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*

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

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

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

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 $\lim_{k\to\infty} \mathbf{e}^{(k)} = \mathbf{0}$ 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* $\{\mathbf{x}^{(k)}\}$ *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 $\lim_{k\to\infty} \mathrm{B}^k \mathbf{e}^{(0)} = \mathbf{0}$ 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 0 as $k \to \infty$, since $|\lambda| > 1$. ◇

From (1.23) and Theorem 1.5 it follows that a sufficient condition for convergence to hold is that $\|\mathrm{B}\| < 1$, for any 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 1.13 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 *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

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

This amounts to performing the following splitting for A

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

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, A=D-(E+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

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

This method amounts to performing the following splitting for A

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

and the associated iteration matrix is

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

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

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

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

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

from which the iteration matrix is

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

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* $2\mathrm{D} - \mathrm{A}$ *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 $2\mathrm{D} - \mathrm{A}$ can be removed, yielding the following result.

Theorem 4.4 *If* A *if 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 positive definite and tridiagonal, it can be shown that also the Jacobi method is convergent and

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

In this case, the Gauss-Seidel method is more rapidly convergent than 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 M=D-E-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 $0 < \omega < 2$. ◇

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,*

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

and the corresponding asymptotic convergence factor is

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

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 $\widehat{\mathbf{x}}^{(k)}$ the solution (in finite arithmetic) generated by an iterative method of the form (4.6) after k steps. Due to rounding errors, $\widehat{\mathbf{x}}^{(k)}$ 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 $\widehat{\mathrm{N}}\mathbf{x}^{(k)} + \mathbf{b}$. 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

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

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

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

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

$$\begin{aligned} &\mathrm{B}_s(\omega) = (\mathrm{D}-\omega\mathrm{F})^{-1}(\omega\mathrm{E}+(1-\omega)\mathrm{D})(\mathrm{D}-\omega\mathrm{E})^{-1}(\omega\mathrm{F}+(1-\omega)\mathrm{D}), \\ &\mathbf{b}_\omega = \omega(2-\omega)(\mathrm{D}-\omega\mathrm{F})^{-1}\mathrm{D}(\mathrm{D}-\omega\mathrm{E})^{-1}\mathbf{b}. \end{aligned}$$

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]).

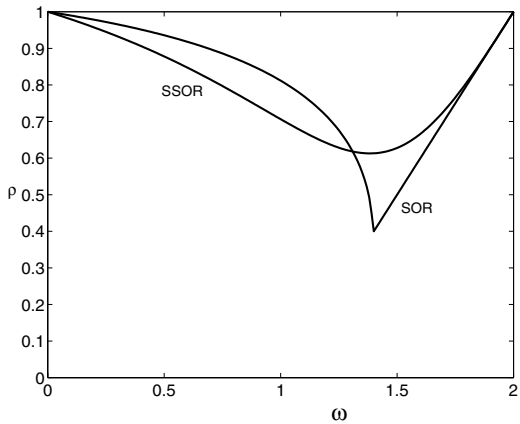

FIGURE 4.1. Spectral radius of the iteration matrix of SOR and SSOR methods, as a function of the relaxation parameter ω for the matrix $\text{tridiag}_{10}(-1, 2, -1)$

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, toll, omega)
[n,n]=size(a);
iter = 0; r = b - a * x0; r0 = norm(r); err = norm (r); x = x0;
while err > toll & 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
   x (i) = omega * ( b(i) - s) / a(i,i) + (1 - omega) * x(i);
 end
 r = b - a * x;          err = norm (r) / r0;
end
```

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, toll, omega)
[n,n]=size(a);
iter = 0; r = b - a * x0; r0 = norm (r); err = norm (r); xold = x0;
while err > toll & 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) = omega * ( b(i) - s) / a(i,i) + (1 - omega) * xold (i);
 end
 x = x'; xold = x;    r = b - a * x;     err = norm (r) / r0;
end
```

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* 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 these methods (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 P=I, the methods will be called *nonpreconditioned*. The Jacobi and Gauss-Seidel methods can be regarded as stationary Richardson methods with $\alpha = 1$, $\mathrm{P} = \mathrm{D}$ and $\mathrm{P} = \mathrm{D} - \mathrm{E}$, respectively.

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*

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

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*

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

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. ◇

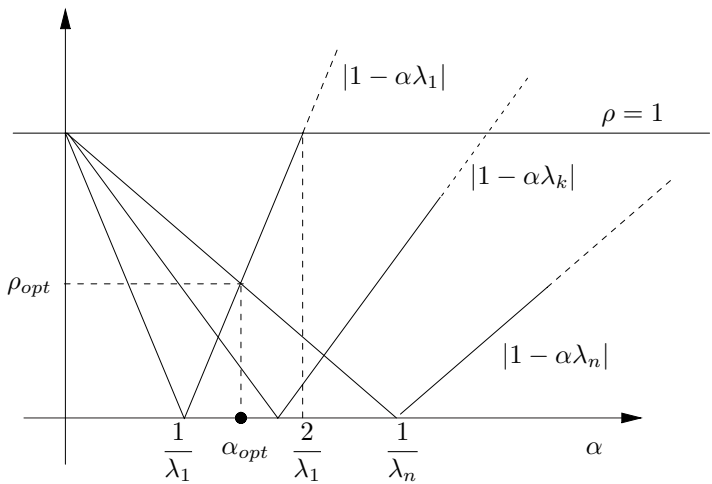

FIGURE 4.2. Spectral radius of R_α as a function of the eigenvalues of $\mathrm{P}^{-1}\mathrm{A}$

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

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

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 4.1 *Let* A *be a symmetric positive definite matrix. Then, the non preconditioned 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 $\mathrm{P}^{-1}\mathrm{A}$. ◇

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 B=P^{-1}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

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

or

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

There are *point preconditioners* or *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 "preconditioning" 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 a lack of general theoretical results that allow to devise optimal preconditioners. However, an established "rule of thumb" is that P is a good preconditioner for A if $P^{-1}A$ 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 procedure, 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 non symmetric 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 $P = L_{in}U_{in}$, 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 $R = A - L_{in}U_{in}$ 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)
[n,n]=size(a);
for k=1:n-1, for i=k+1:n,
   if a(i,k) ~= 0
     a(i,k) = a(i,k) / a(k,k);
     for j=k+1:n
       if a(i,j) ~= 0
         a(i,j) = a(i,j) -a(i,k)*a(k,j);
       end
     end
   end
end, end
```

We notice that having L_{in} and U_{in} with the same patterns as the L and U-parts of A, respectively, does not necessarily imply that R has the same sparsity pattern as A, but guarantees that $r_{ij} = 0$ if $a_{ij} \neq 0$, as is shown in Figure 4.3.

The resulting incomplete factorization is known as ILU(0), where "0" means that no fill-in has been introduced in the factorization process. An alternative strategy might be to fix the structure of L_{in} and U_{in} irrespectively of that of A, in such a way that some computational criteria are satisfied (for example, that the incomplete factors have the simplest possible structure).

The accuracy of the ILU(0) factorization can obviously be improved by allowing some fill-in to arise, and thus, by accepting nonzero entries in the factorization whereas A has elements equal to zero. To this purpose, it is convenient to introduce a function, which we call *fill-in level*, that is associated with each entry of A and that is being modified during the factorization process. If the fill-in level of an

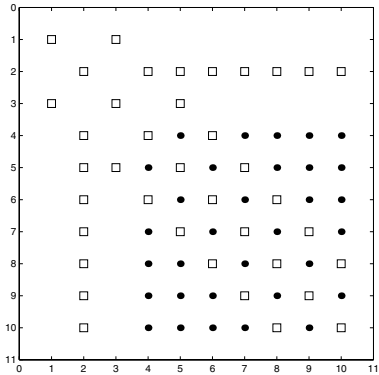

FIGURE 4.3. The sparsity pattern of the original matrix A is represented by the squares, while the pattern of $\mathrm{R} = \mathrm{A} - \mathrm{L}_{in}\mathrm{U}_{in}$, computed by Program 17, is drawn by the bullets

element is greater than an admissible value $p \in \mathbb{N}$, the corresponding entry in U_{in} or L_{in} is set equal to zero.

Let us explain how this procedure works, assuming that the matrices L_{in} and U_{in} are progressively overwritten to A (as happens in Program 4). The fill-in level of an entry $a_{ij}^{(k)}$ is denoted by lev_{ij}, where the dependence on k is understood, and it should provide a reasonable estimate of the size of the entry during the factorization process. Actually, we are assuming that if $lev_{ij} = q$ then $|a_{ij}| \simeq \delta^q$ with $\delta \in (0,1)$, so that q is greater when $|a_{ij}^{(k)}|$ is smaller.

At the starting step of the procedure, the level of the nonzero entries of A and of the diagonal entries is set equal to 0, while the level of the null entries is set equal to infinity. For any row $i = 2, \ldots, n$, the following operations are performed: if $lev_{ik} \leq p$, $k = 1, \ldots, i-1$, the entry m_{ik} of L_{in} and the entries $a_{ij}^{(k+1)}$ of U_{in}, $j = i+1, \ldots, n$, are updated. Moreover, if $a_{ij}^{(k+1)} \neq 0$ the value lev_{ij} is updated as being the minimum between the available value of lev_{ij} and $lev_{ik} + lev_{kj} + 1$. The reason of this choice is that $|a_{ij}^{(k+1)}| = |a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}| \simeq |\delta^{lev_{ij}} - \delta^{lev_{ik}+lev_{kj}+1}|$, so that one can assume that the size of $|a_{ij}^{(k+1)}|$ is the maximum between $\delta^{lev_{ij}}$ and $\delta^{lev_{ik}+lev_{kj}+1}$.

The above factorization process is called ILU(p) and turns out to be extremely efficient (with p small) provided that it is coupled with a suitable matrix reordering (see Section 3.9).

Program 18 implements the ILU(p) factorization; it returns in output the approximate matrices L_{in} and U_{in} (overwritten to the input matrix `a`), with the diagonal entries of L_{in} equal to 1, and the ma-

trix `lev` containing the fill-in level of each entry at the end of the factorization.

Program 18 - ilup : ILU(p) factorization

