Part II

Vector spaces and linear transformations

Chapter 2

Vector spaces

In this chapter we introduce the notion of a vector space, which is fundamental for the approximation methods that we will later develop, in particular in the form of an orthogonal projection onto a subspace representing the best possible approximation in that subspace.

Any vector in an vector space can be expressed in terms of a set of basis vectors, and we here introduce the process of constructing an orthonormal basis from an arbitrary basis, which provides the foundation for a range of matrix factorization methods we will use to solve systems of linear equations and eigenvalue problems.

We use the Euclidian space $\mathbb{R}^n$ as an illustrative example, but the concept of a vector space is much more general than that, forming the basis for the theory of function approximation and partial differential equations.

2.1 Vector spaces

Vector space

We denote the elements of $\mathbb{R}$, the real numbers, as *scalars*, and a *vector space*, or *linear space*, is then defined by a set V which is closed under two basic operations on V, *vector addition* and *scalar multiplication*,

(i) $x, y \in V \Rightarrow x + y \in V$,

(ii) $x \in V, \alpha \in \mathbb{R} \Rightarrow \alpha x \in V$,

satisfying the expected algebraic rules for addition and multiplication. A vector space defined over $\mathbb{R}$ is a *real vector space*. More generally, we can define vector spaces over the complex numbers $\mathbb{C}$, or any *algebraic field* $\mathbb{F}$.

The Euclidian space $\mathbb{R}^n$

The Euclidian space $\mathbb{R}^n$ is a vector space consisting of the set of column vectors

$$x = (x_1, ..., x_n)^T = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \tag{2.1}$$

where $(x_1, ..., x_n)$ is a row vector with $x_j \in \mathbb{R}$, and where v^T denotes the transpose of the vector v. In $\mathbb{R}^n$ the basic operations are defined by component-wise addition and multiplication, such that,

(i) $x + y = (x_1 + y_1, ..., x_n + y_n)^T$,

(ii) $\alpha x = (\alpha x_1, ..., \alpha x_n)^T$.

A geometrical interpretation of a vector space will prove to be useful. For example, the vector space $\mathbb{R}^2$ can be interpreted as the vector arrows in the Euclidian plane, defined by: (i) a direction with respect to a fixed point (origo), and (ii) a magnitude (the Euclidian length).

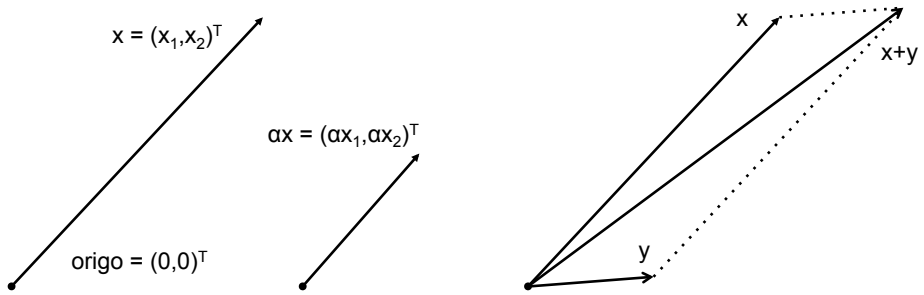

Figure 2.1: Geometrical interpretation of a vector $x = (x_1, x_2)^T$ in the Euclidian plane $\mathbb{R}^2$ (left), scalar multiplication αx with $\alpha = 0.5$ (center), and vector addition $x + y$ (right).

Vector subspace

A *subspace* of a vector space V is a subset $S \subset V$, such that S together with the basic operations in V defines a vector space in its own right. For example, the planes

$$S_1 = \{x \in \mathbb{R}^3 : x_3 = 0\}, \tag{2.2}$$

$$S_2 = \{x \in \mathbb{R}^3 : ax_1 + bx_2 + cx_3 = 0 : a, b, c \in \mathbb{R}\}, \tag{2.3}$$

are both subspaces of $\mathbb{R}^3$, see Figure 2.2.

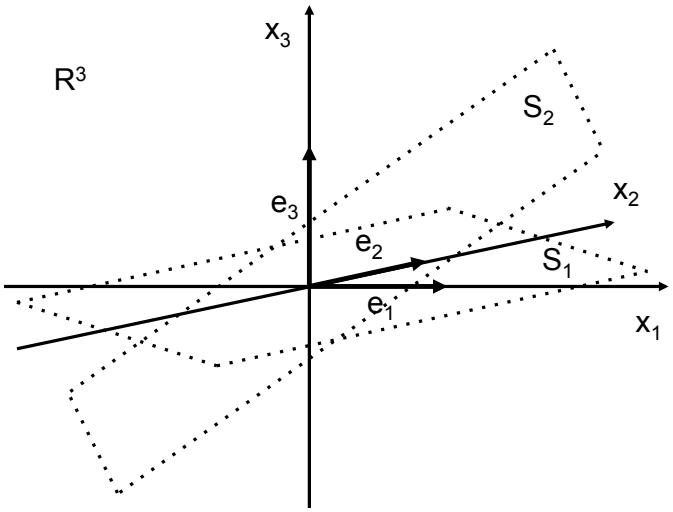

Figure 2.2: Illustration of the Euclidian space $\mathbb{R}^3$ with the three coordinate axes in the directions of the standard basis vectors e_1, e_2, e_3, and two subspaces S_1 and S_2, where S_1 is the x_1x_2-plane and S_2 a generic plane in $\mathbb{R}^3$ that includes origo, with the indicated planes extending to infinity.

Basis

For a set of vectors $\{v_i\}_{i=1}^n$ in V, we refer to the sum $\sum_{i=1}^n \alpha_i v_i$, with $\alpha_i \in \mathbb{R}$, as a *linear combination* of the set of vectors v_i. All possible linear combinations of the set of vectors v_i define a subspace,

$$S = \{v \in V : v = \sum_{i=1}^{n} \alpha_i v_i,\ \alpha_i \in \mathbb{R}\}, \tag{2.4}$$

and we say that the vector space S is *spanned* by the set of vectors $\{v_i\}_{i=1}^n$, denoted by $S = \text{span}\{v_i\}_{i=1}^n = \langle v_1, ..., v_n \rangle$.

We say that the set $\{v_i\}_{i=1}^n$ is *linearly independent*, if

$$\sum_{i=1}^{n} \alpha_i v_i = 0 \quad \Rightarrow \quad \alpha_i = 0, \quad \forall i = 1, ..., n. \tag{2.5}$$

A linearly independent set $\{v_i\}_{i=1}^n$ is a *basis* for the vector space V, if all $v \in V$ can be expressed as a linear combination of the vectors in the basis,

$$v = \sum_{i=1}^{n} \alpha_i v_i, \tag{2.6}$$

where $\alpha_i \in \mathbb{R}$ are the *coordinates* of v with respect to the basis $\{v_i\}_{i=1}^n$. The *dimension* of V, $\dim(V)$, is the number of vectors in any basis for V, and any basis of V has the same dimension.

The *standard basis* $\{e_1, ..., e_n\} = \{(1, 0, ..., 0)^T, ..., (0, ..., 0, 1)^T\}$ spans $\mathbb{R}^n$, such that any $x \in \mathbb{R}^n$ can be expressed as

$$x = \sum_{i=1}^{n} x_i e_i, \tag{2.7}$$

where $\dim \mathbb{R}^n = n$, and we refer to the coordinates $x_i \in \mathbb{R}$ in the standard basis as *Cartesian coordinates*.

Norm

To measure the size of vectors we introduce the *norm* $\|\cdot\|$ of a vector in the vector space V. A norm must satisfy the following conditions:

(i) $\|x\| \geq 0$, $\forall x \in V$, and $\|x\| = 0 \Leftrightarrow x = 0$,

(ii) $\|\alpha x\| = |\alpha| \|x\|$, $\forall x \in V, \alpha \in \mathbb{R}$,

(iii) $\|x + y\| \leq \|x\| + \|y\|$, $\forall x, y \in V$,

where (iii) is the *triangle inequality*.

A *normed vector space* is a vector space on which a norm is defined. For example, $\mathbb{R}^n$ is a normed vector space on which the *l_2-norm* is defined,

$$\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2} = (x_1^2 + ... + x_n^2)^{1/2}, \tag{2.8}$$

which corresponds to the *Euclidian length* of the vector x.

Inner product

A function $(\cdot, \cdot) : V \times V \to \mathbb{R}$ on the vector space V is an *inner product* if

(i) $(\alpha x + \beta y, z) = \alpha(x, z) + \beta(y, z)$,

(ii) $(x, \alpha y + \beta z) = \alpha(x, y) + \beta(x, z)$,

(iii) $(x, y) = (y, x)$,

(iv) $(x, x) \geq 0$, $\forall x \in V$, and $(x, x) = 0 \Leftrightarrow x = 0$,

for all $x, y, z \in V$ and $\alpha, \beta \in \mathbb{R}$.

An *inner product space* is a vector space on which an inner product is defined, and each inner product induces an associated norm by

$$\|x\| = (x, x)^{1/2}, \tag{2.9}$$

and thus an inner product space is also a normed space. An inner product and its associated norm satisfies the *Cauchy-Schwarz inequality*.

Theorem 1 (Cauchy-Schwarz inequality)**.** *For $\|\cdot\|$ the associated norm of the inner product $(\cdot, \cdot)$ in the vector space V, we have that*

$$|(x, y)| \leq \|x\|\|y\|, \quad \forall x, y \in V. \tag{2.10}$$

Proof. Let $s \in \mathbb{R}$ so that

$$0 \leq \|x + sy\|^2 = (x + sy, x + sy) = \|x\|^2 + 2s(x, y) + s^2\|y\|^2,$$

and then choose s as the minimizer of the right hand side of the inequality, that is, $s = -(x, y)/\|y\|^2$, which proves the theorem. □

The Euclidian space $\mathbb{R}^n$ is an inner product space with the *Euclidian inner product*, also referred to as scalar product or dot product, defined by

$$(x, y)_2 = x \cdot y = x_1 y_1 + ... + x_n y_n, \tag{2.11}$$

which induces the l_2-norm $\|x\|_2 = (x, x)_2^{1/2}$. In $\mathbb{R}^n$ we often drop the subscript for the Euclidian inner product and norm, with the understanding that $(x, y) = (x, y)_2$ and $\|x\| = \|x\|_2$.

We can also define general l_p-norms as

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \tag{2.12}$$

for $1 \leq p < \infty$. In Figure 2.3 we illustrate the l_1-norm,

$$\|x\|_1 = |x_1| + ... + |x_n|, \tag{2.13}$$

and the l_∞-norm, defined by

$$\|x\|_\infty = \max_{1 \leq p \leq n} |x_i|. \tag{2.14}$$

In fact, the Cauchy-Schwarz inequality is a special case of the *Hölder inequality* for general l_p-norms in $\mathbb{R}^n$.

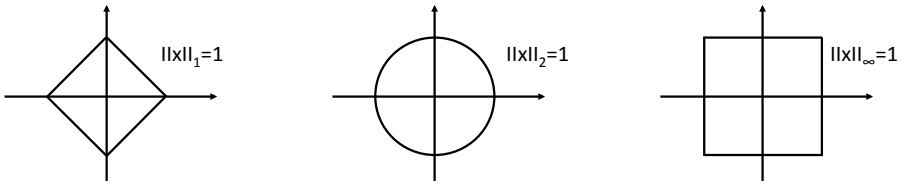

Figure 2.3: Illustration of l_p-norms in $\mathbb{R}^n$ through the unit circles $\|x\|_p = 1$, for $p = 1, 2, \infty$ (from left to right).

Theorem 2 (Hölder inequality)**.** *For* $1 \leq p, q \leq \infty$ *and* $1/p + 1/q = 1$*,*

$$|(x, y)| \leq \|x\|_p \|y\|_q, \quad \forall x, y \in \mathbb{R}^n. \tag{2.15}$$

In particular, we have that

$$|(x, y)| \leq \|x\|_1 \|y\|_\infty, \, \forall x, y \in \mathbb{R}^n. \tag{2.16}$$

2.2 Orthogonal projections

Orthogonality

An inner product space provides a means to generalize the concept of measuring angles between vectors, from the Euclidian plane to general vector spaces, where in particular two vectors x and y are *orthogonal* if $(x, y) = 0$.

If a vector $v \in V$ is orthogonal to all vectors s in a subspace $S \subset V$, that is

$$(v, s) = 0, \quad \forall s \in S,$$

then v is said to be orthogonal to S. For example, the vector $(0, 0, 1)^T \in \mathbb{R}^3$ is orthogonal to the subspace spanned in $\mathbb{R}^3$ by the vectors $(1, 0, 0)^T$ and $(0, 1, 0)^T$.

We denote by $S^\perp$ the *orthogonal complement* of S in V, defined as

$$S^\perp = \{v \in V : (v, s) = 0, \, \forall s \in S\}. \tag{2.17}$$

The only vector in V that is an element of both S and $S^\perp$ is the zero vector, and any vector $v \in V$ can be decomposed into two orthogonal components $s_1 \in S$ and $s_2 \in S^\perp$, such that $v = s_1 + s_2$, where the dimension of $S^\perp$ is equal to the *codimension* of the subspace S in V, that is

$$\dim(S^\perp) = \dim(V) - \dim(S). \tag{2.18}$$

Orthogonal projection

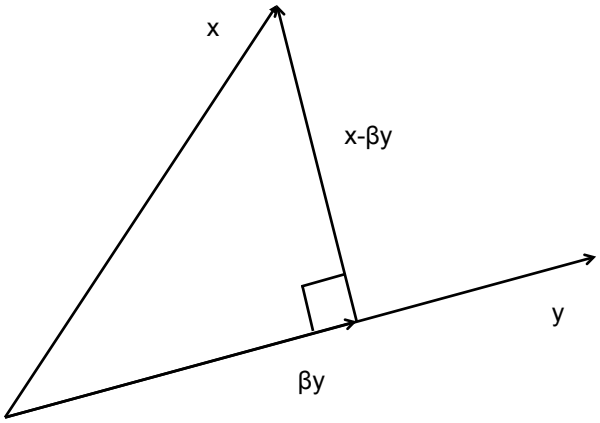

Figure 2.4: Illustration in the Euclidian plane $\mathbb{R}^2$ of βy, the projection of the vector x in the direction of the vector y, with $x - \beta y$ orthogonal to y.

The *orthogonal projection* of a vector x in the direction of another vector y, is the vector βy with $\beta = (x, y)/\|y\|^2$, such that the difference between the two vectors is orthogonal to y, that is

$$(x - \beta y, y) = 0. \tag{2.19}$$

Further, the orthogonal projection of a vector $v \in V$ onto the subspace $S \subset V$, is a vector $v_s \in S$ such that

$$(v - v_s, s) = 0, \quad \forall s \in S, \tag{2.20}$$

where v_s represents the best approximation of v in the subspace $S \subset V$, with respect to the norm induced by the inner product of V.

Theorem 3 (Best approximation property)**.**

$$\|v - v_s\| \leq \|v - s\|, \quad \forall s \in S \tag{2.21}$$

Proof. For any vector $s \in S$ we have that

$$\|v - v_s\|^2 = (v - v_s, v - v_s) = (v - v_s, v - s) + (v - v_s, s - v_s) = (v - v_s, v - s),$$

since $(v - v_s, s - v_s) = 0$, by (2.20) and the fact that $s - v_s \in S$. The result then follows from Cauchy-Schwarz inequality and division of both sides by $\|v - v_s\|$,

$$(v - v_s, v - s) \leq \|v - v_s\|\|v - s\| \Rightarrow \|v - v_s\| \leq \|v - s\|.$$

□

To emphasize the geometric properties of a inner product space V, it is sometimes useful to visualize a subspace S as a plane in $\mathbb{R}^3$, see Figure 2.5.

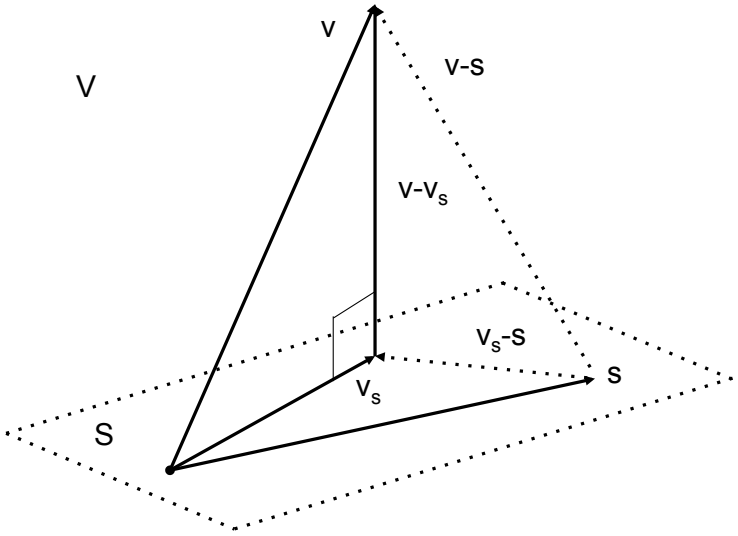

Figure 2.5: The orthogonal projection $v_s \in S$ is the best approximation of $v \in V$ in the subspace $S \subset V$.

Orthonormal basis

We refer to a set of non-zero vectors $\{v_i\}_{i=1}^n$ in the inner product space V as an *orthogonal set*, if all vectors v_i are pairwise orthogonal, that is if

$$(v_i, v_j) = 0, \quad \forall i \neq j. \tag{2.22}$$

If $\{v_i\}_{i=1}^n$ is an orthogonal set in the subspace $S \subset V$, and $\dim(S) = n$, then $\{v_i\}_{i=1}^n$ is a basis for S, that is all $v_s \in S$ can be expressed as

$$v_s = \alpha_1 v_1 + ... + \alpha_n v_n = \sum_{i=1}^{n} \alpha_i v_i, \tag{2.23}$$

with the coordinate $\alpha_i = (v_s, v_i)/\|v_i\|^2$ being the projection of v_s in the direction of the basis vector v_i.

If $Q = \{q_i\}_{i=1}^n$ is an orthogonal set, and $\|q_i\| = 1$ for all i, we say that Q is an *orthonormal set*. Let Q be an orthonormal basis for S, then

$$v_s = (v_s, q_1)q_1 + ... + (v_s, q_n)q_n = \sum_{i=1}^{n} (v_s, q_i)q_i, \quad \forall v_s \in S, \tag{2.24}$$

where the coordinate (v_s, q_i) is the projection of the vector v_s onto the basis vector q_i. An arbitrary vector $v \in V$ can be expressed as

$$v = r + \sum_{i=1}^{n} (v, q_i)q_i, \tag{2.25}$$

where the vector $r = v - \sum_{i=1}^{n} (v, q_i)q_i$ is orthogonal to S, that is $r \in S^\perp$, a fact that we will use repeatedly.

Thus the vector $r \in V$ satisfies the orthogonality condition

$$(r, s) = 0, \quad \forall s \in S, \tag{2.26}$$

and from (2.21) we know that r is the vector in V that corresponds to the minimal projection error of the vector v onto S with respect to the norm in V. We will refer to the vector r as the *residual*.

2.3 Excercises

Problem 1. *Prove that the plane S_1 is a subspace of $\mathbb{R}^3$, where $S_1 = \{x \in \mathbb{R}^3 : x_3 = 0\}$. Under what condition is the plane $S_2 = \{x \in \mathbb{R}^3 : ax_1 + bx_2 + cx_3 + d = 0 : a, b, c, d \in \mathbb{R}\}$ a subspace of $\mathbb{R}^3$?*

Problem 2. *Prove that the standard basis in $\mathbb{R}^n$ is linearly independent.*

Problem 3. *Prove that the Euclidian l_2-norm $\|\cdot\|_2$ is a norm.*

Problem 4. *Prove that the scalar product $(\cdot, \cdot)_2$ is an inner product.*

Problem 5. *Prove that $\|\cdot\|_2$ is induced by the inner product $(\cdot, \cdot)_2$.*

Problem 6. *Prove that* $|(x, y)| \leq \|x\|_1 \|y\|_\infty$, $\forall x, y \in \mathbb{R}^n$.

Problem 7. *Prove that the vector* $(0, 0, 1)^T \in \mathbb{R}^3$ *is orthogonal to the subspace spanned in* $\mathbb{R}^3$ *by the vectors* $(1, 0, 0)^T$ *and* $(0, 1, 0)^T$.

Problem 8. *Let* $\{q_i\}_{i=1}^n$ *be an orthonormal basis for the subspace* $S \subset V$, *prove that* $r \in S^\perp$, *with* $r = v - \sum_{i=1}^n (v, q_i) q_i$.

Problem 9. *Let* $\{w_i\}_{i=1}^n$ *be a basis for the subspace* $S \subset V$, *so that all* $s \in S$ *can be expressed as* $s = \sum_{i=1}^n \alpha_i w_i$.

(a) Prove that (2.20) is equivalent to finding the vector $v_s \in S$ *that satisfies the* n *equations of the form*

$$(v - v_s, w_i) = 0, \quad i = 1, ..., n.$$

(b) Since $v_s \in S$, *we have that* $v_s = \sum_{j=1}^n \beta_j w_j$. *Prove that (2.20) is equivalent to finding the set of coordinates* β_i *that satisfies*

$$\sum_{j=1}^n \beta_j (w_j, w_i) = (v, w_i), \quad i = 1, ..., n.$$

(c) Let $\{q_i\}_{i=1}^n$ *be an orthonormal basis for the subspace* $S \subset V$, *so that we can express* $v_s = \sum_{j=1}^n \beta_j q_j$. *Use the result in (b) to prove that (2.20) is equivalent to the condition that the coordinates are given as* $\beta_j = (v, q_j)$.

Chapter 3

Linear transformations and matrices

Linear transformations, or linear maps, between vector spaces represent an important class of functions, in their own right, but also as approximations of more general nonlinear transformations.

A linear transformation acting on a Euclidian vector can be represented by a matrix. Many of the concepts we introduce in this chapter generalize to linear transformations acting on functions in infinite dimensional spaces, for example integral and differential operators, which are fundamental for the study of differential equations.

3.1 Matrix algebra

Linear transformation as a matrix

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ defines a *linear transformation*, if

(i) $f(x + z) = f(x) + f(z)$,

(ii) $f(\alpha x) = \alpha f(x)$,

for all $x, z \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$. In the standard basis $\{e_1, ..., e_n\}$ we can express the ith component of the vector $y = f(x) \in \mathbb{R}^m$ as

$$y_i = f_i(x) = f_i(\sum_{j=1}^{n} x_j e_j) = \sum_{j=1}^{n} x_j f_i(e_j),$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ for all $i = 1, ..., m$. In component form, we write this as

$$\begin{aligned} y_1 &= a_{11}x_1 + ... + a_{1n}x_n \\ &\vdots \\ y_m &= a_{m1}x_1 + ... + a_{mn}x_n \end{aligned} \tag{3.1}$$

with $a_{ij} = f_i(e_j)$. That is $y = Ax$, where A is an $m \times n$ *matrix*,

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}. \tag{3.2}$$

The set of real valued $m \times n$ matrices defines a vector space $\mathbb{R}^{m\times n}$, by the basic operations of (i) component-wise matrix addition, and (ii) component-wise scalar multiplication, that is

$$A + B = \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & \cdots & a_{mn} + b_{mn} \end{bmatrix}, \quad \alpha A = \begin{bmatrix} \alpha a_{11} & \cdots & \alpha a_{1n} \\ \vdots & \ddots & \vdots \\ \alpha a_{m1} & \cdots & \alpha a_{mn} \end{bmatrix},$$

with $A, B \in \mathbb{R}^{m\times n}$ and $\alpha \in \mathbb{R}$.

