Chapter 4

Solution of linear systems of equation

4.1 Conditioning 83
4.2 Direct solution method 87
4.2.1 LU decomposition 88
4.2.2 Backward and forward substitution 92
4.2.3 Gaussian elimination with pivoting 93
4.2.4 Direct method for Hermitian positive definite matrices 96
4.2.5 Direct methods for banded matrices 97
4.3 Iterative methods for linear systems 100
4.3.1 Basic iterative methods 100
4.3.2 The conjugate gradient method 109
4.4 Exercises 120
4.5 Discussion and bibliography 129

Introduction

This chapter is devoted to the numerical solution of linear problems of the following form:

$$\text{Find } \boldsymbol{x} \in \mathbf{C}^n \text{ such that} \qquad \mathsf{A}\boldsymbol{x} = \boldsymbol{b}, \qquad \mathsf{A} \in \mathbf{C}^{n\times n}, \qquad \boldsymbol{b} \in \mathbf{C}^n. \tag{4.1}$$

Systems of this type appear in a variety of applications. They naturally arise in the context of linear partial differential equations, which we use as main motivating example. Partial differential equations govern a wide range of physical phenomena including heat propagation, gravity, and electromagnetism, to mention just a few. Linear systems in this context often have a particular structure: the matrix A is generally very sparse, which means that most of the entries are equal to 0, and it is often Hermitian and positive definite, provided that these properties are satisfied by the underlying operator.

There are two main approaches for solving linear systems:

- Direct methods enable to calculate the exact solution to systems of linear equations, up to round-off errors, in a finite number of steps. Although this is an attractive property, direct methods are usually too computationally costly for large systems: The cost of inverting a general $n \times n$ matrix, measured in number of floating operations, scales as n^3!
- Iterative methods, on the other hand, enable to progressively calculate increasingly accurate approximations of the solution. Iterations may be stopped once the *the residual* is sufficiently small. These methods are often preferable when the dimension n of the linear system is very large.

This chapter is organized as follows.

- In Section 4.1, we introduce the concept of *conditioning*. The condition number of a matrix provides information on the sensitivity of the solution to perturbations of the right-hand side $\boldsymbol{b}$ or matrix A. It is useful, for example, in order to determine the potential impact of round-off errors.
- In Section 4.2, we present the direct method for solving systems of linear equations. We study in particular the LU decomposition for an invertible matrix, as well as its variant for symmetric positive definite matrices, which is called the Cholesky decomposition.
- In Section 4.3, we present iterative methods for solving linear systems. We focus, in particular, on basic iterative methods based on a splitting of the matrix A and on the conjugate gradient method.

4.1 Conditioning

Before studying the properties of numerical methods for solving linear system, it is crucial to understand the impact of *round-off errors* on the solution, which impose a bound on the accuracy we can hope to achieve. We begin with a motivating example.

Example 4.1. Suppose that we wish to solve the following equation

$$\mathsf{A}\boldsymbol{x} := \begin{pmatrix} 1 & 1 \\ 1 & 1 - 10^{-12} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 10^{-12} \end{pmatrix} =: \boldsymbol{b}.$$

The exact solution is given by $(1 \ -1)^T$. In Julia, this equation can be solved as follows:

```
A = [1 1; 1 (1-1e-12)]
b = [0; 1e-12]
x = A\b
```

The solution returned by the program is the following:

```
  1.0000221222095027
 -1.0000221222095027
```

The relative error on the solution is of the order of 10^{-5}, which is about 12 orders of magnitude larger than the machine epsilon for the `Float64` type. In order to understand this, note that the final error on `x` arises from three sources:

- First, the machine representation `b` of the right-hand side is only *an approximation* of the true right-hand side $\boldsymbol{b}$.
- Similarly, the machine representation `A` of the matrix is only *an approximation* of the true matrix A.
- Finally, the operation `A\b` itself leads to additional round-off errors, as the computer implementation of the backslash operator is based on elementary arithmetic operations. Understanding the magnitude of the error introduced at this step is delicate, and we shall not address this question.

It follows from the first two items that, when writing `A\b`, we are in fact asking the computer for a solution $\boldsymbol{x} + \Delta\boldsymbol{x}$ to a perturbed equation

$$(\mathsf{A} + \Delta\mathsf{A})(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{b} + \Delta\boldsymbol{b},$$

where $\Delta\mathsf{A}$ and $\Delta\boldsymbol{b}$ represent the rounding errors on the matrix and right-hand side, respectively. Understanding the impact of the perturbations $\Delta\boldsymbol{b}$ and $\Delta\mathsf{A}$ is the goal of this section. We shall prove, in particular, that the magnitude of the error $\Delta\boldsymbol{x}$ is related to the perturbations $\Delta\boldsymbol{b}$ and $\Delta\mathsf{A}$ through a quantity known as the *condition number* of the matrix A.

In general, the condition number for a given problem measures the sensitivity of the solution to the input data. In order to define this concept precisely, we consider a general problem of the form $F(x, d) = 0$, with unknown x and data d. The linear system (4.1) can be recast in this form, with the input data equal to $\boldsymbol{b}$ or A or both. We denote the solution corresponding to perturbed input data $d + \Delta d$ by $x + \Delta x$. The absolute and relative condition numbers are defined as follows.

Definition 4.1 (Condition number for the problem $F(x, d) = 0$)**.** The absolute and relative condition numbers with respect to perturbations of d are defined as

$$K_{\text{abs}}(d) = \lim_{\varepsilon \to 0} \left(\sup_{\|\Delta d\| \leqslant \varepsilon} \frac{\|\Delta x\|}{\|\Delta d\|} \right), \qquad K(d) = \lim_{\varepsilon \to 0} \left(\sup_{\|\Delta d\| \leqslant \varepsilon} \frac{\|\Delta x\|/\|x\|}{\|\Delta d\|/\|d\|} \right).$$

The short notation K is reserved for the relative condition number, which is often more useful in applications.

In the rest of this section, we obtain an upper bound on the relative condition number for the linear system (4.1) with respect to perturbations first of $\boldsymbol{b}$, and then of A. We use the notation $\|\bullet\|$ to denote both a vector norm on $\mathbf{C}^n$ and the induced operator norm on matrices.

Proposition 4.1 (Perturbation of the right-hand side)**.** *Let $\boldsymbol{x} + \Delta\boldsymbol{x}$ denote the solution to the perturbed equation $\mathsf{A}(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{b} + \Delta\boldsymbol{b}$. Then it holds that*

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \|\mathsf{A}\|\|\mathsf{A}^{-1}\| \frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}, \tag{4.2}$$

Proof. It holds by definition of $\Delta\boldsymbol{x}$ that $\mathsf{A}\Delta\boldsymbol{x} = \Delta\boldsymbol{b}$. Therefore, we have

$$\|\Delta\boldsymbol{x}\| = \|\mathsf{A}^{-1}\Delta\boldsymbol{b}\| \leqslant \|\mathsf{A}^{-1}\|\|\Delta\boldsymbol{b}\| = \frac{\|\mathsf{A}\boldsymbol{x}\|}{\|\boldsymbol{b}\|}\|\mathsf{A}^{-1}\|\|\Delta\boldsymbol{b}\| \leqslant \frac{\|\mathsf{A}\|\|\boldsymbol{x}\|}{\|\boldsymbol{b}\|}\|\mathsf{A}^{-1}\|\|\Delta\boldsymbol{b}\|. \tag{4.3}$$

Here we employed (A.11), proved in Appendix A, in the first and last inequalities. Rearranging the inequality (4.3), we obtain (4.2). □

Proposition 4.1 implies that the relative condition number of (4.1) with respect to perturbations of the right-hand side is bounded from above by $\|\mathsf{A}\|\|\mathsf{A}^{-1}\|$. Exercise 4.1 shows that there are values of $\boldsymbol{x}$ and $\Delta\boldsymbol{b}$ for which the inequality (4.2) is an equality, indicating that the inequality is sharp.

Studying the impact of perturbations of the matrix A is slightly more difficult, because this time the variation $\Delta\boldsymbol{x}$ of the solution does not depend linearly on the perturbation of the data. Before stating and proving the main result, we show an ancillary lemma.

Lemma 4.2. *Let $\mathsf{B} \in \mathbf{C}^{n\times n}$ be such that $\|\mathsf{B}\| < 1$ in some submultiplicative matrix norm $\|\bullet\|$. Then $\mathsf{I} - \mathsf{B}$ is invertible and*

$$\|(\mathsf{I} - \mathsf{B})^{-1}\| \leqslant \frac{1}{1 - \|\mathsf{B}\|}, \tag{4.4}$$

where $\mathsf{I} \in \mathbf{C}^{n\times n}$ is the identity matrix.

Proof. It holds for any matrix $\mathsf{B} \in \mathbf{C}^{n\times n}$ that

$$\mathsf{I} - \mathsf{B}^{n+1} = (\mathsf{I} - \mathsf{B})(\mathsf{I} + \mathsf{B} + \cdots + \mathsf{B}^n). \tag{4.5}$$

Since $\|\mathsf{B}\| < 1$ in a submultiplicative matrix norm, both sides of the equation are convergent in the limit as $n \to \infty$. The left-hand side converges to the identity matrix I, and the right-hand side converges as $n \to \infty$ because $\{\mathsf{S}_0, \mathsf{S}_1, \dots\}$ with

$$\mathsf{S}_n := \mathsf{I} + \mathsf{B} + \cdots + \mathsf{B}^n$$

is a Cauchy sequence in the vector space of matrices endowed with the norm for which $\|\mathsf{B}\| < 1$. Indeed, by the triangle inequality and the submultiplicative property of the norm, it holds that

$$\begin{aligned}\|\mathsf{S}_{n+m} - \mathsf{S}_n\| &= \|\mathsf{B}^{n+1} + \cdots + \mathsf{B}^{n+m}\| \\ &\leqslant \|\mathsf{B}^{n+1}\| + \cdots + \|\mathsf{B}^{n+m}\| \leqslant \|\mathsf{B}\|^{n+1} + \cdots + \|\mathsf{B}\|^{n+m} \\ &\leqslant \frac{\|\mathsf{B}\|^{n+1}}{1 - \|\mathsf{B}\|} \xrightarrow[n\to\infty]{} 0,\end{aligned}$$

where we employed the formula for a geometric series in the last inequality. Equating the limits of both sides of (4.5), we obtain

$$\mathsf{I} = (\mathsf{I} - \mathsf{B}) \sum_{i=0}^{\infty} \mathsf{B}^i.$$

This implies that $(\mathsf{I} - \mathsf{B})$ is invertible with inverse given by a so-called *Neumann* series

$$(\mathsf{I} - \mathsf{B})^{-1} = \sum_{i=0}^{\infty} \mathsf{B}^i.$$

Applying the triangle inequality repeatedly, and then using the submultiplicative property of the norm, we obtain

$$\forall n \in \mathbf{N}, \qquad \left\| \sum_{i=0}^{n} \mathsf{B}^i \right\| \leqslant \sum_{i=0}^{n} \|\mathsf{B}^i\| \leqslant \sum_{i=0}^{n} \|\mathsf{B}\|^i = \frac{1}{1 - \|\mathsf{B}\|}.$$

where we used the summation formula for geometric series in the last equality. Letting $n \to \infty$ in this equation and using the continuity of the norm enables to conclude the proof. □

Proposition 4.3 (Perturbation of the matrix)**.** *Let $\boldsymbol{x} + \Delta\boldsymbol{x}$ denote the solution to the perturbed equation $(\mathsf{A} + \Delta\mathsf{A})(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{b}$. If A is invertible and $\|\Delta\mathsf{A}\| < \|\mathsf{A}^{-1}\|^{-1}$, then*

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \|\mathsf{A}\|\|\mathsf{A}^{-1}\| \frac{\|\Delta\mathsf{A}\|}{\|\mathsf{A}\|} \left(\frac{1}{1 - \|\mathsf{A}^{-1}\Delta\mathsf{A}\|} \right). \tag{4.6}$$

Proof. Left-multiplying both sides of the perturbed equation with A^{-1}, we obtain

$$(\mathsf{I} + \mathsf{A}^{-1}\Delta\mathsf{A})(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{x} \quad \Leftrightarrow \quad (\mathsf{I} + \mathsf{A}^{-1}\Delta\mathsf{A})\Delta\boldsymbol{x} = -\mathsf{A}^{-1}\Delta\mathsf{A}\boldsymbol{x}. \tag{4.7}$$

Since $\|\mathsf{A}^{-1}\Delta\mathsf{A}\| \leqslant \|\mathsf{A}^{-1}\|\|\Delta\mathsf{A}\| < 1$ by assumption, we deduce from Lemma 4.2 that the matrix on the left-hand side is invertible with a norm bounded as in (4.4). Consequently, using in addition the assumed submultiplicative property of the norm, we obtain that

$$\|\Delta\boldsymbol{x}\| = \|(\mathsf{I} + \mathsf{A}^{-1}\Delta\mathsf{A})^{-1}\mathsf{A}^{-1}\Delta\mathsf{A}\boldsymbol{x}\| \leqslant \frac{\|\mathsf{A}^{-1}\Delta\mathsf{A}\|}{1 - \|\mathsf{A}^{-1}\Delta\mathsf{A}\|}\|\boldsymbol{x}\|.$$

which enables to conclude the proof. □

Using Proposition 4.3, we deduce that the relative condition number of (4.1) with respect to perturbations of the matrix A is also bounded from above by $\|\mathsf{A}\|\|\mathsf{A}^{-1}\|$, because the term between brackets on the right-hand side of (4.6) converges to 1 as $\|\Delta\mathsf{A}\| \to 0$.

Propositions 4.1 and 4.3 show that the condition number of the linear system (4.1), with respect to perturbations of either $\boldsymbol{b}$ or A, depends only on A. This motivates the following definition of the condition number *of a matrix*.

Definition 4.2 (Condition number of a matrix)**.** The condition number of a matrix A associated with a vector norm $\|\bullet\|$ is defined as

$$\kappa(\mathsf{A}) = \|\mathsf{A}\|\|\mathsf{A}^{-1}\|.$$

The condition number for the p-norm, defined in Definition A.4, is denoted by $\kappa_p(\mathsf{A})$.

Note that the condition number $\kappa(\mathsf{A})$ associated with an induced norm, i.e. a matrix norm induced by a vector norm, is at least one. Indeed, since the identity matrix has induced norm 1, it holds that

$$1 = \|\mathsf{I}\| = \|\mathsf{A}\mathsf{A}^{-1}\| \leqslant \|\mathsf{A}\|\|\mathsf{A}^{-1}\|.$$

Since the 2-norm of an invertible matrix $\mathsf{A} \in \mathbf{C}^{n\times n}$ coincides with the spectral radius $\rho(\mathsf{A}^T\mathsf{A})$, the condition number κ_2 corresponding to the 2-norm is equal to

$$\kappa_2(\mathsf{A}) = \sqrt{\frac{\lambda_{\max}(\mathsf{A}^T\mathsf{A})}{\lambda_{\min}(\mathsf{A}^T\mathsf{A})}},$$

where $\lambda_{\max}(\mathsf{A}^T\mathsf{A})$ and $\lambda_{\min}(\mathsf{A}^T\mathsf{A})$ are the maximal and minimal (both real and positive) eigenvalues of the matrix $\mathsf{A}^T\mathsf{A}$.

Example 4.2 (Perturbation of the matrix). Consider the following linear system with perturbed matrix

$$(\mathsf{A}+\Delta\mathsf{A})\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{pmatrix}0\\0.01\end{pmatrix}, \qquad \mathsf{A} = \begin{pmatrix}1 & 0\\0 & 0.01\end{pmatrix}, \qquad \Delta\mathsf{A} = \begin{pmatrix}0 & 0\\0 & \varepsilon\end{pmatrix},$$

where $0 < \varepsilon \ll 0.01$. Here the eigenvalues of A are given by $\lambda_1 = 1$ and $\lambda_2 = 0.01$. The solution when $\varepsilon = 0$ is given by $(0,1)^T$, and the solution to the perturbed equation is

$$\begin{pmatrix}x_1+\Delta x_1\\x_2+\Delta x_2\end{pmatrix} = \begin{pmatrix}0\\\frac{1}{1+100\varepsilon}\end{pmatrix}.$$

Consequently, we deduce that, in the 2-norm,

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \left|\frac{100\varepsilon}{1+100\varepsilon}\right| \approx 100\varepsilon = 100\frac{\|\Delta\mathsf{A}\|}{\|\mathsf{A}\|}.$$

In this case, the relative impact of perturbations of the matrix is close to $\kappa_2(\mathsf{A}) = 100$.

4.2 Direct solution method

In this section, we present the *direct method* for solving linear systems of the form (4.1) with a general invertible matrix $\mathsf{A} \in \mathbf{C}^{n\times n}$. The direct method can be decomposed into three steps:

- First calculate the so-called LU decomposition of A, i.e. find an upper triangular matrix U and a *unit* lower triangular matrix L such that $\mathsf{A} = \mathsf{LU}$. A unit lower triangular matrix is a lower triangular matrix with only ones on the diagonal.
- Then solve $\mathsf{L}\boldsymbol{y} = \boldsymbol{b}$ using a method called *forward substitution*.
- Finally, solve $\mathsf{U}\boldsymbol{x} = \boldsymbol{y}$ using a method called *backward substitution*.

By construction, the solution $\boldsymbol{x}$ thus obtained is a solution to (4.1). Indeed, we have that

$$\mathsf{A}\boldsymbol{x} = \mathsf{LU}\boldsymbol{x} = \mathsf{L}\boldsymbol{y} = \boldsymbol{b}.$$

4.2.1 LU decomposition

In this section, we first discuss the existence and uniqueness of the LU factorization of a matrix. We then describe a numerical algorithm for calculating the factors L and U, based on *Gaussian elimination*.

Existence and uniqueness of the decomposition

We present a necessary and sufficient condition for the existence of a unique LU decomposition of a matrix. To this end, we define the principal submatrix of order i of a matrix $\mathsf{A} \in \mathbf{C}^{n\times n}$ as the matrix $\mathsf{A}_i = \mathsf{A}[1:i, 1:i]$, in Julia notation.

Proposition 4.4. *The LU factorization of a matrix $\mathsf{A} \in \mathbf{C}^{n\times n}$, with L unit lower triangular and U upper triangular, exists and is unique if and only if the principal submatrices of A of all orders are nonsingular.*

Proof. We prove only the "if" direction; see [10, Theorem 3.4] for the "only if" implication.

The statement is clear if $n = 1$. Reasoning by induction, we assume that the result is proved up to $n-1$. Since the matrix A_{n-1} and all its principal submatrices are nonsingular by assumption, it holds that $\mathsf{A}_{n-1} = \mathsf{L}_{n-1}\mathsf{U}_{n-1}$ for a unit lower triangular matrix L_{n-1} and an upper triangular matrix U_{n-1}. These two matrices are nonsingular, for if either of them were singular then the product $\mathsf{A}_{n-1} = \mathsf{L}_{n-1}\mathsf{U}_{n-1}$ would be singular as well. Let us decompose A as follows:

$$\mathsf{A} = \begin{pmatrix} \mathsf{A}_{n-1} & \boldsymbol{c} \\ \boldsymbol{d}^T & a_{nn} \end{pmatrix}.$$

Let ℓ and $\boldsymbol{u}$ denote the solutions to $\mathsf{L}_{n-1}\boldsymbol{u} = \boldsymbol{c}$ and $\mathsf{U}_{n-1}^T\ell = \boldsymbol{d}$. These solutions exist and are unique, because the matrices L_{n-1} and U_{n-1} are nonsingular. Letting $u_{nn} = a_{nn} - (\ell^T\boldsymbol{u})^{-1}$, we check that A factorizes as

$$\begin{pmatrix} \mathsf{A}_{n-1} & \boldsymbol{c} \\ \boldsymbol{d}^T & a_{nn} \end{pmatrix} = \begin{pmatrix} \mathsf{L}_{n-1} & \mathbf{0}_{n-1} \\ \ell^T & 1 \end{pmatrix} \begin{pmatrix} \mathsf{U}_{n-1} & \boldsymbol{u} \\ \mathbf{0}_{n-1}^T & u_{nn} \end{pmatrix}.$$

This completes the proof of the existence of the decomposition. The uniqueness of the factors follows from the uniqueness of ℓ, $\boldsymbol{u}$ and u_{nn}. □

Proposition 4.4 raises the following question: are there classes of matrices whose principal matrices are all nonsingular? The answer is positive, and we mention, as an important example, the class of positive definite matrices. Proving this is the aim of Exercise 4.4.

