Chapter 1 Matrix Algebra

The main focus of this chapter is to review some of the basic notions of matrix algebra. These notions will be later applied in the context of matrix computation with a focus on (i) numerical methods to solve and approximate linear equations and (ii) interpolation and approximation of functions by means of polynomials.

1 Introduction

We start this chapter with a number of simple illustrative examples.

Approximate solution of differential equations Consider the following differential equation which is a simplified version of the diffusion equation

$$-u''(x) + cu'(x) + du(x) = f(x), \quad 0 < x < 1\,,$$

with boundary conditions $u(0) = 0 = u(1) = 0$.[1] Depending on the function f, finding an exact solution may or may not be feasible. An alternative is to solve it approximately using a technique known as the finite difference method which works as follows.

We subdivide the interval $[0, 1]$ into n equal subintervals of length $h = 1/n$. We let $x_j = jh$, $j = 0, \ldots, n$ be the subdivision points which constitute our computational grid. The finite difference technique will produce approximations of $u(x_1), \ldots, u(x_n)$ for which we have

$$-u''(x_i) + cu'(x_i) + du(x_i) = f(x_i)\,.$$

If h is small, a standard approximations for the first and second order derivatives are

$$u'(x_i) \approx \frac{u(x_i + h) - u(x_i - h)}{2h} = \frac{u(x_{i+1}) + u(x_{i-1})}{2h}$$

and

$$u''(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}\,.$$

Substituting these approximations into the differential equations, we obtain

$$\frac{-u(x_{i+1}) + 2u(x_i) - u(x_{i-1})}{h^2} + c\left(\frac{u(x_{i+1}) - u(x_{i-1})}{2h}\right) + du(x_i) \approx f(x_i)\,. \tag{1}$$

For $n = 6$ and $h = 1/6$, we can rewrite this as $u(0) = u(6) = 0$ and for $i = 1$ we have

$$\begin{aligned} f(x_1) &\approx 36\left(-u(x_2) + 2u(x_1) - u(x_0)\right) + 3c\left(u(x_2) - u(x_0)\right) + du(x_1) \\ &= (72 + d)u(x_1) + (-36 + 3c)u(x_2) \end{aligned}$$

and similarly we can get the expression for $f(x_2), \ldots f(x_5)$ by making $i = 2, \ldots, 5$ respectively in equation (1). In matrix form this becomes,

$$\begin{pmatrix} 72+d & -36+3c & 0 & 0 & 0 \\ -36+3c & 72+d & -36+3c & 0 & 0 \\ 0 & -36+3c & 72+d & -36+3c & 0 \\ 0 & 0 & -36+3c & 72+d & -36+3c \\ 0 & 0 & 0 & -36+3c & 72+d \end{pmatrix} \begin{pmatrix} u(x_1) \\ u(x_2) \\ u(x_3) \\ u(x_4) \\ u(x_5) \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \\ f(x_5) \end{pmatrix}$$

[1] $-u''(x)$ is a diffusion term, $cu'(x)$ is a convection term, $du(x)$ is a decay term, and $f(x)$ is a source term.

Data fitting In a data fitting problem, we are in general given n data points (t_i, y_i) where y_i and t_i are some real numbers that represent the value observed (e.g. position of a vehicle, temperature, price) and t_i are time instances where the data have been measured. The goal is to find a function $g(t)$ such that

$$g(t_i) \approx y_i, \quad i = 1, \dots, n \, .$$

To start with we could try to fit the data with a line, i.e. we want to find two scalars a and b such that $y_i \approx a + bt_i$ that give a good fit with the available data. Let us assume that we want to minimise the following error function

$$\sum_{i=1}^{n} (a + bt_i - y_i)^2$$

that accounts for the cumulative error (distance) we are making on each data point. We can rewrite our problem in matrix form as follows. Let $x = \begin{pmatrix} a \\ b \end{pmatrix}$, $A = \begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_n \end{pmatrix}$ and $y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$. Then the problem reduces to minimising

$$||Ax - y||^2 = (Ax - y)^T (Ax - y) = \sum_{i=1}^{n} (a + bt_i - y_i)^2 \, .$$

This is known as the *least-squares problem*, and the solution to this problem is given by the solution to the following linear equation

$$A^T A x = A^T y \, .$$

We could also attempt to fit a more elaborate function g to the data for example

$$g(t) = x_0 + x_1 t + \cdots + x_{n-1} t^{n-1} \, .$$

The aim now is to determine the coefficients $x_0, \dots, x_{n-1}$ which can be again expressed as linear equation with $A = \begin{pmatrix} 1 & t_1 & t_1^2 & \dots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \dots & t_2^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & t_n & t_n^2 & \dots & t_n^{n-1} \end{pmatrix}$ and $y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$. This is known as the *polynomial interpolation* of the data set (t_i, y_i). Provided that all the interpolants are distinct one can show that this matrix is invertible. In fact, the matrix A is known as the Vandermonde matrix, and its determinant is given by

$$\det(A) = \prod_{i=0}^{n-1} \prod_{j=i+1}^{n-1} (t_j - t_i) \, .$$

Optimal linear control and least-norm problem We consider a simple problem from mechanics. A unit mass, moving along a straight line, is subjected to a piecewise-constant force (force plan or programme) $F(t)$ such that in time interval i we have $F(t) = x_i$, $i - 1 \leq t < i$, $i = 1, \dots, n$.

We denote by $y(t)$ and $y'(t)$ its position and velocity at time t and we assume that $y(0) = y'(0) = 0$. We are interested in the velocity and position at time $t = 10$, say.

By Newton's law we know that $y''(t) = F(t)$ and so

$$y'(10) = \int_0^{10} F(u) du = x_1 + \cdots + x_{10}$$

and

$$y(10) = \int_0^{10} \int_0^u F(v) dv du = \frac{19}{2} x_1 + \frac{17}{2} x_2 + \cdots + \frac{1}{2} x_{10} \, .$$

In matrix form we have

$$\begin{pmatrix} y'(10) \\ y(10) \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ 19/2 & 17/2 & \dots & 3/2 & 1/2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Assume that we want to choose an efficient force $F(t)$ that uses minimum energy and moves the mass in 10 steps to position $y(10) = 1$ with final velocity $y'(10) = 0$. The total energy of the force is proportional to

$$\int_0^{10} F(t)^2 = \sum_{i=1} x_i^2 .$$

The control problem consists of minimising $x_1^2 + \cdots + x_{10}^2$ subject to $Ax = y$. This is known as the *least-norm problem.*

The optimal force is given by $A^T z$ where z is the solution to the equation $AA^T z = y$.