Matrix-vector product

A matrix $A \in \mathbb{R}^{m\times n}$ defines a *linear map* $x \mapsto Ax$, by the operations of *matrix-vector product* and *component-wise scalar multiplication*,

$$\begin{aligned} A(x + y) &= Ax + Ay, & x, y \in \mathbb{R}^n, \\ A(\alpha x) &= \alpha Ax, & x \in \mathbb{R}^n, \alpha \in \mathbb{R}. \end{aligned}$$

In *index notation* we write a vector $b = (b_i)$, and a matrix $A = (a_{ij})$, with i the *row index* and j is the *column index*. For an $m \times n$ matrix A, and x an n-dimensional column vector, we define the matrix-vector product $b = Ax$ to be the m-dimensional column vector $b = (b_i)$, such that

$$b_i = \sum_{j=1}^{n} a_{ij}x_j, \quad i = 1, ..., m. \tag{3.3}$$

With a_j the jth column of A, an m-dimensional column vector, we can express the matrix-vector product as a linear combination of the set of column vectors $\{a_j\}_{j=1}^n$,

$$b = Ax = \sum_{j=1}^{n} x_j a_j, \tag{3.4}$$

or in matrix form

$$\left[\begin{array}{c|c|c|c} & & & \\ b & & & \\ & & & \end{array}\right] = \left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \cdots & a_n \\ & & & \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \left[\begin{array}{c} \\ a_1 \\ \\ \end{array}\right] + x_2 \left[\begin{array}{c} \\ a_2 \\ \\ \end{array}\right] + ... + x_n \left[\begin{array}{c} \\ a_n \\ \\ \end{array}\right].$$

The vector space spanned by $\{a_j\}_{j=1}^n$ is the *column space*, or *range*, of the matrix A, so that $\text{range}(A) = \text{span}\{a_j\}_{j=1}^n$. The *null space*, or *kernel*, of an $m \times n$ matrix A is the set of vectors $x \in \mathbb{R}^n$ such that $Ax = 0$, with 0 the zero vector in $\mathbb{R}^m$, that is $\text{null}(A) = \{x \in \mathbb{R}^n : Ax = 0\}$.

The dimension of the column space is the column *rank* of the matrix A, $\text{rank}(A)$. We note that the column rank is equal to the row rank, corresponding to the space spanned by the row vectors of A, and the maximal rank of an $m \times n$ matrix is $\min(m, n)$, which we refer to as *full rank*.

Matrix-matrix product

The *matrix-matrix product* $B = AC$ is a matrix in $\mathbb{R}^{l \times n}$, defined for two matrices $A \in \mathbb{R}^{l \times m}$ and $C \in \mathbb{R}^{m \times n}$, as

$$b_{ij} = \sum_{k=1}^{m} a_{ik} c_{kj}, \tag{3.5}$$

with $B = (b_{ij})$, $A = (a_{ik})$ and $C = (c_{kj})$.

We sometimes omit the summation sign and use the *Einstein convention*, where repeated indices imply summation over those same indices, so that we express the matrix-matrix product (3.5) simply as $b_{ij} = a_{ik} c_{kj}$.

Similarly as for the matrix-vector product, we may interpret the columns b_j of the matrix-matrix product B as a linear combination of the columns a_k with coefficients c_{kj}

$$b_j = Ac_j = \sum_{k=1}^{m} c_{kj} a_k, \tag{3.6}$$

or in matrix form

$$\left[\begin{array}{c|c|c|c} & & & \\ b_1 & b_2 & \cdots & b_n \\ & & & \end{array}\right] = \left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \cdots & a_m \\ & & & \end{array}\right] \left[\begin{array}{c|c|c|c} & & & \\ c_1 & c_2 & \cdots & c_n \\ & & & \end{array}\right].$$

The composition $f \circ g(x) = f(g(x))$, of two linear transformations $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^l$, with associated matrices $A \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{l \times w}$, corresponds to the matrix-matrix product AC acting on $x \in \mathbb{R}^n$.

Matrix transpose and the inner and outer products

The *transpose* (or *adjoint*) of an $m \times n$ matrix $A = (a_{ij})$ is defined as the matrix $A^T = (a_{ji})$, with the column and row indices reversed.

Using the matrix transpose, the inner product of two vectors $v, w \in \mathbb{R}^n$ can be expressed in terms of a matrix-matrix product $v^T w$, as

$$(v, w) = v^T w = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix} = v_1 w_1 + ... + v_n w_n. \tag{3.7}$$

Similarly, the *outer product*, or *tensor product*, of two vectors $v, w \in \mathbb{R}^n$, denoted by $v \otimes w$, is defined as the $m \times n$ matrix corresponding to the matrix-matrix product vw^T, that is

$$v \otimes w = vw^T = \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix} \begin{bmatrix} w_1 & \cdots & w_n \end{bmatrix} = \begin{bmatrix} v_1 w_1 & \cdots & v_1 w_n \\ \vdots & & \vdots \\ v_m v_1 & & v_m w_n \end{bmatrix}.$$

In tensor notation we can express the inner and the outer products as $(v, w) = v_i w_i$ and $v \otimes w = v_i w_j$, respectively.

The transpose has the property that $(AB)^T = B^T A^T$, and thus satisfies the equation $(Ax, y) = (x, A^T y)$, for any $x \in \mathbb{R}^n, y \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$, which follows from the definition of the inner product in Euclidian vector spaces, since

$$(Ax, y) = (Ax)^T y = x^T A^T y = (x, A^T y). \tag{3.8}$$

A square matrix $A \in \mathbb{R}^{n \times n}$ is said to be *symmetric* (or *self-adjoint*) if $A = A^T$, so that $(Ax, y) = (x, Ay)$. If in addition $(Ax, x) > 0$ for all non-zero $x \in \mathbb{R}^n$, we say that A is a *symmetric positive definite* matrix. A square matrix is said to be *normal* if $A^T A = AA^T$.

Matrix norms

To measure the size of a matrix, we first introduce the *Frobenius norm*, corresponding to the l_2-norm of the matrix A interpreted as an mn-vector, that is

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}. \tag{3.9}$$

The Frobenius norm is induced by the following inner product over the space $\mathbb{R}^{m\times n}$,

$$(A, B) = \mathrm{tr}(A^T B), \tag{3.10}$$

with the *trace* of a square $n \times n$ matrix $C = (c_{ij})$ defined by

$$\mathrm{tr}(C) = \sum_{i=1}^{n} c_{ii}. \tag{3.11}$$

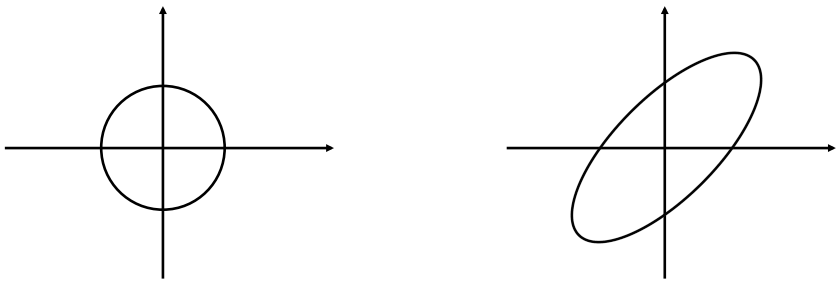

Figure 3.1: Illustration of the map $x \mapsto Ax$; of the unit circle $\|x\|_2 = 1$ (left) to the ellipse Ax (right), corresponding to the matrix A in (3.13).

Matrix norms for $A \in \mathbb{R}^{m\times n}$ are also induced by the respective l_p-norms on $\mathbb{R}^m$ and $\mathbb{R}^n$, in the form

$$\|A\|_p = \sup_{\substack{x\in\mathbb{R}^n \\ x\neq 0}} \frac{\|Ax\|_p}{\|x\|_p} = \sup_{\substack{x\in\mathbb{R}^n \\ \|x\|_p=1}} \|Ax\|_p. \tag{3.12}$$

The last equality follows from the definition of a norm, and shows that the induced matrix norm can be defined in terms of its map of unit vectors, which we illustrate in Figure 3.1 for the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix}. \tag{3.13}$$

We further have the following inequality,

$$\|Ax\|_p \leq \|A\|_p \|x\|_p, \tag{3.14}$$

which follows from (3.12).

Determinant

The *determinant* of a square matrix A is denoted $\det(A)$ or $|A|$. For a 2×2 matrix we have the explicit formula

$$\det(A) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc. \tag{3.15}$$

For example, the determinant of the matrix A in (3.13) is computed as $\det(A) = 1 \cdot 2 - 2 \cdot 0 = 2$.

The formula for the determinant is extended to a 3×3 matrix by

$$\begin{aligned} \det(A) &= \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix} \\ &= a(ei - fh) - b(di - fg) + c(dh - eg), \end{aligned} \tag{3.16}$$

and by recursion this formula can be generalized to any square matrix.

For a 2×2 matrix the absolute value of the determinant is equal to the area of the parallelogram that represents the image of the unit square under the map $x \mapsto Ax$, and similarly for a 3×3 matrix the volume of the parallelepiped representing the mapped unit cube. More generally, the absolute value of the determinant $\det(A)$ represents a scale factor of the linear transformation A.

Matrix inverse

If the column vectors $\{a_j\}_{j=1}^n$ of a square $n \times n$ matrix A form a basis for $\mathbb{R}^n$, then all vectors $b \in \mathbb{R}^n$ can be expressed as $b = Ax$, where the vector $x \in \mathbb{R}^n$ holds the coordinates of b in the basis $\{a_j\}_{j=1}^n$.

In particular, all $x \in \mathbb{R}^n$ can be expressed as $x = Ix$, where I is the square $n \times n$ *identity matrix* in $\mathbb{R}^n$, taking the standard basis as column vectors,

$$x = Ix = \left[\begin{array}{c|c|c|c} & & & \\ e_1 & e_2 & \cdots & e_n \\ & & & \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

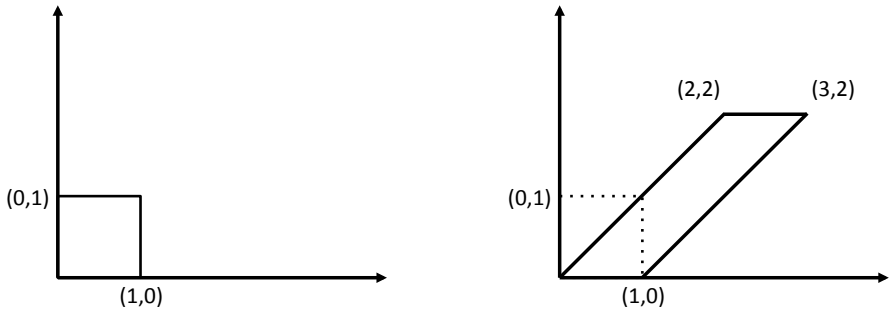

Figure 3.2: The map $x \mapsto Ax$ (right) of the unit square (left), for the matrix A in (3.13), with the corresponding area given as $|\det(A)| = 2$.

with the vector entries x_i corresponding to the Cartesian coordinates of the vector x.

A square matrix $A \in \mathbb{R}^{n\times n}$ is *invertible*, or *non-singular*, if there exists an *inverse matrix* $A^{-1} \in \mathbb{R}^{n\times n}$, such that

$$A^{-1}A = AA^{-1} = I, \tag{3.17}$$

which also means that $(A^{-1})^{-1} = A$. Further, for two $n \times n$ matrices A and B, we have the property that $(AB)^{-1} = B^{-1}A^{-1}$.

Theorem 4 (Inverse matrix)**.** *For a square matrix $A \in \mathbb{R}^{n\times n}$, the following statements are equivalent:*

(i) *A has an inverse A^{-1},*

(ii) $\det(A) \neq 0$,

(iii) $\text{rank}(A) = n$,

(iv) $\text{range}(A) = \mathbb{R}^n$

(v) $\text{null}(A) = \{0\}$.

The matrix inverse is unique. To see this, assume that there exist two matrices B_1 and B_2 such that $AB_1 = AB_2 = I$; which by linearity gives that $A(B_1 - B_2) = 0$, but since $\text{null}(A) = \{0\}$ we have that $B_1 = B_2$.

3.2 Orthogonal projectors

Orthogonal matrix

A square matrix $Q \in \mathbb{R}^{n\times n}$ is *ortogonal*, or *unitary*, if $Q^T = Q^{-1}$. With q_j the columns of Q we thus have that $Q^TQ = I$, or in matrix form,

$$\left[\begin{array}{c} \\ \hline \\ \hline \\ \hline \\ \end{array}\right.\!\!\begin{array}{c} q_1 \\ \hline q_2 \\ \hline \vdots \\ \hline q_n \end{array}\!\!\left.\begin{array}{c} \\ \hline \\ \hline \\ \hline \\ \end{array}\right] \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \cdots & q_n \\ & & & \end{array}\right] = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix},$$

so that the columns q_j form an orthonormal basis for $\mathbb{R}^n$.

Multiplication by an orthogonal matrix preserves the angle between two vectors $x, y \in \mathbb{R}^n$, since

$$(Qx, Qy) = (Qx)^T Qy = x^T Q^T Qy = x^T y = (x, y), \tag{3.18}$$

and thus also the length of a vector,

$$\|Qx\| = (Qx, Qx)^{1/2} = (x, x)^{1/2} = \|x\|. \tag{3.19}$$

For example, counter-clockwise rotation by an angle θ in $\mathbb{R}^2$, takes the form of multiplication by an orthogonal matrix,

$$Q_{rot} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \tag{3.20}$$

whereas reflection in the line with a slope given by the angle θ, corresponds to multiplication by the orthogonal matrix,

$$Q_{ref} = \begin{bmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{bmatrix}, \tag{3.21}$$

where we note the general fact that for a rotation $\det(Q_{rot}) = 1$, and for a reflection $\det(Q_{ref}) = -1$.

Orthogonal projector

A *projection matrix*, or *projector*, is a square matrix P such that

$$P^2 = PP = P. \tag{3.22}$$

It follows that

$$Pv = v, \tag{3.23}$$

for all vectors $v \in \text{range}(P)$, since v is of the form $v = Px$ for some x, and thus $Pv = P^2x = Px = v$. For $v \notin \text{range}(P)$ we have that $P(Pv - v) = P^2v - Pv = 0$, so that the projection error $Pv - v \in \text{null}(P)$.

The matrix $I - P$ is also a projector, the *complementary projector* to P, since $(I - P)^2 = I - 2P + P^2 = I - P$. The range and null space of the two projectors are related as

$$\text{range}(I - P) = \text{null}(P), \tag{3.24}$$

and

$$\text{range}(P) = \text{null}(I - P), \tag{3.25}$$

so that P and $I - P$ separates $\mathbb{R}^n$ into the two subspaces $S_1 = \text{range}(P)$ and $S_2 = \text{range}(I - P)$, since the only $v \in \text{range}(P) \cap \text{range}(I - P)$ is the zero vector; $v = v - Pv = (I - P)v = \{0\}$.

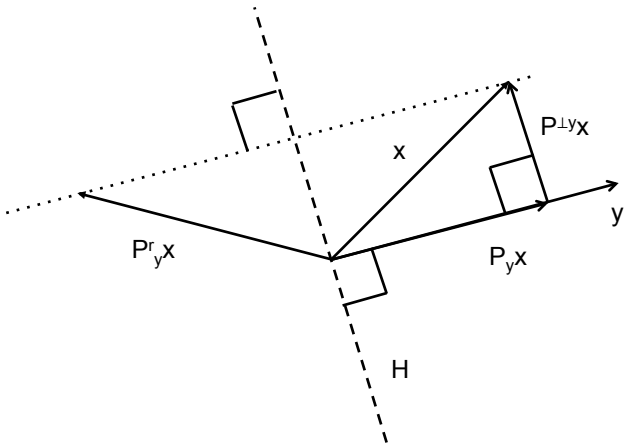

Figure 3.3: The projector $P_y x$ of a vector x in the direction of another vector y, its orthogonal complement $P^{\perp y}x$, and $P_y^r x$, the reflector of x in the hyperplane H defined by y as a normal vector.

If the two subspaces S_1 and S_2 are orthogonal, we say that P is an *orthogonal projector*. This is equivalent to the condition $P = P^T$, since the inner product between two vectors in S_1 and S_2 then vanish,

$$(Px, (I - P)y) = (Px)^T(I - P)y = x^T P^T(I - P)y = x^T(P - P^2)y = 0,$$

and if P is an orthogonal projector, so is $I - P$.

For example, the orthogonal projection of one vector x in the direction of another vector y, expressed in (2.19), corresponds to an orthogonal projector P_y, by

$$\frac{(x,y)y}{\|y\|^2} = \frac{y(y,x)}{\|y\|^2} = \frac{y(y^T x)}{\|y\|^2} = \frac{yy^T}{\|y\|^2}x = P_y x. \tag{3.26}$$

Similarly we can define the orthogonal complement $P^{\perp y}x$, and $P_y^r x$, the reflection of x in the *hyperplane* H defined by y as a normal vector, so that

$$P_y = \frac{yy^T}{\|y\|^2}, \quad P^{\perp y} = I - \frac{yy^T}{\|y\|^2}, \quad P_y^r = I - 2\frac{yy^T}{\|y\|^2}, \tag{3.27}$$

defines orthogonal projectors, where we note that a hyperplane is a subspace in V of codimension 1.

3.3 Exercises

Problem 10. *Formulate the matrix-vector and matrix-matrix products in pseudo code.*

Problem 11. *Prove the equivalence of the definitions of the induced matrix norm, defined by*

$$\|A\|_p = \sup_{\substack{x\in\mathbb{R}^n \\ x\neq 0}} \frac{\|Ax\|_p}{\|x\|_p} = \sup_{\substack{x\in\mathbb{R}^n \\ \|x\|_p=1}} \|Ax\|_p. \tag{3.28}$$

Problem 12. *For $A \in \mathbb{R}^{m\times l}, B \in \mathbb{R}^{l\times n}$, prove that $(AB)^T = B^T A^T$.*

Problem 13. *For $A, B \in \mathbb{R}^{n\times n}$, prove that $(AB)^{-1} = B^{-1}A^{-1}$.*

Problem 14. *Prove the inequality (3.14).*

Problem 15. *Prove that an orthogonal matrix is normal.*

Problem 16. *Show that the matrices A and B are orthogonal and compute their determinants. Which matrix represents a rotation and reflection, respectively?*

$$A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad B = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{bmatrix} \tag{3.29}$$

Problem 17. *For P a projector, prove that* $\text{range}(I - P) = \text{null}(P)$*, and that* $\text{range}(P) = \text{null}(I - P)$.

Problem 18. *For the vector $y = (1, 0)^T$, compute the action of the projectors $P_y, P^{\perp y}, P_y^r$ on a general vector $x = (x_1, x_2)^T$.*

Chapter 4

Linear operators in $\mathbb{R}^n$

In this chapter we give some examples of linear operators in the vector space $\mathbb{R}^n$, used extensively in various fields, including computer graphics, robotics, computer vision, image processing, and computer aided design.

We also meet differential equations for the first time, in the form of matrix operators acting on discrete approximations of functions, defined by their values at the nodes of a grid.

4.1 Differential and integral operators

Difference and summation matrices

Subdivide the interval $[0, 1]$ into a *structured grid* $\mathcal{T}^h$ with n intervals and $n + 1$ *nodes* x_i, such that $0 = x_0 < x_1 < x_2 < ... < x_n = 1$, with a constant interval length, or *grid size*, $h = x_i - x_{i-1}$ for all i, so that $x_i = x_0 + ih$.

For each $x = x_i$, we may approximate the primitive function $F(x)$ of a function $f(x)$, expressed here as a definite integral with $f(0) = 0$, by

$$F(x_i) = \int_0^{x_i} f(s)ds \approx \sum_{k=1}^{i} f(x_k)h \equiv F_h(x_i), \tag{4.1}$$

which defines a function $F_h(x_i) \approx F(x_i)$ for all nodes x_i in the subdivision, based on *Riemann sums*.

The function F_h defines a linear transformation L_h of the vector of sampled function values at the nodes $y = (f(x_1), ..., f(x_n))^T$, which can

be expressed by the following matrix equation,

$$L_h y = \begin{bmatrix} h & 0 & \cdots & 0 \\ h & h & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ h & h & \cdots & h \end{bmatrix} \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix} = \begin{bmatrix} f(x_1)h \\ f(x_1)h + f(x_2)h \\ \vdots \\ \sum_{k=1}^{n} f(x_k)h \end{bmatrix}, \qquad (4.2)$$

where L_h is a summation matrix, with an associated inverse L_h^{-1},

$$L_h = h \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \quad \Rightarrow \quad L_h^{-1} = h^{-1} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & -1 & 1 \end{bmatrix}. \qquad (4.3)$$

The inverse matrix L_h^{-1} corresponds to a difference matrix over the same subdivision $\mathcal{T}^h$, approximating the slope (derivative) of the function $f(x)$. To see this, multiply the matrix L_h^{-1} to $y = (f(x_i))$,

$$L_h^{-1} y = h^{-1} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & -1 & 1 \end{bmatrix} \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix} = \begin{bmatrix} (f(x_1) - f(x_0))/h \\ (f(x_2) - f(x_1))/h \\ \vdots \\ (f(x_n) - f(x_{n-1}))/h \end{bmatrix},$$

where we recall that $f(x_0) = f(0) = 0$.

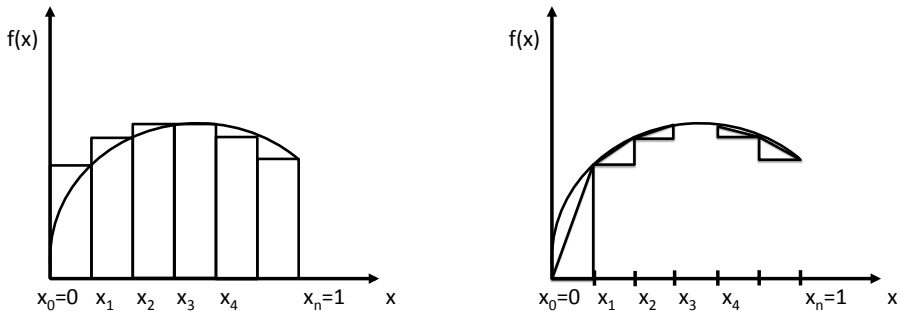

Figure 4.1: Approximation of the integral of a function $f(x)$ in the form of Riemann sums (left), and approximation of the derivative of $f(x)$ by slopes computed from function values in the nodes x_i (right), on a subdivision of $[0, 1]$ with interval length h.

As the interval length $h \to 0$, the summation and difference matrices converge to integral and differential operators, such that for each $x \in (0, 1)$,

$$L_h y \to \int_0^x f(s)ds, \quad L_h^{-1} y \to f'(x). \tag{4.4}$$

Further, we have for the product of the two matrices that

$$y = L_h L_h^{-1} y \to f(x) = \int_a^x f'(s)ds, \tag{4.5}$$

as $h \to 0$, which corresponds to the *Fundamental theorem of Calculus*.

Difference operators

The matrix L_h^{-1} in (4.3) corresponds to a backward difference operator D_h^-, and similarly we can define a forward difference operator D_h^+, by

$$D_h^- = h^{-1} \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & -1 & 1 & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}, \quad D_h^+ = h^{-1} \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \\ 0 & \cdots & 0 & 0 & -1 \end{bmatrix}.$$

The matrix-matrix product $D_h^+ D_h^-$ takes the form,

$$D_h^+ D_h^- = h^{-2} \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}, \tag{4.6}$$

which corresponds to an approximation of a second order differential operator. The matrix $A = -D_h^+ D_h^-$ is *diagonally dominant*, that is

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|, \tag{4.7}$$

and symmetric positive definite, since

$$\begin{aligned} x^T Ax &= \ldots + x_i(-x_{i-1} + 2x_i - x_{i+1}) + \ldots + x_n(-x_{n-1} + 2x_n) \\ &= \ldots - x_i x_{i-1} + 2x_i^2 - x_i x_{i+1} - x_{i+1} x_i + \ldots - x_{n-1} x_n + 2x_n^2 \\ &= \ldots + (x_i - x_{i-1})^2 + (x_{i+1} - x_i)^2 + \ldots + x_n^2 > 0, \end{aligned}$$

for any non-zero vector x.

The finite difference method

For a vector $y = (u(x_i))$, the ith row of the matrix $D_h^+ D_h^-$ corresponds to a *finite difference stencil*, with $u(x_i)$ function values sampled at the nodes x_i of the structured grid representing the subdivision of the interval $I = (0, 1)$,

$$\begin{aligned}[(D_h^+ D_h^-)y]_i &= \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} \\ &= \frac{\dfrac{u(x_{i+1}) - u(x_i)}{h} - \dfrac{u(x_i) - u(x_{i-1})}{h}}{h}.\end{aligned}$$

Similarly, the difference operators D_h^- and D_h^+ correspond to finite difference stencils over the grid, and we have that for $x \in I$,

$$(D_h^+ D_h^-)y \to u''(x), \quad (D_h^-)y \to u'(x), \quad (D_h^+)y \to u'(x), \tag{4.8}$$

as the grid size $h \to 0$.