Gaussian elimination algorithm for computing L and U

So far we have presented a condition under which the LU decomposition of a matrix exists and is unique, but not a practical method for calculating the matrices L and U. We describe in this section an algorithm, known as *Gaussian elimination*, for calculating the LU decomposition of a matrix. We begin by introducing the concept of *Gaussian transformation*.

Definition 4.3. A Gaussian transformation is a matrix of the form $\mathsf{M}_k = \mathsf{I} - \boldsymbol{c}^{(k)} \boldsymbol{e}_k^T$, where $\boldsymbol{e}_k$ is the column vector with entry at index k equal to 1 and all the other entries equal to zero, and $\boldsymbol{c}^{(k)}$ is a column vector of the following form:

$$\boldsymbol{c}^{(k)} = \begin{pmatrix} 0 & 0 & \dots & 0 & c_{k+1}^{(k)} & c_{k+2}^{(k)} & \dots & c_n^{(k)} \end{pmatrix}^T.$$

The action of a Gaussian transformation M_k left-multiplying a matrix $\mathsf{A} \in \mathbf{C}^{n \times n}$ is to replace the rows from index $k+1$ to index n by a linear combination involving themselves and the k-th row. To see this, let us denote by $(\boldsymbol{r}^{(i)})_{1 \leqslant i \leqslant n}$ the rows of a matrix $\mathsf{T} \in \mathbf{C}^{n \times n}$. Then, we have

$$\mathsf{M}_k \mathsf{T} = \left(\mathsf{I} - \boldsymbol{c}^{(k)} \boldsymbol{e}_k^T\right) \mathsf{T} = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & -c_{k+1}^{(k)} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & -c_n^{(k)} & & & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{r}^{(1)} \\ \boldsymbol{r}^{(2)} \\ \vdots \\ \boldsymbol{r}^{(k)} \\ \boldsymbol{r}^{(k+1)} \\ \vdots \\ \boldsymbol{r}^{(n)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}^{(1)} \\ \boldsymbol{r}^{(2)} \\ \vdots \\ \boldsymbol{r}^{(k)} \\ \boldsymbol{r}^{(k+1)} - c_{k+1}^{(k)} \boldsymbol{r}^{(k)} \\ \vdots \\ \boldsymbol{r}^{(n)} - c_n^{(k)} \boldsymbol{r}^{(k)} \end{pmatrix}$$

We show in Exercise 4.2 that the inverse of a Gaussian transformation matrix is given by

$$(\mathsf{I} - \boldsymbol{c}^{(k)} \boldsymbol{e}_k^T)^{-1} = \mathsf{I} + \boldsymbol{c}^{(k)} \boldsymbol{e}_k^T. \tag{4.8}$$

The idea of the Gaussian elimination algorithm is to successively left-multiply A with Gaussian transformation matrices M_1, then M_2, etc. appropriately chosen in such a way that the matrix $\mathsf{A}^{(k)}$, obtained after k iterations, is upper triangular up to column k. That is to say, the Gaussian transformations are constructed so that all the entries in columns 1 to k under the diagonal of the matrix $\mathsf{A}^{(k)}$ are equal to zero. The resulting matrix $\mathsf{A}^{(n-1)}$ after $n-1$ iterations is then upper triangular and satisfies

$$\mathsf{A}^{(n-1)} = \mathsf{M}_{n-1} \dots \mathsf{M}_1 \mathsf{A}.$$

Rearranging this equation, we deduce that

$$\mathsf{A} = (\mathsf{M}_1^{-1} \dots \mathsf{M}_{n-1}^{-1})\mathsf{A}^{(n-1)}.$$

The first factor is lower triangular by (4.8) and Exercise 4.3. The product in the definition of the matrix L admits a simple explicit expression.

Lemma 4.5. *It holds that*

$$\mathsf{M}_1^{-1} \cdots \mathsf{M}_{n-1}^{-1} = (\mathsf{I} + \boldsymbol{c}^{(1)}\boldsymbol{e}_1^T) \cdots (\mathsf{I} + \boldsymbol{c}^{(n-1)}\boldsymbol{e}_{n-1}^T) = \mathsf{I} + \sum_{i=1}^{n-1} \boldsymbol{c}^{(i)}\boldsymbol{e}_i^T.$$

Proof. Notice that, for $i < j$,

$$\boldsymbol{c}^{(i)}\boldsymbol{e}_i^T\boldsymbol{c}^{(j)}\boldsymbol{e}_j^T = \boldsymbol{c}^{(i)}(\boldsymbol{e}_i^T\boldsymbol{c}^{(j)})\boldsymbol{e}_j^T = \boldsymbol{c}^{(i)}0\boldsymbol{e}_j^T = 0.$$

The statement then follows easily by expanding the product. □

A corollary of Lemma 4.5 is that all the diagonal entries of the lower triangular matrix L are equal to 1; the matrix L is *unit lower triangular*. The full expression of the matrix L given the Gaussian transformations is

$$\mathsf{L} = \mathsf{I} + \begin{pmatrix} \boldsymbol{c}^{(1)} & \dots & \boldsymbol{c}^{(n-1)} & \mathbf{0}_n \end{pmatrix} = \begin{pmatrix} 1 & & & & & \\ c_2^{(1)} & 1 & & & & \\ c_3^{(1)} & c_3^{(2)} & 1 & & & \\ c_4^{(1)} & c_4^{(2)} & c_4^{(3)} & 1 & & \\ \vdots & \vdots & \vdots & & \ddots & \\ c_n^{(1)} & c_n^{(2)} & c_n^{(3)} & \dots & c_n^{(n-1)} & 1 \end{pmatrix} \tag{4.9}$$

Therefore, the Gaussian elimination algorithms, if all the steps are well-defined, correctly gives the LU factorization of the matrix A. Of course, the success of the strategy outlined above for the calculation of the LU factorization hinges on the existence of an appropriate Gaussian transformation at each iteration. It is not difficult to show that, if the $(k+1)$-th diagonal entry of the matrix $\mathsf{A}^{(k)}$ is nonzero for all $k \in \{1, \dots, n-2\}$, then the Gaussian transformation matrices exist and are uniquely defined.

Lemma 4.6. *Assume that $\mathsf{A}^{(k)}$ is upper triangular up to column k included, with $k \leqslant n-2$. If $a_{k+1,k+1}^{(k)} > 0$, then there is a unique Gaussian transformation matrix M_{k+1} such that $\mathsf{M}_{k+1}\mathsf{A}^{(k)}$ is upper triangular up to column $k+1$. This transformation matrix is given by $\mathsf{I} - \boldsymbol{c}^{(k+1)}\boldsymbol{e}_{k+1}^T$, where*

$$\boldsymbol{c}^{(k+1)} = \begin{pmatrix} 0 & 0 & \dots & 0 & \frac{a_{k+2,k+1}^{(k)}}{a_{k+1,k+1}^{(k)}} & \frac{a_{k+3,k+1}^{(k)}}{a_{k+1,k+1}^{(k)}} & \dots & \frac{a_{n,k+1}^{(k)}}{a_{k+1,k+1}^{(k)}} \end{pmatrix}^T.$$

Proof. We perform the multiplication explicitly. Denoting denote by $(\boldsymbol{r}^{(i)})_{1 \leqslant i \leqslant n}$ the rows of $\mathsf{A}^{(k)}$, we have

$$\mathsf{M}_{k+1}\mathsf{A}^{(k)} = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & -c_{k+2}^{(k+1)} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & -c_{n}^{(k+1)} & & & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{r}^{(1)} \\ \boldsymbol{r}^{(2)} \\ \vdots \\ \boldsymbol{r}^{(k+1)} \\ \boldsymbol{r}^{(k+2)} \\ \vdots \\ \boldsymbol{r}^{(n)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}^{(1)} \\ \boldsymbol{r}^{(2)} \\ \vdots \\ \boldsymbol{r}^{(k+1)} \\ \boldsymbol{r}^{(k+2)} - c_{k+2}^{(k+1)}\boldsymbol{r}^{(k+1)} \\ \vdots \\ \boldsymbol{r}^{(n)} - c_{n}^{(k+1)}\boldsymbol{r}^{(k+1)} \end{pmatrix}.$$

We need to show that the matrix on the right-hand side is upper triangular up to column $k+1$ included. This is clear by definition of $\boldsymbol{c}^{(k+1)}$ and from the fact that $\mathsf{A}^{(k)}$ is upper triangular up to column k by assumption. □

The diagonal elements $a_{k+1,k+1}^{(k)}$, where $k \in \{0, \dots, n-2\}$, are called the pivots. We now prove that, if an invertible matrix A admits an LU factorization, then the pivots are necessarily nonzero and the Gaussian elimination algorithm is successful.

Proposition 4.7 (Gaussian elimination works). *If A is invertible and admits an LU factorization, then the Gaussian elimination algorithm is well-defined and successfully terminates.*

Proof. We denote by $\boldsymbol{c}^{(1)}, \dots, \boldsymbol{c}^{(n-1)}$, the columns of the matrix $\mathsf{L} - \mathsf{I}$. Then the matrices given by $\mathsf{M}_k = \mathsf{I} - \boldsymbol{c}^{(k)}\boldsymbol{e}_k^T$, for $k \in \{1, \dots, n-1\}$, are Gaussian transformations and it holds that

$$\mathsf{L} = \mathsf{M}_1^{-1} \cdots \mathsf{M}_{n-1}^{-1}$$

in view of Lemma 4.5. Since $\mathsf{A} = \mathsf{LU}$ by assumption, the result of the product

$$\mathsf{M}_{n-1} \cdots \mathsf{M}_1 \mathsf{A} = \mathsf{U}$$

is upper triangular. Let us use the notation $\mathsf{A}^{(k)} = \mathsf{M}_k \cdots \mathsf{M}_1 \mathsf{A}$. Our goal is to show that the matrices $\mathsf{M}_1, \dots, \mathsf{M}_{n-1}$ are the same as those obtained by the Gaussian elimination algorithm.

Of all the Gaussian transformations $\mathsf{M}_1, \dots, \mathsf{M}_{n-1}$, only M_1 acts on the second row of the matrix it multiplies. Therefore, the entry $(2,1)$ of U coincides with the entry $(2,1)$ of $\mathsf{A}^{(1)}$, which implies that $a_{2,1}^{(1)} = 0$. Then notice that $a_{3,1}^{(k)} = a_{3,1}^{(1)}$ for all $k \geqslant 1$, because the entry $(3,1)$ of the matrix $\mathsf{M}_2\mathsf{A}^{(1)}$ is given by $a_{3,1}^{(1)} - c_3^{(2)} a_{2,1}^{(1)} = a_{3,1}^{(1)}$, and all the other transformation matrices $\mathsf{M}_3, \dots, \mathsf{M}_{n-1}$ leave the third row invariant. Consequently, it holds that $a_{3,1}^{(1)} = u_{3,1} = 0$. Continuing in this manner, we deduce that $\mathsf{A}^{(1)}$ is upper triangular in the first column and that, since A is invertible by assumption, the first pivot a_{11} is nonzero. Since this pivot is nonzero, the matrix M_1 is uniquely defined by Lemma 4.6.

The reasoning can then be repeated with other columns, in order to deduce that $\mathsf{A}^{(k)}$ is upper triangular up to column k and that all the pivots $a_{kk}^{(k-1)}$ are nonzero. Therefore, all the Gaussian transformation matrices are uniquely defined given Lemma 4.6. □

Computer implementation

The Gaussian elimination procedure is summarized as follows.

$\mathsf{A}^{(0)} \leftarrow \mathsf{A}, \mathsf{L} \leftarrow \mathsf{I}$
for $i \in \{1, \ldots, n-1\}$ **do**
  Construct M_i as in Lemma 4.6.
  $\mathsf{A}^{(i)} \leftarrow \mathsf{M}_i \mathsf{A}^{(i-1)}, \mathsf{L} \leftarrow \mathsf{L}\mathsf{M}_i^{-1}$
end for
$\mathsf{U} \leftarrow \mathsf{A}^{(n-1)}$.

In practice, it is not necessary to explicitly create the Gaussian transformation matrices, or to perform full matrix multiplications. A more realistic version of the algorithm in Julia is given below. The code exploits the relation (4.9) between L and the parameters of the Gaussian transformations.

```
1  # A is an invertible matrix of size n x n
2  L = [i == j ? 1.0 : 0.0 for i in 1:n, j in 1:n]
3  U = copy(A)
4  for i in 1:n-1
5      for r in i+1:n
6          U[i, i] == 0 && error("Pivotal entry is zero!")
7          ratio = U[r, i] / U[i, i]
8          L[r, i] = ratio
9          U[r, i:end] -= U[i, i:end] * ratio
10     end
11 end
12 # L is unit lower triangular and U is upper triangular
```

Computational cost

The computational cost of the algorithm, measured as the number of floating point operations (flops) required, is dominated by the Gaussian transformations, in line 9 in the above code. All the other operations amount to a computational cost scaling as $\mathcal{O}(n^2)$, which is negligible compared to the cost of the LU factorization when n is large. This factorization requires

$$\overbrace{2\times}^{\text{- and *}} \sum_{\substack{\overbrace{i=1}^{\texttt{for i in 1:n-1}}}}^{n-1} \overbrace{(n-i)}^{\texttt{for r in i+1:n}} \underbrace{(n-i+1)}_{\texttt{indices [i:end]}} \quad \text{flops} = \frac{2}{3}n^3 + \mathcal{O}(n^2) \quad \text{flops}.$$

4.2.2 Backward and forward substitution

Once the LU factorization has been completed, the solution to the linear system can be obtained by first using forward, and then backward substitution, which are just bespoke methods for solving linear systems with lower and upper triangular matrices, respectively. Let us consider the case of a lower triangular system:

$$\mathsf{L}\boldsymbol{y} = \boldsymbol{b}$$

Notice that the unknown y_1 may be obtained from the first equation of the system. Then, since y_1 is known, the value of y_2 can be obtained from the second equation, etc. A simple implementation of this algorithm is as follows:

```
# L is unit lower triangular
y = copy(b)
for i in 2:n
    for j in 1:i-1
        y[i] -= L[i, j] * y[j]
    end
end
```

4.2.3 Gaussian elimination with pivoting

The Gaussian elimination algorithm that we presented in Section 4.2.1 relies on the existence of an LU factorization. In practice, this assumption may not be satisfied, and in this case a modified algorithm, called Gaussian elimination *with pivoting*, is required.

In fact, pivoting is useful even if the usual LU decomposition of A exists, as it enables to reduce the condition number of the matrices matrices L and U. There are two types of pivoting: partial pivoting, where only the rows are rearranged through a permutation at each iteration, and complete pivoting, where both the rows and the columns are rearranged at each iteration.

Showing rigorously why pivoting is useful is beyond the scope of this course. In this section, we only present the partial pivoting method. Its influence on the condition number of the factors L and U is studied empirically in Exercise 4.6. It is useful at this point to introduce the concept of a row permutation matrix.

Row permutation matrix

Definition 4.4. Let $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ be a permutation, i.e. a bijection on the set $\{1, \ldots, n\}$. The row permutation matrix associated with σ is the matrix with entries

$$p_{ij} = \begin{cases} 1 \text{ if } i = \sigma(j), \\ 0 \text{ otherwise.} \end{cases}$$

When a row permutation P left-multiplies a matrix $\mathsf{B} \in \mathbf{C}^{n \times n}$, row i of matrix B is moved to row index $\sigma(i)$ in the resulting matrix, for all $i \in \{1, \ldots, n\}$. A permutation matrix has a single entry equal to 1 per row and per column, and its inverse coincides with its transpose: $\mathsf{P}^{-1} = \mathsf{P}^T$.

Partial pivoting

Gaussian elimination with partial pivoting applies for any invertible matrix A, and it outputs 3 matrices: a row permutation P, a unit triangular matrix L, and an upper triangular matrix U. These are related by the relation

$$\mathsf{PA} = \mathsf{LU}.$$

This is sometimes called a PLU decomposition of the matrix A. It is not unique in general but, unlike the usual LU decomposition, it always exists provided that A is invertible. We take this for granted in this course.

The idea of partial pivoting is to rearrange the rows at each iteration of the Gaussian elimination procedure in such a manner that the pivotal entry is as large as possible in absolute value. One step of the procedure reads

$$\mathsf{A}^{(k+1)} = \mathsf{M}_{k+1}\mathsf{P}_{k+1}\mathsf{A}^{(k)}. \tag{4.10}$$

Here P_{k+1} is a simple row permutation matrix which, when acting on $\mathsf{A}^{(k)}$, interchanges row $k+1$ and row ℓ, for some index $\ell \geqslant k+1$. The row index ℓ is selected in such a way that the absolute value of the pivotal entry, in position $(k+1, k+1)$ of the product $\mathsf{P}_{k+1}\mathsf{A}^{(k)}$, is maximum. The matrix M_{k+1} is then the unique Gaussian transformation matrix ensuring that $\mathsf{A}^{(k+1)}$ is upper triangular up to column $k+1$, obtained as in Lemma 4.6. The resulting matrix $\mathsf{A}^{(n-1)}$ after $n-1$ steps of the form (4.10) is upper triangular and satisfies

$$\mathsf{A}^{(n-1)} = \mathsf{M}_{n-1}\mathsf{P}_{n-1}\cdots\mathsf{M}_1\mathsf{P}_1\mathsf{A} \quad\Leftrightarrow\quad \mathsf{A} = (\mathsf{M}_{n-1}\mathsf{P}_{n-1}\cdots\mathsf{M}_1\mathsf{P}_1)^{-1}\mathsf{A}^{(n-1)}.$$

The first factor in the decomposition of A is not necessarily lower triangular. However, using the notation $\mathsf{M} = \mathsf{M}_{n-1}\mathsf{P}_{n-1}\cdots\mathsf{M}_1\mathsf{P}_1$ and $\mathsf{P} = \mathsf{P}_{n-1}\cdots\mathsf{P}_1$, we have

$$\mathsf{PA} = \mathsf{PM}^{-1}\mathsf{U} = (\mathsf{PM}^{-1})\mathsf{U} =: \mathsf{LU}. \tag{4.11}$$

Lemma 4.8 below shows that, as the notation L suggests, the matrix $\mathsf{L} = (\mathsf{PM}^{-1})$ on the right-hand side is indeed lower triangular. Before stating and proving the lemma, we note that P is a row permutation matrix, and so the solution to the linear system $\mathsf{A}\boldsymbol{x} = \mathsf{b}$ can be obtained by solving $\mathsf{LU}\boldsymbol{x} = \mathsf{P}^T\boldsymbol{b}$ by forward and backward substitution. Since P is a very sparse matrix, the right-hand side $\mathsf{P}^T\boldsymbol{b}$ can be calculated very efficiently.

