Chapter 4

Linear systems

4.1 Introduction

Our focus in this chapter is on *linear* systems of equations of the form

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{4.1}$$

where $A = (a_{ij})$ is an $n \times n$ matrix, $\boldsymbol{b} = (b_j)$ is a vector in $\mathbb{R}^n$ and $\boldsymbol{x} = (x_j)$, also a vector in $\mathbb{R}^n$, is the unknown solution. The expression (4.1) is equivalent to the system of n linear equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \ i = 1, \dots, n.$$

We shall assume throughout this chapter that the matrix A is *invertible* (or *nonsingular*), i.e. that the following equivalent conditions hold:

- $\exists$ an $n \times n$ matrix A^{-1} such that $A^{-1}A = AA^{-1} = I$, where I is the identity matrix.
- $\det A \neq 0$.
- $\operatorname{rank}(A) = n$, i.e. A has n linearly independent columns (equivalently, rows);
- $\ker(A) = \{\mathbf{0}\}$.
- The eigenvalues of A are all non-zero.

If A is invertible then the unique solution of (4.1) is given by $\boldsymbol{x} = A^{-1}\boldsymbol{b}$. General analytical formulas for A^{-1} in terms of determinants do exist (Cramer's rule). But they are completely unsuitable for numerical computation. In practice one applies methods for solving (4.1) which do not involve the explicit computation of A^{-1}. These methods split into two types: *direct* methods, which (in exact arithmetic) provide the solution $\boldsymbol{x}$ in a finite number of operations, and *iterative* methods, which provide a sequence of approximate solutions $\boldsymbol{x}^k$ which (under appropriate assumptions on A) converge to the solution $\boldsymbol{x}$ as the iteration count $k \to \infty$.

The solution of systems of *linear* equations is a very important special case of the solution of general nonlinear systems considered in the previous chapter. As we saw there, many nonlinear iterative solvers (in particular the Newton method for systems) require at least one linear solve to be carried out at each iteration. As we shall see later, systems of linear equations are also central to the numerical solution of differential equations. They also arise naturally in many mathematical models of real-world problems, as the following example illustrates.

4.1.1 Example and motivation

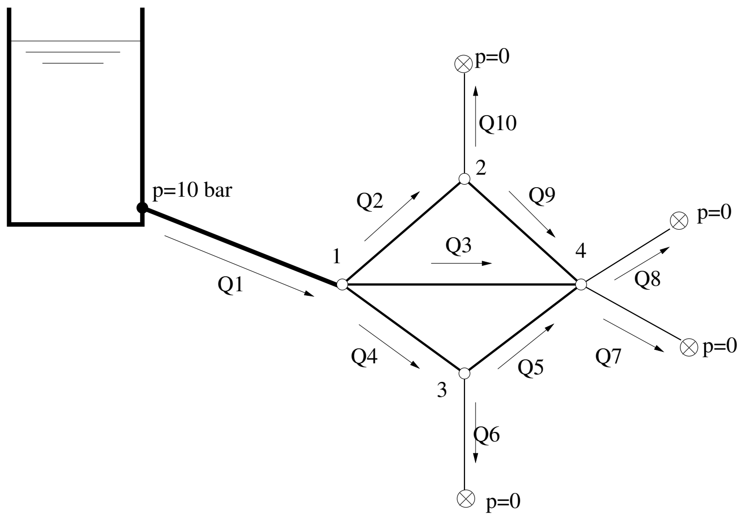


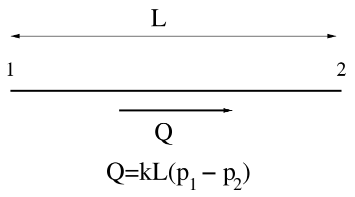
Figure 4.1: *Example: hydraulic network*

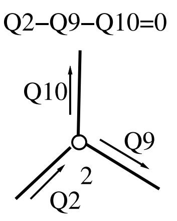
Example 4.1.1 (Hydraulic network)**.** *We consider a hydraulic network (illustrated in Figure 4.1) formed by* 10 *pipes. The network draws water from a reservoir at constant pressure* $p_r = 10$ bar *(all pressures in the example are given as the difference between the actual pressure and the atmospheric pressure measured in bar). In every pipe the following relation between the flow rate* Q $(\mathrm{m}^3/\mathrm{s})$ *and the pressure at the two ends of the pipe* p_1 *and* p_2 *holds:*

$$Q = kL(p_1 - p_2). \tag{4.2}$$

Here k *is the hydraulic resistance* $(\frac{\mathrm{m}^2}{\mathrm{bar\,s}})$ *and* L *is the length of the pipe in meters.*

At every node of the network we can enforce the balance of flows. For example, at node 2 *in the figure below, one has* $Q_2 - Q_{10} - Q_9 = 0$ *(giving a negative sign to exiting flow.)*





Finally, in the exit pipe we assume that the outflow pressure coincides with the atmospheric pressure ($p = 0$ bar).

We wish to find the distribution of pressures and flow rates in the network. Using the two given relations it is straightforward to set up a linear system for the pressure on the form

$$A\boldsymbol{p} = \boldsymbol{b}, \tag{4.3}$$

where $\boldsymbol{p} = [p_1, p_2, p_3, p_4]^T$ is the vector of nodal pressures of the network. Given the following characteristics of the pipes

Pipe	k	L	Pipe	k	L	Pipe	k	L
1	*0.01*	*20*	*2*	*0.005*	*10*	*3*	*0.005*	*712*
4	*0.005*	*10*	*5*	*0.005*	*10*	*6*	*0.002*	*8*
7	*0.002*	*8*	*8*	*0.002*	*8*	*9*	*0.005*	*10*
10	*0.002*	*8*						

we may compute the matrix A and vector $\boldsymbol{b}$ as

$$A = \begin{bmatrix} -0.360 & 0.050 & 0.050 & 0.060 \\ 0.050 & -0.116 & 0.000 & 0.050 \\ 0.050 & 0.000 & -0.116 & 0.050 \\ 0.060 & 0.050 & 0.050 & -0.192 \end{bmatrix}, \qquad b = \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The pressure vector is given by the solution of the linear system, i.e.

$$\boldsymbol{p} = A^{-1}\boldsymbol{b} = [8.147, 5.943, 5.943, 5.641]^T.$$

Finally, the flow rates may be computed using the relation (4.2):

Q1	*0.371*	*Q2*	*0.110*	*Q3*	*0.150*
Q4	*0.110*	*Q5*	*0.015*	*Q6*	*0.095*
Q7	*0.090*	*Q8*	*0.090*	*Q9*	*0.015*
Q10	*0.095*				

In practical applications a hydraulic network may consist of hundreds of pipes which leads to the need to solve large linear systems.

4.2 Matrix norms

The space of all real $m \times n$ matrices is a real vector space denoted by $\mathbb{R}^{m\times n}$. A matrix norm on $\mathbb{R}^{m\times n}$ is defined in accordance with the general notion of norm on a real vector space as in Definition 2.1.1.

Definition 4.2.1 (Matrix norm)**.** *By a* matrix norm *we mean a norm on the vector space $\mathbb{R}^{m\times n}$. According to Definition 2.1.1 a function $\|\cdot\| : \mathbb{R}^{m\times n} \to \mathbb{R}$ is a matrix norm if*

1. *(i) $\|A\| \geq 0$, $\forall A \in \mathbb{R}^{m\times n}$ and (ii) $\|A\| = 0$ if and only if $A = 0$;*
2. *$\|\alpha A\| = |\alpha|\|A\|$ $\forall \alpha \in \mathbb{R}$ and $\forall A \in \mathbb{R}^{m\times n}$;*
3. *$\|A + B\| \leq \|A\| + \|B\|$, $\forall A, B \in \mathbb{R}^{m\times n}$.*

As for vectors, we say that a sequence of matrices $A^{(k)}$, $k = 0, 1, 2, \ldots$, converges to a matrix A, and write $\lim_{k\to\infty} A^{(k)} = A$, if $\lim_{k\to\infty} \|A^{(k)} - A\|_M = 0$ for some matrix norm $\|\cdot\|_M$. Note that since all norms on $\mathbb{R}^{m\times n}$ are equivalent, it doesn't matter which matrix norm we take.

One well-known example of a matrix norm is the *Frobenius norm*

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} \qquad \text{for } A \in \mathbb{R}^{m\times n},$$

which is simply the $p = 2$ norm considered above, in the setting $V = \mathbb{R}^{m\times n}$, i.e. we are viewing the matrix as an element of Euclidean space.

But we are going to be more concerned with a different class of matrix norms, namely those defined by reference to the action of the linear map associated with matrix multiplication. For simplicity we focus on the case of square matrices $A \in \mathbb{R}^{n\times n}$, for which the linear map $T_A \boldsymbol{x} = A\boldsymbol{x}$ maps $\mathbb{R}^n$ to $\mathbb{R}^n$. Then, given a vector norm $\|\cdot\|_V$ on $\mathbb{R}^n$, we can define an *induced* (or *subordinate*) matrix norm on $\mathbb{R}^{n\times n}$ by[1]

$$\|A\|_M = \sup_{\boldsymbol{x}\in\mathbb{R}^n\setminus\{\boldsymbol{0}\}} \frac{\|A\boldsymbol{x}\|_V}{\|\boldsymbol{x}\|_V}, \qquad \text{for } A \in \mathbb{R}^{n\times n}. \tag{4.4}$$

[1]For those familiar with the theory of bounded linear operators, the subordinate matrix norm $\|A\|_M$ is nothing other than the "operator norm" of the bounded linear operator $T_A : \mathbb{R}^n \to \mathbb{R}^n$, $T_A\boldsymbol{x} = A\boldsymbol{x}$.

Lemma 4.2.2. *Let $\|\cdot\|_M$ be the matrix norm (4.4) induced by a vector norm $\|\cdot\|_V$. Then*

- $\|\cdot\|_M$ *is a norm on* $\mathbb{R}^{n\times n}$*;*
- $\|A\boldsymbol{x}\|_V \leq \|A\|_M\|\boldsymbol{x}\|_V$*,* $A \in \mathbb{R}^{n\times n}$*,* $\boldsymbol{x} \in \mathbb{R}^n$ *(compatibility);*
- $\|I\|_M = 1$*;*
- $\|AB\|_M \leq \|A\|_M\|B\|_M$*, for all* $A, B \in \mathbb{R}^{n\times n}$*;*

Proof. The proof is left as an exercise. □

For later use we note that the matrix norm $\|\cdot\|_\infty$ on $\mathbb{R}^{n\times n}$ induced by the supremum norm $\|\cdot\|_\infty$ on $\mathbb{R}^n$ has the following explicit form

$$\|A\|_\infty = \max_{i\in\{1,\ldots,n\}} \sum_{j=1}^{n} |a_{ij}|, \qquad A \in \mathbb{R}^{n\times n}. \tag{4.5}$$

Powers of a matrix. We will frequently encounter powers of a matrix in the analysis of iterative methods for linear systems. For a matrix $A \in \mathbb{R}^{n\times n}$, its k-th power A^k is the product of A with itself k times, e.g. $A^2 = AA$, $A^3 = AAA$, etc. By convention $A^0 = I$ the $n \times n$ identity matrix and $A^1 = A$. The convergence of a sequence of matrices is understood as above, i.e. $A^k \to B$ as $k \to \infty$ if and only if $\|A^k - B\|_M \to 0$ for some matrix norm $\|\cdot\|_M$. We leave it as an exercise to show using Lemma 4.2.2 that

$$\|A^k\|_M \leq \left(\|A\|_M\right)^k \quad \forall A \in \mathbb{R}^{n\times n}, \forall k \in \mathbb{N}, \tag{4.6}$$

for any matrix norm $\|\cdot\|_M$ that is induced by a vector norm $\|\cdot\|_V$.

4.3 Eigenvalues and spectrum of a matrix

We will need the above norms in order to quantify the effect of perturbations on the solutions of linear systems obtained using different numerical methods. Two concepts will be particularly important in our analysis: the *spectral radius* of a square matrix, and its *condition number*. Both properties are related to the eigenvalues of the matrix. We denote the set of eigenvalues of a matrix $A \in \mathbb{R}^{n\times n}$ by

$$\sigma(A) = \{\lambda \in \mathbb{C} : A\boldsymbol{v} = \lambda\boldsymbol{v} \text{ for some } \boldsymbol{v} \in \mathbb{C}^n \setminus \{0\}\}.$$

Complex eigenvectors and relation to real invariant spaces. In general, the eigenvectors of a real matrix are complex vectors. However, we can always relate a complex eigenvector to an invariant space of real vectors as follows[2]. Writing $\boldsymbol{v} = \boldsymbol{x} + i\boldsymbol{y}$ for a nonzero (complex) eigenvector $\boldsymbol{v}$, with $\boldsymbol{x}$ and $\boldsymbol{y}$ its real and imaginary parts respectively, and writing $\lambda = |\lambda|(\cos\theta + i\sin\theta)$ for some $\theta \in [0, 2\pi)$, we find that

$$A\boldsymbol{x} + iA\boldsymbol{y} = A\boldsymbol{v} = \lambda\boldsymbol{v} = |\lambda|(\cos\theta\,\boldsymbol{x} - \sin\theta\,\boldsymbol{y}) + i|\lambda|(\sin\theta\,\boldsymbol{x} + \cos\theta\,\boldsymbol{y}).$$

Since $A\boldsymbol{x}$ and $A\boldsymbol{y}$ are both real vectors, identifying real and imaginary parts yields

$$\begin{aligned} A\boldsymbol{x} &= |\lambda|(\cos\theta\,\boldsymbol{x} - \sin\theta\,\boldsymbol{y}), \\ A\boldsymbol{y} &= |\lambda|(\sin\theta\,\boldsymbol{x} + \cos\theta\,\boldsymbol{y}). \end{aligned} \tag{4.7}$$

This shows that the space $W = \operatorname{span}\{\boldsymbol{x}, \boldsymbol{y}\}$ spanned by the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is an invariant space of A (*Exercise!*). Since the eigenvector $\boldsymbol{v}$ is nonzero, the dimension of W is either 1 or 2.

Using the above invariant subspace, the following technical lemma shows that we can find always real vectors that have similar properties to eigenvectors.

