1
Introduction

In this book, we are concerned with the basic problem of solving a linear system

$$A\mathbf{x} = \mathbf{b},$$

where $A \in \mathbb{R}^{n\times n}$ is a given invertible matrix, $\mathbf{b} \in \mathbb{R}^n$ is a given vector and $\mathbf{x} \in \mathbb{R}^n$ is the solution we seek. The solution is, of course, given by

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

but does this really help if we are interested in actually computing the solution vector $\mathbf{x} \in \mathbb{R}^n$? What are the problems we are facing? First of all, such linear systems have a certain background. They are the results of other mathematical steps. Usually, they are at the end of a long processing chain which starts with setting up a partial differential equation to model a real-world problem, continues with discretising this differential equation using an appropriate approximation space and method, and results in such a linear system. This is important because it often tells us something about the structure of the matrix. The matrix might be symmetric or *sparse*. It is also important since it tells us something about the size n of the matrix. With simulations becoming more and more complex, this number nowadays becomes easily larger than a million, even values of several hundreds of millions are not unusual. Hence, the first obstacle that we encounter is the size of the matrix. Obviously, for larger dimensions n it is not possible to solve a linear system by hand. This means we need an algorithmic description of the solution process and a computer to run our program.

Unfortunately, using a computer leads to our second obstacle. We cannot represent real numbers accurately on a computer because of the limited number system used by a computer. Even worse, each calculation that we do might lead to a number which is not representable in the computer's number system.

Hence, we have to address questions like: Is a matrix that is invertible in the real numbers also invertible in the number system used by a computer? What are the errors that we make when representing the matrix in the computer and when using our algorithm to compute the solution. Further questions that easily come up are as follows.

1. How expensive is the algorithm? How much time (and space) does it require to solve the problem? What is the best way of measuring the cost of an algorithm?
2. How stable is the algorithm? If we slightly change the input, i.e. the matrix A and/or the right-hand side $\mathbf{b}$, how does this affect the solution?
3. Can we exploit the structure of the matrix A, if it has a special structure?
4. What happens if we do not have a square system, i.e. a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $\mathbf{b} \in \mathbb{R}^m$. If $m > n$ then we have an *over-determined* system and usually cannot find a (unique) solution but might still be interested in something which comes close to a solution. If $m < n$ we have an *under-determined* system and we need to choose from several possible solutions.

Besides solving a linear system, we will also be interested in a related topic, the computation of *eigenvalues* and *eigenvectors* of a matrix. This means we are interested in finding numbers $\lambda \in \mathbb{C}$ and vectors $\mathbf{x} \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that

$$A\mathbf{x} = \lambda\mathbf{x}.$$

Finding such eigenvectors and eigenvalues is again motivated by applications. For example, in structural mechanics a vibrating system is represented by finite elements and the eigenvectors of the corresponding discretisation matrix reflect the shape modes and the roots of the eigenvalues reflect the frequencies with which the system is vibrating. But eigenvalues will also be helpful in better understanding some of the questions above. For example, they have a crucial influence on the stability of an algorithm.

In this book, we are mainly interested in systems of real numbers, simply because they arise naturally in most applications. However, as the problem of finding eigenvalues indicates, it is sometimes necessary to consider complex valued systems, as well. Fortunately, most of our algorithms and findings will carry over from the real to the complex case in a straightforward way.

We will look at *direct* and *iterative* methods to solve linear systems. Direct methods compute the solution in a finite number of steps, iterative methods construct a sequence of approximations to the solution.

We will look at how efficient and stable these methods are. The former means that we are interested in how much time and computer memory they require. Particular emphasis will be placed on the number of *floating point operations* required with respect to the dimension of the linear system. The latter means for example investigating whether these methods converge at all, under what conditions they converge and how they respond to small changes in the input data.

1.1 Examples Leading to Linear Systems

As mentioned above, linear systems arise naturally during the discretisation process of mathematical models of real-world problems. Here, we want to collect three examples leading to linear systems. These examples are our model problems, which we will refer to frequently in the rest of this book. They comprise the problem of interpolating an unknown function only known at discrete data sites, the solution of a one-dimensional boundary value problem with finite differences and the solution of a (one-dimensional) integral equation with a Galerkin method. We have chosen these three examples because they are simple and easily explained, yet they are significant enough and each of them represents a specific class of problems. In particular, the second problem leads to a linear system with a matrix A which has a very simple structure. This matrix will serve us as a role model for testing and investigating most of our methods since it is simple to analyse yet complicated enough to demonstrate the advantages and drawbacks of the method under consideration.

1.1.1 Interpolation

Suppose we are given data sites $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subseteq \mathbb{R}^d$ and observations $f_1, \dots, f_n \in \mathbb{R}$. Suppose further that the observations follow an unknown generation process, i.e. there is a function f such that $f(\mathbf{x}_i) = f_i$, $1 \le i \le n$.
One possibility to approximately reconstruct the unknown function f is to choose *basis functions* $\phi_1, \dots, \phi_n \in C(\mathbb{R}^d)$ and to approximate f by a function s of the form

$$s(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j \phi_j(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^d,$$

where the coefficients are determined by the interpolation conditions

$$f_i = s(\mathbf{x}_i) = \sum_{j=1}^{n} \alpha_j \phi_j(\mathbf{x}_i), \qquad 1 \le i \le n.$$

This leads to a linear system, which can be written in matrix form as

$$\begin{pmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_n(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_n(\mathbf{x}_2) \\ \vdots & \vdots & \dots & \vdots \\ \phi_1(\mathbf{x}_n) & \phi_2(\mathbf{x}_n) & \dots & \phi_n(\mathbf{x}_n) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}. \tag{1.1}$$

From standard Numerical Analysis courses we know this topic usually in the setting that the dimension is $d = 1$, that the points are ordered $a \le x_1 < x_2 < \cdots < x_n \le b$ and that the basis is given as a basis for the space of polynomials of degree at most $n-1$. This basis could be the basis of monomials $\phi_i(x) = x^{i-1}$, $1 \le i \le n$, in which case the matrix in (1.1) becomes the transpose of a so-called *Vandermonde matrix*, i.e. a matrix of the form

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{pmatrix}.$$

This matrix is a *full matrix*, meaning that each entry is different from zero, so that the determination of the interpolant requires the solution of a linear system with a full matrix.

However, we also know, from basic Numerical Analysis, that we could alternatively choose the so-called *Lagrange functions* as a basis:

$$\phi_j(x) = L_j(x) = \prod_{\substack{i=1 \\ i \ne j}}^{n} \frac{x - x_i}{x_j - x_i}, \qquad 1 \le j \le n.$$

They obviously have the property $L_j(x_j) = 1$ and $L_j(x_i) = 0$ for $j \ne i$. Thus, with this basis, the matrix in (1.1) simply becomes the identity matrix and the interpolant can be derived without solving a linear system at all.

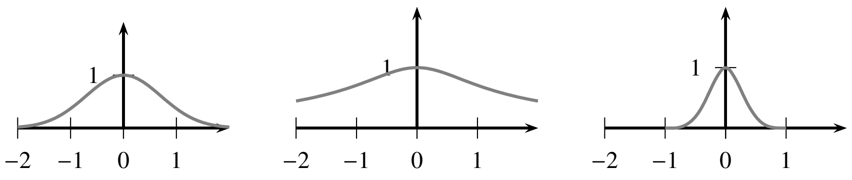

Figure 1.1 Typical radial basis functions: Gaussian, inverse multiquadric and a compactly supported one (from left to right).

In higher dimensions, i.e. $d \ge 2$, polynomial interpolation can become quite problematic and a more elegant way employs a basis of the form $\phi_i = \Phi(\cdot - \mathbf{x}_i)$, where $\Phi : \mathbb{R}^d \to \mathbb{R}$ is a fixed function. In most applications, this function is chosen to be *radial*, i.e. it is of the form $\Phi(\mathbf{x}) = \phi(\|\mathbf{x}\|_2)$, where $\phi : [0, \infty) \to \mathbb{R}$ is a univariate function and $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_d^2}$ denotes the Euclidean norm. Examples of possible univariate functions are

$$\begin{aligned} &\text{Gaussian:} && \phi(r) = \exp(-r^2), \\ &\text{Multiquadric:} && \phi(r) = (r^2+1)^{1/2}, \\ &\text{Inverse Multiquadric:} && \phi(r) = (r^2+1)^{-1/2}, \\ &\text{Compactly Supported:} && \phi(r) = (1-r)_+^4(4r+1), \end{aligned}$$

where $(x)_+$ is defined to be x if $x \geq 0$ and to be 0 if $x < 0$. The functions are visualised in Figure 1.1.

In all these cases, except for the multiquadric basis function, it is known that the resulting interpolation matrix is positive definite (with the restriction of $d \leq 3$ for the compactly supported function). Such functions are therefore called *positive definite*. More generally, a good choice of a basis is given by $\phi_j(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}_j)$ with a *kernel* $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, which is *positive definite* in the sense that for all possible, pairwise distinct points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, the matrix $(K(\mathbf{x}_i, \mathbf{x}_j))$ is symmetric and positive definite.

In the case of the multiquadric basis function, it is known that the interpolation matrix is invertible and has only real, non-vanishing eigenvalues and that all but one of these eigenvalues are negative.