Lemma 4.8. *The matrix $\mathsf{L} = \mathsf{PM}^{-1}$ is unit lower triangular with all entries bounded in absolute value from above by 1. It admits the expression*

$$\mathsf{L} = \mathsf{I} + (\mathsf{P}_{n-1}\cdots\mathsf{P}_2\boldsymbol{c}^{(1)})\boldsymbol{e}_1^T + (\mathsf{P}_{n-1}\cdots\mathsf{P}_3\boldsymbol{c}^{(2)})\boldsymbol{e}_2^T + \cdots + (\mathsf{P}_{n-1}\boldsymbol{c}^{(n-2)})\boldsymbol{e}_{n-2}^T + \boldsymbol{c}^{(n-1)}\boldsymbol{e}_{n-1}^T.$$

Proof. Let $\mathsf{M}^{(k)} = \mathsf{M}_k\mathsf{P}_k\cdots\mathsf{M}_1\mathsf{P}_1$ and $\mathsf{P}^{(k)} = \mathsf{P}_k\cdots\mathsf{P}_1$. It is sufficient to show that

$$\mathsf{P}^{(k)}\big(\mathsf{M}^{(k)}\big)^{-1} = \mathsf{I} + (\mathsf{P}_k\cdots\mathsf{P}_2\boldsymbol{c}^{(1)})\boldsymbol{e}_1^T + (\mathsf{P}_k\cdots\mathsf{P}_3\boldsymbol{c}^{(2)})\boldsymbol{e}_2^T + \cdots + (\mathsf{P}_k\boldsymbol{c}^{(k-1)})\boldsymbol{e}_{k-1}^T + \boldsymbol{c}^{(k)}\boldsymbol{e}_k^T \tag{4.12}$$

for all $k \in \{1, \ldots, n-1\}$. The statement is clear for $k = 1$, and we assume by induction that it is true up to $k-1$. Then notice that

$$\begin{aligned}
\mathsf{P}^{(k)}\big(\mathsf{M}^{(k)}\big)^{-1} &= \mathsf{P}_k\Big(\mathsf{P}^{(k-1)}\big(\mathsf{M}^{(k-1)}\big)^{-1}\Big)\mathsf{P}_k^{-1}\mathsf{M}_k^{-1} \\
&= \mathsf{P}_k\Big(\mathsf{I} + (\mathsf{P}_{k-1}\cdots\mathsf{P}_2\boldsymbol{c}^{(1)})\boldsymbol{e}_1^T + \cdots + (\mathsf{P}_{k-1}\boldsymbol{c}^{(k-2)})\boldsymbol{e}_{k-2}^T + \boldsymbol{c}^{(k-1)}\boldsymbol{e}_{k-1}^T\Big)\mathsf{P}_k^{-1}\mathsf{M}_k^{-1} \\
&= \Big(\mathsf{I} + (\mathsf{P}_k\mathsf{P}_{k-1}\cdots\mathsf{P}_2\boldsymbol{c}^{(1)})\boldsymbol{e}_1^T + \cdots + (\mathsf{P}_k\mathsf{P}_{k-1}\boldsymbol{c}^{(k-2)})\boldsymbol{e}_{k-2}^T + (\mathsf{P}_k\boldsymbol{c}^{(k-1)})\boldsymbol{e}_{k-1}^T\Big)\mathsf{M}_k^{-1}.
\end{aligned}$$

In the last equality, we used that $\boldsymbol{e}_i^T \mathsf{P}_k^{-1} = (\mathsf{P}_k \boldsymbol{e}_i)^T = \boldsymbol{e}_i^T$ for all $i \in \{1, \ldots, k-1\}$, because the row permutation P_k does not affect rows 1 to $k-1$. Using the expression $\mathsf{M}_k^{-1} = \mathsf{I} + \boldsymbol{c}^{(k)} \boldsymbol{e}_k^T$, expanding the product and noting that $\boldsymbol{e}_j^T \boldsymbol{c}^{(k)} = 0$ if $j \leqslant k$, we obtain (4.12). The statement that the entries are bounded in absolute value from above by 1 follows from the choice of the pivot at each iteration. □

The expression of L in Lemma 4.8 suggests the iterative procedure given in Algorithm 2 for performing the LU factorization with partial pivoting. A Julia implementation of this algorithm is presented in Listing 1.

Algorithm 2 LU decomposition with partial pivoting

Assign $\mathsf{A}^{(0)} \leftarrow \mathsf{A}$ and $\mathsf{P} \leftarrow \mathsf{I}$
for $i \in \{1, \ldots, n-1\}$ **do**
  Find the row index $k \geqslant i$ such that $\mathsf{A}_{k,i}^{(i-1)}$ is maximum in absolute value.
  Interchange the rows i and k of matrices $\mathsf{A}^{(i-1)}$ and P, and of vectors $\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(i-1)}$.
  Construct M_i with corresponding column vector $\boldsymbol{c}^{(i)}$ as in Lemma 4.6.
  Assign $\mathsf{A}^{(i)} \leftarrow \mathsf{M}_i \mathsf{A}^{(i-1)}$
end for
Assign $\mathsf{U} \leftarrow \mathsf{A}^{(n-1)}$.
Assign $\mathsf{L} \leftarrow \mathsf{I} + \begin{pmatrix} \boldsymbol{c}^{(1)} & \ldots & \boldsymbol{c}^{(n-1)} & \boldsymbol{0}_n \end{pmatrix}$.

```
# Auxiliary function
function swap_rows!(i, j, matrices...)
    for M in matrices
        M_row_i = M[i, :]
        M[i, :] = M[j, :]
        M[j, :] = M_row_i
    end
end

n = size(A)[1]
L, U = zeros(n, 0), copy(A)
P = [i == j ? 1.0 : 0.0 for i in 1:n, j in 1:n]
for i in 1:n-1
    # Pivoting
    index_row_pivot = i - 1 + argmax(abs.(U[i:end, i]))
    swap_rows!(i, index_row_pivot, U, L, P)

    # Usual Gaussian transformation
    c = [zeros(i-1); 1.0; zeros(n-i)]
    for r in i+1:n
        ratio = U[r, i] / U[i, i]
        c[r] = ratio
        U[r, i:end] -= U[i, i:end] * ratio
    end
    L = [L c]
end
L = [L [zeros(n-1); 1.0]]
# It holds that P*A = L*U
```

Listing 1: LU factorization with partial pivoting.

Remark 4.1. It is possible to show that, if the matrix A is column diagonally dominant in the sense that

$$\forall j \in \{1, \ldots, n\}, \qquad |a_{jj}| \geqslant \sum_{i=1, i\neq j}^{n} |a_{ij}|,$$

then pivoting does not have an effect: at each iteration, the best pivot is already on the diagonal.

4.2.4 Direct method for Hermitian positive definite matrices

The LU factorization with partial pivoting applies to any matrix $\mathsf{A} \in \mathbf{C}^{n\times n}$ that is invertible. If A is Hermitian positive definite, however, it is possible to compute a factorization into lower and upper triangular matrices at half the computational cost, using the so-called *Cholesky decomposition*.

Lemma 4.9 (Cholesky decomposition)**.** *If* A *is Hermitian positive definite, then there exists a lower-triangular matrix* $\mathsf{C} \in \mathbf{C}^{n\times n}$ *such that*

$$\mathsf{A} = \mathsf{C}\mathsf{C}^*. \tag{4.13}$$

Equation (4.13) is called the Cholesky factorization of A*. The matrix* C *is unique if we require that all its diagonal entries are positive.*

Proof. Since A is positive definite, its LU decomposition exists and is unique by Proposition 4.4 and Proposition 4.7. Let D denote the diagonal matrix with the same diagonal as that of U. Then

$$\mathsf{A} = \mathsf{L}\mathsf{D}(\mathsf{D}^{-1}\mathsf{U}).$$

Note that the matrix $\mathsf{D}^{-1}\mathsf{U}$ is unit upper triangular. Since A is Hermitian, we have

$$\mathsf{A} = \mathsf{A}^* = (\mathsf{D}^{-1}\mathsf{U})^*(\mathsf{L}\mathsf{D})^*.$$

The first and second factors on the right-hand side are respectively unit lower triangular and upper triangular, and so we deduce, by uniqueness of the LU decomposition, that $\mathsf{L} = (\mathsf{D}^{-1}\mathsf{U})^*$ and $\mathsf{U} = (\mathsf{L}\mathsf{D})^*$. But then

$$\mathsf{A} = \mathsf{L}\mathsf{U} = \mathsf{L}\mathsf{D}\mathsf{L}^* = (\mathsf{L}\sqrt{\mathsf{D}})(\sqrt{\mathsf{D}}\mathsf{L})^*.$$

Here $\sqrt{\mathsf{D}}$ denotes the diagonal matrix whose diagonal entries are obtained by taking the square root of those of D, which are necessarily real and positive because A is positive definite. This implies the existence of a Cholesky factorization with $\mathsf{C} = \mathsf{L}\sqrt{\mathsf{D}}$. □

Calculation of the Choleski factor

The matrix C can be calculated from (4.13). For example, developing the matrix product gives that $a_{1,1} = c_{1,1}^2$ and so $c_{1,1} = \sqrt{a_{1,1}}$. It is then possible to calculate $c_{2,1}$ from the equation $a_{2,1} = c_{2,1}c_{1,1}$, and so on. Implementing the Cholesky factorization is the goal of Exercise 4.7.

4.2.5 Direct methods for banded matrices

In applications related to partial differential equations, the matrix $\mathsf{A} \in \mathbf{C}^{n\times n}$ very often has a bandwidth which is small in comparison with n.

Definition 4.5. The bandwidth of a matrix $\mathsf{A} \in \mathbf{C}^{n\times n}$ is the smallest number $k \in \mathbf{N}$ such that $a_{ij} = 0$ for all $(i, j) \in \{1, \ldots, n\}^2$ with $|i - j| > k$.

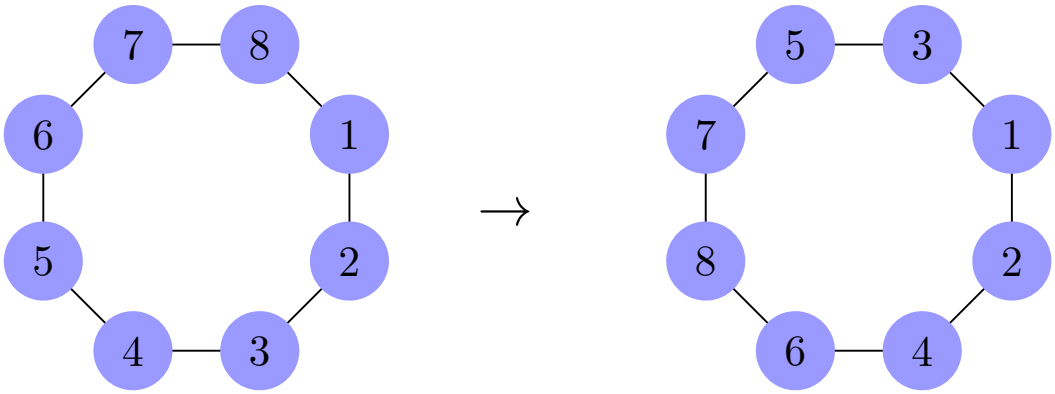
It is not difficult to show that, if A is a matrix with bandwidth k, then so are L and U in the absence of pivoting. This can be proved by equaling the entries of the product LU with those of the matrix A. We emphasize, however, that the sparsity structure *within* the band of A may be destroyed in L and U; this phenomenon is called *fill-in*.

Reducing the bandwidth: the Cuthill–McKee algorithm

The computational cost of calculating the LU or Cholesky decomposition of a matrix with bandwidth k scales as $\mathcal{O}(nk^2)$, which is much better than the general scaling $\mathcal{O}(n^3)$ if $k \ll n$. In applications, the bandwidth k is often related to the matrix size n. For example, if A arises from the discretization of the Laplacian operator, then $k = \mathcal{O}(\sqrt{n})$ provided that a good ordering of the vertices is employed. In this case, the computational cost scales as $\mathcal{O}(n^2)$.

Since a narrow band is associated with a lower computational cost of the LU decomposition, it is natural to wonder whether the bandwidth of a matrix A can be reduced. A possible strategy to this end is to use permutations. More precisely, is it possible to identify a row permutation matrix P such that PAP^T has minimal bandwidth? Given such a matrix, the solution to the linear system (4.1) can be obtained by first solving $(\mathsf{PAP}^T)\boldsymbol{y} = \mathsf{P}b$, and then letting $\boldsymbol{x} = \mathsf{P}^T\boldsymbol{y}$.

The Cuthill–McKee algorithm is a heuristic method for finding a good, but sometimes not optimal, permutation matrix P in the particular case where A is a *Hermitian* matrix. It is based on the fact that, to a Hermitian matrix A, we can associate a unique undirected graph whose adjacency matrix A_* has the same sparsity structure as that of A, i.e. zeros in the same places. For any row permutation matrix P_σ with corresponding permutation σ: $\{1, \ldots, n\} \to \{1, \ldots, n\}$ (see Definition 4.4), the matrices $\mathsf{P}_\sigma\mathsf{A}\mathsf{P}_\sigma^T$ and $\mathsf{P}_\sigma\mathsf{A}_*\mathsf{P}_\sigma^T$ also have the same sparsity structure. Therefore, minimizing the bandwidth of $\mathsf{P}_\sigma\mathsf{A}\mathsf{P}_\sigma^T$ is equivalent to minimizing the bandwidth of $\mathsf{P}_\sigma\mathsf{A}_*\mathsf{P}_\sigma^T$. The key insight for understanding the Cuthill–McKee method is that $\mathsf{P}_\sigma\mathsf{A}_*\mathsf{P}_\sigma^T$ is the adjacency matrix of the graph obtained by renumbering the nodes according to the permutation σ, i.e. by changing the number of the nodes from i to $\sigma(i)$. Consider, for example, the following graph and renumbering:



The associated adjacency matrices are given by:

$$\begin{pmatrix} 1 & 1 & & & & & & 1 \\ 1 & 1 & 1 & & & & & \\ & 1 & 1 & 1 & & & & \\ & & 1 & 1 & 1 & & & \\ & & & 1 & 1 & 1 & & \\ & & & & 1 & 1 & 1 & \\ & & & & & 1 & 1 & 1 \\ 1 & & & & & & 1 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 1 & 1 & & & & & \\ 1 & 1 & & 1 & & & & \\ 1 & & 1 & & 1 & & & \\ & 1 & & 1 & & 1 & & \\ & & 1 & & 1 & & 1 & \\ & & & 1 & & 1 & & 1 \\ & & & & 1 & & 1 & 1 \\ & & & & & 1 & 1 & 1 \end{pmatrix}$$

We assume that the nodes are all self-connected, although this is not depicted, and so the diagonal entries of the adjacency matrices are equal to 1. This renumbering corresponds to the permutation

$$\begin{pmatrix} i: & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \sigma(i): & 1 & 2 & 4 & 6 & 8 & 7 & 5 & 3 \end{pmatrix},$$

and we may verify that the adjacency matrix of the renumbered graph can be obtained from the associated row permutation matrix:

$$\mathsf{PA}_*\mathsf{P}^T = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & & & & & & 1 \\ & & 1 & & & & & \\ & & & & & 1 & & \\ & & & 1 & & & & \\ & & & & & & 1 & \\ & & & & 1 & & & \end{pmatrix} \begin{pmatrix} 1 & 1 & & & & & & 1 \\ 1 & 1 & 1 & & & & & \\ & 1 & 1 & 1 & & & & \\ & & 1 & 1 & 1 & & & \\ & & & 1 & 1 & 1 & & \\ & & & & 1 & 1 & 1 & \\ & & & & & 1 & 1 & 1 \\ 1 & & & & & & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & & 1 & & & & \\ & & & & & 1 & & \\ & & & & & & & 1 \\ & & & & & & 1 & \\ & & & & 1 & & & \\ & & 1 & & & & & \end{pmatrix}$$

In this example, renumbering the nodes of the graph enables a significant reduction of the bandwidth, from 7 to 2. The Cuthill–McKee algorithm, which was employed to calculate the permutation, is an iterative method that produces an ordered n-tuple R containing the nodes in the new order; in other words, it returns $\left(\sigma^{-1}(1), \ldots, \sigma^{-1}(n)\right)$. The first step of the algorithm is to find the node i with the lowest *degree*, i.e. with the smallest number of connections to other nodes, and to initialize $R = (i)$. Then the following steps are repeated until R contains all the nodes of the graph:

- Define A_i as the set containing all the nodes which are adjacent to a node in R but not themselves in R;
- Sort the nodes in A_i according to the following rules: a node $i \in A_i$ comes before $j \in A_i$ if i is connected to a node in R that comes before all the nodes in R to which j is connected. As a tiebreak, precedence is given to the node with highest degree.
- Append the nodes in A_i to R, in the order determined in the previous item.

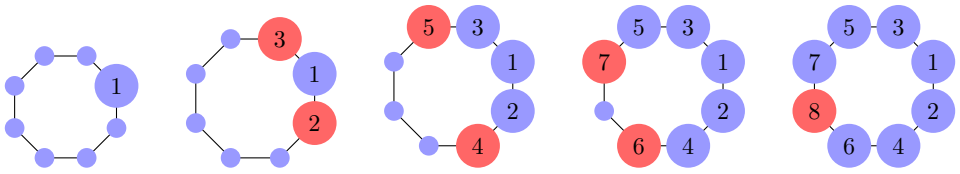


Figure 4.1: Illustration of the Cuthill–McKee algorithm. The new numbering of the nodes is illustrated. The first node was chosen randomly since all the nodes have the same degree. In this example, the ordered tuple R evolves as follows: $(1) \to (1, 2, 8) \to (1, 2, 8, 3, 7) \to (1, 2, 8, 3, 7, 4, 6) \to (1, 2, 8, 3, 7, 4, 6, 5)$.

The steps of the algorithm for the example above are depicted in Figure 4.1. Another example, taken from the original paper by Cuthill and McKeen [1], is presented in Figure 4.2.

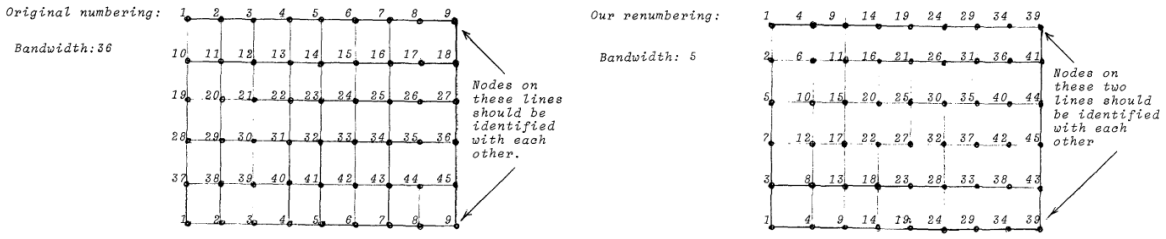


Figure 4.2: Example from the original Cuthill–McKee paper [1].

4.3 Iterative methods for linear systems

Iterative methods enjoy more flexibility than direct methods, because they can be stopped at any point if the residual is deemed sufficiently small. This generally enables to obtain a good solution at a computational cost that is significantly lower than that of direct methods. In this section, we present and study two classes of iterative methods: basic iterative methods based on a splitting of the matrix A, and the so-called Krylov subspace methods.

4.3.1 Basic iterative methods

The basic iterative methods are particular cases of a general *splitting method*. Given a splitting of the matrix of the linear system as $\mathsf{A} = \mathsf{M} - \mathsf{N}$, for a nonsingular matrix $\mathsf{M} \in \mathbf{C}^{n\times n}$ and a matrix $\mathsf{N} \in \mathbf{C}^{n\times n}$, together with an initial guess $\boldsymbol{x}^{(0)}$ of the solution, one step of this general method reads

$$\mathsf{M}\boldsymbol{x}^{(k+1)} = \mathsf{N}\boldsymbol{x}^{(k)} + \boldsymbol{b}. \tag{4.14}$$

For any choice of splitting, the exact solution $\boldsymbol{x}_*$ to the linear system is a fixed point of this iteration, in the sense that if $\boldsymbol{x}^{(0)} = \boldsymbol{x}_*$, then $\boldsymbol{x}^{(k)} = \boldsymbol{x}_*$ for all $k \geqslant 0$. Equation (4.14) is a linear system with matrix M, unknown $\boldsymbol{x}^{(k+1)}$, and right-hand side $\mathsf{N}\boldsymbol{x}^{(k)} + \boldsymbol{b}$. There is a compromise between the cost of a single step and the speed of convergence of the method. In the extreme case where $\mathsf{M} = \mathsf{A}$ and $\mathsf{N} = 0$, the method converges to the exact solution in one step, but performing this step amounts to solving the initial problem. In practice, in order for the method to be useful, the linear system (4.14) should be relatively simple to solve. Concretely, this means that the matrix M should be diagonal, triangular, block diagonal, or block triangular. The error $\boldsymbol{e}^{(k)}$ and residual $\boldsymbol{r}^{(k)}$ at iteration k are defined as follows:

$$\boldsymbol{e}^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{x}_*, \qquad \boldsymbol{r}^{(k)} = \mathsf{A}\boldsymbol{x}^{(k)} - \boldsymbol{b}.$$

Convergence of the splitting method

Before presenting concrete examples of splitting methods, we obtain a necessary and sufficient condition for the convergence of (4.14) for any initial guess $\boldsymbol{x}^{(0)}$.