```
function [a,lev] = ilup (a,p)
[n,n]=size(a);
for i=1:n,  for j=1:n
    if (a(i,j) ~= 0) | (i==j)
      lev(i,j)=0;
    else
      lev(i,j)=Inf;
    end
end,  end
for i=2:n,
  for k=1:i-1
    if lev(i,k) <= p
      a(i,k)=a(i,k)/a(k,k);
      for j=k+1:n
        a(i,j)=a(i,j)-a(i,k)*a(k,j);
        if a(i,j) ~= 0
          lev(i,j)=min(lev(i,j),lev(i,k)+lev(k,j)+1);
        end
      end
    end
  end
  for j=1:n, if lev(i,j) > p, a(i,j) = 0; end, end
end
```

Example 4.4 Consider the matrix $\mathrm{A} \in \mathbb{R}^{46\times 46}$ associated with the finite difference approximation of the Laplace operator $\Delta \cdot = \frac{\partial^2 \cdot}{\partial x^2} + \frac{\partial^2 \cdot}{\partial y^2}$ (see Section 12.6). This matrix can be generated with the following MATLAB commands: `G=numgrid('B',10); A=delsq(G)` and corresponds to the discretization of the differential operator on a domain having the shape of the exterior of a butterfly and included in the square $[-1,1]^2$ (see Section 12.6). The number of nonzero entries of A is 174. Figure 4.4 shows the pattern of matrix A (drawn by the bullets) and the entries in the pattern added by the ILU(1) and ILU(2) factorizations due to fill-in (denoted by the squares and the triangles, respectively). Notice that these entries are all contained within the envelope of A since no pivoting has been performed. •

The ILU(p) process can be carried out without knowing the actual values of the entries of A, but only working on their fill-in levels. Therefore, we can distinguish between a *symbolic factorization* (the generation of the levels) and an *actual factorization* (the computation of the entries of ILU(p) starting from the informations contained in

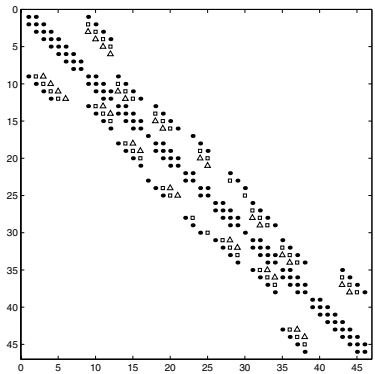

FIGURE 4.4. Pattern of the matrix A in Example 4.4 (bullets); entries added by the ILU(1) and ILU(2) factorizations (squares and triangles, respectively)

the level function). The scheme is thus particularly effective when several linear systems must be solved, with matrices having the same structure but different entries.

On the other hand, for certain classes of matrices, the fill-in level does not always provide a sound indication of the *actual* size attained by the entries. In such cases, it is better to monitor the size of the entries of R by neglecting each time the entries that are too small. For instance, one can drop out the entries $a_{ij}^{(k+1)}$ such that

$$|a_{ij}^{(k+1)}| \leq c|a_{ii}^{(k+1)}a_{jj}^{(k+1)}|^{1/2}, \qquad i,j = 1,\dots,n,$$

with $0 < c < 1$ (see [Axe94]).

In the strategies considered so far, the entries of the matrix that are dropped out can no longer be recovered in the incomplete factorization process. Some remedies exist for this drawback: for instance, at the end of each k-th step of the factorization, one can sum, row by row, the discarded entries to the diagonal entries of U_{in}. By doing so, an incomplete factorization known as MILU (Modified ILU) is obtained, which enjoys the property of being exact with respect to the constant vectors, i.e., such that $\mathrm{R}\mathbf{1}^T = \mathbf{0}^T$ (see [Axe94] for other formulations). In the practice, this simple trick provides, for a wide class of matrices, a better preconditioner than obtained with the ILU method. In the case of symmetric positive definite matrices one can resort to the Modified Incomplete Cholesky Factorization (MICh).

We conclude by mentioning the ILUT factorization, which collects the features of ILU(p) and MILU. This factorization can also include partial pivoting by columns with a slight increase of the computational

cost. For an efficient implementation of incomplete factorizations, we refer to the MATLAB function `luinc` in the toolbox `sparfun`.

The existence of the ILU factorization is not guaranteed for all nonsingular matrices (see for an example [Elm86]) and the process stops if zero pivotal entries arise. Existence theorems can be proved if A is an M-matrix [MdV77] or diagonally dominant [Man80]. It is worth noting that sometimes the ILU factorization turns out to be more stable than the complete LU factorization [GM83].

3. *Polynomial preconditioners*: the preconditioning matrix is defined as

$$\mathrm{P}^{-1} = p(\mathrm{A}),$$

where p is a polynomial in A, usually of low degree.

A remarkable example is given by Neumann polynomial preconditioners. Letting $\mathrm{A} = \mathrm{D} - \mathrm{C}$, we have $\mathrm{A} = (\mathrm{I} - \mathrm{CD}^{-1})\mathrm{D}$, from which

$$\mathrm{A}^{-1} = \mathrm{D}^{-1}(\mathrm{I} - \mathrm{CD}^{-1})^{-1} = \mathrm{D}^{-1}(\mathrm{I} + \mathrm{CD}^{-1} + (\mathrm{CD}^{-1})^2 + \dots).$$

A preconditioner can then be obtained by truncating the series above at a certain power p. This method is actually effective only if $\rho(\mathrm{CD}^{-1}) < 1$, which is the necessary condition in order the series to be convergent.

4. *Least-squares preconditioners*: A^{-1} is approximated by a least-squares polynomial $p_s(\mathrm{A})$ (see Section 3.13). Since the aim is to make matrix $\mathrm{I} - \mathrm{P}^{-1}\mathrm{A}$ as close as possible to the null matrix, the least-squares approximant $p_s(\mathrm{A})$ is chosen in such a way that the function $\varphi(x) = 1 - p_s(x)x$ is minimized. This preconditioning technique works effectively only if A is symmetric and positive definite.

For further results on preconditioners, see [dV89] and [Axe94].

Example 4.5 Consider the matrix $\mathrm{A} \in \mathbb{R}^{324\times 324}$ associated with the finite difference approximation of the Laplace operator on the square $[-1, 1]^2$. This matrix can be generated with the following MATLAB commands: `G=numgrid('N',20); A=delsq(G)`. The condition number of the matrix is $K_2(\mathrm{A}) = 211.3$. In Table 4.1 we show the values of $K_2(\mathrm{P}^{-1}\mathrm{A})$ computed using the ILU(p) and Neumann preconditioners, with $p = 0, 1, 2, 3$. In the last case D is the diagonal part of A. •

Remark 4.2 Let A and P be real symmetric matrices of order n, with P positive definite. The eigenvalues of the preconditioned matrix $\mathrm{P}^{-1}\mathrm{A}$ are solutions of the algebraic equation

$$\mathrm{A}\mathbf{x} = \lambda \mathrm{P}\mathbf{x}, \tag{4.32}$$

p	ILU(p)	Neumann
0	22.3	211.3
1	12	36.91
2	8.6	48.55
3	5.6	18.7

TABLE 4.1. Spectral condition numbers of the preconditioned matrix A of Example 4.5 as a function of p

where $\mathbf{x}$ is an eigenvector associated with the eigenvalue λ. Equation (4.32) is an example of *generalized eigenvalue problem* (see Section 5.9 for a thorough discussion) and the eigenvalue λ can be computed through the following generalized Rayleigh quotient

$$\lambda = \frac{(\mathrm{A}\mathbf{x}, \mathbf{x})}{(\mathrm{P}\mathbf{x}, \mathbf{x})}.$$

Applying the Courant-Fisher Theorem (see Section 5.11) yields

$$\frac{\lambda_{min}(\mathrm{A})}{\lambda_{max}(\mathrm{P})} \leq \lambda \leq \frac{\lambda_{max}(\mathrm{A})}{\lambda_{min}(\mathrm{P})}. \tag{4.33}$$

Relation (4.33) provides a lower and upper bound for the eigenvalues of the preconditioned matrix as a function of the extremal eigenvalues of A and P, and therefore it can be profitably used to estimate the condition number of $\mathrm{P}^{-1}\mathrm{A}$. ■

4.3.3 The Gradient Method

The expression of the optimal parameter that has been provided in Theorem 4.9 is of limited usefulness in practical computations, since it requires the knowledge of the extremal eigenvalues of the matrix $\mathrm{P}^{-1}\mathrm{A}$. In the special case of symmetric and positive definite matrices, however, the optimal acceleration parameter can be *dynamically* computed at each step k as follows.

We first notice that, for such matrices, solving system (3.2) is equivalent to finding the minimizer $\mathbf{x} \in \mathbb{R}^n$ of the quadratic form

$$\Phi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T\mathrm{A}\mathbf{y} - \mathbf{y}^T\mathbf{b},$$

which is called the *energy of system* (3.2). Indeed, the gradient of Φ is given by

$$\nabla\Phi(\mathbf{y}) = \frac{1}{2}(\mathrm{A}^T + \mathrm{A})\mathbf{y} - \mathbf{b} = \mathrm{A}\mathbf{y} - \mathbf{b}. \tag{4.34}$$

As a consequence, if $\nabla\Phi(\mathbf{x}) = \mathbf{0}$ then $\mathbf{x}$ is a solution of the original system. Conversely, if $\mathbf{x}$ is a solution, then

$$\Phi(\mathbf{y}) = \Phi(\mathbf{x} + (\mathbf{y} - \mathbf{x})) = \Phi(\mathbf{x}) + \frac{1}{2}(\mathbf{y} - \mathbf{x})^T\mathrm{A}(\mathbf{y} - \mathbf{x}), \qquad \forall \mathbf{y} \in \mathbb{R}^n$$

and thus, $\Phi(\mathbf{y}) > \Phi(\mathbf{x})$ if $\mathbf{y} \neq \mathbf{x}$, i.e. $\mathbf{x}$ is a minimizer of the functional Φ.

Notice that the previous relation is equivalent to

$$\frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_{\mathrm{A}}^2 = \Phi(\mathbf{y}) - \Phi(\mathbf{x}) \tag{4.35}$$

where $\|\cdot\|_{\mathrm{A}}$ is the A-*norm* or *energy norm*, defined in (1.28).

The problem is thus to determine the minimizer $\mathbf{x}$ of Φ starting from a point $\mathbf{x}^{(0)} \in \mathbb{R}^n$ and, consequently, to select suitable directions along which moving to get as close as possible to the solution $\mathbf{x}$. The optimal direction, that joins the starting point $\mathbf{x}^{(0)}$ to the solution point $\mathbf{x}$, is obviously unknown *a priori*. Therefore, we must take a step from $\mathbf{x}^{(0)}$ along another direction $\mathbf{d}^{(0)}$, and then fix along this latter a new point $\mathbf{x}^{(1)}$ from which to iterate the process until convergence.

Thus, at the generic step k, $\mathbf{x}^{(k+1)}$ is computed as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}, \tag{4.36}$$

where α_k is the value which fixes the length of the step along $\mathbf{d}^{(k)}$. The most natural idea is to take the descent direction of maximum slope $\nabla\Phi(\mathbf{x}^{(k)})$, which yields the *gradient method* or *steepest descent method*.

On the other hand, due to (4.34), $\nabla\Phi(\mathbf{x}^{(k)}) = \mathrm{A}\mathbf{x}^{(k)} - \mathbf{b} = -\mathbf{r}^{(k)}$, so that the direction of the gradient of Φ coincides with that of residual and can be immediately computed using the current iterate. This shows that the gradient method, as well as the Richardson method, moves at each step k along the direction $\mathbf{d}^{(k)} = \mathbf{r}^{(k)}$.

To compute the parameter α_k let us write explicitly $\Phi(\mathbf{x}^{(k+1)})$ as a function of a parameter α

$$\Phi(\mathbf{x}^{(k+1)}) = \frac{1}{2}(\mathbf{x}^{(k)} + \alpha\mathbf{r}^{(k)})^T\mathrm{A}(\mathbf{x}^{(k)} + \alpha\mathbf{r}^{(k)}) - (\mathbf{x}^{(k)} + \alpha\mathbf{r}^{(k)})^T\mathbf{b}.$$

Differentiating with respect to α and setting it equal to zero, yields the desired value of α_k

$$\alpha_k = \frac{{\mathbf{r}^{(k)}}^T\mathbf{r}^{(k)}}{{\mathbf{r}^{(k)}}^T\mathrm{A}\mathbf{r}^{(k)}} \tag{4.37}$$

which depends only on the residual at the k-th step. For this reason, the nonstationary Richardson method employing (4.37) to evaluate the acceleration parameter, is also called the *gradient method with dynamic parameter* (shortly, *gradient method*), to distinguish it from the stationary Richardson method (4.23) or *gradient method with constant parameter*, where $\alpha_k = \alpha$ is a constant for any $k \geq 0$.

Summarizing, the gradient method can be described as follows:

given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, for $k = 0, 1, \ldots$ until convergence, compute

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

$$\alpha_k = \frac{{\mathbf{r}^{(k)}}^T \mathbf{r}^{(k)}}{{\mathbf{r}^{(k)}}^T \mathrm{A}\mathbf{r}^{(k)}}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)}.$$

Theorem 4.10 *Let* A *be a symmetric and positive definite matrix; then the gradient method is convergent for any choice of the initial datum* $\mathbf{x}^{(0)}$ *and*

$$\|\mathbf{e}^{(k+1)}\|_{\mathrm{A}} \leq \frac{K_2(\mathrm{A}) - 1}{K_2(\mathrm{A}) + 1} \|\mathbf{e}^{(k)}\|_{\mathrm{A}}, \qquad k = 0, 1, \ldots, \tag{4.38}$$

where $\|\cdot\|_{\mathrm{A}}$ *is the energy norm defined in* (1.28).

Proof. Let $\mathbf{x}^{(k)}$ be the solution generated by the gradient method at the k-th step. Then, let $\mathbf{x}_R^{(k+1)}$ be the vector generated by taking one step of the non preconditioned Richardson method with optimal parameter starting from $\mathbf{x}^{(k)}$, i.e., $\mathbf{x}_R^{(k+1)} = \mathbf{x}^{(k)} + \alpha_{opt}\mathbf{r}^{(k)}$.

Due to Corollary 4.1 and (4.28), we have

$$\|\mathbf{e}_R^{(k+1)}\|_{\mathrm{A}} \leq \frac{K_2(\mathrm{A}) - 1}{K_2(\mathrm{A}) + 1} \|\mathbf{e}^{(k)}\|_{\mathrm{A}},$$

where $\mathbf{e}_R^{(k+1)} = \mathbf{x}_R^{(k+1)} - \mathbf{x}$. Moreover, from (4.35) we have that the vector $\mathbf{x}^{(k+1)}$, generated by the gradient method, is the one that minimizes the A-norm of the error among all vectors of the form $\mathbf{x}^{(k)} + \theta\mathbf{r}^{(k)}$, with $\theta \in \mathbb{R}$. Therefore, $\|\mathbf{e}^{(k+1)}\|_{\mathrm{A}} \leq \|\mathbf{e}_R^{(k+1)}\|_{\mathrm{A}}$ which is the desired result. ◇

We notice that the line through $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$ is tangent at the point $\mathbf{x}^{(k+1)}$ to the ellipsoidal level surface $\left\{\mathbf{x} \in \mathbb{R}^n : \Phi(\mathbf{x}) = \Phi(\mathbf{x}^{(k+1)})\right\}$ (see also Figure 4.5).

Relation (4.38) shows that convergence of the gradient method can be quite slow if $K_2(\mathrm{A}) = \lambda_1/\lambda_n$ is large. A simple geometric interpretation of this result can be given in the case $n = 2$. Suppose that A=diag(λ_1, λ_2), with $0 < \lambda_2 \leq \lambda_1$ and $\mathbf{b} = (b_1, b_2)^T$.

In such a case, the curves corresponding to $\Phi(x_1, x_2) = c$, as c varies in $\mathbb{R}^+$, form a sequence of concentric ellipses whose semi-axes have length inversely proportional to the values λ_1 and λ_2. If $\lambda_1 = \lambda_2$, the ellipses degenerate into circles and the direction of the gradient crosses the center directly, in such a way that the gradient method converges in one iteration. Conversely, if $\lambda_1 \gg \lambda_2$, the ellipses become strongly eccentric and the method converges quite slowly, as shown in Figure 4.5, moving along a "zig-zag" trajectory.

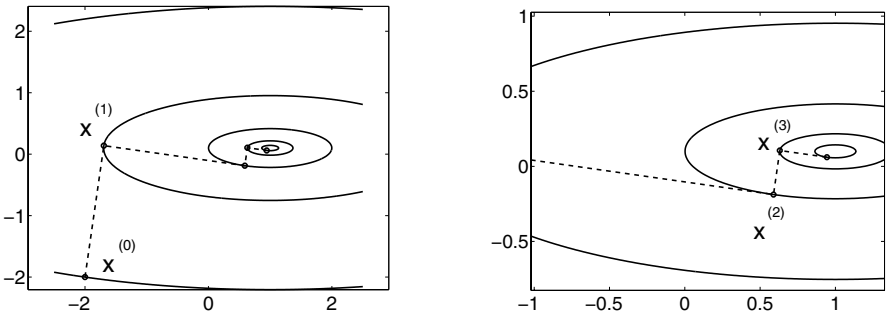

FIGURE 4.5. The first iterates of the gradient method on the level curves of Φ

Program 19 provides an implementation of the gradient method with dynamic parameter. Here and in the programs reported in the remainder of the section, the input parameters `A`, `x`, `b`, `M`, `maxit` and `tol` respectively represent the coefficient matrix of the linear system, the initial datum $\mathbf{x}^{(0)}$, the right side, a possible preconditioner, the maximum number of admissible iterations and a tolerance for the stopping test. This stopping test checks if the ratio $\|\mathbf{r}^{(k)}\|_2/\|\mathbf{b}\|_2$ is less than `tol`. The output parameters of the code are the the number of iterations `niter` required to fulfill the stopping test, the vector `x` with the solution computed after `niter` iterations and the normalized residual `error` $= \|\mathbf{r}^{(\texttt{niter})}\|_2/\|\mathbf{b}\|_2$. A null value of the parameter `flag` warns the user that the algorithm has actually satisfied the stopping test and it has not terminated due to reaching the maximum admissible number of iterations.

Program 19 - gradient : Gradient method with dynamic parameter