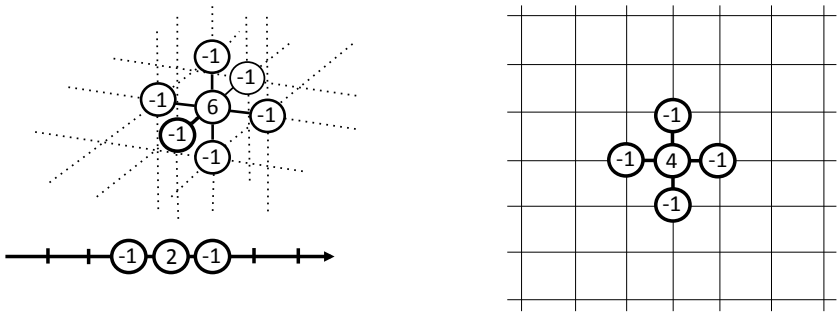

Figure 4.2: Examples of finite difference stencils corresponding to the difference operator $-(D_h^+ D_h^-)$ over structured grids in $\mathbb{R}$ (lower left), $\mathbb{R}^2$ (right) and $\mathbb{R}^3$ (upper left).

The *finite difference method* for solving differential equations is based on approximation of differential operators by such difference stencils over a grid. We can thus, for example, approximate the differential equation

$$-u''(x) + u(x) = f(x), \tag{4.9}$$

by the matrix equation

$$-(D_h^+ D_h^-)y + (D_h^-)y = b, \tag{4.10}$$

with $b_i = (f(x_i))$. The finite difference method extends to multiple dimensions, where the difference stencils are defined over structured (Cartesian) grids in $\mathbb{R}^2$ or $\mathbb{R}^3$, see Figure 4.2.

Solution of differential equations

Since the second order difference matrix $A = -(D_h^+ D_h^-)$ is symmetric positive definite, there exists a unique inverse A^{-1}. For example, in the case of $n = 5$ and the difference matrix A below, we have that

$$A = 1/h^2 \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \quad \Rightarrow \quad A^{-1} = h^2/6 \begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 8 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 8 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}.$$

The matrix A^{-1} corresponds to a symmetric integral (summation) operator, where the matrix elements decay with the distance from the diagonal. The integral operator has the property that when multiplied to a vector $y = (y_i)$, each element y_i is transformed into an average of all the vector elements of y, with most weight given to the elements close to y_i.

Further, for $y = (u(x_i))$ and $b = (f(x_i))$, the solution to the differential equation

$$-u''(x) = f(x) \tag{4.11}$$

can be approximated by

$$y = A^{-1}b. \tag{4.12}$$

We can thus compute approximate solutions for any function $f(x)$ on the right hand side of the equation (4.11). Although, we note that while the $n \times n$ matrix A is *sparse*, with only few non-zero elements near the diagonal, the inverse A^{-1} is a *dense matrix* without zero elements.

In general the dense matrix A^{-1} has a much larger memory footprint than the sparse matrix A. Therefore, for large matrices, it may be impossible to hold the matrix A^{-1} in memory, so that instead iterative solution methods must be used without the explicit construction of the matrix A^{-1}.

4.2 Projective geometry

Affine transformations

An *affine transformation*, or *affine map*, is a linear transformation composed with a translation, corresponding to a multiplication by a matrix A, followed by addition of a position vector b, that is

$$x \mapsto Ax + b. \tag{4.13}$$

For example, an object defined by a set of vectors in $\mathbb{R}^2$ can be scaled by a diagonal matrix, or rotated by a *Givens rotation* matrix,

$$A_{rot} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \tag{4.14}$$

with θ a counter-clockwise rotation angle.

Any triangle in the Euclidian plane $\mathbb{R}^2$ is related to each other through an invertible affine map. There also exists affine maps from $\mathbb{R}^2$ to a surface (manifold) in $\mathbb{R}^3$, although such a map is not invertible, see Figure 4.4.

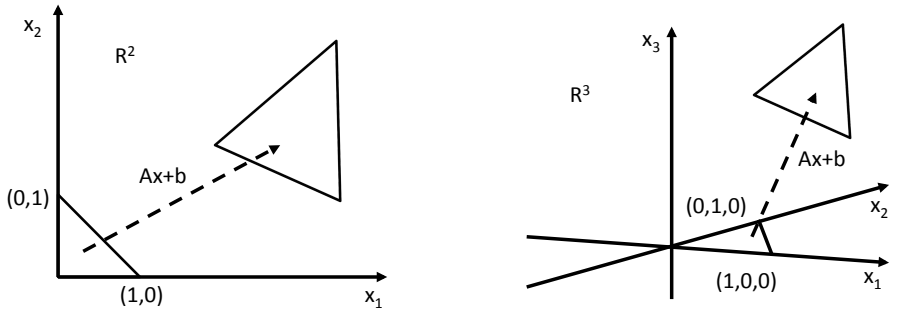

Figure 4.3: Affine maps $x \mapsto Ax + b$ of the *reference triangle*, with corners in $(0, 0)$, $(1, 0)$, $(0, 1)$; in $\mathbb{R}^2$ (left); to a surface (manifold) in $\mathbb{R}^3$ (right).

Homogeneous coordinates

By using *homogeneous coordinates*, or *projective coordinates*, we can express any affine transformation as one single matrix multiplication, including translation. The underlying definition is that the representation of a geometric object x is homogeneous if $\lambda x = x$, for all real numbers $\lambda \neq 0$.

An $\mathbb{R}^2$ vector $x = (x_1, x_2)^T$ in standard Cartesian coordinates is represented as $x = (x_1, x_2, 1)^T$ in homogeneous coordinates, from which follows that any object $u = (u_1, u_2, u_3)$ in homogeneous coordinates can be expressed in Cartesian coordinates, by

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} u_1/u_3 \\ u_2/u_3 \\ 1 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} u_1/u_3 \\ u_2/u_3 \end{bmatrix}. \tag{4.15}$$

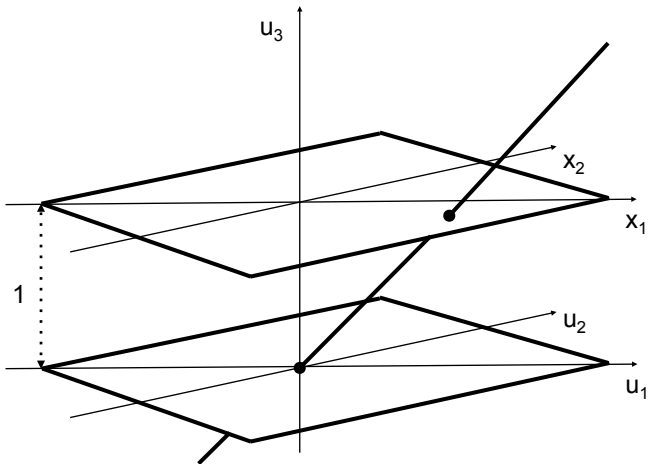

Figure 4.4: The relation of homogeneous (projective) coordinates and Cartesian coordinates.

It follows that in homogeneous coordinates, rotation by an angle θ and translation by a vector (t_1, t_2), both can be expressed as matrices,

$$A_{rot} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad A_{trans} = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.16}$$

An advantage of homogenous coordinates is also the ability to apply combinations of transformations by multiplying the respective matrices, which is used extensively e.g. in robotics, computer vision, and computer graphics. For example, an affine transformation can be expressed by the matrix-matrix product $A_{trans}A_{rot}$.

4.3 Computer graphics

Vector graphics

Vector graphics is based the representation of primitive geometric objects defined by a set of parameters, such as a circle in $\mathbb{R}^2$ defined by its center and radius, or a cube in $\mathbb{R}^3$ defined by its corner points. Lines and polygones are other common objects, and for special purposes more advances objects

are used, such as NURBS (Non-uniform rational B-splines) for computer aided design (CAD), and PostScript fonts for digital type setting.

These objects my be characterized by their parameter values in the form of vectors in $\mathbb{R}^n$, and operations on such objects can be defined by affine transformations acting on the vectors of parameters.

Raster graphics

Whereas vector graphics describes an image in terms of geometric objects such as lines and curves, *raster graphics* represent an image as an array of color values positioned in a grid pattern. In 2D each square cell in the grid is called a *pixel* (from picture element), and in 3D each cube cell is known as a *voxel* (volumetric pixel).

In 2D image processing, the operation of a *convolution*, or *filter*, is the multiplication of each pixel and its neighbours by a *convolution matrix*, or *kernel*, to produce a new image where each pixel is determined by the kernel, similar to the stencil operators in the finite difference method.

Common kernels include the *Sharpen* and *Gaussian blur* filters,

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \qquad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \tag{4.17}$$

where we note the similarity to the finite difference stencil of the second order derivative (Laplacian) and its inverse.

Figure 4.5: Raster image (left), transformed by a Sharpen (middle) and a blur (right) filters.

Part III

Algebraic equations

Chapter 5

Linear system of equations

In this chapter we study methods for solving linear systems of equations. That is, we seek a solution in terms of a vector x that satisfies a set of linear equations that can be formulated as a matrix equation $Ax = b$.

For a square non-singular matrix A, we can construct direct solution methods based on factorization of the matrix A into a product of matrices that are easy to invert. In the case of a rectangular matrix A we formulate a least squares problem, where we seek a solution x that minimizes the norm of the residual $b - Ax$.

5.1 Linear system of equations

A linear system of equations can be expressed as the matrix equation

$$Ax = b, \tag{5.1}$$

with A a given matrix and b a given vector, for which x is the unknown solution vector. Given our previous discussion, b can be interpreted as the image of x under the linear transformation A, or alternatively, x can be interpreted as the coefficients of b expressed in the column space of A.

For a square non-singular matrix A the solution x can be expressed in terms of the inverse matrix as

$$x = A^{-1}b. \tag{5.2}$$

For some matrices the inverse matrix A^{-1} is easy to construct, such as in the case of a *diagonal matrix* $D = (d_{ij})$, for which $d_{ij} = 0$ for all $i \neq j$. Here the inverse is directly given as $D^{-1} = (d_{ij}^{-1})$. Similarly, for an orthogonal matrix Q the inverse is given by the transpose $Q^{-1} = Q^T$. On the contrary,

for a general matrix A, computation of the inverse is not straight forward. Instead we seek to transform the general matrix into a product of matrices that are easy to invert.

We will introduce two factorizations that can be used for solving $Ax = b$, in the case of A being a general square non-singular matrix; *QR factorization* and *LU factorization*. Factorization followed by inversion of the factored matrix is an example of a *direct method* for solving $Ax = b$. We note that to solve the equation we do not have to construct the inverse matrix explicitly, instead we only need to compute the action of matrices on a vector, which is important in terms of the memory footprint of the algorithms.

Triangular matrices

Apart from diagonal and orthogonal matrices, triangular matrices are easy to invert. We distinguish between two classes of triangular matrices: a *lower triangular matrix* $L = (l_{ij})$, with $l_{ij} = 0$ for $i < j$, and an *upper triangular matrix* $U = (u_{ij})$, with $u_{ij} = 0$ for $i > j$. The product of lower triangular matrices is lower triangular, and the product of upper triangular matrices is upper triangular. Similarly, the inverse of a lower triangular matrix is lower triangular, and the inverse of an upper triangular matrix is upper triangular.

The equations $Lx = b$ and $Ux = b$, take the form

$$\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{12} & l_{22} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ l_{1n} & l_{2n} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \quad \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

solved by *forward substitution* and *backward substitution*, respectively,

$$\begin{aligned} x_1 &= \frac{b_1}{l_{11}} & x_n &= \frac{b_n}{u_{nn}} \\ x_2 &= \frac{b_2 - l_{21}x_1}{l_{22}} & x_{n-1} &= \frac{b_{n-1} - u_{n-1n}x_n}{u_{n-1n-1}} \\ & \cdots & & \cdots \\ x_n &= \frac{b_n - \sum_{i=1}^{n-1} l_{ni}x_i}{l_{nn}} & x_1 &= \frac{b_1 - \sum_{i=2}^{n} u_{1i}x_i}{u_{11}} \end{aligned}$$

where both algorithms correspond to $\mathcal{O}(n^2)$ operations.

5.2 QR factorization

Classical Gram-Schmidt orthogonalization

For a square matrix $A \in \mathbb{R}^{n\times n}$ we denote the successive vector spaces spanned by its column vectors a_j as

$$\langle a_1\rangle \subseteq \langle a_1, a_2\rangle \subseteq \langle a_1, a_2, a_3\rangle \subseteq ... \subseteq \langle a_1, ..., a_m\rangle. \tag{5.3}$$

Assuming that A has full rank, for each such vector space we now construct an orthonormal basis q_j, such that $\langle q_1, ..., q_j\rangle = \langle a_1, ..., a_j\rangle$, for all $j \leq n$.

Given a_j, we can successively construct vectors v_j that are orthogonal to the spaces $\langle q_1, ..., q_{j-1}\rangle$, since by (2.25) we have that

$$v_j = a_j - \sum_{i=1}^{j-1}(a_j, q_i)q_i, \tag{5.4}$$

for all $j = 1, ..., n$, where each vector is then normalized to get $q_j = v_j/\|v_j\|$. This is the *classical Gram-Schmidt iteration*.

Modified Gram-Schmidt orthogonalization

If we let $\hat{Q}_{j-1}$ be an $n \times (j-1)$ matrix with the column vectors q_i, for $i \leq j-1$, we can rewrite (5.4) in terms of an orthogonal projector P_j,

$$v_j = a_j - \sum_{i=1}^{j-1}(a_j, q_i)q_i = a_j - \sum_{i=1}^{j-1} q_i q_i^T a_j = (I - \hat{Q}_{j-1}\hat{Q}_{j-1}^T)a_j = P_j a_j,$$

with $\hat{Q}_{j-1}\hat{Q}_{j-1}^T$ an orthogonal projector onto range$(\hat{Q}_{j-1})$, the column space of $\hat{Q}_{j-1}$. The matrix $P_j = I - \hat{Q}_{j-1}\hat{Q}_{j-1}^T$ is thus an orthogonal projector onto the space orthogonal to range$(\hat{Q}_{j-1})$, with $P_1 = I$. The Gram-Schmidt iteration can therefore be expressed in terms of the projector P_j as

$$q_j = P_j a_j/\|P_j a_j\|, \tag{5.5}$$

for $j = 1, ..., n$.

Alternatively, P_j can be constructed by successive multiplication of projectors $P^{\perp q_i} = I - q_i q_i^T$, orthogonal to each individual vector q_i, such that

$$P_j = P^{\perp q_{j-1}} \cdots P^{\perp q_2} P^{\perp q_1}. \tag{5.6}$$

The *modified Gram-Schmidt iteration* corresponds to instead using this formula to construct P_j, which leads to a more robust algorithm than the classical Gram-Schmidt iteration.

Algorithm 1: Modified Gram-Schmidt iteration

```
for i = 1 to n do
    v_i = a_i
end
for i = 1 to n do
    r_ii = ||v_i||
    q_i = v_i / r_ii
    for j = 1 to i + 1 do
        r_ij = q_i^T v_j
        v_j = v_j − r_ij q_i
    end
end
```

QR factorization

By introducing the notation $r_{ij} = (a_j, q_i)$ and $r_{ii} = \|a_j - \sum_{i=1}^{j-1}(a_j, q_i)q_i\|$, we can rewrite the classical Gram-Schmidt iteration (5.4) as

$$\begin{aligned} a_1 &= r_{11}q_1 \\ a_2 &= r_{12}q_1 + r_{22}q_2 \\ &\vdots \\ a_n &= r_{1n}q_1 + ... + r_{2n}q_n \end{aligned} \tag{5.7}$$

which corresponds to the *QR factorization* $A = QR$, with a_j the column vectors of the matrix A, Q an orthogonal matrix and R an upper triangular matrix, that is

$$\left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \cdots & a_n \\ & & & \end{array}\right] = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \cdots & q_n \\ & & & \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix}.$$

Existence and uniqueness of the QR factorization of a non-singular matrix A follows by construction from Algorithm 1.

The modified Gram-Schmidt iteration of Algorithm 1 corresponds to successive multiplication of upper triangular matrices R_k on the right of the matrix A, such that the resulting matrix Q is an orthogonal matrix,

$$AR_1R_2 \cdots R_n = Q, \tag{5.8}$$

and with the notation $R^{-1} = R_1R_2 \cdots R_n$, the matrix $R = (R^{-1})^{-1}$ is also an upper triangular matrix.

Householder QR factorization

Whereas Gram-Schmidt iteration amounts to *triangular orthogonalization* of the matrix A, we may alternatively formulate an algorithm for *orthogonal triangularization*, where entries below the diagonal of A are zeroed out by successive application of orthogonal matrices Q_k, so that

$$Q_n ... Q_2 Q_1 A = R, \tag{5.9}$$

where we note that the matrix product $Q = Q_n ... Q_2 Q_1$ is also orthogonal.

In the *Householder algorithm*, orthogonal matrices are chosen of the form

$$Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix}, \tag{5.10}$$

with I the $(k-1) \times (k-1)$ identity matrix, and with F an $(n-k+1) \times (n-k+1)$ orthogonal matrix. Q_k is constructed to successively introducing $n-k$ zeros below the diagonal of the kth column of A, while leaving the upper $k-1$ rows untouched, thus taking the form

$$Q_k \hat{A}_{k-1} = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ 0 & \hat{A}_{22} \end{bmatrix} = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ 0 & F\hat{A}_{22} \end{bmatrix}, \tag{5.11}$$

with $\hat{A}_{k-1} = Q_{k-1} \cdots Q_2 Q_1 A$, and with $\hat{A}_{ij}$ representing the *sub-matrices*, or *blocks*, of $\hat{A}_{k-1}$ with corresponding block structure as Q_k.

To obtain a triangular matrix, F should introduce zeros in all subdiagonal entries of the matrix. We want to construct F such that for x an $n-k+1$ column vector, we get

$$Fx = \begin{bmatrix} \pm\|x\| \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \pm\|x\| e_1, \tag{5.12}$$

with $e_1 = (1, 0, ..., 0)^T$ a standard basis vector.

Further, we need F to be an orthogonal matrix, which we achieve by formulating F in the form of a reflector, so that Fx is the reflection of x in a hyperplane orthogonal to the vector $v = \pm\|x\| e_1 - x$, that is

$$F = I - 2\frac{vv^T}{v^T v}. \tag{5.13}$$

We now formulate the full algorithm for QR factorization based on this *Householder reflector*, where we use the notation $A_{i:j,k:l}$ for a sub-matrix

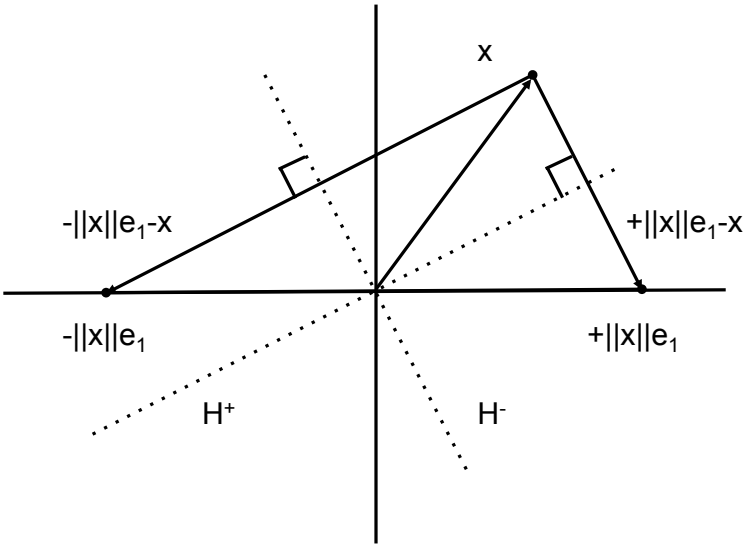

Figure 5.1: Householder reflectors across the two hyperplanes H^+ and H^-.

Algorithm 2: Householder QR factorization

for $k = 1$ **to** n **do**
- $x = A_{k:n,k}$
- $v_k = \text{sign}(x_1)\|x\|_2 e_1 + x$
- $v_k = v_k / \|v_k\|$
- $A_{k:n,k:n} = A_{k:n,k:n} - 2v_k(v_k^T A_{k:n,k:n})$

end

with a row index in the range $i, ..., j$, and column index in the range $k, ..., l$.

We note that Algorithm 2 does not explicitly construct the matrix Q, although from the vectors v_k we can compute the matrix-vector product with $Q = Q_1 Q_2 \cdots Q_n$ or $Q^T = Q_n \cdots Q_2 Q_1$.

5.3 LU factorization

Similar to Householder triangulation, *Gaussian elimination* transforms a square $n \times n$ matrix A into an upper triangular matrix U, by successively

inserting zeros below the diagonal. In the case of Gaussian elimination, this is done by adding multiples of each row to the other rows, which corresponds to multiplication by a sequence of triangular matrices L_k from the left, so that

$$L_{n-1} \cdots L_2 L_1 A = U. \tag{5.14}$$

By setting $L^{-1} = L_{n-1} \cdots L_2 L_1$, we obtain the factorization $A = LU$, with $L = L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1}$.

The kth step in the Gaussian elimination algorithm involves division by the diagonal element u_{kk}, and thus for stability of the algorithm in finite precision arithmetics it is necessary to avoid a small number in that position, which is achieved by reordering the rows, or *pivoting*. With a permutation matrix P, the LU factorization with pivoting may be expressed as $PA = LU$.

Algorithm 3: Gaussian elimination with pivoting

```
Starting from the matrices U = A, L = I, P = I
for k = 1 to n − 1 do
    Select i ≥ k to maximize |u_ik|
    Interchange the rows k and i in the matrices U, L, P
    for j = k + 1 to n do
        l_jk = u_jk / u_kk
        u_{j,k:n} = u_{j,k:n} − l_jk u_{k,k:n}
    end
end
```

Cholesky factorization

For the case of a symmetric positive definite matrix, A can be decomposed into a product of a lower triangular matrix L and its transpose L^T, which is referred to as the *Cholesky factorization*,

$$A = LL^T. \tag{5.15}$$

In the Cholesky factorization algorithm, symmetry is exploited to perform Gaussian elimination from both the left and right of the matrix A at the same time, which results in an algorithm at half the computational cost of LU factorization.

5.4 Least squares problems

We now consider a system of linear of equations $Ax = b$, for which we have n unknowns but $m > n$ equations, that is $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$.

There exists no inverse matrix A^{-1}, and if the vector $b \notin \text{range}(A)$ we say that the system is *overdetermined*, and thus no exact solution x exists to the equation $Ax = b$. Instead we seek the solution $x \in \mathbb{R}^n$ that minimizes the l_2-norm of the *residual* $b - Ax \in \mathbb{R}^m$, which is referred to as the *least squares problem*

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2. \tag{5.16}$$

A geometric interpretation is that we seek the vector $x \in \mathbb{R}^n$ such that the Euclidian distance between Ax and b is minimal, which corresponds to

$$Ax = Pb, \tag{5.17}$$

where $P \in \mathbb{R}^{m\times m}$ is the orthogonal projector onto range(A).

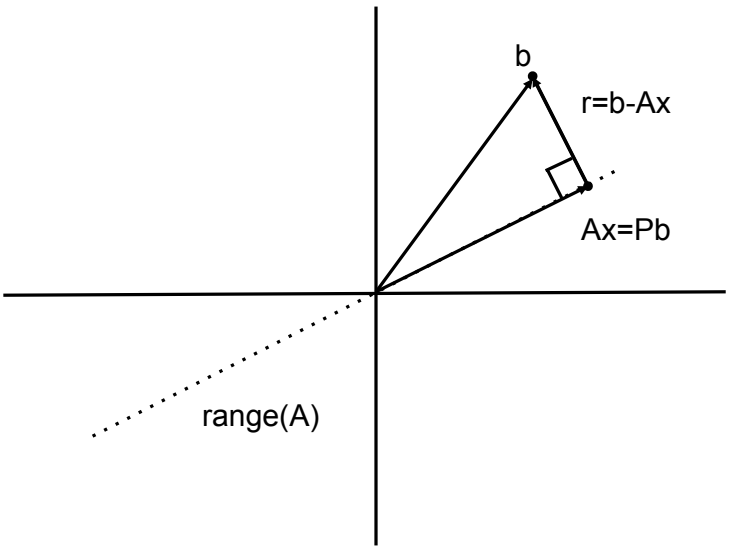

Figure 5.2: Geometric interpretation of the least squares problem.

