

Linear algebra and numerical methods

Øyvind Ryan

2026-08-11

Table of contents

Preface	1
I Linear algebra	3
1 Vectors	5
1.1 Creating vectors	5
1.2 Operations on vectors	7
1.3 Length, scalar product, and projection	12
Exercises	19
2 Matrices and matrix multiplication	27
2.1 Vectorised operations on matrices	30
2.2 Matrix multiplication	35
2.3 Linear transformations and matrix multiplication	39
2.4 Plotting polynomials	41
2.5 Block matrices	43
Exercises	45
3 Gaussian elimination and reduced echelon form	53
3.1 Gaussian elimination and complexity	60
3.2 Simultaneous solution of systems	61
3.3 Implementation of Gaussian elimination	62
Exercises	65
4 The inverse matrix	69
Exercises	74
5 Linear independence and bases	81
5.1 Subspaces	85
5.2 Coordinate mappings and change of coordinates	90
Exercises	94

6	Determinants	105
6.1	The definition of the determinant	106
6.2	Properties of the determinant	108
6.3	Magnification factors for matrices ^(*)	113
6.4	Geometrical interpretation of the determinant. The vector product.	116
	Exercises	119
7	Eigenvalues and eigenvectors	125
7.1	Diagonalisation	133
	Exercises	135
8	Diagonalisation and Systems of differential equations.	145
8.1	Higher order linear differential equations	152
	Exercises	155
9	Orthogonally diagonalisable matrices. Condition numbers.	161
9.1	Orthogonally diagonalisable matrices	162
9.2	Relative and absolute error	165
9.3	Definition of condition numbers	165
	Exercises	168
II	Numerical methods	173
10	Interpolation	175
10.1	Interpolation on canonical form	177
10.2	Interpolation on Newton form	180
10.3	The interpolation error	187
10.4	Interpolation with trigonometric functions	188
	Exercises	193
11	Numerical differentiation	205
11.1	The Newton quotient	206
11.2	The symmetric Newton quotient	209
11.3	A point method for the second derivative	211
11.4	More general point methods	211
11.5	Summary	214
	Exercises	215
12	Numerical integration	225
12.1	Definition of the integral	225
12.2	The midpoint method	228
12.2.1	The local error	229
12.2.2	The global error	231
12.2.3	The step length	232
12.3	General method for deriving numerical integration methods . . .	232
12.4	The trapezoidal rule	234

TABLE OF CONTENTS

v

12.5 Simpson's method	235
Exercises	237

Preface

These notes were first written for the new first semester course MAT1105 - Linear algebra and numerical methods, at the University of Oslo. The course was held the first time the fall of 2024, and gives an introduction to basic mathematical thinking, linear algebra, and numerical methods. In the first version of the course, linear algebra was covered by chapter 1 and 4 in the book “Flervariabel analyse med lineær algebra” by Tom Lindstrøm (in norwegian), while the two other parts of the curriculum were covered by notes on their own. In 2025 new notes for the linear algebra part were added, and merged with the notes on numerical analysis, even though the numerical analysis builds only loosely on linear algebra.

The notes contain many proofs, but many proofs in linear algebra are quite short, and are thus suitable following the mentioned introduction to basic mathematical thinking.

The notes contain several Python programming examples, as well as an introduction to basic Python commands for linear algebra. Most students at the University of Oslo take the course IN1900 - Introduction to Programming with Scientific Applications, in parallel with the notes here. Linear algebra is a suitable testbed for Python programming, and the material contains many code snippets inspired by linear algebra. There are few figures in these notes - the role of figures is partially replaced by short programming examples with their graphical results shown. It is recommended to spend some time on the programming examples. The following commands load the python packages you need.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import hilbert
import sympy as sym
import math, wave, sys, os, subprocess
import pandas as pd
```

The notes have been typeset with the Quarto publishing system (<https://quarto.org>), and come in a pdf version, and an html version. Each of these come in versions

with and without solutions to the exercises.

The html versions have some features not found in the pdf versions:

Each chapter has a quiz consisting of about 10 problems, and the quizzes aggregate the number of correct answers. Interactive programming examples and exercises which use pyodide - i.e., python is embedded in the browser, without the need for a server. For this interactive programming to work well you should avoid the Safari web browser.

Parts of this material were originally based on material written by Knut Mørken for the course MAT-INF1100, Modelling and Computations, at the University of Oslo.

These notes are maintained in english. For those of you who prefer norwegian, AI will be used to create a translation.

Blindern, August 2026

Øyvind Ryan

Part I

Linear algebra

Chapter 1

Vectors

A *vector* is a collection of numbers, also called *components*, listed in a given order. Vectors will be represented with lowercase letters in boldface (i.e., **a**, **b**, e.t.c.), and the components with lowercase letters (i.e., $a_1, \dots, a_n$, where n is the number of components). We also write $(a_1, a_2, \dots, a_n)$ for the vector **a**. In the start we will assume that all vectors are real (i.e., have real components, so that $a_i \in \mathbb{R}$ for alle i). The set of real vectors with n components will be denoted by $\mathbb{R}^n$. Later we will also study $\mathbb{C}^n$, the set of vectors with n complex components.

1.1 Creating vectors

When creating vectors in Python square brackets are used to indicate the start and end of the vector, and comma to separate the components:

```
a = np.array([2,4,6,8,10,12,14,16])  
a
```

```
array([ 2,  4,  6,  8, 10, 12, 14, 16])
```

A vector can be defined in many different ways:

```
np.linspace(0,1,6)
```

```
array([0. , 0.2, 0.4, 0.6, 0.8, 1. ])
```

More generally `np.linspace(a,b,n)` gives the vector $\left\{a + \frac{k(b-a)}{n-1}\right\}_{k=0}^{n-1}$, i.e., the vector with n components in increasing order from the interval $[a, b]$, where the end points a and b are included, and distances between neighbouring points are all equal. We can also create new vectors in the following ways:

```
np.arange(1,7)
```

```
array([1, 2, 3, 4, 5, 6])
```

```
np.arange(1,7,2)
```

```
array([1, 3, 5])
```

Python supports a *colon-notation* which can be used to extract a set of indices into a vector:

- $\mathbf{a}:\mathbf{b}$ gives the vector $(a, a+1, a+2, \dots)$, and
- $\mathbf{a}:\mathbf{b}:\mathbf{h}$ gives the vector $(a, a+h, a+2h, \dots)$,

where only numbers $\leq b$ are listed. h thus indicates a step length. Note in particular that $1:6:2$ does not contain the right end point 6.

If $\mathbf{a} = (a_1, a_2, \dots, a_n)$, and $\mathbf{I}$ is a set of non-negative indices $< n$, then $\mathbf{a}(\mathbf{I})$ returns a vector with the values found at the indices given by $\mathbf{I}$:

```
a[0:6]
```

```
array([ 2,  4,  6,  8, 10, 12])
```

Indices start at 0, contrary to indexing in mathematics where they start at 1. If we want to retrieve every second index in the same range of indices we can write.

```
a[0:6:2]
```

```
array([ 2,  6, 10])
```

There is also a special notation to extract the last part of a vector:

```
a[3:]
```

```
array([ 8, 10, 12, 14, 16])
```

3: here means “all indices ≥ 3 ”. Thus, you don’t need to know how long the vector is, in order to extract the last part of it! Note also the special syntax for extracting the last entry in a vector:

```
a[-1]
```

```
16
```

Example 1.1 (max, min, sum, and prod). One can write as follows if one needs to find the maximum, minimum, sum, or product of all components in a vector:

```
np.max(a)
```

```
16
```

```
np.min(a)
```

```
2
```

```
np.sum(a)
```

```
72
```

```
np.prod(a)
```

```
10321920
```



1.2 Operations on vectors

Vector addition- and subtraction are defined *component-wise*. This means that, if $\mathbf{a}$ and $\mathbf{b}$ both are vectors with n components, then

- $\mathbf{a} + \mathbf{b} = (a_1 + b_1, a_2 + b_2, \dots, a_n + b_n)$,
- $\mathbf{a} - \mathbf{b} = (a_1 - b_1, a_2 - b_2, \dots, a_n - b_n)$.
- $t\mathbf{a} = (ta_1, ta_2, \dots, ta_n)$ when t is a scalar.

Vector addition thus requires two vectors with the same number of components, and returns a vector with the same number of components.

It is common to draw vectors as arrows starting at the origin, and ending at the point indicated by the vector. But a vector can also be drawn starting at another point - it is the direction the vector points in which defines its components. Figure 1.1 illustrates addition and subtraction of vectors. Calculating $\mathbf{a} + \mathbf{b}$ corresponds to placing the vector $\mathbf{b}$ at the end of the vector $\mathbf{a}$, and reading the coordinates you then get. $\mathbf{a} - \mathbf{b}$ can be visualised in two ways: Either by placing the vector $-\mathbf{b}$ at the end of the vector $\mathbf{a}$ (as in Figure 1.1a), or by drawing the arrow which starts in $\mathbf{b}$ and ends in $\mathbf{a}$ (as in Figure 1.1b).

Figure 1.2 illustrates multiplication of a vector with a scalar.

The *Zero-vector*, written $\mathbf{0}$, is the vector where all components are 0, i.e., $\mathbf{0} = (0, 0, \dots, 0)$. It is clear that the zero-vector satisfies $\mathbf{a} + \mathbf{0} = \mathbf{a}$, and that $0\mathbf{a} = \mathbf{0}$, for all $\mathbf{a}$.

Example 1.2. Let $\mathbf{a} = (1, 2, 3)$, $\mathbf{b} = (4, 5, 6)$, and $t = 2$. Then

$$\begin{aligned}\mathbf{a} + \mathbf{b} &= (1, 2, 3) + (4, 5, 6) = (1 + 4, 2 + 5, 3 + 6) = (5, 7, 9) \\ \mathbf{a} - \mathbf{b} &= (1, 2, 3) - (4, 5, 6) = (1 - 4, 2 - 5, 3 - 6) = (-3, -3, -3) \\ t\mathbf{a} &= 2(1, 2, 3) = (2 \cdot 1, 2 \cdot 2, 2 \cdot 3) = (2, 4, 6).\end{aligned}$$

This can be verified as follows:

```
a = np.array([1,2,3])
b = np.array([4,5,6])
t = 2
a + b
```

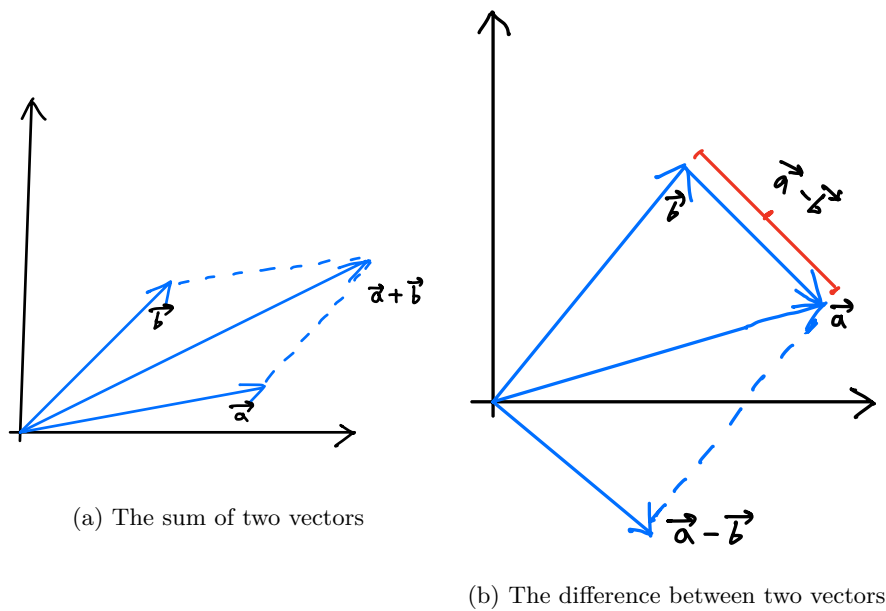


Figure 1.1: Operations on vectors.

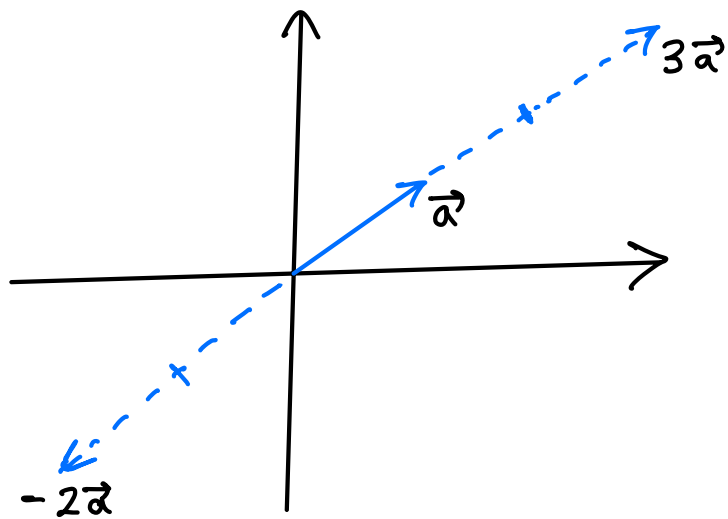


Figure 1.2: Multiplication of a vector with a scalar

```
array([5, 7, 9])
```

```
a-b
```

```
array([-3, -3, -3])
```

```
t*a
```

```
array([2, 4, 6])
```

Note the difference between writing $\mathbf{a}+\mathbf{b}$, and $\text{sum}(\mathbf{a})$. The first computes many sums of two components. The last computes one sum of many components. ♣

One often applies an operation on all components in a vector. One can do this in a loop. In order to limit the number of loops, programming languages use *vectorisation*, i.e., operations defined on $\mathbb{R}$ or $\mathbb{C}$ are naturally extended to vectors by applying the operation on each component. Addition and subtraction of vectors, as defined above, are clearly vectorised operations.

Definition 1.1 (Vectorisation). Suppose f is a function on $\mathbb{R}$ or $\mathbb{C}$, and that $\mathbf{a} = (a_1, a_2, \dots, a_n)$. The *vectorisation* of f is defined by

$$f(\mathbf{a}) = (f(a_1), f(a_2), \dots, f(a_n)).$$

Vectorisation of binary functions (functions in two variables) is defined similarly. As an example the vectorisations of addition, taking powers, and multiplication, are

$$\begin{aligned}\mathbf{a} + \mathbf{b} &= (a_1 + b_1, a_2 + b_2, \dots, a_n + b_n) \\ \mathbf{a}^{\mathbf{b}} &= (a_1^{b_1}, a_2^{b_2}, \dots, a_n^{b_n}) \\ \mathbf{a}\mathbf{b} &= (a_1b_1, a_2b_2, \dots, a_nb_n).\end{aligned}$$

Here both $\mathbf{a}$ and $\mathbf{b}$ are vectors with n components.

Addition of a vector and a scalar is also allowed. This is computed as

$$\mathbf{a} + t = (a_1 + t, a_2 + t, \dots, a_n + t).$$

Note that this is not defined mathematically.

```
a+t
```

```
array([3, 4, 5])
```

Example 1.3 (Vectorised functions). Most functions in Python are vectorised:

```
np.sqrt(np.array([1,4,9,16]))
```

```
array([1., 2., 3., 4.])
```

```
np.abs(np.array([-2,-1,1,2]))

array([2, 1, 1, 2])

np.sin(np.array([0,np.pi/2,np.pi,3*np.pi/2,2*np.pi]))

array([ 0.0000000e+00,  1.0000000e+00,  1.2246468e-16, -1.0000000e+00,
        -2.4492936e-16])

np.cos(np.array([0,np.pi/2,np.pi,3*np.pi/2,2*np.pi]))

array([ 1.0000000e+00,  6.1232340e-17, -1.0000000e+00, -1.8369702e-16,
        1.0000000e+00])
```

The code below verifies that the logarithm and the exponential functions are inverse functions:

```
np.exp(np.log(np.array([1,2,3])))

array([1., 2., 3.])

np.log(np.exp(np.array([0,1,2])))

array([0., 1., 2.])
```



Example 1.4 (Vectorised binary functions). `max` and `min` can be used in the same way as `+` and `-`, if one instead passes two arguments:

```
np.maximum(a,b)

array([4, 5, 6])

np.minimum(a,b)

array([1, 2, 3])
```

In particular we have that

$$\begin{aligned} &\max((a_1, a_2, \dots, a_n), (b_1, b_2, \dots, b_n)) \\ &= (\max(a_1, b_1), \max(a_2, b_2), \dots, \max(a_n, b_n)). \end{aligned}$$

Multiplication and taking powers are also vectorised:

```
a*b

array([ 4, 10, 18])

a**b

array([ 1, 32, 729])
```



There are many other operations which can be applied to vectors. As an example, they can be placed after one-another:

```
np.block([a,b])
```

```
array([1, 2, 3, 4, 5, 6])
```

Since vector addition is vectorised, vectors inherit properties from the real numbers:

Proposition 1.1. *We have that*

1. $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$
2. $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$
3. $s(\mathbf{a} + \mathbf{b}) = s\mathbf{a} + s\mathbf{b}$
4. $(s + t)\mathbf{a} = s\mathbf{a} + t\mathbf{a}$,

where $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are vectors, and s and t are scalars.

In the proof, and several times later, we will use the fact that $\mathbf{a}$ and $\mathbf{b}$ are equal if and only if $a_i = b_i$ for all i .

Proof. 1.-3. follow directly from the properties of vectors, and those for real numbers. Let us therefore only consider 4..

- The left side $(s + t)\mathbf{a}$ has components $(s + t)a_i$, by the definition of scalar-vector multiplication.
- On the right hand side $s\mathbf{a}$ has components sa_i , and $t\mathbf{a}$ has components ta_i . By the definition of vector addition, the right hand side $s\mathbf{a} + t\mathbf{a}$ therefore has components $sa_i + ta_i$.

The right- and left hand sides are therefore equal if and only if $(s + t)a_i = sa_i + ta_i$ for all i . But this we know holds for all real numbers. $\square$

Example 1.5 (Lines). Let us find an expression for the line in $\mathbb{R}^n$ which passes through two given points $\mathbf{x}_1$ and $\mathbf{x}_2$. This must have direction vector $\mathbf{x}_2 - \mathbf{x}_1$. Since the line should pass through $\mathbf{x}_1$, the expression becomes

$$\mathbf{y}(t) = \mathbf{x}_1 + t(\mathbf{x}_2 - \mathbf{x}_1) = (1 - t)\mathbf{x}_1 + t\mathbf{x}_2.$$

We also see that $\mathbf{y}(0) = \mathbf{x}_1$, and $\mathbf{y}(1) = \mathbf{x}_2$. ♣

Up to now all vectors and scalars have been real, but they may just as well have been complex. From now on we will also consider complex vectors and scalars. The complex number $2 + 3i$ can be computed as