```
function [x, error, niter, flag] = gradient(A, x, b, M, maxit, tol)
flag = 0;    niter = 0;   bnrm2 = norm( b );
if  ( bnrm2 == 0.0 ), bnrm2 = 1.0; end
r = b - A*x;  error = norm( r ) / bnrm2;
if ( error < tol ) return, end
for niter = 1:maxit
    z  = M \ r;  rho = (r'*z);
    q = A*z;                alpha = rho / (z'*q );
    x = x + alpha * z;      r = r - alpha*q;
    error = norm( r ) / bnrm2;
    if ( error <= tol ), break, end
end
if ( error > tol ) flag = 1; end
```

Example 4.6 Let us solve with the gradient method the linear system with matrix $\mathrm{A}_m \in \mathbb{R}^{m\times m}$ generated with the MATLAB commands `G=numgrid('S',n); A=delsq(G)` where $m = (n-2)^2$. This matrix is associated with the discretization of the differential Laplace operator on the domain $[-1,1]^2$. The right-hand side $\mathbf{b}_m$ is selected in such a way that the exact solution is the vector $\mathbf{1}^T \in \mathbb{R}^m$. The matrix A_m is symmetric and positive definite for any m and becomes ill-conditioned for large values of m. We run Program 19 in the cases $m = 16$ and $m = 400$, with $\mathbf{x}^{(0)} = \mathbf{0}^T$, `tol`$=10^{-10}$ and `maxit`$=200$. If $m = 400$, the method fails to satisfy the stopping test within the admissible maximum number of iterations and exhibits an extremely slow reduction of the residual (see Figure 4.6). Actually, $K_2(\mathrm{A}_{400}) \simeq 258$. If, however, we precondition the system with the matrix $\mathrm{P} = \mathrm{R}_{in}^T\mathrm{R}_{in}$, where R_{in} is the lower triangular matrix in the Cholesky incomplete factorization of A, the algorithm fulfills the convergence within the maximum admissible number of iterations (indeed, now $K_2(\mathrm{P}^{-1}\mathrm{A}_{400}) \simeq 38$). •

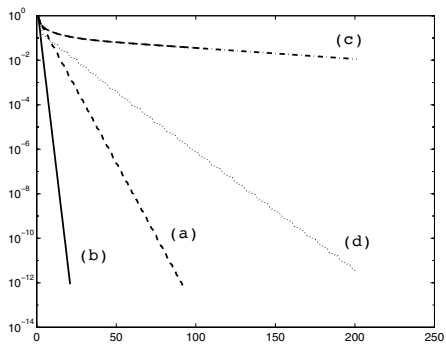

FIGURE 4.6. The residual normalized to the starting one, as a function of the number of iterations, for the gradient method applied to the systems in Example 4.6. The curves labelled (a) and (b) refer to the case $m = 16$ with the non preconditioned and preconditioned method, respectively, while the curves labelled (c) and (d) refer to the case $m = 400$ with the non preconditioned and preconditioned method, respectively

4.3.4 The Conjugate Gradient Method

The gradient method consists essentially of two phases: choosing a descent direction (the one of the residual) and picking up a point of local minimum for Φ along that direction. The second phase is independent of the first one since, for a given direction $\mathbf{p}^{(k)}$, we can determine α_k as being the value of the parameter α such that $\Phi(\mathbf{x}^{(k)} + \alpha\mathbf{p}^{(k)})$ is minimized. Differentiating with respect to α and setting to zero the derivative at the minimizer, yields

$$\alpha_k = \frac{{\mathbf{p}^{(k)}}^T\mathbf{r}^{(k)}}{{\mathbf{p}^{(k)}}^T\mathrm{A}\mathbf{p}^{(k)}}, \tag{4.39}$$

instead of (4.37). The question is how to determine $\mathbf{p}^{(k)}$. A different approach than the one which led to identify $\mathbf{p}^{(k)}$ with $\mathbf{r}^{(k)}$ is suggested by the following definition.

Definition 4.4 A direction $\mathbf{x}^{(k)}$ is said to be *optimal* with respect to a direction $\mathbf{p} \neq \mathbf{0}$ if

$$\Phi(\mathbf{x}^{(k)}) \leq \Phi(\mathbf{x}^{(k)} + \lambda \mathbf{p}), \qquad \forall \lambda \in \mathbb{R}. \tag{4.40}$$

If $\mathbf{x}^{(k)}$ is optimal with respect to any direction in a vector space V, we say that $\mathbf{x}^{(k)}$ is optimal with respect to V. ■

From the definition of optimality, it turns out that $\mathbf{p}$ must be orthogonal to the residual $\mathbf{r}^{(k)}$. Indeed, from (4.40) we conclude that Φ admits a local minimum along $\mathbf{p}$ for $\lambda = 0$, and thus the partial derivative of Φ with respect to λ must vanish at $\lambda = 0$. Since

$$\frac{\partial \Phi}{\partial \lambda}(\mathbf{x}^{(k)} + \lambda \mathbf{p}) = \mathbf{p}^T(\mathrm{A}\mathbf{x}^{(k)} - \mathbf{b}) + \lambda \mathbf{p}^T \mathrm{A}\mathbf{p},$$

we therefore have

$$\frac{\partial \Phi}{\partial \lambda}(\mathbf{x}^{(k)})_{|\lambda=0} = 0 \quad \text{iff} \quad \mathbf{p}^T(\mathbf{r}^{(k)}) = 0,$$

that is, $\mathbf{p} \perp \mathbf{r}^{(k)}$. Notice that the iterate $\mathbf{x}^{(k+1)}$ of the gradient method is optimal with respect to $\mathbf{r}^{(k)}$ since, due to the choice of α_k, we have $\mathbf{r}^{(k+1)} \perp \mathbf{r}^{(k)}$, but this property no longer holds for the successive iterate $\mathbf{x}^{(k+2)}$ (see Exercise 12). It is then natural to ask whether there exist descent directions that maintain the optimality of iterates. Let

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

and assume that $\mathbf{x}^{(k)}$ is optimal with respect to a direction $\mathbf{p}$ (thus, $\mathbf{r}^{(k)} \perp \mathbf{p}$). Let us impose that $\mathbf{x}^{(k+1)}$ is still optimal with respect to $\mathbf{p}$, that is, $\mathbf{r}^{(k+1)} \perp \mathbf{p}$. We obtain

$$0 = \mathbf{p}^T \mathbf{r}^{(k+1)} = \mathbf{p}^T(\mathbf{r}^{(k)} - \mathrm{A}\mathbf{q}) = -\mathbf{p}^T \mathrm{A}\mathbf{q}.$$

The conclusion is that, in order to preserve optimality between successive iterates, the descent directions must be mutually *A-orthogonal* or *A-conjugate*, i.e.

$$\mathbf{p}^T \mathrm{A}\mathbf{q} = 0.$$

A method employing A-conjugate descent directions is called *conjugate*. The next step is how to generate automatically a sequence of conjugate

directions. This can be done as follows. Let $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$ and search for the directions of the form

$$\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{p}^{(k)}, \quad k = 0, 1, \ldots \tag{4.41}$$

where $\beta_k \in \mathbb{R}$ must be determined in such a way that

$$(\mathrm{A}\mathbf{p}^{(j)})^T \mathbf{p}^{(k+1)} = 0, \quad j = 0, 1, \ldots, k. \tag{4.42}$$

Requiring that (4.42) is satisfied for $j = k$, we get from (4.41)

$$\beta_k = \frac{(\mathrm{A}\mathbf{p}^{(k)})^T \mathbf{r}^{(k+1)}}{(\mathrm{A}\mathbf{p}^{(k)})^T \mathbf{p}^{(k)}}, \quad k = 0, 1, \ldots$$

We must now verify that (4.42) holds also for $j = 0, 1, \ldots, k-1$. To do this, let us proceed by induction on k. Due to the choice of β_0, relation (4.42) holds for $k = 0$; let us thus assume that the directions $\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(k-1)}$ are mutually A-orthogonal and, without losing generality, that

$$(\mathbf{p}^{(j)})^T \mathbf{r}^{(k)} = 0, \quad j = 0, 1, \ldots, k-1, \quad k \geq 1. \tag{4.43}$$

Then, from (4.41) it follows that

$$(\mathrm{A}\mathbf{p}^{(j)})^T \mathbf{p}^{(k+1)} = (\mathrm{A}\mathbf{p}^{(j)})^T \mathbf{r}^{(k+1)}, \quad j = 0, 1, \ldots, k-1.$$

Moreover, due to (4.43) and by the assumption of of A-orthogonality we get

$$(\mathbf{p}^{(j)})^T \mathbf{r}^{(k+1)} = (\mathbf{p}^{(j)})^T \mathbf{r}^{(k)} - \alpha_k (\mathbf{p}^{(j)})^T \mathrm{A}\mathbf{p}^{(k)} = 0, \quad j = 0, \ldots, k-1 \tag{4.44}$$

i.e., we conclude that $\mathbf{r}^{(k+1)}$ is orthogonal to every vector of the space $V_k = \mathrm{span}(\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(k-1)})$. Since $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$, from (4.41) it follows that V_k is also equal to $\mathrm{span}(\mathbf{r}^{(0)}, \ldots, \mathbf{r}^{(k-1)})$. Then, (4.41) implies that $\mathrm{A}\mathbf{p}^{(j)} \in V_{j+1}$ and thus, due to (4.44)

$$(\mathrm{A}\mathbf{p}^{(j)})^T \mathbf{r}^{(k+1)} = 0, \quad j = 0, 1, \ldots, k-1.$$

As a consequence, (4.42) holds for $j = 0, \ldots, k$.

The conjugate gradient method (CG) is the method obtained by choosing the descent directions $\mathbf{p}^{(k)}$ given by (4.41) and the acceleration parameter α_k as in (4.39). As a consequence, setting $\mathbf{r}^{(0)} = \mathbf{b} - \mathrm{A}\mathbf{x}^{(0)}$ and $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$, the k-th iteration of the conjugate gradient method takes the following

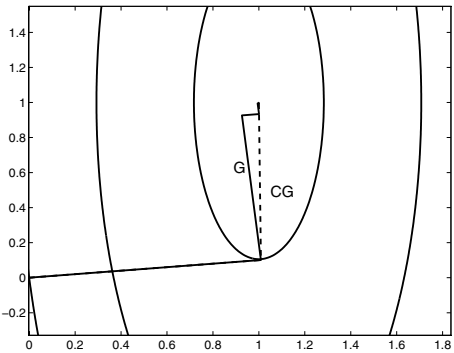

FIGURE 4.7. Descent directions for the conjugate gradient method (denoted by CG, dashed line) and the gradient method (denoted by G, solid line). Notice that the CG method reaches the solution after two iterations

form

$$\alpha_k = \frac{\mathbf{p}^{(k)^T}\mathbf{r}^{(k)}}{\mathbf{p}^{(k)^T}\mathrm{A}\mathbf{p}^{(k)}}$$

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

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathrm{A}\mathbf{p}^{(k)}$$

$$\beta_k = \frac{(\mathrm{A}\mathbf{p}^{(k)})^T\mathbf{r}^{(k+1)}}{(\mathrm{A}\mathbf{p}^{(k)})^T\mathbf{p}^{(k)}}$$

$$\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{p}^{(k)}.$$

It can also be shown (see Exercise 13) that the two parameters α_k and β_k may be alternatively expressed as

$$\alpha_k = \frac{\|\mathbf{r}^{(k)}\|_2^2}{\mathbf{p}^{(k)^T}\mathrm{A}\mathbf{p}^{(k)}}, \quad \beta_k = \frac{\|\mathbf{r}^{(k+1)}\|_2^2}{\|\mathbf{r}^{(k)}\|_2^2}. \tag{4.45}$$

We finally notice that, eliminating the descent directions from $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathrm{A}\mathbf{p}^{(k)}$, the following recursive three-terms relation is obtained for the residuals (see Exercise 14)

$$\mathrm{A}\mathbf{r}^{(k)} = -\frac{1}{\alpha_k}\mathbf{r}^{(k+1)} + \left(\frac{1}{\alpha_k} - \frac{\beta_{k-1}}{\alpha_{k-1}}\right)\mathbf{r}^{(k)} + \frac{\beta_k}{\alpha_{k-1}}\mathbf{r}^{(k-1)}. \tag{4.46}$$

As for the convergence of the CG method, we have the following results.

Theorem 4.11 *Let* A *be a symmetric and positive definite matrix. Any method which employs conjugate directions to solve* (3.2) *terminates after at most n steps, yielding the exact solution.*

Proof. The directions $\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(n-1)}$ form an A-orthogonal basis in $\mathbb{R}^n$. Moreover, since $\mathbf{x}^{(k)}$ is optimal with respect to all the directions $\mathbf{p}^{(j)}$, $j = 0, \ldots, k-1$, it follows that $\mathbf{r}^{(k)}$ is orthogonal to the space $S_{k-1} = \text{span}(\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(k-1)})$. As a consequence, $\mathbf{r}^{(n)} \perp S_{n-1} = \mathbb{R}^n$ and thus $\mathbf{r}^{(n)} = \mathbf{0}$ which implies $\mathbf{x}^{(n)} = \mathbf{x}$. ◇

Theorem 4.12 *Let* A *be a symmetric and positive definite matrix and let* λ_1, λ_n *be its maximum and minimum eigenvalues, respectively. The conjugate gradient method for solving* (3.2) *converges after at most* n *steps. Moreover, the error* $\mathbf{e}^{(k)}$ *at the* k*-th iteration (with* $k < n$*) is orthogonal to* $\mathbf{p}^{(j)}$*, for* $j = 0, \ldots, k-1$ *and*

$$\|\mathbf{e}^{(k)}\|_{\mathrm{A}} \leq \frac{2c^k}{1+c^{2k}} \|\mathbf{e}^{(0)}\|_{\mathrm{A}}, \quad \text{with } c = \frac{\sqrt{K_2(\mathrm{A})}-1}{\sqrt{K_2(\mathrm{A})}+1}. \tag{4.47}$$

Proof. The convergence of the CG method in n steps is a consequence of Theorem 4.11.

Let us prove the error estimate, assuming for simplicity that $\mathbf{x}^{(0)} = \mathbf{0}$. Notice first that, for fixed k

$$\mathbf{x}^{(k+1)} = \sum_{j=0}^{k} \gamma_j \mathrm{A}^j \mathbf{b},$$

for suitable $\gamma_j \in \mathbb{R}$. Moreover, by construction, $\mathbf{x}^{(k+1)}$ is the vector which minimizes the A-norm of the error at step $k+1$, among all vectors of the form $\mathbf{z} = \sum_{j=0}^{k} \delta_j \mathrm{A}^j \mathbf{b} = p_k(\mathrm{A})\mathbf{b}$, where $p_k(\xi) = \sum_{j=0}^{k} \delta_j \xi^j$ is a polynomial of degree k and $p_k(\mathrm{A})$ denotes the corresponding matrix polynomial. As a consequence

$$\|\mathbf{e}^{(k+1)}\|_{\mathrm{A}}^2 \leq (\mathbf{x}-\mathbf{z})^T \mathrm{A}(\mathbf{x}-\mathbf{z}) = \mathbf{x}^T q_{k+1}(\mathrm{A}) \mathrm{A} q_{k+1}(\mathrm{A}) \mathbf{x}, \tag{4.48}$$

where $q_{k+1}(\xi) = 1 - p_k(\xi)\xi \in \mathbb{P}_{k+1}^{0,1}$, being $\mathbb{P}_{k+1}^{0,1} = \{q \in \mathbb{P}_{k+1} : q(0) = 1\}$ and $q_{k+1}(\mathrm{A})$ the associated matrix polynomial. From (4.48) we get

$$\|\mathbf{e}^{(k+1)}\|_{\mathrm{A}}^2 = \min_{q_{k+1} \in \mathbb{P}_{k+1}^{0,1}} \mathbf{x}^T q_{k+1}(\mathrm{A}) \mathrm{A} q_{k+1}(\mathrm{A}) \mathbf{x}. \tag{4.49}$$

Since A is symmetric positive definite, there exists an orthogonal matrix Q such that $\mathrm{A} = \mathrm{Q}\Lambda\mathrm{Q}^T$ with $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$. Noticing that $q_{k+1}(\mathrm{A}) = \mathrm{Q} q_{k+1}(\Lambda) \mathrm{Q}^T$, we get from (4.49)

$$\begin{aligned}
\|\mathbf{e}^{(k+1)}\|_{\mathrm{A}}^2 &= \min_{q_{k+1} \in \mathbb{P}_{k+1}^{0,1}} \mathbf{x}^T \mathrm{Q} q_{k+1}(\Lambda) \mathrm{Q}^T \mathrm{Q} \Lambda \mathrm{Q}^T \mathrm{Q} q_{k+1}(\Lambda) \mathrm{Q}^T \mathbf{x} \\
&= \min_{q_{k+1} \in \mathbb{P}_{k+1}^{0,1}} \mathbf{x}^T \mathrm{Q} q_{k+1}(\Lambda) \Lambda q_{k+1}(\Lambda) \mathrm{Q}^T \mathbf{x} \\
&= \min_{q_{k+1} \in \mathbb{P}_{k+1}^{0,1}} \mathbf{y}^T \text{diag}(q_{k+1}(\lambda_i)\lambda_i q_{k+1}(\lambda_i)) \mathbf{y} \\
&= \min_{q_{k+1} \in \mathbb{P}_{k+1}^{0,1}} \sum_{i=1}^{n} y_i^2 \lambda_i (q_{k+1}(\lambda_i))^2
\end{aligned}$$

having set $\mathbf{y} = \mathrm{Q}\mathbf{x}$. Thus, we can conclude that

$$\|\mathbf{e}^{(k+1)}\|_{\mathrm{A}}^2 \leq \left[\min_{q_{k+1} \in \mathbb{P}_{k+1}^{0,1}} \max_{\lambda_i \in \sigma(\mathrm{A})} (q_{k+1}(\lambda_i))^2 \right] \sum_{i=1}^{n} y_i^2 \lambda_i.$$

Recalling that $\sum_{i=1}^{n} y_i^2 \lambda_i = \|\mathbf{e}^{(0)}\|_{\mathrm{A}}^2$, we have

$$\frac{\|\mathbf{e}^{(k+1)}\|_{\mathrm{A}}}{\|\mathbf{e}^{(0)}\|_{\mathrm{A}}} \leq \min_{q_{k+1} \in \mathbb{P}_{k+1}^{0,1}} \max_{\lambda_i \in \sigma(\mathrm{A})} |q_{k+1}(\lambda_i)|.$$

Let us now recall the following property

Property 4.6 *The problem of minimizing* $\max_{\lambda_n \leq z \leq \lambda_1} |q(z)|$ *over the space* $\mathbb{P}_{k+1}^{0,1}([\lambda_n, \lambda_1])$ *admits a unique solution, given by the polynomial*

$$p_{k+1}(\xi) = T_{k+1}\left(\frac{\lambda_1 + \lambda_n - 2\xi}{\lambda_1 - \lambda_n} \right) / C_{k+1}, \qquad \xi \in [\lambda_n, \lambda_1],$$

where $C_{k+1} = T_{k+1}(\frac{\lambda_1+\lambda_n}{\lambda_1-\lambda_n})$ *and* T_{k+1} *is the Chebyshev polynomial of degree* $k+1$ *(see Section* 10.10*). The value of the minimum is* $1/C_{k+1}$.

Using this property we get

$$\frac{\|\mathbf{e}^{(k+1)}\|_{\mathrm{A}}}{\|\mathbf{e}^{(0)}\|_{\mathrm{A}}} \leq \frac{1}{T_{k+1}\left(\dfrac{\lambda_1 + \lambda_n}{\lambda_1 - \lambda_n} \right)}$$

from which the thesis follows since in the case of a symmetric positive definite matrix

$$\frac{1}{C_{k+1}} = \frac{2c^{k+1}}{1 + c^{2(k+1)}}.$$

◇

The generic k-th iteration of the conjugate gradient method is well defined only if the descent direction $\mathbf{p}^{(k)}$ is non null. Besides, if $\mathbf{p}^{(k)} = \mathbf{0}$, then the iterate $\mathbf{x}^{(k)}$ must necessarily coincide with the solution $\mathbf{x}$ of the system. Moreover, irrespectively of the choice of the parameters β_k, one can show (see [Axe94], p. 463) that the sequence $\mathbf{x}^{(k)}$ generated by the CG method is such that either $\mathbf{x}^{(k)} \neq \mathbf{x}$, $\mathbf{p}^{(k)} \neq \mathbf{0}$, $\alpha_k \neq 0$ for any k, or there must exist an integer m such that $\mathbf{x}^{(m)} = \mathbf{x}$, where $\mathbf{x}^{(k)} \neq \mathbf{x}$, $\mathbf{p}^{(k)} \neq \mathbf{0}$ and $\alpha_k \neq 0$ for $k = 0, 1, \ldots, m-1$.

The particular choice made for β_k in (4.45) ensures that $m \leq n$. In absence of rounding errors, the CG method can thus be regarded as being a direct method, since it terminates after a finite number of steps. However, for matrices of large size, it is usually employed as an iterative scheme,

where the iterations are stopped when the error gets below a fixed tolerance. In this respect, the dependence of the error reduction factor on the condition number of the matrix is more favorable than for the gradient method. We also notice that estimate (4.47) is often overly pessimistic and does not account for the fact that in this method, unlike what happens for the gradient method, the convergence is influenced by the *whole* spectrum of A, and not only by its extremal eigenvalues.

Remark 4.3 (Effect of rounding errors) The termination property of the CG method is rigorously valid only in exact arithmetic. The cumulating rounding errors prevent the descent directions from being A-conjugate and can even generate null denominators in the computation of coefficients α_k and β_k. This latter phenomenon, known as *breakdown*, can be avoided by introducing suitable stabilization procedures; in such an event, we speak about stabilized gradient methods.

Despite the use of these strategies, it may happen that the CG method fails to converge (in finite arithmetic) after n iterations. In such a case, the only reasonable possibility is to restart the iterative process, taking as residual the last computed one. By so doing, the *cyclic CG method* or *CG method with restart* is obtained, for which, however, the convergence properties of the original CG method are no longer valid. ■

4.3.5 The Preconditioned Conjugate Gradient Method

If P is a symmetric and positive definite preconditioning matrix, the preconditioned conjugate gradient method (PCG) consists of applying the CG method to the preconditioned system

$$\mathrm{P}^{-1/2}\mathrm{A}\mathrm{P}^{-1/2}\mathbf{y} = \mathrm{P}^{-1/2}\mathbf{b}, \qquad \text{with } \mathbf{y} = \mathrm{P}^{1/2}\mathbf{x}.$$

In practice, the method is implemented without explicitly requiring the computation of $\mathrm{P}^{1/2}$ or $\mathrm{P}^{-1/2}$. After some algebra, the following scheme is obtained:
given $\mathbf{x}^{(0)}$ and setting $\mathbf{r}^{(0)} = \mathbf{b} - \mathrm{A}\mathbf{x}^{(0)}$, $\mathbf{z}^{(0)} = \mathrm{P}^{-1}\mathbf{r}^{(0)}$ e $\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$, the k-th iteration reads

$$\alpha_k = \frac{\mathbf{p}^{(k)^T}\mathbf{r}^{(k)}}{\mathbf{p}^{(k)^T}\mathrm{A}\mathbf{p}^{(k)}}$$

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

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathrm{A}\mathbf{p}^{(k)}$$

$$\mathrm{P}\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$$

$$\beta_k = \frac{(\mathrm{A}\mathbf{p}^{(k)})^T\mathbf{z}^{(k+1)}}{(\mathrm{A}\mathbf{p}^{(k)})^T\mathbf{p}^{(k)}}$$

$$\mathbf{p}^{(k+1)} = \mathbf{z}^{(k+1)} - \beta_k \mathbf{p}^{(k)}.$$

The computational cost is increased with respect to the CG method, as one needs to solve at each step the linear system $\mathrm{P}\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$. For this system the symmetric preconditioners examined in Section 4.3.2 can be used. The error estimate is the same as for the nonpreconditioned method, provided to replace the matrix A by $\mathrm{P}^{-1}\mathrm{A}$.

In Program 20 an implementation of the PCG method is reported. For a description of the input/output parameters, see Program 19.

Program 20 - conjgrad : Preconditioned conjugate gradient method