In the remainder of the chapter, we will first review matrix operations and examine their complexity. We will then explore some important notions of algebra such the range, the null-space of a linear operator as the well as the notions of span, linear independence of a family of vectors and bases. We will define the concept of orthogonality of vectors and subspaces together with the notion of direct sum and complements. We will briefly touch on solving linear systems that will be analysed more extensively in the next chapter.

2 Matrix Operations and their Complexity

We define the cost of a matrix operation in terms of the number of operations (addition, subtraction, multiplication, division) required to carry it out. To evaluate the complexity of an algorithm, we enumerate the total number of operations expressed as a function of the dimensions of the matrices and vectors involved, keeping track of the leading (dominant) terms. This is not however an accurate predictor of computation time on modern computers (issues of cache boundaries and locality of reference can impact the computation time of numerical algorithms). Nonetheless it still gives a useful rough estimate of the computation time of such algorithms.

Vector operations To compute the *scalar or inner product* of two n-dimensional vectors x and y, we form the products $x_i y_i$ for $i = 1, \dots, n$, which requires n multiplications, and add them, requiring $n-1$ additions, i.e., $2n-1$ operations. Keeping track of the leading term, for n large , we say that the inner product of x and y can be computed in $2n$ operations or $O(n)$ operations.

Matrix-vector multiplication Let $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$. The matrix vector multiplication Ax requires $2mn$ operations: compute the m components of the vector y, each obtained by performing the inner product x with a row of A ($2n$ operations). When $m = n$ we require $2n^2$ operations. However if A is diagonal in $\mathbb{R}^{n\times n}$, then Ax can be computed in $O(n)$ operations instead of $2n^2$ operations or $O(n^2)$ operations..

Suppose that $A = UV$ where $U \in \mathbb{R}^{m\times p}$ and $V \in \mathbb{R}^{p\times n}$. Then we can compute Ax by first computing Vx at the cost of $2pn$ operations, and then $U(Vx)$ at the cost of $2pm$ operations, so the total is $2p(m+n)$ operations. If p is small in comparison to m and n, then $2p(m+n)$ will be also cheaper than $2mn$.

Matrix-matrix multiplication To perform the previous multiplication, we can start by computing the matrix product $A = UV$, and this cost $2mnp$: we have mn elements in A each can be computed by performing an inner product of two p dimensional vectors.

Example Derive the number of operations required to evaluate the product of three matrices $A\,B\,C$ where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 10}$ and $C \in \mathbb{R}^{10\times n}$. For the following two methods evaluate the number of operations required (i) from left to right: $(AB)C$ and (ii) from right to left: $A(BC)$. Which method is faster for large n?

3 Matrix Analysis

For m, n two non-negative integers, let $A \in \mathbb{R}^{m\times n}$ be a matrix with m rows and n columns whose entries a_{ij}, $i = 1, \dots, m$, $j = 1, \dots, n$ are real-valued.

A vector $y \in \mathbb{R}^m$ is in the *image or range (aka column space)* of A, denoted by $\mathrm{Ran}(A)$, if there exists a vector $x \in \mathbb{R}^n$ such that $Ax = y$.

It is not difficult to see that $\mathrm{Ran}(A)$ is a *subspace* of $\mathbb{R}^m$, since if y and y' are in $\mathrm{Ran}(A)$ then so are the vectors $y + y'$ and λy for $\lambda \in \mathbb{R}$.

Remember that if $Ax = y$, then y_i the i-th coordinate of y is given by, for $i = 1, \dots, m$

$$y_i = \sum_{j=1}^{n} a_{ij}x_j = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \,.$$

In particular, if a_j denotes the j-th column of the matrix A then

$$y = x_1a_1 + x_2a_2 + \cdots + x_na_n \,.$$

This explains why the range of A is also referred to as the column space of A. In fact, the range of A is the subspace of all *linear combinations* of the columns of A, i.e., for $a_1, \dots, a_n$ the columns of A, we denote $A = [a_1, \dots, a_n]$,

$$\mathrm{Ran}(A) = \{x_1a_1 + x_2A_2 + \cdots + x_na_n, \text{ where } x_1, \dots, x_n \in \mathbb{R}\} \,.$$

We say that the vectors $a_1, \dots, a_n$ *span* the subspace $\mathrm{Ran}(A)$ and denote it as

$$\mathrm{Ran}(A) = \mathrm{Span}(a_1, \dots, a_n) = \{x_1a_1 + x_2A_2 + \cdots + x_na_n, \text{ where } x_1, \dots, x_n \in \mathbb{R}\} \,.$$

More generally the span of a family of vectors is the set of all linear combinations of these vectors.

Another important space is $\mathrm{Ker}(A)$ the *kernel or null-space* of A which is the subspace of all vectors whose image is equal to the vector $\mathbf{0}$ whose entries are all equal to 0, i.e.

$$\mathrm{Ker}(A) = \{x \in \mathbb{R}^n, \text{ such that}, Ax = \mathbf{0}\} \,.$$

The rank of the matrix A is the dimension of the $\mathrm{Ran}(A)$. The dimension of a subspace U accounts for the number of *linearly independent* vectors that span that space, where a family of vectors $u_1, \dots, u_k$ are linearly independent if none of them can be expressed as a linear combination of the other vectors. More precisely, if $u_1, \dots, u_k$ are linearly independent vectors then

$$\text{if for real numbers (scalars) } \lambda_1, \dots, \lambda_k \text{ we have } \textstyle\sum_{i=1}^{k} \lambda_i u_i = \mathbf{0} \text{ then it must be that } \lambda_1 = \cdots = \lambda_k = 0 \,.$$

In particular if (i) $U = \mathrm{Span}(u_1, \dots, u_k)$ and (ii) $u_1, \dots, u_k$ are linearly independent then $u_1, \dots, u_k$ is said to be a *basis* of the subspace U.