```
2+3j
```

```
(2+3j)
```

`1j` represents the imaginary unit i . `3j` gives the complex number $3i$. Using `j` alone will produce an error message, unless `j` is defined as a variable elsewhere.

1.3 Length, scalar product, and projection

The *length* of a real vector $\mathbf{a}$ is defined as

$$|\mathbf{a}| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}. \quad (1.1)$$

This is just a generalisation of length from $\mathbb{R}^2$ and $\mathbb{R}^3$. In particular we have that $|\mathbf{a}| \geq 0$, and $|\mathbf{a}| = 0$ if and only if $\mathbf{a} = \mathbf{0}$.

If $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ we define the *real scalar product* $\mathbf{a} \cdot \mathbf{b}$ by

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n. \quad (1.2)$$

We say that $\mathbf{a}$ and $\mathbf{b}$ are *orthogonal* if $\mathbf{a} \cdot \mathbf{b} = 0$.

Example 1.6. The lengths of the vectors $\mathbf{a} = (1, -4, 9)$ and $\mathbf{b} = (2, 3, -6)$ are

$$\begin{aligned} |\mathbf{a}| &= \sqrt{1^2 + (-4)^2 + 9^2} = \sqrt{1 + 16 + 81} = \sqrt{98} = 7\sqrt{2} \\ |\mathbf{b}| &= \sqrt{2^2 + 3^2 + (-6)^2} = \sqrt{4 + 9 + 36} = 7. \end{aligned}$$

The scalar product of the two is

$$\mathbf{a} \cdot \mathbf{b} = (1, -4, 9) \cdot (2, 3, -6) = 1 \cdot 2 + (-4) \cdot 3 + 9 \cdot (-6) = 2 - 12 - 54 = -64.$$



We have the following result

Proposition 1.2 (Properties of the scalar product). *For all vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ in $\mathbb{R}^n$, and scalars s , we have that*

1. $\mathbf{a} \cdot \mathbf{a} = |\mathbf{a}|^2$,
2. $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$,
3. $\mathbf{c} \cdot (\mathbf{a} + \mathbf{b}) = \mathbf{c} \cdot \mathbf{a} + \mathbf{c} \cdot \mathbf{b}$ and $(\mathbf{a} + \mathbf{b}) \cdot \mathbf{c} = \mathbf{a} \cdot \mathbf{c} + \mathbf{b} \cdot \mathbf{c}$,
4. $(s\mathbf{a}) \cdot \mathbf{b} = \mathbf{a} \cdot (s\mathbf{b}) = s(\mathbf{a} \cdot \mathbf{b})$

Proof. 1.-3. follow directly from the laws of real numbers, and are given as an exercise. 4. can be proved as follows:

$$\begin{aligned} (s\mathbf{a}) \cdot \mathbf{b} &= (sa_1, \dots, sa_n) \cdot (b_1, \dots, b_n) = sa_1 b_1 + \cdots sa_n b_n \\ \mathbf{a} \cdot (s\mathbf{b}) &= (a_1, \dots, a_n) \cdot (sb_1, \dots, sb_n) = sa_1 b_1 + \cdots sa_n b_n \\ s(\mathbf{a} \cdot \mathbf{b}) &= s(a_1 b_1 + \cdots + a_n b_n) = sa_1 b_1 + \cdots sa_n b_n \end{aligned}$$

where we again used the laws of real numbers. □

One can also show the other way that the scalar product is the only function in two variables ($\mathbf{a}$ and $\mathbf{b}$) which satisfies 1.-3 (see Exercise 1.11).

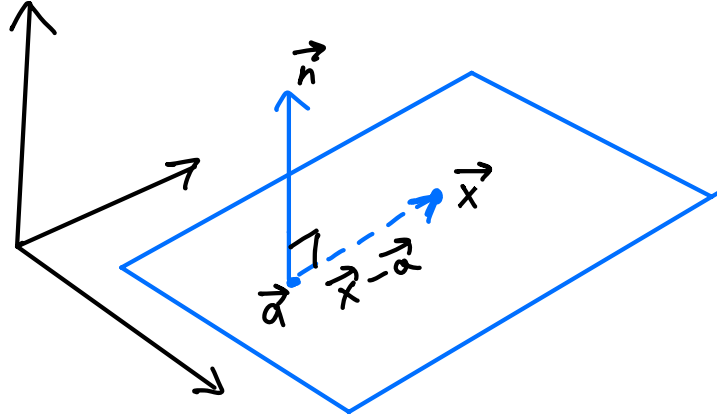


Figure 1.3: A plane with a given normal vector and point.

Example 1.7 (Planes). Let us also find an expression for the plane passing through a given point (x_0, y_0, z_0) , and which is orthogonal to a given vector $\mathbf{n} = (a, b, c)$. (x, y, z) lies in this plane if and only if $(x, y, z) - (x_0, y_0, z_0)$ is orthogonal to $\mathbf{n}$, see Figure 1.3.

This is the same as

$$\begin{aligned} 0 &= ((x, y, z) - (x_0, y_0, z_0)) \cdot \mathbf{n} \\ &= (x - x_0, y - y_0, z - z_0) \cdot (a, b, c) \\ &= a(x - x_0) + b(y - y_0) + c(z - z_0). \end{aligned}$$

The expression for the plane is therefore

$$ax + by + cz = d,$$

where $d = ax_0 + by_0 + cz_0$. ♣

Let us now define length and scalar product for complex vectors. For length the definition given by Equation 1.1 will not work, as this gives complex numbers when $\mathbf{a}$ is complex (we want length to be a real number). Instead we define

$$|\mathbf{a}| = \sqrt{|a_1|^2 + |a_2|^2 + \dots + |a_n|^2}, \quad (1.3)$$

which reduces to the definition given by Equation 1.1 when $\mathbf{a}$ is real. Inside the square root here we take lengths of complex numbers, so that the result becomes ≥ 0 , and 0 only when $\mathbf{a} = \mathbf{0}$. The length of a vector can be computed as follows.

```
a=np.array([3,4])
np.linalg.norm(a)
```

5.0

This can also be computed as follows

```
np.sqrt(np.sum(np.abs(a)**2))
```

5.0

Here four operations are combined. With $\mathbf{a} = (a_1, a_2, \dots, a_n)$, these are, in the following order:

1. $(|a_1|, |a_2|, \dots, |a_n|)$ (vectorised absolute value),
2. $(|a_1|^2, |a_2|^2, \dots, |a_n|^2)$ (vectorised power),
3. $|a_1|^2 + |a_2|^2 + \dots + |a_n|^2$,
4. $\sqrt{|a_1|^2 + |a_2|^2 + \dots + |a_n|^2}$.

The two last operations are not vectorised, as they return a scalar.

Let us now define the complex scalar product. As for the real scalar product we want

$$\mathbf{a} \cdot \mathbf{a} = |\mathbf{a}|^2 = a_1 \overline{a_1} + \dots + a_n \overline{a_n}.$$

It is clear that Equation 1.2 will not work for the complex case, as this will not capture the conjugation here. Let us instead define

$$\mathbf{a} \cdot \mathbf{b} = a_1 \overline{b_1} + \dots + a_n \overline{b_n}, \quad (1.4)$$

since this will be compatible with the expression of length when $\mathbf{a} = \mathbf{b}$. We have the following result.

Proposition 1.3. *For all vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ in $\mathbb{C}^n$, and $s \in \mathbb{C}$, we have that*

1. $\mathbf{a} \cdot \mathbf{a} = |\mathbf{a}|^2$,
2. $\mathbf{a} \cdot \mathbf{b} = \overline{\mathbf{b} \cdot \mathbf{a}}$,
3. $\mathbf{c} \cdot (\mathbf{a} + \mathbf{b}) = \mathbf{c} \cdot \mathbf{a} + \mathbf{c} \cdot \mathbf{b}$ and $(\mathbf{a} + \mathbf{b}) \cdot \mathbf{c} = \mathbf{a} \cdot \mathbf{c} + \mathbf{b} \cdot \mathbf{c}$,
4. $(s\mathbf{a}) \cdot \mathbf{b} = s(\mathbf{a} \cdot \mathbf{b})$ and $\mathbf{a} \cdot (s\mathbf{b}) = \overline{s}(\mathbf{a} \cdot \mathbf{b})$,

when $\mathbf{a} \cdot \mathbf{b}$ is defined by Equation 1.4.

Proof.

1. is clear.
2. follows from that $a\overline{b} = \overline{b\overline{a}}$ for all complex numbers a and b .
3. follows as before.
4. follows from that $(sa)\overline{b} = s(a\overline{b})$ and $a\overline{sb} = \overline{s}a\overline{b}$ when $a, b, s \in \mathbb{C}$.

□

Example 1.8. Consider the vectors $\mathbf{a} = (2i, 1+i, 3)$ and $\mathbf{b} = (1+2i, -1, 2-i)$. For the components we first compute

$$|2i|^2 = 4 \quad |1+i|^2 = 2 \quad |3|^2 = 9 \quad |1+2i|^2 = 5 \quad |-1|^2 = 1 \quad |2-i|^2 = 5$$

The lengths of $\mathbf{a}$ and $\mathbf{b}$ are thus

$$\begin{aligned} |\mathbf{a}| &= \sqrt{|2i|^2 + |1+i|^2 + 3^2} = \sqrt{4 + 2 + 9} = \sqrt{15} \\ |\mathbf{b}| &= \sqrt{|1+2i|^2 + (-1)^2 + |2-i|^2} = \sqrt{5 + 1 + 5} = \sqrt{11} \end{aligned}$$

and the scalar product of the two is

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= (2i, 1+i, 3) \cdot (1+2i, -1, 2-i) \\ &= 2i(\overline{1+2i}) + (1+i)\overline{-1} + 3(\overline{2-i}) \\ &= 2i(1-2i) - (1+i) + 3(2+i) \\ &= 2i + 4 - 1 - i + 6 + 3i = 9 + 4i. \end{aligned}$$

You get the complex scalar product by writing `np.vdot(a,b)`, but here there is a small difference:

```
a = np.array([2j, 1+1j, 3])
b = np.array([1+2j, -1, 2-1j])
np.vdot(a,b)
```

(9-4j)

So here a slightly different definition is used than Equation 1.4. `np.vdot` actually computes the complex scalar product of $\mathbf{a}$ and $\mathbf{b}$ as $\overline{a_1}b_1 + \dots + \overline{a_n}b_n$, as is easily checked:

```
np.sum(np.conj(a)*b)
```

(9-4j)

These two competing definitions of the complex scalar product stem from different science traditions: In the mathematics literature it is usual to conjugate the second vector, while in the physics literature it is common to conjugate the first vector. `np.vdot` thus follows the literature in physics. ♣

If $\mathbf{a}$ and $\mathbf{b}$ are orthogonal a simple calculation gives

$$|\mathbf{a} + \mathbf{b}|^2 = (\mathbf{a} + \mathbf{b}) \cdot (\mathbf{a} + \mathbf{b}) = \mathbf{a} \cdot \mathbf{a} + \mathbf{a} \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{a} + \mathbf{b} \cdot \mathbf{b} = |\mathbf{a}|^2 + |\mathbf{b}|^2,$$

where we used that $\mathbf{b} \cdot \mathbf{a} = \overline{\mathbf{a} \cdot \mathbf{b}} = \overline{0} = 0$. It follows that $|\mathbf{a} + \mathbf{b}|^2 = |\mathbf{a}|^2 + |\mathbf{b}|^2$. If we in particular set

$$n = 2 \quad \mathbf{a} = (a, 0) \quad \mathbf{b} = (0, b) \quad \mathbf{c} = \mathbf{a} + \mathbf{b} = (a, b),$$

this says that $c^2 = a^2 + b^2$, where a, b, c are the lengths of $\mathbf{a}, \mathbf{b}, \mathbf{c}$. Since a and b are legs, and c hypotenuse in a right-angled triangle, we have shown a generalisation of Pythagoras theorem to n dimensions!

We say that two non-zero vectors $\mathbf{a}$ and $\mathbf{b}$ are *parallel* if $\mathbf{a} = s\mathbf{b}$ for some scalar s . Let $\mathbf{a}$ and $\mathbf{b}$ be vectors, where $\mathbf{b} \neq \mathbf{0}$. Intuition says that the closest point to $\mathbf{a}$ on the line between $\mathbf{0}$ and $\mathbf{b}$ is obtained by drawing the normal from that point down to the line (see Figure 1.4). The next proposition says that the point obtained by following the normal vector is unique.

Proposition 1.4 (Orthogonal decomposition theorem). *Suppose $\mathbf{a}, \mathbf{b}$ are vectors, where $\mathbf{b} \neq \mathbf{0}$. $\mathbf{p} = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b}$ is the unique point on the line between $\mathbf{0}$ and $\mathbf{b}$ so that $\mathbf{a} - \mathbf{p}$ and $\mathbf{p}$ are orthogonal.*

We also write $\text{proj}_{\mathbf{b}}(\mathbf{a})$ for $\mathbf{p} = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b}$, and call this the *projection* of $\mathbf{a}$ onto $\mathbf{b}$. The decomposition

$$\mathbf{a} = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b} + \left(\mathbf{a} - \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b} \right) \quad (1.5)$$

thus splits $\mathbf{a}$ in a sum where one vector is parallel to $\mathbf{b}$, and the other orthogonal to $\mathbf{b}$. This is also called an *orthogonal decomposition*.

Proof. That $\mathbf{p} = c\mathbf{b}$ is orthogonal to $\mathbf{a} - c\mathbf{b}$ reads that

$$0 = (\mathbf{a} - c\mathbf{b}) \cdot \mathbf{b} = \mathbf{a} \cdot \mathbf{b} - c|\mathbf{b}|^2,$$

which is equivalent to $c = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2}$, so that $\mathbf{p} = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b}$. □

To see that $\mathbf{p}$ also is the unique point on the line between $\mathbf{0}$ and $\mathbf{b}$ closest to $\mathbf{a}$, for $s \in \mathbb{C}$ write

$$|\mathbf{a} - s\mathbf{b}|^2 = |(\mathbf{a} - \mathbf{p}) + (\mathbf{p} - s\mathbf{b})|^2 = |\mathbf{a} - \mathbf{p}|^2 + |\mathbf{p} - s\mathbf{b}|^2 \geq |\mathbf{a} - \mathbf{p}|^2,$$

where we used Pythagoras theorem. Thus, $\frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b}$ is the unique point on the line which is closest to $\mathbf{a}$. This result is also called the *best approximation theorem*.

$\text{proj}_{\mathbf{b}}(\mathbf{a})$ can be computed as follows