```
function [x, error, niter, flag] = conjgrad(A, x, b, P, maxit, tol)
flag = 0; niter = 0; bnrm2 = norm( b );
if  ( bnrm2 == 0.0 ), bnrm2 = 1.0; end
r = b - A*x; error = norm( r ) / bnrm2;
if ( error < tol ) return, end
for niter = 1:maxit
   z  = P \ r; rho = (r'*z);
   if niter > 1
      beta = rho / rho1;   p = z + beta*p;
   else
      p = z;
   end
   q = A*p;              alpha = rho / (p'*q );
   x = x + alpha * p;    r = r - alpha*q;
   error = norm( r ) / bnrm2;
   if ( error <= tol ), break, end
   rho1 = rho;
end
if ( error > tol ) flag = 1; end
```

Example 4.7 Let us consider again the linear system of Example 4.6. The CG method has been run with the same input data as in the previous example. It converges in 3 iterations for $m = 16$ and in 45 iterations for $m = 400$. Using the same preconditioner as in Example 4.6, the number of iterations decreases from 45 to 26, in the case $m = 400$. •

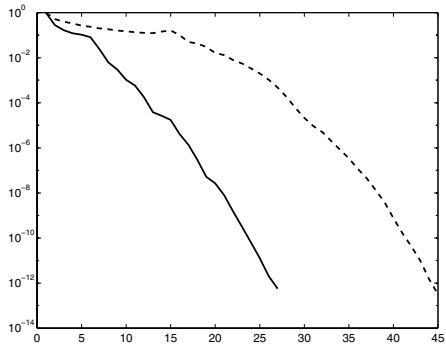

FIGURE 4.8. Behavior of the residual, normalized to the right-hand side, as a function of the number of iterations for the conjugate gradient method applied to the systems of Example 4.6 in the case $m = 400$. The curve in dashed line refers to the non preconditioned method, while the curve in solid line refers to the preconditioned one

4.3.6 The Alternating-Direction Method

Assume that $\mathrm{A} = \mathrm{A}_1 + \mathrm{A}_2$, with A_1 and A_2 symmetric and positive definite. The *alternating direction* method (ADI), as introduced by Peaceman and Rachford [PJ55], is an iterative scheme for (3.2) which consists of solving the following systems $\forall k \geq 0$

$$\begin{aligned} (\mathrm{I} + \alpha_1 \mathrm{A}_1)\mathbf{x}^{(k+1/2)} &= (\mathrm{I} - \alpha_1 \mathrm{A}_2)\mathbf{x}^{(k)} + \alpha_1 \mathbf{b}, \\ (\mathrm{I} + \alpha_2 \mathrm{A}_2)\mathbf{x}^{(k+1)} &= (\mathrm{I} - \alpha_2 \mathrm{A}_1)\mathbf{x}^{(k+1/2)} + \alpha_2 \mathbf{b} \end{aligned} \tag{4.50}$$

where α_1 and α_2 are two real parameters. The ADI method can be cast in the form (4.2) setting

$$\begin{aligned} \mathrm{B} &= (\mathrm{I} + \alpha_2 \mathrm{A}_2)^{-1}(\mathrm{I} - \alpha_2 \mathrm{A}_1)(\mathrm{I} + \alpha_1 \mathrm{A}_1)^{-1}(\mathrm{I} - \alpha_1 \mathrm{A}_2), \\ \mathbf{f} &= \left[\alpha_1(\mathrm{I} - \alpha_2 \mathrm{A}_1)(\mathrm{I} + \alpha_1 \mathrm{A}_1)^{-1} + \alpha_2 \mathrm{I}\right] \mathbf{b}. \end{aligned}$$

Both B and $\mathbf{f}$ depend on α_1 and α_2. The following estimate holds

$$\rho(\mathrm{B}) \leq \max_{i=1,\ldots,n} \left| \frac{1 - \alpha_2 \lambda_i^{(1)}}{1 + \alpha_1 \lambda_i^{(1)}} \right| \max_{i=1,\ldots,n} \left| \frac{1 - \alpha_1 \lambda_i^{(2)}}{1 + \alpha_2 \lambda_i^{(2)}} \right|,$$

where $\lambda_1^{(i)}$ and $\lambda_2^{(i)}$, for $i = 1, \ldots, n$, are the eigenvalues of A_1 and A_2, respectively. The method converges if $\rho(\mathrm{B}) < 1$, which is always verified if $\alpha_1 = \alpha_2 = \alpha > 0$. Moreover (see [Axe94]) if $\gamma \leq \lambda_i^{(j)} \leq \delta \ \forall i = 1, \ldots, n$, $\forall j = 1, 2$, for suitable γ and δ then the ADI method converges with the choice $\alpha_1 = \alpha_2 = 1/\sqrt{\delta\gamma}$, provided that γ/δ tends to 0 as the size of A grows. In such an event the corresponding spectral radius satisfies

$$\rho(\mathrm{B}) \leq \left(\frac{1 - \sqrt{\gamma/\delta}}{1 + \sqrt{\gamma/\delta}} \right)^2 .$$

4.4 Methods Based on Krylov Subspace Iterations

In this section we introduce iterative methods based on Krylov subspace iterations. For the proofs and further analysis, we refer to [Saa96], [Axe94] and [Hac94].

Consider the Richardson method (4.24) with P=I; the residual at the k-th step can be related to the initial residual as

$$\mathbf{r}^{(k)} = \prod_{j=0}^{k-1} (\mathrm{I} - \alpha_j \mathrm{A}) \mathbf{r}^{(0)} \tag{4.51}$$

so that $\mathbf{r}^{(k)} = p_k(\mathrm{A})\mathbf{r}^{(0)}$, where $p_k(\mathrm{A})$ is a polynomial in A of degree k. If we introduce the space

$$K_m(\mathrm{A}; \mathbf{v}) = \mathrm{span}\left\{ \mathbf{v}, \mathrm{A}\mathbf{v}, \ldots, \mathrm{A}^{m-1}\mathbf{v} \right\}, \tag{4.52}$$

it immediately appears from (4.51) that $\mathbf{r}^{(k)} \in K_{k+1}(\mathrm{A}; \mathbf{r}^{(0)})$. The space defined in (4.52) is called the *Krylov subspace* of order m. It is a subspace of the set spanned by all the vectors $\mathbf{u} \in \mathbb{R}^n$ that can be written as $\mathbf{u} = p_{m-1}(\mathrm{A})\mathbf{v}$, where p_{m-1} is a polynomial in A of degree $\leq m - 1$.

In an analogous manner as for (4.51), it is seen that the iterate $\mathbf{x}^{(k)}$ of the Richardson method is given by

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \sum_{j=0}^{k-1} \alpha_j \mathbf{r}^{(j)}$$

so that $\mathbf{x}^{(k)}$ belongs to the following space

$$W_k = \left\{ \mathbf{v} = \mathbf{x}^{(0)} + \mathbf{y}, \ \mathbf{y} \in K_k(\mathrm{A}; \mathbf{r}^{(0)}) \right\}. \tag{4.53}$$

Notice also that $\sum_{j=0}^{k-1} \alpha_j \mathbf{r}^{(j)}$ is a polynomial in A of degree less than $k-1$. In the non preconditioned Richardson method we are thus looking for an

approximate solution to $\mathbf{x}$ in the space W_k. More generally, we can think of devising methods that search for approximate solutions of the form

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + q_{k-1}(\mathrm{A})\mathbf{r}^{(0)}, \tag{4.54}$$

where q_{k-1} is a polynomial selected in such a way that $\mathbf{x}^{(k)}$ be, in a sense that must be made precise, the best approximation of $\mathbf{x}$ in W_k. A method that looks for a solution of the form (4.54) with W_k defined as in (4.53) is called a *Krylov method*.

A first question concerning Krylov subspace iterations is whether the dimension of $K_m(\mathrm{A};\mathbf{v})$ increases as the order m grows. A partial answer is provided by the following result.

Property 4.7 *Let $\mathrm{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{v} \in \mathbb{R}^n$. The Krylov subspace $K_m(\mathrm{A};\mathbf{v})$ has dimension equal to m iff the degree of $\mathbf{v}$ with respect to A, denoted by $\deg_{\mathrm{A}}(\mathbf{v})$, is not less than m, where the degree of $\mathbf{v}$ is defined as the minimum degree of a monic non null polynomial p in A, for which $p(\mathrm{A})\mathbf{v} = \mathbf{0}$.*

The dimension of $K_m(\mathrm{A};\mathbf{v})$ is thus equal to the minimum between m and the degree of $\mathbf{v}$ with respect to A and, as a consequence, the dimension of the Krylov subspaces is certainly a nondecreasing function of m. Notice that the degree of $\mathbf{v}$ cannot be greater than n due to the Cayley-Hamilton Theorem (see Section 1.7).

Example 4.8 Consider the matrix $\mathrm{A} = \mathrm{tridiag}_4(-1, 2, -1)$. The vector $\mathbf{v} = (1, 1, 1, 1)^T$ has degree 2 with respect to A since $p_2(\mathrm{A})\mathbf{v} = \mathbf{0}$ with $p_2(\mathrm{A}) = \mathrm{I}_4 - 3\mathrm{A} + \mathrm{A}^2$, while there is no monic polynomial p_1 of degree 1 for which $p_1(\mathrm{A})\mathbf{v} = \mathbf{0}$. As a consequence, all Krylov subspaces from $K_2(\mathrm{A};\mathbf{v})$ on, have dimension equal to 2. The vector $\mathbf{w} = (1, 1, -1, 1)^T$ has, instead, degree 4 with respect to A. •

For a fixed m, it is possible to compute an orthonormal basis for $K_m(\mathrm{A};\mathbf{v})$ using the so-called *Arnoldi algorithm*.

Setting $\mathbf{v}_1 = \mathbf{v}/\|\mathbf{v}\|_2$, this method generates an orthonormal basis $\{\mathbf{v}_i\}$ for $K_m(\mathrm{A};\mathbf{v}_1)$ using the Gram-Schmidt procedure (see Section 3.4.3). For $k = 1, \ldots, m$, the Arnoldi algorithm computes

$$\begin{aligned} &h_{ik} = \mathbf{v}_i^T \mathrm{A}\mathbf{v}_k, \qquad\qquad i = 1, 2, \ldots, k, \\ &\mathbf{w}_k = \mathrm{A}\mathbf{v}_k - \sum_{i=1}^{k} h_{ik}\mathbf{v}_i, \quad h_{k+1,k} = \|\mathbf{w}_k\|_2. \end{aligned} \tag{4.55}$$

If $\mathbf{w}_k = \mathbf{0}$ the process terminates and in such a case we say that a *breakdown* of the algorithm has occurred; otherwise, we set $\mathbf{v}_{k+1} = \mathbf{w}_k/\|\mathbf{w}_k\|_2$ and the algorithm restarts, incrementing k by 1.

It can be shown that if the method terminates at the step m then the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$ form a basis for $K_m(\mathrm{A}; \mathbf{v})$. In such a case, if we denote by $\mathrm{V}_m \in \mathbb{R}^{n \times m}$ the matrix whose columns are the vectors $\mathbf{v}_i$, we have

$$\mathrm{V}_m^T \mathrm{A} \mathrm{V}_m = \mathrm{H}_m, \quad \mathrm{V}_{m+1}^T \mathrm{A} \mathrm{V}_m = \widehat{\mathrm{H}}_m, \tag{4.56}$$

where $\widehat{\mathrm{H}}_m \in \mathbb{R}^{(m+1) \times m}$ is the upper Hessenberg matrix whose entries h_{ij} are given by (4.55) and $\mathrm{H}_m \in \mathbb{R}^{m \times m}$ is the restriction of $\widehat{\mathrm{H}}_m$ to the first m rows and m columns.

The algorithm terminates at an intermediate step $k < m$ iff $\deg_{\mathrm{A}}(\mathbf{v}_1) = k$. As for the stability of the procedure, all the considerations valid for the Gram-Schmidt method hold. For more efficient and stable computational variants of (4.55), we refer to [Saa96].

The functions `arnoldi_alg` and `GSarnoldi`, invoked by Program 21, provide an implementation of the Arnoldi algorithm. In output, the columns of `V` contain the vectors of the generated basis, while the matrix `H` stores the coefficients h_{ik} computed by the algorithm. If m steps are carried out, $\mathtt{V} = \mathrm{V}_m$ and $\mathtt{H}(1:m, 1:m) = \mathrm{H}_m$.

Program 21 - arnoldi_alg : The Arnoldi algorithm