Lemma 4.3.1. *Let $\|\cdot\|_V$ be a vector norm on $\mathbb{R}^n$. Let $A \in \mathbb{R}^{n\times n}$ and let $\lambda \in \sigma(A)$. Then there exists a nonzero real vector $\hat{\boldsymbol{v}} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}$ and an $\epsilon > 0$ such that*

$$|\lambda| \leq \frac{\|A\hat{\boldsymbol{v}}\|_V}{\|\hat{\boldsymbol{v}}\|_V}, \qquad \epsilon|\lambda|^k \leq \frac{\|A^k\hat{\boldsymbol{v}}\|_V}{\|\hat{\boldsymbol{v}}\|_V} \quad \forall k \in \mathbb{N}. \tag{4.8}$$

Proof. Skip this on first reading! Not examinable. If there exists a real eigenvector $\boldsymbol{v} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}$ then the result is immediate since then λ must be real because A is real, and we can simply take $\hat{\boldsymbol{v}} = \boldsymbol{v}$, and we get $A^k\hat{\boldsymbol{v}} = \lambda^k\hat{\boldsymbol{v}}$ for all $k \in \mathbb{N}$, from which it is easily seen that (4.8) holds with $\epsilon = 1$.

It remains only to consider the case where there is no real eigenvector for the eigenvalue λ. Let $\boldsymbol{v} = \boldsymbol{x} + i\boldsymbol{y}$ be a nonzero complex eigenvector, with $\boldsymbol{x}$ and $\boldsymbol{y}$ its real and imaginary parts respectively. Under the assumption that there is no real eigenvector, we must have $\boldsymbol{y} \neq 0$ and also $\boldsymbol{x}$ and $\boldsymbol{y}$ must be linearly independent *(Exercise! Hint: if $\boldsymbol{x}$ and $\boldsymbol{y}$ are linearly dependent i.e. $\boldsymbol{x} = \alpha\boldsymbol{y}$ for some $\alpha \in \mathbb{R}$, then $\boldsymbol{y}$ would be a real eigenvector, contradicting the assumption that there is no real eigenvector).*

Using an induction on (4.7), we find that, for all $k \in \mathbb{N}$,

$$\begin{aligned} A^k\boldsymbol{x} &= |\lambda|^k\big(\cos(k\theta)\,\boldsymbol{x} - \sin(k\theta)\,\boldsymbol{y}\big), \\ A^k\boldsymbol{y} &= |\lambda|^k\big(\sin(k\theta)\,\boldsymbol{x} + \cos(k\theta)\,\boldsymbol{y}\big). \end{aligned} \tag{4.9}$$

[2]In Linear Algebra, we say that a subspace W of $\mathbb{R}^n$ is an invariant space of A if $A\boldsymbol{w} \in W$ for all $\boldsymbol{w} \in W$.

We now define

$$\hat{\boldsymbol{v}} = \cos\omega\,\boldsymbol{x} - \sin\omega\,\boldsymbol{y}, \tag{4.10}$$

where $\omega \in \mathbb{R}$ is chosen as a maximizer of $\|A\hat{\boldsymbol{v}}\|_V$, i.e. we choose ω such that

$$\|\cos(\theta+\omega)\boldsymbol{x} - \sin(\theta+\omega)\boldsymbol{y}\|_V = \max_{\phi\in\mathbb{R}} \|\cos\phi\,\boldsymbol{x} - \sin\phi\,\boldsymbol{y}\|_V,$$

where we note that a maximizer always exists by 2π-periodicity of trigonometric functions and by continuity of the norm.

First we claim that $\hat{\boldsymbol{v}}$ is nonzero for any choice of $\omega \in \mathbb{R}$. To show this, we claim more generally that there exists a $\delta > 0$ such that

$$\inf_{\phi\in\mathbb{R}} \|\cos\phi\,\boldsymbol{x} - \sin\phi\,\boldsymbol{y}\|_V \geq \delta. \tag{4.11}$$

Indeed, if this were false, then by continuity of the norm and 2π-periodicity of trigonometric functions, there would exist a $\phi_* \in [0, 2\pi]$ such that $\|\cos(\phi_*)\boldsymbol{x} - \sin(\phi_*)\boldsymbol{y}\|_V = 0$ or equivalently $\cos(\phi_*)\boldsymbol{x} = \sin(\phi_*)\boldsymbol{y}$ which would imply that $\boldsymbol{x}$ and $\boldsymbol{y}$ are linearly dependent, and thus a contradiction.

For the choice of ω above, we get

$$\frac{\|A\hat{\boldsymbol{v}}\|_V}{\|\hat{\boldsymbol{v}}\|_V} = |\lambda| \underbrace{\frac{\|\cos(\theta+\omega)\,\boldsymbol{x} - \sin(\theta+\omega)\,\boldsymbol{y}\|_V}{\|\cos\omega\,\boldsymbol{x} - \sin\omega\,\boldsymbol{y}\|_V}}_{\geq 1} \geq |\lambda|.$$

thus showing the first inequality in (4.8).

To obtain the second inequality in (4.8), we use (4.9) and (4.10) to find that, for all $k \in \mathbb{N}$,

$$A^k\hat{\boldsymbol{v}} = |\lambda|^k\big(\cos(k\theta+\omega)\,\boldsymbol{x} - \sin(k\theta+\omega)\,\boldsymbol{y}\big).$$

Then using the lower bound (4.11) we see that

$$\frac{\|A^k\hat{\boldsymbol{v}}\|_V}{\|\hat{\boldsymbol{v}}\|_V} = |\lambda|^k \frac{\|\cos(k\theta+\omega)\,\boldsymbol{x} - \sin(k\theta+\omega)\,\boldsymbol{y}\|_V}{\|\hat{\boldsymbol{v}}\|_V} \geq \frac{|\lambda|^k\delta}{\|\boldsymbol{x}\|_V + \|\boldsymbol{y}\|_V} \quad \forall k \in \mathbb{N},$$

where we have used the triangle inequality $\|\hat{\boldsymbol{v}}\|_V \leq \|\boldsymbol{x}\|_V + \|\boldsymbol{y}\|_V$ in the denominator. So the second inequality in (4.8) holds with $\epsilon = \delta\left(\|\boldsymbol{x}\|_V + \|\boldsymbol{y}\|_V\right)^{-1}$. □

4.4 Spectral radius and condition number

Spectral radius. Using the spectrum $\sigma(A)$ we may define the spectral radius of A.

Definition 4.4.1 (Spectral radius of a matrix)**.** *The* spectral radius $\rho(A)$ *of a matrix A is defined to be the maximum modulus of the eigenvalues of A, i.e.*

$$\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|.$$

It should be noted that the spectral radius is a matrix "semi-norm", but *not* a matrix norm, in the sense that it satisfies all the properties of Definition 4.2.1 except for property 1(*ii*). It can, however, be viewed in a certain sense as a surrogate for a matrix norm, because of the following lemma, which we state without proof.

Lemma 4.4.2. (Relationship between spectral radius and induced matrix norms.)

- *Let $\|\cdot\|_M$ be the matrix norm on $\mathbb{R}^{n\times n}$ induced by some vector norm $\|\cdot\|_V$ on $\mathbb{R}^n$. Then*
$$\rho(A) \le \|A\|_M \quad \forall A \in \mathbb{R}^{n\times n}. \tag{4.12}$$
- *Given $A \in \mathbb{R}^{n\times n}$, for every $\epsilon > 0$ there exists a matrix norm $\|\cdot\|_{M,A,\epsilon}$ induced by some vector norm $\|\cdot\|_{V,A,\epsilon}$ (both depending on A and ϵ) such that*
$$\|A\|_{M,A,\epsilon} \le \rho(A) + \epsilon. \tag{4.13}$$

The proof of Lemma 4.4.2 is beyond the scope of this course. However the inequality in (4.12) is easily deduced from Lemma 4.3.1, since if $\lambda \in \sigma(A)$ is an eigenvalue such that $|\lambda| = \rho(A)$, then Lemma 4.3.1 shows that there exists a nonzero real vector $\hat{\boldsymbol{v}} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ such that

$$\rho(A) = |\lambda| \le \frac{\|A\hat{\boldsymbol{v}}\|_V}{\|\hat{\boldsymbol{v}}\|_V} \le \sup_{\boldsymbol{v}\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \frac{\|A\boldsymbol{v}\|_V}{\|\boldsymbol{v}\|_V} = \|A\|_M.$$

Lemma 4.4.2 thus gives an alternative characterisation of the spectral radius as

$$\rho(A) = \inf_{\|\cdot\|_M} \|A\|_M,$$

where the infimum is taken over the set of all matrix norms induced by some vector norm on $\mathbb{R}^n$.

For symmetric matrices and the Euclidean norm the situation is simpler.

Lemma 4.4.3. *If $A \in \mathbb{R}^{n\times n}$ is symmetric then the matrix norm $\|\cdot\|_2$ induced by the Euclidean norm $\|\cdot\|_2$ on $\mathbb{R}^n$ satisfies*

$$\|A\|_2 = \rho(A).$$

One can also relate the spectral radius to convergence of matrix powers. This will be important when we consider iterative methods for solving linear systems. In the following lemma, A^k

means the product of A with itself k times, e.g. $A^2 = AA$, $A^3 = AAA$ etc., and the convergence of a sequence of matrices is understood as in the previous section, i.e. $\lim_{k\to\infty} A^k = 0$ means that $\lim_{k\to\infty} \|A^k\|_M = 0$ for some, and hence every, matrix norm $\|\cdot\|_M$.

Lemma 4.4.4. *Let $A \in \mathbb{R}^{n\times n}$. Then*

$$\lim_{k\to\infty} A^k = 0 \Leftrightarrow \rho(A) < 1. \tag{4.14}$$

Proof. For the forward implication, assume that $\lim_{k\to\infty} A^k = 0$, i.e. $\|A^k\| \to 0$ as $k \to \infty$ for any norm $\|\cdot\|$ on $\mathbb{R}^{n\times n}$. Let $\|\cdot\|_V$ be any norm on $\mathbb{R}^n$, and let $\|\cdot\|_M$ be the induced matrix norm. Then, since $\rho(A)^k = \rho(A^k)$ for all $k \in \mathbb{N}$ *(exercise!)*, Lemma 4.4.2 gives

$$\rho(A)^k = \rho(A^k) \leq \|A^k\|_M \to 0 \text{ as } k \to \infty.$$

This implies $\rho(A) < 1$. For the reverse implication, assume that $\rho(A) < 1$. Then there exists $\epsilon > 0$ such that $\rho(A) < 1-\epsilon$. (For example one can take $\epsilon = (1-\rho(A))/2$.) By the second part of Lemma 4.4.2 there exists an induced matrix norm $\|\cdot\|_M$ such that $\|A\|_M \leq \rho(A) + \epsilon < 1$. But then using (4.6) we see that $\|A^k\|_M \leq (\|A\|_M)^k \to 0$ as $k \to \infty$, which proves that $\lim_{k\to\infty} A^k = 0$. □

We now turn to the definition of condition number.

Definition 4.4.5 (Condition number of a matrix)**.** *Let $\|\cdot\|_M$ be the matrix norm induced by a vector norm $\|\cdot\|_V$, as in (4.4). The* condition number *$K_M(A)$ of a matrix $A \in \mathbb{R}^{n\times n}$ with respect to $\|\cdot\|_M$ is defined by*

$$K_M(A) = \|A\|_M \|A^{-1}\|_M.$$

Observe that $K_M(A) \geq 1$ since $1 = \|I\|_M = \|AA^{-1}\|_M \leq \|A\|_M\|A^{-1}\|_M = K_M(A)$ and that $K_M(A^{-1}) = K_M(A)$. Note that the definition of the condition number requires the matrix to be invertible. For singular matrices the convention is that $K_M(A) = \infty$. One may show that $K_M(A)^{-1}$ measures the distance (in the $\|\cdot\|_M$ norm) from A to the closest singular matrix. If the p-norm is used in the definition of the condition number one may use the notation $K_p(A)$.

In the special case $p = 2$ there exist convenient characterizations of the condition number. For instance, suppose that A is symmetric positive definite (SPD), meaning that A is symmetric and $\boldsymbol{v}^T A\boldsymbol{v} > 0$ for all $\mathbf{0} \neq \boldsymbol{v} \in \mathbb{R}^n$. An equivalent characterisation of SPD is that A is symmetric and has only positive eigenvalues, i.e. $\sigma(A) \subset (0,\infty)$. Then

$$K_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\lambda_{\max}}{\lambda_{\min}}, \tag{4.15}$$

where $0 < \lambda_{\min} \leq \lambda_{\max}$ are the smallest and largest eigenvalues of A.

As we will see, the condition number of a matrix A provides an indication of how easy it is to numerically solve the linear system

$$A\boldsymbol{x} = \boldsymbol{b} \tag{4.16}$$

for $\boldsymbol{x} \in \mathbb{R}^n$, given $\boldsymbol{b} \in \mathbb{R}^n$. Matrices with $K_M(A) \approx 1$ are said to be *well-conditioned*, and those with $K_M(A) \gg 1$ are said to be *ill-conditioned*.

One particular situation in which the condition number plays a key role is in estimating the accuracy of a numerical solution by computing the residual. When solving the linear system (4.16) numerically, whichever method we use we will in general obtain a perturbed solution $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\delta x}$ satisfying

$$A(\boldsymbol{x} + \boldsymbol{\delta x}) = \boldsymbol{b} + \delta\mathbf{b},$$

for some perturbations $\boldsymbol{\delta x}$ and $\boldsymbol{\delta b}$. We would like to understand how the accuracy of the numerical solution (i.e. the size of $\delta\boldsymbol{x}$) relates to conditioning. Let's attempt to bound the size of the error $\boldsymbol{\delta x}$. Note that, by linearity, $\boldsymbol{\delta x} = A^{-1}\boldsymbol{\delta b}$. Then, dropping subscripts $_V$ and $_M$ on norms and condition numbers (to avoid clutter), since

$$\|\boldsymbol{\delta x}\| = \|A^{-1}\boldsymbol{\delta b}\| \leq \|A^{-1}\|\|\boldsymbol{\delta b}\|$$

and

$$\|\boldsymbol{b}\| = \|A\boldsymbol{x}\| \leq \|A\|\|\boldsymbol{x}\|,$$

we have that

$$\frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} \leq \|A^{-1}\|\|A\|\frac{\|\boldsymbol{\delta b}\|}{\|\boldsymbol{b}\|} = K(A)\frac{\|\boldsymbol{\delta b}\|}{\|\boldsymbol{b}\|}.$$

Similarly we can get a lower bound on the error by observing that

$$\|\boldsymbol{\delta b}\| = \|A\boldsymbol{\delta x}\| \leq \|A\|\|\boldsymbol{\delta x}\|$$

and

$$\|\boldsymbol{x}\| = \|A^{-1}\boldsymbol{b}\| \leq \|A^{-1}\|\|\boldsymbol{b}\|,$$

which imply that

$$\frac{1}{K(A)}\frac{\|\boldsymbol{\delta b}\|}{\|\boldsymbol{b}\|} \leq \frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|}.$$

Combining these estimates we find that

$$\frac{1}{K(A)}\frac{\|A\tilde{\boldsymbol{x}} - \boldsymbol{b}\|}{\|\boldsymbol{b}\|} \leq \frac{\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq K(A)\frac{\|A\tilde{\boldsymbol{x}} - \boldsymbol{b}\|}{\|\boldsymbol{b}\|}. \tag{4.17}$$

Hence if $K(A)$ is moderate, then the relative solution error $\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \frac{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|}$ is guaranteed to be of a similar order to the relative residual $\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} = \frac{\|A\tilde{\boldsymbol{x}}-\boldsymbol{b}\|}{\|\boldsymbol{b}\|}$. However, if $K(A)$ is large, then the relative residual will not necessarily give a good estimate of the relative solution error.