Note that in the case of the inverse multiquadric and the Gaussian the matrices are dense while in the case of the compactly supported basis function the matrix can have a lot of zeros depending on the distribution of the data sites. Details on this topic can be found in Wendland [133].

1.1.2 Boundary Value Problem

Another application is to compute a stationary solution to the heat equation. In one dimension, we could imagine an infinitely thin rod of length one, which is heated in the interior of $(0, 1)$ with a heat source f and is kept at zero degrees at the boundary points 0, 1. Mathematically, this means that we want to find a function $u : [0, 1] \to \mathbb{R}$ with

$$-u''(x) = f(x), \qquad x \in (0, 1),$$

with boundary conditions $u(0) = u(1) = 0$. If the function f is too complicated or even given only at discrete points then it is not possible to compute the solution u analytically. In this case a numerical scheme has to be used and the simplest idea is to approximate the derivative by differences:

$$u'(x) \approx \frac{u(x+h)-u(x)}{h}, \qquad \text{or} \qquad u'(x) \approx \frac{u(x)-u(x-h)}{h}.$$

The first rule could be referred to as a forward rule while the second is a backward rule. Using first a forward and then a backward rule for the second derivative leads to

$$\begin{aligned} u''(x) &\approx \frac{u'(x+h)-u'(x)}{h} \approx \frac{1}{h}\left(\frac{u(x+h)-u(x)}{h} - \frac{u(x)-u(x-h)}{h}\right) \\ &= \frac{u(x+h)-2u(x)+u(x-h)}{h^2}. \end{aligned}$$

For finding a numerical approximation using such finite differences we may divide the domain $[0,1]$ into $n+1$ pieces of equal length $h = 1/(n+1)$ with nodes

$$x_i = ih = \frac{i}{n+1}, \qquad 0 \le i \le n+1,$$

and set $u_i := u(x_i)$. We now define the finite difference approximation u^h to u, as follows: find u^h such that $u^h_0 = u^h_{n+1} = 0$ and

$$-\left(\frac{u^h_{i+1} - 2u^h_i + u^h_{i-1}}{h^2}\right) = f_i, \qquad 1 \le i \le n.$$

Alternatively this linear system of n equations may be written in the form

$$\frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & & 0 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ 0 & & & & -1 & 2 \end{pmatrix}\begin{pmatrix} u^h_1 \\ u^h_2 \\ u^h_3 \\ \vdots \\ u^h_{n-1} \\ u^h_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-1} \\ f_n \end{pmatrix}. \tag{1.2}$$

This system of equations is *sparse*, meaning that the number of non-zero entries is much smaller than n^2. This sparsity can be used to store the matrix and to implement matrix–vector and matrix–matrix multiplications efficiently.
To obtain an accurate approximation to u, we may have to choose h very small, thereby increasing the size of the linear system.
For a general boundary value problem in d-dimensions the size of the linear system can grow rapidly. For example, three-dimensional problems grow over eight times larger with each uniform refinement of the domain.

1.1.3 Integral Equations

In the last section we have introduced a way of solving a differential equation. A differential equation can also be recast as an integral equation but integral equations often also come up naturally during the modelling process. Hence, let us consider a typical integral equation as another example.

We now seek a function $u : [0,1] \to \mathbb{R}$ satisfying

$$\int_0^1 \log(|x-y|)u(y)dy = f(x), \qquad x \in [0,1],$$

where $f : [0,1] \to \mathbb{R}$ is given. Note that the integral on the left-hand side contains the *kernel* $K(x,y) := \log(|x-y|)$, which is singular on the diagonal $x = y$.

To solve this integral equation numerically we will use a *Galerkin approximation*. The idea here is to choose an approximate solution u_n from a fixed, finite-dimensional subspace $V = \operatorname{span}\{\phi_1, \dots, \phi_n\}$ and to test the approximate solution via

$$\int_0^1 \int_0^1 \log(|x-y|)u_n(y)dy\, \phi_i(x)dx = \int_0^1 f(x)\phi_i(x)dx, \qquad 1 \le i \le n. \quad (1.3)$$

Since we choose $u_n \in V$ it must have a representation $u_n = \sum_{j=1}^n c_j\phi_j$ with certain coefficients c_j. Inserting this representation into (1.3) and changing the order of summation and integration yields

$$\sum_{j=1}^n c_j \int_0^1 \int_0^1 \log(|x-y|)\phi_j(y)\phi_i(x)dy\, dx = \int_0^1 f(x)\phi_i(x)dx, \qquad 1 \le i \le n,$$

which we easily identify as a linear system $A\mathbf{c} = \mathbf{f}$ with the matrix A having entries

$$a_{ij} = \int_0^1 \int_0^1 \log(|x-y|)\phi_j(y)\phi_i(x)dy\, dx, \qquad 1 \le i,j \le n.$$

A typical choice for the space V is the space of piece-wise constant functions. To be more precise, we can choose

$$\phi_i(x) = \begin{cases} 1 & \text{if } \frac{i-1}{n} \le x < \frac{i}{n}, \\ 0 & \text{else}, \end{cases}$$

but other basis functions and approximation spaces are possible. But we note that particularly in this case the matrix A is once again a full matrix as its entries are given by

$$a_{ij} = \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} \log(|x-y|)dy\, dx.$$

An obvious generalisation of this problem to arbitrary domains $\Omega \subseteq \mathbb{R}^d$ leads to matrix entries of the form

$$a_{ij} = \int_\Omega \int_\Omega K(\mathbf{x}, \mathbf{y}) \phi_i(\mathbf{x}) \phi_j(\mathbf{y}) d\mathbf{y}\, d\mathbf{x}$$

with a given kernel $K : \Omega \times \Omega \to \mathbb{R}$.

1.2 Notation

Now, it is time to set up the notation which we will use throughout this book. However, we will specify only the most basic notation and definitions here and introduce further concepts whenever required.

1.2.1 Mathematics

We will denote the real, complex, natural and integer numbers as usual with $\mathbb{R}$, $\mathbb{C}$, $\mathbb{N}$ and $\mathbb{Z}$, respectively. The natural numbers will not include zero. We will use the notation $\mathbf{x} \in \mathbb{R}^n$ to denote vectors. The components of $\mathbf{x}$ will be denoted by $x_j \in \mathbb{R}$, i.e. $\mathbf{x} = (x_1, \dots, x_n)^\mathrm{T}$. Thus vectors will always be column vectors. We will denote the unit standard basis of $\mathbb{R}^n$ by $\mathbf{e}_1, \dots, \mathbf{e}_n$, where the ith unit vector $\mathbf{e}_i$ has only zero entries except for a one at position i. In general, we will suppress the dimension n when it comes to this basis and we might use the same notation to denote the ith unit vector for $\mathbb{R}^n$ and, say, $\mathbb{R}^m$. It should be clear from the context which one is meant. On $\mathbb{R}^n$ we will denote the inner product between two vectors $\mathbf{x}$ and $\mathbf{y}$ by either $\mathbf{x}^\mathrm{T}\mathbf{y}$ or $\langle \mathbf{x}, \mathbf{y} \rangle_2$, i.e.

$$\mathbf{x}^\mathrm{T}\mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle_2 = \sum_{j=1}^n x_j y_j.$$

For a matrix A with m rows, n columns and real entries we will write $A \in \mathbb{R}^{m \times n}$ and $A = (a_{ij})$, where the index i refers to the rows and the index j refers to the columns:

$$A := (a_{ij}) := \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}.$$

For a non-square matrix $A \in \mathbb{R}^{m \times n}$, we can write $a_{ij} = \mathbf{e}_i^\mathrm{T} A \mathbf{e}_j$, where the first unit vector is from $\mathbb{R}^m$ while the second unit vector is from $\mathbb{R}^n$.

We will use the *Kronecker δ-symbol* δ_{ij}, which is defined as

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

For the *identity matrix* in $\mathbb{R}^n$ we will use the symbol $I \in \mathbb{R}^{n\times n}$. We obviously have $I = (\delta_{ij})_{1\le i,j\le n}$. Again, as in the case of the unit vectors, we will usually refrain from explicitly indicating the dimension n of the underlying space $\mathbb{R}^n$. We will also denote the columns of a matrix $A \in \mathbb{R}^{m\times n}$ by $\mathbf{a}_j := A\mathbf{e}_j \in \mathbb{R}^m$, $1 \le j \le n$. Hence, we have

$$A = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n).$$

For a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $\mathbf{x} \in \mathbb{R}^n$, we can write $\mathbf{x} = \sum_j x_j \mathbf{e}_j$ and hence

$$A\mathbf{x} = \sum_{j=1}^{n} x_j \mathbf{a}_j.$$

We will encounter specific forms of matrices and want to use the following, well-known names.

Definition 1.1 A matrix $A = (a_{ij}) \in \mathbb{R}^{m\times n}$ is

- a *square matrix*, if $m = n$,
- a *diagonal matrix*, if $a_{ij} = 0$ for $i \neq j$,
- an *upper triangular matrix*, if $a_{ij} = 0$ for $i > j$,
- a *lower triangular matrix*, if $a_{ij} = 0$ for $i < j$,
- a *band-matrix*, if there are $k, \ell \in \mathbb{N}_0$ such that $a_{ij} = 0$ if $j < i - k$ or $j > i + \ell$,
- *sparse*, if more than half of the entries are zero,
- *dense* or *full*, if it is not sparse.