```
function [V,H]=arnoldi_alg(A,v,m)
v=v/norm(v,2); V=[v1];  H=[];  k=0;
while k <= m-1
  [k,V,H] = GSarnoldi(A,m,k,V,H);
end

function [k,V,H]=GSarnoldi(A,m,k,V,H)
k=k+1;  H=[H,V(:,1:k)'*A*V(:,k)];
s=0;  for i=1:k,  s=s+H(i,k)*V(:,i);  end
w=A*V(:,k)-s;      H(k+1,k)=norm(w,2);
if ( H(k+1,k) <= eps ) & ( k < m )
  V=[V,w/H(k+1,k)];
else
  k=m+1;
end
```

Having introduced an algorithm for generating the basis for a Krylov subspace of any order, we can now solve the linear system (3.2) by a Krylov method. As already noticed, for all of these methods the iterate $\mathbf{x}^{(k)}$ is always of the form (4.54) and, for a given $\mathbf{r}^{(0)}$, the vector $\mathbf{x}^{(k)}$ is selected as being the unique element in W_k which satisfies a criterion of minimal distance from $\mathbf{x}$. Thus, the feature distinguishing two different Krylov methods is the criterion for selecting $\mathbf{x}^{(k)}$.

The most natural idea consists of searching for $\mathbf{x}^{(k)} \in W_k$ as the vector which minimizes the Euclidean norm of the error. This approach, how-

ever, does not work in practice since $\mathbf{x}^{(k)}$ would depend on the (unknown) solution $\mathbf{x}$.

Two alternative strategies can be pursued:

1. compute $\mathbf{x}^{(k)} \in W_k$ enforcing that the residual $\mathbf{r}^{(k)}$ is orthogonal to any vector in $K_k(\mathrm{A}; \mathbf{r}^{(0)})$, i.e., we look for $\mathbf{x}^{(k)} \in W_k$ such that

$$\mathbf{v}^T(\mathbf{b} - \mathrm{A}\mathbf{x}^{(k)}) = 0 \qquad \forall \mathbf{v} \in K_k(\mathrm{A}; \mathbf{r}^{(0)}); \tag{4.57}$$

2. compute $\mathbf{x}^{(k)} \in W_k$ minimizing the Euclidean norm of the residual $\|\mathbf{r}^{(k)}\|_2$, i.e.

$$\|\mathbf{b} - \mathrm{A}\mathbf{x}^{(k)}\|_2 = \min_{\mathbf{v} \in W_k} \|\mathbf{b} - \mathrm{A}\mathbf{v}\|_2. \tag{4.58}$$

Satisfying (4.57) leads to the Arnoldi method for linear systems (more commonly known as FOM, *full orthogonalization method*), while satisfying (4.58) yields the GMRES (*generalized minimum residual*) method.

In the two forthcoming sections we shall assume that k steps of the Arnoldi algorithm have been carried out, in such a way that an orthonormal basis for $K_k(\mathrm{A}; \mathbf{r}^{(0)})$ has been generated and stored into the column vectors of the matrix V_k with $\mathbf{v}_1 = \mathbf{r}^{(0)}/\|\mathbf{r}^{(0)}\|_2$. In such a case the new iterate $\mathbf{x}^{(k)}$ can always be written as

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \mathrm{V}_k \mathbf{z}^{(k)}, \tag{4.59}$$

where $\mathbf{z}^{(k)}$ must be selected according to a fixed criterion.

4.4.1 The Arnoldi Method for Linear Systems

Let us enforce that $\mathbf{r}^{(k)}$ be orthogonal to $K_k(\mathrm{A}; \mathbf{r}^{(0)})$ by requiring that (4.57) holds for all the basis vectors $\mathbf{v}_i$, i.e.

$$\mathrm{V}_k^T \mathbf{r}^{(k)} = 0. \tag{4.60}$$

Since $\mathbf{r}^{(k)} = \mathbf{b} - \mathrm{A}\mathbf{x}^{(k)}$ with $\mathbf{x}^{(k)}$ of the form (4.59), relation (4.60) becomes

$$\mathrm{V}_k^T(\mathbf{b} - \mathrm{A}\mathbf{x}^{(0)}) - \mathrm{V}_k^T \mathrm{A}\mathrm{V}_k \mathbf{z}^{(k)} = \mathrm{V}_k^T \mathbf{r}^{(0)} - \mathrm{V}_k^T \mathrm{A}\mathrm{V}_k \mathbf{z}^{(k)} = 0. \tag{4.61}$$

Due to the orthonormality of the basis and the choice of $\mathbf{v}_1$, $\mathrm{V}_k^T \mathbf{r}^{(0)} = \|\mathbf{r}^{(0)}\|_2 \mathbf{e}_1$, $\mathbf{e}_1$ being the first unit vector of $\mathbb{R}^m$. Recalling (4.56), from (4.61) it turns out that $\mathbf{z}^{(k)}$ is the solution to the linear system

$$\mathrm{H}_k \mathbf{z}^{(k)} = \|\mathbf{r}^{(0)}\|_2 \mathbf{e}_1. \tag{4.62}$$

Once $\mathbf{z}^{(k)}$ is known, we can compute $\mathbf{x}^{(k)}$ from (4.59). Since H_k is an upper Hessenberg matrix, the linear system in (4.62) can be easily solved, for instance, resorting to the LU factorization of H_k.

We notice that the method, if working in exact arithmetic, cannot execute more than n steps and that it terminates after $m < n$ steps only if a *breakdown* in the Arnoldi algorithm occurs. As for the convergence of the method, the following result holds.

Theorem 4.13 *In exact arithmetic the Arnoldi method yields the solution of* (3.2) *after at most n iterations.*

Proof. If the method terminates at the n-th iteration, then it must necessarily be $\mathbf{x}^{(n)} = \mathbf{x}$ since $K_n(\mathrm{A}; \mathbf{r}^{(0)}) = \mathbb{R}^n$. Conversely, if a breakdown occurs after m iterations, for a suitable $m < n$, then $\mathbf{x}^{(m)} = \mathbf{x}$. Indeed, inverting the first relation in (4.56), we get

$$\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathrm{V}_m \mathbf{z}^{(m)} = \mathbf{x}^{(0)} + \mathrm{V}_m \mathrm{H}_m^{-1} \mathrm{V}_m^T \mathbf{r}^{(0)} = \mathrm{A}^{-1}\mathbf{b}.$$

◇

In its naive form, FOM does not require an explicit computation of the solution or the residual, unless a breakdown occurs. Therefore, monitoring its convergence (by computing, for instance, the residual at each step) might be computationally expensive. The residual, however, is available without explicitly requiring to compute the solution since at the k-th step we have

$$\|\mathbf{b} - \mathrm{A}\mathbf{x}^{(k)}\|_2 = h_{k+1,k} |\mathbf{e}_k^T \mathbf{z}_k|$$

and, as a consequence, one can decide to stop the method if

$$h_{k+1,k} |\mathbf{e}_k^T \mathbf{z}_k| / \|\mathbf{r}^{(0)}\|_2 \leq \varepsilon \tag{4.63}$$

$\varepsilon > 0$ being a fixed tolerance.

The most relevant consequence of Theorem 4.13 is that FOM can be regarded as a direct method, since it yields the exact solution after a finite number of steps. However, this fails to hold when working in floating point arithmetic due to the cumulating rounding errors. Moreover, if we also account for the high computational effort, which, for a number of m steps and a sparse matrix of order n with n_z nonzero entries, is of the order of $2(n_z + mn)$ flops, and the large memory occupation needed to store the matrix V_m, we conclude that the Arnoldi method cannot be used in the practice, except for small values of m.

Several remedies to this drawback are available, one of which consisting of preconditioning the system (using, for instance, one of the preconditioners proposed in Section 4.3.2). Alternatively, we can also introduce some modified versions of the Arnoldi method following two approaches:

1. no more than m consecutive steps of FOM are taken, m being a small fixed number (usually, $m \simeq 10$). If the method fails to converge, we set

$\mathbf{x}^{(0)} = \mathbf{x}^{(m)}$ and FOM is repeated for other m steps. This procedure is carried out until convergence is achieved. This method, known as FOM(m) or FOM with *restart*, reduces the memory occupation, only requiring to store matrices with m columns at most;

2. a limitation is set on the number of directions involved in the orthogonalization procedure in the Arnoldi algorithm, yielding the incomplete orthogonalization method or IOM. In the practice, the k-th step of the Arnoldi algorithm generates a vector $\mathbf{v}_{k+1}$ which is orthonormal, at most, to the q preceding vectors, where q is fixed according to the amount of available memory.

It is worth noticing that Theorem 4.13 does no longer hold for the methods stemming from the two strategies above.

Program 22 provides an implementation of the FOM algorithm with a stopping criterion based on the residual (4.63). The input parameter `m` is the maximum admissible size of the Krylov subspace that is being generated and represents, as a consequence, the maximum admissible number of iterations.

Program 22 - arnoldi_met : The Arnoldi method for linear systems

```
function [x,k]=arnoldi_met(A,b,m,x0,toll)
r0=b-A*x0;  nr0=norm(r0,2);
if nr0 ~= 0
   v1=r0/nr0; V=[v1]; H=[]; k=0; istop=0;
   while (k <= m-1) & (istop == 0)
     [k,V,H] = GSarnoldi(A,m,k,V,H);
     [nr,nc]=size(H); e1=eye(nc);
     y=(e1(:,1)'*nr0)/H(1:nc,:);
     residual = H(nr,nc)*abs(y*e1(:,nc));
     if residual <= toll
       istop = 1; y=y';
     end
   end
   if istop==0
     [nr,nc]=size(H);  e1=eye(nc);
     y=(e1(:,1)'*nr0)/H(1:nc,:); y=y';
   end
   x=x0+V(:,1:nc)*y;
else
   x=x0;
end
```

Example 4.9 Let us solve the linear system $A\mathbf{x} = \mathbf{b}$ with $A = \text{tridiag}_{100}(-1, 2, -1)$ and $\mathbf{b}$ such that the solution is $\mathbf{x} = \mathbf{1}^T$. The initial vector is $\mathbf{x}^{(0)} = \mathbf{0}^T$ and `toll`$=10^{-10}$. The method converges in 50 iterations and Figure 4.9 reports its convergence history. Notice the sudden, dramatic, reduction of the residual,

which is a typical warning that the last generated subspace W_k is sufficiently rich to contain the exact solution of the system. •

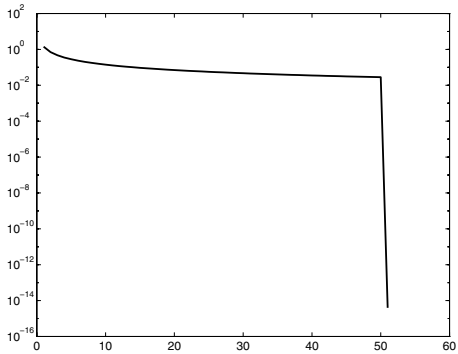

FIGURE 4.9. The behavior of the residual as a function of the number of iterations for the Arnoldi method applied to the linear system in Example 4.9

4.4.2 The GMRES Method

This method is characterized by selecting $\mathbf{x}^{(k)}$ in such a way to minimize the Euclidean norm of the residual at each k-th step. Recalling (4.59) we have

$$\mathbf{r}^{(k)} = \mathbf{r}^{(0)} - \mathrm{A}\mathrm{V}_k\mathbf{z}^{(k)}, \tag{4.64}$$

but, since $\mathbf{r}^{(0)} = \mathbf{v}_1\|\mathbf{r}^{(0)}\|_2$ and (4.56) holds, relation (4.64) becomes

$$\mathbf{r}^{(k)} = \mathrm{V}_{k+1}(\|\mathbf{r}^{(0)}\|_2\mathbf{e}_1 - \widehat{\mathrm{H}}_k\mathbf{z}^{(k)}), \tag{4.65}$$

where $\mathbf{e}_1$ is the first unit vector of $\mathbb{R}^{k+1}$. Therefore, in the GMRES method the solution at step k can be computed through (4.59) as

$$\mathbf{z}^{(k)} \text{ chosen in such a way to minimize } \| \|\mathbf{r}^{(0)}\|_2\mathbf{e}_1 - \widehat{\mathrm{H}}_k\mathbf{z}^{(k)}\|_2 \tag{4.66}$$

(the matrix V_{k+1} appearing in (4.65) does not change the value of $\|\cdot\|_2$ since it is orthogonal). Having to solve at each step a least-squares problem of size k, the GMRES method will be the more effective the smaller is the number of iterations. Exactly as for the Arnoldi method, the GMRES method terminates at most after n iterations, yielding the exact solution. Premature stops are due to a *breakdown* in the orthonormalization Arnoldi algorithm. More precisely, we have the following result.

Property 4.8 *A breakdown occurs for the GMRES method at a step m (with $m < n$) iff the computed solution $\mathbf{x}^{(m)}$ coincides with the exact solution to the system.*

A basic implementation of the GMRES method is provided in Program 23. This latter requires in input the maximum admissible size `m` for the Krylov subspace and the tolerance `toll` on the Euclidean norm of the residual normalized to the initial residual. This implementation of the method computes the solution $\mathbf{x}^{(k)}$ at each step in order to evaluate the residual, with a consequent increase of the computational effort.

Program 23 - GMRES : The GMRES method for linear systems