Thus the residual $r = b - Ax$ is orthogonal to range(A), that is $(Ay, r) = (y, A^T r) = 0$, for all $y \in \mathbb{R}^n$, so that (5.16) is equivalent to

$$A^T r = 0, \tag{5.18}$$

which corresponds to the $n \times n$ system

$$A^T Ax = A^T b, \tag{5.19}$$

referred to as the *normal equations*.

The normal equations thus provide a way to solve the $m \times n$ least squares problem by solving instead a square $n \times n$ system. The square matrix $A^T A$ is nonsingular if and only if A has full rank, for which the solution is given as $x = (A^T A)^{-1} A^T b$, where the matrix $(A^T A)^{-1} A^T$ is known as the *pseudoinverse* of A.

5.5 Exercises

Problem 19. *Prove that the product of lower triangular matrices is lower triangular, and the product of upper triangular matrices is upper triangular.*

Problem 20. *Test the algorithms for QR and LU factorization for a* 3×3 *matrix* A.

Problem 21. *Implement the algorithms for QR and LU factorization, and test the computer program for* $n \times n$ *matrices with* n *large.*

Problem 22. *Derive the normal equations for the system*

$$\begin{bmatrix} -2 & 3 \\ -1 & 4 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ -3 \end{bmatrix}. \tag{5.20}$$

Chapter 6

The eigenvalue problem

An eigenvalue of a square matrix represents the linear transformation as a scaling of the associated eigenvector, and the action of certain matrices on general vectors can be represented as a scaling of the set of basis vectors used to represent the vector.

If iterated some of these basis vectors will be amplified, whereas others will decay, which can charaterize the stability of iterative algorithms, or a physical system described by the matrix, and also implies algorithms for computational approximation of eigenvalues and eigenvectors based on power iteration. We also present the QR algorithm, based on constructing a similarity transform of the matrix.

6.1 Eigenvalues and eigenvectors

Complex vector spaces

In this section we change the focus from real vector spaces to complex vector spaces. We let $z \in \mathbb{C}$ denote a complex scalar, of the form $z = a + ib$, with $i^2 = -1$, and $a, b \in \mathbb{R}$ the real and imaginary parts, for which we define the complex conjugate as $\bar{z} = a - ib$.

The complex vector space $\mathbb{C}^n$ is defined by the basic operations of componentwise addition and scalar multiplication of complex numbers, and with the transpose of a vector $x \in \mathbb{C}^n$ replaced by the *adjoint* x^*, corresponding to the transpose of the vector with the entries replaced by their complex conjugates. Similarly, the adjoint of a complex $m \times n$ matrix $A = (a_{ij})$ is the $n \times m$ matrix $A^* = (\bar{a}_{ji})$. If $A = A^*$ the matrix A is *Hermitian*, and if $AA^* = A^*A$ it is *normal*.

The inner product of $x, y \in \mathbb{C}^n$ is defined by

$$(x, y) = x^* y = \sum_{i=i}^{n} \bar{x}_i y_i, \tag{6.1}$$

with the associated norm for $x \in \mathbb{C}^n$,

$$\|x\| = (x, x)^{1/2}. \tag{6.2}$$

Matrix spectrum and eigenspaces

We consider a square matrix $A \in \mathbb{C}^{n \times n}$ acting on a complex vector space $\mathbb{C}^n$. An *eigenvector* of A is a nonzero vector $x \in \mathbb{C}^n$, such that

$$Ax = \lambda x, \tag{6.3}$$

with $\lambda \in \mathbb{C}$ the corresponding *eigenvalue*. If A is a nonsingular matrix then λ^{-1} is an eigenvalue of the inverse matrix A^{-1}, which is clear from multiplication of (6.3) by A^{-1} from the left,

$$A^{-1}Ax = A^{-1}\lambda x \Leftrightarrow x = A^{-1}\lambda x \Leftrightarrow \lambda^{-1}x = A^{-1}x. \tag{6.4}$$

The set of all eigenvalues is denoted the *spectrum* of A,

$$\Lambda(A) = \{\lambda_j\}_{j=1}^{n}, \tag{6.5}$$

and the *spectral radius* of A is defined as

$$\rho(A) = \max_{\lambda_j \in \Lambda(A)} |\lambda_j|. \tag{6.6}$$

The sum and the product of all eigenvalues are related to the trace and the determinant of A as

$$\det(A) = \prod_{j=1}^{n} \lambda_j, \qquad \mathrm{tr}(A) = \sum_{j=1}^{n} \lambda_j. \tag{6.7}$$

The subspace of $\mathbb{C}^n$ spanned by the eigenvectors corresponding to λ, together with the zero vector, is the *eigenspace* E_λ. The eigenspace E_λ is an *invariant subspace* under A, so that $AE_\lambda \subseteq E_\lambda$, and $\dim(E_\lambda)$ is the number of linearly independent eigenvectors corresponding to the eigenvalue λ, known as the *geometric multiplicity* of λ.

We have that the eigenspace $E_\lambda = \mathrm{null}(\lambda I - A)$, since $(\lambda I - A)x = 0$, and thus for a nonempty eigenspace E_λ, the matrix $\lambda I - A$ is singular, so that

$$\det(\lambda I - A) = 0. \tag{6.8}$$

Characteristic equation

The *characteristic polynomial* of the matrix $A \in \mathbb{C}^{n\times n}$, is the degree n polynomial

$$p_A(z) = \det(zI - A), \tag{6.9}$$

with $z \in \mathbb{C}$. For λ an eigenvalue of A, we thus have that

$$p_A(\lambda) = 0, \tag{6.10}$$

which we refer to as the *characteristic equation*, and by the fundamental theorem of algebra we can express $p_A(\lambda)$ as

$$p_A(\lambda) = (z - \lambda_1)(z - \lambda_2)\cdots(z - \lambda_n), \tag{6.11}$$

where each λ_j is an eigenvalue of A, not necessary unique. The multiplicity of each eigenvalue λ as a root to the equation $p_A(\lambda) = 0$ is the *algebraic multiplicity* of λ, where an eigenvalue is said to be *simple* if its algebraic multiplicity is 1. The algebraic multiplicity of an eigenvalue λ is at least as great as its geometric multiplicity.

Theorem 5 (Caley-Hamilton theorem)**.** *Every nonsingular matrix satisfies its own characteristic equation, $p_A(A) = 0$, that is,*

$$A^n + c_{n-1}A^{n-1} + ... + c_1A^1 + c_0I = 0, \tag{6.12}$$

for some constants c_i, which gives that

$$A^{-1} = \frac{1}{c_0}(A^{n-1} + c_{n-1}A^{n-2} + ... + c_1I), \tag{6.13}$$

thus that the inverse matrix A^{-1} is equal to a sum of powers of A.

Eigenvalue decompositions

A *defective matrix* is a matrix which has one or more *defective eigenvalues*, where a defective eigenvalue is an eigenvalue for which its algebraic multiplicity exceeds its geometric multiplicity.

Theorem 6 (Eigenvalue decomposition)**.** *Each nondefective matrix $A \in \mathbb{C}^{n\times n}$ has an eigenvalue decomposition*

$$A = X\Lambda X^{-1}, \tag{6.14}$$

where $X \in \mathbb{C}^{n\times n}$ is a nonsingular matrix with the eigenvectors of A as column vectors, and where $\Lambda \in \mathbb{C}^{n\times n}$ is diagonal matrix with the eigenvalues of A on the diagonal.

We also say that a nondefective matrix is *diagonalizable*. Given that the factorization (6.14) exits, we have that

$$AX = X\Lambda, \tag{6.15}$$

which expresses (6.3) as

$$Ax_j = \lambda_j x_j, \tag{6.16}$$

with λ_j the jth diagonal entry of Λ, and x_j the jth column of X.

For some matrices eigenvectors can be chosen to be pairwise orthogonal, so that a matrix A is *unitary diagonalizable*, that is

$$A = Q\Lambda Q^*, \tag{6.17}$$

with $Q \in \mathbb{C}^{n\times n}$ an orthogonal matrix with orthonormal eigenvectors of A as column vectors, and $\Lambda \in \mathbb{C}^{n\times n}$ a diagonal matrix with the eigenvalues of A on the diagonal.

Theorem 7. *A matrix is unitary diagonalizable if and only if it is normal.*

Hermitian matrices have real eigenvalues, and thus in the particular case of a real symmetric matrix, no complex vector spaces are needed to characterize the matrix spectrum and eigenspaces.

Theorem 8. *An Hermitian matrix is unitary diagonalizable with real eigenvalues.*

Irrespectively if the matrix is nondefective or Hermitian, any square matrix always has a Schur factorization, with the diagonal matrix replaced by an upper triangular matrix.

Theorem 9 (Schur factorization)**.** *For every square matrix A there exists a Schur factorization*

$$A = QTQ^*, \tag{6.18}$$

where Q is an orthogonal matrix, and T is an upper triangular matrix with the eigenvalues of A on the diagonal.

More generally, if $X \in \mathbb{C}^{n\times n}$ is nonsingular, the map $A \mapsto X^{-1}AX$ is a *similarity transformation* of A, and we say that two matrices A and B are *similar* if there exists a similarity transformation such that $B = X^{-1}AX$.

Theorem 10. *Two similar matrices have the same eigenvalues with the same algebraic and geometric multiplicity.*

6.2 Eigenvalue algorithms

QR algorithm

To compute the eigenvalues of a matrix A, one may seek the roots of the characteristic polynomial. Although, for a large matrix polynomial root finding is expensive and unstable. Instead the most efficient algorithms are based on computing eigenvalues and eigenvectors by constructing one of the factorizations (6.14), (6.17) or (6.18).

We now present the *QR algorithm*, in which a Schur factorization (6.18) of a matrix A is constructed from successive QR factorizations.

Algorithm 4: QR algorithm

$A^{(0)} = A$
for $k = 1, 2, ...$ **do**
  $Q^{(k)}R^{(k)} = A^{(k-1)}$
  $A^k = R^{(k)}Q^{(k)}$
end

We note that for each iteration $A^{(k)}$ of the algorithm, we have that

$$A^{(k)} = R^{(k)}Q^{(k)} = (Q^{(k)})^{-1}A^{(k-1)}Q^{(k)}, \tag{6.19}$$

so that $A^{(k)}$ and $A^{(k-1)}$ are similar, and thus have the same eigenvalues. Under suitable assumptions $A^{(k)}$ will converge to an upper triangular matrix, or in the case of a Hermitian matrix a diagonal matrix, with the eigenvalues on the diagonal.

The basic QR algorithm can be accelerated: (i) by Householder reflectors to reduce the initial matrix $A^{(0)}$ to *Hessenberg form*, that is a matrix with zeros below the first subdiagonal (or in the case of an Hermitian matrix a *tridiagonal form*), (ii) by introducing a *shift* to instead of $A^{(k)}$ factorize the matrix $A^{(k)} - \mu^{(k)}I$, which has identical eigenvectors, and where $\mu^{(k)}$ an eigenvalue estimate, and (iii) if any off-diagonal element is close to zero, all off-diagonal elements on the row are zeroed out to deflate the matrix $A^{(k)}$ into submatrices on which the QR algorithm is then applied.

Rayleigh quotient

To simplify the presentation, in the rest of this section we restrict attention to matrices that are real and symmetric, for which all eigenvalues λ_j are real and the corresponding eigenvectors q_j are orthonormal.

We now consider the question: given a vector $x \in \mathbb{R}^n$, what is the real number $\alpha \in \mathbb{R}$ that best approximate an eigenvalue of A in the sense that $\|Ax - \alpha x\|$ is minimized?

If $x = q_j$ is an eigenvector of A, then $\alpha = \lambda_j$ is the corresponding eigenvalue. If not, α is the solution to the $n \times 1$ least squares problem

$$\min_{\alpha \in \mathbb{R}} \|Ax - \alpha x\|, \tag{6.20}$$

for which the normal equations are given as

$$x^T Ax = x^T \alpha x. \tag{6.21}$$

With $\alpha = r(x)$, we define the *Rayleigh quotient* as

$$r(x) = \frac{x^T Ax}{x^T x}, \tag{6.22}$$

where $r(x)$ is an approximation of an eigenvalue λ_j, if x is close to the eigenvector q_j. In fact, $r(x)$ converges quadratically to $r(q_j) = \lambda_j$, that is

$$r(x) - r(q_j) = O(\|x - q_j\|^2), \tag{6.23}$$

as $x \to q_j$.

Power iteration

For a real symmetric $n \times n$ matrix A, the eigenvectors $\{q_j\}_{j=1}^n$ form an orthonormal basis for $\mathbb{R}^n$ so that we can express any vector $v \in \mathbb{R}^n$ in terms of the eigenvectors,

$$v = \sum_{j=1}^{n} \alpha_j q_j, \tag{6.24}$$

with the coordinates $\alpha_j = (v, q_j)$. Further, we can express the map $v \mapsto Av$ in terms of the corresponding eigenvalues λ_j, as

$$Av = \sum_{j=1}^{n} \alpha_j Aq_j = \sum_{j=1}^{n} \alpha_j \lambda_j q_j, \tag{6.25}$$

and thus the map amounts to a scaling of each eigenvector q_j by λ_j. If iterated, this map gives

$$A^k v = \sum_{j=1}^{n} \alpha_j A^k q_j = \sum_{j=1}^{n} \alpha_j \lambda_j^k q_j, \tag{6.26}$$

so that each eigenvector $\lambda_j^k q_j$ of A^k converges to zero if $|\lambda_j| < 1$, or infinity if $|\lambda_j| > 1$.

Now assume that $\alpha_1 = (v, q_1) \neq 0$, and that the eigenvalues of A are ordered such that

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|, \tag{6.27}$$

where we say that λ_1 is the *dominant eigenvalue*, and q_1 the *dominant eigenvector*. Thus $|\lambda_j/\lambda_1| < 1$ for all j, which implies that

$$(\lambda_j/\lambda_1)^k \to 0, \tag{6.28}$$

as $k \to \infty$. We can write

$$A^k v = \lambda_1^k(\alpha_1 q_1 + \sum_{j=2}^{n} \alpha_j(\lambda_j/\lambda_1)^k q_j), \tag{6.29}$$

and thus the approximation

$$A^k v \approx \lambda_1^k \alpha_1 q_1, \tag{6.30}$$

improves as k increases. That is, $A^k v$ approaches a multiple of the eigenvector q_j, which can be obtained by normalizating, so that

$$v^{(k)} \equiv A^k v / \|A^k v\| \approx q_j, \tag{6.31}$$

from which an approximation of the corresponding eigenvalue can be obtained by the Rayleigh quotient, which leads us to *power iteration*.

Algorithm 5: Power iteration

$v^{(0)}$ such that $\|v^{(0)}\| = 1$
for $k = 1, 2, ...$ **do**
  $w = Av^{(k-1)}$ ▷ `apply` A
  $v^{(k)} = w/\|w\|$ ▷ `normalize`
  $\lambda^{(k)} = (v^{(k)})^T A v^{(k)}$ ▷ `Raylegh quotient`
end

Inverse iteration

The convergence of the Power iteration to the dominant eigenvector is linear by a constant factor $|\lambda_2/\lambda_1|$, whereas the convergence to the dominant

eigenvalue is quadratic in the same factor, due to the convergence of the Rayleigh quotient.

Efficiency of the algorithm thus depends on the size of the factor $|\lambda_2/\lambda_1|$. The idea of *inverse iteration*, is to apply power iteration to the matrix $(A-\mu I)^{-1}$, with eigenvalues $\{(\lambda_j-\mu)^{-1}\}$ and with the same eigenvectors as A, since

$$Av = \lambda v \Leftrightarrow (A-\mu I)v = (\lambda-\mu)v \Leftrightarrow (\lambda-\mu)^{-1}v = (A-\mu I)^{-1}v. \quad (6.32)$$

With μ an approximation of λ_j, the eigenvalue $(\lambda_j-\mu)^{-1}$ can be expected to be dominant and much larger than the other eigenvalues, which results in an accelerated convergence of power iteration.

Rayleigh quotient iteration is inverse iteration where μ is updated to the eigenvalue approximation of the previous step of the iteration, and the convergence to an eigenvalue/eigenvector pair is cubic.

Algorithm 6: Rayleigh quotient iteration

$v^{(0)}$ such that $\|v^{(0)}\| = 1$
$\lambda^{(0)} = (v^{(0)})^T A v^{(0)}$
for $k = 1, 2, \ldots$ **do**
  Solve $(A-\lambda^{(k-1)})w = v^{(k-1)}$ for w ▷ `apply` $(A-\lambda^{(k-1)}I)^{-1}$
  $v^{(k)} = w/\|w\|$ ▷ `normalize`
  $\lambda^{(k)} = (v^{(k)})^T A v^{(k)}$ ▷ `Raylegh quotient`
end

The QR algorithm as a power iteration

We now revisit the QR algorithm. Let $Q^{(k)}$ and $R^{(k)}$ be the matrices generated from the (unshifted) QR algorithm, then the matrix products

$$\underline{Q}^{(k)} = Q^{(1)}Q^{(2)}\cdots Q^{(k)}, \quad (6.33)$$

and

$$\underline{R}^{(k)} = R^{(k)}R^{(k-1)}\cdots R^{(1)}, \quad (6.34)$$

correspond to a QR factorization of the kth power of A,

$$A^k = \underline{Q}^{(k)}\underline{R}^{(k)}, \quad (6.35)$$

which can be proven by induction.

That is, the QR algorithm constructs successive orthonormal bases for the powers A^k, thus functioning like a power iteration that simultaneously iterates on the whole set of approximate eigenvectors.

Further, the diagonal elements of the kth iterate $A^{(k)}$ are the Rayleigh quotients of A corresponding to the column vectors of $\underline{Q}^{(k)}$,

$$A^{(k)} = (\underline{Q}^{(k)})^T A \underline{Q}^{(k)}, \tag{6.36}$$

and thus the diagonal elements of $A^{(k)}$ converges (quadratically) to the eigenvalues of A.

With the accelerations (i)-(iii) the QR algorithm exhibit cubic converge rate in both eigenvalues and eigenvectors.

6.3 Exercises

Problem 23. *Prove that the eigenspace E_λ, with λ an eigenvalue of the matrix A, is invariant under A.*

Chapter 7

Iterative methods for large sparse systems

In this chapter we revisit the problem of solving linear systems of equations, but now in the context of large sparse systems. The price to pay for the direct methods based on matrix factorization is that the factors of a sparse matrix may not be sparse, so that for large sparse systems the memory cost make direct methods too expensive, in memory and in execution time.

Instead we introduce iterative methods, for which matrix sparsity is exploited to develop fast algorithms with a low memory footprint.

7.1 Sparse matrix algebra

Large sparse matrices

We say that the matrix $A \in \mathbb{R}^n$ is large if n is large, and that A is *sparse* if most of the elements are zero. If a matrix is not sparse, we say that the matrix is *dense*. Whereas for a dense matrix the number of nonzero elements is $\mathcal{O}(n^2)$, for a sparse matrix it is only $\mathcal{O}(n)$, which has obvious implications for the memory footprint and efficiency for algorithms that exploit the sparsity of a matrix.

A diagonal matrix is a sparse matrix $A = (a_{ij})$, for which $a_{ij} = 0$ for all $i \neq j$, and a diagonal matrix can be generalized to a *banded* matrix, for which there exists a number p, the *bandwidth*, such that $a_{ij} = 0$ for all $i < j - p$ or $i > j + p$. For example, a *tridiagonal* matrix A is a banded

matrix with $p = 1$,

$$A = \begin{bmatrix} \mathbf{x} & \mathbf{x} & 0 & 0 & 0 & 0 \\ \mathbf{x} & \mathbf{x} & \mathbf{x} & 0 & 0 & 0 \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} & 0 & 0 \\ 0 & 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} & 0 \\ 0 & 0 & 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & 0 & 0 & 0 & \mathbf{x} & \mathbf{x} \end{bmatrix}, \tag{7.1}$$

where $\mathbf{x}$ represents a nonzero element.

Compressed row storage

The compressed row storage (CRS) format is a data structure for efficient represention of a sparse matrix by three arrays, containing the nonzero values, the respective column indices, and the extents of the rows.

For example, the following sparse matrix

$$A = \begin{bmatrix} 3 & 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 2 \\ 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 2 & 4 & 0 \\ 0 & 4 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 3 \end{bmatrix}, \tag{7.2}$$

is represented as

$$\begin{aligned} val &= [3 \;\; 2 \;\; 2 \;\; 1 \;\; 2 \;\; 2 \;\; 1 \;\; 3 \;\; 2 \;\; 4 \;\; 4 \;\; 1 \;\; 2 \;\; 3] \\ col\_idx &= [1 \;\; 2 \;\; 2 \;\; 3 \;\; 6 \;\; 2 \;\; 3 \;\; 3 \;\; 4 \;\; 5 \;\; 2 \;\; 5 \;\; 5 \;\; 6] \\ row\_ptr &= [1 \;\; 3 \;\; 6 \;\; 8 \;\; 11 \;\; 13] \end{aligned}$$

where *val* contains the nonzero matrix elements, *col_idx* their column indices, and *row_ptr* the indices in the other two arrays corresponding to the start of each row.

Sparse matrix-vector product

For a sparse matrix A, algorithms can be constructed for efficient matrix-vector multiplication $b = Ax$, exploiting the sparsity of A by avoiding multiplications by the zero elements of A.

For example, the CRS data structure implies an efficient algorithm for *sparse matrix-vector* multiplication, for which both the memory footprint and the number of floating point operations are of the order $\mathcal{O}(n)$, rather than $\mathcal{O}(n^2)$ as in the case of a dense matrix.

Algorithm 7: Sparse matrix-vector multiplication

```
for i = 1 : n do
    b_i = 0
    for j = row_ptr(i) : row_ptr(i + 1) − 1 do
        b_i = b_i + val(j)x(col_idx(j))
    end
end
```

7.2 Iterative methods

Iterative methods for large sparse linear systems

For a given nonsingular matrix $A \in \mathbb{R}^{n\times n}$ and vector $b \in \mathbb{R}^n$, we consider the problem of finding a vector $x \in \mathbb{R}^n$, such that

$$Ax = b, \tag{7.3}$$

where n is large, and the matrix A is sparse.

Now, in contrast to direct methods, we do not seek to construct the exact solution $x = A^{-1}b$ by matrix factorization, which is too expensive. Instead we develop *iterative methods* based on multiplication by a (sparse) iteration matrix, which generates a sequence of approximations $\{x^{(k)}\}_{k\geq 0}$ that converges towards x, with the *error* at iteration k given as

$$e^{(k)} = x - x^{(k)}. \tag{7.4}$$

Error estimation and stopping criterion

Since the exact solution is unknown, the error is not directly computable, but can be expressed in terms of a computable *residual*,

$$r^{(k)} = b - Ax^{(k)} = Ax - Ax^{(k)} = Ae^{(k)}. \tag{7.5}$$

The relative error can be estimated in terms of the relative residual and the *condition number* of A with respect to the norm $\|\cdot\|$, defined as

$$\kappa(A) = \|A\|\|A^{-1}\|. \tag{7.6}$$

Theorem 11 (Error estimate)**.** *For $\{x^{(k)}\}_{k\geq 0}$ a sequence of approximate solutions to the linear system of equations $Ax = b$, the relative error can be estimated as*

$$\frac{\|e^{(k)}\|}{\|e^{(0)}\|} \leq \kappa(A)\frac{\|r^{(k)}\|}{\|r^{(0)}\|}. \tag{7.7}$$

Proof. By (7.5), we have that

$$\|e^{(k)}\| = \|A^{-1}r^{(k)}\| \leq \|A^{-1}\|\|r^{(k)}\|, \tag{7.8}$$

and similarly

$$\|r^{(0)}\| = \|Ae^{(0)}\| \leq \|A\|\|e^{(0)}\|, \tag{7.9}$$

from which the result follows by the definition of the condition number. □

The error estimate (7.7) may be used as a *stopping criterion* for the iterative algorithm, since we know that the relative error is bounded by the computable residual. That is, we terminate the iterative algorithm if the following condition is satisfied,

$$\frac{\|r^{(k)}\|}{\|r^{(0)}\|} < TOL, \tag{7.10}$$

with $TOL > 0$ the chosen tolerance.