Proposition 4.10 (Convergence). *The splitting method (4.14) converges for any initial $\boldsymbol{x}^{(0)}$ if and only if $\rho(\mathsf{M}^{-1}\mathsf{N}) < 1$. In addition, for any $\varepsilon > 0$ there exists $K > 0$ such that*

$$\forall k \geqslant K, \qquad \|\boldsymbol{e}^{(k)}\| \leqslant \left(\rho(\mathsf{M}^{-1}\mathsf{N}) + \varepsilon\right)^k \|\boldsymbol{e}^{(0)}\|. \tag{4.15}$$

Proof. Let $\boldsymbol{x}_*$ denote the solution to the linear system. Since $\mathsf{M}\boldsymbol{x}_* - \mathsf{N}\boldsymbol{x}_* = \boldsymbol{b}$, we have

$$\mathsf{M}(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}_*) = \mathsf{N}(\boldsymbol{x}^{(k)} - \boldsymbol{x}_*).$$

Using the assumption that M is nonsingular, we obtain that the error satisfies the equation

$$\boldsymbol{e}^{(k+1)} = (\mathsf{M}^{-1}\mathsf{N})\boldsymbol{e}^{(k)}.$$

Applying this equality repeatedly, we deduce that

$$\boldsymbol{e}^{(k)} = (\mathsf{M}^{-1}\mathsf{N})\boldsymbol{e}^{(k-1)} = \ldots = (\mathsf{M}^{-1}\mathsf{N})^k \boldsymbol{e}^{(0)}. \tag{4.16}$$

Proof that $\rho(\mathsf{M}^{-1}\mathsf{N}) < 1$ is necessary for convergence. We prove the equivalent claim that if $\rho(\mathsf{M}^{-1}\mathsf{N}) \geqslant 1$, then there exists $\boldsymbol{x}^{(0)}$ such that the method is not convergent. Indeed, assume that $\boldsymbol{x}^{(0)} = \boldsymbol{x}_* + \boldsymbol{v}^{(0)}$, where $\boldsymbol{v}^{(0)}$ is the eigenvector of $\mathsf{M}^{-1}\mathsf{N}$ associated with the eigenvalue of largest modulus. Then $\boldsymbol{e}^{(0)} = \boldsymbol{v}^{(0)}$ and the right-hand side of (4.16) does not converge to 0 in the limit as $k \to 0$, because

$$\|(\mathsf{M}^{-1}\mathsf{N})^k \boldsymbol{e}^{(0)}\| = \rho(\mathsf{M}^{-1}\mathsf{N})^k \|\boldsymbol{v}^{(0)}\| \geqslant \|\boldsymbol{v}^{(0)}\|.$$

Thus, the condition $\rho(\mathsf{M}^{-1}\mathsf{N}) < 1$ is necessary to ensure convergence for all initial guess $\boldsymbol{x}^{(0)}$.

Proof that $\rho(\mathsf{M}^{-1}\mathsf{N}) < 1$ is sufficient for convergence. In order to show that the condition is also sufficient, note that by (4.16)

$$\forall k \geqslant 0, \qquad \|\boldsymbol{e}^{(k)}\| \leqslant \left\|(\mathsf{M}^{-1}\mathsf{N})^k\right\| \|\boldsymbol{e}^{(0)}\|.$$

By Gelfand's formula, proved in Proposition A.10 of Appendix A, it holds that

$$\lim_{k\to\infty} \left\|(\mathsf{M}^{-1}\mathsf{N})^k\right\|^{\frac{1}{k}} = \rho(\mathsf{M}^{-1}\mathsf{N}). \tag{4.17}$$

Therefore, we deduce that if $\rho(\mathsf{M}^{-1}\mathsf{N}) < 1$, then $\left\|(\mathsf{M}^{-1}\mathsf{N})^k\right\| \to 0$ and so $\boldsymbol{e}^{(k)} \to \boldsymbol{0}$. In addition, it follows from (4.17) that for all $\varepsilon > 0$ there is $K \in \mathbf{N}$ such that

$$\forall k \geqslant K, \qquad \left\|(\mathsf{M}^{-1}\mathsf{N})^k\right\|^{\frac{1}{k}} \leqslant \rho(\mathsf{M}^{-1}\mathsf{N}) + \varepsilon.$$

Rearranging this inequality gives (4.15).

□

At this point, it is natural to wonder whether there exist sufficient conditions on the matrix A such that the inequality $\rho(\mathsf{M}^{-1}\mathsf{N}) < 1$ is satisfied, which is best achieved on a case by case basis. In the next sections, we present four instances of splitting methods. For each of them, we obtain a sufficient condition for convergence. We are particularly interested in the case where

the matrix A is Hermitian and positive definite, which often arises in applications, and in the case where A is strictly row or column diagonally dominant. We recall that a matrix A is said to be row or column diagonally dominant if, respectively,

$$|a_{ii}| \geqslant \sum_{j \neq i} |a_{ij}| \quad \forall i \qquad \text{or} \qquad |a_{jj}| \geqslant \sum_{i \neq j} |a_{ij}| \quad \forall j.$$

Richardson's method

Arguably the simplest splitting of the matrix A is given by $\mathsf{A} = \frac{1}{\omega}\mathsf{I} - \left(\frac{1}{\omega}\mathsf{I} - \mathsf{A}\right)$, for some parameter $\omega \in \mathbf{R}$, which leads to *Richardson's method*:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \omega(\boldsymbol{b} - \mathsf{A}\boldsymbol{x}^{(k)}). \tag{4.18}$$

In this case the spectral radius which enters in the asymptotic rate of convergence is given by

$$\rho(\mathsf{M}^{-1}\mathsf{N}) = \rho\left(\omega\left(\frac{1}{\omega}\mathsf{I} - \mathsf{A}\right)\right) = \rho\left(\mathsf{I} - \omega\mathsf{A}\right)$$

The eigenvalues of the matrix $\mathsf{I} - \omega\mathsf{A}$ are given by $1 - \omega\lambda_i$, where $(\lambda_i)_{1 \leqslant i \leqslant L}$ are the eigenvalues of A. Therefore, the spectral radius is given by

$$\rho(\mathsf{M}^{-1}\mathsf{N}) = \max_{1 \leqslant i \leqslant L} |1 - \omega\lambda_i|.$$

If the eigenvalues of the matrix A do not either (i) all have a positive real part or (ii) all have a negative real part, then

$$\forall \omega \in \mathbf{R}, \qquad \max_{1 \leqslant i \leqslant L} |1 - \omega\lambda_i| \geqslant 1.$$

In other words, by Proposition 4.10, there is for any choice of $\omega \in \mathbf{R}$ some $\boldsymbol{x}^{(0)}$ such that Richardson's method is non-convergent. Therefore, in order for convergence to hold for all $\boldsymbol{x}^{(0)}$, it is necessary that the eigenvalues of A either all have a negative real part, or all have a positive real part. We focus in the next paragraph on the latter case and we also assume, for simplicity, that A is Hermitian.

Case of symmetric positive definite A. If the matrix A is Hermitian and positive definite, the eigenvalues of A are all real and positive, and it is possible to explicitly calculate the optimal value of ω for convergence. In order for convergence to be as fast as possible, the spectral radius of $\mathsf{M}^{-1}\mathsf{N}$ should be as small as possible, in view of Proposition 4.10. Denoting by $\lambda_{\min}$ and $\lambda_{\max}$ the minimum and maximum eigenvalues of A, it is not difficult to show that

$$\rho(\mathsf{M}^{-1}\mathsf{N}) = \max_{1 \leqslant i \leqslant L} |1 - \omega\lambda_i| = \max\Big\{|1 - \omega\lambda_{\min}|, |1 - \omega\lambda_{\max}|\Big\}. \tag{4.19}$$

The right-hand side is minimized $1 - \omega\lambda_{\min} = \omega\lambda_{\max} - 1$, in which case the two arguments of the maximum in (4.19) are equal. From this we deduce the optimal value of ω and the associated spectral radius:

$$\omega_{\text{opt}} = \frac{2}{\lambda_{\max} + \lambda_{\min}}, \qquad \rho_{\text{opt}} = 1 - \frac{2\lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{\kappa_2(\mathsf{A}) - 1}{\kappa_2(\mathsf{A}) + 1}.$$

We observe that the smaller the condition number of the matrix A, the better the asymptotic rate of convergence.

Remark 4.2 (Link to optimization). In the case where $\mathsf{A} \in \mathbf{R}^{n\times n}$ is symmetric and positive definite, the Richardson update (4.18) may be viewed as a step of the steepest descent algorithm, for the function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\mathsf{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \omega\nabla f(\boldsymbol{x}^{(k)}). \tag{4.20}$$

The gradient of this function is $\nabla f(\boldsymbol{x}) = \mathsf{A}\boldsymbol{x} - \boldsymbol{b}$, and its Hessian matrix is A. Since the Hessian matrix is positive definite, the function is convex and attains its global minimum when ∇f is zero, i.e. when $\mathsf{A}x = \boldsymbol{b}$.

Jacobi's method

In Jacobi's method, the matrix M in the splitting is the diagonal matrix D with the same entries as those of A on the diagonal. We denote by L and U the lower and upper triangular parts of A, without the diagonal. One step of the method reads

$$\mathsf{D}\boldsymbol{x}^{(k+1)} = (\mathsf{D} - \mathsf{A})\boldsymbol{x}^{(k)} + \boldsymbol{b} = -(\mathsf{L} + \mathsf{U})\boldsymbol{x}^{(k)} + \boldsymbol{b} \tag{4.21}$$

Since the matrix D on the left-hand side is diagonal, this linear system with unknown $\boldsymbol{x}^{(k+1)}$ is very simple to solve. The equation (4.21) can be rewritten as

$$\begin{cases} a_{11}x_1^{(k+1)} + a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)} = b_1 \\ a_{21}x_1^{(k)} + a_{22}x_2^{(k+1)} + \cdots + a_{2n}x_n^{(k)} = b_2 \\ \vdots \\ a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \cdots + a_{nn}x_n^{(k+1)} = b_n. \end{cases}$$

The updates for each of the entries of $\boldsymbol{x}^{(k+1)}$ are independent, and so the Jacobi method lends itself well to parallel implementation. The computational cost of one iteration, measured in number of floating point operations required, scales as $\mathcal{O}(n^2)$ if A is a full matrix, or $\mathcal{O}(nk)$ if A is a sparse matrix with k nonzero elements per row on average. It is simple to prove the convergence of Jacobi's method is the case where A is diagonally dominant.

Proposition 4.11. *Assume that* A *is strictly (row or column) diagonally dominant. Then it holds that* $\rho(\mathsf{M}^{-1}\mathsf{N}) < 1$ *for the Jacobi splitting.*

Proof. Assume that λ is an eigenvalue of $\mathsf{M}^{-1}\mathsf{N}$ and $\boldsymbol{v}$ is the associated unit eigenvector. Then

$$\mathsf{M}^{-1}\mathsf{N}\boldsymbol{v} = \lambda\boldsymbol{v} \quad \Leftrightarrow \quad \mathsf{N}\boldsymbol{v} = \lambda\mathsf{M}\boldsymbol{v} \quad \Leftrightarrow \quad (\mathsf{N} - \lambda\mathsf{M})\boldsymbol{v} = 0.$$

In the case of Jacobi's splitting, this is equivalent to

$$-(\mathsf{L} + \lambda\mathsf{D} + \mathsf{U})\boldsymbol{v} = 0.$$

If $|\lambda| > 1$, then the matrix on the left-hand side of this equation is diagonally dominant and thus invertible (see Exercise 4.9). Therefore $\boldsymbol{v} = 0$, but this is a contradiction because $\boldsymbol{v}$ is vector of unit norm. Consequently, all the eigenvalues are bounded from above strictly by 1 in modulus. □

Gauss–Seidel's method

In Gauss Seidel's method, the matrix M in the splitting is the lower triangular part of A, including the diagonal. One step of the method then reads

$$(\mathsf{L} + \mathsf{D})\boldsymbol{x}^{(k+1)} = -\mathsf{U}\boldsymbol{x}^{(k)} + \boldsymbol{b} \tag{4.22}$$

The system is solved by forward substitution. The equation (4.22) can be rewritten equivalently as

$$\begin{cases} a_{11}x_1^{(k+1)} + a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \cdots + a_{1n}x_n^{(k)} = b_1 \\ a_{21}x_1^{(k+1)} + a_{22}x_2^{(k+1)} + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)} = b_2 \\ a_{32}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{33}x_3^{(k+1)} + \cdots + a_{3n}x_n^{(k)} = b_3 \\ \vdots \\ a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + a_{n3}x_3^{(k+1)} + \cdots + a_{nn}x_n^{(k+1)} = b_n. \end{cases}$$

Given $\boldsymbol{x}^{(k)}$, the first entry of $\boldsymbol{x}^{(k+1)}$ is obtained from the first equation. Then the value of the second entry is obtained from the second equation, etc. Unlike Jacobi's method, the Gauss–Seidel method is sequential and the entries of $\boldsymbol{x}^{(k+1)}$ cannot be updated in parallel.

It is possible to prove the convergence of the Gauss–Seidel method in particular cases. For example, the method converges if A is strictly diagonally dominant. Proving this, using an approach similar to that in the proof of Proposition 4.11, is the goal of Exercise 4.18. It is also possible to prove convergence when A is Hermitian and positive definite. We show this in the next section for the relaxation method, which generalizes the Gauss–Seidel method.

Relaxation method

The relaxation method generalizes the Gauss–Seidel method. It corresponds to the splitting

$$\mathsf{A} = \left(\frac{\mathsf{D}}{\omega} + \mathsf{L}\right) - \left(\frac{1-\omega}{\omega}\mathsf{D} - \mathsf{U}\right). \tag{4.23}$$

When $\omega = 1$, this is simply the Gauss–Seidel splitting. The idea of the relaxation method is that, by letting ω be a parameter that can differ from 1, faster convergence can be achieved. This intuition will be verified later. The equation (4.14) for this splitting can be rewritten equivalently as

$$\begin{cases} a_{11}\big(x_1^{(k+1)} - x_1^{(k)}\big) = -\omega\Big(a_{11}x_1^{(k)} + a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)} - b_1\Big) \\ a_{22}\big(x_2^{(k+1)} - x_2^{(k)}\big) = -\omega\Big(a_{21}x_1^{(k+1)} + a_{22}x_2^{(k)} + \cdots + a_{2n}x_n^{(k)} - b_2\Big) \\ \vdots \\ a_{nn}\big(x_n^{(k+1)} - x_n^{(k)}\big) = -\omega\Big(a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + \cdots + a_{nn}x_n^{(k)} - b_n\Big). \end{cases}$$

The coefficient on the right-hand side is larger than in the Gauss–Seidel method if $\omega > 1$, and smaller if $\omega < 1$. These regimes are called *over-relaxation* and *under-relaxation*, respectively.

To conclude this section, we establish a sufficient condition for the convergence of the relaxation method, and also of the Gauss–Seidel method as a particular case when $\omega = 1$, when the matrix A is Hermitian and positive definite. To this end, we begin by showing the following preparatory result, which concerns a general splitting $\mathsf{A} = \mathsf{M} - \mathsf{N}$.

Proposition 4.12. *Let A be Hermitian and positive definite. If the Hermitian matrix $\mathsf{M}^* + \mathsf{N}$ is positive definite, then $\rho(\mathsf{M}^{-1}\mathsf{N}) < 1$.*

Proof. First, notice that $\mathsf{M}^* + \mathsf{N}$ is indeed Hermitian because

$$(\mathsf{M}^* + \mathsf{N})^* = \mathsf{M} + \mathsf{N}^* = \mathsf{A} + \mathsf{N} + \mathsf{N}^*.$$

We will show that $\|\mathsf{M}^{-1}\mathsf{N}\|_{\mathsf{A}} < 1$, where $\|\bullet\|_{\mathsf{A}}$ is the matrix norm induced by the following norm on vectors:

$$\|\boldsymbol{x}\|_{\mathsf{A}} := \sqrt{\boldsymbol{x}^*\mathsf{A}\boldsymbol{x}}.$$

Showing that this indeed defines a vector norm is the goal of Exercise 4.11. Since $\mathsf{N} = \mathsf{M} - \mathsf{A}$, it holds that

$$\|\mathsf{M}^{-1}\mathsf{N}\|_{\mathsf{A}} = \|\mathsf{I} - \mathsf{M}^{-1}\mathsf{A}\|_{\mathsf{A}},$$

and so

$$\|\mathsf{M}^{-1}\mathsf{N}\|_{\mathsf{A}} = \sup\Big\{\|\boldsymbol{x} - \mathsf{M}^{-1}\mathsf{A}\boldsymbol{x}\|_{\mathsf{A}} \colon \|\boldsymbol{x}\|_{\mathsf{A}} \leqslant 1\Big\}.$$

Take $\boldsymbol{x} \in \mathbf{C}^n$ with $\|\boldsymbol{x}\|_{\mathsf{A}} \leqslant 1$ and let $\boldsymbol{y} = \mathsf{M}^{-1}\mathsf{A}\boldsymbol{x}$. We calculate

$$\begin{aligned} \|\boldsymbol{x} - \mathsf{M}^{-1}\mathsf{A}\boldsymbol{x}\|_{\mathsf{A}}^2 &= \boldsymbol{x}^*\mathsf{A}\boldsymbol{x} - \boldsymbol{y}^*\mathsf{A}\boldsymbol{x} - \boldsymbol{x}^*\mathsf{A}\boldsymbol{y} + \boldsymbol{y}^*\mathsf{A}\boldsymbol{y} \\ &= \boldsymbol{x}^*\mathsf{A}\boldsymbol{x} - \boldsymbol{y}^*\mathsf{M}\mathsf{M}^{-1}\mathsf{A}\boldsymbol{x} - (\mathsf{M}^{-1}\mathsf{A}\boldsymbol{x})^*\mathsf{M}^*\boldsymbol{y} + \boldsymbol{y}^*\mathsf{A}\boldsymbol{y} \\ &= \boldsymbol{x}^*\mathsf{A}\boldsymbol{x} - \boldsymbol{y}^*\mathsf{M}\boldsymbol{y} - \boldsymbol{y}^*\mathsf{M}^*\boldsymbol{y} + \boldsymbol{y}^*(\mathsf{M} - \mathsf{N})\boldsymbol{y} \\ &= \boldsymbol{x}^*\mathsf{A}\boldsymbol{x} - \boldsymbol{y}^*(\mathsf{M}^* + \mathsf{N})\boldsymbol{y} \leqslant 1 - \boldsymbol{y}^*(\mathsf{M}^* + \mathsf{N})\boldsymbol{y} < 1, \end{aligned}$$

where we used in the last inequality the assumption that $\mathsf{M}^* + \mathsf{N}$ is positive definite. This inequality holds true for all $\boldsymbol{x} \in \mathbf{C}^n$ with $\|\boldsymbol{x}\|_{\mathsf{A}} = 1$, and so we conclude that $\|\mathsf{M}^{-1}\mathsf{N}\|_{\mathsf{A}} < 1$, which implies that $\rho(\mathsf{M}^{-1}\mathsf{N}) < 1$. □

As a corollary, we obtain a sufficient condition for the convergence of the relaxation method.

Corollary 4.13. *Assume that* A *is Hermitian and positive definite. Then the relaxation method converges if* $\omega \in (0, 2)$.

Proof. For the relaxation method, we have

$$\mathsf{M} + \mathsf{N}^* = \left(\frac{\mathsf{D}}{\omega} + \mathsf{L}\right) + \left(\frac{1-\omega}{\omega}\mathsf{D} - \mathsf{U}\right)^*.$$

Since A is Hermitian, it holds that $\mathsf{D}^* = \mathsf{D}$ and $\mathsf{U}^* = \mathsf{L}$. Therefore,

$$\mathsf{M} + \mathsf{N}^* = \frac{2-\omega}{\omega}\mathsf{D}.$$

The diagonal elements of D are all positive, because A is positive definite. (Indeed, if there was an index i such that $d_{ii} \leqslant 0$, then it would hold that $\boldsymbol{e}_i^T \mathsf{A} \boldsymbol{e}_i = d_{ii} \leqslant 0$, contradicting the assumption that A is positive definite.) We deduce that $\mathsf{M} + \mathsf{N}^*$ is positive definite if and only if $\omega \in (0, 2)$. We can then conclude the proof by using Proposition 4.12. □

Note that Corollary 4.13 implies as a particular case the convergence of the Gauss–Seidel method when A is Hermitian and positive definite. The condition $\omega \in (0, 2)$ is in fact necessary for the convergence of the relaxation method, not only in the case of a Hermitian positive definite matrix A but in general.

Proposition 4.14 (Necessary condition for the convergence of the relaxation method)**.** *Let* $\mathsf{A} \in \mathbf{C}^{n\times n}$ *be an invertible matrix, and let* $\mathsf{A} = \mathsf{M}_\omega - \mathsf{N}_\omega$ *denote the splitting of the relaxation method with parameter* ω*. It holds that*

$$\forall \omega \neq 0, \qquad \rho(\mathsf{M}_\omega^{-1}\mathsf{N}_\omega) \geqslant |\omega - 1|.$$

Proof. We recall the following facts:

- the determinant of a product of matrices is equal to the product of the determinants.
- the determinant of a triangular matrix is equal to the product of its diagonal entries;
- the determinant of a matrix is equal to the product of its eigenvalues, to the power of their algebraic multiplicity. This can be shown from the previous two items, by passing to the Jordan normal form.