Example We let $n = m = 3$ and define the matrix

$$A = \begin{pmatrix} 4 & -1 & 5 \\ -2 & -1 & -1 \\ -4 & 1 & -5 \end{pmatrix} \,.$$

Derive the range and null-spade of A and determine a basis for each of them.

Definition 1 (Direct sums and Complements) *Let V_1 and V_2 two subspaces of $\mathbb{R}^n$. We say that V_1 and V_2 form a* direct sum *if we have $V_1 \cap V_2 = \{\mathbf{0}\}$.*

If, in addition, $V_1 + V_2 = \mathbb{R}^n$, we say that V_1 and V_2 are complementary *and we write*

$$\mathbb{R}^n = V_1 \oplus V_2 \, .$$

In this case, each vector v in $\mathbb{R}^n$ can be uniquely written as $x = v_1 + v_2$, $v_1 \in V_1$ and $v_2 \in V_2$.

Example Back to the previous numerical example.

- Do we have $\mathrm{Ran}(A) \oplus \mathrm{Ker}(A) = \mathbb{R}^3$?
- Let $A^2 = A \times A$, derive the range and null-spade of A^2 and determine a basis for each of them.
- Show that $\mathrm{Ran}(A^2) \oplus \mathrm{Ker}(A^2) = \mathbb{R}^3$.

Example: Projectors A projector P is a linear transformation such that $P^2 = P$. In this example we explore some of the properties of such matrices.

The range of P is given by the vectors y for which there exists a vector x such that $y = Px$, in particular

$$Py = P^2x = Px = y$$

Therefore if P is a projector then

$$\mathrm{Rang}(P) = \{y \mid Py = y\} \, .$$

We now show that if P is projector from $\mathbb{R}^n$ to $\mathbb{R}^n$ then

$$\mathrm{Ran}(P) \oplus \mathrm{Ker}(P) = \mathbb{R}^n \, .$$

To do this we need to prove the following two properties

- if $x \in \mathrm{Ran}(P) \cap \mathrm{Ker}(P)$ then $x = \mathbf{0}$.

  Since $x \in \mathrm{Ran}(P)$ and P is a projector then by the previous property $Px = x$ yet $x \in \mathrm{Ker}(P)$ so $Px = \mathbf{0}$ so $x = \mathbf{0}$.

- for every $x \in \mathbb{R}^n$, there exist $y \in \mathrm{Ran}(P)$ and $z \in \mathrm{Ker}(P)$ such that $x = y + z$. In fact we can write

  $$x = (x - Px) + Px$$

  with $Px \in \mathrm{Ran}(P)$ and

  $$P(x - Px) = Px - P^2x = Px - Px = 0$$

  such that $x - Px \in \mathrm{Ker}(P)$.

4 Orthogonality

The transpose of a vector x or a matrix A are denoted by x^T and A^T respectively. We say that two vectors x and y in $\mathbb{R}^n$ are *orthogonal* if $x^Ty = \sum_{i=1}^n x_i y_i = 0$. We denote this by $x \perp y$.

Similarly, two subspaces U and $V$are *orthogonal*, denoted $U \perp V$, if for all $u \in U$ and all $v \in V$ we have that $u^Tv = 0$.

As a matter of example, let $A \in \mathbb{R}^{m \times n}$ then

$$\mathrm{Ran}(A) \perp \mathrm{Ker}(A^T) \quad \text{and} \quad \mathrm{Ran}(A^T) \perp \mathrm{Ker}(A)$$

In fact if $y \in \text{Ran}(A)$ then there exists a vector y such that $Ax = y$ and so for $z \in \text{Ker}(A^T)$ we have

$$y^T z = (Ax)^T z = x^T (A^T z) = \mathbf{0} .$$

We now define the subspace $U^\perp$, the *orthogonal subspace* of a subspace U, as the set of all vectors that are orthogonal to the subspace U, i.e.

$$U^\perp = \{v \in \mathbb{R}^n \text{ such that } v^T u = 0 \text{ for all } u \in U\} .$$

In particular

Theorem *For every subspace U of $\mathbb{R}^n$ we have that*

$$U \oplus U^\perp = \mathbb{R}^n .$$

Proof To prove the theorem we will define a projector P such that $\text{Ran}(P) = U$ and $\text{Ker}(P) = U^\perp$ and use the fact that

$$\text{Ran}(P) \oplus \text{Ker}(P) = \mathbb{R}^n .$$

For a subspace U of dimension k we can define a basis of k, say $u_1, \dots, u_k$ such that

$$U = \text{Span}\{u_1, \dots, u_k\} .$$

In fact, thanks to a technique known as Gramm-Schmidt orthogonalisation that we will see in the next chapter, we can define an orthonormal basis, i.e. a basis $u_1, \dots, u_k$ of U such that

$$u_i^T u_i = 1 \quad \text{and} \quad u_i^T u_j = 0 \text{ if } i \neq j .$$

We now introduce the matrix

$$P = \sum_{i=1}^{k} u_i u_i^T$$

It is not difficult to see that P has the following properties

- $P^T = P$, we say that P is *symmetric*. In fact,

$$P^T = \left(\sum_{i=1}^{k} u_i u_i^T \right)^T = \sum_{i=1}^{k} \left(u_i u_i^T \right)^T = \sum_{i=1}^{k} u_i u_i^T .$$

- P is a projector, i.e. $P^2 = P$. In fact,

$$P^2 = \sum_{i=1}^{k} u_i u_i^T \sum_{j=1}^{k} u_j u_j^T = \sum_{i=1}^{k} \sum_{j=1}^{k} u_i (u_i^T u_j) u_j^T = \sum_{j=1}^{k} u_i u_i^T$$

  where we used the fact that the vectors $u_1, \dots, u_k$ form an orthonormal basis.

In the remainder of the proof we will show that 1. $\text{Ran}(P) = U$ and 2. $\text{Ker}(P) = U^\perp$.