```
function [x,k]=gmres(A,b,m,toll,x0)
r0=b-A*x0;  nr0=norm(r0,2);
if nr0 ~= 0
  v1=r0/nr0; V=[v1]; H=[]; k=0; residual=1;
  while k <= m-1 & residual > toll,
    [k,V,H] = GSarnoldi(A,m,k,V,H);
    [nr,nc]=size(H);    y=(H'*H) \  (H'*nr0*[1;zeros(nr-1,1)]);
    x=x0+V(:,1:nc)*y;   residual = norm(b-A*x,2)/nr0;
  end
else
  x=x0;
end
```

To improve the efficiency of the GMRES algorithm it is necessary to devise a stopping criterion which does not require the explicit evaluation of the residual at each step. This is possible, provided that the linear system with upper Hessenberg matrix $\widehat{\mathrm{H}}_k$ is appropriately solved.

In practice, $\widehat{\mathrm{H}}_k$ is transformed into an upper triangular matrix $\mathrm{R}_k \in \mathbb{R}^{(k+1)\times k}$ with $r_{k+1,k} = 0$ such that $\mathrm{Q}_k^T\mathrm{R}_k = \widehat{\mathrm{H}}_k$, where Q_k is a matrix obtained as the product of k Givens rotations (see Section 5.6.3). Then, since Q_k is orthogonal, it can be seen that minimizing $\|\|\mathbf{r}^{(0)}\|_2\mathbf{e}_1 - \widehat{\mathrm{H}}_k\mathbf{z}^{(k)}\|_2$ is equivalent to minimize $\|\mathbf{f}_k - \mathrm{R}_k\mathbf{z}^{(k)}\|_2$, with $\mathbf{f}_k = \mathrm{Q}_k\|\mathbf{r}^{(0)}\|_2\mathbf{e}_1$. It can also be shown that the $k+1$-th component of $\mathbf{f}_k$ is, in absolute value, the Euclidean norm of the residual at the k-th step.

As FOM, the GMRES method entails a high computational effort and a large amount of memory, unless convergence occurs after few iterations. For this reason, two variants of the algorithm are available, one named GMRES(m) and based on the *restart* after m steps, the other named Quasi-GMRES or QGMRES and based on stopping the Arnoldi orthogonalization process. It is worth noting that these two methods do not enjoy Property 4.8.

Remark 4.4 (Projection methods) Denoting by Y_k and L_k two generic m-dimensional subspaces of $\mathbb{R}^n$, we call *projection method* a process which generates an approximate solution $\mathbf{x}^{(k)}$ at step k, enforcing that $\mathbf{x}^{(k)} \in Y_k$

and that the residual $\mathbf{r}^{(k)} = \mathbf{b} - \mathrm{A}\mathbf{x}^{(k)}$ be orthogonal to L_k. If $Y_k = L_k$, the projection process is said to be orthogonal, oblique otherwise (see [Saa96]).

The Krylov subspace iterations can be regarded as being projection methods. For instance, the Arnoldi method is an orthogonal projection method where $L_k = Y_k = K_k(\mathrm{A}; \mathbf{r}^{(0)})$, while the GMRES method is an oblique projection method with $Y_k = K_k(\mathrm{A}; \mathbf{r}^{(0)})$ and $L_k = \mathrm{A}Y_k$. It is worth noticing that some classical methods introduced in previous sections fall into this category. For example, the Gauss-Seidel method is an orthogonal projection method where at the k-th step $K_k(\mathrm{A}; \mathbf{r}^{(0)}) = \mathrm{span}(\mathbf{e}_k)$, with $k = 1, \ldots, n$. The projection steps are carried out cyclically from 1 to n until convergence. ■

4.4.3 The Lanczos Method for Symmetric Systems

The Arnoldi algorithm simplifies considerably if A is symmetric since the matrix H_m is tridiagonal and symmetric (indeed, from (4.56) it turns out that H_m must be symmetric, so that, being upper Hessenberg by construction, it must necessarily be tridiagonal). In such an event the method is more commonly known as the *Lanczos algorithm*. For ease of notation, we henceforth let $\alpha_i = h_{ii}$ and $\beta_i = h_{i-1,i}$.

An implementation of the Lanczos algorithm is provided in Program 24. Vectors `alpha` and `beta` contain the coefficients α_i and β_i computed by the scheme.

Program 24 - Lanczos : The Lanczos method for linear systems

```
function [V,alpha,beta]=lanczos(A,m)
n=size(A); V=[0*[1:n]',[1,0*[1:n-1]]'];
beta(1)=0; normb=1; k=1;
while  k <= m & normb >= eps
  vk = V(:,k+1);          w = A*vk-beta(k)*V(:,k);
  alpha(k)= w'*vk;        w = w - alpha(k)*vk
  normb = norm(w,2);
  if normb ~= 0
    beta(k+1)=normb;    V=[V,w/normb];    k=k+1;
  end
end
[n,m]=size(V); V=V(:,2:m-1);
alpha=alpha(1:n); beta=beta(2:n);
```

The algorithm, which is far superior to Arnoldi's one as far as memory saving is concerned, is not numerically stable since only the first generated vectors are actually orthogonal. For this reason, several stable variants have been devised.

As in previous cases, also the Lanczos algorithm can be employed as a solver for linear systems, yielding a symmetric form of the FOM method. It

can be shown that $\mathbf{r}^{(k)} = \gamma_k \mathbf{v}_{k+1}$, for a suitable γ_k (analogously to (4.63)) so that the residuals are all mutually orthogonal.

Remark 4.5 (The conjugate gradient method) If A is symmetric and positive definite, starting from the Lanczos method for linear systems it is possible to derive the conjugate gradient method already introduced in Section 4.3.4 (see [Saa96]). The conjugate gradient method is a variant of the Lanczos method where the orthonormalization process remains incomplete.

As a matter of fact, the A-conjugate directions of the CG method can be characterized as follows. If we carry out at the generic k-th step the LU factorization $\mathrm{H}_k = \mathrm{L}_k\mathrm{U}_k$, with L_k (U_k) lower (upper) bidiagonal, the iterate $\mathbf{x}^{(k)}$ of the Lanczos method for systems reads

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \mathrm{P}_k\mathrm{L}_k^{-1}\|\mathbf{r}^{(0)}\|_2\mathbf{e}_1,$$

with $\mathrm{P}_k = \mathrm{V}_k\mathrm{U}_k^{-1}$. The column vectors of P_k are mutually A-conjugate. Indeed, $\mathrm{P}_k^T\mathrm{A}\mathrm{P}_k$ is symmetric and bidiagonal since

$$\mathrm{P}_k^T\mathrm{A}\mathrm{P}_k = \mathrm{U}_k^{-T}\mathrm{H}_k\mathrm{U}_k^{-1} = \mathrm{U}_k^{-T}\mathrm{L}_k,$$

so that it must necessarily be diagonal. As a result, $\mathbf{p}_j^T\mathrm{A}\mathbf{p}_i = 0$ if $i \neq j$, having denoted by $\mathbf{p}_i$ the i-th column vector of matrix P_k. ■

As happens for the FOM method, also the GMRES method simplifies if A is symmetric. The resulting scheme is called *conjugate residuals* or CR method since it enjoys the property that the residuals are mutually A-conjugate. Variants of this method are the generalized conjugate residuals method (GCR) and the method commonly known as ORTHOMIN (obtained by truncation of the orthonormalization process as done for the IOM method).

4.5 The Lanczos Method for Unsymmetric Systems

The Lanczos orthogonalization process can be extended to deal with unsymmetric matrices through a *bi-orthogonalization* procedure as follows. Two bases, $\{\mathbf{v}_i\}_{i=1}^m$ and $\{\mathbf{z}_i\}_{i=1}^m$, are generated for the subspaces $\mathrm{K}_m(\mathrm{A};\mathbf{v}_1)$ and $\mathrm{K}_m(\mathrm{A}^T;\mathbf{z}_1)$, respectively, with $\mathbf{z}_1^T\mathbf{v}_1 = 1$, such that

$$\mathbf{z}_i^T\mathbf{v}_j = \delta_{ij}, \qquad i,j = 1,\dots,m. \tag{4.67}$$

Two sets of vectors satisfying (4.67) are said to be *bi-orthogonal* and can be obtained through the following algorithm: setting $\beta_1 = \gamma_1 = 0$ and $\mathbf{z}_0 = \mathbf{v}_0 = \mathbf{0}^T$, at the generic k-th step, with $k = 1,\dots,m$, we set $\alpha_k = \mathbf{z}_k^T\mathrm{A}\mathbf{v}_k$, then we compute

$$\tilde{\mathbf{v}}_{k+1} = \mathrm{A}\mathbf{v}_k - \alpha_k\mathbf{v}_k - \beta_k\mathbf{v}_{k-1}, \quad \tilde{\mathbf{z}}_{k+1} = \mathrm{A}^T\mathbf{z}_k - \alpha_k\mathbf{z}_k - \gamma_k\mathbf{z}_{k-1}.$$

If $\gamma_{k+1} = \sqrt{|\tilde{\mathbf{z}}_{k+1}^T \tilde{\mathbf{v}}_{k+1}|} = 0$ the algorithm is stopped, otherwise we set $\beta_{k+1} = \tilde{\mathbf{z}}_{k+1}^T \tilde{\mathbf{v}}_{k+1}/\gamma_{k+1}$ and generate two new vectors in the basis as

$$\mathbf{v}_{k+1} = \tilde{\mathbf{v}}_{k+1}/\gamma_{k+1}, \quad \mathbf{z}_{k+1} = \tilde{\mathbf{z}}_{k+1}/\beta_{k+1}.$$

If the process terminates after m steps, denoting by V_m and Z_m the matrices whose columns are the vectors of the basis that has been generated, we have

$$\mathrm{Z}_m^T \mathrm{A} \mathrm{V}_m = \mathrm{T}_m,$$

T_m being the following tridiagonal matrix

$$\mathrm{T}_m = \begin{bmatrix} \alpha_1 & \beta_2 & & 0 \\ \gamma_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_m \\ 0 & & \gamma_m & \alpha_m \end{bmatrix}.$$

As in the symmetric case, the bi-orthogonalization Lanczos algorithm can be utilized to solve the linear system (3.2). For this purpose, for m fixed, once the bases $\{\mathbf{v}_i\}_{i=1}^m$ and $\{\mathbf{z}_i\}_{i=1}^m$ have been constructed, it suffices to set

$$\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathrm{V}_m \mathbf{y}^{(m)},$$

where $\mathbf{y}^{(m)}$ is the solution to the linear system $\mathrm{T}_m \mathbf{y}^{(m)} = \|\mathbf{r}^{(0)}\|_2 \mathbf{e}_1$. It is also possible to introduce a stopping criterion based on the residual, without computing it explicitly, since

$$\|\mathbf{r}^{(m)}\|_2 = |\gamma_{m+1} \mathbf{e}_m^T \mathbf{y}_m| \, \|\mathbf{v}_{m+1}\|_2.$$

An implementation of the Lanczos method for unsymmetric systems is given in Program 25. If a *breakdown* of the algorithm occurs, i.e., if $\gamma_{k+1} = 0$, the method stops returning in output a negative value of the variable `niter` which denotes the number of iterations necessary to reduce the initial residual by a factor `tol1`.

Program 25 - Lanczosnosym : The Lanczos method for unsymmetric systems