```
np.vdot(b,a)*b/np.linalg.norm(b**2)
```

```
array([ 0.14002801+3.08061618j, -1.26025208-0.56011203j,
        3.08061618-0.14002801j])
```

Note that the order of $\mathbf{a}$ and $\mathbf{b}$ has been changed inside the scalar product here. Do you see why?

We see that $|\text{proj}_{\mathbf{b}}(\mathbf{a})| = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|}$. From the proof above it follows that we also can write

$$\mathbf{a} = (\mathbf{a} - \text{proj}_{\mathbf{b}}(\mathbf{a})) + \text{proj}_{\mathbf{b}}(\mathbf{a}),$$

where the two vectors on the right side are orthogonal. Thus, by using projections we can write $\mathbf{a}$ as a sum of two vectors: one orthogonal to $\mathbf{b}$, and one parallel with $\mathbf{b}$. This is also called an *orthogonal decomposition*.

Example 1.9. Let $\mathbf{a} = (10, 5)$, and $\mathbf{b} = (2, 4)$. Then

$$|\mathbf{b}|^2 = 4 + 16 = 20 \qquad \mathbf{a} \cdot \mathbf{b} = 20 + 20 = 40,$$

so that

$$\text{proj}_{\mathbf{b}}(\mathbf{a}) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b} = \frac{40}{20} (2, 4) = (4, 8).$$

We also get

$$\mathbf{a} - \text{proj}_{\mathbf{b}}(\mathbf{a}) = (10, 5) - (4, 8) = (6, -3).$$

Therefore we have that

$$(10, 5) = (4, 8) + (6, -3)$$

is an orthogonal decomposition of $(10, 5)$ into a vector orthogonal to $\mathbf{b}$, and one parallel with $\mathbf{b}$. ♣

Figure 1.4 illustrates projection in $\mathbb{R}^2$. The closest point to $\mathbf{a}$ on the line through $\mathbf{b}$ and the origin is obtained by drawing the line orthogonal to $\mathbf{b}$.

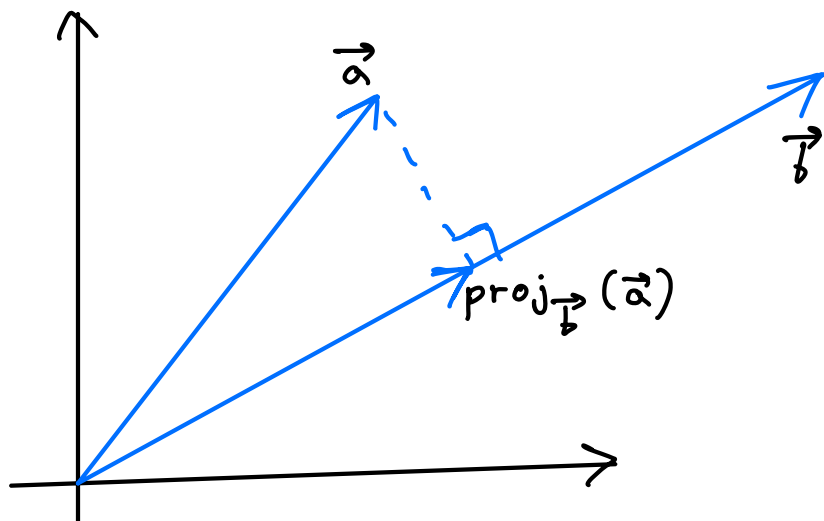


Figure 1.4: Projection of the vector $\mathbf{a}$ onto the vector $\mathbf{b}$ in the plane.

From the figure it is also clear that

$$\frac{|\text{proj}_{\mathbf{b}}(\mathbf{a})|}{|\mathbf{a}|} = \cos \theta,$$

where θ is the angle between $\mathbf{a}$ and $\mathbf{b}$. If we here insert $\frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b}$ for $\text{proj}_{\mathbf{b}}(\mathbf{a})$ we get after some computation that $|\mathbf{a} \cdot \mathbf{b}| = |\mathbf{a}||\mathbf{b}| \cos \theta$, which is a formula known to most of us. From this formula we can deduce several known trigonometric identities. As an example, with $\mathbf{a} = (\cos \theta_1, \sin \theta_1)$, $\mathbf{b} = (\cos \theta_2, \sin \theta_2)$, we have that

$$\mathbf{a} \cdot \mathbf{b} = \cos \theta_1 \cos \theta_2 + \sin \theta_1 \sin \theta_2 \quad |\mathbf{a}| = 1 \quad |\mathbf{b}| = 1.$$

Since also the angle between $\mathbf{a}$ and $\mathbf{b}$ is $\theta = \theta_2 - \theta_1$, insertion in $|\mathbf{a} \cdot \mathbf{b}| = |\mathbf{a}||\mathbf{b}| \cos \theta$ therefore gives

$$\cos(\theta_2 - \theta_1) = \cos \theta_1 \cos \theta_2 + \sin \theta_1 \sin \theta_2,$$

so that we have deduced the known formula for the cosine of a difference of two vectors.

From $|\mathbf{a} \cdot \mathbf{b}| = |\mathbf{a}||\mathbf{b}|\cos\theta$ above it also follows that $|\mathbf{a} \cdot \mathbf{b}| \leq |\mathbf{a}||\mathbf{b}|$, at least for vectors in the plane. This inequality turns out to hold more generally. To see this we will use Pythagoras theorem on the orthogonal decomposition given by Equation 1.5, which gives

$$|\mathbf{a}|^2 = |\mathbf{a} - \text{proj}_{\mathbf{b}}(\mathbf{a})|^2 + |\text{proj}_{\mathbf{b}}(\mathbf{a})|^2 \geq |\text{proj}_{\mathbf{b}}(\mathbf{a})|^2. \quad (1.6)$$

Taking square roots on each side we get

$$|\mathbf{a}| \geq |\text{proj}_{\mathbf{b}}(\mathbf{a})| = \left| \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b} \right| = \left| \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \right| |\mathbf{b}| = \frac{|\mathbf{a} \cdot \mathbf{b}|}{|\mathbf{b}|}.$$

Multiplying up we get the following result.

Proposition 1.5 (Schwarz' inequality). *We have that $|\mathbf{a} \cdot \mathbf{b}| \leq |\mathbf{a}||\mathbf{b}|$ for all vectors $\mathbf{a}, \mathbf{b} \in \mathbb{C}^n$. We have equality if and only if either $\mathbf{a}$ or $\mathbf{b}$ are parallel, or at least one of $\mathbf{a}$ and $\mathbf{b}$ is 0.*

Let us comment on how one can find the requirements for equality in Schwarz' inequality. Because of Equation 1.6 this can occur only when $\text{proj}_{\mathbf{b}}(\mathbf{a}) = \mathbf{a}$. If $\mathbf{a}, \mathbf{b} \neq \mathbf{0}$ it is clear that this is the case only when $\mathbf{a}$ and $\mathbf{b}$ are parallel. Finally, equality in Schwarz' inequality when either $\mathbf{a}$ or $\mathbf{b}$ is $\mathbf{0}$ is obvious.

An immediate consequence of Schwarz' inequality is the *triangle inequality*.

Proposition 1.6 (The triangle inequality). *We have that $|\mathbf{a} + \mathbf{b}| \leq |\mathbf{a}| + |\mathbf{b}|$ for all vectors $\mathbf{a}, \mathbf{b} \in \mathbb{C}^n$. We have equality if and only if either $\mathbf{a} = s\mathbf{b}$ for an $s > 0$, or at least one of $\mathbf{a}$ and $\mathbf{b}$ is 0.*

The obvious interpretation of the triangle inequality is that “the shortest way between two points is a straight line”. This we see by comparing the distance from the origin to $\mathbf{a} + \mathbf{b}$, with the distance obtained by going to $\mathbf{a} + \mathbf{b}$ by first following the vector $\mathbf{a}$, and then the vector $\mathbf{b}$ (see Figure 1.1 (a) again).

Proof. We have that

$$\begin{aligned} |\mathbf{a} + \mathbf{b}|^2 &= |\mathbf{a}|^2 + \mathbf{a} \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{a} + |\mathbf{b}|^2 = |\mathbf{a}|^2 + 2\Re(\mathbf{a} \cdot \mathbf{b}) + |\mathbf{b}|^2 \\ &\leq |\mathbf{a}|^2 + 2|\mathbf{a} \cdot \mathbf{b}| + |\mathbf{b}|^2 \leq |\mathbf{a}|^2 + 2|\mathbf{a}||\mathbf{b}| + |\mathbf{b}|^2 = (|\mathbf{a}| + |\mathbf{b}|)^2, \end{aligned}$$

where we used Schwarz' inequality. The result now follows by taking square roots.

To obtain equality we first must have equality in Schwarz' inequality, i.e., $\mathbf{a}$ and $\mathbf{b}$ are parallel, or at least one of $\mathbf{a}$ and $\mathbf{b}$ is 0. If at least one of $\mathbf{a}$ and $\mathbf{b}$ is 0, we see that we also have equality in the other inequality, and then there is also equality in the triangle inequality. Suppose now that $\mathbf{a}$ and $\mathbf{b}$ are parallel and both $\neq \mathbf{0}$. We have equality in the other inequality if and only if $|\mathbf{a} \cdot \mathbf{b}| = \Re(\mathbf{a} \cdot \mathbf{b})$. But then $\mathbf{a} \cdot \mathbf{b}$ must be real and positive. Since $\mathbf{b} = s\mathbf{a}$ we get that $\mathbf{a} \cdot \mathbf{b} = \bar{s}|\mathbf{a}|^2$. It follows that $s > 0$. $\square$

Exercises

Exercise 1.1. Compute

$$(a) \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix} + 3 \begin{pmatrix} -1 \\ -4 \\ 3 \end{pmatrix}$$

! Solution

$$\begin{aligned} \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix} + 3 \begin{pmatrix} -1 \\ -4 \\ 3 \end{pmatrix} &= \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix} + \begin{pmatrix} 3 \cdot (-1) \\ 3 \cdot (-4) \\ 3 \cdot 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix} + \begin{pmatrix} -3 \\ -12 \\ 9 \end{pmatrix} = \begin{pmatrix} 2-3 \\ 1-12 \\ 4+9 \end{pmatrix} \\ &= \begin{pmatrix} -1 \\ -11 \\ 13 \end{pmatrix} \end{aligned}$$

```
np.array([2,1,4]) + 3*np.array([-1,-4,3])
array([ -1, -11, 13])
```

$$(b) \begin{pmatrix} -i \\ -1 \\ 3 \end{pmatrix} + i \begin{pmatrix} -i \\ 2 \\ 2 \end{pmatrix}.$$

! Solution

$$\begin{aligned} \begin{pmatrix} -i \\ -1 \\ 3 \end{pmatrix} + i \begin{pmatrix} -i \\ 2 \\ 2 \end{pmatrix} &= \begin{pmatrix} -i \\ -1 \\ 3 \end{pmatrix} + \begin{pmatrix} i \cdot (-i) \\ i \cdot 2 \\ i \cdot 2 \end{pmatrix} = \begin{pmatrix} -i \\ -1 \\ 3 \end{pmatrix} + \begin{pmatrix} 1 \\ 2i \\ 2i \end{pmatrix} = \begin{pmatrix} -i+1 \\ -1+2i \\ 3+2i \end{pmatrix} \\ &= \begin{pmatrix} 1-i \\ -1+2i \\ 3+2i \end{pmatrix}. \end{aligned}$$

```
np.array([-1j,-1,3]) + 1j*np.array([-1j,2,2])
array([ 1.-1.j, -1.+2.j, 3.+2.j])
```

Exercise 1.2. Find

- (a) a parametrisation of the line going through the two points $(1, 1, 1)$ and $(2, 4, 3)$.

! Solution

The line can be parametrised by

$$\mathbf{y}(t) = (1-t) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + t \begin{pmatrix} 2 \\ 4 \\ 3 \end{pmatrix} = \begin{pmatrix} 1-t \\ 1-t \\ 1-t \end{pmatrix} + \begin{pmatrix} 2t \\ 4t \\ 3t \end{pmatrix} = \begin{pmatrix} 1+t \\ 1+3t \\ 1+2t \end{pmatrix}.$$

- (b) an expression for the plane going through $(3, -1, -1)$ which is orthogonal to $(1, 1, 1)$.

! Solution

The plane consists of all points (x, y, z) so that $(x, y, z) - (3, -1, -1)$ is orthogonal to $(1, 1, 1)$, meaning that

$$\begin{aligned} 0 &= \left(\begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} 3 \\ -1 \\ -1 \end{pmatrix} \right) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} x-3 \\ y+1 \\ z+1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \\ &= x-3+y+1+z+1 = x+y+z-1. \end{aligned}$$

The plane therefore consists of all points so that $x+y+z=1$.

Exercise 1.3. Compute the lengths of the vectors

- (a) $(3, 4, 5)$

! Solution

$$|(3, 4, 5)| = \sqrt{3^2 + 4^2 + 5^2} = \sqrt{9 + 16 + 25} = \sqrt{50} = 5\sqrt{2}$$

```
np.linalg.norm(np.array([3,4,5]))
```

```
7.0710678118654755
```

- (b) $(i, -i, 1, -1)$.

! Solution

$$\begin{aligned} |(i, -i, 1, -1)| &= \sqrt{|i|^2 + |-i|^2 + |1|^2 + |-1|^2} \\ &= \sqrt{1^2 + (-1)^2 + 1 + 1} = \sqrt{4} = 2. \end{aligned}$$