1. To show that $\text{Ran}(P) = U$ we show that (a) $\text{Ran}(P) \subset U$ and (b) $U \subset \text{Ran}(P)$

(a) Let $y \in \text{Ran}(P)$ then $\exists\ x$ such that

$$y = Px = \sum_{i=1}^{k} u_i(u_i^T x) \in \text{Span}\{u_1, \ldots, u_k\} = U$$

which implies that $\text{Ran}(P) \subset U$.

(b) Let $x \in U$ then $x = \sum_{i=1}^{k} x_i u_i$ where $x_i \in \mathbb{R}$ for all $i = 1, \ldots, k$

$$Px = \sum_{j=1}^{k} u_j u_j^T \sum_{i=1}^{k} x_i u_i = \sum_{j=1}^{k}\sum_{i=1}^{k} x_i u_j u_j^T u_i = \sum_{i=1}^{k} x_i u_i = x$$

since $u_1, \ldots, u_k$ is an orthonormal basis. Recall that if P is a projector then $\text{Ran}(P) = \{x \mid Px = x\}$ so $x = \sum_{i=1}^{k} x_i u_i$ is in $\text{Ran}(P)$. Thus $U \subset \text{Ran}(P)$.

2. We now focus on the null-space of P and show that $\text{Ker}(P) = U^\perp$. To this end we show that (a) $\text{Ker}(P) \subset U^\perp$ and (b) $U^\perp \subset \text{Ker}(P)$

(a) If $x \in \text{Ker}(P)$ then $Px = 0$. For $z \in U = \text{Ran}(P)$ we know that $Pz = z$, and so

$$z^T x = (Pz)^T x = z^T P^T x = z^T Px = 0$$

since we know from the properties of the projector $P = \sum_{i=1}^{k} u_i u_i^T$ that $P^T = P$. Hence $\text{Ker}(P) \subset U^\perp$.

(b) Now let $x \in U^\perp$, for every $y \in \mathbb{R}^n$ we have $(Py)^T x = 0$ since

$$Py \in \text{Ran}(P) = U \quad \Rightarrow \quad \forall x \in U^\perp,\ (Py)^T x = 0$$

In particular $0 = y^T P^T x = y^T Px$ and this is true for all $y \in \mathbb{R}^n$. It is not difficult to convince yourself that the only vector that is orthogonal to all other vectors is the vector $\mathbf{0}$ so

$$Px = \mathbf{0} \quad \Rightarrow \quad x \in \text{Ker}(P)$$

and $U^\perp \subset \text{Ker}(P)$

which completes the proof.

QED

A consequence of this result is the *fundamental theorem of algebra* that goes as follows

Theorem For $A \in \mathbb{R}^{m \times n}$, we have

$$\text{Ran}(A) \oplus \text{Ker}(A^T) = \mathbb{R}^m\ .$$

Proof We will show that $\text{Ran}(A)^\perp = \text{Ker}(A^T)$.

1. Let $x \in \text{Ker}(A^T)$ and let $z = Ay \in \text{Ran}(A)$ then

$$z^T x = y^T A^T x = 0 \quad \Rightarrow \quad x \in \text{Ran}(A)^\perp\ .$$

Hence $\text{Ker}(A^T) \subset \text{Ran}(A)^\perp$.

2. Let $x \in \text{Ran}(A)^\perp$ then for all $y \in \mathbb{R}^n$, $(Ay)^T x = 0$ since $Ay \in \text{Ran}(A)$. In particular,

$$y^T(A^T x) = 0,\ \forall y \in \mathbb{R}^n \quad \Rightarrow \quad A^T x = \mathbf{0} \quad \Rightarrow \quad x \in \text{Ker}(A^T)\ .$$

Hence $\text{Ran}(A)^\perp \subset \text{Ker}(A^T)$.

which completes the proof.

QED

Remark Note that it is easy to see that $\text{Ran}(A) = \text{Ker}(A^T)^\perp$, $\text{Ran}(A^T) = \text{Ker}(A)^\perp$ and $\text{Ran}(A^T)^\perp = \text{Ker}(A)$. In addition, for $A \in \mathbb{R}^{m\times n}$, we have

$$\text{Ran}(A^T) \oplus \text{Ker}(A) = \mathbb{R}^n \, .$$

Example Try this out on the following matrix

$$A = \begin{pmatrix} 4 & -1 & 5 \\ -2 & -1 & -1 \\ -4 & 1 & -5 \end{pmatrix} .$$

Example Let us go back to the minimum least squares problem: we want to find a vector $x \in \mathbb{R}^n$ that minimises $||Ax-y||^2 = (Ax-y)^T(Ax-y)$. By the fundamental theorem of linear algebra we have $y = y_R+y_N$ where $y_R \in \text{Ran}(A)$ and $y_N \in \text{Ker}(A^T)$ and such that $y_R^T y_N = 0$ since $\text{Ran}(A) \perp \text{Ker}(A^T)$.

We define the residual error as

$$r = y - Ax = (y_R - Ax) + y_N \, .$$

Note that $y_R - Ax \in \text{Ran}(A)$ so that $(y_R - Ax)^T y_N = 0$

$$r^T r = [(y_R - Ax) + y_N]^T[(y_R - Ax) + y_N] = (y_R - Ax)^T(y_R - Ax) + y_N^T y_N$$

this is known as *Pythagorean theorem.*

Therefore minimising $||Ax - y||$ reduces to minimising $||y_R - Ax||$. Since $y_R \in \text{Ran}(A)$ then there exists a vector x^* such that $Ax^* = y_R$ and this vector is the minimiser of $||y_R - Ax||$. This equivalent to saying that

$$y - Ax^* = y_N \quad \Rightarrow \quad A^T(Ax^* - y) = A^T y_N \quad \Rightarrow \quad A^T A x^* = A^T y \, .$$

To sum up, to solve the minimum least squares problem (find a vector $x \in \mathbb{R}^n$ that minimises $||Ax - y||^2 = (Ax - y)^T(Ax - y)$) we need to solve the equation $A^T Ax = A^T y$. We will return to this problem in the next chapter.

