;
end
return
```

4.3 Stationary and Nonstationary Iterative Methods

Denote by

$$\mathrm{R_P} = \mathrm{I} - \mathrm{P}^{-1}\mathrm{A}$$

the iteration matrix associated with (4.7). Proceeding as in the case of relaxation methods, (4.7) can be generalized introducing a relaxation (or acceleration) parameter α. This leads to the following *stationary Richardson method*

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathrm{P}^{-1}\mathbf{r}^{(k)}, \qquad k \geq 0. \tag{4.23}$$

More generally, allowing α to depend on the iteration index, the *nonstationary Richardson method* or *semi-iterative method* is given by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathrm{P}^{-1}\mathbf{r}^{(k)}, \qquad k \geq 0. \tag{4.24}$$

The iteration matrix at the k-th step for (4.24) (depending on k) is

$$\mathrm{R}_{\alpha_k} = \mathrm{I} - \alpha_k \mathrm{P}^{-1}\mathrm{A},$$

with $\alpha_k = \alpha$ in the stationary case. If $\mathrm{P} = \mathrm{I}$, the family of methods (4.24) will be called *nonpreconditioned*. The Jacobi and Gauss-Seidel methods can be regarded as stationary Richardson methods with $\mathrm{P} = \mathrm{D}$ and $\mathrm{P} = \mathrm{D} - \mathrm{E}$, respectively (and $\alpha = 1$ in both cases).

We can rewrite (4.24) (and, thus, also (4.23)) in a form of greater interest for computation. Letting $\mathbf{z}^{(k)} = \mathrm{P}^{-1}\mathbf{r}^{(k)}$ (the so-called *preconditioned residual*), we get $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)}$ and $\mathbf{r}^{(k+1)} = \mathbf{b} - \mathrm{A}\mathbf{x}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathrm{A}\mathbf{z}^{(k)}$.

To summarize, a nonstationary Richardson method requires at each $k+1$-th step the following operations:

$$\begin{aligned}
&\text{solve the linear system } \mathrm{P}\mathbf{z}^{(k)} = \mathbf{r}^{(k)},\\
&\text{compute the acceleration parameter } \alpha_k,\\
&\text{update the solution } \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)},\\
&\text{update the residual } \mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathrm{A}\mathbf{z}^{(k)}.
\end{aligned} \tag{4.25}$$

4.3.1 Convergence Analysis of the Richardson Method

Let us first consider the stationary Richardson methods for which $\alpha_k = \alpha$ for $k \geq 0$. The following convergence result holds.

Theorem 4.8 *For any nonsingular matrix* P*, the stationary Richardson method* (4.23) *is convergent iff*

$$\frac{2\mathrm{Re}\lambda_i}{\alpha|\lambda_i|^2} > 1 \;\forall i = 1, \ldots, n, \tag{4.26}$$

where $\lambda_i \in \mathbb{C}$ are the eigenvalues of $\mathrm{P}^{-1}\mathrm{A}$*.*

Proof. Let us apply Theorem 4.1 to the iteration matrix $\mathrm{R}_\alpha = \mathrm{I} - \alpha \mathrm{P}^{-1}\mathrm{A}$. The condition $|1 - \alpha\lambda_i| < 1$ for $i = 1, \ldots, n$ yields the inequality

$$(1 - \alpha \mathrm{Re}\lambda_i)^2 + \alpha^2 (\mathrm{Im}\lambda_i)^2 < 1,$$

from which (4.26) immediately follows. ◇

Let us notice that, if the sign of the real parts of the eigenvalues of $\mathrm{P}^{-1}\mathrm{A}$ is not constant, the stationary Richardson method *cannot* converge.
More specific results can be obtained provided that suitable assumptions are made on the spectrum of $\mathrm{P}^{-1}\mathrm{A}$.