In the case of a diagonal matrix A with diagonal entries $a_{ii} = \lambda_i$, we will also use the notation

$$A = \operatorname{diag}(\lambda_1, \dots, \lambda_n).$$

In particular, we have for the identity matrix $I = \operatorname{diag}(1, \dots, 1) \in \mathbb{R}^{n\times n}$.

Most of these names are self-explanatory. In the case of a band matrix, we have all entries zero outside a diagonally bordered band. Only those entries a_{ij} with indices $i - k \le j \le i + \ell$ may be different from zero. This means we have at most k sub-diagonals and ℓ super-diagonals with non-zero entries. The most prominent example is given by $k = \ell = 1$, which has one super-diagonal and one sub-diagonal of non-zero entries and is hence called a *tridiagonal* matrix.

Schematically, upper triangular, lower triangular and tridiagonal matrices look as follows:

$$\begin{pmatrix} * & * & * & \cdots & * \\ & * & * & & \vdots \\ & & \ddots & \ddots & * \\ & & & * & * \\ 0 & & & & * \end{pmatrix}, \quad \begin{pmatrix} * & & & & 0 \\ * & * & & & \\ * & \ddots & \ddots & & \\ \vdots & & * & * & \\ * & \cdots & * & * & * \end{pmatrix}, \quad \begin{pmatrix} * & * & 0 & \cdots & 0 \\ * & * & * & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & * & * & * \\ 0 & \cdots & 0 & * & * \end{pmatrix},$$

where a $*$ marks a possible non-zero entry.
For a matrix $A = (a_{ij}) \in \mathbb{R}^{m\times n}$ we denote the *transpose* of A by A^{T}. It is given by exchanging columns and rows from A, i.e. $A^{\mathrm{T}} = (a_{ji}) \in \mathbb{R}^{n\times m}$. A square matrix $A \in \mathbb{R}^{n\times n}$ is said to be *symmetric* if $A^{\mathrm{T}} = A$. If the matrix A is invertible, we have $(A^{-1})^{\mathrm{T}} = (A^{\mathrm{T}})^{-1}$ which we will simply denote with $A^{-\mathrm{T}}$. If A is symmetric and invertible then also the inverse A^{-1} is symmetric. Both the transpose and the inverse satisfy the rules

$$(AB)^{\mathrm{T}} = B^{\mathrm{T}}A^{\mathrm{T}}, \qquad (AB)^{-1} = B^{-1}A^{-1},$$

as long as these operations are well-defined.

1.2.2 Algorithms

We will not use a specific computing language to describe the algorithms in this book. However, we will assume that the reader is familiar with basic programming techniques. In particular, we expect the reader to know what a **for** and a **while** loop are. We will use **if**, **then** and **else** in the usual way and, when assigning a new value to a variable x, this variable might appear also on the right-hand side of the assignment, i.e. such an assignment can, for example, be of the form $x := x + y$, which means that x and y are first evaluated and the sum of their values is then assigned to x. For each algorithm we will declare the essential input data and the output data. There will, however, be no explicit return statement. Each algorithm should also have a deterministic stopping criterion. To demonstrate this, a first example of an algorithm is given in Algorithm 1, which computes the inner product $s := \mathbf{x}^{\mathrm{T}}\mathbf{y}$ of the two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.
At the beginning, we will formulate algorithms very close to actual programs using only basic operations. An implementation within a modern computing language should be straightforward. Later on, when it comes to more sophisticated algorithms, we will use higher-level mathematical notation to compress the representation of the algorithm. For example, an inner product will then only appear as $s := \mathbf{x}^{\mathrm{T}}\mathbf{y}$.

Algorithm 1: Inner product

```
Input  : $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.
Output: $s = \mathbf{x}^{\mathrm{T}}\mathbf{y}$.
1 $s := 0$
2 for $j = 1$ to $n$ do
3     $s := s + x_j y_j$
```

In this book, we will not discuss low-level data structures, i.e. ways of storing a vector or a matrix within a computer. We will assume that the reader is familiar with the concepts of arrays, which are usually used to store (full) matrices, and index lists, which can be used to store sparse matrices.

1.3 Landau Symbols and Computational Cost

Before developing algorithms to solve linear equations, we will introduce concepts to analyse the cost of such algorithms and their stability. Though, of course, it is possible to compare the actual run-times of two algorithms on a computer, the actual run-time is not a particularly good measure. It is more important to understand how the computational cost of an algorithm changes with the number of unknowns.

The multiplication of a matrix $A \in \mathbb{R}^{m\times n}$ with a vector $\mathbf{x} \in \mathbb{R}^n$ results in a vector $\mathbf{b} = A\mathbf{x} \in \mathbb{R}^m$ with components

$$b_i = (A\mathbf{x})_i = \sum_{j=1}^{n} a_{ij}x_j, \qquad 1 \le i \le m.$$

The multiplication of a matrix $A \in \mathbb{R}^{m\times n}$ with a matrix $B \in \mathbb{R}^{n\times p}$ gives a matrix $C = AB \in \mathbb{R}^{m\times p}$, with entries

$$c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}, \qquad 1 \le i \le m, \quad 1 \le j \le p.$$

So, how much does the computation of it cost? Usually, the cost is measured in *flops*, which stands for *floating point operations*. A floating point operation consists of one addition plus one multiplication. Sometimes, it is helpful to distinguish between additions and multiplications and count them separately. This was particularly true when a multiplication on a computer was substantially more expensive than an addition. However, as this is no longer the case on modern computers, where multiplications are realised as efficiently as additions, we will stick to the above definition of flops. It is nonetheless important to note that while subtractions are as efficient as additions and multiplications, this is not true for divisions, which are significantly slower.

Most of the time, we will not be interested in the actual number of floating point operations but rather in the asymptotic behaviour with respect to the dimension.

For example, if we look at a matrix–vector multiplication $\mathbf{b} = A\mathbf{x}$, then we must for every index i compute the sum

$$\sum_{j=1}^{n} a_{ij}x_j,$$

which means we have n multiplications and $n-1$ additions, i.e. n floating point operations. Hence, if we double the size of the matrix, we would require twice as many flops for each component. This, however, would also be true if the actual computing cost would be cn with a positive constant $c > 0$. The total cost of the matrix–vector multiplication becomes mn since we have m entries to compute.

If we are not interested in the constant $c > 0$ then we will use the following notation.

Definition 1.2 (Landau notation) For two functions $f, g : \mathbb{N}^d \to \mathbb{R}$, we will write

$$f(\mathbf{n}) = \mathcal{O}(g(\mathbf{n}))$$

if there is a constant $c > 0$ such that

$$|f(\mathbf{n})| \le c|g(\mathbf{n})|, \qquad \mathbf{n} \in \mathbb{N}^d.$$

It is important to see that the constant has to be independent of the argument $\mathbf{n} \in \mathbb{N}^d$. Moreover, though in most cases we will ignore the constant c in our considerations, a huge $c > 0$ can mean that we will never or only for very large $\mathbf{n}$ see the actual asymptotic behaviour.

With this definition, we can say that matrix–vector multiplication of a matrix $A \in \mathbb{R}^{m\times n}$ with a vector $\mathbf{x} \in \mathbb{R}^n$ costs

$$\text{time}(A\mathbf{x}) = \mathcal{O}(mn).$$

In the case of a square matrix $m = n$, the cost is therefore $\mathcal{O}(n^2)$, which means that doubling the input size of the matrix and the vector, i.e. replacing n by $2n$, will require four times the time of the original matrix–vector multiplication.

We can also use this notation to analyse the space required to store the information on a computer. We will have to store each matrix entry a_{ij} and each entry of $\mathbf{x}$ as well as the result $A\mathbf{x}$. This requires

$$\mathcal{O}(mn + n + m) = \mathcal{O}(mn)$$

space. When developing algorithms it is important to consider both resources, time and space, and it might sometimes be necessary to sacrifice something of one resource to gain in the other.

It is now easy to see that for the matrix–matrix multiplication $C = AB$, we would require $\mathcal{O}(mnp)$ operations. Hence, for square systems with $m = n = p$ the time is $\mathcal{O}(n^3)$ and doubling the input size results in computations that are eight times longer.

Let us summarise our findings so far, with some obvious additions.

Lemma 1.3 *Let $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, $\mathbf{x} \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$.*

- *It costs $\mathcal{O}(n)$ space to store the vector $\mathbf{x} \in \mathbb{R}^n$ and $\mathcal{O}(mn)$ space to store the matrix A.*
- *It costs $\mathcal{O}(n)$ time to compute the product $\alpha\mathbf{x}$.*
- *It costs $\mathcal{O}(mn)$ time to compute the product $A\mathbf{x}$.*
- *It costs $\mathcal{O}(mnp)$ time to compute the product AB.*

The cost is called *linear* if it is $\mathcal{O}(n)$, *quadratic* if it is $\mathcal{O}(n^2)$ and *cubic* if it is $\mathcal{O}(n^3)$.