4.5 Direct solvers

In certain special cases it is very easy to write down a direct method for solving (4.1). The simplest case is when A is a *diagonal* matrix, i.e.

$$a_{ij} = 0 \qquad \forall i \neq j.$$

Then the inverse of A is also a diagonal matrix, with diagonal entries $a_{ii}^{-1} = (a_{ii})^{-1}$ (note that $a_{ii} \neq 0$ for any i else A would not be invertible), and the solution vector has components

$$x_i = \frac{1}{a_{ii}} b_i, \quad i = 1, \ldots, n.$$

In this case the solution can be computed using exactly n floating point operations (flops).

Two other important special cases are when A is either *upper triangular*, meaning that

$$a_{ij} = 0 \quad \forall i, j \; : \; 1 \leq j < i \leq n,$$

or *lower triangular*, meaning that

$$a_{ij} = 0 \quad \forall i, j \; : \; 1 \leq i < j \leq n.$$

Upper triangular systems can be solved using the *backward substitution algorithm*:

$$x_n = b_n / a_{nn},$$

and for $i = n-1, n-2, \ldots, 2, 1$

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j \right).$$

In the lower triangular case, the analogous *forward substitution algorithm* applies:

$$x_1 = b_1 / a_{11},$$

and for $i = 2, 3, \ldots, n$

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j \right).$$

(Note that in both cases we know $a_{ii} \neq 0$ for all $i = 1, \ldots, n$, because A is invertible and $\det A = \prod a_{ii}$ for a triangular matrix.)

The number of divisions and multiplications involved in these algorithms is of order $n(n+1)/2$ and the number of additions and subtractions is of order $n(n-1)/2$. Hence the algorithms both require $O(n^2)$ floating point operations (flops).

The best-known direct solution method for *general* invertible linear systems is *Gaussian elimination* (GE). This algorithm involves moving row-by-row down the system, subtracting appropriate multiples of the current row from the rows below, so as to arrive at an upper triangular system which can be solved using the backward substitution algorithm above. In fact the operations that transform the system into upper triangular form correspond to multiplying the original matrix A by the inverse of an appropriate lower triangular matrix L. Hence GE is often referred to as "LU factorization", because given a matrix A the algorithm outputs lower and upper triangular matrices L and U such that $A = LU$.

There are many subtleties involved in implementing and analysing the GE algorithm (e.g. "pivoting") that we shall not go into in these lectures. The main thing we note is that the computational cost of the algorithm scales like $O(n^3)$ as the size of the matrix tends to infinity. This means that, while stable implementations (as in Matlab's "backslash" command) are effective for relatively small systems, for very large systems GE is impractical.[3] For such systems one is forced to turn to iterative methods, some examples of which we shall study in the next section. These are often much cheaper than GE, provided that one can obtain an accurate solution with a relatively small number of iterations.

Another fact one should remember about direct methods like GE is that, while in theory the algorithm produces the exact solution $\boldsymbol{x}$, this only holds if one works in infinite precision arithmetic. In practical computations using finite precision arithmetic the result of the direct solver will not be $\boldsymbol{x}$, but rather some approximation $\tilde{\boldsymbol{x}}$. Recalling (4.17), we have

$$\frac{\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|} \leq K(A)\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|},$$

where the *residual* is defined by $\boldsymbol{r} = \boldsymbol{b} - A\tilde{\boldsymbol{x}}$, and the condition number $K(A) = \|A\|\|A^{-1}\|$. (Recall from (4.15) that for a symmetric positive definite matrix the condition number with respect to the Euclidean norm $\|\cdot\|_2$ is equal to the quotient of the largest and smallest eigenvalues.) Typically $\boldsymbol{r} \neq \boldsymbol{0}$ due to rounding errors, and then if $K(A)$ is large, the error can be big even for a small residual $\boldsymbol{r}$. It follows that the larger the condition number of the matrix, the poorer the results of the direct solver.

Here we find another advantage of iterative methods: as we shall see shortly, they provide a natural framework in which to "precondition" the system (4.1) so as to mitigate the detrimental effect of any ill-conditioning in A. An illustration of the power of preconditioning is given in Example 4.5.2 below.

[3] We note however that once the expensive $O(n^3)$ factorization $A = LU$ has been computed, solving the system $A\boldsymbol{x} = \tilde{\boldsymbol{b}}$ for any other right-hand side $\tilde{\boldsymbol{b}}$ is relatively cheap, as it can be evaluated using a combination of the forward and backward substitution algorithms, at $O(n^2)$ cost. This is useful if we intend to solve the linear system for multiple different right-hand sides $\boldsymbol{b}$.

Example 4.5.1. *Consider the system $A\boldsymbol{u} = \boldsymbol{b}$ of size $n \times n$, where*

$$A = \begin{pmatrix} \frac{2}{h} & -\frac{1}{h} & 0 & \cdots & \cdots & 0 \\ -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & & \vdots \\ 0 & -\frac{1}{h} & \ddots & \ddots & & \\ \vdots & & \ddots & & -\frac{1}{h} & \vdots \\ \vdots & & & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} \\ 0 & \cdots & \cdots & 0 & -\frac{1}{h} & \frac{2}{h} \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} h \\ h \\ \vdots \\ \\ \vdots \\ h \\ h \end{pmatrix}, \qquad h = \frac{1}{n+1}. \tag{4.18}$$

The solution of this system represents an approximation to the deformation under a uniform load of an elastic string of length one, fixed at $x = 0$ and $x = 1$.

Using Matlab we construct the matrix and compute its condition number (with respect to $\|\cdot\|_2$) for a range of values of n. The resulting graph is presented in Figure 4.2.

```
>> for i=1:10
    n=5*i; h=1/(n+1);
    A = (2/h)*diag(ones(n,1)) - (1/h)*diag(ones(n-1,1),1) ...
        - (1/h)*diag(ones(n-1,1),-1)
    K(i) = cond(A)
   end
>> plot([5:5:50],K)
```

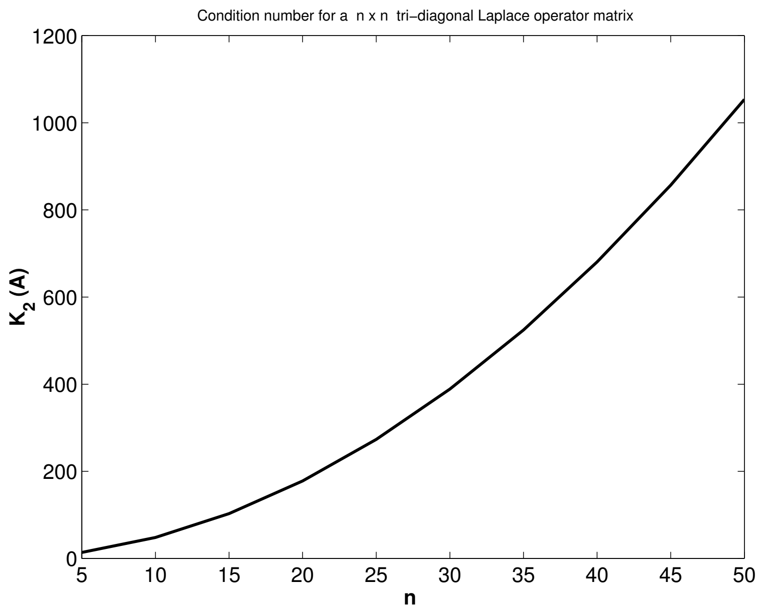


Figure 4.2: *Condition number of the matrix A from Example 4.5.1 as a function of the size n.*

Example 4.5.2. *The Hilbert matrix of size $n \times n$ is a symmetric matrix defined by*

$$A_{ij} = \frac{1}{i+j-1}, \qquad i,j = 1,\ldots,n.$$

This is a classic example of a very ill-conditioned matrix. In Matlab we may construct a Hilbert matrix of size n using the command ***hilb(n)***.

We construct linear systems $A\boldsymbol{x} = \boldsymbol{b}$ of size $n = 4, 6, 8, 10, 12, 14, \ldots$ where A is a Hilbert matrix, the exact solution $\boldsymbol{x}_{\text{ex}}$ is the unit vector and the right hand side is chosen as $\boldsymbol{b} = A\boldsymbol{x}_{\text{ex}}$. For every n, we compute the condition number of the matrix (with respect to $\|\cdot\|_2$), solve the system using LU factorization, and compare the obtained solution $\boldsymbol{x}_{\text{lu}}$ with the exact solution.

```
>> for j=2:7;
     n=2*j; i=j-1; nn(i)=n;
     A=hilb(n);
     x_ex=ones(n,1); b=A*x_ex;
     Acond(i)=cond(A);
     x=A\b;
     error_lu(i)=norm(x-x_ex)/norm(x_ex);
     end
>> semilogy(nn,Acond,'-',nn,error_lu,'--')
```

Figure 4.3 shows the condition number as well as the relative error $\|\boldsymbol{x}_{\text{ex}} - \boldsymbol{x}_{\text{lu}}\|/\|\boldsymbol{x}_{\text{ex}}\|$. In all the calculations, $\|\cdot\| = \|\cdot\|_2$ is the Euclidean norm $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T \cdot \boldsymbol{x}}$. We also show the error in a solution $\boldsymbol{x}_{\text{gr}}$ obtained using a preconditioned iterative method. Note that the error for the direct method increases with the condition number, whereas that for the iterative method remains constant.

4.6 Iterative methods for linear systems

As in the chapter on nonlinear equations, the idea behind an iterative method for the linear system $A\boldsymbol{x} = \boldsymbol{b}$ is to construct a sequence of vectors $\boldsymbol{x}^k$ that converge to the solution $\boldsymbol{x}$ as $k \to \infty$, i.e.

$$\lim_{k\to\infty} \boldsymbol{x}^k = \boldsymbol{x}.$$

The classical setting is where the iterates satisfy a linear recurrence relation

$$\boldsymbol{x}^{k+1} = B\boldsymbol{x}^k + \boldsymbol{c}, \qquad k = 0, 1, 2, \ldots, \tag{4.19}$$

where B is called the *iteration matrix* (depending on A) and $\boldsymbol{c}$ is a vector (depending on $\boldsymbol{b}$). If B and $\boldsymbol{c}$ stay the same at each iteration then the iterative method (4.19) is said to be

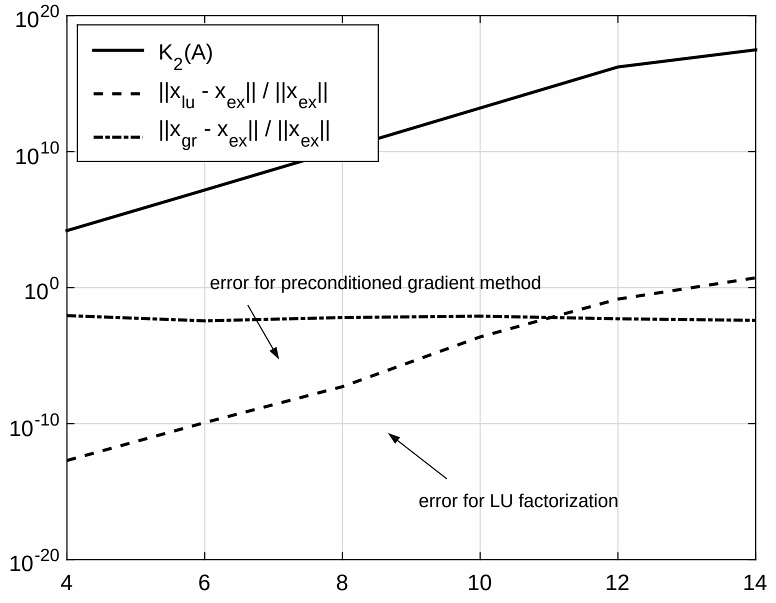


Figure 4.3: *Solution of a linear system based on the Hilbert matrix: relative error obtained using LU-factorization and using a preconditioned iterative method. The horizontal axis shows n, the size of the system.*

stationary. If B and $\boldsymbol{c}$ are such that the exact solution $\boldsymbol{x}$ of $A\boldsymbol{x} = \boldsymbol{b}$ satisfies

$$\boldsymbol{x} = B\boldsymbol{x} + \boldsymbol{c} \tag{4.20}$$

then the method is said to be *consistent*. We note that a consistent method is defined by its iteration matrix B alone: using the relation $\boldsymbol{x} = A^{-1}\boldsymbol{b}$, we can determine the vector $\boldsymbol{c}$ uniquely as $\boldsymbol{c} = (I - B)A^{-1}\boldsymbol{b}$. Note also that consistent stationary methods are nothing more than fixed point formulations (in the sense of §3.7.1) of the problem $\boldsymbol{f}(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$, in which the fixed point function $\boldsymbol{g}(\boldsymbol{x})$ (which should satisfy $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x})$) is a *linear* function of $\boldsymbol{x}$, i.e. $\boldsymbol{g}(\boldsymbol{x}) = B\boldsymbol{x} + \boldsymbol{c}$. The iteration (4.19) is precisely the "simultaneous iteration" (or "fixed point iteration") of Definition 3.7.4.