Although, to use the relative error with respect to the initial approximation is problematic, since the choice of $x^{(0)}$ may be arbitrary, without significance for the problem at hand. It is often more suitable to formulate a stopping criterion based on the following condition,

$$\frac{\|r^{(k)}\|}{\|b\|} < TOL, \tag{7.11}$$

corresponding to $x^{(0)} = 0$.

Convergence of iterative methods

The iterative methods that we will develop are all based on the idea of *fixed point iteration*,

$$x^{(k+1)} = g(x^{(k)}), \tag{7.12}$$

where the map $x \mapsto g(x)$ may be a linear transformation in the form of a matrix, or a general nonlinear function. By *Banach fixed point theorem*, if the map satisfies certain stability conditions, the fixed point iteration (7.12) generates a *Cauchy sequence*, that is, a sequence for which the approximations $x^{(k)}$ become closer and closer as k increases.

A Cauchy sequence in a normed vector space X is defined as a sequence $\{x^{(k)}\}_{k=1}^{\infty}$, for which

$$\lim_{n\to\infty} \|x^{(m)} - x^{(n)}\| = 0, \tag{7.13}$$

with $m > n$. If all Cauchy sequences in X converges to an element $x \in X$, that is,

$$\lim_{n\to\infty} \|x - x^{(n)}\| = 0, \quad x \in X, \tag{7.14}$$

we refer to X as a *Banach space*.

The vector spaces $\mathbb{R}^n$ and $\mathbb{C}^n$ are both Banach spaces, whereas, for example, the vector space of rational numbers $\mathbb{Q}$ is not. To see this, recall that $\sqrt{2}$ is a real number that is the limit of a Cauchy sequence of rational numbers constructed by iterated bisection of the interval $[1, 2]$.

Further, a Banach space that is also an inner product space is referred to as a *Hilbert space*, which is central for the theory of differential equations.

Rate of convergence

We are not only interested in *if* an iterative method converges, but also *how fast*, that is the *rate of convergence*. We say that a sequence of approximate solutions $\{x^{(k)}\}_{k=1}^{\infty}$ converges with *order* p to the exact solution x, if

$$\lim_{k\to\infty} \frac{|x - x^{(k+1)}|}{|x - x^{(k)}|^p} = C, \quad C > 0, \tag{7.15}$$

where $p = 1$ corresponds to a *linear order of convergence*, and $p = 2$ a *quadratic order of convergence*.

We can approximate the rate of convergence by extrapolation,

$$p \approx \frac{\log \dfrac{|x^{(k+1)} - x^{(k)}|}{|x^{(k)} - x^{(k-1)}|}}{\log \dfrac{|x^{(k)} - x^{(k-1)}|}{|x^{(k-1)} - x^{(k-2)}|}}, \tag{7.16}$$

which is useful in practice when the exact solution is not available.

7.3 Stationary iterative methods

Stationary iterative methods

Stationary iterative methods are formulated as a linear *fixed point iteration* of the form

$$x^{(k+1)} = Mx^{(k)} + c, \tag{7.17}$$

with $M \in \mathbb{R}^{n\times n}$ the *iteration matrix*, $\{x^{(k)}\}_{k\geq 0} \subset \mathbb{R}^n$ a sequence of approximations, and $c \in \mathbb{R}^n$ a vector.

Theorem 12 (Banach fixed point theorem for matrices)**.** *If $\|M\| < 1$, the fixed point iteration (7.17) converges to the solution of the equation $x = Mx + c$.*

Proof. For any $k > 1$, we have that

$$\begin{aligned} \|x^{(k+1)} - x^{(k)}\| &= \|Mx^{(k)} - Mx^{(k-1)}\| = \|M(x^{(k)} - x^{(k-1)})\| \\ &\leq \|M\|\|x^{(k)} - x^{(k-1)}\| \leq \|M\|^k \|x^{(1)} - x^{(0)}\|. \end{aligned}$$

Further, for $m > n$,

$$\begin{aligned} \|x^{(m)} - x^{(n)}\| &= \|x^{(m)} - x^{(m-1)}\| + ... + \|x^{(n+1)} - x^{(n)}\| \\ &\leq \left(\|M\|^{m-1} + ... + \|M\|^n\right) \|x^{(1)} - x^{(0)}\|, \end{aligned}$$

so that with $\|M\| < 1$. We thus have that

$$\lim_{n\to\infty} \|x^{(m)} - x^{(n)}\| = 0, \tag{7.18}$$

that is $\{x^{(n)}\}_{n=1}^{\infty}$ is a Cauchy sequence, and since the vector space $\mathbb{R}^n$ is complete, all Cauchy sequences converge, so there exists an $x \in \mathbb{R}^n$ such that

$$x = \lim_{n\to\infty} x^{(n)}. \tag{7.19}$$

By taking the limit of both sides of (7.17) we find that x satisfies the equation $x = Mx + c$. □

An equivalent condition for convergence of (7.17) is that the spectral radius $\rho(M) < 1$. In particular, for a real symmetric matrix A, the spectral radius is identical to the induced 2-norm, that is $\rho(A) = \|A\|$.

Richardson iteration

The linear system $Ax = b$ can be formulated as a fixed point iteration through the *Richardson iteration*, with an iteration matrix $M = I - A$,

$$x^{(k+1)} = (I - A)x^{(k)} + b, \tag{7.20}$$

which will converge if $\|I - A\| < 1$, or $\rho(A) < 1$. We note that for an initial approximation $x^{(0)} = 0$, we obtain for $k = 0$,

$$x^{(1)} = (I - A)x^{(0)} + b = b$$

for $k = 1$,

$$x^{(2)} = (I - A)x^{(1)} + b = (I - A)b + b = 2b - Ab,$$

for $k = 2$,

$$x^{(3)} = (I - A)x^{(2)} + b = (I - A)(2b - Ab) + b = 3b - 3Ab + A^2b,$$

and more generally, that the iterate $x^{(k)}$ is a linear combination of powers of the matrix A acting on b, that is

$$x^{(k)} = \sum_{i=0}^{k-1} \alpha_i A^i b, \tag{7.21}$$

with $\alpha_i \in \mathbb{R}$.

Preconditioned Richardson iteration

To improve convergence of Richardson iteration we can *precondition* the system $Ax = b$ by multiplication of both sides of the equation by a matrix B, so that we get the new system

$$BAx = Bb, \tag{7.22}$$

for which Richardson iteration will converge if $\|I - BA\| < 1$, or equivalently $\rho(BA) < 1$, and we then refer to B as an *approximate inverse* of A. The preconditioned Richardson iteration takes the form

$$x^{(k+1)} = (I - BA)x^{(k)} + Bb, \tag{7.23}$$

and the preconditioned residual $Bb - BAx^{(k)}$ is used as basis for a stopping criterion.

Iterative methods based on matrix splitting

An alternative to Richardson iteration is *matrix splitting*, where stationary iterative methods are formulated based on splitting the matrix into a sum

$$A = A_1 + A_2, \tag{7.24}$$

where A_1 is chosen as a nonsingular matrix easy to invert, such as a diagonal matrix D, a lower triangular matrix L or upper triangular matrix U.

Jacobi iteration

Jacobi iteration is based on the splitting

$$A_1 = D, \quad A_2 = R = A - D, \tag{7.25}$$

which gives the iteration matrix $M_J = -D^{-1}R$ and $c = D^{-1}b$, or in terms of the elements of $A = (a_{ij})$,

$$x_i^{(k+1)} = a_{ii}^{-1}\left(b - \sum_{j \neq i} a_{ij}x_j^{(k)}\right), \tag{7.26}$$

where the diagonal matrix D is trivial to invert. To use Jacobi iteration as a preconditioner, we choose $B = D^{-1}$.

Gauss-Seidel iteration

Gauss-Seidel iteration is based on the splitting

$$A_1 = L, \quad A_2 = R = A - L, \tag{7.27}$$

which gives the iteration matrix $M_{GS} = -L^{-1}R$ and $c = L^{-1}b$, or

$$x_i^{(k+1)} = a_{ii}^{-1}\left(b - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\right), \tag{7.28}$$

where the lower triangular matrix L is inverted by forward substitution. Gauss-Seidel iteration as a preconditioner leads to the choice of $B = L^{-1}$, where the inversion corresponds to a forward substitution.

7.4 Krylov methods

Krylov subspace

A *Krylov method* is an iterative method for the solution of the system $Ax = b$ based on, for each iteration, finding an approximation $x^{(k)} \approx x = A^{-1}b$ in a *Krylov subspace* $\mathcal{K}_k$, spanned by the vectors $b, Ab, ..., A^{k-1}b$, that is

$$\mathcal{K}_k = \langle b, Ab, ..., A^{k-1}b \rangle. \tag{7.29}$$

The basis for Krylov methods is that, by the *Cayley-Hamilton theorem*, the inverse of a matrix A^{-1} is a linear combination of its powers A^k, which is also expressed in (7.21).

GMRES

The idea of *GMRES* (generalized minimal residuals) is that, at each step k of the iteration, find the vector $x^{(k)} \in \mathcal{K}_k$ that minimizes the norm of the residual $r^{(k)} = b - Ax^{(k)}$, which corresponds to the least squares problem

$$\min_{x^{(k)} \in \mathcal{K}_k} \|b - Ax^{(k)}\|. \tag{7.30}$$

But instead of expressing the approximation $x^{(k)}$ as a linear combination of the Krylov vectors $b, Ab, ..., A^{k-1}b$, which leads to an unstable algorithm, we construct an orthonormal basis $\{q_j\}_{j=1}^{k}$ for $\mathcal{K}_k$, such that

$$\mathcal{K}_k = \langle q_1, q_2, ..., q_k \rangle, \tag{7.31}$$

with Q_k the $n \times k$ matrix with the basis vectors q_j as columns.

Thus we can express the approximation as $x^{(k)} = Q_k y$, with $y \in \mathbb{R}^k$ a vector with the coordinates of $x^{(k)}$, so that the least squares problem take the form

$$\min_{y \in \mathbb{R}^k} \|b - AQ_k y\|. \tag{7.32}$$

Algorithm 8: Arnoldi iteration

```
q_1 = b/‖b‖
for k = 1, 2, 3, ... do
    v = Aq_k
    for j = 1 : k do
        h_jk = q_j^T v
        v = v − h_jk q_j
    end
    h_{n+1n} = ‖v‖
    q_{n+1} = v/h_{n+1n}
end
```

The *Arnoldi iteration* is just the modified Gram-Schmidt iteration (Algorithm 1) that constructs a partial similarity transformation of A into an *Hessenberg matrix* $\tilde{H}_k \in \mathbb{R}^{k+1 \times k}$,

$$AQ_k = Q_{k+1}\tilde{H}_k, \tag{7.33}$$

that is

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \left[\begin{array}{c|c|c} & & \\ q_1 & \cdots & q_k \\ & & \end{array}\right] = \left[\begin{array}{c|c|c} & & \\ q_1 & \cdots & q_{k+1} \\ & & \end{array}\right] \begin{bmatrix} h_{11} & \cdots & h_{1k} \\ h_{21} & \cdots & \\ & \ddots & \vdots \\ & & h_{k+1k} \end{bmatrix}.$$

Multiplication of (7.32) by Q_{k+1}^T does not change the norm, so that the least squares problem takes the form,

$$\min_{y \in \mathbb{R}^k} \|Q_{k+1}^T b - \tilde{H}_k y\|, \tag{7.34}$$

where we note that since $q_1 = b/\|b\|$, we have that $Q_{k+1}^T b = \|b\| e_1$ with $e_1 = (1, 0, ..., 0)^T$ the first vector in the standard basis in $\mathbb{R}^{k+1}$, so that we can write (7.34) as

$$\min_{y \in \mathbb{R}^k} \|\|b\| e_1 - \tilde{H}_k y\|, \tag{7.35}$$

which is a $(k+1) \times k$ least squares problem that we solve for $y \in \mathbb{R}^k$ at each iteration k, to get $x^{(k)} = Q_k y$.

Algorithm 9: GMRES

$q_1 = b/\|b\|$
while $\|r^{(k)}\|/\|r^{(0)}\| \geq TOL$ **do**
  Arnoldi iteration step $k \rightarrow Q_k, \tilde{H}_k$ ▷ `orthogonalize`
  $\min_{y \in \mathbb{R}^k} \|\|b\| e_1 - \tilde{H}_k y\|$ ▷ `least squares problem`
  $x^{(k)} = Q_k y$ ▷ `construct solution`
end

Conjugate Gradient method

For a symmetric positive definite matrix A, we can define the *A-norm* of a vector $x \in \mathbb{R}^n$, as

$$\|x\|_A = (x, Ax)^{1/2}, \tag{7.36}$$

with $(\cdot, \cdot)$ the l_2-norm. The *Conjugate Gradient method* (CG) is based on minimization of the error $e^{(k)} = x - x^{(k)}$ in the A-norm, or equivalently, by (7.5), minimization of the residual $r^{(k)} = b - Ax^{(k)}$ in the A^{-1}-norm,

$$\|e^{(k)}\|_A = (e^{(k)}, Ae^{(k)})^{1/2} = (e^{(k)}, r^{(k)})^{1/2} = (A^{-1} r^{(k)}, r^{(k)})^{1/2} = \|r^{(k)}\|_{A^{-1}},$$

to compare to GMRES where the residual is minimized in the l_2-norm.

Further, to solve the minimization problem in CG we do not solve a least squares problem over the Krylov subspace $\mathcal{K}_k$, but instead we iteratively construct a *search direction* $p^{(k)}$ and a *step length* $\alpha^{(k)}$ to find the new approximate solution $x^{(k)}$ from the previous iterate $x^{(k-1)}$. In particular, this means that we do not have to store the full Krylov basis.

Algorithm 10: Conjugate Gradient method

$x^{(0)} = 0$, $r^{(0)} = b$, $p^{(k)} = r^{(0)}$
while $\|r^{(k)}\|/\|r^{(0)}\| \geq TOL$ **do**
  $\alpha^{(k)} = \|r^{(k-1)}\|/\|p^{(k-1)}\|_A$ ▷ `step length`
  $x^{(k)} = x^{(k-1)} + \alpha^{(k)} p^{(k-1)}$ ▷ `approximate solution`
  $r^{(k)} = r^{(k-1)} - \alpha^{(k)} A p^{(k-1)}$ ▷ `residual`
  $\beta^{(k)} = \|r^{(k)}\|/\|r^{(k-1)}\|$ ▷ `improvement`
  $p^{(k)} = r^{(k)} + \beta^{(k)} p^{(k-1)}$ ▷ `search direction`
end

The key to the success of the CG method is that the residuals are mutually orthogonal,

$$(r^{(k)}, r^{(j)}) = 0, \quad \forall j < k, \tag{7.37}$$

and that the search directions are *A-conjugate*,

$$(p^{(k)}, p^{(j)})_A = 0, \quad \forall j < k, \tag{7.38}$$

where $(\cdot,\cdot)_A$ is the *weighted inner product*, defined for symmetric positive definite matrices as

$$(x, y)_A = x^T A y = (Ay)^T x = y^T A^T x = y^T A x = (y, x)_A, \tag{7.39}$$

where we note that $(\cdot,\cdot)_A$ induces the *A-norm*,

$$\|x\|_A = (x, x)_A^{1/2}, \tag{7.40}$$

which is also referred to as the *energy norm* for the equation $Ax = b$.

Theorem 13 (CG characteristics)**.** *For the CG method applied to the equation $Ax = b$, with A an $n \times n$ symmetric positive definite matrix, the orthogonality relations (7.37) and (7.38) are true, and*

$$\begin{aligned} \mathcal{K}_k &= \langle b, Ab, ..., A^{k-1}b\rangle = \langle x^{(1)}, x^{(2)}, ..., x^{(k)}\rangle \\ &= \langle p^{(0)}, p^{(1)}, ..., p^{(k-1)}\rangle = \langle r^{(0)}, r^{(1)}, ..., r^{(k-1)}\rangle, \end{aligned}$$

with the approximate solutions $x^{(k)}$, search directions $p^{(k)}$ and residuals $r^{(k)}$ constructed from Algorithm 10. Further, $x^{(k)}$ is the unique point in $\mathcal{K}_k$ that minimizes $\|e^{(k)}\|_A$, and the convergence is monotonic, that is

$$\|e^{(k)}\|_A \leq \|e^{(k-1)}\|_A, \tag{7.41}$$

with $e^{(k)} = 0$ for some $k \leq n$.

Chapter 8

Nonlinear algebraic equations

We now turn to systems of nonlinear algebraic equations that cannot be expressed as matrix equations. The fundamental idea to solve such equations is fixed point iteration, which we have met previously for linear systems of equations, in the context of stationary iterative methods. Convergence of fixed point iteration depends linearly on the degree to which the iteration function is continuous.

In case we have access to a sufficiently good initial guess, we can formulate Newton's method which exhibits quadratic rate of convergence.

8.1 Continuous functions

A continuous function can be roughly characterized as a function for which small changes in input results in small changes in output. More formally, a function $f : I \to \mathbb{R}$, is said to be (uniformly) *continuous* on the interval $I = [a, b]$, if for each $\epsilon > 0$ we can find a $\delta > 0$, such that

$$|x - y| < \delta \Rightarrow |f(x) - f(y)| < \epsilon, \quad \forall x, y \in I, \tag{8.1}$$

and *Lipschitz continuous* on the interval I, if there exists a real number $L_f > 0$, the *Lipschitz constant* of f, such that

$$|f(x) - f(y)| \leq L_f |x - y|, \quad \forall x, y \in I. \tag{8.2}$$

The vector space of real valued continuous functions on the interval I is denoted by $\mathcal{C}^0(I)$, or $\mathcal{C}(I)$, which is closed under the basic operations of pointwise addition and scalar multiplication, defined by,

$$(f + g)(x) = f(x) + g(x), \qquad \forall x \in I, \tag{8.3}$$

$$(\alpha f)(x) = \alpha f(x), \qquad \forall x \in I, \tag{8.4}$$

for $f, g \in \mathcal{C}(I)$ and $\alpha \in \mathbb{R}$. Similarly, we let $Lip(I)$ denote the vector space of Lipschitz continuous functions together with the same basic operations, and we note that any Lipschitz continuous function is also continuous, that is $Lip(I) \subset \mathcal{C}(I)$.

Further, we denote the vector space of continuous functions with also continuous derivatives up to the order k by $\mathcal{C}^k(I)$, with $\mathcal{C}^\infty(I)$ the vector space of continuous functions with continuous derivatives of arbitrary order.

Local approximation of continuous functions

By *Taylor's theorem* we can construct a local approximation of a function $f \in \mathcal{C}^k(I)$ near any point $y \in I$, in terms of the function and its first k derivatives evaluated at the point y. For example, we can approximate $f(x)$ by a linear function,

$$f(x) \approx f(y) + f'(y)(x - y), \tag{8.5}$$

corresponding to the *tangent line* of the function at $x = y$, with the approximation error reduced quadratically when decreasing the distance $|x - y|$.

Theorem 14 (Tayor's theorem)**.** *For $f \in \mathcal{C}^2(I)$, we have that*

$$f(x) = f(y) + f'(y)(x - y) + \frac{1}{2} f''(\xi)(x - y)^2, \tag{8.6}$$

for $y \in I$ and $\xi \in [x, y]$.

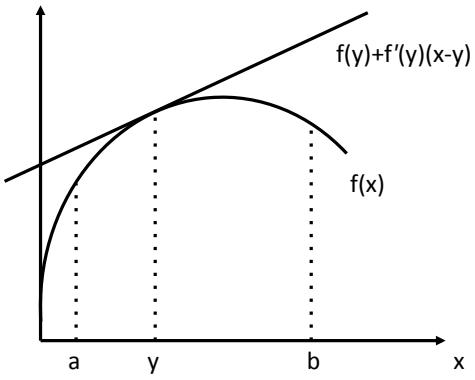

Figure 8.1: The tangent line $f(y) + f'(y)(x - y)$ at $y \in [a, b]$.

8.2 Nonlinear scalar equations

Fixed point iteration

For a nonlinear function $f : \mathbb{R} \to \mathbb{R}$, we seek a solution x to the equation

$$f(x) = 0, \tag{8.7}$$

for which we can formulate a fixed point iteration $x^{(k+1)} = g(x^{(k)})$, as

$$x^{(k+1)} = g(x^{(k)}) = x^{(k)} + \alpha f(x^{(k)}), \tag{8.8}$$

where α is a parameter to be chosen. The fixed point iteration (8.8) converges to a unique fixed point $x = g(x)$ that satisfies equation (8.7), under the condition that the function $g \in Lip(\mathbb{R})$ with $L_g < 1$, which we can prove by similar arguments as in the case of a linear system of equations (7.17).

For any $k > 1$, we have that

$$|x^{(k+1)} - x^{(k)}| = |g(x^{(k)}) - g(x^{(k-1)})| \leq L_g|x^{(k)} - x^{(k-1)}| \leq L_g^k|x^{(1)} - x^{(0)}|,$$

so that for $m > n$,

$$\begin{aligned} |x^{(m)} - x^{(n)}| &= |x^{(m)} - x^{(m-1)}| + ... + |x^{(n+1)} - x^{(n)}| \\ &\leq (L_g^{m-1} + ... + L_g^n)\,|x^{(1)} - x^{(0)}|, \end{aligned}$$

by the triangle inequality, and with $L_g < 1$ we have that $\{x^{(n)}\}_{n=1}^{\infty}$ is a Cauchy sequence,

$$\lim_{n\to\infty} |x^{(m)} - x^{(n)}| = 0, \tag{8.9}$$

which implies that there exists an $x \in \mathbb{R}$, such that

$$\lim_{n\to\infty} |x - x^{(n)}| = 0, \tag{8.10}$$

since $\mathbb{R}$ is a Banach space.

Uniqueness of x follows from assuming that there exists another solution $y \in \mathbb{R}$ such that $y = g(y)$, for which we have that

$$|x - y| = |g(x) - g(y)| \leq L_g|x - y| \Rightarrow (1 - L_g)|x - y| \leq 0 \Rightarrow |x - y| = 0,$$

and thus $x = y$ is the unique solution to the equation $x = g(x)$.

Newton's method

The analysis above suggests that the fixed point iteration (8.8) converges linearly, since

$$|x - x^{(k+1)}| = |g(x) - g(x^{(k)})| \leq L_g |x - x^{(k)}|, \tag{8.11}$$

so that for the error $e^{(k)} = x - x^{(k)}$ we have that $|e^{(k+1)}| \leq L_g |e^{(k)}|$.

Although, for the choice $\alpha = -f'(x^{(k)})^{-1}$, which we refer to as *Newton's method*, the fixed point iteration (8.8) exhibits quadratic convergence. The geometric interpretation of Newton's method is that $x^{(k+1)}$ is determined from the tangent line of the function $f(x)$ at $x^{(k)}$.

Algorithm 11: Newton's method

Given an initial approximation $x^{(0)} \in \mathbb{R}$ and $f : \mathbb{R} \to \mathbb{R}$
while $|f(x^{(k)})| \geq TOL$ **do**
  Compute $f'(x^{(k)})$
  $x^{(k+1)} = x^{(k)} - f'(x^{(k)})^{-1} f(x^{(k)})$
end

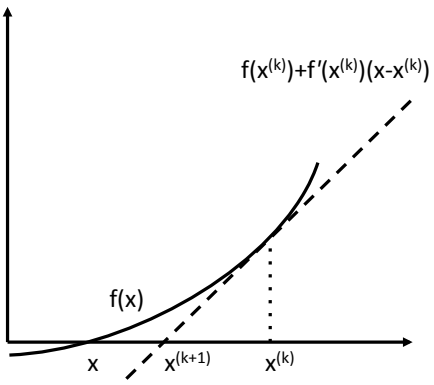

Figure 8.2: Geometric interpretation of Newtons method with the approximation $x^{(k+1)}$ obtained as the zero value of the tangent at $x^{(k)}$.

The quadratic convergence rate of Newton's method follows from *Taylor's theorem*, evaluated at $x^{(k)}$,

$$0 = f(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(\xi)(x - x^{(k)})^2, \tag{8.12}$$

with $\xi \in [x, x^{(k)}]$. We divide by $f'(x^{(k)})$ to get

$$x - (x^{(k)} - f'(x^{(k)})^{-1} f(x^{(k)})) = -\frac{1}{2} f'(x^{(k)})^{-1} f''(\xi)(x - x^{(k)})^2, \tag{8.13}$$

so that, with $x^{(k+1)} = x^{(k)} - f'(x^{(k)})^{-1} f(x^{(k)})$, we get

$$|e^{(k+1)}| = \frac{1}{2} |f'(x^{(k)})^{-1} f''(\xi)| \, |e^{(k)}|^2, \tag{8.14}$$

which displays the quadratic convergence of the sequence $x^{(k)}$ close to x.