5 Solving Linear Equations

The subspaces $\text{Ker}(A)$ and $\text{Ran}(A)$ are relevant in the context of solving linear equations of the form $Ax = y$ where we want to find $x \in \mathbb{R}^m$ given $A \in \mathbb{R}^{m\times n}$ and $y \in \mathbb{R}^n$ which happens in a number of scenarios as illustrated in section 1.

The equation $Ax = y$ is called *overdetermined* if $m > n$, *underdetermined* if $m < n$, and *square* if $m = n$. If y is the vector $\mathbf{0}$ it is called *homogeneous.*

We now give some conditions for the existence and uniqueness of solutions to a given linear system.

Proposition 1 *Given a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $y \in \mathbb{R}^n$*

- *The linear equation $Ax = y$ is solvable if and only if*

$$y \in Ran(A) = \{Ax,\ x \in \mathbb{R}^n\} \, ,$$

- *If the kernel or null space of A*

$$Ker(A) = \{z \in \mathbb{R}^n,\ Az = \mathbf{0}\} = \{\mathbf{0}\}$$

  then there is at most one solution.

The matrix A is *full-range* if $\text{Ran}(A) = \mathbb{R}^m$ $(n \geq m)$ and A has *zero null-space* if $\text{Ker}(A) = \{\mathbf{0}\}$ $(m \geq n)$. When $m = n$, the notions of full-range and zero null space are equivalent and A is said to be *non-singular or invertible.*

There are many sets of special matrices for which one can solve linear equations fairly easily. We will focus on *square matrices*, i.e., $m = n$.

Diagonal matrices If A is non-singular diagonal matrix, i.e., $a_{ij} = 0$ if $i \neq j$ and $a_{ii} \neq 0$, $i, j = 1, \ldots, n$, we only require n operations to solve the corresponding linear system and the solution is given by

$$x = \begin{pmatrix} y_1/a_{11} \\ y_2/a_{22} \\ \vdots \\ y_n/a_{nn} \end{pmatrix} .$$

Triangular matrices If A is non-singular lower triangular, i.e., $a_{ij} = 0$, $j > i$ and $a_{ii} \neq 0$, $i, j = 1, \ldots, n$, we require $O(n^2)$ operations to solve the corresponding linear system and the solution is given by

$$\begin{aligned} x_1 &= y_1/a_{11} \\ x_2 &= (y_2 - a_{21}x_1)/a_{22} \\ &\vdots \\ x_n &= (y_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1})/a_{nn} . \end{aligned}$$

The above operations are known as *forward substitution*.

Pseudocode: Pseudocode for the forward substitution

1. **input** n, a_{ij}, y_i
2. **for** $i = 1$ to n **do**
3. **for** $j = 1$ to $i - 1$ **do** $y_i \leftarrow y_i - a_{ij}x_j$ **end do**
4. $x_i \leftarrow y_i/a_{ii}$
5. **end do**
6. **output** x_i

Let us compute the number of operations involved. It is not difficult to see that in step i there are $2i - 1$ operations involved. The total number of operations is

$$\sum_{i=1}^{n}(2i - 1) = n(n + 1) - n = n^2 .$$

One can proceed similarly for upper triangular matrices with a slightly different expression of the solution.

Orthogonal matrices If A is an orthogonal matrix, i.e, $A^TA = AA^T = I$, we require $2n^2$ operations to solve the corresponding linear system by computing the matrix vector product $x = A^Ty$.

6 Gaussian elimination

The standard technique for solving linear equations is by means of *Gaussian elimination*: To solve $Ax = y$ for a general matrix $A \in \mathbb{R}^{n\times n}$, one first transforms the problem into solving $\tilde{A}x = \tilde{y}$ where $\tilde{A}$ is an upper triangular matrix and then solves the system for an upper triangular matrix as described in the previous chapter. The first part of the algorithm goes as follows.

Pseudocode: Pivoting phase of Gaussian elimination

1. **input** n, a_{ij}, y_i
2. **for** $j = 1$ to $n - 1$

3. **if** $a_{jj} = 0$ **then error**
4. **else for** $i = j + 1$ to n
5. **do** $m \leftarrow a_{ij}/a_{jj}$, $y_i \leftarrow y_i - m \times y_j$, $a_{jj} \leftarrow 0$,
6. **for** $k = j + 1$ to n **do** $a_{ik} \leftarrow a_{ik} - m \times a_{jk}$ **end do**
7. **end do**
8. **end if**
9. **output** a_{ij}, y_i

It is not difficult to see that the above algorithm performs $2n^3/3$ operations. In fact, line 5 accounts for 3 operations and line 6 of the pseudocode accounts for $2(n-j)$ operations. Then for a fixed j the loop in line 4 takes $(n-j)(2n-2j+3)$ operations. The total number of operations is therefore

$$\sum_{j=1}^{n-1}(n-j)(2n-2j+3) = 2\sum_{j=1}^{n-1}(n-j)^2 + 3\sum_{j=1}^{n-1}(n-j) = 2\frac{(n-1)n(2n-1)}{6} + 3\frac{n(n-1)}{2} = \frac{2n^3}{3} + \dots .$$

Observe that the pivoting step is relatively expensive compared to the second step consisting of solving a linear system with a triangular matrix. Overall the complexity of the Gaussian elimination algorithm is $O(n^3)$.

For a given matrix, to solve the following linear equations $Ax_1 = y_1$, $Ax_2 = y_2, \dots, Ax_k = y_k$ we need to perform a Gaussian elimination per equation. This is for example the case if we want to invert a matrix $A \in \mathbb{R}^{n\times n}$ where we need solve the following system of linear equations

$$Ax_1 = e_1,\ Ax_2 = e_2, \dots, Ax_n = e_n\ ,$$

where $e_i \in \mathbb{R}^n$ is the vector where all the entries are equal to 0 except the i-th entry which is equal to 1. Subsequently the inverse of the matrix A if it exists is given by

$$A^{-1} = [x_1, \dots, x_n]\ ,$$

i.e. the i-th column of A^{-1} is the solution to the equation $Ax_i = e_i$. If we use the Gaussian elimination for each of these n linear equation we will need to perform $O(n^4)$ operations. This is in fact the standard procedure we use to invert matrices.