For a consistent stationary method the error $\boldsymbol{e}^k = \boldsymbol{x} - \boldsymbol{x}^k$ at iteration k satisfies the recurrence relation

$$\boldsymbol{e}^{k+1} = B\boldsymbol{e}^k, \quad k = 0, 1, \ldots, \tag{4.21}$$

which implies that

$$\boldsymbol{e}^k = B^k\boldsymbol{e}^0, \quad k = 0, 1, \ldots, \tag{4.22}$$

where $\boldsymbol{e}^0 = \boldsymbol{x} - \boldsymbol{x}^0$ is the initial error. We would like to determine conditions under which the method converges, i.e. $\boldsymbol{e}^k \to \boldsymbol{0}$ as $k \to \infty$. The key parameter is the spectral radius $\rho(B) = \max_{\lambda \in \sigma(B)} |\lambda|$, where $\sigma(B)$ is the set of eigenvalues of B.

Theorem 4.6.1. *If the method (4.19) is consistent (in the sense of (4.20)) then* $\lim_{k\to\infty} \boldsymbol{e}^k = \boldsymbol{0}$ *for all* $\boldsymbol{e}^0$ *(and therefore* $\lim_{k\to\infty}\boldsymbol{x}^k = \boldsymbol{x}$ *for all* $\boldsymbol{x}^0$*) if and only if* $\rho(B) < 1$.

Proof. If $\rho(B) < 1$ then by Lemma 4.4.4 we have that $\lim_{k\to\infty} B^k = 0$. This means that for any $\boldsymbol{e}^0$ and any induced matrix norm, $\|\boldsymbol{e}^k\|_V = \|B^k\boldsymbol{e}^0\|_V \leq \|B^k\|_M\|\boldsymbol{e}^0\|_V \to 0$ as $k \to \infty$, i.e. the method converges. To show the converse implication, we claim that if $\rho(B) \geq 1$ then there is an initial guess for which the iteration is not convergent. Indeed, Lemma 4.3.1 shows that there exists a vector $\hat{\boldsymbol{v}} \in \mathbb{R}^n\setminus\{\boldsymbol{0}\}$ and an $\epsilon > 0$ such that $\|B^k\hat{\boldsymbol{v}}\|_V \geq \epsilon\rho(B)^k\|\hat{\boldsymbol{v}}\|_V \geq \epsilon\|\hat{\boldsymbol{v}}\|_V > 0$ for all $k \in \mathbb{N}$. Choosing $\boldsymbol{e}^0 = \hat{\boldsymbol{v}}$ then gives $\boldsymbol{e}^k = B^k\hat{\boldsymbol{v}}$ for all $k \in \mathbb{N}$, for which it does not hold that $\boldsymbol{e}^k \to 0$ as $k \to \infty$. □

The connection between the spectral radius and the rate of convergence is slightly more subtle.

Theorem 4.6.2. *If* $\|\cdot\|_V$ *is a vector norm inducing a matrix norm* $\|\cdot\|_M$ *for which* $\|B\|_M < 1$*, then the method converges linearly w.r.t.* $\|\cdot\|_V$ *with constant* $C = \|B\|_M$*, and*

$$\|\boldsymbol{e}^k\|_V \leq (\|B\|_M)^k\|\boldsymbol{e}^0\|_V, \quad k = 0, 1, \dots. \tag{4.23}$$

Proof. Follows easily from (4.21) and (4.22). □

For symmetric matrices we have the following corollary, due to Lemma 4.4.3. (See also Theorem 4.8.1 below.)

Corollary 4.6.3. *If B is a symmetric matrix and* $\rho(B) < 1$ *then the method converges linearly w.r.t.* $\|\cdot\|_2$ *with constant* $C = \rho(B)$*, and*

$$\|\boldsymbol{e}^k\|_2 \leq (\rho(B))^k\|\boldsymbol{e}^0\|_2, \quad k = 0, 1, \dots. \tag{4.24}$$

For non-symmetric matrices, the spectral radius will not in general be a convergence constant in the $\|\cdot\|_2$ norm. But by Lemma 4.4.2 we do at least know that if $\rho(B) < 1$ then for every $\epsilon \in (0, 1-\rho(B))$ there exists *some* norm with respect to which the method converges linearly with convergence constant $\rho(B) + \epsilon$.

4.7 Basic stationary method and preconditioning

The simplest choice of iteration matrix and vector is $B = I - A$ and $\boldsymbol{c} = \boldsymbol{b}$, leading to the *basic stationary method*

$$\boldsymbol{x}^{k+1} = (I - A)\boldsymbol{x}^k + \boldsymbol{b}, \quad k = 0, 1, \dots. \tag{4.25}$$

By Theorem 4.6.1 this method converges for all initial guesses $\boldsymbol{x}_0$ if and only if $\rho(I-A)<1$. Since $\sigma(I-A)=1-\sigma(A)$, this is equivalent to requiring that all eigenvalues of A must lie in the unit ball centred at the point 1 in the complex plane. This condition is rather restrictive and for general matrices the basic stationary method (4.25) will not converge. Fortunately there are many ways of obtaining methods which converge for other classes of matrices.

Notice that the iteration (4.25) is obtained from the original equation $A\boldsymbol{x}=\boldsymbol{b}$ by splitting the term $A\boldsymbol{x}$ on the left-hand side into the sum $A\boldsymbol{x}=\boldsymbol{x}-(\boldsymbol{x}-A\boldsymbol{x})$, moving the second term to the right-hand side, and then replacing $\boldsymbol{x}$ by $\boldsymbol{x}^{k+1}$ on the left-hand side and by $\boldsymbol{x}^k$ on the right-hand side. This corresponds to a splitting of the matrix A into the sum $A=I-(I-A)$. Suppose that instead we split A as $A=P-N$ for some invertible matrix P, with $N=P-A$. Then following the same procedure we arrive at the iteration

$$P\boldsymbol{x}^{k+1}=N\boldsymbol{x}^k+\boldsymbol{b}, \quad k=0,1,\dots, \tag{4.26}$$

which can also be written in increment form

$$P(\boldsymbol{x}^{k+1}-\boldsymbol{x}^k)=\boldsymbol{r}^k, \quad k=0,1,\dots, \tag{4.27}$$

where $\boldsymbol{r}^k=\boldsymbol{b}-A\boldsymbol{x}^k$ is the residual. Multiplying (4.26) by P^{-1} gives the equivalent form

$$\boldsymbol{x}^{k+1}=(I-P^{-1}A)\boldsymbol{x}^k+P^{-1}\boldsymbol{b}, \quad k=0,1,\dots, \tag{4.28}$$

which reveals that (4.26) is a consistent stationary method of the form (4.19) with $B=I-P^{-1}A$ and $\boldsymbol{c}=P^{-1}\boldsymbol{b}$. (But note that in practical implementations one usually works with (4.26) or (4.27) rather than (4.19), to avoid explicit computation of P^{-1}). Note that one can also derive (4.28) (and hence (4.26)) by applying the basic stationary method (4.25) to the modified system

$$P^{-1}A\boldsymbol{x}=P^{-1}\boldsymbol{b}. \tag{4.29}$$

The matrix P is called the *preconditioner* of A. By (4.28) and Theorem 4.6.1 the method (4.26) converges for all initial guesses $\boldsymbol{x}_0$ if and only if $\rho(I-P^{-1}A)<1$. Given a matrix A, one would like to choose P in such a way as to make $\rho(I-P^{-1}A)$ as small as possible (in particular, smaller than 1), to guarantee fast convergence. But one also wants the linear system (4.26) to be easy to solve for $\boldsymbol{x}^{k+1}$, so that each iteration step is cheap to evaluate. These are two competing requirements, and the extreme cases are:

- $P=I$; (4.26) is trivial to solve but there is no preconditioning effect.
- $P=A$; perfect preconditioning, in the sense that the iteration would converge in one step, but computing this step would be as difficult as solving the original system.

The key to the art of preconditioning is to find a balance between the two competing requirements of effectiveness and easy application. The right choice of P depends strongly on the type of matrix A one is trying to precondition.

4.8 Stationary Richardson method

The simplest preconditioner is obtained by taking $P = \frac{1}{\alpha} I$ for some $0 \neq \alpha \in \mathbb{R}$, sometimes called a *relaxation* or *acceleration* parameter. This corresponds to a simple rescaling of the system to $\alpha A\boldsymbol{x} = \alpha\boldsymbol{b}$, and an iteration matrix $B_\alpha = I - \alpha A$. (The case $\alpha = 1$ is just the basic stationary method.) Since $\sigma(I - \alpha A) = 1 - \alpha\sigma(A)$, the method converges if and only if all eigenvalues of A lie in the ball of radius $1/|\alpha|$ centred at the point $1/\alpha$ in the complex plane. This method is sometimes called a stationary Richardson method.

The following theorem summarises the convergence theory for symmetric positive definite (SPD) matrices.

Theorem 4.8.1. *Let A be a symmetric positive definite matrix and let $0 < \lambda_{\min} \leq \lambda_{\max}$ denote the smallest and largest eigenvalues of A. Then the stationary Richardson method is convergent if and only if $0 < \alpha < 2/\lambda_{\max}$. Furthermore, when the method converges it does so linearly with*

$$\|\boldsymbol{e}^{k+1}\|_2 \leq \rho(B_\alpha)\|\boldsymbol{e}^k\|_2, \quad k = 0, 1, \ldots. \tag{4.30}$$

The optimal choice of $0 < \alpha < 2/\lambda_{\max}$ minimising the convergence constant $\rho(B_\alpha)$ is given by

$$\alpha_* = \frac{2}{\lambda_{\max} + \lambda_{\min}}, \qquad \text{for which} \quad \rho(B_{\alpha_*}) = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{K_2(A) - 1}{K_2(A) + 1}. \tag{4.31}$$

Proof. The fact that the method is convergent if and only if $0 < \alpha < 2/\lambda_{\max}$ follows from our discussion above. The bound (4.30) follows directly from (4.24). That (4.31) gives the optimal convergence constant holds because for $0 < \alpha < 2/\lambda_{\max}$ the eigenvalues of B_α all lie between the extremal eigenvalues $1 - \alpha\lambda_{\max}$ and $1 - \alpha\lambda_{\min}$, which satisfy $-1 < 1 - \alpha\lambda_{\max} \leq 1 - \alpha\lambda_{\min} < 1$. The spectral radius $\rho(B_\alpha) = \max\{|1 - \alpha\lambda_{\max}|, |1 - \alpha\lambda_{\min}|\}$ is minimised when $1 - \alpha\lambda_{\max} = -(1 - \alpha\lambda_{\min})$, which provides the claimed optimal values α_* and $\rho(B_{\alpha_*})$. The degenerate case where $\lambda_{\min} = \lambda_{\max} = \lambda$ corresponds to $A = \lambda I$, in which case the optimal choice $\alpha = \alpha_* = 1/\lambda$ gives $\rho B_{\alpha_*} = 0$ and the iteration converges in one step. $\square$

Remark 4.8.2. *For later reference, we note that when the stationary Richardson converges, the linear convergence estimate (4.30) also holds with the Euclidean norm $\|\cdot\|_2$ replaced by a certain A-weighted norm $\|\cdot\|_A$, defined below. That is,*

$$\|\boldsymbol{e}^{k+1}\|_A \leq \rho(B_\alpha)\|\boldsymbol{e}^k\|_A, \quad k = 0, 1, \ldots. \tag{4.32}$$

Here

$$\|\boldsymbol{v}\|_A = \sqrt{(\boldsymbol{v}, \boldsymbol{v})_A} = \sqrt{(A\boldsymbol{v}, \boldsymbol{v})} = \sqrt{\boldsymbol{v}^T A\boldsymbol{v}}, \qquad \boldsymbol{v} \in \mathbb{R}^n, \tag{4.33}$$

where the inner product $(\cdot,\cdot)_A$ satisfies

$$(\boldsymbol{v},\boldsymbol{w})_A = (A\boldsymbol{v},\boldsymbol{w}) = \boldsymbol{v}^T A\boldsymbol{w} = \boldsymbol{w}^T A\boldsymbol{v}, \qquad \boldsymbol{v},\boldsymbol{w} \in \mathbb{R}^n. \tag{4.34}$$

That (4.33) and (4.34) define a norm and an inner product relies on the fact that A is SPD.

4.9 The Jacobi method and the Gauss-Seidel method

The Jacobi and Gauss-Seidel methods exploit the fact that diagonal and triangular systems are easy to solve (recall §4.5). Both methods are obtained by decomposing the matrix A as

$$A = L + D + U$$

where

- D is the diagonal component of A, i.e. $D = (d_{ij})$ with $d_{ii} = a_{ii}$ for each $i = 1,\ldots,n$ and $d_{ij} = 0$ for $i \neq j$;
- L is the strictly lower triangular component of A, i.e. $L = (L_{ij})$ with $L_{ij} = a_{ij}$ for $i > j$ and 0 otherwise;
- U is the strictly upper triangular component of A, i.e. $U = (U_{ij})$ with $U_{ij} = a_{ij}$ for $i < j$ and 0 otherwise.

(Note here that a strictly upper (lower) triangular matrix is an upper (lower) triangular matrix whose diagonal elements are all zeroes. Also the matrices L and U in this section are not to be confused with the matrices appearing in the LU factorization algorithm.)

In the Jacobi method we take the preconditioner to be $P = D$ (so that $N = -(L+U)$). This corresponds to an iteration matrix $B_{\mathrm{J}} = I - D^{-1}A = -D^{-1}(L+U)$. Note that for D to be invertible, we need A to have all non-zero diagonal elements. Provided this is the case, the iteration step for the Jacobi method can be written explicitly in component form as

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^k\right), \quad i = 1,\ldots,n. \tag{4.35}$$

Example 4.9.1. *Let*

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}.$$

Then

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 16 \end{pmatrix}, \quad L = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 \\ 9 & 10 & 0 & 0 \\ 13 & 14 & 15 & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & 2 & 3 & 4 \\ 0 & 0 & 7 & 8 \\ 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

To extract D, L and U from A in Matlab, we use the following commands:

```
>> A = [1,2,3,4;5,6,7,8;9,10,11,12;13,14,15,16];
>> D = diag(diag(A));
>> L = (tril(A) - D);
>> U = (triu(A) - D);
```

The Gauss-Seidel method uses the preconditioner $P = L + D$ (the lower triangular component of A including the diagonal), so that $N = -U$. The iteration matrix is then $B_{\mathrm{GS}} = I - (L + D)^{-1}A = -(L+D)^{-1}U$, and the method may be written explicitly in component form as

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k \right), \quad i = 1, \ldots, n. \tag{4.36}$$

The following convergence results for the Jacobi and Gauss-Seidel methods are classical.