Theorem 4.9 *Assume that* P *is a nonsingular matrix and that* $\mathrm{P}^{-1}\mathrm{A}$ *has positive real eigenvalues, ordered in such a way that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0$. Then, the stationary Richardson method* (4.23) *is convergent iff* $0 < \alpha < 2/\lambda_1$*. Moreover, letting*

$$\alpha_{opt} = \frac{2}{\lambda_1 + \lambda_n}, \tag{4.27}$$

the spectral radius of the iteration matrix R_α *is minimum if $\alpha = \alpha_{opt}$, with*

$$\rho_{opt} = \min_\alpha \left[\rho(\mathrm{R}_\alpha)\right] = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n}. \tag{4.28}$$

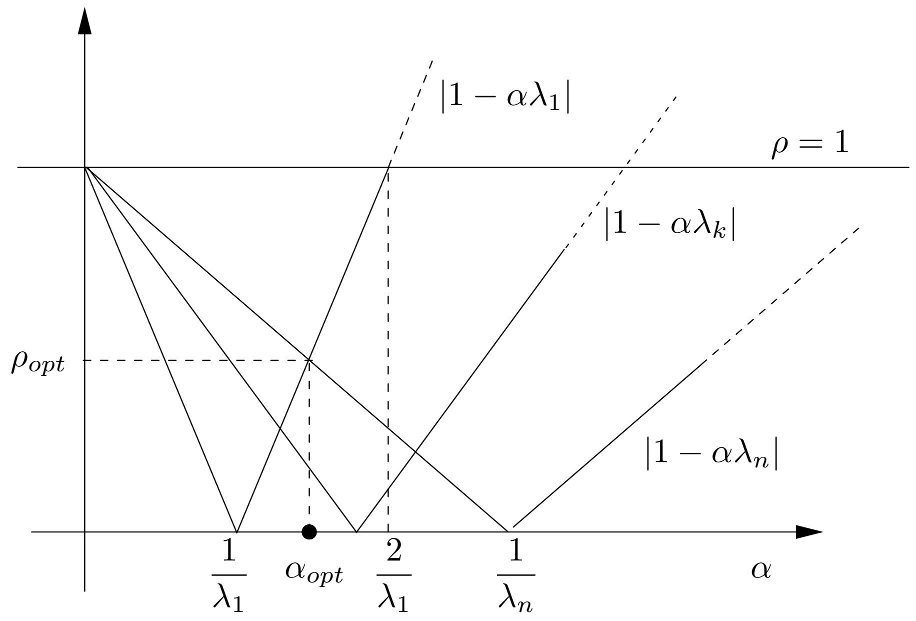

Fig. 4.2. Spectral radius of R_α as a function of the eigenvalues of $\mathrm{P}^{-1}\mathrm{A}$

Proof. The eigenvalues of R_α are given by $\lambda_i(\mathrm{R}_\alpha) = 1 - \alpha\lambda_i$, so that (4.23) is convergent iff $|\lambda_i(\mathrm{R}_\alpha)| < 1$ for $i = 1, \ldots, n$, that is, if $0 < \alpha < 2/\lambda_1$. It follows (see Figure 4.2) that $\rho(\mathrm{R}_\alpha)$ is minimum when $1 - \alpha\lambda_n = \alpha\lambda_1 - 1$, that is, for $\alpha = 2/(\lambda_1 + \lambda_n)$, which furnishes the desired value for α_{opt}. By substitution, the desired value of ρ_{opt} is obtained. ◇

If $\mathrm{P}^{-1}\mathrm{A}$ is symmetric positive definite, it can be shown that the convergence of the Richardson method is monotone with respect to either $\|\cdot\|_2$ and $\|\cdot\|_\mathrm{A}$. In such a case, using (4.28), we can also relate ρ_{opt} to $K_2(\mathrm{P}^{-1}\mathrm{A})$ as follows

$$\rho_{opt} = \frac{K_2(\mathrm{P}^{-1}\mathrm{A}) - 1}{K_2(\mathrm{P}^{-1}\mathrm{A}) + 1}, \ \alpha_{opt} = \frac{2\|\mathrm{A}^{-1}\mathrm{P}\|_2}{K_2(\mathrm{P}^{-1}\mathrm{A}) + 1}. \tag{4.29}$$

The choice of a suitable preconditioner P is, therefore, of paramount importance for improving the convergence of a Richardson method. Of course, such a choice should also account for the need of keeping the computational effort as low as possible. In Section 4.3.2, some preconditioners of common use in practice will be described.

Corollary 1 *Assume that* A *is a symmetric positive definite matrix with eigenvalues* $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. *Then, if* $0 < \alpha < 2/\lambda_1$, *the nonpreconditioned stationary Richardson method is convergent and*

$$\|\mathbf{e}^{(k+1)}\|_\mathrm{A} \leq \rho(\mathrm{R}_\alpha)\|\mathbf{e}^{(k)}\|_\mathrm{A}, \quad k \geq 0. \tag{4.30}$$

The same result holds for the preconditioned Richardson method, provided that the matrices P, A *and* $\mathrm{P}^{-1}\mathrm{A}$ *are symmetric positive definite.*

Proof. The convergence is a consequence of Theorem 4.8. Moreover, we notice that

$$\|\mathbf{e}^{(k+1)}\|_\mathrm{A} = \|\mathrm{R}_\alpha\mathbf{e}^{(k)}\|_\mathrm{A} = \|\mathrm{A}^{1/2}\mathrm{R}_\alpha\mathbf{e}^{(k)}\|_2 \leq \|\mathrm{A}^{1/2}\mathrm{R}_\alpha\mathrm{A}^{-1/2}\|_2\|\mathrm{A}^{1/2}\mathbf{e}^{(k)}\|_2.$$

The matrix R_α is symmetric positive definite and is similar to $\mathrm{A}^{1/2}\mathrm{R}_\alpha\mathrm{A}^{-1/2}$. Therefore,

$$\|\mathrm{A}^{1/2}\mathrm{R}_\alpha\mathrm{A}^{-1/2}\|_2 = \rho(\mathrm{R}_\alpha).$$

The result (4.30) follows by noting that $\|\mathrm{A}^{1/2}\mathbf{e}^{(k)}\|_2 = \|\mathbf{e}^{(k)}\|_\mathrm{A}$. A similar proof can be carried out in the preconditioned case, provided we replace A with $\mathrm{P}^{-1}\mathrm{A}$. $\diamond$

Finally, the inequality (4.30) holds even if only P and A are symmetric positive definite (for the proof, see [QV94], Chapter 2).