More sophisticated matrix–matrix products have been developed with the goal of reducing the computational cost by reducing the number of multiplications at the cost of a mild increase in the number of additions. The most famous one is a recursive algorithm by Strassen (see [120]) which can compute the product of two $n \times n$ matrices using at most $4.7 \cdot n^{\log_2 7} = \mathcal{O}(n^{2.807355})$ flops. Since then, other such algorithms have been introduced, most notably one by Coppersmith and Winograd in [37] which reduces the cost to $\mathcal{O}(n^{2.375477})$. The latter algorithm is, however, more complicated and the constant hidden in the $\mathcal{O}$-notation substantially larger so that the algorithm is not used in practice. Though superior in this context, even the Strassen algorithm is not seriously used in practical applications, which is due to the fact that it is not as stable as the conventional scheme, see Higham [84].

Finally, let us see how additional information can be used to reduce the cost. If, for example, we have a *tridiagonal matrix*, i.e. a matrix which has only non-zero entries on the diagonal and the sub- and super-diagonal, i.e. which is of the form

$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & \cdots & 0 & a_{n,n-1} & a_{nn} \end{pmatrix},$$

then a matrix–vector multiplication reduces to

$$(A\mathbf{x})_i = \sum_{j=1}^{n} a_{ij}x_j = a_{i,i-1}x_{i-1} + a_{ii}x_i + a_{i,i+1}x_{i+1}.$$

Hence, the time for the full matrix–vector product is now only $O(n)$ instead of $O(n^2)$. Also the matrix can be stored by exploiting its special form in $O(n)$ space instead of $O(n^2)$.

We will sometimes encounter algorithms which have a recursive structure. This means, for example, that solving a problem with problem size n is reduced to solving problems with problem size n/b, where $b > 1$. The Strassen algorithm mentioned above is one such example. To analyse the cost of such an algorithm it is often helpful to assume that $n = b^k$ with some $k \in \mathbb{N}$. The following result will be helpful.

Lemma 1.4 *The recursion $T(1) = c$ and $T(n) = aT(n/b)+cn^\alpha$ with $c, a, \alpha > 0$ and $b > 1$ satisfies*

$$T(n) \le \begin{cases} c\frac{b^\alpha}{b^\alpha - a}n^\alpha & \text{if } a < b^\alpha, \\ cn^\alpha(\log_b n + 1) & \text{if } a = b^\alpha, \\ c\frac{a}{a-b^\alpha}n^{\log_b a} & \text{if } a > b^\alpha. \end{cases}$$

Proof As mentioned above, we will prove this only for $n = b^k$. Though the results remain true for general n, the proof is more tedious in the general case. Using induction, it is easy to see that

$$T(b^k) = c\sum_{i=0}^{k} a^i(b^\alpha)^{k-i}.$$

Hence, if $a < b^\alpha$, we have

$$T(b^k) = cb^{k\alpha}\sum_{i=0}^{k}\left(\frac{a}{b^\alpha}\right)^i = cn^\alpha\frac{1-\left(\frac{a}{b^\alpha}\right)^{k+1}}{1-\frac{a}{b^\alpha}} < cn^\alpha\frac{b^\alpha}{b^\alpha - a}.$$

For $a = b^\alpha$ we have

$$T(b^k) = cn^\alpha(k+1) = cn^\alpha(\log_b n + 1)$$

and for $a > b^\alpha$ we finally use

$$T(b^k) = ca^k \sum_{i=0}^{k} \left(\frac{b^\alpha}{a}\right)^i = ca^k \frac{1 - \left(\frac{b^\alpha}{a}\right)^{k+1}}{1 - \frac{b^\alpha}{a}} \le ca^k \frac{a}{a - b^\alpha} = c\frac{a}{a - b^\alpha} n^{\log_b a},$$

where we have used

$$a^k = e^{k \log a} = \exp\left[\frac{\log a}{\log b} \log b^k\right] = (b^k)^{\frac{\log a}{\log b}} = n^{\log_b a}. \qquad \square$$

Finally, let us mention that the definition of the Landau symbol O can even be further generalised in the following sense.

Definition 1.5 For two functions $f, g : \mathbb{R}^n \to \mathbb{R}$ and $\mathbf{x}_0 \in \mathbb{R}^n$, we will write

$$f(\mathbf{x}) = O(g(\mathbf{x})), \qquad \mathbf{x} \to \mathbf{x}_0,$$

if there is a constant $c > 0$ and a surrounding $U = U(\mathbf{x}_0) \subseteq \mathbb{R}^n$ of $\mathbf{x}_0$ such that

$$|f(x)| \le c|g(\mathbf{x})|, \qquad \mathbf{x} \in U.$$

1.4 Facts from Linear Algebra

In this section, we want to collect further material on matrices and vectors, which should be known from classical, basic linear algebra courses. The character of this section is more to remind the reader of the material and to introduce the notation. However, we will also prove some results which are less familiar.

In $\mathbb{R}^n$ we have the canonical inner product defined by $\langle \mathbf{x}, \mathbf{y} \rangle_2 := \mathbf{x}^T\mathbf{y}$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. It particularly satisfies $\langle \mathbf{x}, \mathbf{x} \rangle_2 = x_1^2 + \cdots + x_n^2 > 0$ for all $\mathbf{x} \neq \mathbf{0}$. As mentioned above, we will use both notations $\langle \mathbf{x}, \mathbf{y} \rangle_2$ and $\mathbf{x}^T\mathbf{y}$ equally. The canonical inner product defines a canonical norm or length $\|\mathbf{x}\|_2 := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_2}$, the *Euclidean norm*. This norm has the usual properties of a norm, which can more generally be defined for an arbitrary linear space. Of course, we assume the reader to be familiar with the concept of linear spaces, linear sub-spaces, linear independent vectors and the dimension of a linear space.

Definition 1.6 Let V be a real (or complex) linear space. A mapping $\|\cdot\| : V \to [0, \infty)$ is called a *norm* on V if it satisfies

1. homogeneity: $\|\lambda \mathbf{x}\| = |\lambda|\|\mathbf{x}\|$ for all $\mathbf{x} \in V$ and $\lambda \in \mathbb{R}$ (or $\lambda \in \mathbb{C}$),

2. definiteness: $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = \mathbf{0}$,
3. triangle inequality: $\|\mathbf{x} + \mathbf{y}\| \le \|\mathbf{x}\| + \|\mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in V$.

The space V with norm $\|\cdot\|$ is called a *normed space*.

In the case of $V = \mathbb{R}^n$ and $\|\cdot\| = \|\cdot\|_2$, the first two properties follow immediately from the definition, while the triangle inequality follows from the *Cauchy–Schwarz inequality*

$$|\langle \mathbf{x}, \mathbf{y} \rangle_2| \le \|\mathbf{x}\|_2 \|\mathbf{y}\|_2$$

which holds for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and where equality occurs if and only if $\mathbf{y}$ is a scalar multiple of $\mathbf{x}$. This all remains true, in the more general situation when the norm is defined by an inner product.

Definition 1.7 Let V be a real linear space. A mapping $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ is called an *inner product* if it is

1. symmetric: $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle$ for all $\mathbf{x}, \mathbf{y} \in V$,
2. linear: $\langle \alpha\mathbf{x} + \beta\mathbf{y}, \mathbf{z} \rangle = \alpha\langle \mathbf{x}, \mathbf{z} \rangle + \beta\langle \mathbf{y}, \mathbf{z} \rangle$ for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha, \beta \in \mathbb{R}$,
3. definite: $\langle \mathbf{x}, \mathbf{x} \rangle > 0$ for all $\mathbf{x} \in V \setminus \{\mathbf{0}\}$.

A space V with an inner product $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ is called a *pre-Hilbert space*.

The second property together with the first property indeed guarantees that the mapping $\langle \cdot, \cdot \rangle$ is bilinear, i.e. it is also linear in the second argument. Each pre-Hilbert space becomes a normed space upon defining the canonical norm $\|\mathbf{x}\| := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$.

If V is a complex linear space then an inner product on V is again a mapping $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{C}$ but the first two conditions above have to be modified appropriately. For example, the first condition becomes $\langle \mathbf{x}, \mathbf{y} \rangle = \overline{\langle \mathbf{y}, \mathbf{x} \rangle}$ for all $\mathbf{x}, \mathbf{y} \in V$, where $\overline{\alpha}$ is the complex conjugate of $\alpha \in \mathbb{C}$. The second property must now hold for all $\alpha, \beta \in \mathbb{C}$, which means that the inner product is linear in its first and anti-linear in its second component. The canonical norm is then defined as before.

Throughout this book, we will mainly be concerned with the space $V = \mathbb{R}^n$ and hence introduce most concepts only for this space. But it is worth noting that some of them immediately carry over to more general spaces.

As usual, for given $\mathbf{x}_1, \ldots, \mathbf{x}_n \in V$, we use the notation $\operatorname{span}\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ to denote the linear sub-space of V spanned by these elements, i.e.

$$\operatorname{span}\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} = \left\{ \sum_{j=1}^{n} \alpha_j \mathbf{x}_j : \alpha_1, \ldots, \alpha_n \in \mathbb{R} \right\}.$$

Definition 1.8 Let V be a finite-dimensional pre-Hilbert space with basis $\mathbf{x}_1, \dots, \mathbf{x}_n$. A basis is called an *orthonormal basis* if all basis vectors have unit length, i.e. $\|\mathbf{x}_i\| = \langle \mathbf{x}_i, \mathbf{x}_i \rangle^{1/2} = 1$ and two different vectors are *orthogonal*, i.e $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = 0$ for $i \neq j$.

It is always possible to transform a given basis into an orthonormal basis.