Theorem 4.9.2. *If A is strictly diagonally dominant by rows, that is, if*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \text{ for } i = 1, \ldots, n,$$

then the Jacobi method and the Gauss-Seidel method are both convergent.

Theorem 4.9.3. *If A is symmetric positive definite then the Gauss-Seidel method converges.*

The proofs of the convergence results for Gauss-Seidel are beyond the scope of this course. Here we only give the proof of Theorem 4.9.2 for the Jacobi method.

Proof of Theorem 4.9.2, Jacobi case. The Jacobi iteration matrix B_{J} has entries (b_{ij}) given by $b_{ij} = -a_{ij}/a_{ii}$ for $i \neq j$ and $b_{ii} = 0$ for each i. Hence the infinity norm of B_{J} is equal to (recall (4.5))

$$\|B_{\mathrm{J}}\|_\infty = \max_{i \in \{1,\ldots,n\}} \sum_{j=1}^{n} |b_{ij}| = \max_{i \in \{1,\ldots,n\}} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} = \max_{i \in \{1,\ldots,n\}} \frac{1}{|a_{ii}|} \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|.$$

From this it follows that if A is strictly diagonally dominant by rows, then $\|B_{\mathrm{J}}\|_\infty < 1$, which in turn (by Lemma 4.4.2) implies that $\rho(B_{\mathrm{J}}) < 1$, so the Jacobi method converges. In fact, by the discussion leading up to (4.23) the convergence is linear and we have the error bound

$$\|\boldsymbol{e}^k\|_\infty \leq (\|B_{\mathrm{J}}\|_\infty)^k \|\boldsymbol{e}^0\|_\infty, \quad k = 0, 1, \ldots.$$

□

4.10 Non-stationary methods

The iterative methods we have considered so far are all *stationary*, meaning that the iteration matrix B and vector $\boldsymbol{c}$ in (4.19) stay the same at each iteration. One might ask whether there is any benefit in allowing these to change at each iteration. Such *non-stationary* methods do not correspond to fixed-point iterations. We will consider two examples applicable to symmetric positive definite matrices: the gradient method and the conjugate gradient method. Both have unpreconditioned and preconditioned forms.

Our starting point for both methods is the stationary Richardson iteration, for which $B_\alpha = I - \alpha A$ for some $0 \neq \alpha \in \mathbb{R}$. It is instructive to rewrite the iteration in the "update form"

$$\boldsymbol{x}^{k+1} - \boldsymbol{x}^k = \alpha \boldsymbol{r}^k, \qquad k = 0, 1, 2, \ldots, \tag{4.37}$$

where $\boldsymbol{r}^k = \boldsymbol{b} - A\boldsymbol{x}^k$ is the residual at step k, which itself updates at step $k+1$ by the formula

$$\boldsymbol{r}^{k+1} - \boldsymbol{r}^k = -\alpha A \boldsymbol{r}^k, \qquad k = 0, 1, 2, \ldots. \tag{4.38}$$

Formula (4.37) shows that in the stationary Richardson iteration, to obtain $\boldsymbol{x}^{k+1}$ from $\boldsymbol{x}^k$ we move in the direction of the residual $\boldsymbol{r}^k$, with a relative step length α. We saw in Theorem 4.8.1 that if A is SPD then we can find an optimal value of α so as to minimise the spectral radius $\rho(B_\alpha)$ and obtain the fastest possible convergence. But this requires knowledge of the extremal eigenvalues of A, something which we might not have.

Suppose that we now consider a more general update

$$\boldsymbol{x}^{k+1} - \boldsymbol{x}^k = \alpha_k \boldsymbol{p}^k, \qquad k = 0, 1, 2, \ldots, \tag{4.39}$$

for some update direction $\boldsymbol{p}^k$ and relative step length α_k, which might change at each iteration. We recall our notation $\boldsymbol{e}^k = \boldsymbol{x} - \boldsymbol{x}^k$ for the error at step k, which is related to the residual by

$$\boldsymbol{r}^k = A\boldsymbol{e}^k, \qquad k = 0, 1, 2, \ldots, \tag{4.40}$$

and note that the residual itself updates by

$$\boldsymbol{r}^{k+1} - \boldsymbol{r}^k = -\alpha_k A \boldsymbol{p}^k, \qquad k = 0, 1, 2, \ldots. \tag{4.41}$$

Our aim is to choose $\boldsymbol{p}^k$ and α_k as the iteration progresses (without knowledge of the spectrum of A), so as to optimize the accuracy of the next iterate in a certain sense. As we shall see, the two methods considered below (gradient and conjugate gradient) are both obtained by making a certain choice for $\boldsymbol{p}^k$ and then calculating α_k so as to minimize the size of the error $\boldsymbol{e}^{k+1}$ measured in the A-weighted norm $\|\cdot\|_A$ defined in Remark 4.8.2. This choice is made because it allows the minimizer α_k to be found in closed form. (The same is not true if one attempts to minimize $\|\boldsymbol{e}^{k+1}\|_2$.)

4.10.1 The gradient method

In the gradient method, given $\boldsymbol{x}^k$ we take $\boldsymbol{p}^k = \boldsymbol{r}^k$ (as in the stationary Richardson method), and choose α_k so as to minimise $\|\boldsymbol{e}^{k+1}\|_A$. To achieve this we note that

$$\begin{aligned}\|\boldsymbol{e}^{k+1}\|_A^2 &= (A\boldsymbol{e}^{k+1}, \boldsymbol{e}^{k+1}) \\ &= (\boldsymbol{r}^{k+1}, A^{-1}\boldsymbol{r}^{k+1}) \\ &= (\boldsymbol{r}^k - \alpha_k A\boldsymbol{r}^k, A^{-1}\boldsymbol{r}^k - \alpha_k\boldsymbol{r}^k) \\ &= \|\boldsymbol{e}^k\|_A^2 - 2\alpha_k\|\boldsymbol{r}^k\|_2^2 + \alpha_k^2\|\boldsymbol{r}^k\|_A^2.\end{aligned}$$

This is a quadratic in α_k, and by differentiating with respect to α_k and setting the resulting expression equal to zero, we find that the unique minimum is at

$$\alpha_k = \frac{\|\boldsymbol{r}^k\|_2^2}{\|\boldsymbol{r}^k\|_A^2}. \tag{4.42}$$

To summarise, the *gradient method* is defined as follows: Given $\boldsymbol{x}^0$ set $\boldsymbol{r}^0 = \boldsymbol{b} - A\boldsymbol{x}^0$. Then for $k = 0, 1, 2, \ldots$ until convergence, update

$$\begin{aligned}\alpha_k &= \frac{\|\boldsymbol{r}^k\|_2^2}{\|\boldsymbol{r}^k\|_A^2}, \\ \boldsymbol{x}^{k+1} &= \boldsymbol{x}^k + \alpha_k\boldsymbol{r}^k, \\ \boldsymbol{r}^{k+1} &= \boldsymbol{r}^k - \alpha_k A\boldsymbol{r}^k.\end{aligned}$$

Theorem 4.10.1. *If A is SPD then the gradient method converges linearly, with*

$$\|\boldsymbol{e}^{k+1}\|_A \le \frac{K_2(A)-1}{K_2(A)+1}\|\boldsymbol{e}^k\|_A, \quad k \ge 0. \tag{4.43}$$

Proof. Given $\boldsymbol{x}_k$, let $\boldsymbol{e}_*^{k+1}$ be the error resulting from a *single* stationary Richardson step with $\alpha = \alpha_*$. The definition of α_k says $\|\boldsymbol{e}^{k+1}\|_A \le \|\boldsymbol{e}_*^{k+1}\|_A$, so by Remark 4.8.2 it follows that

$$\|\boldsymbol{e}^{k+1}\|_A \le \|\boldsymbol{e}_*^{k+1}\|_A \le \rho(B_{\alpha_*})\|\boldsymbol{e}^k\|_A = \frac{K_2(A)-1}{K_2(A)+1}\|\boldsymbol{e}^k\|_A.$$

$\square$

Note that the convergence constant in (4.43) for the gradient method (in which we choose the step-length α_k adaptively for each k) is exactly the same as that achieved by the optimal choice of α in the stationary Richardson method (see Theorem 4.8.1 and Remark 4.8.2).

Noting that

$$\frac{K_2(A)-1}{K_2(A)+1} = 1 - \frac{2}{K_2(A)+1},$$

we see that if $K_2(A)$ is large then the convergence constant may be close to 1, giving very slow reduction in the error. To combat this, the gradient method can be generalised to include a preconditioner. For the unpreconditioned version, the (nonstationary) iteration matrix was given by $B = I - \alpha_k A$. For the preconditioned version we change this to $B = I - \alpha_k P^{-1}A$, where P is some symmetric positive definite matrix. This corresponds to using an update direction $\boldsymbol{p}^k = P^{-1}\boldsymbol{r}^k$. As before we then choose α_k to minimise $\|\boldsymbol{e}^{k+1}\|_A$. A similar calculation to that presented above shows that the optimal choice is

$$\alpha_k = \frac{(\boldsymbol{r}^k, \boldsymbol{p}^k)}{(\boldsymbol{p}^k, A\boldsymbol{p}^k)} = \frac{\|\boldsymbol{p}^k\|_P^2}{\|\boldsymbol{p}^k\|_A^2}. \tag{4.44}$$

The resulting *preconditioned gradient method* is defined as follows:

Given $\boldsymbol{x}^0$ and P set $\boldsymbol{r}^0 = \boldsymbol{b} - A\boldsymbol{x}^0$. Then for $k = 0, 1, 2, \ldots$ until convergence, update

$$\begin{aligned} P\boldsymbol{p}^k &= \boldsymbol{r}^k, \\ \alpha_k &= \frac{\|\boldsymbol{p}^k\|_P^2}{\|\boldsymbol{p}^k\|_A^2}, \\ \boldsymbol{x}^{k+1} &= \boldsymbol{x}^k + \alpha_k \boldsymbol{p}^k, \\ \boldsymbol{r}^{k+1} &= \boldsymbol{r}^k - \alpha_k A\boldsymbol{p}^k. \end{aligned}$$

If P is SPD then one can prove that the method converges linearly with

$$\|\boldsymbol{e}^{k+1}\|_A \leq \frac{K_2(P^{-1}A)-1}{K_2(P^{-1}A)+1}\|\boldsymbol{e}^k\|_A, \quad k \geq 0, \tag{4.45}$$

implying that

$$\|\boldsymbol{e}^k\|_A \leq \left(\frac{K_2(P^{-1}A)-1}{K_2(P^{-1}A)+1}\right)^k \|\boldsymbol{e}^0\|_A, \quad k \geq 1. \tag{4.46}$$

Given A, the point is to try and choose P such that $K_2(P^{-1}A)$ is much smaller than $K_2(A)$, so that $\frac{K_2(P^{-1}A)-1}{K_2(P^{-1}A)+1} < \frac{K_2(A)-1}{K_2(A)+1}$, improving the rate of convergence of the method over the unpreconditioned version.

In the following table we compare the definitions of the gradient method (G) and the preconditioned gradient method (GP).

G	**GP**
$\boldsymbol{r}^k = \boldsymbol{b} - A\boldsymbol{x}^k$	$\boldsymbol{r}^k = \boldsymbol{b} - A\boldsymbol{x}^k$
	$P\boldsymbol{p}^k = \boldsymbol{r}^k$
$\alpha_k = \frac{(\boldsymbol{r}^k,\boldsymbol{r}^k)}{(A\boldsymbol{r}^k,\boldsymbol{r}^k)}$	$\alpha_k = \frac{(\boldsymbol{r}^k,\boldsymbol{p}^k)}{(A\boldsymbol{p}^k,\boldsymbol{p}^k)}$
$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha_k\boldsymbol{r}^k$	$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha_k\boldsymbol{p}^k$

Example 4.10.2. *Consider the system $A\boldsymbol{x} = \boldsymbol{b}$, where $A = \left[\begin{smallmatrix} 5 & 1 \\ 6 & 8 \end{smallmatrix}\right]$, and $\boldsymbol{b}$ is such that the exact solution is given by $\boldsymbol{x} = (1,1)^T$. We approximate the solution using the gradient method (G) and the preconditioned gradient method with $P = D$ (GP). The relative error, plotted as a function of the iteration count k, is shown in Figure 4.4. Note the faster convergence of GP compared to G. Note also that A is not SPD, so this example shows that the gradient method can work for some non-SPD matrices. (But see also Example 4.10.7.)*

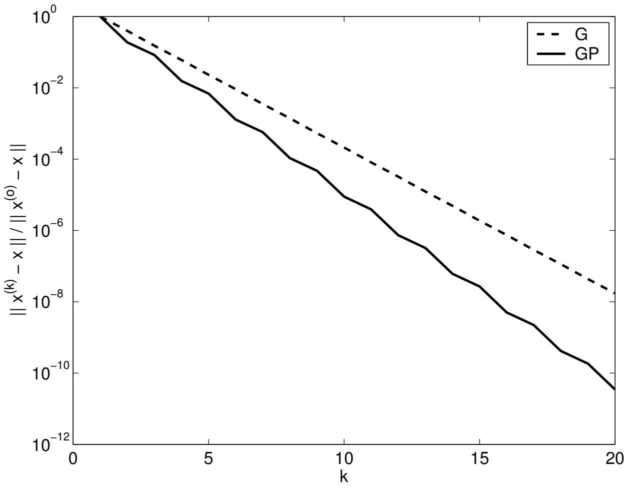


Figure 4.4: *Relative error for the gradient method (G) and the preconditioned gradient method with $P = D$ (GP).*