8.3 Systems of nonlinear equations

Continuous functions and derivatives in $\mathbb{R}^n$

In the normed space $\mathbb{R}^n$, a function $f : \mathbb{R}^n \to \mathbb{R}^n$ is (uniformly) *continuous*, denoted $f \in \mathcal{C}(\mathbb{R}^n)$, if for each $\epsilon > 0$ we can find a $\delta > 0$, such that

$$\|x - y\| < \delta \Rightarrow \|f(x) - f(y)\| < \epsilon, \quad \forall x, y \in \mathbb{R}^n, \tag{8.15}$$

and *Lipschitz continuous*, denoted $f \in Lip(\mathbb{R}^n)$, if there exists a real number $L_f > 0$, such that,

$$\|f(x) - f(y)\| \leq L_f \|x - y\|, \quad \forall x, y \in \mathbb{R}^n. \tag{8.16}$$

We define the *partial derivative* as

$$\frac{\partial f_i}{\partial x_j} = \lim_{h \to 0} \frac{f_i(x_1, ..., x_j + h, ..., x_n) - f_i(x_1, ..., x_j, ..., x_n)}{h}, \tag{8.17}$$

for i the index of the function component of $f = (f_1, ..., f_n)$, and j the index of the coordinate $x = (x_1, ..., x_n)$. The *Jacobian matrix* $f'(x) \in \mathbb{R}^{n \times n}$, at $x \in \mathbb{R}^n$, is defined as

$$f'(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(x) & \cdots & \frac{\partial f_n}{\partial x_n}(x) \end{bmatrix} = \begin{bmatrix} (\nabla f_1(x))^T \\ \vdots \\ (\nabla f_n(x))^T \end{bmatrix}, \tag{8.18}$$

with the *gradient* $\nabla f_i(x) \in \mathbb{R}^n$, defined by

$$\nabla f_i(x) = \left(\frac{\partial f_i}{\partial x_1}(x), ..., \frac{\partial f_i}{\partial x_n}(x) \right)^T, \tag{8.19}$$

for $i = 1, ..., n$.

The vector space of continuous functions $f : \mathbb{R}^n \to \mathbb{R}^n$ with also continuous partial derivatives up to the order k is denoted by $\mathcal{C}^k(I)$, with $\mathcal{C}^\infty(I)$ the vector space of continuous functions with continuous derivatives of arbitrary order.

Fixed point iteration for nonlinear systems

Now consider a system of nonlinear equations: find $x \in \mathbb{R}^n$, such that

$$f(x) = 0, \tag{8.20}$$

with $f : \mathbb{R}^n \to \mathbb{R}^n$, for which we can formulate a fixed point iteration $x^{(k+1)} = g(x^{(k)})$, with $g : \mathbb{R}^n \to \mathbb{R}^n$, just as in the case of the scalar problem.

Algorithm 12: Fixed point iteration for solving the system $f(x) = 0$

Given initial approximation $x^{(0)} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$
while $\|f(x^{(k)})\| \geq TOL$ **do**
  $x^{(k+1)} = x^{(k)} + \alpha f(x^{(k)})$
end

Existence of a unique solution to the fixed point iteration follows by Banach fixed point theorem.

Theorem 15 (Banach fixed point theorem in $\mathbb{R}^n$)**.** *The fixed point iteration of Algorithm 12 converges to a unique solution if $L_g < 1$, with L_g the Lipschitz constant of the function $g(x) = x + \alpha f(x)$.*

Proof. For $k > 1$ we have that

$$\begin{aligned} \|x^{(k+1)} - x^{(k)}\| &= \|x^{(k)} - x^{(k-1)} + \alpha(f(x^{(k)}) - f(x^{(k-1)}))\| \\ &= \|g(x^{(k)}) - g(x^{(k-1)})\| \leq L_g^k \|x^{(1)} - x^{(0)}\|, \end{aligned}$$

and for $m > n$,

$$\begin{aligned} \|x^{(m)} - x^{(n)}\| &= \|x^{(m)} - x^{(m-1)}\| + ... + \|x^{(n+1)} - x^{(n)}\| \\ &\leq (L_g^{m-1} + ... + L_g^n)\|x^{(1)} - x^{(0)}\|. \end{aligned}$$

Since $L_g < 1$, $\{x^{(n)}\}_{n=1}^{\infty}$ is a Cauchy sequence, which implies that there exists an $x \in \mathbb{R}^n$ such that

$$\lim_{n\to\infty} \|x - x^{(n)}\| = 0,$$

since $\mathbb{R}^n$ is a Banach space. Uniqueness follows from assuming that there exists another solution $y \in \mathbb{R}^n$ such that $f(y) = 0$, so that

$$\|x - y\| = \|g(x) - g(y)\| \leq L_g \|x - y\| \Rightarrow (1 - L_g)\|x - y\| \leq 0,$$

and thus $x = y$ is the unique solution to the equation $f(x) = 0$. □

Newton's method for nonlinear systems

Newton's method for a system in $\mathbb{R}^n$ is analogous to the method for scalar equations, but with the inverse of the derivative replaced by the inverse of the Jacobian matrix $(f'(x^{(k)}))^{-1}$. The inverse is not constructed explicitly, instead we solve a linear system of equations $Ax = b$, with $A = f'(x^{(k)})$, $x = \Delta x^{(k+1)} = x^{(k+1)} - x^{(k)}$ the increment, and $b = f(x^{(k)})$ the residual.

Depending on how the Jacobian is computed, and the linear system solved, we get different versions of Newton's method. If the system is large and sparse, we use iterative methods for solving the linear system $Ax = b$, and if the Jacobian $f'(\cdot)$ is not directly available we use an approximation, obtained, for example, by a difference approximation based on $f(\cdot)$.

Algorithm 13: Newton's method for systems of nonlinear equations

Given initial approximation $x^{(0)} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$
while $\|f(x^{(k)})\| \geq TOL$ **do**
  Compute $f'(x^{(k)})$ ▷ `compute Jacobian`
  $f'(x^{(k)})\Delta x^{(k+1)} = -f(x^{(k)})$ ▷ `solve for` $\Delta x^{(k+1)}$
  $x^{(k+1)} = x^{(k)} + \Delta x^{(k+1)}$ ▷ `update by` $\Delta x^{(k+1)}$
end

Quadratic convergence rate of Newton's method for systems follows from Taylor's formula in $\mathbb{R}^n$, which states that

$$f(x) - (f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)})) = \mathcal{O}(\|x - x^{(k)}\|^2).$$

For $f(x) = 0$, and assuming the Jacobian matrix $f'(x^{(k)})$ to be nonsingular, we have that,

$$x - (x^{(k)} - f'(x^{(k)})^{-1} f(x^{(k)})) = \mathcal{O}(\|x - x^{(k)}\|^2),$$

and with $x^{(k+1)} = x^{(k)} - f'(x^{(k)})^{-1} f(x^{(k)})$, we get that

$$\frac{\|e^{(k+1)}\|}{\|e^{(k)}\|^2} = \mathcal{O}(1), \tag{8.21}$$

for the error $e^{(k)} = x - x^{(k)}$. The quadratic convergence rate then follows for $x^{(k)}$ close to x.

Part IV

Differential equations

Chapter 9

Initial value problems

Differential equations are fundamental to model the laws of Nature, such as Newton's laws of motion, Einstein's general relativity, Schrödinger's equation of quantuum mechanics, and Maxwell's equations of electromagnetics.

The case of one single independent variable, we refer to as ordinary differential equations, whereas partial differential equations involve several independent variables. We first consider the initial value problem, an ordinary differential equation where the independent variable naturally represents time, for which we develop solution methods based on time-stepping algorithms.

9.1 The scalar initial value problem

We consider the following *ordinary differential equation* (ODE) for a scalar function $u : \mathbb{R}^+ \to \mathbb{R}$, with derivative $\dot{u} = du/dt$,

$$\begin{aligned} &\dot{u}(t) = f(u(t), t), \quad 0 < t \leq T, \\ &u(0) = u_0, \end{aligned} \tag{9.1}$$

which we refer to as a *scalar initial value problem*, defined on the interval $I = [0, T]$ by the function $f : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}$, and the *initial condition* $u(0) = u_0$.

Only in simple special cases can analytical solutions be found. Instead, in the general case, numerical methods must be used to compute approximate solutions to (9.1).

9.2 Time stepping methods

The variable $t \in [0, T]$ is often interpreted as time, and numerical methods to solve (9.1) can be based on the idea of *time-stepping*, where successive approximations $U(t_n)$ are computed on a partition $0 = t_0 < t_1 < ... < t_N = T$, starting from $U(t_0) = u_0$, with a suitable *interpolation* of $U(t)$ on each subinterval $I_n = (t_{n-1}, t_n)$ of length $k_n = t_n - t_{n-1}$.

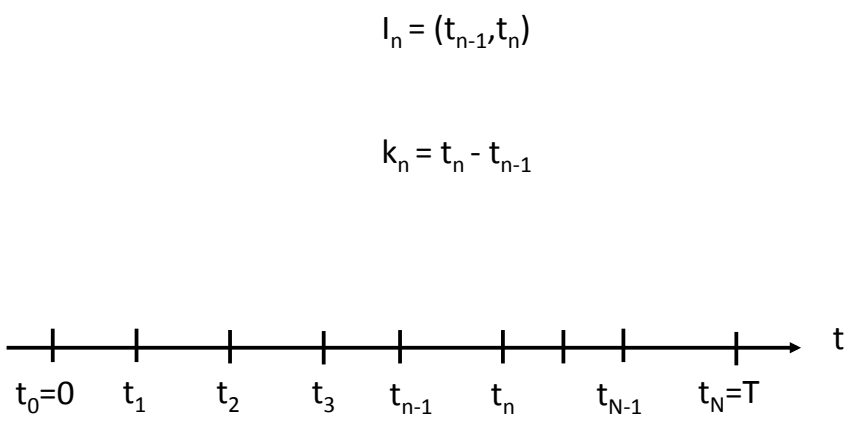

Figure 9.1: Partition of the interval $I = [0, T]$, $0 = t_0 < ... < t_N = T$.

Forward Euler method

For node t_n, we may approximate the derivative $\dot{u}(t_n)$ by

$$\dot{u}(t_{n-1}) \approx \frac{u(t_n) - u(t_{n-1})}{k_n}, \tag{9.2}$$

so that

$$u(t_n) \approx u(t_{n-1}) + k_n \dot{u}(t_{n-1}) = u(t_{n-1}) + k_n f(u(t_{n-1}), t_{n-1}), \tag{9.3}$$

which motivates the *forward Euler method* for successive computational approximation of $U_n = U(t_n)$.

Algorithm 14: Forward Euler method

$U_0 = u_0$ ▷ `initial approximation`
for $n = 1, 2, ..., N$ **do**
  $U_n = U_{n-1} + k_n f(U_{n-1}, t_{n-1})$ ▷ `explicit update`
end

We note that the forward Euler method is *explicit*, meaning that U_{n+1} is directly computable from the previous solution U_n in the time-stepping algorithm. The method is thus also referred to as the *explicit Euler method*.

Backward Euler method

Alternatively, we may approximate the derivative $\dot{u}(t_{n+1})$ by

$$\dot{u}(t_n) \approx \frac{u(t_n) - u(t_{n-1})}{k_n}, \tag{9.4}$$

so that

$$u(t_n) \approx u(t_{n-1}) + k_n \dot{u}(t_n) = u(t_{n-1}) + k_n f(u(t_n), t_n), \tag{9.5}$$

which motivates the *backward Euler method* for successive computational approximation of $U_n = U(t_n)$.

Algorithm 15: Backward Euler method

$U_0 = u_0$ ▷ `initial approximation`
for $n = 1, 2, ..., N$ **do**
  $U_n = U_{n-1} + k_n f(U_n, t_n)$ ▷ `solve algebraic equation`
end

Contrary to the forward Euler method, the backward Euler method is *implicit*, thus also referred to as the *implicit Euler method*, meaning that U_{n+1} is not directly computable from U_n, but is obtained from the solution of an algebraic (possibly nonlinear) equation,

$$x = U_{n-1} + k_n f(x, t_n), \tag{9.6}$$

for example, by the fixed point iteration,

$$x^{(k+1)} = U_{n-1} + k_n f(x^{(k)}, t_n). \tag{9.7}$$

We note that the fixed point iteration (9.7) converges if $k_n L_f < 1$, with L_f the Lipschitz constant of the function $f(\cdot, t_n)$, thus if the time step k_n is small enough.

Time stepping as quadrature

There is a strong connection between time-stepping methods and numerical approximation of integrals, referred to as *quadrature*. For example, assume that the initial condition is zero and that the function f in (9.1) does not depend on the solution u, but the time t only, that is $f = f(t)$. The solution $u(t)$ is then the primitive function of $f(t)$ that satisfies $u(0) = 0$, corresponding to the area under the graph of the function $f(t)$ over the interval $[0, t]$.

We can approximate this primitive function by left and right *rectangular rule* quadrature, or *Riemann sums*, which we illustrate in Figure 9.2. The two approximations of the area under the graph then corresponds to the forward and backward Euler approximations to the initial value problem (9.1) with $u_0 = 0$ and $f = f(t)$.

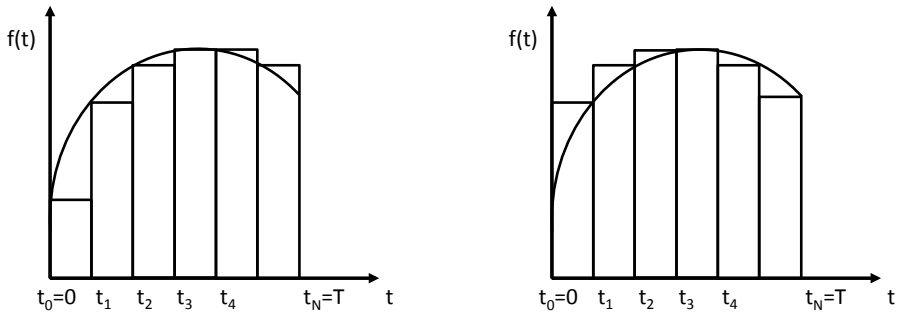

Figure 9.2: Left (left) and right (right) rectangular rule quadrature, or *Riemann sums*, approximating the primitive function of $f(t)$, corresponding to the area under the graph.

More generally, by the *Fundamental Theorem of Calculus* we have, for each subinterval $I_n = (t_{n-1}, t_n)$, that

$$u(t_n) = u(t_{n-1}) + \int_{t_{n-1}}^{t_n} f(u(t), t)\, dt, \tag{9.8}$$

from which we can derive suitable time stepping methods corresponding to different quadrature rules used to evaluate the integral in (9.8).

Quadrature as interpolation

Quadrature as an approximate integral of an exact function, can alternatively be expressed as exact integration of an approximate function. For example, the rectangular quadrature rules in Figure 9.2, corresponds to exact integration of a piecewise constant (over each subinterval I_n) approximation of the function $f(t)$, with its value determined by the function value at the left or right endpoint of the interval, $f(t_{n-1})$ or $f(t_n)$.

From this perspective, one may ask if such a piecewise constant approximation can be chosen in a more clever way to reduce the error in the approximation of the integral, which naturally leads to the *midpoint rule* where the piecewise constant function is chosen based on the function value at the midpoint of the subinterval, that is $(f(t_{n-1}) + f_n)/2$.

Further, we may seek to approximate the function by a higher order polynomial. By linear *interpolation* over the partition $\mathcal{T}_k$, corresponding approximation of the function by a continuous piecewise linear polynomial which is exact at each node t_n, exact integration corresponds to the *trapezoidal rule*.

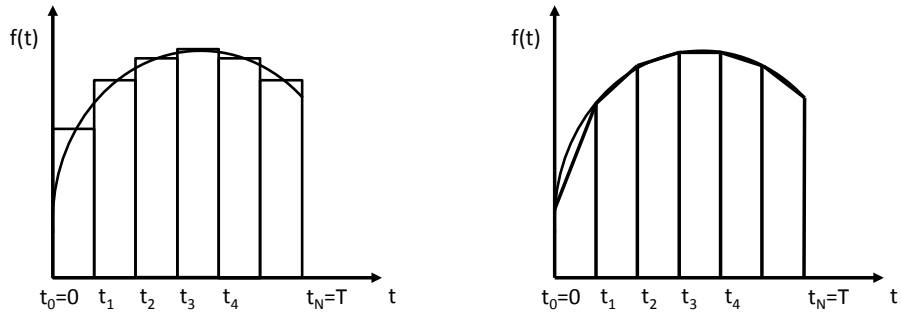

Figure 9.3: Midpoint (left) and trapezoidal (right) quadrature rules, corresponding to interpolation by a piecewise constant and piecewise linear function respectively.

Interpolation as time stepping

By (9.8), the midpoint and trapezoidal rules can also be formulated as time stepping methods. Although, in the case of a time stepping method, interpolation cannot be directly based on the function $f(u(t), t)$ since $u(t)$ is unknown. Instead we seek an approximate solution $U(t)$ to the initial value

problem (9.1) as a piecewise polynomial of a certain order, determined by $f(U_n, t_n)$ through the successive approximations U_n.

Algorithm 16: Trapezoidal time stepping method

$U_0 = u_0$ ▷ `initial approximation`

for $n = 1, 2, ..., N$ **do**

  $U_n = U_{n-1} + \frac{k_n}{2}(f(U_n, t_n) + f(U_{n-1}, t_{n-1}))$ ▷ `solve equation`

end

Both the midpoint method and the trapezoidal method are implicit, and thus require the solution of an algebraic equation, possibly nonlinear, at each time step.

To seek approximate solutions to differential equations as piecewise polynomials is a powerful idea that we will meet many times. Any piecewise polynomial function can be expressed as a linear combination of basis functions, for which the coordinates are to be determined, for example, based on minimization or orthogonality conditions on the residual of the differential equation.

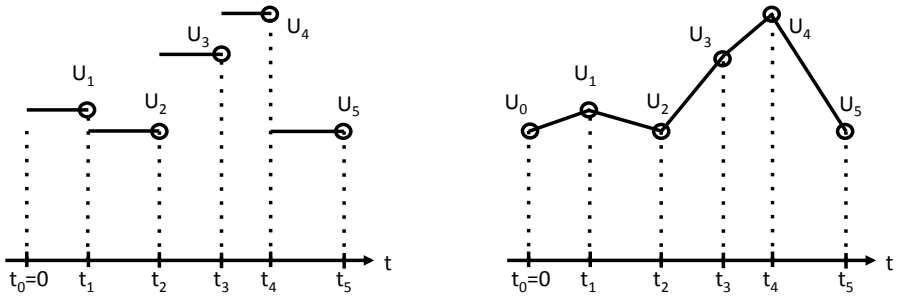

Figure 9.4: Examples of a discontinuous piecewise constant polynomial determined by its value at the right endpoint of the subinterval (left), and a continuous piecewise linear polynomial determined by its value in the nodes of the partition (right).

The residual of (9.1) is given by

$$R(U(t)) = f(U(t), t) - \dot{U}(t), \tag{9.9}$$

and in the case of a continuous piecewise linear approximation $U(t)$, an orthogonality condition enforces the integral of the residual to be zero over

each time interval I_n, that is

$$\int_{t_{n-1}}^{t_n} R(U(t))\ dt = 0, \tag{9.10}$$

or equivalently,

$$U_n = U_{n-1} + \int_{t_{n-1}}^{t_n} f(U(t), t)\ dt,$$

which corresponds to (9.8). If $f(\cdot, \cdot)$ is a linear function, it follows that

$$U_n = U_{n-1} + \frac{k_n}{2}(f(U_n, t_n) + f(U_{n-1}, t_{n-1})), \tag{9.11}$$

else we have to choose a quadrature rule to approximate the integral, with (9.11) corresponding to a trapezoidal rule for a nonlinear function $f(\cdot, \cdot)$.

The θ-method

We can formulate the forward and backward Euler methods, and the trapezoidal method, as one single method with a parameter θ, the *θ-method*. For $\theta = 1$, we get the explicit Euler method, for $\theta = 0$ the implicit Euler method, and $\theta = 0.5$ corresponds to the trapezoidal rule.

Algorithm 17: The θ-method for initial value problems

$U_0 = u_0$ ▷ `initial approximation`
for $n = 1, 2, ..., N-1$ **do**
  $U_n = U_{n-1} + k_n((1-\theta)f(U_n, t_n) + \theta f(U_{n-1}, t_{n-1}))$ ▷ `update`
end

Theorem 16 (Local error estimate for the θ-method)**.** *For the θ-method over one subinterval $I_n = (t_{n-1}, t_n)$ of length $k_n = t_n - t_{n-1}$, with $U_{n-1} = u(t_{n-1})$, we have the following local error estimate,*

$$|u(t_n) - U_n| = \mathcal{O}(k_n^3), \tag{9.12}$$

for $\theta = 1/2$, and if $\theta \neq 1/2$,

$$|u(t_n) - U_n| = \mathcal{O}(k_n^2). \tag{9.13}$$

Proof. With the notation $f_n = f(U_n, t_n)$, so that $f_{n-1} = \dot{u}(t_{n-1})$ and $f_n = \dot{u}(t_n)$, and observing that $U_{n-1} = u(t_{n-1})$, we have by Taylor's formula that

$$\begin{aligned}
|u(t_n) - U_n| &= |u(t_n) - (U_{n-1} + k_n((1-\theta)f_n + \theta f_{n-1}))| \\
&= |u(t_{n-1}) + k_n\dot{u}(t_{n-1}) + \frac{1}{2}k_n^2\ddot{u}(t_{n-1}) + \mathcal{O}(k_n^3) \\
&\quad -(u(t_{n-1}) + k_n((1-\theta)\dot{u}(t_n) + \theta\dot{u}(t_{n-1}))| \\
&= |k_n\dot{u}(t_{n-1}) + \frac{1}{2}k_n^2\ddot{u}(t_{n-1}) + \mathcal{O}(k_n^3) \\
&\quad -(k_n((1-\theta)(\dot{u}(t_{n-1}) + k_n\ddot{u}(t_{n-1}) + \mathcal{O}(k_n^2)) + \theta\dot{u}(t_{n-1}))| \\
&= |\theta - \frac{1}{2}||\ddot{u}(t_{n-1})|k_n^2 + \mathcal{O}(k_n^3).
\end{aligned}$$

□

9.3 System of initial value problems

We now consider systems of initial value problems for a vector valued function $u : \mathbb{R}^+ \to \mathbb{R}^n$ defined on the interval $I = [0, T]$, with derivative $\dot{u} = du/dt = (du_1/dt, ..., du_n/dt)^T$ defined by the function $f : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^n$, such that

$$\begin{aligned}
&\dot{u}(t) = f(u(t), t), \quad 0 < t \leq T, \\
&u(0) = u_0.
\end{aligned} \tag{9.14}$$

Time stepping methods for (9.14) are analogous to the scalar case (9.1), including the θ-method of Algorithm 17, with the difference that for implicit methods a system of (possibly nonlinear) equations needs to be solved.

Newton's laws of motion

Newton's laws of motion for a particle can be formulated as an initial value problem (9.14), with *Newton's 2nd law* expressing that force equals mass times acceleration,

$$m\ddot{x}(t) = F(t), \tag{9.15}$$

given by

$$u = \begin{bmatrix} v \\ x \end{bmatrix}, \quad f = \begin{bmatrix} F/m \\ v \end{bmatrix}, \tag{9.16}$$

for $x = x(t)$ the particle position, $v = v(t) = \dot{x}(t)$ the velocity, m the mass of the particle, and $F = F(t)$ the force applied.

For example, the force $F = mg$ models gravitation, with g the gravitation constant, and the force $F = -kx$ models an elastic spring (Hooke's law) with spring constant k. *Newton's first law* that expresses that a particle remains in its state in the absence of a force, follows from (9.15) in the case $F = 0$.

The N-body problem

Newton's third law states that if one particle p_i exerts a force F_{ji} on another particle p_j, then p_j exerts a force $F_{ij} = -F_{ji}$ on p_i, of the same magnitude but in the opposite direction.