Therefore, we have that

$$\det(\mathsf{M}_\omega^{-1}\mathsf{N}_\omega) = \det(\mathsf{M}_\omega)^{-1}\det(\mathsf{N}_\omega) = \frac{\det\left(\frac{1-\omega}{\omega}\mathsf{D} - \mathsf{U}\right)}{\det\left(\frac{\mathsf{D}}{\omega} + \mathsf{L}\right)} = (1-\omega)^n.$$

Since the determinant on the left-hand side is the product of the eigenvalues of $\mathsf{M}_\omega^{-1}\mathsf{N}_\omega$, it is bounded from above in modulus by $\rho(\mathsf{M}_\omega^{-1}\mathsf{N}_\omega)^n$, and so we deduce $\rho(\mathsf{M}_\omega^{-1}\mathsf{N}_\omega)^n \geqslant |1-\omega|^n$. The statement then follows by taking the n-th root. □

Comparison between Jacobi and Gauss–Seidel for tridiagonal matrices

For tridiagonal matrices, the convergence rate of the Jacobi and Gauss–Seidel methods satisfy an explicit relation, which we prove in this section. We denote the Jacobi and Gauss–Seidel splittings by $\mathsf{M}_\mathcal{J} - \mathsf{N}_\mathcal{J}$ and $\mathsf{M}_\mathcal{G} - \mathsf{N}_\mathcal{G}$, respectively, and use the following notation for the entries of the matrix A:

$$\begin{pmatrix} a_1 & b_1 & & \\ c_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & c_{n-1} & a_n \end{pmatrix}.$$

Before presenting and proving the result, notice that for any $\mu \neq 0$ it holds that

$$\begin{pmatrix} \mu & & & \\ & \mu^2 & & \\ & & \ddots & \\ & & & \mu^n \end{pmatrix} \mathsf{A} \begin{pmatrix} \mu^{-1} & & & \\ & \mu^{-2} & & \\ & & \ddots & \\ & & & \mu^{-n} \end{pmatrix} = \begin{pmatrix} a_1 & \mu^{-1}b_1 & & \\ \mu c_1 & \ddots & \ddots & \\ & \ddots & \ddots & \mu^{-1}b_{n-1} \\ & & \mu c_{n-1} & a_n \end{pmatrix}. \tag{4.24}$$

Proposition 4.15. *Assume that* A *is tridiagonal with nonzero diagonal elements, so that both* $\mathsf{M}_\mathcal{J} = \mathsf{D}$ *and* $\mathsf{M}_\mathcal{G} = \mathsf{L} + \mathsf{D}$ *are invertible. Then*

$$\rho(\mathsf{M}_\mathcal{G}^{-1}\mathsf{N}_\mathcal{G}) = \rho(\mathsf{M}_\mathcal{J}^{-1}\mathsf{N}_\mathcal{J})^2$$

Proof. If λ is an eigenvalue of $\mathsf{M}_\mathcal{G}^{-1}\mathsf{N}_\mathcal{G}$ with associated unit eigenvector $\boldsymbol{v}$, then

$$\mathsf{M}_\mathcal{G}^{-1}\mathsf{N}_\mathcal{G}\boldsymbol{v} = \lambda\boldsymbol{v} \quad\Leftrightarrow\quad \mathsf{N}_\mathcal{G}\boldsymbol{v} = \lambda\mathsf{M}_\mathcal{G}\boldsymbol{v} \quad\Leftrightarrow\quad (\mathsf{N}_\mathcal{G} - \lambda\mathsf{M}_\mathcal{G})\boldsymbol{v} = 0.$$

For fixed λ, there exists a nontrivial solution $\boldsymbol{v}$ to the last equation if and only if

$$p_\mathcal{G}(\lambda) := \det(\mathsf{N}_\mathcal{G} - \lambda\mathsf{M}_\mathcal{G}) = \det(-\lambda\mathsf{L} - \lambda\mathsf{D} - \mathsf{U}) = 0.$$

Likewise, λ is an eigenvalue of $\mathsf{M}_\mathcal{J}^{-1}\mathsf{N}_\mathcal{J}$ if and only if

$$p_\mathcal{J}(\lambda) := \det(\mathsf{N}_\mathcal{J} - \lambda\mathsf{M}_\mathcal{J}) = \det(-\mathsf{L} - \lambda\mathsf{D} - \mathsf{U}) = 0.$$

Now notice that, for $\lambda \neq 0$,

$$p_\mathcal{G}(\lambda^2) = \det\big(-\lambda^2\mathsf{L} - \lambda^2\mathsf{D} - \mathsf{U}\big) = \lambda^n \det\big(-\lambda\mathsf{L} - \lambda\mathsf{D} - \lambda^{-1}\mathsf{U}\big).$$

Applying (4.24) with $\mu = \lambda \neq 0$, we deduce

$$p_\mathcal{G}(\lambda^2) = \lambda^n \det\big(-\mathsf{L} - \lambda\mathsf{D} - \mathsf{U}\big) = \lambda^n p_\mathcal{J}(\lambda)$$

It is clear that this relation is true also if $\lambda = 0$. Consequently, it holds that if λ is an eigenvalue of the matrix $\mathsf{M}_{\mathcal{J}}^{-1}\mathsf{N}_{\mathcal{J}}$ then λ^2 is an eigenvalue of $\mathsf{M}_{\mathcal{G}}^{-1}\mathsf{N}_{\mathcal{G}}$. Conversely, if λ is a nonzero eigenvalue of $\mathsf{M}_{\mathcal{G}}^{-1}\mathsf{N}_{\mathcal{G}}$, then the two square roots of λ are eigenvalues of $\mathsf{M}_{\mathcal{J}}^{-1}\mathsf{N}_{\mathcal{J}}$. □

If a matrix A is tridiagonal and Toeplitz, i.e. if it is of the form

$$\begin{pmatrix} a & b & & \\ c & \ddots & \ddots & \\ & \ddots & \ddots & b \\ & & c & a \end{pmatrix},$$

then it is possible to prove that the eigenvalues of A are given by

$$\lambda_k = a + 2\sqrt{bc}\cos\left(\frac{k\pi}{n+1}\right), \qquad k = 1, \ldots, n. \tag{4.25}$$

In this case, the spectral radius of $\mathsf{M}_{\mathcal{J}}^{-1}\mathsf{N}_{\mathcal{J}}$ can be determined explicitly.

Monitoring the convergence

In practice, we have access to the residual $\boldsymbol{r}^{(k)} = \mathsf{A}\boldsymbol{x}^{(k)} - \boldsymbol{b}$ at each iteration, but not to the error $\boldsymbol{e}^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{x}_*$, as calculating the latter would require to know the exact solution of the problem. Nevertheless, the two are related by the equation

$$\boldsymbol{r}^{(k)} = \mathsf{A}\boldsymbol{e}^{(k)} \quad \Leftrightarrow \quad \boldsymbol{e}^{(k)} = \mathsf{A}^{-1}\boldsymbol{r}^{(k)}.$$

Therefore, it holds that $\|\boldsymbol{e}^{(k)}\| \leqslant \|\mathsf{A}^{-1}\|\|\boldsymbol{r}^{(k)}\|$. Likewise, the relative error satisfies

$$\frac{\|\boldsymbol{e}^{(k)}\|}{\|\boldsymbol{x}_*\|} = \frac{\|\mathsf{A}^{-1}\boldsymbol{r}^{(k)}\|}{\|\mathsf{A}^{-1}\boldsymbol{b}\|},$$

and since $\|\boldsymbol{b}\| = \|\mathsf{A}\mathsf{A}^{-1}\boldsymbol{b}\| \leqslant \|\mathsf{A}\|\|\mathsf{A}^{-1}\boldsymbol{b}\|$, we deduce

$$\frac{\|\boldsymbol{e}^{(k)}\|}{\|\boldsymbol{x}_*\|} \leqslant \kappa(\mathsf{A})\frac{\|\boldsymbol{r}^{(k)}\|}{\|\boldsymbol{b}\|}.$$

The fraction on the right-hand side is the *relative residual*. If the system is well conditioned, that is if $\kappa(A)$ is close to one, then controlling the relative residual enables a good control of the relative error.

Stopping criterion

In practice, several criteria can be employed in order to decide when to stop iterating. Given a small number ε (unrelated to the machine epsilon in Chapter 1), the following alternatives are available:

- Stop when $\|\boldsymbol{r}^{(k)}\| \leqslant \varepsilon$. The downside of this approach is that it is not *scaling invariant*: when used for solving the following rescaled system

$$k\mathsf{A}\boldsymbol{x} = k\boldsymbol{b}, \qquad k \neq 1,$$

a splitting method with rescaled initial guess $k\boldsymbol{x}^{(0)}$ will require a number of iterations that depends on k: fewer if $k \ll 1$ and more if $k \gg 1$. In practice, controlling the relative residual and the relative error is often preferable.

- Stop when $\|\boldsymbol{r}^{(k)}\|/\|\boldsymbol{r}^{(0)}\| \leqslant \varepsilon$. This criterion is scaling invariant, but the number of iterations is dependent on the quality of the initial guess $\boldsymbol{x}^{(0)}$.
- Stop when $\|\boldsymbol{r}^{(k)}\|/\|\boldsymbol{b}\|$. This criterion is generally the best, because it is both scaling invariant and the quality of the final iterate is independent of that of the initial guess.

4.3.2 The conjugate gradient method

As already mentioned in Remark 4.2, when the matrix $\mathsf{A} \in \mathbf{C}^{n\times n}$ in the linear system $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ is symmetric and positive definite, the system can be interpreted as a minimization problem for the function

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\mathsf{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}. \tag{4.26}$$

The fact that the exact solution $\boldsymbol{x}_*$ to the linear system is the unique minimizer of this function appears clearly when rewriting f as follows:

$$f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_*)^T\mathsf{A}(\boldsymbol{x} - \boldsymbol{x}_*) - \frac{1}{2}\boldsymbol{x}_*^T\mathsf{A}\boldsymbol{x}_*. \tag{4.27}$$

The second term is constant with $\boldsymbol{x}$, and the first term is strictly positive if $\boldsymbol{x} - \boldsymbol{x}_* \neq \boldsymbol{0}$, because A is positive definite. We saw that Richardson's method can be interpreted as a steepest descent with fixed step size,

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \omega\nabla f(\boldsymbol{x}^{(k)}).$$

In this section, we will present and study other methods for solving the linear system (4.1) which can be viewed as optimization methods. Since A is symmetric, it is diagonalizable and the function f can be rewritten as

$$\begin{aligned} f(\boldsymbol{x}) &= \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_*)^T\mathsf{Q}\mathsf{D}\mathsf{Q}^T(\boldsymbol{x} - \boldsymbol{x}_*) - \frac{1}{2}\boldsymbol{x}_*^T\mathsf{A}\boldsymbol{x}_* \\ &= \frac{1}{2}(\mathsf{Q}^T\boldsymbol{e})^T\mathsf{D}(\mathsf{Q}^T\boldsymbol{e}) - \frac{1}{2}\boldsymbol{x}_*^T\mathsf{A}\boldsymbol{x}_*, \qquad \boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_*. \end{aligned}$$

Therefore, we have that

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{n}\lambda_i\eta_i^2 - \frac{1}{2}\boldsymbol{x}_*^T\mathsf{A}\boldsymbol{x}_*, \qquad \boldsymbol{\eta} = \boldsymbol{Q}^T(\boldsymbol{x} - \boldsymbol{x}_*),$$

where $(\lambda_i)_{1\leqslant i\leqslant n}$ are the diagonal entries of D. This shows that f is a paraboloid after a change of coordinates.

Steepest descent method

The steepest descent method is more general than Richardson's method in the sense that the step size changes from iteration to iteration and the method is not restricted to quadratic functions of the form (4.26). Each iteration is of the form

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \omega_k \nabla f(\boldsymbol{x}^{(k)}).$$

It is natural to wonder whether the step size ω_k can be fixed in such a way that $f(\boldsymbol{x}^{(k+1)})$ is as small as possible. For the case of the quadratic function (4.26), this value of ω_k can be calculated explicitly for a general search direction $\boldsymbol{d}$, and in particular also when $\boldsymbol{d} = \nabla f(\boldsymbol{x}^{(k)})$. We calculate that

$$\begin{aligned} f(\boldsymbol{x}^{(k+1)}) &= f\Big(\boldsymbol{x}^{(k)} - \omega_k \boldsymbol{d}\Big) = \frac{1}{2}\big(\boldsymbol{x}^{(k)} - \omega_k \boldsymbol{d}\big)^T \mathsf{A}\big(\boldsymbol{x}^{(k)} - \omega_k \boldsymbol{d}\big) - \big(\boldsymbol{x}^{(k)} - \omega_k \boldsymbol{d}\big)^T \boldsymbol{b} \\ &= f\Big(\boldsymbol{x}^{(k)}\Big) + \frac{\omega_k^2}{2} \boldsymbol{d}^T \mathsf{A} \boldsymbol{d} - \omega_k \boldsymbol{d}^T \boldsymbol{r}^{(k)}. \end{aligned} \tag{4.28}$$

When viewed as a function of the real parameter ω_k, the right-hand side is a convex quadratic function. It is minimized when its derivative is equal to zero, i.e. when

$$\omega_k \boldsymbol{d}^T \mathsf{A} \boldsymbol{d} - \boldsymbol{d}^T (\mathsf{A}\boldsymbol{x}_k - \boldsymbol{b}) = 0 \qquad \Rightarrow \qquad \omega_k = \frac{\boldsymbol{d}^T \boldsymbol{r}^{(k)}}{\boldsymbol{d}^T \mathsf{A} \boldsymbol{d}}. \tag{4.29}$$

The steepest descent algorithm with step size obtained from this equation is summarized in Algorithm 3 below. By construction, the function value $f(\boldsymbol{x}^{(k)})$ is nonincreasing with k, which is equivalent to saying that the error $\boldsymbol{x} - \boldsymbol{x}_*$ is nonincreasing in the norm $\boldsymbol{x} \mapsto \sqrt{\boldsymbol{x}^T \mathsf{A} \boldsymbol{x}}$. In order to quantify more precisely the decrease of the error in this norm, we introduce the notation

$$E_k = \|\boldsymbol{x} - \boldsymbol{x}_*\|_{\mathsf{A}}^2 := (\boldsymbol{x}^{(k)} - \boldsymbol{x}_*)^T \mathsf{A} (\boldsymbol{x}^{(k)} - \boldsymbol{x}_*) = (\mathsf{A}\boldsymbol{x}^{(k)} - \boldsymbol{b})^T \mathsf{A}^{-1} (\mathsf{A}\boldsymbol{x}^{(k)} - \boldsymbol{b}).$$

We begin by showing the following auxiliary lemma.

Lemma 4.16 (Kantorovich inequality)**.** *Let $\mathsf{A} \in \mathbf{R}^{n \times n}$ be a symmetric and positive definite matrix, and let $\lambda_0 \leqslant \ldots \leqslant \lambda_n$ denote its eigenvalues. Then for all nonzero $\boldsymbol{z} \in \mathbf{R}^n$ it holds that*

$$\frac{(\boldsymbol{z}^T \boldsymbol{z})^2}{(\boldsymbol{z}^T \mathsf{A} \boldsymbol{z})(\boldsymbol{z}^T \mathsf{A}^{-1} \boldsymbol{z})} \geqslant \frac{4\lambda_1 \lambda_n}{(\lambda_1 + \lambda_n)^2}.$$

Proof. By the AM-GM (arithmetic mean-geometric mean) inequality, it holds for all $t > 0$ that

$$\begin{aligned} \sqrt{(\boldsymbol{z}^T \mathsf{A} \boldsymbol{z})(\boldsymbol{z}^T \mathsf{A}^{-1} \boldsymbol{z})} &= \sqrt{(t\boldsymbol{z}^T \mathsf{A} \boldsymbol{z})(t^{-1} \boldsymbol{z}^T \mathsf{A}^{-1} \boldsymbol{z})} \leqslant \frac{1}{2}\left(t\boldsymbol{z}^T \mathsf{A} \mathsf{z} + \frac{1}{t} \boldsymbol{z}^T \mathsf{A}^{-1} \mathsf{z}\right) \\ &= \frac{1}{2} \boldsymbol{z}^T \left(t\mathsf{A} + \frac{1}{t} \mathsf{A}^{-1}\right) \boldsymbol{z}. \end{aligned}$$

The matrix on the right-hand side is also symmetric and positive definite, with eigenvalues equal to $t\lambda_i + (t\lambda_i)^{-1}$. Therefore, we deduce

$$\forall t \geqslant 0, \qquad \sqrt{(\boldsymbol{z}^T \mathsf{A} \boldsymbol{z})(\boldsymbol{z}^T \mathsf{A}^{-1} \boldsymbol{z})} \leqslant \frac{1}{2} \left(\max_{i \in \{1, \dots, n\}} t\lambda_i + (t\lambda_i)^{-1} \right) \boldsymbol{z}^T \boldsymbol{z}. \tag{4.30}$$

The function $x \mapsto x + x^{-1}$ is convex, and so over any closed interval $[x_{\min}, x_{\max}]$ it attains its maximum either at $x_{\min}$ or at $x_{\max}$. Consequently, it holds that

$$\left(\max_{i \in \{1, \dots, n\}} t\lambda_i + (t\lambda_i)^{-1} \right) = \max \left\{ t\lambda_1 + \frac{1}{t\lambda_1}, t\lambda_n + \frac{1}{t\lambda_n} \right\}.$$

In order to obtain the best possible bound from (4.30), we should let t be such that the maximum is minimized, which occurs when the two arguments of the maximum are equal:

$$t\lambda_1 + \frac{1}{t\lambda_1} = t\lambda_n + \frac{1}{t\lambda_n} \qquad \Rightarrow \qquad t = \frac{1}{\sqrt{\lambda_1 \lambda_n}}.$$

For this value of t, the maximum in (4.30) is equal to

$$\sqrt{\frac{\lambda_1}{\lambda_n}} + \sqrt{\frac{\lambda_n}{\lambda_1}}.$$

By substituting this expression in (4.30) and rearranging, we obtain the statement. □

We are now able to prove the convergence of the steepest descent method.

Theorem 4.17 (Convergence of the steepest descent method)**.** *It holds that*

$$E_{k+1} \leqslant \left(\frac{\kappa_2(\mathsf{A}) - 1}{\kappa_2(\mathsf{A}) + 1} \right)^2 E_k.$$

Proof. Substituting $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \omega_k \boldsymbol{d}$ in the expression for E_{k+1}, we obtain

$$\begin{aligned} E_{k+1} &= (\boldsymbol{x}^{(k)} - \omega_k \boldsymbol{d} - \boldsymbol{x}_*)^T \mathsf{A} (\boldsymbol{x}^{(k)} - \omega_k \boldsymbol{d} - \boldsymbol{x}_*) \\ &= E_k - 2\omega_k \boldsymbol{d}^T \boldsymbol{r}^{(k)} + \omega_k^2 \boldsymbol{d}^T \mathsf{A} \boldsymbol{d} \\ &= E_k - \frac{(\boldsymbol{d}^T \boldsymbol{d})^2}{\boldsymbol{d}^T \mathsf{A} \boldsymbol{d}} = \left(1 - \frac{(\boldsymbol{d}^T \boldsymbol{d})^2}{(\boldsymbol{d}^T \mathsf{A} \boldsymbol{d})(\boldsymbol{d}^T \mathsf{A}^{-1} \boldsymbol{d})} \right) E_k, \end{aligned}$$

Using the Kantorovich inequality, we have

$$E_{k+1} \leqslant \left(1 - \frac{4\lambda_1 \lambda_n}{(\lambda_1 + \lambda_n)^2} \right) E_k \leqslant \left(\frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} \right)^2 E_k = \left(\frac{\kappa_2(\mathsf{A}) - 1}{\kappa_2(\mathsf{A}) + 1} \right)^2 E_k.$$

We immediately deduce the statement from this inequality. □

Algorithm 3 Steepest descent method

Pick ε and initial $\boldsymbol{x}$
$\boldsymbol{r} \leftarrow \mathsf{A}\boldsymbol{x} - \boldsymbol{b}$
while $\|\boldsymbol{r}\| \geqslant \varepsilon\|\boldsymbol{b}\|$ **do**
    $\boldsymbol{d} \leftarrow \boldsymbol{r}$
    $\omega \leftarrow \boldsymbol{d}^T\boldsymbol{r}/\boldsymbol{d}^T\mathsf{A}\boldsymbol{d}$
    $\boldsymbol{x} \leftarrow \boldsymbol{x} - \omega\boldsymbol{d}$
    $\boldsymbol{r} \leftarrow \mathsf{A}\boldsymbol{x} - \boldsymbol{b}$
end while

Remark 4.3. The attentive reader will have noticed that the rate of convergence rate of the steepest descent method is not apparently better than that of Richardson's method with optimal ω; in both cases, some upper bound on the norm of the error is multiplied by

$$\frac{\kappa_2(\mathsf{A}) - 1}{\kappa_2(\mathsf{A}) + 1}$$

at each iteration. For the steepest descent method, this rate of convergence is always guaranteed, but for Richardson's method, this rate of convergence holds only for the optimal value of ω, which itself depends on the condition number $\kappa_2(\mathsf{A})$ and is computationally expensive to approximate. Indeed, as we shall see in Chapter 6, the simplest method for calculating the smallest eigenvalue of a matrix, which is necessary for estimating the condition number, requires to solve linear systems with matrix A.