A more efficient way would be to perform the pivoting once and then apply the algorithm for each linear equation to solve them. This idea known as *factor-solve* consists of starting with a factorisation step wherein the matrix A is expressed as a product of a small number (2 to 4) of matrices with special properties:

$$A = A_1 A_2 \dots A_l\ .$$

We then solve the equations of the form $A_1 A_2 \dots A_l x = y$ recursively. In fact, a slight modification of the Gaussian elimination algorithm enables to derive the LU factorisation of A as the product of lower triangular matrix L and an upper triangular matrix U. This decomposition have to be computed only once and to solve an equation $Ax = y$ we first solve $Lz = y$ and then $Ux = z$ each of which requires $O(n^2)$ operations.

7 LU decomposition or Gauss elimination

Let A be non-singular matrix in $\mathbb{R}^{n\times n}$ where

$$A = \begin{pmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

where a_{11} is a scalar, $A_{21} \in \mathbb{R}^{n-1}$, and $A_{22} \in \mathbb{R}^{(n-1)\times(n-1)}$ and $A_{12} \in \mathbb{R}^{1\times(n-1)}$, and define the matrix $C \in \mathbb{R}^{(n-1)\times(n-1)}$, the *Schur complement* of a_{11} in A, i.e.,

$$C = A_{22} - \frac{1}{a_{11}} A_{21} A_{12}\ .$$

Proposition 2 *If A is non-singular, then*

- *in each column of A there is a non-zero entry,*
- *if $a_{11} \neq 0$ then the matrix C is non-singular.*

The LU decomposition (with pivoting) decomposes a non-singular matrix A in terms of the product PLU where P is a permutation matrix, L a lower triangular matrix and U an upper triangular one. We examine the existence of such decomposition proceeding by induction.

Let us first assume that $n = 1$ then A is equal to the scalar $a_{11} \neq 0$ and by taking $P = 1$, $L = 1$ and $U = A$ we have a trivial LU decomposition of A. Next we show that if it is true that every non-singular matrix in $\mathbb{R}^{(n-1)\times(n-1)}$ has an LU factorisation, then the same is true for non-singular matrices in $\mathbb{R}^{n\times n}$. By induction, this proves that any non-singular matrix has an LU decomposition.

As one of the entries of the first column of A must be non-zero (Proposition 2) we can find a matrix P_1 such that $\tilde{A} = P_1^T A$ is such that $\tilde{a}_{11}$ is non-zero. The matrix is trivially non-singular and we can write

$$\tilde{A} = \begin{pmatrix} \tilde{a}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{pmatrix}$$

and define its Schur complement $\tilde{C}$ which is non-singular (Proposition 2)

$$\tilde{C} = \tilde{A}_{22} - \frac{1}{\tilde{a}_{11}} \tilde{A}_{21}\tilde{A}_{12} \, .$$

By assumption this can be factored $\tilde{C} = P_2 L_2 U_2$. Hence going back to A we have

$$\begin{aligned} A &= P_1 \begin{pmatrix} \tilde{a}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{pmatrix} \\ &= P_1 \begin{pmatrix} 1 & 0 \\ 0 & P_2 \end{pmatrix} \begin{pmatrix} \tilde{a}_{11} & \tilde{A}_{12} \\ P_2^T \tilde{A}_{21} & P_2^T \tilde{A}_{22} \end{pmatrix} \\ &= P_1 \begin{pmatrix} 1 & 0 \\ 0 & P_2 \end{pmatrix} \begin{pmatrix} \tilde{a}_{11} & \tilde{A}_{12} \\ P_2^T \tilde{A}_{21} & L_2 U_2 + \frac{1}{\tilde{a}_{11}} P_2^T \tilde{A}_{21}\tilde{A}_{12} \end{pmatrix} \\ &= P_1 \begin{pmatrix} 1 & 0 \\ 0 & P_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{1}{\tilde{a}_{11}} P_2^T \tilde{A}_{21} & L_2 \end{pmatrix} \begin{pmatrix} \tilde{a}_{11} & \tilde{A}_{12} \\ 0 & U_2 \end{pmatrix} = PLU \, . \end{aligned}$$

LU factorisation We now describe an algorithm for computing the LU factorisation of a non-singular matrix A. Using the above decomposition of A we perform the following recursive algorithm,

Pseudocode: LU factorisation

1. **input** $A \in \mathbb{R}^{n\times n}$
2. Choose a permutation P_1 such that $\tilde{A} = P_1^T A$ satisfies $\tilde{a}_{11} \neq 0$
3. Compute the LU factorization of $\tilde{A}_{22} - \frac{1}{\tilde{a}_{11}} \tilde{A}_{21}\tilde{A}_{12} = P_2 L_2 U_2$
4. Return (P, L, U) as the LU factorization of A with

$$P = P_1 \begin{pmatrix} 1 & 0 \\ 0 & P_2 \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 \\ \frac{1}{\tilde{a}_{11}} P_2^T \tilde{A}_{21} & L_2 \end{pmatrix}, \quad U = \begin{pmatrix} \tilde{a}_{11} & \tilde{A}_{12} \\ 0 & U_2 \end{pmatrix} .$$

This algorithm requires $\frac{2}{3}n^3$ operations.

To the solve the equation $Ax = y$, we first perform the LU decomposition and then solve the equation $PLUx = y$ as follows:

We first compute the LU factorisation $A = LU$, ($\frac{2}{3}n^3$ operations) and then

Pseudocode: Solving linear equations by LU factorisation

1. **input** $P, L, U \in \mathbb{R}^{n\times n}$ and $y \in \mathbb{R}^n$
2. Solve $PLUx = y$ as follows
3. Solve $Px_1 = y$ at 0 cost
4. Solve $Lx_2 = z$ at the cost of $O(n^2)$ operations
5. Solve $Ux = x_2$ at the cost of $O(n^2)$ operations
6. Return x.

Example Use the LU factorisation to solve the equation

$$\begin{pmatrix} 0 & 5 & 5 \\ 2 & 9 & 0 \\ 6 & 8 & 8 \end{pmatrix} x = \begin{pmatrix} 15 \\ 7 \\ 18 \end{pmatrix}.$$

Now to solve k disctinct linear equations we first apply the LU-decomposition at the cost of $O(n^3)$ and $k(n^2)$ operations to solve the systems given that $A = PLU$. In particular when $k = n$ as in the case of matrix inversions we perform $O(n^3)$ instead of $O(n^4)$ operations using the standard Gaussian elimination.