Lemma 1.9 (Gram–Schmidt) *Let $\mathbf{x}_1, \dots, \mathbf{x}_n$ be linearly independent vectors of a pre-Hilbert space V. Define $\mathbf{u}_1 = \mathbf{x}_1/\|\mathbf{x}_1\|$ and then for $k = 1, 2, \dots, n$,*

$$\tilde{\mathbf{u}}_{j+1} := \mathbf{u}_{j+1} - \sum_{j=1}^{k} \langle \mathbf{u}_{j+1}, \mathbf{u}_j \rangle \mathbf{u}_j,$$
$$\mathbf{u}_{j+1} := \tilde{\mathbf{u}}_{j+1}/\|\tilde{\mathbf{u}}_{j+1}\|.$$

Then, the set $\{\mathbf{u}_1, \dots, \mathbf{u}_k\}$ forms an orthonormal basis of $\operatorname{span}\{\mathbf{x}_1, \dots, \mathbf{x}_k\}$ *for $1 \leq k \leq n$.*

The Gram–Schmidt orthonormalisation process should be well-known and we leave a proof of the above lemma to the reader. Though the Gram–Schmidt procedure is obviously easily implemented, it is numerically often problematic. We will later discuss better methods of finding orthonormal bases.

Definition 1.10 Let $A \in \mathbb{R}^{m \times n}$ be given. Its *null space* is defined to be the set

$$\ker(A) = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = \mathbf{0}\} \subseteq \mathbb{R}^n$$

and its *range* to be

$$\operatorname{range}(A) = \{A\mathbf{x} : \mathbf{x} \in \mathbb{R}^n\} \subseteq \mathbb{R}^m.$$

The *rank* of $A \in \mathbb{R}^{m \times n}$ is defined as

$$\operatorname{rank}(A) = \dim\{A\mathbf{x} : \mathbf{x} \in \mathbb{R}^n\} = \dim \operatorname{range}(A).$$

It is easy to see that the null space of a matrix $A \in \mathbb{R}^{m \times n}$ is a subspace of $\mathbb{R}^n$, while the range of A is a subspace of $\mathbb{R}^m$.

Definition 1.11 Let $A \in \mathbb{R}^{n \times n}$ be a square matrix.

1. The matrix is called *positive semi-definite* if, for all $\mathbf{x} \in \mathbb{R}^n$, we have

$$\mathbf{x}^{\mathrm{T}} A \mathbf{x} = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j a_{ij} \geq 0.$$

   It is called *positive definite* if the above expression is positive for all $\mathbf{x} \neq \mathbf{0}$.
2. The matrix is called *orthogonal* if $A^{\mathrm{T}} A = I$.

3. The matrix is called *diagonalisable* if there exist an invertible matrix $S \in \mathbb{R}^{n\times n}$ (or more generally $S \in \mathbb{C}^{n\times n}$) and a diagonal matrix $D \in \mathbb{R}^{n\times n}$ ($\in \mathbb{C}^{n\times n}$) such that $A = SDS^{-1}$.

If a matrix $A \in \mathbb{R}^{n\times n}$ is symmetric then there is an orthonormal basis of $\mathbb{R}^n$ consisting of eigenvectors of A. This means that there are real values $\lambda_1, \dots, \lambda_n \in \mathbb{R}$ and vectors $\mathbf{w}_1, \dots, \mathbf{w}_n \in \mathbb{R}^n \setminus \{\mathbf{0}\}$, which satisfy

$$A\mathbf{w}_j = \lambda_j \mathbf{w}_j, \qquad \langle \mathbf{w}_j, \mathbf{w}_k \rangle_2 = \delta_{jk}.$$

The eigenvectors form an orthogonal matrix $Q = (\mathbf{w}_1, \dots, \mathbf{w}_n) \in \mathbb{R}^{n\times n}$ and hence we have the following result.

Proposition 1.12 *If $A \in \mathbb{R}^{n\times n}$ is symmetric then there is an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ such that $Q^{\mathrm{T}}AQ = D$ is a diagonal matrix with the eigenvalues of A as diagonal entries.*
A symmetric matrix is positive definite (positive semi-definite) if and only if all its eigenvalues are positive (non-negative).

As a consequence, a symmetric and positive definite matrix possesses a *square root*.

Corollary 1.13 *If $A \in \mathbb{R}^{n\times n}$ is symmetric and positive (semi-)definite then there is a symmetric and positive (semi-)definite matrix $A^{1/2} \in \mathbb{R}^{n\times n}$ such that $A = A^{1/2}A^{1/2}$. The matrix $A^{1/2}$ is called a* square root *of A.*

Proof According to Proposition 1.12, we can write $A = QDQ^{\mathrm{T}}$ with a diagonal matrix $D = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ and an orthogonal matrix Q. The diagonal entries of D are non-negative. Hence, we can define the diagonal matrix $D^{1/2} := \operatorname{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n})$ and then $A^{1/2} := QD^{1/2}Q^{\mathrm{T}}$. This matrix is obviously symmetric and because of

$$\mathbf{x}^{\mathrm{T}}A^{1/2}\mathbf{x} = \mathbf{x}^{\mathrm{T}}QD^{1/2}Q^{\mathrm{T}}\mathbf{x} = \mathbf{y}^{\mathrm{T}}D^{1/2}\mathbf{y} = \sum_{j=1}^{n} y_j^2 \sqrt{\lambda_j} \geq 0$$

also positive (semi-)definite, where we have set $\mathbf{y} := Q^{\mathrm{T}}\mathbf{x}$. Finally, we have

$$A^{1/2}A^{1/2} = (QD^{1/2}Q^{\mathrm{T}})(QD^{1/2}Q^{\mathrm{T}}) = QD^{1/2}D^{1/2}Q^{\mathrm{T}} = QDQ^{\mathrm{T}} = A. \qquad \square$$

We also remark that for a matrix $A \in \mathbb{R}^{m\times n}$, the matrix $A^{\mathrm{T}}A \in \mathbb{R}^{n\times n}$ is obviously symmetric and positive semi-definite, simply because of

$$\mathbf{x}^{\mathrm{T}}(A^{\mathrm{T}}A)\mathbf{x} = (A\mathbf{x})^{\mathrm{T}}(A\mathbf{x}) = \|A\mathbf{x}\|_2^2 \geq 0.$$

As a matter of fact, each symmetric and positive definite matrix can be written

in this form, see Exercise 1.5. An orthogonal matrix is always invertible and its rows and columns, respectively, form an orthonormal basis of $\mathbb{R}^n$.
Sometimes it is necessary to consider a generalised eigenvalue problem. This occurs, for example, when discussing discretised systems coming from elasticity theory. Later on, it will also be important when we look at preconditioning.

Definition 1.14 Let $A, B \in \mathbb{R}^{n\times n}$ be matrices. Then, $\lambda \in \mathbb{C}$ is a *generalised eigenvalue* with respect to A, B if there is a non-zero vector $\mathbf{x} \in \mathbb{C}^n$ such that

$$A\mathbf{x} = \lambda B\mathbf{x}.$$

The vector $\mathbf{x}$ is called *generalised eigenvector*.

If the matrix B is invertible, finding a generalised eigenvalue and eigenvector is equivalent to finding a classical eigenvalue and eigenvector of the matrix $B^{-1}A$. However, even if A and B are symmetric, i.e. both have only real eigenvalues and a complete set of eigenvectors, then the matrix $B^{-1}A$ is in general not symmetric. Nonetheless, we have the following result.

Theorem 1.15 *Let $A, B \in \mathbb{R}^{n\times n}$ be symmetric and positive definite. Then, all eigenvalues of $B^{-1}A$ are real and positive. Moreover, there exist n pairs of generalised eigenvalues λ_j and eigenvectors $\mathbf{v}_j$, $1 \le j \le n$, satisfying $A\mathbf{v}_j = \lambda_j B\mathbf{v}_j$ and*

$$\mathbf{v}_i^{\mathrm{T}} B\mathbf{v}_j = \mathbf{v}_i^{\mathrm{T}} A\mathbf{v}_j = 0, \qquad i \neq j.$$

The matrix $V = (\mathbf{v}_1, \dots, \mathbf{v}_n)$ diagonalises the matrices A and B simultaneously. If the eigenvectors are normalised such that $\mathbf{v}_i^{\mathrm{T}} B\mathbf{v}_i = 1$, $1 \le i \le n$, then

$$V^{\mathrm{T}} BV = I, \qquad V^{\mathrm{T}} AV = \mathrm{diag}(\lambda_1, \dots, \lambda_n).$$

Proof As mentioned above, a generalised eigenvalue is simply an eigenvalue of $B^{-1}A$. Since B is symmetric and positive definite, it has a symmetric and positive definite root $B^{1/2}$ with inverse denoted by $B^{-1/2}$. The matrix $B^{-1/2}AB^{-1/2}$ is symmetric and positive definite since we have on the one hand

$$(B^{-1/2}AB^{-1/2})^{\mathrm{T}} = (B^{-1/2})^{\mathrm{T}} A^{\mathrm{T}} (B^{-1/2})^{\mathrm{T}} = B^{-1/2}AB^{-1/2}$$

and on the other hand for $\mathbf{x} \in \mathbb{R}^n$:

$$\mathbf{x}^{\mathrm{T}} B^{-1/2}AB^{-1/2}\mathbf{x} = (B^{-1/2}\mathbf{x})^{\mathrm{T}} A(B^{-1/2}\mathbf{x}) > 0,$$

unless $B^{-1/2}\mathbf{x} = \mathbf{0}$ which is equivalent to $\mathbf{x} = \mathbf{0}$. Thus, all eigenvalues of