The *N-body problem* refers to the initial value problem (9.14) describing Newton's laws of motion for a system of N particles $\{p_i\}_{i=1}^N$, with the pairwise force interactions F_{ij} given by the system under study, for example gravitation in celestial mechanics, Coulomb interactions in electrostatics, Hookean springs in elasticity theory, or interatomic potentials in molecular dynamics simulations.

The N-body problem takes the form (9.14) in $\mathbb{R}^{2N}$,

$$u = \begin{bmatrix} v_1 \\ \vdots \\ v_N \\ x_1 \\ \vdots \\ x_N \end{bmatrix}, \quad f = \begin{bmatrix} F_1/m_1 \\ \vdots \\ F_N/m_N \\ v_1 \\ \vdots \\ v_N \end{bmatrix} \tag{9.17}$$

with the resulting force on particle p_i given by the sum of all pairwise interactions,

$$F_i = \sum_{j \neq i}^{N} F_{ij}. \tag{9.18}$$

To solve (9.17) using a time-stepping is an $\mathcal{O}(N^2)$ algorithm, which is very expensive for N large. Optimized algorithms of order $\mathcal{O}(N)$ can be developed based on the idea of clustering the force from multiple particles at a distance.

Celestial mechanics

Newton's gravitational law models pairwise gravitational force interactions, which can be used to model the solar system for example. Every particle

p_i is effected by the sum of all other particles, and the force F_{ij} acting on particle p_i by p_j, is given by,

$$F_{ij} = G\frac{m_i m_j}{\|x_i - x_j\|^2}, \tag{9.19}$$

where $m_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^n$ denotes the mass and position of particle p_i.

Mass-spring model

The forces in a mass-spring model represent pairwise force interactions between adjacent particles in a *lattice*, connected via springs, such that the force F_{ij} acting on particle p_i by p_j is given by,

$$F_{ij} = -k_{ij}(x_i - x_j), \tag{9.20}$$

with $k_{ij} = k_{ji}$ the relevant spring constant.

Chapter 10

Function approximation

We have studied methods for computing solutions to algebraic equations in the form of real numbers or finite dimensional vectors of real numbers. In contrast, solutions to differential equations are scalar or vector valued functions, which only in simple special cases are analytical functions that can be expressed by a closed mathematical formula.

Instead we use the idea to approximate general functions by linear combinations of a finite set of simple analytical functions, for example trigonometric functions, splines or polynomials, for which attractive features are orthogonality and locality. We focus in particular on piecewise polynomials defined by the finite set of nodes of a mesh, which exhibit both near orthogonality and local support.

10.1 Function approximation

The Lebesgue space $L^2(I)$

Inner product spaces provide tools for approximation based on orthogonal projections on subspaces. We now introduce an inner product space for functions on the interval $I = [a, b]$, the *Lebesgue space* $L^2(I)$, defined as the class of all square integrable functions $f : I \to \mathbb{R}$,

$$L^2(I) = \{f : \int_a^b |f(x)|^2 \, dx < \infty\}. \tag{10.1}$$

The vector space $L^2(I)$ is closed under the basic operations of pointwise addition and scalar multiplication, by the inequality,

$$(a + b)^2 \leq 2(a^2 + b^2), \quad \forall a, b \geq 0, \tag{10.2}$$

which follows from *Young's inequality*.

Theorem 17 (Young's inequality)**.** *For $a, b \geq 0$ and $\epsilon > 0$,*

$$ab \leq \frac{1}{2\epsilon}a^2 + \frac{\epsilon}{2}b^2 \tag{10.3}$$

Proof. $0 \leq (a - \epsilon b)^2 = a^2 + \epsilon^2 b^2 - 2ab\epsilon$. □

The L^2-inner product is defined by

$$(f, g) = (f, g)_{L^2(I)} = \int_a^b f(x)g(x)\,dx, \tag{10.4}$$

with the associated L^2 norm,

$$\|f\| = \|f\|_{L^2(I)} = (f, f)^{1/2} = \left(\int_a^b |f(x)|^2\,dx\right)^{1/2}, \tag{10.5}$$

for which the Cauchy-Schwarz inequality is satisfied,

$$|(f, g)| \leq \|f\|\|g\|. \tag{10.6}$$

Approximation of functions in $L^2(I)$

We seek to approximate a function f in a vector space V by a linear combination of functions $\phi_j \in V$, that is

$$f(x) \approx f_n(x) = \sum_{j=1}^{n} \alpha_j \phi_j(x), \tag{10.7}$$

with $\alpha_j \in \mathbb{R}$. If linearly independent, the set $\{\phi_j\}_{j=1}^n$ spans a subspace $S \subset V$, that is

$$S = \{f_n \in V : f_n = \sum_{j=1}^{n} \alpha_j \phi_j(x),\ \forall \alpha_j \in \mathbb{R}\}, \tag{10.8}$$

with the set $\{\phi_j\}_{j=1}^n$ a basis for S. For example, in a *Fourier series* the basis functions ϕ_j are trigonometric functions, in a *power series* monomials.

The question is now how to determine the coordinates α_j so that $f_n(x)$ is a good approximation of $f(x)$ in the subspace S. One approach to the problem is to use the techniques of orthogonal projections previously studied for vectors in $\mathbb{R}^n$, an alternative approach is interpolation, where α_j are chosen such that $f_n(x_i) = f(x_i)$, in a set of nodes x_i, for $i = 1, ..., n$. If we cannot evaluate the function $f(x)$ in arbitrary points x, but only have access to a set of sampled data points $\{(x_i, f_i)\}_{i=1}^m$, with $m \geq n$, we can formulate a least squares problem to determine the coordinates α_j that minimize the error $f(x_i) - f_i$, in a suitable norm.

L^2 projection

The L^2 *projection* Pf, onto the subspace $S \subset V$, defined by (10.8), of a function $f \in V$, with $V = L^2(I)$, is the orthogonal projection of f on S, that is,

$$(f - Pf, s) = 0, \quad \forall s \in S, \tag{10.9}$$

which corresponds to,

$$\sum_{j=1}^{n} \alpha_j(\phi_i, \phi_j) = (f, \phi_i), \quad \forall i = 1, ..., n. \tag{10.10}$$

By solving the matrix equation $Ax = b$, with $a_{ij} = (\phi_i, \phi_j)$, $x_j = \alpha_j$, and $b_i = (f, \phi_i)$, we obtain the L_2 projection as

$$Pf(x) = \sum_{j=1}^{n} \alpha_j \phi_j(x). \tag{10.11}$$

We note that if $\phi_i(x)$ has *local support*, that is $\phi_i(x) \neq 0$ only for a subinterval of I, then the matrix A is sparse, and for $\{\phi_i\}_{i=1}^n$ an orthonormal basis, A is the identity matrix with $\alpha_j = (f, \phi_j)$.

Interpolation

The *interpolant* $\pi f \in S$, is determined by the condition that $\pi f(x_i) = f(x_i)$, for n nodes $\{x_i\}_{i=1}^n$. That is,

$$f(x_i) = \pi f(x_i) = \sum_{j=1}^{n} \alpha_j \phi_j(x_i), \quad i = 1, ..., n, \tag{10.12}$$

which corresponds to the matrix equation $Ax = b$, with $a_{ij} = \phi_j(x_i)$, $x_j = \alpha_j$, and $b_i = f(x_i)$.

The matrix A is an identity matrix under the condition that $\phi_j(x_i) = 1$, for $i = j$, and zero else. We then refer to $\{\phi_i\}_{i=1}^n$ as a *nodal basis*, for which $\alpha_j = f(x_j)$, and we can express the interpolant as

$$\pi f(x) = \sum_{j=1}^{n} \alpha_j \phi_j(x) = \sum_{j=1}^{n} f(x_j)\phi_j(x). \tag{10.13}$$

Regression

If we cannot evaluate the function $f(x)$ in arbitrary points, but only have access to a set of data points $\{(x_i, f_i)\}_{i=1}^m$, with $m \geq n$, we can formulate the least squares problem,

$$\min_{f_n \in S} \|f_i - f_n(x_i)\| = \min_{\{\alpha_j\}_{j=1}^n} \|f_i - \sum_{j=1}^n \alpha_j \phi_j(x_i)\|, \quad i = 1, ..., m, \tag{10.14}$$

which corresponds to minimization of the residual $b - Ax$, with $a_{ij} = \phi_j(x_i)$, $b_i = f_i$, and $x_j = \alpha_j$, which we can solve by forming the normal equations,

$$A^T Ax = A^T b. \tag{10.15}$$

10.2 Piecewise polynomial approximation

Polynomial spaces

We introduce the vector space $\mathcal{P}^q(I)$, defined by the set of polynomials

$$p(x) = \sum_{i=0}^q c_i x^i, \quad x \in I, \tag{10.16}$$

of at most order q on an interval $I \in \mathbb{R}$, with the basis functions x^i and coordinates c_i, and the basic operations of pointwise addition and scalar multiplication,

$$(p + r)(x) = p(x) + r(x), \quad (\alpha p)(x) = \alpha p(x), \tag{10.17}$$

for $p, r \in \mathcal{P}^q(I)$ and $\alpha \in \mathbb{R}$. One basis for $\mathcal{P}^q(I)$ is the set of *monomials* $\{x^i\}_{i=0}^q$, another is $\{(x - c)^i\}_{i=0}^q$, which gives the *power series*,

$$p(x) = \sum_{i=0}^q a_i (x - c)^i = a_0 + a_1(x - c) + ... + a_q(x - c)^q, \tag{10.18}$$

for $c \in I$, with a Taylor series being an example of a power series,

$$f(x) = f(y) + f'(y)(x - y) + \frac{1}{2} f''(y)(x - y)^2 + ... \tag{10.19}$$

Langrange polynomials

For a set of nodes $\{x_i\}_{i=0}^q$, we define the *Lagrange polynomials* $\{\lambda\}_{i=0}^q$, by

$$\lambda_i(x) = \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_q)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_q)} = \prod_{i\neq j}\frac{x-x_j}{x_i-x_j},$$

that constitutes a basis for $\mathcal{P}^q(I)$, and we note that

$$\lambda_i(x_j) = \delta_{ij}, \tag{10.20}$$

with the *Dirac delta function* defined as

$$\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{10.21}$$

so that $\{\lambda\}_{i=0}^q$ is a nodal basis, which we refer to as the *Lagrange basis*. We can express any $p \in \mathcal{P}^q(I)$ as

$$p(x) = \sum_{i=1}^{q} p(x_i)\lambda_i(x), \tag{10.22}$$

and by (10.13) we can define the *polynomial interpolant* $\pi_q f \in \mathcal{P}^q(I)$,

$$\pi_q f(x) = \sum_{i=1}^{q} f(x_i)\lambda_i(x), \quad x \in I, \tag{10.23}$$

for a continuous function $f \in \mathcal{C}(I)$.

Piecewise polynomial spaces

We now introduce piecewise polynomials defined over a partition of the interval $I = [a, b]$,

$$a = x_0 < x_1 < \cdots < x_{m+1} = b, \tag{10.24}$$

for which we let the *mesh* $\mathcal{T}_h = \{I_i\}$ denote the set of subintervals $I_j = (x_{i-1}, x_i)$ of length $h_i = x_i - x_{i-1}$, with the *mesh function*,

$$h(x) = h_i, \quad \text{for } x \in I_i. \tag{10.25}$$

We define two vector spaces of *piecewise polynomials*, the discontinuous piecewise polynomials on I, defined by

$$W_h^{(q)} = \{v : v|_{I_i} \in \mathcal{P}^q(I_i),\ i = 1, ..., m+1\}, \tag{10.26}$$

and the continuous piecewise polynomials on I, defined by

$$V_h^{(q)} = \{v \in W_h^{(q)} : v \in \mathcal{C}(I)\}. \tag{10.27}$$

The basis functions for $W_h^{(q)}$ can be defined in terms of the Lagrange basis, for example,

$$\lambda_{i,0}(x) = \frac{x_i - x}{x_i - x_{i-1}} = \frac{x_i - x}{h_i} \tag{10.28}$$

$$\lambda_{i,1}(x) = \frac{x - x_{i-1}}{x_i - x_{i-1}} = \frac{x - x_{i-1}}{h_i} \tag{10.29}$$

defining the basis functions for $W_h^{(1)}$, by

$$\phi_{i,j}(x) = \begin{cases} 0, & x \neq [x_{i-1}, x_i], \\ \lambda_{i,j}, & x \in [x_{i-1}, x_i], \end{cases} \tag{10.30}$$

for $i = 1, ..., m+1$, and $j = 0, 1$. For $V_h^{(q)}$ we need to construct continuous basis functions, for example,

$$\phi_i(x) = \begin{cases} 0, & x \neq [x_{i-1}, x_{i+1}], \\ \lambda_{i,1}, & x \in [x_{i-1}, x_i], \\ \lambda_{i+1,0}, & x \in [x_i, x_{i+1}], \end{cases} \tag{10.31}$$

for $V_h^{(1)}$, which we also refer to as *hat functions*.

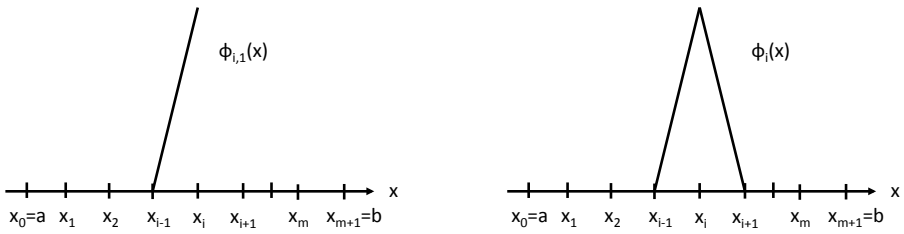

Figure 10.1: Illustration of a *mesh* $\mathcal{T}_h = \{I_i\}$, with subintervals $I_j = (x_{i-1}, x_i)$ of length $h_i = x_i - x_{i-1}$, and $\phi_{i,1}(x)$ a basis function for $W_h^{(1)}$ (left), and a basis function $\phi_i(x)$ for $V_h^{(1)}$ (right).

L^2 projection in $V_h^{(1)}$

The L^2 projection of a function $f \in L^2(I)$ onto the space of continuous piecewise linear polynomials $V_h^{(1)}$, is given by

$$Pf(x) = \sum_{j=1}^{n} \alpha_j \phi_j(x), \tag{10.32}$$

with the coordinates α_j determined by from the matrix equation

$$Mx = b, \tag{10.33}$$

with $m_{ij} = (\phi_j, \phi_i)$, $x_j = \alpha_j$, and $b_i = (f, \phi_i)$. The matrix M is sparse, since $m_{ij} = 0$ for $|i - j| > 1$, and for large n we need to use an iterative method to solve (10.33). We compute the entries of the matrix M, referred to as a *mass matrix*, from the definition of the basis functions (10.31), starting with the diagonal entries,

$$\begin{aligned} m_{ii} &= (\phi_i, \phi_i) = \int_0^1 \phi_i^2(x)\,dx = \int_{x_{i-1}}^{x_i} \lambda_{i,1}^2(x)\,dx + \int_{x_i}^{x_{i+1}} \lambda_{i+1,0}^2(x)\,dx \\ &= \int_{x_{i-1}}^{x_i} \frac{(x - x_{i-1})^2}{h_i^2}\,dx + \int_{x_i}^{x_{i+1}} \frac{(x_{i+1} - x)^2}{h_{i+1}^2}\,dx \\ &= \frac{1}{h_i^2}\left[\frac{(x - x_{i-1})^3}{3}\right]_{x_{i-1}}^{x_i} + \frac{1}{h_{i+1}^2}\left[\frac{-(x_{i+1} - x)^3}{3}\right]_{x_i}^{x_{i+1}} = \frac{h_i}{3} + \frac{h_{i+1}}{3}, \end{aligned}$$

and similarly we compute the off-diagonal entries,

$$\begin{aligned} m_{ii+1} &= (\phi_i, \phi_{i+1}) = \int_0^1 \phi_i(x)\phi_{i+1}(x)\,dx = \int_{x_i}^{x_{i+1}} \lambda_{i+1,0}(x)\lambda_{i+1,1}(x)\,dx \\ &= \int_{x_i}^{x_{i+1}} \frac{(x_{i+1} - x)}{h_{i+1}} \frac{(x - x_i)}{h_{i+1}}\,dx \\ &= \frac{1}{h_{i+1}^2} \int_{x_i}^{x_{i+1}} (x_{i+1}x - x_{i+1}x_i - x^2 + xx_i)\,dx \\ &= \frac{1}{h_{i+1}^2}\left[\frac{x_{i+1}x^2}{2} - x_{i+1}x_i x - \frac{x^3}{3} + \frac{x^2 x_i}{2}\right]_{x_i}^{x_{i+1}} \\ &= \frac{1}{6h_{i+1}^2}(x_{i+1}^3 - 3x_{i+1}^2 x_i + 3x_{i+1}x_i^2 - x_i^3) \\ &= \frac{1}{6h_{i+1}^2}(x_{i+1} - x_i)^3 = \frac{h_{i+1}}{6}, \end{aligned}$$

and

$$m_{ii-1} = (\phi_i, \phi_{i-1}) = \int_0^1 \phi_i(x)\phi_{i-1}(x)\,dx = ... = \frac{h_i}{6}.$$

Chapter 11

The boundary value problem

The boundary value problem in one variable is an ordinary differential equation, for which an initial condition is not enough, instead we need to specify boundary conditions at each end of the interval. Contrary to the initial value problem, the dependent variable does not represent time, but should rather we thought of as a spatial coordinate.

11.1 The boundary value problem

The boundary value problem

We consider the following *boundary value problem*, for which we seek a function $u(x) \in \mathcal{C}^2(0,1)$, such that

$$-u''(x) = f(x), \quad x \in (0,1), \tag{11.1}$$
$$u(0) = u(1) = 0, \tag{11.2}$$

given a source term $f(x)$, and *boundary conditions* at the endpoints of the interval $I = [0,1]$.

We want to find an approximate solution to the boundary value problem in the form of a continuous piecewise polynomial that satisfies the boundary conditions (11.2), that is we seek

$$U \in V_h = \{v \in V_h^{(q)} : v(0) = v(1) = 0\}, \tag{11.3}$$

such that the error $e = u - U$ is small in some suitable norm $\|\cdot\|$.

The *residual* of the equation is defined as

$$R(w) = w'' + f, \tag{11.4}$$

with $R(u) = 0$, for $u = u(x)$ the solution of the boundary value problem. Our strategy is now to find an approximate solution $U \in V_h \subset \mathcal{C}^2(0,1)$ such that $R(U) \approx 0$.

We have two natural methods to find a solution $U \in V_h$ with a minimal residual: (i) the *least squares method*, where we seek the solution with the minimal residual measured in the L_2-norm,

$$\min_{U \in V_h} \|R(U)\|, \tag{11.5}$$

and (ii) *Galerkin's method*, where we seek the solution for which the residual is orthogonal the subspace V_h,

$$(R(U), v) = 0, \quad \forall v \in V_h. \tag{11.6}$$

With an approximation space consisting of piecewise polynomials, we refer to the methods as a *least squares finite element method*, and a *Galerkin finite element method*. With a trigonometric approximation space we refer to Galerkin's method as a *spectral method*.

Galerkin finite element method

The finite element method (FEM) based on (11.6) takes the form: find $U \in V_h$, such that

$$\int_0^1 -U''(x)v(x)\,dx = \int_0^1 f(x)v(x)\,dx, \tag{11.7}$$

for all *test functions* $v \in V_h$. For (11.12) to be well defined, we need to be able to represent the second order derivative U'', which is not obvious for low order polynomials, such as linear polynomials, or piecewise constants.

To reduce this constraint, we can use partial integration to move one derivative from the approximation U to the test function v, so that

$$\int_0^1 -U''(x)v(x)\,dx = \int_0^1 U'(x)v'(x)\,dx - [U'(x)v(x)]_0^1 = \int_0^1 U'(x)v'(x)\,dx,$$

since $v \in V_h$, and thus satisfies the boundary conditions. The finite element method now reads: find $U \in V_h$, such that,

$$\int_0^1 U'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx, \tag{11.8}$$

for all $v \in V_h$.

The discrete problem

We now let V_h be the space of continuous piecewise linear functions, that satisfies the boundary conditions (11.2), that is,

$$U \in V_h = \{v \in V_h^{(1)} : v(0) = v(1) = 0\}, \tag{11.9}$$

so that we can write any function $v \in V_h$ as

$$v(x) = \sum_{i=1}^{n} v_i \phi_i(x), \tag{11.10}$$

over a mesh $\mathcal{T}_h$ with n internal nodes x_i, and $v_i = v(x_i)$ since $\{\phi_i\}_{i=1}^n$ is a nodal basis.

We thus search for an approximate solution

$$U(x) = \sum_{j=1}^{n} U_j \phi_j(x), \tag{11.11}$$

with $U_j = U(x_j)$. If we insert (11.10) and (11.11) into (11.12), we get

$$\sum_{j=1}^{n} U_j \int_0^1 \phi_j'(x)\phi_i'(x)\,dx = \int_0^1 f(x)\phi_i(x)\,dx, \quad i = 1, ..., n, \tag{11.12}$$

which corresponds to the matrix equation

$$Sx = b, \tag{11.13}$$

with $s_{ij} = (\phi_j', \phi_i')$, $x_j = U_j$, and $b_i = (f, \phi_i)$. The matrix S is sparse, since $s_{ij} = 0$ for $|i - j| > 1$, and for large n we need to use an iterative method to solve (11.13).

We compute the entries of the matrix S, referred to as a *stiffness matrix*, from the definition of the basis functions (10.31), starting with the diagonal entries,

$$\begin{aligned} s_{ii} &= (\phi_i', \phi_i') = \int_0^1 (\phi_i')^2(x)\,dx = \int_{x_{i-1}}^{x_i} (\lambda_{i,1}')^2(x)\,dx + \int_{x_i}^{x_{i+1}} (\lambda_{i+1,0}')^2(x)\,dx \\ &= \int_{x_{i-1}}^{x_i} \left(\frac{1}{h_i}\right)^2 dx + \int_{x_i}^{x_{i+1}} \left(\frac{1}{h_{i+1}}\right)^2 dx = \frac{1}{h_i} + \frac{1}{h_{i+1}}, \end{aligned}$$

and similarly we compute the off-diagonal entries,

$$s_{ii+1} = (\phi_i', \phi_{i+1}') = \int_0^1 \phi_i'(x)\phi_{i+1}'(x)\,dx = \int_{x_i}^{x_{i+1}} \frac{-1}{h_{i+1}}\frac{1}{h_{i+1}}\,dx = -\frac{1}{h_{i+1}},$$

and

$$s_{ii-1} = (\phi_i', \phi_{i-1}') = \int_0^1 \phi_i'(x)\phi_{i-1}'(x)\,dx = ... = -\frac{1}{h_i}.$$

The variational problem

Galerkin's method is based on the *variational formulation*, or *weak form*, of the boundary value problem, where we search for solution in a vector space V, for which the variational form is well defined: find $u \in V$, such that

$$\int_0^1 u'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx, \tag{11.14}$$

for all $v \in V$.

To construct an appropriate vector space V for (11.1) to be well defined, we need to extend L^2 spaces to include also derivatives, which we refer to as *Sobolev spaces*. We introduce the vector space $H^1(0,1)$, defined by,

$$H^1(0,1) = \{v \in L^2(0,1) : v' \in L^2(0,1)\}, \tag{11.15}$$

and the vector space that also satisfies the boundary conditions (11.2),

$$H_0^1(0,1) = \{v \in H^1(0,1) : v(0) = v(1) = 0\}. \tag{11.16}$$

The variational form (11.1) is now well defined for $V = H_0^1(0,1)$, since

$$\int_0^1 u'(x)v'(x)\,dx \le \|u'\|\|v'\| < \infty \tag{11.17}$$

by Cauchy-Schwarz inequality, and

$$\int_0^1 f(x)v(x)\,dx \le \|f\|\|v\| < \infty, \tag{11.18}$$

for $f \in L^2(0,1)$.

Optimality of Galerkin's method

Galerkin's method (11.12) corresponds to searching for an approximate solution in a finite dimensional subspace $V_h \subset V$, for which (11.1) is satisfied for all test functions $v \in V_h$.