```
np.linalg.norm(np.array([1j,-1j,1,-1]))
2.0
```

Exercise 1.4. Compute the scalar product of

- (a) $(-1, 2, 4)$ and $(3, 3, -1)$.

! Solution

$$(-1, 2, 4) \cdot (3, 3, -1) = -1 \cdot 3 + 2 \cdot 3 + 4 \cdot (-1) = -3 + 6 - 4 = -1.$$

- (b) $(-i, 2, 3 + i)$ and $(2 - i, -4i, -2)$.

! Solution

$$\begin{aligned} (-i, 2, 3 + i) \cdot (2 - i, -4i, -2) &= -i \cdot \overline{2 - i} + 2 \cdot \overline{-4i} + (3 + i) \cdot \overline{-2} \\ &= -i(2 + i) + 2 \cdot (4i) + (3 + i) \cdot (-2) \\ &= -2i + 1 + 8i - 6 - 2i = -5 + 4i. \end{aligned}$$

Exercise 1.5.

- (a) Compute the projection of the vector $\mathbf{a} = (1, 2, 4)$ onto $\mathbf{b} = (2, 4, 2)$.

! Solution

We compute

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= (1, 2, 4) \cdot (2, 4, 2) = 2 + 8 + 8 = 18 \\ |\mathbf{b}|^2 &= 2^2 + 4^2 + 2^2 = 4 + 16 + 4 = 24 \end{aligned}$$

We therefore get

$$\text{proj}_{\mathbf{b}} \mathbf{a} = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b} = \frac{18}{24} \begin{pmatrix} 2 \\ 4 \\ 2 \end{pmatrix} = \begin{pmatrix} 3/2 \\ 3 \\ 3/2 \end{pmatrix}.$$

- (b) find an orthogonal decomposition of $\mathbf{a}$ in a vector orthogonal to $\mathbf{b}$, and one parallel with $\mathbf{b}$.

! Solution

We have that $\mathbf{a} - \text{proj}_{\mathbf{b}} \mathbf{a} = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} - \begin{pmatrix} 3/2 \\ 3 \\ 3/2 \end{pmatrix} = \begin{pmatrix} -1/2 \\ -1 \\ 5/2 \end{pmatrix}$, and we get the orthogonal decomposition

$$\mathbf{a} = \text{proj}_{\mathbf{b}} \mathbf{a} + (\mathbf{a} - \text{proj}_{\mathbf{b}} \mathbf{a}) = \begin{pmatrix} 3/2 \\ 3 \\ 3/2 \end{pmatrix} + \begin{pmatrix} -1/2 \\ -1 \\ 5/2 \end{pmatrix}$$

Exercise 1.6. Prove property 1.-3. in Proposition 1.1 and Proposition 1.2.

! Solution

All these properties follow from the corresponding properties for real numbers, when we combine with the definition of vector addition and multiplication with scalars. We prove 1.-3. in Proposition 1.1 first.

1. $\mathbf{a} + \mathbf{b}$ has components $a_i + b_i$, and $\mathbf{b} + \mathbf{a}$ has components $b_i + a_i$. Since $a_i + b_i = b_i + a_i$ (when adding numbers the order doesn't matter), all components are equal, so that $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$.
2. $\mathbf{a} + \mathbf{b}$ has components $a_i + b_i$, and $(\mathbf{a} + \mathbf{b}) + \mathbf{c}$ then has components $(a_i + b_i) + c_i$. In the same way $\mathbf{a} + (\mathbf{b} + \mathbf{c})$ has components $a_i + (b_i + c_i)$. Since

$$(a_i + b_i) + c_i = a_i + (b_i + c_i)$$

is known to hold for real numbers, it follows again that all components are equal, so that $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$.

3. $s(\mathbf{a} + \mathbf{b})$ has components $s(a_i + b_i)$, and $s\mathbf{a} + s\mathbf{b}$ has components $sa_i + sb_i$. $s(a_i + b_i) = sa_i + sb_i$ is also known to hold for real numbers, so that $s(\mathbf{a} + \mathbf{b}) = s\mathbf{a} + s\mathbf{b}$.

Let us now prove 1.-3. in Proposition 1.2.

1.

$$\mathbf{a} \cdot \mathbf{a} = (a_1, \dots, a_n) \cdot (a_1, \dots, a_n) = a_1 \cdot a_1 + \dots + a_n \cdot a_n = a_1^2 + \dots + a_n^2 = |\mathbf{a}|^2$$

2.

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= (a_1, \dots, a_n) \cdot (b_1, \dots, b_n) = a_1 b_1 + \dots + a_n b_n \\ &= b_1 a_1 + \dots + b_n a_n = (b_1, \dots, b_n) \cdot (a_1, \dots, a_n) = \mathbf{b} \cdot \mathbf{a} \end{aligned}$$

3.

$$\begin{aligned}
\mathbf{c} \cdot (\mathbf{a} + \mathbf{b}) &= (c_1, \dots, c_n) \cdot ((a_1, \dots, a_n) + (b_1, \dots, b_n)) \\
&= (c_1, \dots, c_n) \cdot (a_1 + b_1, \dots, a_n + b_n) \\
&= c_1(a_1 + b_1) + \dots + c_n(a_n + b_n) \\
&= c_1a_1 + c_1b_1 + \dots + c_na_n + c_nb_n \\
&= c_1a_1 + \dots + c_na_n + c_1b_1 + \dots + c_nb_n \\
&= (c_1, \dots, c_n) \cdot (a_1, \dots, a_n) + (c_1, \dots, c_n) \cdot (b_1, \dots, b_n) \\
&= \mathbf{c} \cdot \mathbf{a} + \mathbf{c} \cdot \mathbf{b}.
\end{aligned}$$

The second part of 3. is proved similarly.

Exercise 1.7. Show that $|\mathbf{sa}| = |s||\mathbf{a}|$ when $\mathbf{a} \in \mathbb{C}^n$ and $s \in \mathbb{C}$.

! Solution

We have that

$$\begin{aligned}
|\mathbf{sa}| &= |(sa_1, \dots, sa_n)| = \sqrt{|sa_1|^2 + \dots + |sa_n|^2} \\
&= \sqrt{|s|^2|a_1|^2 + \dots + |s|^2|a_n|^2} = |s|\sqrt{|a_1|^2 + \dots + |a_n|^2} = |s||\mathbf{a}|.
\end{aligned}$$

Exercise 1.8 (Reverse triangle inequality). Prove that $||\mathbf{a}| - |\mathbf{b}|| \leq |\mathbf{a} - \mathbf{b}|$ for all $\mathbf{a}, \mathbf{b} \in \mathbb{C}^n$. This is also called the *reverse triangle inequality*.

! Solution

From the triangle inequality it follows that

$$|\mathbf{a}| = |(\mathbf{a} - \mathbf{b}) + \mathbf{b}| \leq |\mathbf{a} - \mathbf{b}| + |\mathbf{b}|.$$

From this we get $|\mathbf{a}| - |\mathbf{b}| \leq |\mathbf{a} - \mathbf{b}|$. If we let $\mathbf{a}$ and $\mathbf{b}$ swap roles we also get $|\mathbf{b}| - |\mathbf{a}| \leq |\mathbf{b} - \mathbf{a}| = |\mathbf{a} - \mathbf{b}|$. Both $|\mathbf{b}| - |\mathbf{a}|$ and $|\mathbf{a}| - |\mathbf{b}|$ are also $\leq |\mathbf{a} - \mathbf{b}|$. But then it follows that

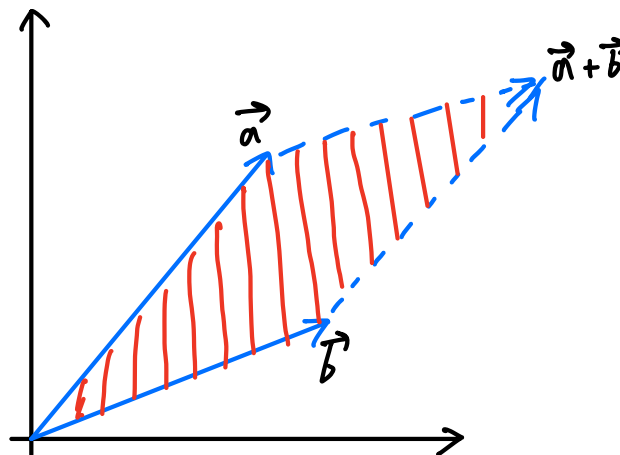
$$||\mathbf{a}| - |\mathbf{b}|| \leq |\mathbf{a} - \mathbf{b}|.$$

Exercise 1.9. By the *parallelogram* spanned by the vectors $\mathbf{a}$ and $\mathbf{b}$ we mean the quadrilateral with vertices $\mathbf{0}$, $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{a} + \mathbf{b}$. Figure 1.5 illustrates a parallelogram in the plane.

The name comes from the fact that opposite sides are parallel.

- (a) Explain geometrically that the parallelogram spanned by $\mathbf{a}$ and $\mathbf{b}$ can be written as the set

$$\{r\mathbf{a} + s\mathbf{b} : 0 \leq r, s \leq 1\}.$$

Figure 1.5: Parallelogram spanned by the vectors $\mathbf{a}$ and $\mathbf{b}$.**! Solution**

Set $A = \{r\mathbf{a} + s\mathbf{b} : 0 \leq r, s \leq 1\}$. The line $r\mathbf{a} + s\mathbf{b}$, $0 \leq s \leq 1$ describes a line in the parallelogram which connects points $r\mathbf{a}$ and $r\mathbf{a} + \mathbf{b}$. Since this holds for all r it is clear that A is contained in the parallelogram.

It is also geometrically clear that, for every point in the parallelogram, there exists an r so that the point lies on the line through $r\mathbf{a}$ and $r\mathbf{a} + \mathbf{b}$. The parallelogram is therefore also contained in A , so that A must be identical to the whole parallelogram.

- (b) If $\mathbf{a}$ and $\mathbf{b}$ are vectors either in $\mathbb{R}^2$ or in $\mathbb{R}^3$, it gives meaning to talk about the angle between the two. Let us denote this by θ . Prove that the area of the parallelogram spanned by $\mathbf{a}$ and $\mathbf{b}$ is given by $|\mathbf{a}||\mathbf{b}|\sin\theta$ (we can assume that $0 \leq \theta \leq \pi$, so that $\sin\theta \geq 0$).

! Solution

We can think of $|\mathbf{a}|$ as the base of the parallelogram, and $|\mathbf{b}|\sin\theta$ as the height. The area is given by base times height, i.e., $|\mathbf{a}||\mathbf{b}|\sin\theta$.

Exercise 1.10 (The parallelogram law). Show that, for $\mathbf{a}, \mathbf{b} \in \mathbb{C}^n$, we have that

$$|\mathbf{a} + \mathbf{b}|^2 + |\mathbf{a} - \mathbf{b}|^2 = 2(|\mathbf{a}|^2 + |\mathbf{b}|^2).$$

This identity is called the *parallelogram law*. Can you think of why the law is called this?

! Solution

$\mathbf{a} + \mathbf{b}$ and $\mathbf{a} - \mathbf{b}$ correspond to the diagonals in the parallelogram spanned by $\mathbf{a}$ and $\mathbf{b}$. Since $\mathbf{a}$ and $\mathbf{b}$ correspond to the sides in the parallelogram, the law gives us a connection between the diagonals and sides in a parallelogram. This explains how this identity obtained its name.

Exercise 1.11. Suppose that f is a real function which takes two vectors in $\mathbb{R}^n$ as parameters. Suppose that

1. $f(\mathbf{a}, \mathbf{a}) = |\mathbf{a}|^2$
2. $f(\mathbf{a}, \mathbf{b}) = f(\mathbf{b}, \mathbf{a})$
3. $f(\mathbf{c}, \mathbf{a} + \mathbf{b}) = f(\mathbf{c}, \mathbf{a}) + f(\mathbf{c}, \mathbf{b})$ and $f(\mathbf{a} + \mathbf{b}, \mathbf{c}) = f(\mathbf{a}, \mathbf{c}) + f(\mathbf{b}, \mathbf{c})$

for all vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ in $\mathbb{R}^n$. Show that

$$f(\mathbf{a}, \mathbf{b}) = \mathbf{a} \cdot \mathbf{b}$$

for all $\mathbf{a}, \mathbf{b}$. In particular it follows that $f(s\mathbf{a}, \mathbf{b}) = f(\mathbf{a}, s\mathbf{b}) = sf(\mathbf{a}, \mathbf{b})$, since we know that $(s\mathbf{a}) \cdot \mathbf{b} = \mathbf{a} \cdot (s\mathbf{b}) = s(\mathbf{a} \cdot \mathbf{b})$ holds.

! Solution

We have that

$$\begin{aligned} |\mathbf{a} + \mathbf{b}|^2 &= f(\mathbf{a} + \mathbf{b}, \mathbf{a} + \mathbf{b}) = f(\mathbf{a}, \mathbf{a}) + f(\mathbf{a}, \mathbf{b}) + f(\mathbf{b}, \mathbf{a}) + f(\mathbf{b}, \mathbf{b}) \\ &= |\mathbf{a}|^2 + 2f(\mathbf{a}, \mathbf{b}) + |\mathbf{b}|^2. \end{aligned}$$

It follows that

$$f(\mathbf{a}, \mathbf{b}) = \frac{1}{2}(|\mathbf{a} + \mathbf{b}|^2 - |\mathbf{a}|^2 - |\mathbf{b}|^2) = \frac{1}{2} \sum_{k=1}^n (a_k + b_k)^2 - a_k^2 - b_k^2 = \sum_{k=1}^n a_k b_k.$$

Exercise 1.12. Consider the following code. What does the code do? Justify your answer. Consult the `numpy` documentation on the function `np.where`.