$B^{-1/2}AB^{-1/2}$ are positive. Next, for $\lambda \in \mathbb{C}$ we have

$$\begin{aligned}\det(B^{-1/2}AB^{-1/2} - \lambda I) &= \det(B^{-1/2}A - \lambda B^{1/2})\det(B^{-1/2})\\ &= \det(B^{-1/2})\det(B^{-1/2}A - \lambda B^{1/2})\\ &= \det(B^{-1}A - \lambda I),\end{aligned}$$

meaning that $B^{-1/2}AB^{-1/2}$ and $B^{-1}A$ have exactly the same eigenvalues. Finally, let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of $B^{-1/2}AB^{-1/2}$ with corresponding, orthonormal eigenvectors $\mathbf{w}_1, \dots, \mathbf{w}_n$. If we set $\mathbf{v}_j := B^{-1/2}\mathbf{w}_j$ then we have

$$B^{-1/2}A\mathbf{v}_j = B^{-1/2}AB^{-1/2}\mathbf{w}_j = \lambda_j\mathbf{w}_j = \lambda_j B^{1/2}\mathbf{v}_j,$$

or $A\mathbf{v}_j = \lambda_j B\mathbf{v}_j$. Moreover, we have

$$\mathbf{v}_j^{\mathrm{T}}B\mathbf{v}_k = \mathbf{w}_j^{\mathrm{T}}B^{-1/2}BB^{-1/2}\mathbf{w}_k = \mathbf{w}_j^{\mathrm{T}}\mathbf{w}_k = \delta_{jk}$$

and

$$\mathbf{v}_j^{\mathrm{T}}A\mathbf{v}_k = \mathbf{w}_j^{\mathrm{T}}B^{-1/2}AB^{-1/2}\mathbf{w}_k = \lambda_k\delta_{jk}. \qquad \square$$

In several places we will require the notion of orthogonal projection. Again, we define things immediately in a more general setting though we will mainly be interested in the case of $\mathbb{R}^n$.

Definition 1.16 Let V be a pre-Hilbert space. A linear mapping $P : V \to V$ is called a *projection* if $P^2 = P$. A linear mapping $P : V \to U \subseteq V$ is called *orthogonal projection onto* U, if

$$\langle \mathbf{x} - P\mathbf{x}, \mathbf{u}\rangle = 0, \qquad \mathbf{x} \in V, \quad \mathbf{u} \in U. \tag{1.4}$$

It is easy to see that a linear mapping $P : V \to V$ is a projection if and only if $P\mathbf{u} = \mathbf{u}$ for all $\mathbf{u} \in \operatorname{range}(P) = \{P\mathbf{x} : \mathbf{x} \in V\}$. Moreover, an orthogonal projection is indeed a projection since we have for $\mathbf{x} \in U$ also $\mathbf{x} - P\mathbf{x} \in U$ and hence by the orthogonality condition $\|\mathbf{x} - P\mathbf{x}\|^2 = \langle \mathbf{x} - P\mathbf{x}, \mathbf{x} - P\mathbf{x}\rangle = 0$, i.e. $P\mathbf{x} = \mathbf{x}$ for all $\mathbf{x} \in U$.

If $\mathbf{u}_1, \dots, \mathbf{u}_k \in U$ form an orthonormal basis of the finite-dimensional subspace $U \subseteq V$, then the orthogonal projection onto U is given by

$$P\mathbf{x} = \sum_{j=1}^{k} \langle \mathbf{x}, \mathbf{u}_j\rangle \mathbf{u}_j. \tag{1.5}$$

Obviously, this so-defined P is linear and maps onto U. To see that it is also the orthogonal projection, let us assume that V is also finite-dimensional, as we are only concerned with finite-dimensional spaces in this book, though the result is also true for infinite-dimensional spaces. If V is finite-dimensional, we can extend $\mathbf{u}_1, \ldots, \mathbf{u}_k$ to an orthonormal basis $\mathbf{u}_1, \ldots, \mathbf{u}_k, \ldots, \mathbf{u}_n$ of V. Then, we have, for any $\mathbf{x} \in V$,

$$\mathbf{x} = \sum_{j=1}^{n} \langle \mathbf{x}, \mathbf{u}_j \rangle \mathbf{u}_j, \qquad \mathbf{x} - P\mathbf{x} = \sum_{j=k+1}^{n} \langle \mathbf{x}, \mathbf{u}_j \rangle \mathbf{u}_j,$$

so that the orthogonal condition (1.4) yields for P defined by (1.5) that

$$\langle \mathbf{x} - P\mathbf{x}, \mathbf{u}_i \rangle = \left\langle \sum_{j=k+1}^{n} \langle \mathbf{x}, \mathbf{u}_j \rangle \mathbf{u}_j, \mathbf{u}_i \right\rangle = \sum_{j=k+1}^{n} \langle \mathbf{x}, \mathbf{u}_j \rangle \langle \mathbf{u}_j, \mathbf{u}_i \rangle = 0$$

for all $1 \le i \le k$.

The orthogonal projection $P\mathbf{x}$ also describes the *best approximation* to $\mathbf{x}$ from U.

Proposition 1.17 *Let V be a pre-Hilbert space and let $U \subseteq V$ be a finite-dimensional subspace. Then, for $\mathbf{x} \in V$ and $\mathbf{u}^* \in U$ are equivalent:*

1. *$\mathbf{u}^*$ is the best approximation to $\mathbf{x}$ from U, i.e.*

$$\|\mathbf{x} - \mathbf{u}^*\| = \min_{\mathbf{u} \in U} \|\mathbf{x} - \mathbf{u}\|.$$

2. *$\mathbf{u}^*$ is the orthogonal projection of $\mathbf{x}$ onto U, i.e.*

$$\langle \mathbf{x} - \mathbf{u}^*, \mathbf{u} \rangle = 0, \qquad \mathbf{u} \in U.$$

Proof Let $\mathbf{u}_1, \ldots, \mathbf{u}_k$ be an orthonormal basis of U. Then, using the orthonormality we have with $P\mathbf{x}$ from (1.5) for arbitrary $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$,

$$\begin{aligned}
\|\mathbf{x} - P\mathbf{x}\|^2 &= \left\| \mathbf{x} - \sum_{j=1}^{k} \langle \mathbf{x}, \mathbf{u}_j \rangle \mathbf{u}_j \right\|^2 = \left\langle \mathbf{x} - \sum_{i=1}^{k} \langle \mathbf{x}, \mathbf{u}_i \rangle \mathbf{u}_i, \mathbf{x} - \sum_{j=1}^{k} \langle \mathbf{x}, \mathbf{u}_j \rangle \mathbf{u}_j \right\rangle \\
&= \|\mathbf{x}\|^2 - 2 \sum_{j=1}^{k} |\langle \mathbf{x}, \mathbf{u}_j \rangle|^2 + \sum_{i,j=1}^{k} \langle \mathbf{x}, \mathbf{u}_j \rangle \langle \mathbf{x}, \mathbf{u}_i \rangle_2 \langle \mathbf{u}_i, \mathbf{u}_j \rangle \\
&= \|\mathbf{x}\|^2 - \sum_{j=1}^{k} |\langle \mathbf{x}, \mathbf{u}_j \rangle|^2 \le \|\mathbf{x}\|^2 - \sum_{j=1}^{k} |\langle \mathbf{x}, \mathbf{u}_j \rangle|^2 + \sum_{j=1}^{k} (\langle \mathbf{x}, \mathbf{u}_j \rangle - \alpha_j)^2 \\
&= \|\mathbf{x}\|^2 - 2 \sum_{j=1}^{k} \alpha_j \langle \mathbf{x}, \mathbf{u}_j \rangle + \sum_{j=1}^{k} \alpha_j^2 \langle \mathbf{u}_j, \mathbf{u}_j \rangle \\
&= \left\| \mathbf{x} - \sum_{j=1}^{k} \alpha_j \mathbf{u}_j \right\|^2 .
\end{aligned}$$

This shows that $P\mathbf{x}$ from (1.5) is the unique best approximation to $\mathbf{x}$ from U.

As we already know that $P\mathbf{x}$ is also the orthogonal projection of $\mathbf{x}$ onto U, the proof is finished. □

1.5 Singular Value Decomposition

The geometric interpretation of Proposition 1.12 is simple. The linear mapping $A : \mathbb{R}^n \to \mathbb{R}^n$, which is given by the matrix A in the standard basis, has a much simpler representation when going over to the basis consisting of the eigenvectors of A. If this basis is used in both the domain and the range of the mapping then it can be represented by the diagonal matrix D. While the simplicity of this representation is indeed appealing, it is not possible to find such a simple representation for all matrices $A \in \mathbb{R}^{n\times n}$ let alone for non-square matrices $A \in \mathbb{R}^{m\times n}$. However, if we relax the requirement of using the same orthogonal basis in the domain and the range, i.e. in $\mathbb{R}^n$ and $\mathbb{R}^m$, to represent the mapping, we can find a similarly simple representation. This is the so-called *singular value decomposition*, which we want to discuss now.