The Galerkin solution U is the best possible approximation in V_h, in the sense that,

$$\|u - U\|_E \le \|u - v\|_E, \quad \forall v \in V_h, \tag{11.19}$$

with the *energy norm* defined by

$$\|w\|_E = \left(\int_0^1 |w'(x)|^2\,dx\right)^{1/2}. \tag{11.20}$$

Thus $U \in V_h$ represents a projection of $u \in V$ onto V_h, with respect to the inner product defined on V,

$$(v, w)_E = \int_0^1 v'(x)w'(x)\,dx, \tag{11.21}$$

with $\|w\|_E^2 = (w, w)_E$. The *Galerkin orthogonality*,

$$(u - U, v)_E = 0, \quad \forall v \in V_h, \tag{11.22}$$

expresses the optimality of the approximation U, as

$$\|u - U\|_E \leq \|u - v\|_E, \quad \forall v \in V_h, \tag{11.23}$$

which follows by

$$\begin{aligned} \|u - U\|_E^2 &= (u - U, u - u_h)_E = (u - U, u - v)_E + (u - U, v - u_h)_E \\ &= (u - U, u - v)_E \leq \|u - U\|_E \|u - v\|_E, \end{aligned}$$

for any $v \in V_h$.

Chapter 12

Partial differential equations

12.1 Differential operators in $\mathbb{R}^n$

Differential operators

We recall the definition of the gradient of a scalar function as

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n} \right)^T, \tag{12.1}$$

which we can interpret as the differential operator

$$\nabla = \left(\frac{\partial}{\partial x_1}, ..., \frac{\partial}{\partial x_n} \right)^T, \tag{12.2}$$

acting on the function $f = f(x)$, with $x \in \Omega \subset \mathbb{R}^n$. With this interpretation we express two second order differential operators, the *Laplacian* Δf,

$$\Delta f = \nabla^T \nabla f = \frac{\partial^2 f}{\partial x_1^2} + ... + \frac{\partial^2 f}{\partial x_n^2}, \tag{12.3}$$

and the *Hessian* Hf,

$$Hf = \nabla \nabla^T f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}. \tag{12.4}$$

For a vector valued function $f : \mathbb{R}^n \to \mathbb{R}^m$, we define the Jacobian matrix by

$$f' = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} (\nabla f_1)^T \\ \vdots \\ (\nabla f_m)^T \end{bmatrix}, \tag{12.5}$$

and for $m = n$, we define the *divergence* by

$$\nabla \cdot f = \frac{\partial f_1}{\partial x_1} + ... + \frac{\partial f_n}{\partial x_n}. \tag{12.6}$$

Partial integration in $\mathbb{R}^n$

For the scalar function $f : \mathbb{R}^n \to \mathbb{R}$, and the vector valued function $g : \mathbb{R}^n \to \mathbb{R}^n$, we have the following generalization of partial integration over $\Omega \subset \mathbb{R}^n$, referred to as *Green's theorem*,

$$(\nabla f, g) = -(f, \nabla \cdot g) + (f, g \cdot n)_\Gamma, \tag{12.7}$$

where we use the notation,

$$(v, w)_\Gamma = (v, w)_{L^2(\Gamma)} = \int_\Omega vw \, ds, \tag{12.8}$$

for the boundary integral, with $L^2(\Gamma)$ the Lebesgue space defined over the boundary Γ.

Sobolev spaces

The L^2 space for $\Omega \subset \mathbb{R}^n$, is defined by

$$L^2(\Omega) = \{v : \int_\Omega |v|^2 \, dx < \infty\}, \tag{12.9}$$

where in the case of a vector valued function $v : \mathbb{R}^n \to \mathbb{R}^n$, we let

$$|v|^2 = \|v\|_2^2 = v_1^2 + ... + v_n^2. \tag{12.10}$$

To construct appropriate vector spaces for the variational formulation of partial differential equations, we need to extend L^2 spaces to include also derivatives. The *Sobolev space* $H^1(\Omega)$ is defined by,

$$H^1(\Omega) = \{v \in L^2(\Omega) : \frac{\partial v_i}{\partial x_j} \in L^2(\Omega), \, \forall i, j = 1, ..., n\}, \tag{12.11}$$

and we define

$$H_0^1(\Omega) = \{v \in H^1(\Omega) : v(x) = 0, \, x \in \Gamma\}, \tag{12.12}$$

to be the space with functions that are zero on the boundary Γ.

12.2 Poisson's equation

The Poisson equation

We now consider the *Poisson equation* for a function $u \in \mathcal{C}^2(\Omega)$,

$$-\Delta u = f, \quad x \in \Omega, \tag{12.13}$$

with $\Omega \subset \mathbb{R}^n$, and $f \in \mathcal{C}(\Omega)$. For the equation to have a unique solution we need to specify boundary conditions. We may prescribe *Dirichlet boundary conditions*,

$$u = g_D, \quad x \in \Gamma, \tag{12.14}$$

Neumann boundary conditions,

$$\nabla u \cdot n = g_N, \quad x \in \Gamma, \tag{12.15}$$

with $n = n(x)$ the outward unit normal on Γ_N, or a linear combination of the two, which we refer to as a *Robin boundary condition*.

Homogeneous Dirichlet boundary conditions

We now state the variational formulation of Poisson equation with homogeneous Dirichlet boundary conditions,

$$-\Delta u = f, \qquad x \in \Omega, \tag{12.16}$$
$$u = 0, \qquad x \in \Gamma, \tag{12.17}$$

which we obtain by multiplication by a test function $v \in V = H_0^1(\Omega)$ and integration over Ω, using Green's theorem, which gives,

$$(\nabla u, \nabla v) = (f, v), \tag{12.18}$$

since the boundary term vanishes as the test function is an element of the vector space $H_0^1(\Omega)$.

Homogeneous Neumann boundary conditions

We now state the variational formulation of Poisson equation with homogeneous Neumann boundary conditions,

$$-\Delta u = f, \qquad x \in \Omega, \tag{12.19}$$
$$\nabla u \cdot n = 0, \qquad x \in \Gamma, \tag{12.20}$$

which we obtain by multiplication by a test function $v \in V = H^1(\Omega)$ and integration over Ω, using Green's theorem, which gives,

$$(\nabla u, \nabla v) = (f, v), \tag{12.21}$$

since the boundary term vanishes by the Neumann boundary condition. Thus the variational forms (12.18) and (12.21) are similar, with the only difference being the choice of test and trial spaces.

However, it turns out that the variational problem (12.21) has no unique solution, since for any solution $u \in V$, also $v + C$ is a solution, with C a constant. To ensure a unique solution, we need an extra condition for the solution, for example, we may change the approximation space to

$$V = \{v \in H^1(\Omega) : \int_\Omega v(x)\, dx = 0\}. \tag{12.22}$$

Non homogeneous boundary conditions

We now state the variational formulation of Poisson equation with non homogeneous boundary conditions,

$$-\Delta u = f, \qquad x \in \Omega, \tag{12.23}$$
$$u(x) = g_D, \qquad x \in \Gamma_D, \tag{12.24}$$
$$\nabla u \cdot n = g_N, \qquad x \in \Gamma_N, \tag{12.25}$$

with $\Gamma = \Gamma_D \cup \Gamma_N$, which we obtain by multiplication by a test function $v \in V$, with

$$V = \{v \in H^1(\Omega) : v(x) = g_D(x),\, x \in \Gamma_D\}, \tag{12.26}$$

and integration over Ω, using Green's theorem, which gives,

$$(\nabla u, \nabla v) = (f, v) + (g_N, v)_{\Gamma_N}. \tag{12.27}$$

The Dirichlet boundary condition is enforced through the trial space, and is thus referred to as an *essential boundary condition*, whereas the Neumann boundary condition is enforced through the variational form, thus referred to as a *natural boundary condition*.

The finite element method

To compute approximate solutions to the Poisson equation, we can formulate a finite element method based on the variational formulation of the

equation, replacing the Sobolev space V with a polynomial space V_h, constructed by a set of basis functions $\{\phi_i\}_{i=1}^M$, over a mesh $\mathcal{T}_h$, defined as a collection of elements $\{K_i\}_{i=1}^N$ and nodes $\{N_i\}_{i=1}^M$.

For the Poisson equation with homogeneous Dirichlet boundary conditions, the finite element method takes the form: Find $U \in V_h$, such that,

$$(\nabla U, \nabla v) = (f, v), \quad v \in V_h, \tag{12.28}$$

with $V_h \subset H_0^1(\Omega)$.

The variational form (12.28) corresponds to a linear system of equations $Ax = b$, with $a_{ij} = (\phi_j, \phi_i)$, $x_j = U(N_j)$, and $b_i = (f, \phi_i)$, with $\phi_i(x)$ the basis function associated with the node N_i.

12.3 Linear partial differential equations

The abstract problem

We express a linear partial differential equation as the abstract problem,

$$Lu = f, \quad x \in \Omega, \tag{12.29}$$

with boundary conditions,

$$Bu = g, \quad x \in \Gamma, \tag{12.30}$$

for which we can derive a variational formulation: find $u \in V$ such that,

$$a(u, v) = L(v), \quad v \in V, \tag{12.31}$$

with $a : V \times V \to \mathbb{R}$ a *bilinear form*, that is a function which is linear in both arguments, and $L : V \to \mathbb{R}$ a *linear form*.

In a Galerkin method we seek an approximation $U \in V_h$ such that

$$a(U, v) = L(v), \quad v \in V_h, \tag{12.32}$$

with $V_h \subset V$ a finite dimensional subspace, which in the case of a finite element method is a polynomial space.

Energy error estimation

A bilinear form $a(\cdot, \cdot)$ on the Hilbert space V is *symmetric*, if

$$a(v, w) = a(w, v), \quad v, w \in V, \tag{12.33}$$

and *coercive*, or *elliptic*, if

$$a(v,v) \geq c\|v\|, \quad v \in V, \tag{12.34}$$

with $c > 0$. A symmetric and elliptic bilinear form defines an inner product on V, which induces a norm which we refer to as the *energy norm*,

$$\|w\|_E = a(w,w)^{1/2}. \tag{12.35}$$

The Galerkin approximation is optimal in the norm, since by Galerkin orthogonality,

$$a(u-U,v) = 0, \quad v \in V_h, \tag{12.36}$$

we have that

$$\begin{aligned} \|u-U\|_E^2 &= a(u-U,u-u_h) = a(u-U,u-v) + a(u-U,v-u_h) \\ &= a(u-U,u-v) \leq \|u-U\|_E \|u-v\|_E, \end{aligned}$$

so that

$$\|u-U\|_E \leq \|u-v\|_E, \quad v \in V_h. \tag{12.37}$$

Part V

Optimization

Chapter 13

Minimization problems

13.1 Unconstrained minimization

The minimization problem

Find $\bar{x} \in D \subset \mathbb{R}^n$, such that

$$f(\bar{x}) \leq f(x), \quad \forall x \in D, \tag{13.1}$$

with $D \subset \mathbb{R}^n$ the *search space*, $\bar{x} \in D$ the *optimal solution*, and $f : D \to \mathbb{R}$ the *objective function* (or *cost function*).

A *stationary point*, or *critical point*, $\hat{x} \in D$ is a point for which the gradient of the objective function is zero, that is,

$$\nabla f(\hat{x}) = 0, \tag{13.2}$$

and we refer to $x^* \in D$ as a *local minimum* if there exists $\delta > 0$, such that,

$$f(x^*) \leq f(x), \quad \forall x : \|x - x^*\| \leq \delta. \tag{13.3}$$

If the minimization problem is *convex*, an interior local minimum is a global minimum, where in a convex minimization problem the search space is convex, i.e.

$$(1-t)x + ty \in D, \tag{13.4}$$

and the objective function is convex, i.e.

$$(1-t)f(x) + tf(y) \leq f((1-t)x + ty), \tag{13.5}$$

for all $x, y \in D$ and $t \in [0, 1]$.

Gradient descent method

The *level set* of the function $f : D \to \mathbb{R}^n$ is defined as

$$L_c(f) = \{x \in D : f(x) = c\}, \tag{13.6}$$

where we note that $L_c(f)$ represents a level curve in $\mathbb{R}^2$, a level surface in $\mathbb{R}^3$, and more generally a hypersurface of dimension $n-1$ in $\mathbb{R}^n$.

Theorem 18. *If $f \in \mathcal{C}^1(D)$, then the gradient $\nabla f(x)$ is orthogonal to the level set $L_c(f)$ at $x \in D$.*

The *gradient descent method* is an iterative method that compute approximations to a local minimum of (13.1), by searching for the next iterate in a direction orthogonal to the level set $L_c(f)$ in which the objective function decreases, the direction of *steepest descent*, with a *step length* α.

Algorithm 18: Method of steepest descent

Start from $x^{(0)}$ ▷ `initial approximation`
for $k = 1, 2, ...$ **do**
  $x^{(k+1)} = x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)})$ ▷ `Step with length` $\alpha^{(k)}$
end

Newton's method

For $f \in \mathcal{C}^1(D)$ we know by Taylor's formula that,

$$f(x) \approx f(y) + \nabla f(y) \cdot (x - y) + \frac{1}{2}(x - y)^T Hf(y)(x - y), \tag{13.7}$$

for $x, y \in D$, with Hf the Hessian matrix.

Newton's method to find a local minimum in the form of a stationary point is based on (13.7) with $x = x^{(k+1)}$, $y = x^{(k)}$ and $\Delta x = x^{(k+1)} - x^{(k)}$, for which we seek the stationary point, by

$$\begin{aligned} 0 &= \frac{d}{d(\Delta x)} \left(f(x^{(k)}) + \nabla f(x^{(k)}) \cdot \Delta x + \frac{1}{2} \Delta x^T Hf(x^{(k)}) \Delta x \right) \\ &= \nabla f(x^{(k)}) + Hf(x^{(k)}) \Delta x, \end{aligned}$$

which gives Newton's method as an iterative method with increment

$$\Delta x = -(Hf(x^{(k)}))^{-1} \nabla f(x^{(k)}). \tag{13.8}$$

Algorithm 19: Newton's method for finding a stationary point

Start from $x^{(0)}$ ▷ `initial approximation`
for $k = 1, 2, ...$ **do**
  $Hf(x^{(k)})\Delta x = -\nabla f(x^{(k)})$ ▷ `solve linear system for` Δx
  $x^{(k+1)} = x^{(k)} + \Delta x$ ▷ `Update approximation`
end

13.2 Linear system of equations

We now revisit the problem to find a solution $x \in \mathbb{R}^n$ to the system of linear equations

$$Ax = b, \tag{13.9}$$

with $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, with $m \geq n$.

Least square method

The linear system of equations $Ax = b$ can be solved by minimzation algorithms, for example, in the form a least squares problem,

$$\min_{x\in D} f(x), \quad f(x) = \|Ax - b\|^2, \tag{13.10}$$

with $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, and $m \geq n$. The gradient is computed as,

$$\begin{aligned}\nabla f(x) &= \nabla(\|Ax - b\|^2) = \nabla((Ax)^T Ax - (Ax)^T b - b^T Ax + b^T b)\\ &= \nabla(x^T A^T Ax - 2x^T A^T b + b^T b) = A^T Ax + x^T A^T A - 2A^T b\\ &= A^T Ax + A^T Ax - 2A^T b = 2A^T(Ax - b),\end{aligned}$$

which gives the following gradient descent method

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} 2A^T(Ax^{(k)} - b) \tag{13.11}$$

Quadratic forms

We consider the minimization problem,

$$\min_{x\in D} f(x), \tag{13.12}$$

where $f(x)$ is the quadratic form

$$f(x) = x^T Ax - b^T x + c, \tag{13.13}$$

with $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$ and $n \in \mathbb{R}$, with a stationary point given by

$$0 = \nabla f(x) = \nabla(\frac{1}{2}x^T Ax - b^T x + c) = \frac{1}{2}Ax + \frac{1}{2}A^T x - b, \tag{13.14}$$

which in the case A is a symmetric matrix corresponds to the linear system of equations,

$$Ax = b. \tag{13.15}$$

To prove that the solution $x = A^{-1}b$ is the solution of the minimization problem, study the error $e = u - y$, with $y \in D$, for which we have that

$$\begin{aligned} f(x+e) &= \frac{1}{2}(x+e)^T A(x+e) - b^T(x+e) + c \\ &= \frac{1}{2}x^T Ax + e^T Ax + \frac{1}{2}e^T Ae - b^T x - b^T e + c \\ &= (\frac{1}{2}x^T Ax - b^T x + c) + \frac{1}{2}e^T Ae + (e^T b - b^T e) \\ &= f(x) + \frac{1}{2}e^T Ae. \end{aligned}$$

We find that if A is a positive definite matrix $x = A^{-1}b$ is a global minimum, and thus any system of linear equations with a symmetric positive definite matrix may be reformulated as minimization of the associated quadratic form.

If A is not positive definite, it may be negative definite with minimum being $-\infty$, singular with non unique minima, or else the quadratic form $f(x)$ has a *saddle-point*.

Gradient descent method

To solve the minimization problem for a quadratic form, we may use a gradient descent method for which the gradient gives the residual,

$$-\nabla f(x^{(k)}) = b - Ax^{(k)} = r^{(k)}, \tag{13.16}$$

that is,

$$x^{(k+1)} = x^{(k)} - \alpha\nabla f(x^{(k)}) = x^{(k)} + \alpha r^{(k)}. \tag{13.17}$$

To choose a step length α that minimizes $x^{(k+1)}$, we compute the derivative

$$\frac{d}{d\alpha}f(x^{(k+1)}) = \nabla f(x^{(k+1)})^T \frac{d}{d\alpha}x^{(k+1)} = \nabla f(x^{(k+1)})^T r^{(k)} = -(r^{(k+1)})^T r^{(k)},$$

which gives that α should be chosen such that the successive residuals are orthogonal, that is

$$(r^{(k+1)})^T r^{(k)} = 0, \tag{13.18}$$

which gives that

$$\begin{aligned} (r^{(k+1)})^T r^{(k)} &= 0 \\ (b - Ax^{(k+1)})^T r^{(k)} &= 0 \\ (b - A(x^{(k)} + \alpha r^{(k)}))^T r^{(k)} &= 0 \\ (b - A(x^{(k)})^T r^{(k)} - \alpha (Ar^{(k)})^T r^{(k)} &= 0 \\ (r^{(k)})^T r^{(k)} &= \alpha (Ar^{(k)})^T r^{(k)} \end{aligned}$$

so that

$$\alpha = \frac{(r^{(k)})^T r^{(k)}}{(Ar^{(k)})^T r^{(k)}}. \tag{13.19}$$

Algorithm 20: Steepest descent method for $Ax = b$

Start from $x^{(0)}$ ▷ `initial approximation`
for $k = 1, 2, ...$ **do**
  $r^{(k)} = b - Ax^{(k)}$ ▷ `Compute residual` $r^{(k)}$
  $\alpha = (r^{(k)})^T r^{(k)} / (Ar^{(k)})^T r^{(k)}$ ▷ `Compute step length` $\alpha^{(k)}$
  $x^{(k+1)} = x^{(k)} + \alpha^{(k)} r^{(k)}$ ▷ `Step with length` $\alpha^{(k)}$
end

Conjugate gradient method revisited

We now revisit the conjugate gradient (CG) method in the form of a method for solving the minimization of the quadratic form corresponding to a linear system of equations with a symmetric positive definite matrix.

The idea is to formulate a search method,

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}, \tag{13.20}$$

with a set of orthogonal search directions $\{d^{(k)}\}_{k=0}^{n-1}$, where the step length $\alpha^{(k)}$ is determined by the condition that $e^{(k+1)} = x - x^{(k+1)}$ should be *A-orthogonal*, or *conjugate*, to $d^{(k)}$, thus

$$\begin{aligned} (d^{(k)})^T A e^{(k+1)} &= 0 \\ (d^{(k)})^T A (e^{(k)} - \alpha^{(k)} d^{(k)}) &= 0, \end{aligned}$$

so that

$$\alpha = \frac{(d^{(k)})^T A e^{(k)}}{(d^{(k)})^T A d^{(k)}} = \frac{(d^{(k)})^T r^{(k)}}{(d^{(k)})^T A d^{(k)}}. \tag{13.21}$$

To construct the orthogonal search directions $d^{(k)}$ we can use the Gram-Schmidt iteration, whereas if we choose the search direction to be the residual we get the steepest descent method.

13.3 Constrained minimization

The constrained minimization problem

We now consider the *constrained minimization problem*,

$$\min_{x \in D} f(x) \tag{13.22}$$

$$g(x) = c, \tag{13.23}$$

with the objective function $f : D \to \mathbb{R}$, and the *constraints* $g : D \to \mathbb{R}^m$, with $x \in D \subset \mathbb{R}^n$ and $c \in \mathbb{R}^m$.

We define the *Lagrangian* $\mathcal{L} : \mathbb{R}^{n+m} \to \mathbb{R}$, as

$$\mathcal{L}(x, \lambda) = f(x) - \lambda \cdot (g(x) - c), \tag{13.24}$$

with the *dual variables*, or *Lagrangian multipliers*, $\lambda \in \mathbb{R}^m$, from which we obtain the optimality conditions,

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) - \lambda \cdot \nabla g(x) = 0, \tag{13.25}$$

$$\nabla_\lambda \mathcal{L}(x, \lambda) = g(x) - c = 0, \tag{13.26}$$

that is, $n+m$ equations from which we can solve for the unknown variables $(x, \lambda) \in \mathbb{R}^{n+m}$.

Example in $\mathbb{R}^2$

For $f : \mathbb{R}^2 \to \mathbb{R}$, $g : \mathbb{R}^2 \to \mathbb{R}$ and $c = 0$, the Lagrangian takes the form

$$\mathcal{L}(x, \lambda) = f(x) - \lambda g(x), \tag{13.27}$$

with the optimality conditions

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) - \lambda \nabla g(x) = 0, \tag{13.28}$$

$$\nabla_\lambda \mathcal{L}(x, \lambda) = g(x) = 0, \tag{13.29}$$

so that $\nabla f = \lambda \nabla g(x)$, which corresponds to the curve defined by the constraint $g(x) = 0$ being parallel to a level curve of $f(x)$ in $\bar{x} \in \mathbb{R}^2$, the solution to the constrained minimization problem.

13.4 Optimal control

The constrained optimal control problem

We now consider the *constrained optimal control problem*,

$$\min_{\alpha} f(x, \alpha) \tag{13.30}$$

$$g(x, \alpha) = c, \tag{13.31}$$

with the objective function $f : D \times A \to \mathbb{R}$, and the *constraints* $g : D \times A \to \mathbb{R}^m$, with $x \in D \subset \mathbb{R}^n$, $\alpha \in A \subset \mathbb{R}^l$ and $c \in \mathbb{R}^m$.

We define the *Lagrangian* $\mathcal{L} : \mathbb{R}^{n+m} \to \mathbb{R}$, as

$$\mathcal{L}(x, \lambda, \alpha) = f(x, \alpha) - \lambda \cdot (g(x, \alpha) - c), \tag{13.32}$$

with the *dual variables*, or *Lagrangian multipliers*, $\lambda \in \mathbb{R}^m$, from which we obtain the optimality conditions,

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla_x f(x, \alpha) - \lambda \cdot \nabla_x g(x, \alpha) = 0, \tag{13.33}$$

$$\nabla_\lambda \mathcal{L}(x, \lambda) = g(x, \alpha) - c = 0, \tag{13.34}$$

$$\nabla_\alpha \mathcal{L}(x, \lambda) = \nabla_\alpha f(x, \alpha) - \lambda \cdot \nabla_\alpha g(x) = 0, \tag{13.35}$$

that is, $n + m + l$ equations from which we can solve for the unknown variables $(x, \lambda, \alpha) \in \mathbb{R}^{n+m+l}$.

Example

We now consider the *constrained minimization problem*,

$$\min_{x \in D} c^T x \tag{13.36}$$

$$Ax = b, \tag{13.37}$$

with $x, b, c \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n \times n}$.

We define the *Lagrangian* $\mathcal{L} : \mathbb{R}^{2n} \to \mathbb{R}$, as

$$\mathcal{L}(x, \lambda) = c^T x - \lambda^T (Ax - b), \tag{13.38}$$

with $\lambda \in \mathbb{R}^n$, from which we obtain the optimality conditions,

$$\nabla_x \mathcal{L}(x, \lambda) = c + A^T \lambda = 0, \tag{13.39}$$

$$\nabla_\lambda \mathcal{L}(x, \lambda) = Ax - b = 0. \tag{13.40}$$