Preconditioned steepest descent

We observe from Theorem 4.17 that the convergence of the steepest descent method is faster when the condition number of the matrix A is low. This naturally leads to the following question: can we reformulate the minimization of $f(\boldsymbol{x})$ in (4.26) as another optimization problem which is of the same form but involves a matrix with a lower condition number, thereby providing scope for faster convergence? In order to answer this question, we consider a linear change of coordinates $\boldsymbol{y} = \mathsf{T}^{-1}\boldsymbol{x}$, where T is an invertible matrix, and we define

$$\tilde{f}(\boldsymbol{y}) = f(\mathsf{T}\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^T(\mathsf{T}^T\mathsf{A}\mathsf{T})\boldsymbol{y} - (\mathsf{T}^T\boldsymbol{b})^T\boldsymbol{y}. \tag{4.31}$$

This function is of the same form as f in (4.26), with the matrix $\tilde{\mathsf{A}} := \mathsf{T}^T\mathsf{A}\mathsf{T}$ instead of A and the vector $\tilde{\boldsymbol{b}} := \mathsf{T}^T\boldsymbol{b}$ instead of $\boldsymbol{b}$. Its minimizer is $\boldsymbol{y}_* = \mathsf{T}^{-1}\boldsymbol{x}_*$. The steepest descent algorithm can be applied to (4.31) and, from an approximation $\boldsymbol{y}^{(k)}$ of the minimizer $\boldsymbol{y}_*$, an approximation $\boldsymbol{x}^{(k)}$ of $\boldsymbol{x}_*$ is obtained by the change of variable $\boldsymbol{x}^{(k)} = \mathsf{T}\boldsymbol{y}^{(k)}$. This approach is called *preconditioning*. By Theorem 4.17, the steepest descent method satisfies the following error estimate when applied to the function (4.31):

$$E_{k+1} \leqslant \left(\frac{\kappa_2(\mathsf{T}^T\mathsf{A}\mathsf{T})-1}{\kappa_2(\mathsf{T}^T\mathsf{A}\mathsf{T})+1}\right)^2 E_k, \qquad E_k = (\boldsymbol{y}^{(k)}-\boldsymbol{y}_*)^T\tilde{\mathsf{A}}(\boldsymbol{y}^{(k)}-\boldsymbol{y}_*),$$
$$= (\boldsymbol{x}^{(k)}-\boldsymbol{x}_*)^T\mathsf{A}(\boldsymbol{x}^{(k)}-\boldsymbol{x}_*).$$

Consequently, the convergence is faster than that of the usual steepest descent method if $\kappa_2(\mathsf{T}^T\mathsf{A}\mathsf{T}) < \kappa_2(\mathsf{A})$. The optimal change of coordinates is given by $\mathsf{T} = \mathsf{C}^{-T}$, where C is the factor of the Cholesky factorization of A as $\mathsf{C}\mathsf{C}^T$. Indeed, in this case

$$\mathsf{T}^T\mathsf{A}\mathsf{T} = \mathsf{C}^{-1}\mathsf{C}\mathsf{C}^T\mathsf{C}^{-T} = \mathsf{I} \qquad \Rightarrow \qquad \kappa_2(\mathsf{T}^T\mathsf{A}\mathsf{T}) = 1,$$

and the method converges in a single iteration! However, this iteration amounts to solving the linear system by direct Cholesky factorization of A. In practice, it is usual to define T from an approximation of the Cholesky factorization, such as the *incomplete Cholesky factorization*.

To conclude this section, we demonstrate that the change of variable from $\boldsymbol{x}$ to $\boldsymbol{y}$ need not be performed explicitly in practice. Indeed, one step of the steepest descent algorithm applied to function $\tilde{f}$ reads

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{y}^{(k)} - \tilde{\omega}_k(\tilde{\mathsf{A}}\boldsymbol{y}^{(k)}-\tilde{\boldsymbol{b}}), \qquad \tilde{\omega}_k = \frac{(\tilde{\mathsf{A}}\boldsymbol{y}^{(k)}-\tilde{\boldsymbol{b}})^T(\tilde{\mathsf{A}}\boldsymbol{y}^{(k)}-\tilde{\boldsymbol{b}})}{(\tilde{\mathsf{A}}\boldsymbol{y}^{(k)}-\tilde{\boldsymbol{b}})^T\tilde{\mathsf{A}}(\tilde{\mathsf{A}}\boldsymbol{y}^{(k)}-\tilde{\boldsymbol{b}})}.$$

Letting $\boldsymbol{x}^{(k)} = \mathsf{T}\boldsymbol{y}^{(k)}$, this equation can be rewritten as the following iteration:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \tilde{\omega}_k\boldsymbol{d}_k, \qquad \tilde{\omega}_k = \frac{\boldsymbol{d}_k^T\boldsymbol{r}^{(k)}}{\boldsymbol{d}_k^T\mathsf{A}\boldsymbol{d}_k}, \qquad \boldsymbol{d}_k = \mathsf{T}\mathsf{T}^T(\mathsf{A}\boldsymbol{x}^{(k)}-\boldsymbol{b}).$$

A comparison with (4.29) shows that the step size $\tilde{\omega}_k$ is such that $f(\boldsymbol{x}^{(k+1)})$ is minimized. This reasoning shows that the preconditioned conjugate gradient method amounts to choosing the direction $\boldsymbol{d}_k = \mathsf{T}\mathsf{T}^T\boldsymbol{r}^{(k)}$ at each iteration, instead of just $\boldsymbol{r}^{(k)}$, as is apparent in Algorithm 4. It is simple to check that $-\boldsymbol{d}_k$ is a descent direction for f:

$$-\nabla f(\boldsymbol{x})^T\Big(\mathsf{T}\mathsf{T}^T(\mathsf{A}\boldsymbol{x}-\boldsymbol{b})\Big) = -\Big(\mathsf{T}^T(\mathsf{A}\boldsymbol{x}-\boldsymbol{b})\Big)^T\Big(\mathsf{T}^T(\mathsf{A}\boldsymbol{x}-\boldsymbol{b})\Big) \leqslant 0.$$

Algorithm 4 Preconditioned steepest descent method

Pick ε, invertible T and initial $\boldsymbol{x}$
$\boldsymbol{r} \leftarrow \mathsf{A}\boldsymbol{x}-\boldsymbol{b}$
while $\|\boldsymbol{r}\| \geqslant \varepsilon\|\boldsymbol{b}\|$ **do**
    $\boldsymbol{d} \leftarrow \mathsf{T}\mathsf{T}^T\boldsymbol{r}$
    $\omega \leftarrow \boldsymbol{d}^T\boldsymbol{r}/\boldsymbol{d}^T\mathsf{A}\boldsymbol{d}$
    $\boldsymbol{x} \leftarrow \boldsymbol{x}-\omega\boldsymbol{d}$
    $\boldsymbol{r} \leftarrow \mathsf{A}\boldsymbol{x}-\boldsymbol{b}$
end while

Conjugate directions method

Definition 4.6 (Conjugate directions)**.** Let A be a symmetric positive definite matrix. Two vectors $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are called A-orthogonal or conjugate with respect to A if $\boldsymbol{d}_1^T \mathsf{A} \boldsymbol{d}_2 = 0$, i.e. if they are orthogonal for the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathsf{A}} = \boldsymbol{x}^T \mathsf{A} \boldsymbol{y}$.

Assume that $\boldsymbol{d}_0, \ldots, \boldsymbol{d}_{n-1}$ are n pairwise A-orthogonal nonzero directions. By Exercise 4.19, these vectors are linearly independent, and so they form a basis of $\mathbf{R}^n$. Consequently, for any initial guess $\boldsymbol{x}^{(0)}$, the vector $\boldsymbol{x}^{(0)} - \boldsymbol{x}_*$, where $\boldsymbol{x}_*$ is the solution to the linear system $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$, can be decomposed as

$$\boldsymbol{x}^{(0)} - \boldsymbol{x}_* = \alpha_0 \boldsymbol{d}_0 + \cdots + \alpha_{n-1} \boldsymbol{d}_{n-1}.$$

Taking the $\langle \bullet, \bullet \rangle_{\mathsf{A}}$ inner product of both sides with $\boldsymbol{d}_k$, with $k \in \{0, \ldots, n-1\}$, we obtain an expression for the scalar coefficient α_k:

$$\alpha_k = \frac{\boldsymbol{d}_k^T \mathsf{A}(\boldsymbol{x}^{(0)} - \boldsymbol{x}_*)}{\boldsymbol{d}_k^T \mathsf{A} \boldsymbol{d}_k} = \frac{\boldsymbol{d}_k^T (\mathsf{A}\boldsymbol{x}^{(0)} - \boldsymbol{b})}{\boldsymbol{d}_k^T \mathsf{A} \boldsymbol{d}_k}.$$

Therefore, calculating the expression of the coefficient does not require to know the exact solution $\boldsymbol{x}_*$, but only the residual $\boldsymbol{r}^{(0)}$! Given conjugate directions, the exact solution can be obtained as

$$\boldsymbol{x}_* = \boldsymbol{x}^{(0)} - \sum_{k=0}^{n-1} \alpha_k \boldsymbol{d}_k, \qquad \alpha_k = \frac{\boldsymbol{d}_k^T \boldsymbol{r}^{(0)}}{\boldsymbol{d}_k^T \mathsf{A} \boldsymbol{d}_k}. \tag{4.32}$$

If $\boldsymbol{x}^{(0)} = \mathbf{0}$, then $\boldsymbol{r}^{(0)} = -\boldsymbol{b}$ and this equations gives that

$$\boldsymbol{x}_* = \sum_{k=0}^{n-1} \frac{\boldsymbol{d}_k^T \boldsymbol{b}}{\boldsymbol{d}_k^T \mathsf{A} \boldsymbol{d}_k} \boldsymbol{d}_k = \left(\sum_{k=0}^{n-1} \frac{\boldsymbol{d}_k \boldsymbol{d}_k^T}{\boldsymbol{d}_k^T \mathsf{A} \boldsymbol{d}_k} \right) \boldsymbol{b},$$

which implies that that the inverse of A is given by

$$\mathsf{A}^{-1} = \sum_{k=0}^{n-1} \frac{\boldsymbol{d}_k \boldsymbol{d}_k^T}{\nu_k}, \qquad \nu_k = \boldsymbol{d}_k^T \mathsf{A} \boldsymbol{d}_k.$$

The conjugate directions method is illustrated in Algorithm 5. Its implementation is very similar to that of the steepest descent method, the only difference being that the descent direction at iteration k is given by $\boldsymbol{d}_k$ instead of $\boldsymbol{r}^{(k)}$. In particular, the step size at each iteration is such that $f(\boldsymbol{x}^{(k+1)})$ is minimized.

Algorithm 5 Conjugate directions method

Assuming $\boldsymbol{d}_0, \ldots, \boldsymbol{d}_{n-1}$ are given.
Pick initial $\boldsymbol{x}^{(0)}$
for k in $\{0, \ldots, n-1\}$ **do**
    $\boldsymbol{r}^{(k)} = \mathsf{A}\boldsymbol{x}^{(k)} - \boldsymbol{b}$
    $\omega_k = \boldsymbol{d}_k^T \boldsymbol{r}^{(k)} / \boldsymbol{d}_k^T \mathsf{A} \boldsymbol{d}_k$
    $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \omega_k \boldsymbol{d}_k$
end for

Let us now establish the connection between the Algorithm 5 and (4.32), which may not be immediately apparent because (4.32) involves only the initial residual $\mathsf{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}$, while the residual at the current iteration $\boldsymbol{r}^{(k)}$ is used in the algorithm.

Proposition 4.18 (Convergence of the conjugate directions method)**.** *The vector $\boldsymbol{x}^{(k)}$ obtained after k iterations of the conjugate directions method is given by*

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(0)} - \sum_{i=0}^{k-1} \alpha_i \boldsymbol{d}_i, \qquad \alpha_i = \frac{\boldsymbol{d}_i^T \boldsymbol{r}^{(0)}}{\boldsymbol{d}_i^T \mathsf{A} \boldsymbol{d}_i}. \tag{4.33}$$

In particular, the method converges in at most n iterations.

Proof. Let us denote by $\boldsymbol{y}^{(k)}$ the solution obtained after k steps of Algorithm 5. Our goal is to show that $\boldsymbol{y}^{(k)}$ coincides with $\boldsymbol{x}^{(k)}$ defined in (4.33). The result is trivial for $k = 0$. Reasoning by induction, we assume that it is true up to k. Then performing step $k+1$ of the algorithm gives

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{y}^{(k)} - \omega_k \boldsymbol{d}_k, \qquad \omega_k = \frac{\boldsymbol{d}_k^T \boldsymbol{r}^{(k)}}{\boldsymbol{d}_k^T \mathsf{A} \boldsymbol{d}_k}.$$

On the other hand, it holds from (4.33) that

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{d}_k, \qquad \alpha_k = \frac{\boldsymbol{d}_k^T \boldsymbol{r}^{(0)}}{\boldsymbol{d}_k^T \mathsf{A} \boldsymbol{d}_k}.$$

By the induction hypothesis, it holds that $\boldsymbol{y}^{(k)} = \boldsymbol{x}^{(k)}$, so in order to prove that $\boldsymbol{y}^{(k+1)} = \boldsymbol{x}^{(k+1)}$, it is sufficient to show that $\omega_k = \alpha_k$, i.e. that

$$\boldsymbol{d}_k^T \boldsymbol{r}^{(k)} = \boldsymbol{d}_k^T \boldsymbol{r}^{(0)} \quad \Leftrightarrow \quad \boldsymbol{d}_k^T (\boldsymbol{r}^{(k)} - \boldsymbol{r}^{(0)}) = 0 \quad \Leftrightarrow \quad \boldsymbol{d}_k^T \mathsf{A} (\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(0)}) = 0.$$

The latter equality is obvious from the A-orthonormality of the directions. □

Since ω_k in Algorithm 5 coincides with the expression in (4.29), the conjugate directions algorithm satisfies the following "local optimization" property: the iterate $\boldsymbol{x}^{(k+1)}$ minimizes f on the straight line $\omega \mapsto \boldsymbol{x}^{(k)} - \omega \boldsymbol{d}_k$. In contrast with the steepest descent method, however, the conjugate directions method also satisfies the following stronger property.

Proposition 4.19 (Optimality of the conjugate directions method)**.** *The iterate $\boldsymbol{x}^{(k)}$ is the minimizer of f over the set $\boldsymbol{x}^{(0)} + \mathcal{B}_k$, where $\mathcal{B}_k = \mathrm{Span}\{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_{k-1}\}$.*

Proof. By (4.32), it holds that

$$\boldsymbol{x}_* = \boldsymbol{x}^{(0)} - \sum_{i=0}^{n-1} \alpha_i \boldsymbol{d}_i, \qquad \alpha_i = \frac{\boldsymbol{d}_i^T \boldsymbol{r}^{(0)}}{\boldsymbol{d}_i^T \mathsf{A} \boldsymbol{d}_i}$$

On the other hand, any vector $\boldsymbol{y} \in \boldsymbol{x}^{(0)} + \mathcal{B}_k$ can be expanded as

$$\boldsymbol{y} = \boldsymbol{x}^{(0)} - \beta_0 \boldsymbol{d}_0 - \cdots - \beta_{k-1} \boldsymbol{d}_{k-1}.$$

Employing these two expressions, the formula for f in (4.27), and the A-orthogonality of the directions, we obtain

$$\begin{aligned} f(\boldsymbol{y}) &= \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x}_*)^T \mathsf{A}(\boldsymbol{y} - \boldsymbol{x}_*) - \frac{1}{2}\boldsymbol{x}_*^T \mathsf{A} \boldsymbol{x}_* \\ &= \frac{1}{2}\sum_{i=0}^{k-1} (\beta_i - \alpha_i)^2 \boldsymbol{d}_i^T \mathsf{A} \boldsymbol{d}_i + \frac{1}{2}\sum_{i=k}^{n-1} \alpha_i^2 \boldsymbol{d}_i^T \mathsf{A} \boldsymbol{d}_i - \frac{1}{2}\boldsymbol{x}_*^T \mathsf{A} \boldsymbol{x}_* \end{aligned}$$

This is minimized when $\beta_i = \alpha_i$ for all $i \in \{0, \dots, k-1\}$, in which case $\boldsymbol{y}$ coincides with the k -th iterate $\boldsymbol{x}^{(k)}$ of the conjugate directions method in view of Proposition 4.18. □

Remark 4.4. Let $\|\bullet\|_{\mathsf{A}}$ denote the norm induced by the inner product $\langle \bullet, \bullet \rangle_{\mathsf{A}}$. Since

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}_*\|_{\mathsf{A}} = \sqrt{2f(\boldsymbol{x}^{(k)}) + \boldsymbol{x}_*^T \mathsf{A} \boldsymbol{x}_*},$$

Proposition 4.19 shows that $\boldsymbol{x}^{(k)}$ minimizes the norm $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}_*\|_{\mathsf{A}}$ over $\boldsymbol{x}^{(0)} + \mathcal{B}_k$. This is not surprising since, by construction, the vector $\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(0)}$ is the orthogonal projection of $\boldsymbol{x}_* - \boldsymbol{x}^{(0)}$ onto $\mathcal{B}_k$, for the inner product $\langle \bullet, \bullet \rangle_{\mathsf{A}}$.

A corollary of (4.19) is that the gradient of f at $\boldsymbol{x}^{(k)}$, i.e. the residual $\boldsymbol{r}^{(k)} = \mathsf{A}\boldsymbol{x}^{(k)} - \boldsymbol{b}$, is orthogonal to any vector in $\{\boldsymbol{d}_0, \dots, \boldsymbol{d}_{k-1}\}$ for the usual Euclidean inner product. This can also be checked directly from the formula

$$\boldsymbol{x}^{(k)} - \boldsymbol{x}_* = \sum_{i=k}^{n-1} \alpha_i \boldsymbol{d}_i, \qquad \alpha_i = \frac{\boldsymbol{d}_i^T \boldsymbol{r}^{(0)}}{\boldsymbol{d}_i^T \mathsf{A} \boldsymbol{d}_i},$$

which follows directly from Proposition 4.18. Indeed, it holds that

$$\forall j \in \{0, \dots, k-1\}, \qquad \boldsymbol{d}_j^T \boldsymbol{r}^{(k)} = \boldsymbol{d}_j \mathsf{A}(\boldsymbol{x}^{(k)} - \boldsymbol{x}_*) = \sum_{i=k}^{n-1} \alpha_i \boldsymbol{d}_j^T \mathsf{A} \boldsymbol{d}_i = 0. \tag{4.34}$$

The conjugate gradient method

In the previous section, we showed that, given n conjugate directions, the solution to the linear system $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ can be obtained in a finite number of iterations using Algorithm 5. The conjugate gradient method can be viewed as a particular case of the conjugate directions method. Instead of assuming that the conjugate directions are given, they are constructed iteratively as part of the algorithm. Given an initial guess $\boldsymbol{x}^{(0)}$, the first direction is the residual $\boldsymbol{r}^{(0)}$, which coincides

with the gradient of f at $\boldsymbol{x}^{(0)}$. The directions employed for the next iterations are obtained by applying the Gram-Schmidt process to the residuals. More precisely, given conjugate directions $\boldsymbol{d}_0, \ldots, \boldsymbol{d}_{k-1}$, and letting $\boldsymbol{x}^{(k)}$ denote the k-th iterate of the conjugate directions method, the direction $\boldsymbol{d}_k$ is obtained by

$$\boldsymbol{d}_k = \boldsymbol{r}^{(k)} - \sum_{i=0}^{k-1} \frac{\boldsymbol{d}_i^T \mathsf{A} \boldsymbol{r}^{(k)}}{\boldsymbol{d}_i^T \mathsf{A} \boldsymbol{d}_i} \boldsymbol{d}_i, \qquad \boldsymbol{r}^{(k)} = \mathsf{A}\boldsymbol{x}^{(k)} - \boldsymbol{b}. \tag{4.35}$$

It is simple to check that $\boldsymbol{d}_k$ is indeed A-orthogonal to $\boldsymbol{d}_i$ for $i \in \{0, \ldots, k-1\}$, and that $\boldsymbol{d}_k$ is nonzero if $\boldsymbol{r}^{(k)}$ is nonzero. To prove the latter claim, we can take the Euclidean inner product of both sides with $\boldsymbol{r}^{(k)}$ and use Proposition 4.19 to deduce that

$$\boldsymbol{d}_k^T \boldsymbol{r}^{(k)} = (\boldsymbol{r}^{(k)})^T \boldsymbol{r}^{(k)} > 0. \tag{4.36}$$

Note also that since the directions are obtained by applying the Gram–Schmidt process to the residuals, it holds that

$$\forall k \in \{0, \ldots, n-1\}, \qquad \mathcal{B}_{k+1} := \operatorname{Span}\{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_k\} = \operatorname{Span}\{\boldsymbol{r}^{(0)}, \ldots, \boldsymbol{r}^{(k)}\}. \tag{4.37}$$

The following result characterizes precisely the subspace $\mathcal{B}_{k+1}$.