8 Exercises

Exercise 1 Compute the n-th power of the following matrices

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \quad D = \begin{pmatrix} a & a-b \\ 0 & b \end{pmatrix}$$

Exercise 2 Let A be a two-by-two real matrix. Solve the following equation $A^2 = A$.

Exercise 3 Let $M = \begin{pmatrix} 1 & 3 \\ 0 & 2 \end{pmatrix}$. Show that there exists P and Q two 2-by-2 matrices such that

$$M = P \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} Q$$

Compute the n-th power of M. Similarly for $M = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$.

Exercise 4 Give the expression of u_n, v_n and w_n such that $u_0 = v_0 = w_0 = 1$ and

$$\begin{aligned} u_{n+1} &= 5v_n + 6w_n \\ v_{n+1} &= 5u_n + 3v_n \\ w_{n+1} &= 6u_n + 2w_n \,. \end{aligned}$$

Exercise 5 Solve the following system of equations

$$\begin{aligned} x + y + z &= a \\ x + jy + j^2 z &= b \\ x + j^2 y + jz &= c \,, \end{aligned}$$

where $j = e^{2\pi i/3}$.

Exercise 6 Compute the inverses of the following matrices

$$A = \begin{pmatrix} 1 & a & a^2 & a^3 & a^4 \\ 0 & 1 & a & a^2 & a^3 \\ 0 & 0 & 1 & a & a^2 \\ 0 & 0 & 0 & 1 & a \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Exercise 7 Find determinant, eigenvalues and eigenvectors of

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Exercise 8 Let $A = (a_{ij})_{i,j=1,\ldots,n}$ a square matrix of order n, compute $tr(AA^T)$, tr stands for trace and A^T is the transpose of A.

Exercise 9 Derive the number of operations required to evaluate the product of three matrices $A\,B\,C$ where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 10}$ and $C \in \mathbb{R}^{10\times n}$. For the following two methods evaluate the number of operations required (i) from left to right: $(AB)C$ and (ii) from right to left: $A(BC)$. Which method is faster for large n?

Exercise 9 (2011) For a matrix A in $\mathbb{R}^{n\times n}$, we define the k-th power of A as $A^k = A^{k-1} \times A = A \times A^{k-1}$, for $k \geq 1$ and $A^0 = I$ the identity matrix. We denote by $\mathrm{Ran}(A)$ and $\mathrm{Ker}(A)$ the range (or image) and the nullspace (or kernel) of A, respectively.

We say that two subspaces V and W of $\mathbb{R}^n$ are complementary, denoted by $V \oplus W = \mathbb{R}^n$, if (i) $V \cap W = \{\mathbf{0}\}$, where $\mathbf{0}$ is the zero vector in $\mathbb{R}^n$, and (ii) any vector $x \in \mathbb{R}^n$ can be written as $x = v + w$ where $v \in V$ and $w \in W$.

1. We let $n = 3$ and define the matrix A by

$$A = \begin{pmatrix} 4 & -1 & 5 \\ -2 & -1 & -1 \\ -4 & 1 & -5 \end{pmatrix}.$$

   (a) Derive $\mathrm{Ran}(A)$ and $\mathrm{Ker}(A)$ and determine a basis for each of them.

   (b) Do we have $\mathrm{Ran}(A) \oplus \mathrm{Ker}(A) = \mathbb{R}^3$? Justify your answer.

   (c) Let $A^2 = A \times A$. Derive $\mathrm{Ran}(A^2)$ and $\mathrm{Ker}(A^2)$ and determine a basis for each of them.

   (d) Show that $\mathrm{Ran}(A^2) \oplus \mathrm{Ker}(A^2) = \mathbb{R}^3$.

2. We now let $n = 4$ and define the matrix A_m as follows

$$A_m = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & m & 0 & 0 \\ 1 & 0 & -m & -1 \\ 0 & 1 & 0 & 0 \end{pmatrix},$$

   where $m \in \mathbb{R}$ is a parameter.

   (a) Derive bases for $\mathrm{Ker}(A_m)$ and $\mathrm{Ran}(A_m)$.

   (b) For $m \neq 0$, show that $\mathrm{Ran}(A_m) \oplus \mathrm{Ker}(A_m) = \mathbb{R}^4$.

   (c) We now fix $m = 0$. Compute A_0^3.
   Do we have $\mathrm{Ran}(A_0^3) \oplus \mathrm{Ker}(A_0^3) = \mathbb{R}^4$?
   Justify your answer.

1. We define the following property

$$\text{For } A \in \mathbb{R}^{n\times n}, \text{ there exists an integer } p \geq 1 \text{ such that } \mathrm{Ran}(A^p) \oplus \mathrm{Ker}(A^p) = \mathbb{R}^n\,, \quad (\star)$$

   (a) Let A be a non-singular (invertible) matrix. Find p such that the property $(\star)$ is satisfied for A. Justify your answer.

   (b) Let A be a projection. Find a value p such that $(\star)$ is satisfied.
   Explain your answer. A formal proof is not required.

2. In fact, the property $(\star)$ is satisfied for any matrix A. Try to figure out why this is true.[2]

[2] Hard question ommited in the exam.

Exercise 10 (2009) Let $u_1, \ldots, u_n$ be a set of orthonormal vectors in $\mathbb{R}^n$, i.e., pairwise orthogonal

$$u_i^T u_j = 0\,, \text{ for } i \neq j \quad \text{and} \quad u_i^T u_i = 1\,, \text{ for all } i\,.$$

We denote by $||x||^2 = x^T x$.