Remark 4.10.3. *Why the name "gradient method"? To explain this, we first note that for a symmetric positive definite matrix A the problem of solving $A\boldsymbol{x} = \boldsymbol{b}$ is equivalent to finding the unique minimiser $\boldsymbol{x}$ of the convex function*

$$\Phi : \mathbb{R}^n \to \mathbb{R}, \quad \Phi(\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^T A\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{b} = \frac{1}{2}\sum_{i,j=1}^{n} y_i a_{ij} y_j - \sum_{i=1}^{n} b_i y_i,$$

a quadratic form sometimes referred to as the "energy" of the system. To see the equivalence of these two problems, it is enough to note that the gradient of Φ *is given by*

$$\nabla\Phi(\boldsymbol{y}) = \frac{1}{2}(A^T + A)\boldsymbol{y} - \boldsymbol{b} = A\boldsymbol{y} - \boldsymbol{b}\,,$$

and that if $A\boldsymbol{x} = \boldsymbol{b}$ *then*

$$\Phi(\boldsymbol{y}) - \Phi(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x})^T A(\boldsymbol{y} - \boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{x}\|_A^2, \qquad \boldsymbol{y} \neq \boldsymbol{x}.$$

These formulas also show that the direction of steepest descent of the function Φ *at a point* $\boldsymbol{y}$ *is given by* $-\nabla\Phi(\boldsymbol{y}) = \boldsymbol{b} - A\boldsymbol{y}$*, i.e. the residual, and that minimising* $\Phi(\boldsymbol{y}(\alpha))$ *over some parameter* α *is equivalent to minimising* $\|\boldsymbol{y}(\alpha) - \boldsymbol{x}\|_A$*. Hence each iteration of the gradient method corresponds precisely to a steepest descent minimization of* Φ*.*

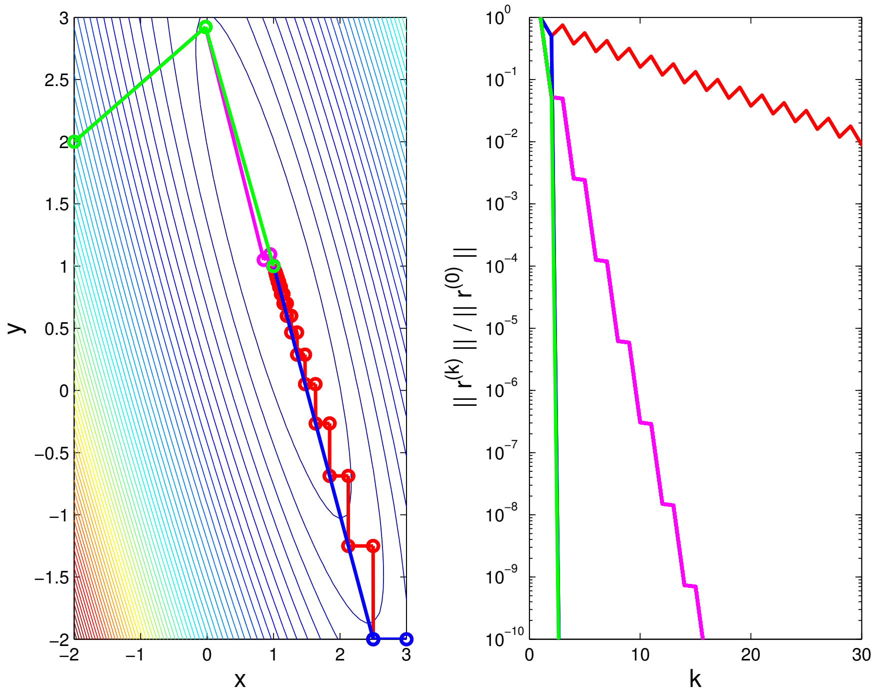


Figure 4.5: *Convergence histories for the gradient method (red, pink) and the conjugate gradient method (blue, green) for a* 2×2 *system, using two different different starting points. The contour lines are level curves of the associated energy functional* Φ *of Remark 4.10.3. Observe that if an unfortunate choice of initial data is made the gradient method converges very slowly due to a zig-zagging behaviour. This is eliminated by the conjugate gradient method, which always converges after* $n = 2$ *iterations (see Remark 4.10.5).*

4.10.2 The conjugate gradient method

One property of the gradient method that limits its speed of convergence is the fact that each increment is orthogonal to the previous increment (with respect to the Euclidean inner product $(\cdot,\cdot)$. To see this, note that for the gradient method

$$(\boldsymbol{p}^k, \boldsymbol{p}^{k-1}) = (\boldsymbol{r}^k, \boldsymbol{r}^{k-1}) = (\boldsymbol{r}^{k-1} - \alpha_{k-1} A \boldsymbol{r}^{k-1}, \boldsymbol{r}^{k-1}) = \|\boldsymbol{r}^{k-1}\|_2^2 - \alpha_{k-1}\|\boldsymbol{r}^{k-1}\|_A^2,$$

which is equal to zero by the definition of α_{k-1}. The effect of this orthogonality is that for bad choices of initial data the iterates can converge in a zig-zag fashion, taking a long time to approach the solution. For an illustration of this see Figure 4.5.

We saw that the gradient method consists of the iteration

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha_k \boldsymbol{r}^k.$$

Therefore

$$\boldsymbol{r}^k = \boldsymbol{b} - A\boldsymbol{x}^k = \boldsymbol{b} - A\boldsymbol{x}^{k-1} - \alpha_{k-1}\boldsymbol{r}^{k-1} = (I - \alpha_{k-1}A)\boldsymbol{r}^{k-1}$$

Hence an induction argument shows that

$$\boldsymbol{r}^k = \prod_{j=0}^{k-1} (I - \alpha_j A)\, \boldsymbol{r}^0$$

The product of matrices $\prod_{j=0}^{k-1}(I - \alpha_j A)$ can be expanded into a linear combination of powers of A as

$$\prod_{j=0}^{k-1} (I - \alpha_j A) = \sum_{j=0}^{k} c_j A^j,$$

for some real numbers c_j, $j = 0, \dots, k$. For example, if $k = 2$ then

$$(I - \alpha_1 A)(I - \alpha_0 A) = I - (\alpha_1 + \alpha_2)A + \alpha_1\alpha_2 A^2,$$

so we get $c_0 = 1$, $c_1 = \alpha_1 + \alpha_2$ and $c_2 = \alpha_1\alpha_2$. Therefore the residual $\boldsymbol{r}^k$ can be written as the linear combination of powers of A times $\boldsymbol{r}^0$

$$\boldsymbol{r}^k = \sum_{j=0}^{k} c_j A^j \boldsymbol{r}^0.$$

This shows that $\boldsymbol{r}^k$ is always an element of the linear subspace V_{k+1} of $\mathbb{R}^n$ defined by

$$V_{k+1} = \operatorname{span}\left\{\boldsymbol{r}^0,\ A\boldsymbol{r}^0, \dots, A^k\boldsymbol{r}^0\right\}.$$

The space V_k is called the k-th Krylov subspace generated by A and $\boldsymbol{r}^0$. Notice that the sequence of Krylov subspaces are nested, i.e.

$$V_1 \subset V_2 \subset \cdots \subset V_k \quad \forall k.$$

Therefore, the sequence of residuals $\boldsymbol{r}^j \in V_{j+1} \subset V_{k+1}$ for all $j \leq k$, Notice that the iterates $\boldsymbol{x}^k$ satisfy

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha_k \boldsymbol{r}^k = \boldsymbol{x}^{k-1} + \alpha_{k-1}\boldsymbol{r}^{k-1} + \alpha_k \boldsymbol{r}^k = \boldsymbol{x}^0 + \sum_{j=0}^{k} \alpha_j \boldsymbol{r}^j,$$

from which we find that $\boldsymbol{x}^{k+1} - \boldsymbol{x}^0 \in V_{k+1}$ for all $k \geq 0$, or equivalently

$$\boldsymbol{x}^k - \boldsymbol{x}^0 \in V_k \quad \forall k \geq 1.$$

In other words, the iterates of the gradient method are elements of the Krylov subspaces plus the initial guess. Since the gradient method is convergent, it is clear that the sequence of Krylov subspaces V_k contain increasingly accurate approximations of $\boldsymbol{x} - \boldsymbol{x}^0$. To improve on the gradient method, we can consider the problem of finding *the best approximation* that minimizes the A-norm of the error over all vectors from the k-Krylov subspace. This leads to the Conjugate Gradient method, where we define the k-th iterate as

$$\boxed{\boldsymbol{x}^k = \underset{\tilde{\boldsymbol{x}} - \boldsymbol{x}^0 \in V_k}{\operatorname{argmin}} \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_A = \underset{\tilde{\boldsymbol{v}} \in V_k}{\operatorname{argmin}} \|\boldsymbol{x} - \boldsymbol{x}^0 - \tilde{\boldsymbol{v}}\|_A.}$$

In other words, we aim to find $\boldsymbol{x}^k$ such that $\boldsymbol{x}^k - \boldsymbol{x}^0 \in V_k$ and

$$\|\boldsymbol{x} - \boldsymbol{x}^k\|_A = \min_{\tilde{\boldsymbol{x}} - \boldsymbol{x}^0 \in V_k} \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_A.$$

An efficient way of computing the vector $\boldsymbol{x}^k$ is to build an alternative basis of the space V_k. In particular we will construct a basis of vectors that are A-*orthogonal*: i.e. we will build a basis $\{\boldsymbol{p}^j\}_{j=0}^{k-1}$ of V_k such that

$$(\boldsymbol{p}^j, \boldsymbol{p}^i)_A = \delta_{ij}\|\boldsymbol{p}^i\|_A^2 = \begin{cases} \|\boldsymbol{p}^i\|_A^2 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

We will see how to build this basis shortly below, but for now assuming that such a basis is available, then we can express the solution $\boldsymbol{x}^k$ as

$$\boldsymbol{x}^k = \boldsymbol{x}^0 + \sum_{j=0}^{k-1} \alpha_j \boldsymbol{p}^j.$$

Using then the equation for the error $\boldsymbol{e}^k = \boldsymbol{e}^0 - \sum_{j=0}^{k-1} \alpha_j \boldsymbol{p}^j$, we can expand

$$\begin{aligned} \|\boldsymbol{e}^k\|_A^2 = \|\boldsymbol{e}^0 - \sum_{j=0}^{k-1} \alpha_j \boldsymbol{p}^j\|_A^2 &= \|\boldsymbol{e}^0\|_A^2 - \sum_{j=0}^{k-1} 2\alpha_j \left(\boldsymbol{e}^0, \boldsymbol{p}^j\right)_A + \|\sum_{j=0}^{k-1} \alpha_j \boldsymbol{p}^j\|_A^2 \\ &= \|\boldsymbol{e}^0\|_A^2 - \sum_{j=0}^{k-1} 2\alpha_j \left(\boldsymbol{e}^0, \boldsymbol{p}^j\right)_A + \sum_{j=0}^{k-1} \alpha_j^2 \|\boldsymbol{p}^j\|_A^2, \end{aligned}$$

where we have used the A-orthogonality of the basis to expand $\|\sum_{j=0}^{k-1}\alpha_j \boldsymbol{p}^j\|_A^2 = \sum_{j=0}^{k-1}\alpha_j^2\|\boldsymbol{p}^j\|_A^2$. It is then seen that the error is minimized when

$$\alpha_j = \frac{(\boldsymbol{e}^0, \boldsymbol{p}^j)_A}{\|\boldsymbol{p}^j\|_A^2} = \frac{(A\boldsymbol{e}^0, \boldsymbol{p}^j)}{\|\boldsymbol{p}^j\|_A^2} = \frac{(\boldsymbol{r}^0, \boldsymbol{p}^j)}{\|\boldsymbol{p}^j\|_A^2} \quad \forall j = 0, \dots, n-1.$$

Notice that α_j is then easily computable since it can be expressed solely in terms of the computable vectors $\boldsymbol{r}^0$, $\boldsymbol{p}^j$ and the matrix A. Notice also that the value of α_j does not depend on the iteration number k. This property implies that

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^0 + \sum_{j=0}^{k}\alpha_j\boldsymbol{p}^j = \underbrace{\boldsymbol{x}^0 + \sum_{j=0}^{k-1}\alpha_j\boldsymbol{p}^j}_{=\boldsymbol{x}^k} + \alpha_k\boldsymbol{p}^k = \boldsymbol{x}^k + \alpha_k\boldsymbol{p}^k. \tag{4.47}$$

The significance of (4.47) is that once $\boldsymbol{x}^k$ and $\boldsymbol{p}^k$ are computed, we can obtain $\boldsymbol{x}^{k+1}$ very easily by adding the increment $\alpha_k\boldsymbol{p}^k$ to the previous iterate $\boldsymbol{x}^k$. Hence we don't need to store the whole sequence of basis vectors $\{\boldsymbol{p}^j\}_{j=0}^{k-1}$ when computing the next iterate $\boldsymbol{x}^{k+1}$. This means that the Conjugate Gradient method can be implemented as a *nonstationary* iterative method. A further simplification for the practical implementation is that we also don't need to store $\boldsymbol{r}^0$, since we have the identity

$$(\boldsymbol{r}^k, \boldsymbol{p}^k) = (\boldsymbol{r}^0, \boldsymbol{p}^k) - \sum_{j=0}^{k-1}\alpha_j \underbrace{(A\boldsymbol{p}^j, \boldsymbol{p}^k)}_{=(\boldsymbol{p}^j,\boldsymbol{p}^k)_A=0} = (\boldsymbol{r}^0, \boldsymbol{p}^k),$$

so in fact the value of α_k can be computed directly from just $\boldsymbol{r}^k$ and $\boldsymbol{p}^k$ by

$$\alpha_k = \frac{(\boldsymbol{r}^k, \boldsymbol{p}^k)}{\|\boldsymbol{p}^k\|_A^2}, \quad k \geq 0.$$

Notice that the values of the α_j also imply that the residual vector $\boldsymbol{r}^k \in V_{k+1}$ also satisfies $(\boldsymbol{r}^k, \boldsymbol{p}^j) = 0$ for all $j = 0, \dots, k-1$, so in other words $\boldsymbol{r}^k$ is orthogonal to the space V_k with respect to the Euclidean inner-product.