4.3.2 Preconditioning Matrices

All the methods introduced in the previous sections can be cast in the form (4.2), so that they can be regarded as being methods for solving the system

$$(\mathrm{I} - \mathrm{B})\mathbf{x} = \mathbf{f} = \mathrm{P}^{-1}\mathbf{b}.$$

On the other hand, since $\mathrm{B}=\mathrm{P}^{-1}\mathrm{N}$, system (3.2) can be equivalently reformulated as

$$\mathrm{P}^{-1}\mathrm{A}\mathbf{x} = \mathrm{P}^{-1}\mathbf{b}. \tag{4.31}$$

The latter is the *preconditioned system*, being P the *preconditioning matrix* or *left preconditioner*. *Right* and *centered* preconditioners can be introduced as well, if system (3.2) is transformed, respectively, as

$$\mathrm{A}\mathrm{P}^{-1}\mathbf{y} = \mathbf{b},\ \mathbf{y} = \mathrm{P}\mathbf{x},$$

or

$$\mathrm{P}_L^{-1}\mathrm{A}\mathrm{P}_R^{-1}\mathbf{y} = \mathrm{P}_L^{-1}\mathbf{b},\ \mathbf{y} = \mathrm{P}_R\mathbf{x}.$$

There are *point preconditioners* and *block preconditioners*, depending on whether they are applied to the single entries of A or to the blocks of a partition of A. The iterative methods considered so far correspond to fixed-point iterations on a left-preconditioned system. As stressed by (4.25), computing the inverse of P is not mandatory; actually, the role of P is to "precondition" the residual $\mathbf{r}^{(k)}$ through the solution of the additional system $\mathrm{P}\mathbf{z}^{(k)} = \mathbf{r}^{(k)}$.

Since the preconditioner acts on the spectral radius of the iteration matrix, it would be useful to pick up, for a given linear system, an *optimal preconditioner*, i.e., a preconditioner which is able to make the number of iterations required for convergence independent of the size of the system. Notice that the choice P=A is optimal but, trivially, "inefficient"; some alternatives of greater computational interest will be examined below.

There is not a general roadmap to devise optimal preconditioners. However, an established "rule of thumb" is that P is a good preconditioner for A if $\mathrm{P}^{-1}\mathrm{A}$ is near to being a normal matrix and if its eigenvalues are clustered within a sufficiently small region of the complex field. The choice of a preconditioner must also be guided by practical considerations, noticeably, its computational cost and its memory requirements.

Preconditioners can be divided into two main categories: algebraic and functional preconditioners, the difference being that the algebraic preconditioners are independent of the problem that originated the system to be solved, and are actually constructed via algebraic procedures, while the functional preconditioners take advantage of the knowledge of the problem and are constructed as a function of it. In addition to the preconditioners already introduced in Section 4.2.6, we give a description of other algebraic preconditioners of common use.

1. *Diagonal preconditioners*: choosing P as the diagonal of A is generally effective if A is symmetric positive definite. A usual choice in the nonsymmetric case is to set

$$p_{ii} = \left(\sum_{j=1}^{n} a_{ij}^2 \right)^{1/2} .$$

   Block diagonal preconditioners can be constructed in a similar manner. We remark that devising an optimal diagonal preconditioner is far from being trivial, as previously noticed in Section 3.12.1 when dealing with the scaling of a matrix.
2. *Incomplete LU factorization* (shortly ILU) and *Incomplete Cholesky factorization* (shortly IC).
   An incomplete factorization of A is a process that computes $\mathrm{P} = \mathrm{L}_{in}\mathrm{U}_{in}$, where L_{in} is a lower triangular matrix and U_{in} is an upper triangular matrix. These matrices are approximations of the *exact* matrices L, U of the LU factorization of A and are chosen in such a way that the residual matrix $\mathrm{R} = \mathrm{A} - \mathrm{L}_{in}\mathrm{U}_{in}$ satisfies some prescribed requirements, such as having zero entries in specified locations.
   For a given matrix M, the L-part (U-part) of M will mean henceforth the lower (upper) triangular part of M. Moreover, we assume that the factorization process can be carried out without resorting to pivoting.
   The basic approach to incomplete factorization, consists of requiring the approximate factors L_{in} and U_{in} to have the same sparsity pattern as the L-part and U-part of A, respectively. A general algorithm for constructing an incomplete factorization is to perform Gauss elimination as follows: at each step k, compute $m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$ only if $a_{ik} \neq 0$ for $i = k+1, \ldots, n$. Then, compute for $j = k+1, \ldots, n$ $a_{ij}^{(k+1)}$ only if $a_{ij} \neq 0$. This algorithm is implemented in Program 17, where the matrices L_{in} and U_{in} are progressively overwritten onto the L-part and U-part of A.

Program 17 - basicILU : Incomplete LU factorization

```
function [A] = basicILU(A)
%BASICILU Incomplete LU factorization.
```