To understand it, we need to recall a few more facts from linear algebra. We already introduced the rank and null space of a matrix. We also need the intrinsic, well-known relations

$$n = \dim\ker(A) + \operatorname{rank}(A), \qquad \operatorname{rank}(A) = \operatorname{rank}(A^{\mathrm{T}}). \tag{1.6}$$

Note that an element $\mathbf{x} \in \ker(A)$ satisfies $A\mathbf{x} = \mathbf{0}$ and hence also $A^{\mathrm{T}}A\mathbf{x} = \mathbf{0}$, i.e. it belongs to $\ker(A^{\mathrm{T}}A)$. If, on the other hand $\mathbf{x} \in \ker(A^{\mathrm{T}}A)$ then $A^{\mathrm{T}}A\mathbf{x} = \mathbf{0}$, thus $0 = \mathbf{x}^{\mathrm{T}}(A^{\mathrm{T}}A\mathbf{x}) = \|A\mathbf{x}\|_2^2$ and hence $\mathbf{x} \in \ker(A)$. This means

$$\ker(A) = \ker(A^{\mathrm{T}}A).$$

Similarly, we can easily see that $\ker(A^{\mathrm{T}}) = \ker(AA^{\mathrm{T}})$. Hence, the dimension formula above immediately gives

$$\operatorname{rank}(A) = \operatorname{rank}(A^{\mathrm{T}}) = \operatorname{rank}(A^{\mathrm{T}}A) = \operatorname{rank}(AA^{\mathrm{T}}).$$

This means that the symmetric and positive semi-definite matrices $A^{\mathrm{T}}A \in \mathbb{R}^{n\times n}$ and $AA^{\mathrm{T}} \in \mathbb{R}^{m\times m}$ have the same rank.

As before, we choose for $A^{\mathrm{T}}A$ a system of eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$ and associated eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$ satisfying $A^{\mathrm{T}}A\mathbf{v}_j = \lambda_j\mathbf{v}_j$ and $\mathbf{v}_j^{\mathrm{T}}\mathbf{v}_k = \delta_{jk}$. Exactly the first $r = \operatorname{rank}(A)$ eigenvalues are positive and the remaining ones are zero.

According to the considerations we just made, also the matrix AA^{T} must have exactly r positive eigenvalues. We will now see that, interestingly, the positive eigenvalues of AA^{T} are exactly the positive eigenvalues of $A^{\mathrm{T}}A$ and that there is a very simple connection between the corresponding eigenvectors. To see this, let us define

$$\sigma_j := \sqrt{\lambda_j}, \qquad 1 \le j \le n,$$
$$\mathbf{u}_j := \frac{1}{\sigma_j} A\mathbf{v}_j, \qquad 1 \le j \le r.$$

Then, we have on the one hand that

$$AA^{\mathrm{T}}\mathbf{u}_j = \frac{1}{\sigma_j} AA^{\mathrm{T}}A\mathbf{v}_j = \frac{1}{\sigma_j}\lambda_j A\mathbf{v}_j = \lambda_j \mathbf{u}_j, \qquad 1 \le j \le r.$$

On the other hand, we have that

$$\mathbf{u}_j^{\mathrm{T}}\mathbf{u}_k = \frac{1}{\sigma_j\sigma_k}\mathbf{v}_j^{\mathrm{T}}A^{\mathrm{T}}A\mathbf{v}_k = \frac{\lambda_k}{\sigma_j\sigma_k}\mathbf{v}_j^{\mathrm{T}}\mathbf{v}_k = \delta_{jk}, \qquad 1 \le j,k \le r.$$

Thus, $\mathbf{u}_1, \dots, \mathbf{u}_r \in \mathbb{R}^m$ form an orthonormal system of eigenvectors corresponding to the eigenvalues $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_r > 0$ of AA^{T}. Since this matrix also has rank r and is positive semi-definite, all other eigenvalues have to be zero. It is now possible to complete the system $\{\mathbf{u}_1, \dots, \mathbf{u}_r\}$ to an orthonormal basis $\{\mathbf{u}_1, \dots, \mathbf{u}_m\}$ consisting of eigenvectors. If we define the matrices $U := (\mathbf{u}_1, \dots, \mathbf{u}_m) \in \mathbb{R}^{m\times m}$ and $V = (\mathbf{v}_1, \dots, \mathbf{v}_n) \in \mathbb{R}^{n\times n}$ then we have

$$A\mathbf{v}_j = \sigma_j \mathbf{u}_j, \qquad 1 \le j \le r,$$
$$A\mathbf{v}_j = \mathbf{0}, \qquad r+1 \le j \le n,$$

by definition and by the fact that $\ker(A) = \ker(A^{\mathrm{T}}A)$. This can alternatively be written as

$$AV = U\Sigma,$$

where $\Sigma = (\sigma_j\delta_{ij}) \in \mathbb{R}^{m\times n}$ is a non-square diagonal matrix. This altogether proves the following result.

Theorem 1.18 (Singular Value Decomposition (SVD)) *A matrix $A \in \mathbb{R}^{m\times n}$ has a* singular value decomposition $A = U\Sigma V^{\mathrm{T}}$ *with orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ and a diagonal matrix $\Sigma = (\sigma_j\delta_{ij}) \in \mathbb{R}^{m\times n}$.*

Note that some of the σ_j may be zero, depending on the rank of the matrix A. If r is the rank of A, then there are exactly r non-zero and hence positive σ_j. Furthermore, the singular value decomposition is not unique. To be more precise, the matrices U and V are not unique, since we can, for example, change the sign of the columns. Moreover, if some of the σ_j have the same value, then we can choose different bases for the corresponding eigenspaces. However, the values σ_j are uniquely determined by A and thus, if we order the σ_j, then the matrix Σ is unique.

Definition 1.19 Those σ_j in the singular value decomposition which are positive are called the *singular values* of the matrix A.

Taking $r = \operatorname{rank}(A)$ into account, we can also rewrite the representation $A = U\Sigma V^{\mathrm{T}}$ from Theorem 1.18 in the form

$$A = \sum_{j=1}^{n} \sigma_j \mathbf{u}_j \mathbf{v}_j^{\mathrm{T}} = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^{\mathrm{T}}.$$

Hence, if we define the matrices $\hat{U} := (\mathbf{u}_1, \dots, \mathbf{u}_r) \in \mathbb{R}^{m\times r}$, $\hat{V} := (\mathbf{v}_1, \dots, \mathbf{v}_r) \in \mathbb{R}^{n\times r}$ and $\hat{\Sigma} := \operatorname{diag}(\sigma_1, \dots, \sigma_r) \in \mathbb{R}^{r\times r}$ then we have the alternative representation

$$A = \hat{U}\hat{\Sigma}\hat{V}^{\mathrm{T}},$$

which is also called *reduced singular value decomposition* of the matrix A. Since we can also write the reduced form as $A = \hat{U}(\hat{V}\hat{\Sigma})^{\mathrm{T}}$, we have the following result.

Corollary 1.20 *Every matrix $A \in \mathbb{R}^{m\times n}$ of* $\operatorname{rank}(A) = r$ *can be written in the form $A = BC^{\mathrm{T}}$ with $B \in \mathbb{R}^{m\times r}$ and $C \in \mathbb{R}^{n\times r}$.*

In particular, we see that every rank 1 matrix is necessarily of the form $\mathbf{b}\mathbf{c}^{\mathrm{T}}$ with $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^n$.
There are efficient methods available to compute the singular value decomposition and we will discuss some of them in Section 5.7.

1.6 Pseudo-inverse

We can use the singular value decomposition to define a kind of inverse to a singular and even non-square matrix.

Definition 1.21 Let $A \in \mathbb{R}^{m\times n}$ have rank $r = \operatorname{rank}(A)$. Let $A = U\Sigma V^{\mathrm{T}}$ be the singular value decomposition of A with orthogonal matrices $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ and with $\Sigma \in \mathbb{R}^{m\times n}$ being a diagonal matrix with non-zero diagonal entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. Let $\Sigma^+ \in \mathbb{R}^{n\times m}$ be the matrix

$$\Sigma^+ = \begin{pmatrix} 1/\sigma_1 & & & 0 & \dots & 0 \\ & \ddots & & \vdots & & \vdots \\ & & 1/\sigma_r & 0 & & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & & 0 & \dots & 0 \end{pmatrix}.$$

Then, the *pseudo-inverse* of A is defined as

$$A^+ := V\Sigma^+ U^{\mathrm{T}} \in \mathbb{R}^{n\times m}. \tag{1.7}$$

We know that the singular values of a matrix $A \in \mathbb{R}^{m\times n}$ are uniquely determined. However, the orthogonal matrices U and V are not unique and hence we have to make sure that the pseudo-inverse is well-defined before we investigate its properties.

Theorem 1.22 *Let $A \in \mathbb{R}^{m\times n}$ with $r = \mathrm{rank}(A)$.*

1. *Each pseudo-inverse $A^+ \in \mathbb{R}^{n\times m}$ of A satisfies*

$$AA^+ = (AA^+)^{\mathrm{T}}, \quad A^+A = (A^+A)^{\mathrm{T}}, \quad AA^+A = A, \quad A^+AA^+ = A^+. \tag{1.8}$$

2. *The properties*

$$AB = (AB)^{\mathrm{T}}, \quad BA = (BA)^{\mathrm{T}}, \quad ABA = A, \quad BAB = B \tag{1.9}$$

determine a matrix $B \in \mathbb{R}^{n\times m}$ uniquely. This means in particular that the pseudo-inverse is well-defined.