We now show how to construct the basis vectors $\boldsymbol{p}^j$, $j \geq 0$. Since $V_1 = \operatorname{span}\{\boldsymbol{r}^0\}$ we set $\boldsymbol{p}^0 = \boldsymbol{r}^0$. Then, for each $k \geq 1$, supposing that the vectors $\{\boldsymbol{p}^j\}_{j=0}^{k-1}$ form a basis of V_k, we set

$$\boldsymbol{p}^k = \boldsymbol{r}^k - \beta_{k-1}\boldsymbol{p}^{k-1}, \quad \beta_{k-1} = \frac{(\boldsymbol{r}^k, \boldsymbol{p}^{k-1})_A}{\|\boldsymbol{p}^{k-1}\|_A^2} \tag{4.48}$$

The following Lemma shows that this implies that A-orthogonality of $\boldsymbol{p}^k$ to all previous vectors $\{\boldsymbol{p}^j\}_{j=0}^{k-1}$

Lemma 4.10.4. *The vector $\boldsymbol{p}^k$ defined by (4.48) is A-orthogonal to $\boldsymbol{p}^j$ for all $j = 0, \dots, k-1$, i.e.*

$$(\boldsymbol{p}^k, \boldsymbol{p}^j)_A = 0 \quad \forall j = 0, \dots, k-1.$$

Proof. First, notice that for $\boldsymbol{p}^j \in V_{j+1}$ we have $A\boldsymbol{p}^j \in V_{j+2} \subset V_k$ whenever $j \leq k-2$. Therefore, for all $j \leq k-2$,

$$(\boldsymbol{p}^k, \boldsymbol{p}^j)_A = (\boldsymbol{r}^k, A\boldsymbol{p}^j) - \beta_{k-1}(\boldsymbol{p}^{k-1}, \boldsymbol{p}^j)_A = 0,$$

where the first term in the second equation vanishes since $\boldsymbol{r}^k$ is orthogonal to $A\boldsymbol{p}^j \in V_k$ with respect to the Euclidean inner-product, and the second term vanishes since $\boldsymbol{p}^{k-1}$ is A-orthogonal to $\boldsymbol{p}^j$. We then also easily check that the choice of β_{k-1} implies that $(\boldsymbol{p}^k, \boldsymbol{p}^{k-1})_A = (\boldsymbol{r}^k, \boldsymbol{p}^{k-1})_A - \beta_{k-1}\|\boldsymbol{p}^{k-1}\|_A^2 = 0$. $\square$

Remark 4.10.5. *Since the Krylov subspaces are nested, i.e.* $V_1 \subset V_2 \subset \ldots V_k$*, we always have* $\dim V_k \leq \dim V_{k+1}$*. Usually we might expect that* $\dim V_{k+1} = \dim V_k + 1$*, however in general we can only guarantee that* $\dim V_{k+1} \leq \dim V_k + 1$*, since it is in principle possible that some of the vectors* $\boldsymbol{r}^0$*,* $A\boldsymbol{r}^0$*, …* $A^k\boldsymbol{r}^0$ *are linearly dependent (for instance if* $\boldsymbol{r}^0$ *is an eigenvector of* A*). However, cases where the Krylov subspaces do not have strictly increasing dimension does not pose a problem for the Conjugate Gradient method. Indeed, suppose that* $k \geq 1$ *is the smallest integer such that* $\dim V_{k+1} = \dim V_k$*. Since* V_k *is a subspace of* V_{k+1} *with same dimension, we then must have* $V_{k+1} = V_k$*. From the facts that the residual* $\boldsymbol{r}^k \in V_{k+1}$ *is orthogonal to* V_k*, we therefore conclude that* $\boldsymbol{r}^k \in V_k$ *is orthogonal to* V_k*, which implies that* $\boldsymbol{r}^k = \boldsymbol{0}$*, which is equivalent to* $\boldsymbol{x}^k = \boldsymbol{x}$ *the exact solution. In other words the Conjugate Gradient method has found the exact solution at the k-th iteration.The converse also holds, and thus for each iteration k, either* $\dim V_{k+1} = \dim V_k + 1$ *or the Conjugate Gradient method has found the exact solution* $\boldsymbol{x}^k = \boldsymbol{x}$*. Since in all cases we have* $\dim V_k \leq n$ *the dimension of* $\mathbb{R}^n$ *for all* $k \geq 1$*, it is then easy to see that the Conjugate Gradient method must converge to the exact solution in at most n iterations.*

The form of Conjugate Gradient method shown above does not involve the use of a preconditioner. For ill-conditioned problems where $\kappa_2(A)$ is large, this can lead to slow convergence. This can be alleviated by considering the *preconditioned conjugate gradient method*, where for some SPD preconditioner P we use instead the A-orthogonal increment vector $\boldsymbol{p}^k$ defined by

$$\boldsymbol{p}^k = P^{-1}\boldsymbol{r}^k - \beta_{k-1}\boldsymbol{p}^{k-1}, \quad \beta_{k-1} = \frac{(P^{-1}\boldsymbol{r}^k, \boldsymbol{p}^{k-1})_A}{\|\boldsymbol{p}^{k-1}\|_A^2}.$$

Notice that this coincides with the unpreconditiond form when $P = I$ the identity matrix.

To summarize, we now state the algorithm of the *preconditioned conjugate gradient method.*

Given an initial guess $\boldsymbol{x}^0$, compute $\boldsymbol{r}^0 = \boldsymbol{b} - A\boldsymbol{x}^0$, $P\boldsymbol{z}^0 = \boldsymbol{r}^0$, $\boldsymbol{p}^0 = \boldsymbol{z}^0$. Then for $k \geq 0$ until convergence, update

$$\begin{aligned}
\alpha_k &= \frac{(\boldsymbol{p}^k, \boldsymbol{r}^k)}{(\boldsymbol{p}^k, A\boldsymbol{p}^k)}, \\
\boldsymbol{x}^{k+1} &= \boldsymbol{x}^k + \alpha_k \boldsymbol{p}^k, \\
\boldsymbol{r}^{k+1} &= \boldsymbol{r}^k - \alpha_k A\boldsymbol{p}^k, \\
P\boldsymbol{z}^{k+1} &= \boldsymbol{r}^{k+1}, \\
\beta_k &= \frac{(A\boldsymbol{p}^k, \boldsymbol{z}^{k+1})}{(A\boldsymbol{p}^k, \boldsymbol{p}^k)}, \\
\boldsymbol{p}^{k+1} &= \boldsymbol{z}^{k+1} - \beta_k \boldsymbol{p}^k .
\end{aligned}$$

Under the assumption that P is SPD, one can prove the error bound

$$\|\boldsymbol{e}^k\|_A \leq 2 \left(\frac{\sqrt{K_2(P^{-1}A)} - 1}{\sqrt{K_2(P^{-1}A)} + 1} \right)^k \|\boldsymbol{e}^0\|_A , \quad k \geq 0 . \tag{4.49}$$

Comparing (4.49) to (4.46) suggests that when $K_2(P^{-1}A) \gg 1$, the preconditioned conjugate gradient method should converge much more rapidly than the preconditioned gradient method, because $\sqrt{K_2(P^{-1}A)} \ll K_2(P^{-1}A)$.

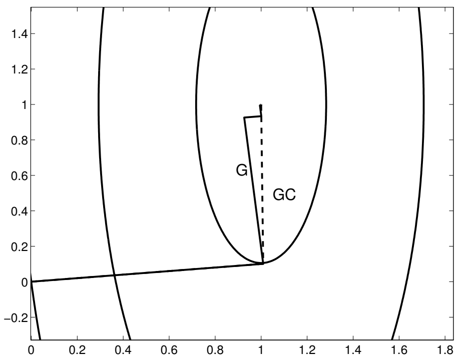


Figure 4.6: *Descent directions for the conjugate gradient method (CG, dotted line) and the gradient method (G, solid line). Observe that CG converges to the solution in two iterations.*

Example 4.10.6. *Consider the linear system*

$$\begin{cases} 2x_1 + x_2 &= 1 \\ x_1 + 3x_2 &= 0 \end{cases}, \tag{4.50}$$

with corresponding matrix $A = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$*, which is symmetric and positive definite. The exact solution to the system is* $x_1 = 3/5$, $x_2 = -1/5$.

We approximate the solution using a number of different iterative methods, all initialised with

$$\boldsymbol{x}^0 = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{2} \end{pmatrix}.$$

In Figure 4.7 we show the value of $\frac{\|\boldsymbol{e}^k\|}{\|\boldsymbol{e}^0\|} = \frac{\|\boldsymbol{x}^k - \boldsymbol{x}\|}{\|\boldsymbol{x}^0 - \boldsymbol{x}\|}$ *for the Jacobi, Gauss-Seidel, and diagonally preconditioned gradient methods. We also show results for a further method not described in detail here, the successive over-relaxation (SOR) method, which performs even better than the three other methods for this problem.*

The first steps for the Jacobi, Gauss-Seidel, and diagonally preconditioned gradient methods are as follows.

1. ***Jacobi method:***

$$\begin{cases} x_1^{(1)} &= \frac{1}{2}(1 - x_2^{(0)}) \\ x_2^{(1)} &= -\frac{1}{3}x_1^{(0)} \end{cases} \Rightarrow \begin{cases} x_1^{(1)} &= \frac{1}{4} \\ x_2^{(1)} &= -\frac{1}{3} \end{cases}$$

2. ***Gauss-Seidel method:***

$$\begin{cases} x_1^{(1)} &= \frac{1}{2}(1 - x_2^{(0)}) \\ x_2^{(1)} &= -\frac{1}{3}x_1^{(1)} \end{cases} \Rightarrow \begin{cases} x_1^{(1)} &= \frac{1}{4} \\ x_2^{(1)} &= -\frac{1}{12} \end{cases}$$

3. ***Preconditioned gradient method with*** $P = D = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$***:***

Writing the first iteration in increment form, we have to solve

$$\begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} x_1^{(1)} - 1 \\ x_2^{(1)} - \frac{1}{2} \end{pmatrix} = \alpha_0 \boldsymbol{r}^0,$$

where

$$\boldsymbol{r}^0 = \boldsymbol{b} - A\boldsymbol{x}^0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix} \boldsymbol{x}^0 = \begin{pmatrix} -3/2 \\ -5/2 \end{pmatrix}.$$

Clearly,

$$P^{-1} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/3 \end{pmatrix}$$

so

$$\boldsymbol{p}^0 = P^{-1}\boldsymbol{r}^0 = \begin{pmatrix} -3/4 \\ -5/6 \end{pmatrix} \quad \textit{and then} \quad \alpha_0 = \frac{(\boldsymbol{r}^0, \boldsymbol{z}^0)}{(A\boldsymbol{z}^0, \boldsymbol{z}^0)} = \frac{77}{107}.$$

Thus $\boldsymbol{x}^1 = \boldsymbol{x}^0 + \alpha_0 \boldsymbol{z}^0 = \begin{pmatrix} 1.5397 \\ 1.0997 \end{pmatrix}$.

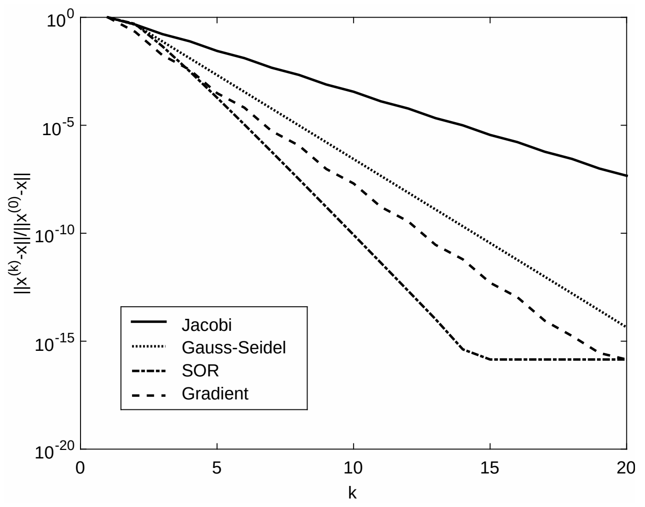


Figure 4.7: *Behaviour of the relative error for different iterative methods applied to the system (4.50)*

Example 4.10.7. *Now consider the system*

$$\begin{cases} 2x_1 + x_2 & = 1 \\ -x_1 + 3x_2 & = 0 \end{cases} \tag{4.51}$$

the solution of which is $x_1 = 3/7$, $x_2 = 1/7$.

The matrix $A = \begin{pmatrix} 2 & 1 \\ -1 & 3 \end{pmatrix}$ is not symmetric, but is clearly diagonally dominant by rows, and hence both Jacobi and Gauss-Seidel should converge.

We initialise both methods using the initial guess

$$\boldsymbol{x}^0 = \begin{pmatrix} x_1^0 \\ x_2^0 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{2} \end{pmatrix}.$$

Figure 4.8 shows the resulting plots of $\frac{\|\mathbf{e}^k\|}{\|\mathbf{e}^0\|} = \frac{\|\boldsymbol{x}^k - \boldsymbol{x}\|}{\|\boldsymbol{x}^0 - \boldsymbol{x}\|}$ for the Jacobi and Gauss-Seidel methods. Note that Gauss-Seidel converges more rapidly than Jacobi. This is to be expected as the preconditioner is working harder. The price we pay is that each iteration step is slightly more expensive.

Note that since the matrix A is not symmetric positive definite, if we were to apply the gradient or the conjugate gradient methods there is no guarantee that they would converge. Even if they do converge the convergence may be slow. In fact the conjugate gradient method does not converge for this problem.

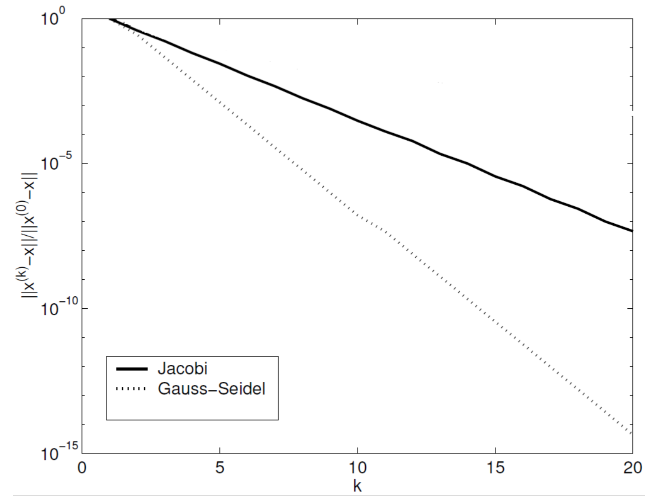


Figure 4.8: *Behaviour of the relative error for different iterative methods applied to the system (4.51)*

4.11 Stopping criteria for iterative methods

Just as in the nonlinear case, we need to specify stopping criteria to control when our iterative method should terminate. Ideally we would like to stop as soon as the relative error is below some user defined tolerance *tol*, i.e.

$$\frac{\|\boldsymbol{x}^k - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq tol.$$

But since the relative error is unknown we need some *a posteriori* error estimator that is easy to compute at each iterate k. As before, two natural quantities to consider as surrogates for the actual error are the computational residual and the iterative increment.