```
function [xk,nres,niter]=lanczosnosym(A,b,x0,m,toll)
r0=b-A*x0; nres0=norm(r0,2);
if nres0 ~= 0
  V=r0/nres0;  Z=V; gamma(1)=0; beta(1)=0;  k=1; nres=1;
  while k <= m & nres > toll
     vk=V(:,k);  zk=Z(:,k);
     if    k==1,  vk1=0*vk;   zk1=0*zk;
     else,  vk1=V(:,k-1);  zk1=Z(:,k-1);  end
     alpha(k)=zk'*A*vk;
     tildev=A*vk-alpha(k)*vk-beta(k)*vk1;
     tildez=A'*zk-alpha(k)*zk-gamma(k)*zk1;
     gamma(k+1)=sqrt(abs(tildez'*tildev));
     if gamma(k+1) == 0,    k=m+2;
     else
       beta(k+1)=tildez'*tildev/gamma(k+1);
       Z=[Z,tildez/beta(k+1)];
       V=[V,tildev/gamma(k+1)];
     end
     if k~=m+2
       if k==1
         Tk = alpha;
       else
         Tk=diag(alpha)+diag(beta(2:k),1)+diag(gamma(2:k),-1);
       end
       yk=Tk \ (nres0*[1,0*[1:k-1]]');
       xk=x0+V(:,1:k)*yk;
       nres=abs(gamma(k+1)*[0*[1:k-1],1]*yk)*norm(V(:,k+1),2)/nres0;
       k=k+1;
     end
 end
else
  x=x0;
end
if k==m+2, niter=-k; else, niter=k-1; end
```

Example 4.10 Let us solve the linear system with matrix $\mathrm{A} = \mathrm{tridiag}_{100}(-0.5, 2, -1)$ and right-side $\mathbf{b}$ selected in such a way that the exact solution is $\mathbf{x} = \mathbf{1}^T$. Using Program 25 with `toll`$= 10^{-13}$ and a randomly generated `x0`, the algorithm converges in 59 iterations. Figure 4.10 shows the convergence history reporting the graph of $\|\mathbf{r}^{(k)}\|_2/\|\mathbf{r}^{(0)}\|_2$ as a function of the number of iterations. •

We conclude recalling that some variants of the unsymmetric Lanczos method have been devised, that are characterized by a reduced computational cost. We refer the interested reader to the bibliography below for a

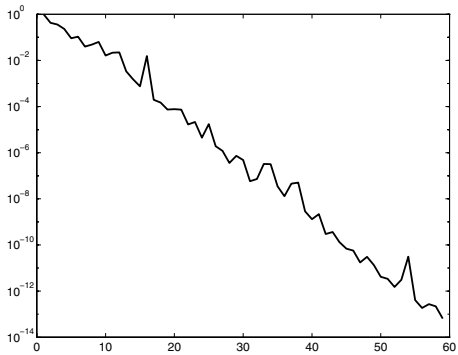

FIGURE 4.10. Graph of the residual normalized to the initial residual as a function of the number of iterations for the Lanczos method applied to the system in Example 4.10

complete description of the algorithms and to the programs included in the MATLAB version of the public domain library `templates` for their efficient implementation [BBC+94].

1. The *bi-conjugate gradient method* (BiCG): it can be derived by the unsymmetric Lanczos method in the same way as the conjugate gradient method is obtained from the FOM method [Fle75];

2. the *Quasi-Minimal Residual method* (QMR): it is analogous to the GMRES method, the only difference being the fact that the Arnoldi orthonormalization process is replaced by the Lanczos bi-orthogonalization;

3. the *conjugate gradient squared method* (CGS): the matrix-vector products involving the transposed matrix A^T are removed. A variant of this method, known as BiCGStab, is characterized by a more regular convergence than provided by the CGS method (see [Son89], [vdV92]).

4.6 Stopping Criteria

In this section we address the problem of how to estimate the error introduced by an iterative method and the number k_{min} of iterations needed to reduce the initial error by a factor ε.
In practice, k_{min} can be obtained by estimating the convergence rate of (4.2), i.e. the rate at which $\|\mathbf{e}^{(k)}\| \to 0$ as k tends to infinity. From (4.4), we get

$$\frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{e}^{(0)}\|} \leq \|\mathrm{B}^k\|,$$

so that $\|\mathrm{B}^k\|$ is an estimate of the reducing factor of the norm of the error after k steps. Typically, the iterative process is continued until $\|\mathbf{e}^{(k)}\|$ has reduced with respect to $\|\mathbf{e}^{(0)}\|$ by a certain factor $\varepsilon < 1$, that is

$$\|\mathbf{e}^{(k)}\| \leq \varepsilon \|\mathbf{e}^{(0)}\|. \tag{4.68}$$

If we assume that $\rho(\mathrm{B}) < 1$, then Property 1.13 implies that there exists a suitable matrix norm $\|\cdot\|$ such that $\|\mathrm{B}\| < 1$. As a consequence, $\|\mathrm{B}^k\|$ tends to zero as k tends to infinity, so that (4.68) can be satisfied for a sufficiently large k such that $\|\mathrm{B}^k\| \leq \varepsilon$ holds. However, since $\|\mathrm{B}^k\| < 1$, the previous inequality amounts to requiring that

$$k \geq \log(\varepsilon) / \left(\frac{1}{k} \log \|\mathrm{B}^k\| \right) = -\log(\varepsilon)/R_k(\mathrm{B}), \tag{4.69}$$

where $R_k(\mathrm{B})$ is the average convergence rate introduced in Definition 4.2. From a practical standpoint, (4.69) is useless, being nonlinear in k; if, however, the asymptotic convergence rate is adopted, instead of the average one, the following estimate for k_{min} is obtained

$$k_{min} \simeq -\log(\varepsilon)/R(\mathrm{B}). \tag{4.70}$$

This latter estimate is usually rather optimistic, as confirmed by Example 4.11.

Example 4.11 For the matrix A_3 of Example 4.2, in the case of Jacobi method, letting $\varepsilon = 10^{-5}$, condition (4.69) is satisfied with $k_{min} = 16$, while (4.70) yields $k_{min} = 15$, with a good agreement between the two estimates. Instead, on the matrix A_4 of Example 4.2, we find that (4.69) is satisfied with $k_{min} = 30$, while (4.70) yields $k_{min} = 26$. •

4.6.1 A Stopping Test Based on the Increment

From the recursive error relation $\mathbf{e}^{(k+1)} = \mathrm{B}\mathbf{e}^{(k)}$, we get

$$\|\mathbf{e}^{(k+1)}\| \leq \|\mathrm{B}\| \|\mathbf{e}^{(k)}\|. \tag{4.71}$$

Using the triangular inequality we get

$$\|\mathbf{e}^{(k+1)}\| \leq \|\mathrm{B}\| (\|\mathbf{e}^{(k+1)}\| + \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|),$$

from which it follows that

$$\|\mathbf{x} - \mathbf{x}^{(k+1)}\| \leq \frac{\|\mathrm{B}\|}{1 - \|\mathrm{B}\|} \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|. \tag{4.72}$$

In particular, taking $k = 0$ in (4.72) and applying recursively (4.71) we also get

$$\|\mathbf{x} - \mathbf{x}^{(k+1)}\| \leq \frac{\|\mathrm{B}\|^{k+1}}{1 - \|\mathrm{B}\|} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|,$$

which can be used to estimate the number of iterations necessary to fulfill the condition $\|\mathbf{e}^{(k+1)}\| \leq \varepsilon$, for a given tolerance ε.

In the practice, $\|\mathrm{B}\|$ can be estimated as follows: since

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

a lower bound of $\|\mathrm{B}\|$ is provided by $c = \delta_{k+1}/\delta_k$, where $\delta_{j+1} = \|\mathbf{x}^{(j+1)} - \mathbf{x}^{(j)}\|$, with $j = k-1, k$. Replacing $\|\mathrm{B}\|$ by c, the right-hand side of (4.72) suggests using the following indicator for $\|\mathbf{e}^{(k+1)}\|$

$$\epsilon^{(k+1)} = \frac{\delta_{k+1}^2}{\delta_k - \delta_{k+1}}. \tag{4.73}$$

Due to the kind of approximation of $\|\mathrm{B}\|$ that has been used, the reader is warned that $\epsilon^{(k+1)}$ should not be regarded as an upper bound for $\|\mathbf{e}^{(k+1)}\|$. However, often $\epsilon^{(k+1)}$ provides a reasonable indication about the true error behavior, as we can see in the following example.

Example 4.12 Consider the linear system A$\mathbf{x}$=$\mathbf{b}$ with

$$\mathrm{A} = \begin{bmatrix} 4 & 1 & 1 \\ 2 & -9 & 0 \\ 0 & -8 & -6 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 6 \\ -7 \\ -14 \end{bmatrix},$$

which admits the unit vector as exact solution. Let us apply the Jacobi method and estimate the error at each step by using (4.73). Figure 4.11 shows an acceptable agreement between the behavior of the error $\|\mathbf{e}^{(k+1)}\|_\infty$ and that of its estimate $\epsilon^{(k+1)}$. •

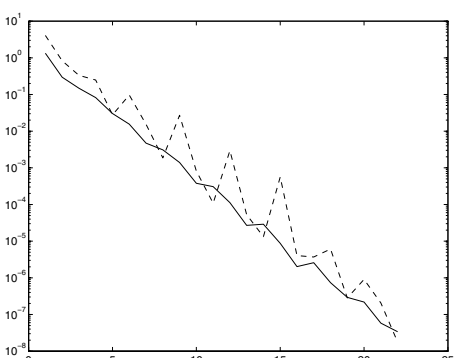

FIGURE 4.11. Absolute error (in solid line) versus the error estimated by (4.73) (dashed line). The number of iterations is indicated on the x-axis

4.6.2 A Stopping Test Based on the Residual

A different stopping criterion consists of continuing the iteration until $\|\mathbf{r}^{(k)}\| \leq \varepsilon$, ε being a fixed tolerance. Note that

$$\|\mathbf{x} - \mathbf{x}^{(k)}\| = \|\mathrm{A}^{-1}\mathbf{b} - \mathbf{x}^{(k)}\| = \|\mathrm{A}^{-1}\mathbf{r}^{(k)}\| \leq \|\mathrm{A}^{-1}\|\, \varepsilon.$$

Considering instead a normalized residual, i.e. stopping the iteration as soon as $\|\mathbf{r}^{(k)}\|/\|\mathbf{b}\| \leq \varepsilon$, we obtain the following control on the relative error

$$\frac{\|\mathbf{x} - \mathbf{x}^{(k)}\|}{\|\mathbf{x}\|} \leq \frac{\|\mathrm{A}^{-1}\|\,\|\mathbf{r}^{(k)}\|}{\|\mathbf{x}\|} \leq K(\mathrm{A})\|\frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{b}\|} \leq \varepsilon K(\mathrm{A}).$$

In the case of preconditioned methods, the residual is replaced by the preconditioned residual, so that the previous criterion becomes

$$\frac{\|\mathrm{P}^{-1}\mathbf{r}^{(k)}\|}{\|\mathrm{P}^{-1}\mathbf{r}^{(0)}\|} \leq \varepsilon,$$

where P is the preconditioning matrix.

4.7 Applications

In this section we consider two examples arising in electrical network analysis and structural mechanics which lead to the solution of large sparse linear systems.

4.7.1 Analysis of an Electric Network

We consider a purely resistive electric network (shown in Figure 4.12, left) which consists of a connection of n stages S (Figure 4.12, right) through the series resistances R. The circuit is completed by the driving current generator I_0 and the load resistance R_L. As an example, a purely resistive network is a model of a signal attenuator for low-frequency applications where capacitive and inductive effects can be neglected. The connecting points between the electrical components will be referred to henceforth as *nodes* and are progressively labeled as drawn in the figure. For $n \geq 1$, the total number of nodes is $4n$. Each node is associated with a value of the electric potential V_i, $i = 0, \ldots, 4n$, which are the unknowns of the problem.

The *nodal analysis* method is employed to solve the problem. Precisely, the *Kirchhoff current law* is written at any node of the network leading to the linear system $\tilde{\mathrm{Y}}\tilde{\mathbf{V}} = \tilde{\mathbf{I}}$, where $\tilde{\mathbf{V}} \in \mathbb{R}^{N+1}$ is the vector of nodal potentials, $\tilde{\mathbf{I}} \in \mathbb{R}^{N+1}$ is the load vector and the entries of the matrix $\tilde{\mathrm{Y}} \in$

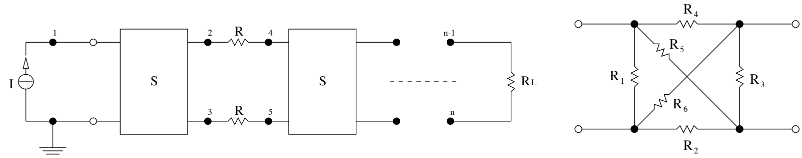

FIGURE 4.12. Resistive electric network (left) and resistive stage S (right)

$\mathbb{R}^{(N+1)\times(N+1)}$, for $i, j = 0, \ldots, 4n$, are given by

$$\tilde{Y}_{ij} = \begin{cases} \displaystyle\sum_{j \in k(i)} G_{ij}, & \text{for } i = j, \\ -G_{ij}, & \text{for } i \neq j, \end{cases}$$

where $k(i)$ is the index set of the neighboring nodes of node i and $G_{ij} = 1/R_{ij}$ is the admittance between node i and node j, provided R_{ij} denotes the resistance between the two nodes i and j. Since the potential is defined up to an additive constant, we arbitrarily set $V_0 = 0$ (*ground potential*). As a consequence, the number of independent nodes for potential difference computations is $N = 4n - 1$ and the linear system to be solved becomes $\mathrm{Y}\mathbf{V} = \mathbf{I}$, where $\mathrm{Y} \in \mathbb{R}^{N\times N}$, $\mathbf{V} \in \mathbb{R}^N$ and $\mathbf{I} \in \mathbb{R}^N$ are obtained eliminating the first row and column in $\tilde{\mathrm{Y}}$ and the first entry in $\tilde{\mathbf{V}}$ and $\tilde{\mathbf{I}}$, respectively. The matrix Y is symmetric, diagonally dominant and positive definite. This last property follows by noting that

$$\tilde{\mathbf{V}}^T \tilde{\mathrm{Y}} \tilde{\mathbf{V}} = \sum_{i=1}^{N} G_{ii} V_i^2 + \sum_{i,j=1}^{N} G_{ij}(V_i - V_j)^2,$$

which is always a positive quantity, being equal to zero only if $\tilde{\mathbf{V}} = \mathbf{0}$. The sparsity pattern of Y in the case $n = 3$ is shown in Figure 4.13 (left) while the spectral condition number of Y as a function of the number of blocks n is reported in Figure 4.13 (right). Our numerical computations have been carried out setting the resistance values equal to 1 Ω while $I_0 = 1\,A$.

In Figure 4.14 we report the convergence history of several non preconditioned iterative methods in the case $n = 5$ corresponding to a matrix size of 19×19. The plots show the Euclidean norms of the residual normalized to the initial residual. The dashed curve refers to the Gauss-Seidel method, the dash-dotted line refers to the gradient method, while the solid and circled lines refer respectively to the conjugate gradient (CG) and SOR method (with an optimal value of the relaxation parameter $\omega \simeq 1.76$ computed according to (4.19) since Y is block tridiagonal symmetric positive definite). The SOR method converges in 109 iterations, while the CG method converges in 10 iterations.

We have also considered the solution of the system at hand by the conjugate gradient (CG) method using the Cholesky version of the ILU(0) and

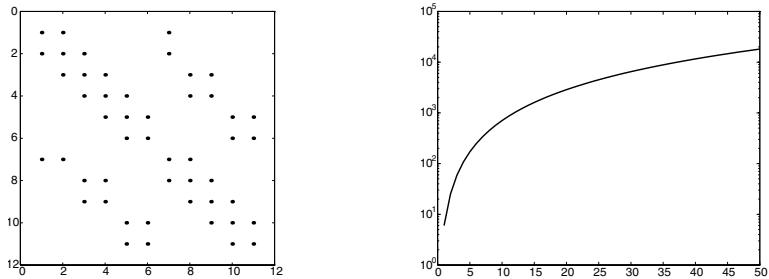

FIGURE 4.13. Sparsity pattern of Y for $n = 3$ (left) and spectral condition number of Y as a function of n (right)

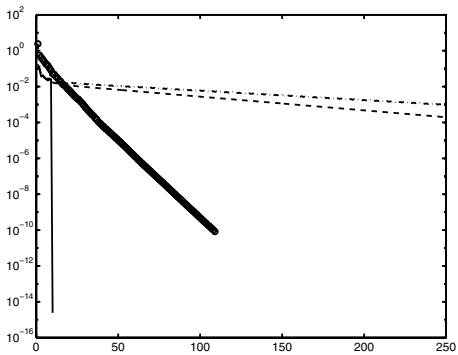

FIGURE 4.14. Convergence history of several non preconditioned iterative methods

MILU(0) preconditioners, where drop tolerances equal to $\varepsilon = 10^{-2}, 10^{-3}$ have been chosen for the MILU(0) preconditioner (see Section 4.3.2). Calculations with both preconditioners have been done using the MATLAB functions `cholinc` and `michol`. Table 4.2 shows the convergence iterations of the method for $n = 5, 10, 20, 40, 80, 160$ and for the considered values of ε. We report in the second column the number of nonzero entries in the Cholesky factor of matrix Y, in the third column the number of iterations for the CG method without preconditioning to converge, while the columns ICh(0) and MICh(0) with $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-3}$ show the same information for the CG method using the incomplete Cholesky and modified incomplete Cholesky preconditioners, respectively.