1. Let $U = [u_1, \ldots, u_n]$ be a matrix in $\mathbb{R}^{n\times n}$.
   (a) Show that $U^T U = UU^T = I$, I being the identity matrix.
   (b) Prove that $x = \sum_{i=1}^n (u_i^T x) u_i$ for x in $\mathbb{R}^n$.
   (c) Show that $||Ux||^2 = (Ux)^T Ux = x^T x = ||x||^2$.
2. Let $V_1 = \text{Span}(u_1, \ldots u_k)$ and $V_2 = \text{Span}(u_{k+1}, \ldots u_n)$.
   (a) Show that $\mathbb{R}^n = V_1 \oplus V_2$, i.e. V_1 and V_2 are complementary.
   (b) Let $Px = \sum_{i=1}^k (u_i^T x) u_i$; show that P is a projection.
   (c) Let $Sx = \sum_{i=1}^k (u_i^T x) u_i - \sum_{i=k+1}^n (u_i^T x) u_i$; show that S is a reflexion, i.e. $S^2 = I$. Express S in terms of P.

Exercise 11 Consider the following matrix: $A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$.

1. Compute A^2 and A^3 and show that there exist two real numbers a, b such that $A^3 = aA^2 + bA$.
2. Show that A and A^2 are linearly independent in $\mathcal{M}_3(\mathbb{R})$, the set of 3×3-matrices with real entries, i.e. If we $\alpha A + \beta A^2 = \mathbf{0}$, where $\mathbf{0}$ is the matrix with all its entries equal to 0 and α and β are two real numbers, then $\alpha = \beta = 0$.
3. Show that for all integers $n \geqslant 1$ there is an unique pair (a_n, b_n) of real numbers such that $A^n = a_n A + b_n A^2$ and express a_{n+1} and b_{n+1} depending on a_n and b_n.
   Hint: Proceed by induction.
4. Write a pseudo-code how to compute a_n and b_n for $n \geqslant 1$.
   (a) Show for all integers $n \geqslant 1$: $a_{n+2} = a_{n+1} + 2a_n$.
   (b) From that, derive a_n and b_n depending on n.
   Hint: use the characteristical equation $r^2 - r - 2 = 0$ with solutions λ_1, λ_2 and solve $a_n = \alpha\lambda_1^n + \beta\lambda_2^n$.
   (c) Give the expression of A^n depending on A, A^2 and n.

Exercise 12 (2010) Let P be the matrix defined as

$$P = \frac{1}{2}\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$$

1. Describe a basis of $\text{Ker}(P)$ the null-space (kernel) of P and $\text{Ran}(P)$ the range of P. Justify your answer.
2. Show that $\mathbb{R}^4 = \text{Ker}(P) \oplus \text{Ran}(P)$.
3. Show that for $x \in \text{Ker}(P)$ and $y \in \text{Ran}(P)$ then $x^T y = 0$.

4. Conclude that P is an orthogonal projection.

5. We assume that $(e_1, \ldots, e_n)$ is an orthonormal basis of $\mathbb{R}^n$ and, for $k = 1, \ldots, n-1$, we define $F_k = \text{Span}(e_1, \ldots, e_k)$.

   (a) Let $z \in \mathbb{R}^n$. Provide the expression of Πz the orthogonal projection of z on F_k in terms of $(e_1, \ldots, e_k)$.

   (b) Prove that for all $z \in \mathbb{R}^n$, we have $||\Pi z|| \leq ||z||$, $||z|| = \sqrt{z^T z}$.

   (c) Express Π in terms $(e_1, \ldots, e_k)$, and show that $\Pi^2 = \Pi \times \Pi = \Pi$ and $(\Pi x)^T y = x^T (\Pi y)$.

   (d) Suppose that Q is a projection (not necessarily orthogonal), such that $||Qz|| \leq ||z||$. Show that Q is an orthogonal projection.
   Hint: You can show that $x^T y = 0$ for all $x \in \text{Ran}(Q)$ and $y \in \text{Ker}(Q)$. To this end, consider $z = \lambda x + y$ for all $\lambda \in \mathbb{R}$.

6. Assume that we have two orthogonal projectors P and Q on the subspaces F and G respectively.

   We consider the matrix $R = PQ$, the product of the matrices P and Q, $\lambda \in \mathbb{R}, \lambda \neq 0$ an eigenvalue of R and $u \in \mathbb{R}^n$ an associated eigenvector, i.e. $Ru = \lambda u$.

   (a) Show that $u \in \text{Ran}(P)$ and that $Qu - \lambda u \in \text{Ker}(P)$.

   (b) Using question 5.c. and the previous question, prove that $||Qu||^2 = \lambda ||u||^2$.

   (c) Using question 5.b., conclude that the eigenvalues of R are in $[0, 1]$.

Problem 13 [3] A *Hamming-Code* is a linear error-correcting code. To handle it the *Hamming-Distance* is very useful. It is defined on F^n (with F a finite set) through

$$d(v, w) \;=\; d((v_1, , v_2, \ldots, v_n), (w_1, w_2, \ldots, w_n)) \;=\; |\{i : \; v_i \neq w_i\}| \;.$$

1. Show that it is really a metric on F^n with the following properties:

   (i) $d(v, w) = 0$ if and only if $v = w$; $v, w \in F^n$,

   (ii) $d(v, w) = d(w, v) > 0$ for all $v, w \in F^n$, $v \neq w$,

   (iii) $d(u, w) \leqslant d(u, v) + d(v, w)$ for all $u, v, w \in F^n$.

2. You can obtain all codewords $\mathbf{x}$ (in their vectorial representation) of a given Code $C = \{\mathbf{x} \in F^n : H\mathbf{x} = \mathbf{0}\}$ by looking at its *Parity-Check Matrix* H.

   Determine all possible binary codewords $\mathbf{x} \in \mathbb{Z}_2^2$ of C with the following Parity-Check-Matrix:

$$H \quad = \quad \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \;.$$

3. The *Minimal-Distance* d of a Code C is defined by $d(x, y) \geqslant d \in \mathbb{N}_0$ for all $x, y \in \mathbf{C}$ with $x \neq y$ and there must be elements x, y for which the equality holds. One is able to correct up to e errors (e.g. of transmission), where $d \geqslant 2e + 1$ ($e \in \mathbb{N}_0$). To prove this inequality, use the idea of (non-intersecting) spheres around each codeword and the Hamming-Distance. Compute d and e for the given C.

[3] This exercise is an illustration of use of linear algebra in coding theory. It is somewhat beyond the scope of the course