Proposition 4.20. *Assume that $\|\boldsymbol{r}^{(k)}\| \neq 0$ for all $k < m \leqslant n$. Then it holds that*

$$\forall k \in \{0, \ldots m\}, \quad \operatorname{Span}\{\boldsymbol{r}^{(0)}, \boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(k)}\} = \operatorname{Span}\{\boldsymbol{r}^{(0)}, \mathsf{A}\boldsymbol{r}^{(0)}, \ldots, \mathsf{A}^k \boldsymbol{r}^{(0)}\} \tag{4.38}$$

The subspace on the right-hand side is called a Krylov subspace.

Proof. The result is clear for $k = 0$. Reasoning by induction, we prove that if the result is true up to $k < m$, then it is also true for $k + 1$. A simple calculation gives that

$$\begin{aligned} \boldsymbol{r}^{(k+1)} &= \mathsf{A}\big(\boldsymbol{x}^{(k)} - \omega_k \boldsymbol{d}_k\big) - \boldsymbol{b} \\ &= \boldsymbol{r}^{(k)} - \omega_k \mathsf{A}\boldsymbol{d}_k. \end{aligned} \tag{4.39}$$

From (4.35), we deduce that

$$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \omega_k \mathsf{A} \left(\boldsymbol{r}^{(k)} - \sum_{i=0}^{k-1} \frac{\boldsymbol{d}_i^T \mathsf{A} \boldsymbol{r}^{(k)}}{\boldsymbol{d}_i^T \mathsf{A} \boldsymbol{d}_i} \boldsymbol{d}_i \right).$$

By (4.37) and the induction hypothesis, the bracketed expression on the right-hand side belongs to $\mathcal{B}_{k+1}$, so the inclusion $\subset$ in (4.38) is clear. The inclusion $\supset$ then follows from the fact the dimension of the subspace

$$\operatorname{Span}\{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_k\} = \operatorname{Span}\{\boldsymbol{r}^{(0)}, \ldots, \boldsymbol{r}^{(k)}\}$$

is equal to $k + 1$. □

It appears from (4.35) that the cost of calculating a new direction grows linearly with the iteration index. In fact, it turns out that only the last term in the sum is nonzero, and so the cost of calculating a new direction does not grow with the iteration index k. Indeed, notice that if $i \leqslant k-2$, then

$$\boldsymbol{d}_i^T \mathsf{A}\boldsymbol{r}^{(k)} = (\mathsf{A}\boldsymbol{d}_i)^T \mathsf{A}\boldsymbol{r}^{(k)} = 0,$$

because $\mathsf{A}\boldsymbol{d}_i \in \mathcal{B}_{i+2} \subset \mathcal{B}_k$ by Proposition 4.20, and $\boldsymbol{r}^{(k)}$ orthogonal to $\mathcal{B}_k$ for the Euclidean inner product by (4.34). This observation leads to Algorithm 6.

Algorithm 6 Conjugate gradient method

Pick initial $\boldsymbol{x}^{(0)}$
$\boldsymbol{d}_0 = \boldsymbol{r}^{(0)} = \mathsf{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}$
for k in $\{0, \dots, n-1\}$ **do**
    if $\|\boldsymbol{r}^{(k)}\| = 0$ **then**
        Stop
    end if
    $\omega_k = \boldsymbol{d}_k^T \boldsymbol{r}^{(k)} / \boldsymbol{d}_k^T \mathsf{A}\boldsymbol{d}_k$
    $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \omega_k \boldsymbol{d}_k$
    $\boldsymbol{r}^{(k+1)} = \mathsf{A}\boldsymbol{x}^{(k+1)} - \boldsymbol{b}$
    $\beta_k = \boldsymbol{d}_k^T \mathsf{A}\boldsymbol{r}^{(k+1)} / \boldsymbol{d}_k^T \mathsf{A}\boldsymbol{d}_k$.
    $\boldsymbol{d}_{k+1} = \boldsymbol{r}^{(k+1)} - \beta_k \boldsymbol{d}_k$.
end for

Although the conjugate gradient method converges in a finite number of iterations, performing n iterations for very large systems would require an excessive computational cost, and so it is sometimes desirable to stop iterating when the residual is sufficiently small. To conclude this section, we study the convergence of the method.

Theorem 4.21 (Convergence of the conjugate gradient method)**.** *The error for the conjugate gradient method, measured as*

$$E_k := (\boldsymbol{x}^{(k)} - \boldsymbol{x}_*)^T \mathsf{A}(\boldsymbol{x}^{(k)} - \boldsymbol{x}_*),$$

satisfies the following inequality:

$$\forall q_k \in \mathbf{P}(k), \qquad E_{k+1} \leqslant \max_{1 \leqslant i \leqslant n} \big(1 + \lambda_i q_k(\lambda_i)\big)^2 E_0. \tag{4.40}$$

Here $\mathbf{P}(k)$ is the vector space of polynomials of degree less than or equal to k.

Proof. In view of Proposition 4.20, the iterate $\boldsymbol{x}^{(k+1)}$ can be written as

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(0)} + \sum_{i=0}^{k} \alpha_i \mathsf{A}^i \boldsymbol{r}^{(0)} = \boldsymbol{x}^{(0)} + p_k(\mathsf{A})\boldsymbol{r}^{(0)},$$

where p_k is a polynomial of degree k. By Proposition 4.19, p_k is in fact the polynomial of degree k such that $f(\boldsymbol{x}^{(k+1)})$ is minimized, and thus also E_{k+1} by (4.27). Noting that

$$\begin{aligned}\boldsymbol{x}^{(k+1)} - \boldsymbol{x}_* &= \boldsymbol{x}^{(0)} - \boldsymbol{x}_* + p_k(\mathsf{A})\boldsymbol{r}^{(0)} = \boldsymbol{x}^{(0)} - \boldsymbol{x}_* + p_k(\mathsf{A})\mathsf{A}(\boldsymbol{x}^{(0)} - \boldsymbol{x}_*) \\ &= \Big(\mathsf{I} + \mathsf{A}p_k(\mathsf{A})\Big)(\boldsymbol{x}^{(0)} - \boldsymbol{x}_*),\end{aligned}$$

we deduce that

$$\forall q_k \in \mathbf{P}(k), \qquad E_{k+1} \leqslant (\boldsymbol{x}^{(0)} - \boldsymbol{x}_*)^T \mathsf{A}\Big(\mathsf{I} + \mathsf{A}q_k(\mathsf{A})\Big)^2 (\boldsymbol{x}^{(0)} - \boldsymbol{x}_*).$$

In order to exploit this inequality, it is useful to diagonalize A as $\mathsf{A} = \mathsf{Q}\mathsf{D}\mathsf{Q}^T$, for an orthogonal matrix Q and a diagonal matrix D. Since $q_k(\mathsf{A}) = \mathsf{Q}q_k(\mathsf{D})\mathsf{Q}^T$ for all $q_k \in \mathbf{P}(k)$, it holds that

$$\begin{aligned}\forall q_k \in \mathbf{P}(k), \qquad E_{k+1} &= \Big(\mathsf{Q}^T(\boldsymbol{x}^{(0)} - \boldsymbol{x}_*)\Big)^T \mathsf{D}\Big(\mathsf{I} + \mathsf{D}q_k(\mathsf{D})\Big)^2 \Big(\mathsf{Q}^T(\boldsymbol{x}^{(0)} - \boldsymbol{x}_*)\Big) \\ &\leqslant \max_{1 \leqslant i \leqslant n} \Big(1 + \lambda_i q_k(\lambda_i)\Big)^2 \underbrace{\Big(\mathsf{Q}^T(\boldsymbol{x}^{(0)} - \boldsymbol{x}_*)\Big)^T \mathsf{D}\Big(\mathsf{Q}^T(\boldsymbol{x}^{(0)} - \boldsymbol{x}_*)\Big)}_{E_0},\end{aligned}$$

which completes the proof. □

A corollary of Theorem 4.21 is that, if A has $m \leqslant n$ distinct eigenvalues, then the conjugate gradient method converges in at most m iterations. Indeed, in this case we can take

$$q_{m-1}(\lambda) = \frac{1}{\lambda}\left(\frac{(\lambda_1 - \lambda)\dots(\lambda_m - \lambda)}{\lambda_1 \dots \lambda_m} - 1\right).$$

It is simple to check that the right-hand side is indeed a polynomial, and that $1 + \lambda_i q_{m-1}(\lambda_i) = 0$ for all eigenvalues of A.

In general, finding the polynomial that minimizes the right-hand side of (4.40) is not possible, because the eigenvalues of A are unknown. However, it is possible to derive from this equation an error estimate with an explicit dependence on the condition number $\kappa = \kappa_2(\mathsf{A})$.

Theorem 4.22. *It holds that*

$$\forall k \geqslant 0, \qquad E_k \leqslant 4\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{2k} E_0,$$

Proof. Theorem 4.21 implies that

$$\forall q_k \in \mathbf{P}(k), \qquad E_{k+1} \leqslant \max_{\lambda \in [\lambda_1, \lambda_n]} \Big(1 + \lambda q_k(\lambda)\Big)^2 E_0,$$

where λ_1 and λ_n are the minimum and maximum eigenvalues of A. Notice that

$$\Big\{1 + \lambda q_k \colon q_k \in \mathbf{P}(k)\Big\} = \Big\{p_k \colon p_k \in \mathbf{P}(k+1) \ \text{ and } \ p_k(0) = 1\Big\}$$

Therefore, it follows from Exercise C.7 that the right-hand side is minimized when

$$1 + \lambda q_k(\lambda) = \frac{T_{k+1}\left(\frac{\lambda_n+\lambda_1-2\lambda}{\lambda_n-\lambda_1}\right)}{T_{k+1}\left(\frac{\lambda_n+\lambda_1}{\lambda_n-\lambda_1}\right)}, \tag{4.41}$$

where T_{k+1} is the *Chebyshev* polynomial of degree $k+1$, see (C.1). We recall that $|T_{k+1}(x)| \leqslant 1$ for all $x \in [-1, 1]$. Consequently, by the expression of Chebyshev polynomials given in Exercise C.3, the following inequality holds true for all $\lambda \in [\lambda_1, \lambda_n]$:

$$\begin{aligned}\left|1 + \lambda q_k(\lambda)\right| \leqslant \frac{1}{T_{k+1}\left(\frac{\lambda_n+\lambda_1}{\lambda_n-\lambda_1}\right)} &= 2\left(\left(r + \sqrt{r^2-1}\right)^{k+1} + \left(r - \sqrt{r^2-1}\right)^{k+1}\right)^{-1}, \\ &= 2\left(\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{k+1} + \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{k+1}\right)^{-1}.\end{aligned}$$

where $r = \frac{\lambda_n+\lambda_1}{\lambda_n-\lambda_1}$. Since the first term in the bracket converges to zero as $k \to \infty$, it is natural to bound this expression by keeping only the second term, which after simple algebraic manipulations leads to

$$\forall \lambda \in [\lambda_1, \lambda_n], \qquad \left|1 + \lambda q_k(\lambda)\right| \leqslant 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{k+1}.$$

From this inequality, the statement of the theorem follows immediately. □

4.4 Exercises

Exercise 4.1. *In the simple case where* A *is symmetric, find values of* $\boldsymbol{x}$, $\boldsymbol{b}$ *and* $\Delta\boldsymbol{b}$ *for which the inequality* (4.2) *is in fact an equality?*

Exercise 4.2 (Inverse of Gaussian transformation)**.** *Prove the formula* (4.8)*.*

Exercise 4.3. *Prove that the product of two lower triangular matrices is lower triangular.*

Exercise 4.4. *Assume that* $\mathsf{A} \in \mathbf{R}^{n\times n}$ *is positive definite, i.e. that*

$$\forall \boldsymbol{x} \in \mathbf{R}^n \backslash \{\mathbf{0}_n\}, \qquad \boldsymbol{x}^T \mathsf{A} \boldsymbol{x} > 0.$$

Show that all the principal submatrices of A *are nonsingular.*

Exercise 4.5. *Implement the backward substitution algorithm for solving* $\mathsf{U}x = y$*. What is the computational cost of the algorithm?*

Exercise 4.6. *Compare the condition number of the matrices* L *and* U *with and without partial pivoting. For testing, use a matrix with pseudo-random entries generated as follows*

```
import Random
# Set the seed so that the code is deterministic
Random.seed!(0)
n = 1000 # You can change this parameter
A = randn(n, n)
```

Solution. See the Jupyter notebook for this chapter. △

Exercise 4.7. *Write a code for calculating the Cholesky factorization of a symmetric positive definite matrix* A *by comparing the entries of the product* CC^T *with those of the matrix* A*. What is the associated computational cost, and how does it compare with that of the* LU *factorization?*

Extra credit: *… if your code is able to exploit the potential banded structure of the matrix passed as argument for better efficiency. Specifically, your code will be tested with a matrix is of the type* `BandedMatrix` *defined in the* ***`BandedMatrices.jl`*** *package, which you will need to install. The following code can be useful for testing purposes.*

```
import BandedMatrices
import LinearAlgebra

function cholesky(A)
    m, n = size(A)
    m != n && error("Matrix must be square")
    # Convert to banded matrix
    B = BandedMatrices.BandedMatrix(A)
    B.u != B.l && error("Matrix must be symmetric")
    # --> Your code comes here <--
end
n, u, l = 20000, 2, 2
A = BandedMatrices.brand(n, u, l)
A = A*A'
# so that A is symmetric and positive definite (with probability 1).
C = @time cholesky(A)
LinearAlgebra.norm(C*C' - A, Inf)
```

For information, my code takes about 1 second to run with the parameters given here.

Exercise 4.8 (Matrix square root)**.** *Let* $A \in \mathbf{R}^{n\times n}$ *be a symmetric positive definite matrix. Show that* A *has a positive definite square root, i.e. that there exists a symmetric matrix* B *such that* $BB = A$*.*

Solution. Since A is symmetric, there exist a diagonal matrix D and an orthogonal matrix Q such that $A = QDQ^T$. Let $D^{1/2}$ denote the diagonal matrix obtained by applying the square root function to the entries of D, and notice that $D^{1/2}D^{1/2} = D$. Then it holds that

$$\mathsf{A} = \left(\mathsf{Q}\mathsf{D}^{1/2}\mathsf{Q}^T\right)\left(\mathsf{Q}\mathsf{D}^{1/2}\mathsf{Q}^T\right).$$

The matrix $\mathsf{A}^{1/2} := \mathsf{Q}\mathsf{D}^{1/2}\mathsf{Q}^T$ is a square root of the matrix A, in the sense that $\mathsf{A}^{1/2}\mathsf{A}^{1/2}$, and it is positive definite because the diagonal elements of $\mathsf{D}^{1/2}$ are strictly positive. △

Exercise 4.9. *Show that if* A *is row or column diagonally dominant, then* A *is invertible.*

Exercise 4.10. *Let* T *be a nonsingular matrix. Show that*

$$\|\mathsf{A}\|_{\mathsf{T}} := \|\mathsf{T}^{-1}\mathsf{A}\mathsf{T}\|_2$$

defines a matrix norm induced by a vector norm.

Exercise 4.11. *Let* $\mathsf{A} \in \mathbf{R}^{n\times n}$ *be a symmetric positive definite matrix. Show that the functional*

$$\|\bullet\|_{\mathsf{A}} \colon \boldsymbol{x} \mapsto \sqrt{\boldsymbol{x}^T\mathsf{A}\boldsymbol{x}}$$

defines a norm on $\mathbf{R}^n$*.*

Solution. We need to prove that the three axioms of a norm are satisfied:

- (**Positivity**) Since A is positive definite, it holds that $\|\boldsymbol{x}\|_{\mathsf{A}} > 0$ for any $\boldsymbol{x} \in \mathbf{R}^n \setminus \{\mathbf{0}\}$.
- (**Homogeneity**) It is clear that $\|c\boldsymbol{x}\|_{\mathsf{A}} = |c|\ \|\boldsymbol{x}\|_{\mathsf{A}}$ for any $c \in \mathbf{R}$.
- (**Triangle inequality**) Let $\mathsf{A}^{1/2}$ denote the positive definite square root of A, which exists by Exercise 4.8. Then

  $$\|\boldsymbol{x}\|_{\mathsf{A}} = \|\mathsf{A}^{1/2}\boldsymbol{x}\|_2.$$

  The triangle inequality for $\|\bullet\|_{\mathsf{A}}$ then follows from that for $\|\bullet\|_2$:

  $$\|\boldsymbol{x}+\boldsymbol{y}\|_{\mathsf{A}} = \|\mathsf{A}^{1/2}\boldsymbol{x}+\mathsf{A}^{1/2}\boldsymbol{y}\|_2 \leqslant \|\mathsf{A}^{1/2}\boldsymbol{x}\| + \|\mathsf{A}^{1/2}\boldsymbol{y}\|_2 = \|\boldsymbol{x}\|_{\mathsf{A}} + \|\boldsymbol{y}\|_{\mathsf{A}}.$$

Another option for solving this exercise is to show that

$$\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathsf{A}} := \boldsymbol{x}^T\mathsf{A}\boldsymbol{y}$$

defines an inner product, with induced norm given by $\|\bullet\|_{\mathsf{A}}$. △

Exercise 4.12. *Show that the residual satisfies the equation*

$$\boldsymbol{r}^{(k+1)} = \mathsf{N}\mathsf{M}^{-1}\boldsymbol{r}^{(k)} = (\mathsf{I} - \mathsf{A}\mathsf{M}^{-1})\boldsymbol{r}^{(k)}.$$

Exercise 4.13. *Show that, if* A *and* B *are two square matrices, then* $\rho(\mathsf{A}\mathsf{B}) = \rho(\mathsf{B}\mathsf{A})$*.*

Exercise 4.14. *Is* $\rho(\bullet)$ *a norm? Prove or disprove.*

Exercise 4.15. *Prove that, if* A *is a diagonal matrix, then*

$$\|\mathsf{A}\|_1 = \|\mathsf{A}\|_2 = \|\mathsf{A}\|_\infty = \rho(\mathsf{A}).$$

Exercise 4.16. *Show that, for any matrix norm $\|\bullet\|$ induced by a vector norm,*

$$\rho(\mathsf{A}) \leqslant \|\mathsf{A}\|.$$

Exercise 4.17. *Let $\|\bullet\|$ denote the Euclidean vector norm on $\mathbf{R}^n$. We define in Appendix A the induced matrix norm as*

$$\|\mathsf{A}\| = \sup\Big\{\|\mathsf{A}\boldsymbol{x}\| \colon \|\boldsymbol{x}\| \leqslant 1\Big\}.$$

Show from this definition that, if A is symmetric and positive definite, then

$$\|\mathsf{A}\| = \|\mathsf{A}\|_* := \sup\Big\{|\boldsymbol{x}^T\mathsf{A}\boldsymbol{x}| \colon \|\boldsymbol{x}\| \leqslant 1\Big\}.$$

Solution. By the Cauchy–Schwarz inequality and the definition of $\|\mathsf{A}\|$, it holds that

$$\forall \boldsymbol{x} \in \mathbf{R}^n \text{ with } \|\boldsymbol{x}\| \leqslant 1, \qquad |\boldsymbol{x}^T\mathsf{A}\boldsymbol{x}| \leqslant \|\boldsymbol{x}\|\|\mathsf{A}\boldsymbol{x}\| \leqslant \|\boldsymbol{x}\|\|\mathsf{A}\|\|\boldsymbol{x}\| \leqslant \|\mathsf{A}\|.$$