The entries in the table are the number of iterations to converge and the number in the brackets are the nonzero entries of the L-factor of the corresponding preconditioners. Notice the decrease of the iterations as ε decreases, as expected. Notice also the increase of the number of iterations with respect to the increase of the size of the problem.

n	nz	CG	ICh(0)	MICh(0) $\varepsilon = 10^{-2}$	MICh(0) $\varepsilon = 10^{-3}$
5	114	10	9 (54)	6 (78)	4 (98)
10	429	20	15 (114)	7 (173)	5 (233)
20	1659	40	23 (234)	10 (363)	6 (503)
40	6519	80	36 (474)	14 (743)	7 (1043)
80	25839	160	62 (954)	21 (1503)	10 (2123)
160	102879	320	110 (1914)	34 (3023)	14 (4283)

TABLE 4.2. Convergence iterations for the preconditioned CG method

4.7.2 Finite Difference Analysis of Beam Bending

Consider the beam clamped at the endpoints that is drawn in Figure 4.15 (left). The structure, of length L, is subject to a distributed load P, varying along the free coordinate x and expressed in $[Kgm^{-1}]$. We assume henceforth that the beam has uniform rectangular section, of width r and depth s, momentum of inertia $J = rs^3/12$ and Young's module E, expressed in $[m^4]$ and $[Kg\,m^{-2}]$, respectively.

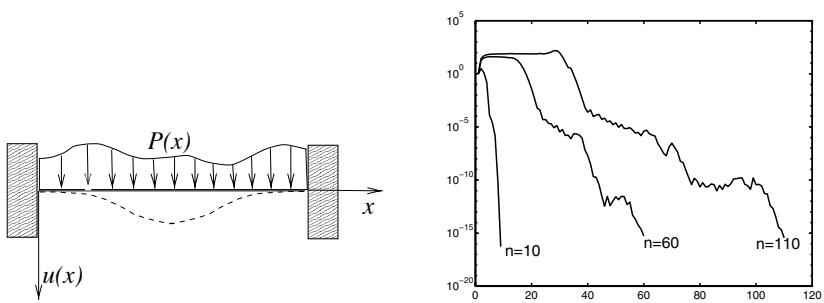

FIGURE 4.15. Clamped beam (left); convergence histories for the preconditioned conjugate gradient method in the solution of system (4.76) (right)

The transverse bending of the beam, under the assumption of small displacements, is governed by the following fourth-order differential equation

$$(EJu'')''(x) = P(x), \qquad 0 < x < L, \tag{4.74}$$

where $u = u(x)$ denotes the vertical displacement. The following boundary conditions (at the endpoints $x = 0$ and $x = L$)

$$u(0) = u(L) = 0, \qquad u'(0) = u'(L) = 0, \tag{4.75}$$

model the effect of the two clampings (vanishing displacements and rotations). To solve numerically the boundary-value problem (4.74)-(4.75), we use the finite difference method (see Section 10.10.1 and Exercise 11 of Chapter 12).

With this aim, let us introduce the discretization nodes $x_j = jh$, with $h = L/N_h$ and $j = 0, \ldots, N_h$, and substitute at each node x_j the fourth-order derivative with an approximation through centered finite differences. Letting $f(x) = P(x)/(EJ)$, $f_j = f(x_j)$ and denoting by η_j the (approximate) *nodal displacement* of the beam at node x_j, the finite difference discretization of (4.74)-(4.75) is

$$\begin{cases} \eta_{j-2} - 4\eta_{j-1} + 6\eta_j - 4\eta_{j+1} + \eta_{j+2} = h^4 f_j, \ \forall j = 2, \ldots, N_h - 2, \\ \eta_0 = \eta_1 = \eta_{N_h-1} = \eta_{N_h} = 0. \end{cases} \tag{4.76}$$

The null displacement boundary conditions in (4.76) that have been imposed at the first and the last two nodes of the grid, require that $N_h \geq 4$. Notice that a fourth-order scheme has been used to approximate the fourth-order derivative, while, for sake of simplicity, a first-order approximation has been employed to deal with the boundary conditions (see Section 10.10.1).

The $N_h - 3$ discrete equations (4.76) yield a linear system of the form $\mathrm{A}\mathbf{x} = \mathbf{b}$ where the unknown vector $\mathbf{x} \in \mathbb{R}^{N_h-3}$ and the load vector $\mathbf{b} \in \mathbb{R}^{N_h-3}$ are given respectively by $\mathbf{x} = (\eta_2, \eta_3, \ldots, \eta_{N_h-2})^T$ and $\mathbf{b} = (f_2, f_3, \ldots, f_{N_h-2})^T$, while the coefficient matrix $\mathrm{A} \in \mathbb{R}^{(N_h-3)\times(N_h-3)}$ is pentadiagonal and symmetric, given by $\mathrm{A} = \mathrm{pentadiag}_{N_h-3}(1, -4, 6, -4, 1)$.

The matrix A is symmetric and positive definite. Therefore, to solve system $\mathrm{A}\mathbf{x} = \mathbf{b}$, the SSOR preconditioned conjugated gradient method (see Section 4.21) and the Cholesky factorization method have been employed. In the remainder of the section, the two methods are identified by the symbols (CG) and (CH).

The convergence histories of CG are reported in Figure 4.15 (right), where the sequences $\|\mathbf{r}^{(k)}\|_2/\|\mathbf{b}^{(k)}\|_2$, for the values $n = 10, 60, 110$, are plotted, $\mathbf{r}^{(k)} = \mathbf{b} - \mathrm{A}\mathbf{x}^{(k)}$ being the residual at the k-th step. The results have been obtained using Program 20, with `toll`$=10^{-15}$ and $\omega = 1.8$ in (4.22). The initial vector $\mathbf{x}^{(0)}$ has been set equal to the null vector.

As a comment to the graphs, it is worth noting that CG has required 7, 33 and 64 iterations to converge, respectively, with a maximum absolute error of $5 \cdot 10^{-15}$ with respect to the solution produced by CH. This latter has an overall computational cost of 136, 1286 and 2436 flops respectively, to be compared with the corresponding 3117, 149424 and 541647 flops of method CG. As for the performances of the SSOR preconditioner, we remark that the spectral condition number of matrix A is equal to 192, $3.8 \cdot 10^5$ and $4.5 \cdot 10^6$, respectively, while the corresponding values in the preconditioned case are 65, $1.2 \cdot 10^4$ and $1.3 \cdot 10^5$.

4.8 Exercises

1. The spectral radius of the matrix

$$\mathrm{B} = \begin{bmatrix} a & 4 \\ 0 & a \end{bmatrix}$$

is $\rho(\mathrm{B}) = a$. Check that if $0 < a < 1$, then $\rho(\mathrm{B}) < 1$, while $\|\mathrm{B}^m\|_2^{1/m}$ can be greater than 1.

2. Let $\mathrm{A} \in \mathbb{R}^{n\times n}$ be a strictly diagonally dominant matrix by rows. Show that the Gauss-Seidel method for the solution of the linear system (3.2) is convergent.

3. Check that the matrix $\mathrm{A} = \mathrm{tridiag}(-1, \alpha, -1)$, with $\alpha \in \mathbb{R}$, has eigenvalues given by

$$\lambda_j = \alpha - 2\cos(j\theta), \quad j = 1, \ldots, n$$

where $\theta = \pi/(n+1)$ and the corresponding eigenvectors are

$$\mathbf{q}_j = [\sin(j\theta),\ \sin(2j\theta), \ldots, \sin(nj\theta)]^T.$$

Under which conditions on α is the matrix positive definite?
[*Solution* : $\alpha \geq 2$.]

4. Consider the pentadiagonal matrix $\mathrm{A} = \mathrm{pentadiag}_n(-1, -1, 10, -1, -1)$. Assume $n = 10$ and $\mathrm{A} = \mathrm{M} + \mathrm{N} + \mathrm{D}$, with $\mathrm{D} = \mathrm{diag}(8, \ldots, 8) \in \mathbb{R}^{10\times 10}$, $\mathrm{M} = \mathrm{pentadiag}_{10}(-1, -1, 1, 0, 0)$ and $\mathrm{N} = \mathrm{M}^T$. To solve $\mathrm{A}\mathbf{x} = \mathbf{b}$, analyze the convergence of the following iterative methods

$$(a) \quad (\mathrm{M} + \mathrm{D})\mathbf{x}^{(k+1)} = -\mathrm{N}\mathbf{x}^{(k)} + \mathbf{b},$$

$$(b) \quad \mathrm{D}\mathbf{x}^{(k+1)} = -(\mathrm{M} + \mathrm{N})\mathbf{x}^{(k)} + \mathbf{b},$$

$$(c) \quad (\mathrm{M} + \mathrm{N})\mathbf{x}^{(k+1)} = -\mathrm{D}\mathbf{x}^{(k)} + \mathbf{b}.$$

[*Solution* : denoting respectively by ρ_a, ρ_b and ρ_c the spectral radii of the iteration matrices of the three methods, we have $\rho_a = 0.1450$, $\rho_b = 0.5$ and $\rho_c = 12.2870$ which implies convergence for methods (a) and (b) and divergence for method (c).]

5. For the solution of the linear system $\mathrm{A}\mathbf{x} = \mathbf{b}$ with

$$\mathrm{A} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ 5 \end{bmatrix},$$

consider the following iterative method

$$\mathbf{x}^{(k+1)} = \mathrm{B}(\theta)\mathbf{x}^{(k)} + \mathbf{g}(\theta), \qquad k \geq 0, \qquad \text{with } \mathbf{x}^{(0)} \text{ given},$$

where θ is a real parameter and

$$\mathrm{B}(\theta) = \frac{1}{4}\begin{bmatrix} 2\theta^2 + 2\theta + 1 & -2\theta^2 + 2\theta + 1 \\ -2\theta^2 + 2\theta + 1 & 2\theta^2 + 2\theta + 1 \end{bmatrix}, \quad \mathbf{g}(\theta) = \begin{bmatrix} \frac{1}{2} - \theta \\ \frac{1}{2} - \theta \end{bmatrix}.$$

Check that the method is consistent $\forall\,\theta \in \mathbb{R}$. Then, determine the values of θ for which the method is convergent and compute the optimal value of θ (i.e., the value of the parameter for which the convergence rate is maximum).

[*Solution* : the method is convergent iff $-1 < \theta < 1/2$ and the convergence rate is maximum if $\theta = (1 - \sqrt{3})/2$.]

6. To solve the following block linear system

$$\begin{bmatrix} \mathrm{A}_1 & \mathrm{B} \\ \mathrm{B} & \mathrm{A}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix},$$

consider the two methods

$$(1) \quad \mathrm{A}_1\mathbf{x}^{(k+1)} + \mathrm{B}\mathbf{y}^{(k)} = \mathbf{b}_1, \quad \mathrm{B}\mathbf{x}^{(k)} + \mathrm{A}_2\mathbf{y}^{(k+1)} = \mathbf{b}_2;$$

$$(2) \quad \mathrm{A}_1\mathbf{x}^{(k+1)} + \mathrm{B}\mathbf{y}^{(k)} = \mathbf{b}_1, \quad \mathrm{B}\mathbf{x}^{(k+1)} + \mathrm{A}_2\mathbf{y}^{(k+1)} = \mathbf{b}_2.$$

Find sufficient conditions in order for the two schemes to be convergent for any choice of the initial data $\mathbf{x}^{(0)}$, $\mathbf{y}^{(0)}$.

[*Solution* : method (1) is a decoupled system in the unknowns $\mathbf{x}^{(k+1)}$ and $\mathbf{y}^{(k+1)}$. Assuming that A_1 and A_2 are invertible, method (1) converges if $\rho(\mathrm{A}_1^{-1}\mathrm{B}) < 1$ and $\rho(\mathrm{A}_2^{-1}\mathrm{B}) < 1$. In the case of method (2) we have a coupled system to solve at each step in the unknowns $\mathbf{x}^{(k+1)}$ and $\mathbf{y}^{(k+1)}$. Solving formally the first equation with respect to $\mathbf{x}^{(k+1)}$ (which requires A_1 to be invertible) and substituting into the second one we see that method (2) is convergent if $\rho(\mathrm{A}_2^{-1}\mathrm{B}\mathrm{A}_1^{-1}\mathrm{B}) < 1$ (again A_2 must be invertible).]

7. Consider the linear system $\mathrm{A}\mathbf{x} = \mathbf{b}$ with

$$\mathrm{A} = \begin{bmatrix} 62 & 24 & 1 & 8 & 15 \\ 23 & 50 & 7 & 14 & 16 \\ 4 & 6 & 58 & 20 & 22 \\ 10 & 12 & 19 & 66 & 3 \\ 11 & 18 & 25 & 2 & 54 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 110 \\ 110 \\ 110 \\ 110 \\ 110 \end{bmatrix}.$$

(1) Check if the Jacobi and Gauss-Seidel methods can be applied to solve the system. (2) Check if the stationary Richardson method with optimal parameter can be applied with $\mathrm{P} = \mathrm{I}$ and $\mathrm{P} = \mathrm{D}$, where D is the diagonal part of A, and compute the corresponding values of α_{opt} and ρ_{opt}.

[*Solution* : (1): matrix A is neither diagonally dominant nor symmetric positive definite, so that we must compute the spectral radii of the iteration matrices of the Jacobi and Gauss-Seidel methods to verify if they are convergent. It turns out that $\rho_J = 0.9280$ and $\rho_{GS} = 0.3066$ which implies convergence for both methods. (2): in the case $\mathrm{P} = \mathrm{I}$ all the eigenvalues of A are positive so that the Richardson method can be applied yielding $\alpha_{opt} = 0.015$ and $\rho_{opt} = 0.6452$. If $\mathrm{P} = \mathrm{D}$ the method is still applicable and $\alpha_{opt} = 0.8510$, $\rho_{opt} = 0.6407$.]

8. Consider the linear system $\mathrm{A}\mathbf{x} = \mathbf{b}$ with

$$\mathrm{A} = \begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 23 \\ 32 \\ 33 \\ 31 \end{bmatrix}.$$

Analyze the convergence properties of the Jacobi and Gauss-Seidel methods applied to the system above in their point and block forms (for a 2×2 block partition of A).

[*Solution* : both methods are convergent, the block form being the faster one. Moreover, $\rho^2(\mathrm{B}_J) = \rho(\mathrm{B}_{GS})$.]

9. To solve the linear system $\mathrm{A}\mathbf{x} = \mathbf{b}$, consider the iterative method (4.6), with $\mathrm{P} = \mathrm{D} + \omega\mathrm{F}$ and $\mathrm{N} = -\beta\mathrm{F} - \mathrm{E}$, ω and β being real numbers. Check that the method is consistent only if $\beta = 1 - \omega$. In such a case, express the eigenvalues of the iteration matrix as a function of ω and determine for which values of ω the method is convergent, as well as the value of ω_{opt}, assuming that $\mathrm{A} = \mathrm{tridiag}_{10}(-1, 2, -1)$.

   [*Hint* : Take advantage of the result in Exercise 3.]

10. Let $\mathrm{A} \in \mathbb{R}^{n\times n}$ be such that $\mathrm{A} = (1+\omega)\mathrm{P} - (\mathrm{N}+\omega\mathrm{P})$, with $\mathrm{P}^{-1}\mathrm{N}$ nonsingular and with real eigenvalues $1 > \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Find the values of $\omega \in \mathbb{R}$ for which the following iterative method

$$(1+\omega)\mathrm{P}\mathbf{x}^{(k+1)} = (\mathrm{N}+\omega\mathrm{P})\mathbf{x}^{(k)} + \mathbf{b}, \qquad k \geq 0,$$

    converges $\forall \mathbf{x}^{(0)}$ to the solution of the linear system (3.2). Determine also the value of ω for which the convergence rate is maximum.

    [*Solution* : $\omega > -(1+\lambda_n)/2$; $\omega_{opt} = -(\lambda_1 + \lambda_n)/2$.]

11. Consider the linear system

$$\mathrm{A}\mathbf{x} = \mathbf{b} \quad \text{with } \mathrm{A} = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ -8 \end{bmatrix}.$$

    Write the associated functional $\Phi(\mathbf{x})$ and give a graphical interpretation of the solution of the linear system. Perform some iterations of the gradient method, after proving convergence for it.

12. Check that in the gradient method $\mathbf{x}^{(k+2)}$ is not an optimal direction with respect to $\mathbf{r}^{(k)}$.

13. Show that the coefficients α_k and β_k in the conjugate gradient method can be written in the alternative form (4.45).

    [*Solution*: notice $\mathrm{A}\mathbf{p}^{(k)} = (\mathbf{r}^{(k)} - \mathbf{r}^{(k+1)})/\alpha_k$ and thus $(\mathrm{A}\mathbf{p}^{(k)})^T\mathbf{r}^{(k+1)} = -\|\mathbf{r}^{(k+1)}\|_2^2/\alpha_k$. Moreover, $\alpha_k(\mathrm{A}\mathbf{p}^{(k)})^T\mathbf{p}^{(k)} = -\|\mathbf{r}^{(k)}\|_2^2$.]

14. Prove the three-terms recursive relation (4.46) for the residual in the conjugate gradient method.

    [*Solution*: subtract from both sides of $\mathrm{A}\mathbf{p}^{(k)} = (\mathbf{r}^{(k)} - \mathbf{r}^{(k+1)})/\alpha_k$ the quantity $\beta_{k-1}/\alpha_k\mathbf{r}^{(k)}$ and recall that $\mathrm{A}\mathbf{p}^{(k)} = \mathrm{A}\mathbf{r}^{(k)} - \beta_{k-1}\mathrm{A}\mathbf{p}^{(k-1)}$. Then, expressing the residual $\mathbf{r}^{(k)}$ as a function of $\mathbf{r}^{(k-1)}$ one immediately gets the desired relation.]