Proof 1. By definition of a pseudo-inverse, we have

$$AA^+ = (U\Sigma V^{\mathrm{T}})(V\Sigma^+ U^{\mathrm{T}}) = U\Sigma\Sigma^+ U^{\mathrm{T}}.$$

Since $\Sigma\Sigma^+ \in \mathbb{R}^{m\times m}$ is a diagonal matrix with r diagonal entries equal to 1 and $m - r$ diagonal entries equal to 0 it is obviously symmetric and hence $AA^+ = (AA^+)^{\mathrm{T}}$. The equality $A^+A = (A^+A)^{\mathrm{T}}$ is shown in the same way. Moreover, we have

$$AA^+A = (U\Sigma V^{\mathrm{T}})(V\Sigma^+ U^{\mathrm{T}})(U\Sigma V^{\mathrm{T}}) = U\Sigma\Sigma^+\Sigma V^{\mathrm{T}} = U\Sigma V^{\mathrm{T}} = A$$

and $A^+AA^+ = A^+$ follows in the same way.

2. Now assume that both B and C satisfy the stated equalities (1.9). Then, we can conclude that

$$\begin{aligned} B &= BAB = B(AB)^{\mathrm{T}} = (BB^{\mathrm{T}})A^{\mathrm{T}} = BB^{\mathrm{T}}A^{\mathrm{T}}C^{\mathrm{T}}A^{\mathrm{T}} = B(AB)^{\mathrm{T}}(AC)^{\mathrm{T}} \\ &= (BAB)AC = BAC = (BA)^{\mathrm{T}}C = (BA)^{\mathrm{T}}(CAC) = (BA)^{\mathrm{T}}(CA)^{\mathrm{T}}C \\ &= A^{\mathrm{T}}B^{\mathrm{T}}A^{\mathrm{T}}C^{\mathrm{T}}C = A^{\mathrm{T}}C^{\mathrm{T}}C = (CA)^{\mathrm{T}}C = CAC = C. \end{aligned}$$ □

From this we have the uniqueness of the pseudo-inverse. Moreover, we see that the pseudo-inverse is uniquely determined by the so-called *Penrose conditions* (1.9), i.e. we could have also used these conditions to define the pseudo-inverse.

Moreover, if $A \in \mathbb{R}^{n\times n}$ is invertible then $B = A^{-1}$ obviously satisfies the conditions (1.9). This means that for an invertible matrix the pseudo-inverse and the classical inverse are the same.
Finally, if $m \geq n$ and $\operatorname{rank}(A) = n$, the following representation of the pseudo-inverse will become important later on.

Lemma 1.23 *Let $A \in \mathbb{R}^{m\times n}$ satisfy $\operatorname{rank}(A) = n$. Then, the pseudo-inverse of A is given by*

$$A^+ = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}. \tag{1.10}$$

Proof Since $A \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(A) = n$, we find that $A^{\mathrm{T}}A \in \mathbb{R}^{n\times n}$ has also rank n and is hence invertible. Thus, the expression $B := (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}$ is well-defined. Moreover, B satisfies (1.9), since, using the symmetry of $A^{\mathrm{T}}A$, we have

$$\begin{aligned}
(AB)^{\mathrm{T}} &= (A(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}})^{\mathrm{T}} = A(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}} = AB,\\
(BA)^{\mathrm{T}} &= ((A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}A)^{\mathrm{T}} = I^{\mathrm{T}} = I = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}A = BA,\\
ABA &= A(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}A = A,\\
BAB &= ((A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}})A(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}} = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}} = B. \qquad \square
\end{aligned}$$

The pseudo-inverse will play an important role when it comes to solving and analysing least-squares problems. The next result relates the pseudo-inverse to the orthogonal projections onto the ranges of A and A^{T}.

Lemma 1.24 *Let $A \in \mathbb{R}^{m\times n}$ with pseudo-inverse $A^+ \in \mathbb{R}^{n\times m}$. Then,*

1. *$P := AA^+ : \mathbb{R}^m \to \mathbb{R}^m$ is the orthogonal projection onto* $\operatorname{range}(A)$*,*
2. *$P := A^+A : \mathbb{R}^n \to \mathbb{R}^n$ is the orthogonal projection onto* $\operatorname{range}(A^{\mathrm{T}})$*.*

Proof We will only prove the first statement since the proof of the second statement is very similar, as $\operatorname{range}(A^+) = \operatorname{range}(A^T)$.
By definition $P\mathbf{x} = A(A^+\mathbf{x}) \in \operatorname{range}(A)$. Moreover, using the properties (1.8) we find for each $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{u} = A\mathbf{v} \in \operatorname{range}(A)$ that

$$\begin{aligned}
\langle \mathbf{x} - P\mathbf{x}, \mathbf{u}\rangle_2 &= \langle \mathbf{x} - P\mathbf{x}, A\mathbf{v}\rangle_2 = \mathbf{v}^{\mathrm{T}}A^{\mathrm{T}}(\mathbf{x} - AA^+\mathbf{x})\\
&= \mathbf{v}^{\mathrm{T}}[A^{\mathrm{T}}\mathbf{x} - A^{\mathrm{T}}(AA^+)^{\mathrm{T}}\mathbf{x}] = \mathbf{v}^{\mathrm{T}}[A^{\mathrm{T}}\mathbf{x} - (AA^+A)^{\mathrm{T}}\mathbf{x}]\\
&= \mathbf{v}^{\mathrm{T}}[A^{\mathrm{T}}\mathbf{x} - A^{\mathrm{T}}\mathbf{x}] = 0,
\end{aligned}$$

which shows that $P\mathbf{x} = AA^+\mathbf{x}$ is indeed the orthogonal projection of $\mathbf{x}$ onto $\operatorname{range}(A)$. $\square$

Exercises

1.1 Let V be a pre-Hilbert space. Use $0 \le \langle \alpha\mathbf{x} + \beta\mathbf{y}, \alpha\mathbf{x} + \beta\mathbf{y}\rangle$ for $\alpha, \beta \in \mathbb{R}$ and $\mathbf{x}, \mathbf{y} \in V$ to prove the Cauchy–Schwarz inequality.

1.2 Let V be a normed space with norm $\|\cdot\| : V \to \mathbb{R}$. Show that V is a pre-Hilbert space and that the norm comes from an inner product $\langle\cdot,\cdot\rangle : V \times V \to \mathbb{R}$ if and only if the *parallelogram equality*

$$\|\mathbf{x}+\mathbf{y}\|^2 + \|\mathbf{x}-\mathbf{y}\|^2 = 2\|\mathbf{x}\|^2 + 2\|\mathbf{y}\|^2, \qquad \mathbf{x}, \mathbf{y} \in V$$

holds and that in this case the inner product satisfies

$$\langle \mathbf{x}, \mathbf{y}\rangle = \frac{1}{4}\left[\|\mathbf{x}+\mathbf{y}\|^2 - \|\mathbf{x}-\mathbf{y}\|^2\right], \qquad \mathbf{x}, \mathbf{y} \in V.$$

1.3 Let V be a pre-Hilbert space. Show that the norm induced by the inner product satisfies $\|\mathbf{x}+\mathbf{y}\| < 2$ for all $\mathbf{x} \neq \mathbf{y} \in V$ with $\|\mathbf{x}\| = \|\mathbf{y}\| = 1$.

1.4 Let $A \in \mathbb{R}^{n\times n}$ be symmetric and positive definite and let $C \in \mathbb{R}^{n\times m}$. Show:

1. the matrix $C^{\mathrm{T}}AC$ is positive semi-definite,
2. $\operatorname{rank}(C^{\mathrm{T}}AC) = \operatorname{rank}(C)$,
3. the matrix $C^{\mathrm{T}}AC$ is positive definite if and only if $\operatorname{rank}(C) = m$.

1.5 Show that a matrix $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite if and only if it is of the form $A = BB^{\mathrm{T}}$ with an invertible matrix $B \in \mathbb{R}^{n\times n}$.

1.6 Let $A, B, C \in \mathbb{R}^{2\times 2}$ with $C = AB$. Let

$$\begin{aligned}
p &= (a_{11} + a_{22})(b_{11} + b_{22}), & q &= (a_{21} + a_{22})b_{11},\\
r &= a_{11}(b_{12} - b_{22}), & s &= a_{22}(b_{21} - b_{11}),\\
t &= (a_{11} + a_{12})b_{22}, & u &= (a_{21} - a_{11})(b_{11} + b_{12}),\\
v &= (a_{12} - a_{22})(b_{21} + b_{22}).
\end{aligned}$$

Show that the elements of C can then be computed via

$$\begin{aligned}
c_{11} &= p + s - t + v, & c_{12} &= r + t,\\
c_{21} &= q + s, & c_{22} &= p + r - q + u.
\end{aligned}$$

Compare the number of multiplications and additions for this method with the number of multiplications and additions for the standard method of multiplying two 2×2 matrices.

Finally, show that if the above method is recursively applied to matrices $A, B \in \mathbb{R}^{n\times n}$ with $n = 2^k$ then the method requires 7^k multiplications and $6 \cdot 7^k - 6 \cdot 2^{2k}$ additions and subtractions.