This shows that $\|\mathsf{A}\|_* \leqslant \|\mathsf{A}\|$. Conversely, letting B denote a matrix square root of A (see Exercise 4.8), we have

$$\begin{aligned}\forall \boldsymbol{x} \in \mathbf{R}^n \text{ with } \|\boldsymbol{x}\| \leqslant 1, \qquad \|\mathsf{A}\boldsymbol{x}\| &= \sqrt{\boldsymbol{x}^T\mathsf{A}^T\mathsf{A}\boldsymbol{x}} = \sqrt{(\mathsf{B}\boldsymbol{x})^T\mathsf{B}\mathsf{B}(\mathsf{B}\boldsymbol{x})} = \sqrt{(\mathsf{B}\boldsymbol{x})^T\mathsf{A}(\mathsf{B}\boldsymbol{x})} \\ &= \|\mathsf{B}\boldsymbol{x}\|\sqrt{\boldsymbol{y}^T\mathsf{A}\boldsymbol{y}}, \qquad \boldsymbol{y} = \frac{\mathsf{B}\boldsymbol{x}}{\|\mathsf{B}\boldsymbol{x}\|}.\end{aligned}$$

It holds that $\|\mathsf{B}\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T\mathsf{A}\boldsymbol{x}} \leqslant \sqrt{\|\mathsf{A}\|_*}$. In addition $\|\boldsymbol{y}\| = 1$, so the expression inside the square root is bounded from above by $\|\mathsf{A}\|_*$, which enables to conclude the proof. △

Exercise 4.18. *Prove that, if the matrix A is strictly diagonally dominant (by rows or columns), then the Gauss–Seidel method converges, i.e. $\rho(\mathsf{M}^{-1}\mathsf{N}) < 1$. You can use the same approach as in the proof of Proposition 4.11.*

Exercise 4.19. *Let $\mathsf{A} \in \mathbf{R}^{n\times n}$ denote a symmetric positive definite matrix, and assume that the vectors $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n$ are pairwise A-orthogonal directions. Show that $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n$ are linearly independent.*

Exercise 4.20 (Steepest descent algorithm)**.** *Consider the linear system*

$$\mathsf{A}\boldsymbol{x} := \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} =: \boldsymbol{b}. \tag{4.42}$$

- *Show that* A *is positive definite.*
- *Draw the contour lines of the function*

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\mathsf{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}.$$

- *Plot the contour lines of* f *in Julia using the function* `contourf` *from the package* `Plots`.
- *Using Theorem 4.17, estimate the number* K *of iterations of the steepest descent algorithm required in order to guarantee that* $E_K \leqslant 10^{-8}$*, when starting from the vector* $\boldsymbol{x}^{(0)} = (2\ 3)^T$.
- *Implement the steepest descent method for finding the solution to (4.42), and plot the iterates as linked dots over the filled contour of* f.
- *Plot the error* E_k *as a function of the iteration index, using a linear scale for the* x *axis and a logarithmic scale for the* y *axis.*

Exercise 4.21. *Compute the number of floating point operations required for performing one iteration of the conjugate gradient method, assuming that the matrix* A *contains* $\alpha \ll n$ *nonzero elements per row.*

Exercise 4.22 (Solving the Poisson equation over a rectangle)**.** *We consider in this exercise Poisson's equation in the domain* $\Omega = (0,2) \times (0,1)$*, equipped with homogeneous Dirichlet boundary conditions:*

$$\begin{aligned} -\triangle f(x,y) &= b(x,y), \qquad x \in \Omega, \\ f(x) &= 0, \qquad\quad x \in \partial\Omega. \end{aligned}$$

The right-hand side is

$$b(x,y) = \sin(4\pi x) + \sin(2\pi y).$$

A number of methods can be employed in order to discretize this partial differential equation. After discretization, a finite-dimensional linear system of the form $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ *is obtained. A Julia function for calculating the matrix* A *and the vector* $\boldsymbol{b}$ *using the finite difference method is given to you on the course website, as well as a function to plot the solution. The goal of this exercise is to solve the linear system using the conjugate gradient method. Use the same stopping criterion as in Exercise 4.25.*

Exercise 4.23. *Show that if* $\mathsf{A} \in \mathbf{R}^{n\times n}$ *is nonsingular, then the solution to the equation* $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ *belongs to the Krylov subspace*

$$\mathcal{K}_n(\mathsf{A}, \boldsymbol{b}) = \mathrm{Span}\Big\{\boldsymbol{b}, \mathsf{A}\boldsymbol{b}, \mathsf{A}^2\boldsymbol{b}, \ldots, \mathsf{A}^{n-1}\boldsymbol{b}\Big\}.$$

Exercise 4.24. *Write a function* `lu(A)` *for calculating the* LU *decomposition of a square matrix* $\mathsf{A} \in \mathbf{R}^{n\times n}$*, with* L *unit lower triangular and* U *upper triangular, not by Gaussian*

elimination but by comparing the entries of the product LU *with those of* A*. To this end, one option is to compare the entries one by one in the order* $(1,1)$*,* $(1,2)$*, …,* $(1,n)$*,* $(2,1)$*,* $(2,2)$*, …, i.e. row by row starting from the top. For example,*

- *Comparing the entry* $(1,k)$ *with* $k \in \{1,\ldots,n\}$ *gives*

$$\ell_{11}u_{1k} = a_{1k}.$$

  Since $\ell_{11} = 1$ *as* L *is unit lower triangular, this implies that* $u_{1k} = a_{1k}$*.*

- *Comparing the entry* $(2,1)$ *gives*

$$\ell_{21}u_{11} = a_{21}$$

  and so $\ell_{21} = a_{21}/u_{11}$*.*

- *Comparing the entry* $(2,k)$ *with* $k \in \{2,\ldots,n\}$ *gives*

$$\ell_{21}u_{1k} + \ell_{22}u_{2k} = a_{2k}.$$

  Given the previous items, the only unknown in this equation is u_{2k}*.*

- *Comparing the entry* $(3,1)$ *gives*

$$\ell_{31}u_{11} = a_{31},$$

  and so $\ell_{31} = a_{31}/u_{11}$*.*

- *Comparing the entry* $(3,2)$ *gives*

$$\ell_{31}u_{12} + \ell_{32}u_{22} = a_{32},$$

  Given the previous items, the only unknown in this equation is ℓ_{32}*.*

- *Comparing the entry* $(3,k)$ *with* $k \in \{3,\ldots,n\}$ *gives*

$$\ell_{31}u_{1k} + \ell_{32}u_{2k} + \ell_{33}u_{3k} = a_{3k},$$

  Given the previous items, the only unknown in this equation is u_{3k}*.*

Notice that a pattern seems to be emerging: when going through the entries row by row starting from the top left corner of the matrix, comparing the entry (i,j) *provides an equation for* ℓ_{ij} *if* $j < i$*, and an equation for* u_{ij} *if* $j \geqslant i$*. Do not use any external package for this exercise.*
Extra credit: *… if your code is able to exploit the potential banded structure of the matrix passed as argument for better efficiency. Specifically, your code will be tested with a matrix created as follows*

```
b, n = 5, 10000
A = [abs(i-j) <= b ? rand() : 0.0 for i in 1:n, j in 1:n]
```

Exercise 4.25. *Implement an iterative method based on a splitting for finding a solution to the following linear system on* $\mathbf{R}^n$.

$$\frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}, \qquad h = \frac{1}{n+1}.$$

Plot the norm of the residual as a function of the iteration index. Use as stopping criterion the condition

$$\|\boldsymbol{r}^{(k)}\| \leqslant \varepsilon \|\boldsymbol{b}\|, \qquad \varepsilon = 10^{-8}.$$

As initial guess, use a vector of zeros. The code will be tested with $n = 500$. *Do not use any library (except for plotting), and do not use the backslash operator.*

Exercise 4.26. *Find a formula for the optimal value of* ω *in the relaxation method given* n, *for the linear system in Exercise 4.25. The proof of Proposition 4.15, as well as the formula (4.25) for the eigenvalues of a tridiagonal matrix, are useful to this end.*

Solution. Corollary 4.13 and Proposition 4.14 imply that a sufficient and necessary condition for convergence, when A is Hermitian and positive definite, is that $\omega \in (0, 2)$. Let $M_\omega = \frac{1}{\omega}\mathsf{D} + \mathsf{L}$ and $N_\omega = \frac{1-\omega}{\omega}\mathsf{D} - \mathsf{U}$. A nonzero scalar $\lambda \in \mathbf{C}$ is an eigenvalue of $\mathsf{M}_\omega^{-1}\mathsf{N}_\omega$ if and only if

$$\det(\mathsf{M}_\omega^{-1}\mathsf{N}_\omega - \lambda \mathsf{I}) = 0 \quad \Leftrightarrow \quad \det(\mathsf{M}_\omega^{-1}) \det(\mathsf{N}_\omega - \lambda \mathsf{M}_\omega) = 0 \quad \Leftrightarrow \quad \det(\lambda \mathsf{M}_\omega - \mathsf{N}_\omega) = 0.$$

Substituting the expressions of M_ω and N_ω, we obtain that this condition can be equivalently rewritten as

$$\det\left(\lambda \mathsf{L} + \left(\frac{\lambda + \omega - 1}{\omega}\right)\mathsf{D} + \mathsf{U}\right) = 0 \quad \Leftrightarrow \quad \det\left(\sqrt{\lambda}\mathsf{L} + \left(\frac{\lambda + \omega - 1}{\omega}\right)\mathsf{D} + \sqrt{\lambda}\mathsf{U}\right) = 0$$

where we used (4.24) for the last equivalence. The equality of the determinants in these two equations is valid for $\sqrt{\lambda}$ denoting either of the two complex square roots of λ. This condition is equivalent to

$$\det\left(\mathsf{L} + \left(\frac{\lambda + \omega - 1}{\sqrt{\lambda}\omega}\right)\mathsf{D} + \mathsf{U}\right) = 0.$$

We recognize from the proof of Proposition 4.15 that this condition is equivalent to

$$\frac{\lambda + \omega - 1}{\sqrt{\lambda}\omega} \in \text{spectrum}(\mathsf{M}_\mathcal{J}^{-1}\mathsf{N}_\mathcal{J}).$$

In other words, for any $(\lambda, \mu) \in \mathbf{C}^2$ such that

$$\frac{(\lambda + \omega - 1)^2}{\lambda \omega^2} = \mu^2, \tag{4.43}$$

it holds that $\mu \in \text{spectrum}(\mathsf{M}_{\mathcal{J}}^{-1}\mathsf{N}_{\mathcal{J}})$ if and only if $\lambda \in \text{spectrum}(\mathsf{M}_{\omega}^{-1}\mathsf{N}_{\omega})$. By (4.25), the eigenvalues of $\mathsf{M}_{\mathcal{J}}^{-1}\mathsf{N}_{\mathcal{J}}$ are real and given by

$$\mu_j = \cos\Big(\frac{j\pi}{n+1}\Big), \qquad 1 \leqslant j \leqslant n. \tag{4.44}$$

Rearranging (4.43), we find

$$\lambda^2 + \lambda(2(\omega-1) - \omega^2\mu^2) + (\omega-1)^2 = 0.$$

For given $\omega \in (0,2)$ and $\mu \in \mathbf{R}$, this is a quadratic equation for λ with solutions

$$\lambda_\pm = \left(\frac{\omega^2\mu^2}{2} + 1 - \omega\right) \pm \omega\mu\sqrt{\frac{\omega^2\mu^2}{4} + 1 - \omega},$$

Since the first bracket is positive when the argument of the square root is positive, it is clear that

$$\max\big\{|\lambda_-|, |\lambda_+|\big\} = \left|\frac{\omega^2\mu^2}{2} + 1 - \omega + \omega|\mu|\sqrt{\frac{\omega^2\mu^2}{4} + 1 - \omega}\right|.$$

Combining this with (4.44), we deduce that the spectral radius of $\mathsf{M}_{\omega}^{-1}\mathsf{N}_{\omega}$ is given by

$$\rho(\mathsf{M}_{\omega}^{-1}\mathsf{N}_{\omega}) = \max_{j\in\{1,\dots,n\}}\left|\frac{\omega^2\mu_j^2}{2} + 1 - \omega + \omega|\mu_j|\sqrt{\frac{\omega^2\mu_j^2}{4} + 1 - \omega}\right|. \tag{4.45}$$

We wish to minimize this expression over the interval $\omega \in (0,2)$. While this can be achieved by algebraic manipulations, we content ourselves here with graphical exploration. Figure 4.3 depicts the amplitude of the modulus in (4.45) for different values of μ. It is apparent that, for given ω, the modulus increases as μ increases, which suggests that

$$\rho(\mathsf{M}_{\omega}^{-1}\mathsf{N}_{\omega}) = \left|\frac{\omega^2\mu_*^2}{2} + 1 - \omega + \omega|\mu_*|\sqrt{\frac{\omega^2\mu_*^2}{4} + 1 - \omega}\right|, \qquad \mu_* = \rho(\mathsf{M}_{\mathcal{J}}^{-1}\mathsf{N}_{\mathcal{J}}). \tag{4.46}$$

The figure also suggests that for a given value of μ, the modulus is minimized at the discontinuity of the first derivative, which occurs when the argument of the square root is zero. We conclude that the optimal ω satisfies

$$\frac{\omega_{\text{opt}}^2\mu_*^2}{4} + 1 - \omega_{\text{opt}} = 0 \quad \underset{\omega<2}{\Longrightarrow} \quad \omega_{\text{opt}} = 2\frac{1-\sqrt{1-\mu_*^2}}{\mu_*^2} = \frac{2}{1+\sqrt{1-\mu_*^2}} = \frac{2}{1+\sin\big(\frac{\pi}{n+1}\big)}.$$

△

Exercise 4.27 (Midterm 2022). *Let $\mathsf{A} \in \mathbf{R}^{n\times n}$ be a symmetric positive definite matrix and let $\boldsymbol{b} \in \mathbf{R}^n$. The steepest descent algorithm for solving $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ is given hereafter:*

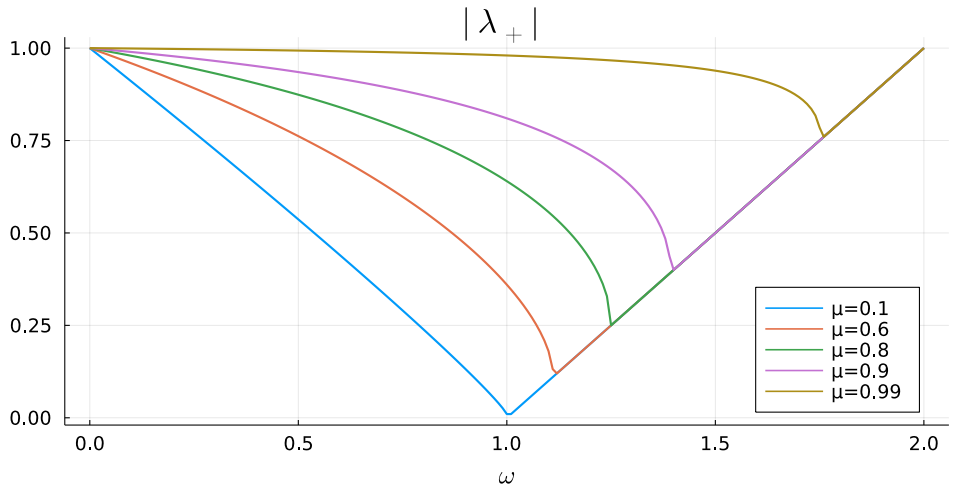


Figure 4.3: Modulus of $|\lambda_+|$ as a function of ω, for different eigenvalues of μ.

Pick $\varepsilon > 0$ *and initial* $\boldsymbol{x}$
$\boldsymbol{r} \leftarrow \mathsf{A}\boldsymbol{x} - \boldsymbol{b}$
while $\|\boldsymbol{r}\| \geqslant \varepsilon\|\boldsymbol{b}\|$ ***do***
    $\omega \leftarrow \boldsymbol{r}^T\boldsymbol{r}/\boldsymbol{r}^T\mathsf{A}\boldsymbol{r}$
    $\boldsymbol{x} \leftarrow \boldsymbol{x} - \omega\boldsymbol{r}$
    $\boldsymbol{r} \leftarrow \mathsf{A}\boldsymbol{x} - \boldsymbol{b}$
end while

- *Why is this method called the* steepest descent *algorithm?*
- *How many floating point operations does an iteration of this algorithm require?*
- *Are the following statements true of false? (2 marks)*

    1. *There exists a unique solution* $\boldsymbol{x}_*$ *to the linear system* $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$.
    2. *The iterates converge to* $\boldsymbol{x}_*$ *in at most* n *iterations.*
    3. *We consider the following modification of the algorithm:*

        Pick $\varepsilon > 0$, $\omega > 0$ *and initial* $\boldsymbol{x}$
        $\boldsymbol{r} \leftarrow \mathsf{A}\boldsymbol{x} - \boldsymbol{b}$
        while $\|\boldsymbol{r}\| \geqslant \varepsilon\|\boldsymbol{b}\|$ ***do***
            $\boldsymbol{x} \leftarrow \boldsymbol{x} - \omega\boldsymbol{r}$
            $\boldsymbol{r} \leftarrow \mathsf{A}\boldsymbol{x} - \boldsymbol{b}$
        end while

        If ω *is sufficiently small, then this algorithm converges.*

    4. *Here we no longer assume that* A *is positive definite. Instead, we consider that*

$$\mathsf{A} = \begin{pmatrix} -1 & 0 \\ 0 & -2 \end{pmatrix}.$$

        In this case, the steepest descent algorithm is convergent for any initial $\boldsymbol{x}$.

Exercise 4.28 (Final exam Spring 2022). *Assume that* $\mathsf{A} \in \mathbf{R}^{n\times n}$ *is a nonsingular matrix and that* $\boldsymbol{b} \in \mathbf{R}^n$. *We wish to solve the linear system* (4.1) *using an iterative method where each iteration is of the form*

$$\mathsf{M}\boldsymbol{x}_{k+1} = \mathsf{N}\boldsymbol{x}_k + \boldsymbol{b}. \tag{4.47}$$

Here $\mathsf{A} = \mathsf{M} - \mathsf{N}$ *is a splitting of* A *such that* M *is nonsingular, and* $\boldsymbol{x}_k \in \mathbf{R}^n$ *denotes the* k*-th iterate of the numerical scheme.*

1. *Let* $\boldsymbol{e}_k := \boldsymbol{x}_k - \boldsymbol{x}_*$, *where* $\boldsymbol{x}_*$ *is the exact solution to* (4.1). *Prove that*

$$\boldsymbol{e}_{k+1} = \mathsf{M}^{-1}\mathsf{N}\boldsymbol{e}_k.$$

2. *Let* $L = \|\mathsf{M}^{-1}\mathsf{N}\|_\infty$. *Prove that*

$$\forall k \in \mathbf{N}, \qquad \|\boldsymbol{e}_k\|_\infty \leqslant L^k \|\boldsymbol{e}_0\|_\infty. \tag{4.48}$$

3. *Is the condition* $\|\mathsf{M}^{-1}\mathsf{N}\|_\infty < 1$ *necessary for convergence when* $\boldsymbol{x}_0 \neq \boldsymbol{x}_*$?

4. *Assume that* A *is strictly row diagonally dominant, in the sense that*

$$\forall i \in \{1, \ldots, n\}, \qquad |a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|.$$

   Show that, in this case, the inequality $\|\mathsf{M}^{-1}\mathsf{N}\|_\infty < 1$ *holds for the Jacobi method, i.e. when* M *contains just the diagonal of* A. *You may take for granted the following expression for the* ∞*-norm of a matrix* $\mathsf{X} \in \mathbf{R}^{n\times n}$*:*

$$\|\mathsf{X}\|_\infty = \max_{1 \leqslant i \leqslant n} \sum_{j=1}^{n} |x_{ij}|.$$

5. *Write down a few iterations of the Jacobi method when*

$$\mathsf{A} = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \qquad \boldsymbol{b}\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \boldsymbol{x}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

   Is the method convergent?

4.5 Discussion and bibliography

In this chapter, we presented direct methods and some of the standard iterative methods for solving linear systems. We focused particularly on linear systems with a symmetric positive definite matrix. Section 4.2 is based on [10] [18] and Section 4.3 roughly follows [15, Chapter 2]. The book [11] is a very detailed reference on iterative methods for solving sparse linear systems. The reference [13] is an excellent introduction to the conjugate gradient method.