A residual-based criterion would stop the iteration once

$$\|\boldsymbol{r}^k\| = \|\boldsymbol{b} - A\boldsymbol{x}^k\| \leq tol\|\boldsymbol{b}\|.$$

Recalling from (4.17) that

$$\frac{\|\boldsymbol{x}^k - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq K(A)\frac{\|\boldsymbol{r}^k\|}{\|\boldsymbol{b}\|}, \tag{4.52}$$

we see that the actual relative error could be a factor of $K(A)$ larger than *tol* if this residual-based criterion is used. This may be acceptable if $K(A)$ is moderate, or at least readily estimated. But if $K(A)$ is very large or completely unknown, then the residual-based criterion may not give reliable results. Effective preconditioning will make residual-based error estimation more accurate, because for a preconditioned method one would obtain the modified estimate

$$\frac{\|\boldsymbol{x}^k - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq K(P^{-1}A)\frac{\|P^{-1}\boldsymbol{r}^k\|}{\|P^{-1}\boldsymbol{b}\|}.$$

Note that the multiplicative factor in front of the residual is now the condition number of the preconditioned system, which should be significantly smaller than that of the original system. If this holds then the modified residual-based criterion

$$\|P^{-1}\boldsymbol{r}^k\| \leq tol\|P^{-1}\boldsymbol{b}\|$$

should be more effective than the unpreconditioned version.

An estimator based on the increment $\boldsymbol{\delta}^k = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k$ would terminate the iteration once

$$\|\boldsymbol{\delta}^k\| \leq tol\|\boldsymbol{b}\|.$$

For a consistent stationary method with iteration matrix B we have $\boldsymbol{e}^{k+1} = B\boldsymbol{e}^k$, and hence, assuming that the matrix and vector norms are compatible, the triangle inequality gives

$$\|\boldsymbol{e}^k\| = \|\boldsymbol{e}^{k+1} + \boldsymbol{\delta}^k\| \leq \|B\|\|\boldsymbol{e}^k\| + \|\boldsymbol{\delta}^k\|.$$

Assuming that $\|B\| < 1$, we can rearrange this estimate to give

$$\|\boldsymbol{e}^k\| \leq \frac{1}{1-\|B\|}\|\boldsymbol{\delta}^k\|.$$

The norm $\|B\|$ may not be straightforward to compute or estimate. But if B is symmetric then $\|B\|_2 = \rho(B)$ (recall Lemma 4.4.3), and if we work with $\|\cdot\|_2$ throughout, we have that

$$\|\boldsymbol{e}^k\|_2 \leq \frac{1}{1-\rho(B)}\|\boldsymbol{\delta}^k\|_2.$$

Recall that a necessary and sufficient condition for convergence is $\rho(B) < 1$. Smaller $\rho(B)$ implies both faster convergence, and a more reliable increment-based stopping criterion.

4.12 Computational cost

As we discussed earlier in the chapter, the cost of a direct method such as Gaussian elimination (LU factorization) generally scales like $O(n^3)$ flops as the size n of the matrix increases. Analysing the cost of iterative methods is more complicated because one has to take into account both the cost of each iteration, and the number of iterations required to achieve the desired tolerance. In the methods considered here, the cost of each iteration will in general be dominated by the cost of matrix-vector multiplication involving the iteration matrix B. In general the cost of this is $O(n^2)$ flops. Therefore, in order for the iterative method to be cheaper than the direct solver we need the number of iterations to be much less than n. Ideally one would like the number of iterations to stay bounded or increase only very slowly as n increases. But achieving this usually requires the construction of a good preconditioner, which is not always straightforward.

This general analysis is somewhat pessimistic because if the matrix is sparse or has a special structure (such as the tridiagonal matrix of (4.5.1)) then matrix-vector multiplication may be significantly cheaper than $O(n^2)$, even as cheap as $O(n)$. In this case iterative methods can be extremely cheap compared to direct methods, unless the latter can also exploit the special structure of the matrix.[4]

We end the chapter by illustrating these results with some numerical examples.

Example 4.12.1. *Consider the matrix*

$$A = \begin{pmatrix} 5 & 7 \\ 7 & 10 \end{pmatrix} \tag{4.53}$$

for which $K(A) \approx 223$. In Figure 4.9 we compare the residual and the relative error when the Gauss-Seidel method is used. Contrasting this with Figure 4.10, which shows the analogous plot for a better conditioned 2×2 system that has $K(A) \approx 2.6$, we clearly see the effect of conditioning on the quality of the residual-based error estimator.

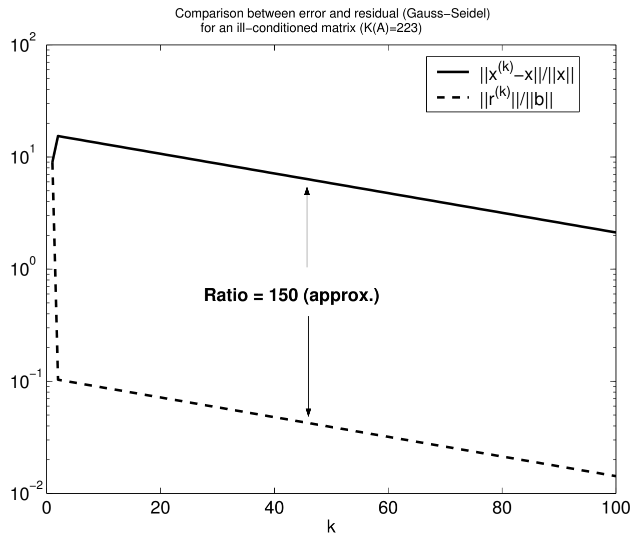


Figure 4.9: *Comparison between relative error and the residual (Gauss-Seidel) for the ill-conditioned matrix (4.53).*

Example 4.12.2. *(***4.5.1 continued***)*

We revisit Example 4.5.1 and solve the system for $n = 25$ using the Gauss-Seidel method and the preconditioned gradient method with $P = D$, starting from the initial vector $\boldsymbol{x}^0 = \mathbf{0}$ and with a tolerance of 10^{-6} on the relative residual. We use the following Matlab commands :

```
>> n=25; h=1/(n+1);
```

[4]In the tridiagonal case this is actually possible: the celebrated *Thomas algorithm* is a direct method for tridiagonal systems with $O(n)$ complexity - see section 3.3 of Süli and Mayers for details.

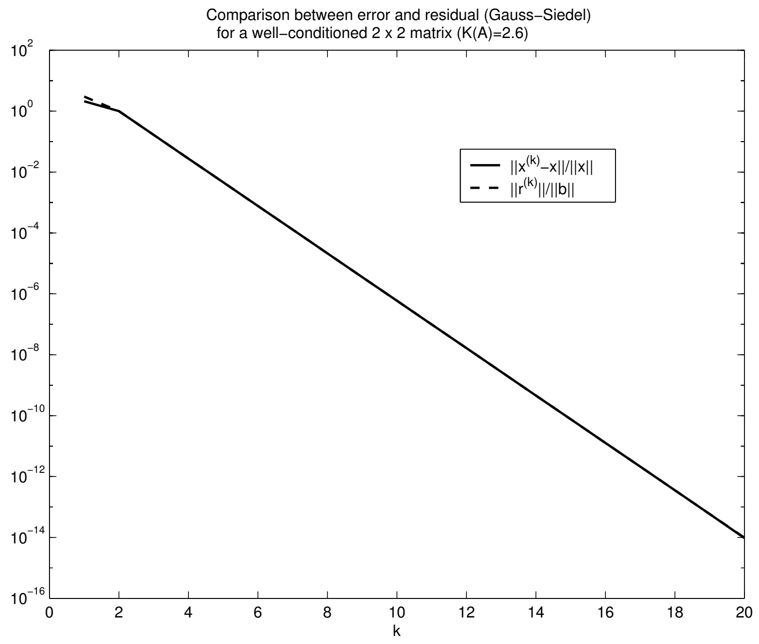


Figure 4.10: *Comparison between relative error and the residual (Gauss-Seidel) for a well-conditioned matrix.*

```
>> A = (2/h)*diag(ones(n,1)) - (1/h)*diag(ones(n-1,1),1) ...
       - (1/h)*diag(ones(n-1,1),-1);
>> b = h*ones(n,1);
>> [x_gs,iter_gs] = itermeth(A,b,zeros(25,1),5000,1e-6,'G');
>> [x_grad,iter_grad] = ...
           gradient(A,b,zeros(n,1),5000,1e-6,diag(diag(A)));
```

The number of iterations required by the two methods is as follows:

```
>> iter_gs
iter_gs =
   940
>> iter_grad
iter_grad =
        1896
```

By contrast, for this problem the preconditioned conjugate gradient method with $P = D$ requires just 13 iterations!

Figure 4.11 shows how the number of iterations needed by the three methods grows as a function of the size n of the matrix. These result show the superior performance of the conjugate gradient method for this ill-conditioned problem.

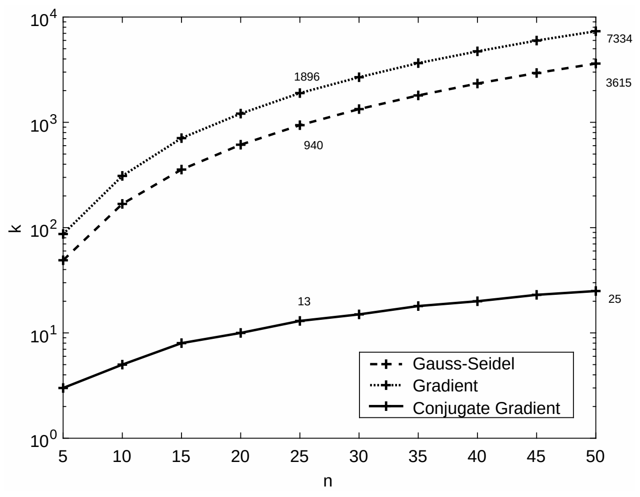


Figure 4.11: *Number of iterations as a function of matrix size n, to reach the tolerance 10^{-6} on the residual.*

Example 4.12.3. *(***4.5.2 continued***)*

Now we revisit Example 4.5.2 and solve a linear system involving the Hilbert matrix of size n using the preconditioned gradient method with a diagonal preconditioner, an initial vector $\boldsymbol{x}_0 = \mathbf{0}$, and a tolerance of 10^{-6} on the relative residual.

The computation was performed using the following Matlab commands.

```
>> for j=2:7;
     n=2*j; i=j-1; nn(i)=n;
     A=hilb(n);
     x_ex=ones(n,1); b=A*x_ex;
     Acond(i)=cond(A);
     tol=1.e-6; maxit=10000;
     R=diag(diag(A));
     x0=zeros(n,1);
     [x,iter_gr(i)]=gradient(A,b,x0,maxit,tol,R);
     error_gr(i)=norm(x-x_ex)/norm(x_ex);
   end
>> semilogy(nn,Acond,'-',nn,error_gr,':')
```

The resulting relative errors for increasing values of n were already reported in Figure 4.3, where we compared them with those obtained using a direct solver (LU-factorization). For ease of reference we also summarize the computational data in the following table. Note that since the matrix is very ill-conditioned the error in the gradient method is significantly bigger than the tolerance 10^{-6}. However, importantly it remains bounded with respect to increasing n, instead of growing rapidly like for the direct solver.

		LU	*Gradient method*		
n	$K(A)$	*Error*	*Error*	*Iterations*	*Residual*
4	*1.55e+04*	*1.94e-13*	*8.72e-03*	*995*	*1.00e-06*
6	*1.50e+07*	*1.18e-10*	*3.60e-03*	*1813*	*9.99e-07*
8	*1.53e+10*	*5.23e-08*	*6.30e-03*	*1089*	*9.96e-07*
10	*1.60e+13*	*2.38e-04*	*7.99e-03*	*875*	*9.99e-07*
12	*1.67e+16*	*1.41e-01*	*5.09e-03*	*1355*	*9.99e-07*
14	*2.04e+17*	*9.82e-01*	*3.91e-03*	*1379*	*9.98e-07*

Example 4.12.4. *(***4.5.1 continued***)*

Finally, we again revisit the problem of the deflected string from Example 4.5.1 and consider discretizations with increasing number of degrees of freedom. This results in linear systems that have increasing size and increasing condition number. Indeed one can show that the condition number of the problem scales as n^2. In the table below we compare the results from computing the solution of the linear system using a direct method (this time the "Cholesky method") and an iterative method (the conjugate gradient method preconditioned using an "incomplete Cholesky factorization"). While these methods have not been described in this course, the purpose of this example is just to show how the computational cost of an iterative method can be significantly lower than that of a direct method.

The first two columns in the table give the size n and the sparsity of the matrix measured as m/n^2 where m denotes the number of non-zero elements in the matrix. Then we report the number of operations and memory needed for the two methods. We see that for the larger systems there is a clear advantage in using the iterative method.

		Cholesky		*Conjugate Gradient*		*flops(Chol.)/*	*Mem(Chol.)/*
n	m/n^2	*flops*	*Memory*	*flops*	*Memory*	*flops(GC)*	*Mem(GC)*
47	*0.12*	*8.05e+03*	*464*	*1.26e+04*	*228*	*0.64*	*2.04*
83	*0.07*	*3.96e+04*	*1406*	*3.03e+04*	*533*	*1.31*	*2.64*
150	*0.04*	*2.01e+05*	*4235*	*8.86e+04*	*1245*	*2.26*	*3.4*
225	*0.03*	*6.39e+05*	*9260*	*1.95e+05*	*2073*	*3.27*	*4.47*
329	*0.02*	*1.74e+06*	*17974*	*3.39e+05*	*3330*	*5.15*	*5.39*
424	*0.02*	*3.78e+06*	*30815*	*5.49e+05*	*4513*	*6.88*	*6.83*
530	*0.01*	*8.31e+06*	*50785*	*8.61e+05*	*5981*	*9.65*	*8.49*
661	*0.01*	*1.19e+07*	*68468*	*1.11e+06*	*7421*	*10.66*	*9.23*