```
import numpy as np

a = np.arange(0,100); a[0] = -1; a[1]=-1
for k in range(2,10):
    a[(2*k):k] = -1
a[np.where(a!=-1)]
```

! Solution

The code prints out all prime numbers less than 100. First the numbers themselves are filled in. Non-prime numbers are iteratively marked with

-1, and we start by marking 0 and 1 as non-primes. The `for`-loop marks $2k, 3k, \dots$, and so on, as non-primes, since they are all divisible by k . After the loop all non-primes have a -1-entry. The remaining entries are the primes, and are found with help of the `np.where`-command, and printed out. Note that any non-prime number less than 100 must be divisible with a number less than 10. The code exploits this.

Chapter 2

Matrices and matrix multiplication

In a vector all components are listed sequentially. In a *matrix* they are listed in two directions: vertically and horizontally. We will use uppercase letters to represent matrices. If A is a matrix one writes a_{ij} for the component in row i and column j , and we use parenthesis to indicate the start and end of the matrix:

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

It is also assumed that all rows have an equal number of components, and similarly for the columns. We say that A is an $m \times n$ -matrix (or has dimension $m \times n$) if it has m rows and n columns. A is called *square* if $m = n$.

A horizontal segment in A is called a *row vector*, a vertical segment a *column vector*. A vector can be interpreted both as a row vector, and as a column vector. In the code

```
A = np.array([[1,2,3]])  
B = np.array([[1],[2],[3]])
```

A is a row vector (or a 1×3 -matrix), B a column vector (or a 3×1 -matrix). We will as before write a_i for the components in a vector, regardless of whether it is a row vector or a column vector. Rows and columns will be indexed from 0 and upwards, just as for vectors:

```
A[0,1]
```

```
B[1,0]
```

2

In many math books it is common to interpret a vector as a column vector, if nothing else is said. When writing $\mathbf{a} = (a_1, a_2, \dots, a_n)$ we shall therefore interpret $\mathbf{a}$ as a column vector.

The 3×4 -matrix $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$ is obtained by writing

```
A = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
```

To find the dimensions of A write

```
m, n = np.shape(A)
m, n
```

(3, 4)

Sometimes we will write $\mathbf{a}_{i,:}$ for the row vector obtained from row i in A , and $\mathbf{a}_{:,j}$ for the column vector obtained from column j in A .

```
A[i,:]
A[:,j]
```

Square brackets are thus used to indicate the part of the matrix we want. Colon here stands for “all components”. Note the differences in the following two calls:

```
A[1,:]
```

```
array([5, 6, 7, 8])
```

```
A[[1],:]
```

```
array([[5, 6, 7, 8]])
```

The latter should be used if the result is to be used in computations involving matrices.

There is also a more general syntax for extracting a *sub-matrix* of a matrix. The code below extracts a 2×3 -matrix consisting of all components from row 1 and 3, which also are in column 1, 3, and 4.

```
A[np.ix_([0,2],[0,2,3])]
```

```
array([[ 1,  3,  4],
       [ 9, 11, 12]])
```

If A is an $m \times n$ -matrix we define the *transpose* of A as the $n \times m$ -matrix where component (i, j) is $a_{j,i}$. We write A^T for the transpose of A . For the matrix

$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$ above we get that $A^T = \begin{pmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{pmatrix}$. Matrices are

easily transposed as follows:

A.T

```
array([[ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11],
       [ 4,  8, 12]])
```

Hvis $A = A^T$, A is said to be *symmetric*. In particular

$$A = \begin{pmatrix} 5 & 2 & 3 \\ 2 & 6 & 4 \\ 3 & 4 & 7 \end{pmatrix}.$$

is symmetric. Symmetric matrices are always square, and the components are symmetric about the diagonal. The diagonal in this example consists of the numbers 5, 6, 7.

The transpose satisfies simple rules:

$$(A + B)^T = A^T + B^T \quad (A - B)^T = A^T - B^T \quad (A^T)^T = A \quad (sA)^T = sA^T$$

Some simple matrices will turn out to be useful.

- The *zero-matrix*, written $\mathbf{0}$, is the matrix where all components are 0.
- The *ones-matrix*, written $\mathbf{1}$, is the matrix where all components are 1.
- A *diagonal matrix* is a matrix where there are only zeroes outside the diagonal.
- The *identity matrix*, written I_n , is the $n \times n$ -diagonal matrix where all diagonal entries equal 1.

These can be obtained as follows:

```
np.zeros((m,n)) # Creates an mxn matrix of zeros
np.ones((m,n))  # Creates an mxn matrix of ones
np.diag(a)       # Creates a diagonal matrix with the components
                  # of a on the diagonal.
np.eye(n)        # Creates an nxn identity matrix
```

Many operations give meaning both for vectors and matrices:

```
np.max(A,axis=0)
```

```
array([ 9, 10, 11, 12])
```

```
np.min(A,axis=0)

array([1, 2, 3, 4])

np.sum(A,axis=0)

array([15, 18, 21, 24])

np.prod(A,axis=0)

array([ 45, 120, 231, 384])
```

The difference from the vector version is that the operations are performed column by column.

2.1 Vectorised operations on matrices

As for vectors, matrix addition and subtraction are vectorised: If A and B are both $m \times n$ then

- $A + B$ is the matrix with component (i, j) being $a_{ij} + b_{ij}$,
- $A - B$ is the matrix with component (i, j) being $a_{ij} - b_{ij}$,
- tA is the matrix with component (i, j) being $ta_{i,j}$ (where t is a scalar).

We thus have

$$A + B = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{pmatrix}$$

It is a simple exercise to check that the properties in Proposition 1.1 also hold if we replace vectors with matrices.

Example 2.1. We have that

$$\begin{aligned} 3 \begin{pmatrix} 1 & 4 \\ 2 & 1 \end{pmatrix} + 2 \begin{pmatrix} -1 & 2 \\ -2 & 1 \end{pmatrix} &= \begin{pmatrix} 3 \cdot 1 & 3 \cdot 4 \\ 3 \cdot 2 & 3 \cdot 1 \end{pmatrix} + \begin{pmatrix} 2 \cdot (-1) & 2 \cdot 2 \\ 2 \cdot (-2) & 2 \cdot 1 \end{pmatrix} \\ &= \begin{pmatrix} 3 & 12 \\ 6 & 3 \end{pmatrix} + \begin{pmatrix} -2 & 4 \\ -4 & 2 \end{pmatrix} = \begin{pmatrix} 3-2 & 12+4 \\ 6-4 & 3+2 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 16 \\ 2 & 5 \end{pmatrix} \end{aligned}$$



Also for matrices Python allows vectorised operations with a scalar: If we write

$A + b$ where A is a matrix and b a scalar,

$$\begin{pmatrix} a_{11} + b & a_{12} + b & \cdots & a_{1n} + b \\ a_{21} + b & a_{22} + b & \cdots & a_{2n} + b \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b & a_{m2} + b & \cdots & a_{mn} + b \end{pmatrix}$$

is returned. In particular will $A - 1$ return the matrix where 1 is subtracted in all components, and $1/A$ will return the matrix with components $1/a_{ij}$.

Vectorised functions are defined in the same way as for vectors. Let us consider some examples.

Example 2.2. We saw previously that the operations `sqrt`, `abs`, `exp`, and `log`, were vectorised. This is also the case for matrix input. This is easily seen for trigonometric functions:

$$\sin \begin{pmatrix} 0 & \pi/2 \\ \pi & 3\pi/2 \end{pmatrix} = \begin{pmatrix} \sin 0 & \sin(\pi/2) \\ \sin \pi & \sin(3\pi/2) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix}$$

This is simple to verify:

```
np.sin(np.array([[0,np.pi/2],[np.pi,3*np.pi/2]]))
```

```
array([[ 0.0000000e+00,  1.0000000e+00],
       [ 1.2246468e-16, -1.0000000e+00]])
```



Example 2.3. `max` and `min` are also vectorised for matrix input.

```
A = np.array([[1,2,3],[4,5,6]])
B = np.array([[4,5,6],[1,2,3]])
np.maximum(A,B)
```

```
array([[4, 5, 6],
       [4, 5, 6]])
```

```
np.minimum(A,B)
```

```
array([[1, 2, 3],
       [1, 2, 3]])
```

The same applies for multiplication and taking powers.

```
A*B
```

```
array([[ 4, 10, 18],
       [ 4, 10, 18]])
```

```
A**B
```

```
array([[ 1, 32, 729],
       [ 4, 25, 216]])
```



A special syntax exists when A is a column vector (**a**) and B a row vector (**b**): **a + b** will then return

$$\begin{pmatrix} a_1 + b_1 & a_1 + b_2 & \cdots & a_1 + b_n \\ a_2 + b_1 & a_2 + b_2 & \cdots & a_2 + b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_m + b_1 & a_m + b_2 & \cdots & a_m + b_n \end{pmatrix},$$

i.e., the matrix with components (i, j) equal to $a_i + b_j$. It is not really defined mathematically to add a column vector and a row vector, but some programming languages still allow this. This may sound strange, but it can be useful in some cases, since it can give very compact code. Let us give some examples.

Example 2.4 (The Hilbert matrix). The *Hilbert matrix* H_n is defined as the $n \times n$ -matrix with components

$$h_{kl} = 1/(k + l - 1),$$

where $1 \leq k, l \leq n$. In particular

$$H_4 = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{pmatrix}.$$

The matrix H_n can be obtained with help of the function `scipy.hilbert(n)`, but let us instead create it with the syntax above. First we write

```
y = np.arange(1,5)
y
```

```
array([1, 2, 3, 4])
```

Here a vector with 4 components was created, without interpretation in rows and columns. With the command `reshape` this can be interpreted as a row vector (a 1×4 -matrix):

```
y.reshape((1,-1))
```

```
array([[1, 2, 3, 4]])
```

or a column vector (a 4×1 -matrix):

```
y.reshape((-1,1))
```

```
array([[1],
       [2],
       [3],
       [4]])
```

The number -1 here means that the number of rows/columns is calculated automatically, from the requirement that there should be only one row/column. The Hilbert matrix can now be obtained as

```
x = y.reshape((1,-1))
1/(x + x.T - 1)
```

```
array([[1.          , 0.5          , 0.33333333, 0.25          ],
       [0.5          , 0.33333333, 0.25          , 0.2          ],
       [0.33333333, 0.25          , 0.2          , 0.16666667],
       [0.25          , 0.2          , 0.16666667, 0.14285714]])
```

If we, the other way, would like to convert a row- or column vector back to a vector (without interpretation of rows/columns), this can also be done with the help of `reshape`:

```
x.reshape(-1)
```

```
array([1, 2, 3, 4])
```

We will return to the Hilbert matrix later, as it will turn out to be quite vulnerable to roundoff errors on a computer. As a result it is difficult to use numerically.



Example 2.5 (The Vandermonde matrix). The matrix

$$V = \begin{pmatrix} 1 & x_1 & \dots & x_1^n \\ 1 & x_2 & \dots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \dots & x_m^n \end{pmatrix} \quad (2.1)$$

is called a *Vandermonde matrix*, and $x_1, \dots, x_m$ are called its *generators*. They are thus obtained by assembling the powers of the generators in each row. Such matrices need not be square - the number of generators (m) can be different from the number of powers ($n + 1$). Vandermonde matrices will appear later in connection with interpolation, and numerical differentiation and -integration. If we collect the generators of the vandermonde matrix in the column vector $\mathbf{x}$, and the powers $0, \dots, n$ in the row vector $\mathbf{p}$, we get the matrix in Equation 2.1 by writing

```
p = np.arange(0,n+1).reshape((1,-1));
V = x**p
```



Example 2.6 (The discrete Fourier transform). Matrices can, as vectors, have complex components. The *DFT matrix*, F_N , is an example of this. It is defined as the $N \times N$ -vandermonde matrix with generators $\{e^{-2\pi i k/N}\}_{k=0}^{N-1}$. DFT_N thus has components $e^{-2\pi i kn/N}$, where $0 \leq k, n < N$, and it can be obtained with the code

```
p = np.arange(0,N).reshape((1,-1))
F = np.exp( -2*np.pi*1j*p.T*p )
```

Here there is vectorised multiplication of a column vector with a row vector in the exponent, giving us all possible values for kn . The exponential function is also vectorised, i.e., e^A gives the matrix with components (k,n) equal to $e^{a_{kn}}$.

The Fourier matrix has a very useful application to sound. In this setting N can be very large, so large that the matrix may not fit in the memory of the computer! It turns out that the Fourier matrix can be rewritten so that it can be used for computations after all. To be more precise, the function `np.fft.fft(x)` computes the multiplication $F_N \mathbf{x}$, without actually writing down the full matrix F_N (`fft` here stands for *Fast Fourier Transform*). Here we should add some comments.

Both $\mathbf{x}$ and $F_N \mathbf{x}$ are column vectors. But it turns out that `np.fft.fft(x)` actually accepts a vector as input, and returns a vector. If one wants to compute the matrix-vector multiplication $F_N \mathbf{x}$ (i.e., with both input and output assumed to be column vectors), one should instead call `np.fft.fft(x, axis=0)`. These two things return the same, but the shape is either that of a vector, or that of a matrix:

```
x = np.array([1,2,3,4])
np.fft.fft(x)
```

```
array([10.+0.j, -2.+2.j, -2.+0.j, -2.-2.j])
```

```
x = x.reshape((-1,1))
np.fft.fft(x, axis=0)
```

```
array([[10.+0.j],
       [-2.+2.j],
       [-2.+0.j],
       [-2.-2.j]])
```

Many other functions in `numpy` accept this form where the inputs and output are *flattened* to vectors. Since we will heavily use linear algebra notation, we will usually reshape the input to a column vector before we multiply with a matrix to get the resulting column vector. ♣

2.2 Matrix multiplication

Later we will look at how one can solve linear systems with m equations and n unknowns. A general such system can be written

$$\begin{array}{cccccc} a_{11}x_1 & +a_{12}x_2 & +\cdots & +a_{1n}x_n & = & b_1 \\ a_{21}x_1 & +a_{22}x_2 & +\cdots & +a_{2n}x_n & = & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1}x_1 & +a_{m2}x_2 & +\cdots & +a_{mn}x_n & = & b_m \end{array} \quad (2.2)$$

Let us write

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

A is called the *coefficient matrix* of the system. In order to write Equation 2.2 more compactly the following definition is useful.

Definition 2.1 (Matrix-vector multiplication). Let A be an $m \times n$ -matrix, and $\mathbf{x}$ a column vector with n components. We define $A\mathbf{x}$ as the vector

$$A\mathbf{x} = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{pmatrix} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_n\mathbf{a}_n.$$

With this definition Equation 2.2 can be written more compactly as $A\mathbf{x} = \mathbf{b}$. The rewriting to $x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_n\mathbf{a}_n$ above follows from the properties of vectors. Note that $A\mathbf{x}$ is defined if and only if the number of columns in A equals the number of components in $\mathbf{x}$, and that $A\mathbf{x}$ then can be interpreted as m scalar products (between the rows in A , and $\mathbf{x}$).

Example 2.7. Let us compute $A\mathbf{x}$ when $A = \begin{pmatrix} 2 & 1 & -2 \\ 3 & 0 & -1 \\ 7 & -4 & 2 \end{pmatrix}$ and $\mathbf{x} = \begin{pmatrix} 3 \\ -1 \\ -2 \end{pmatrix}$.

This is well-defined since the number of columns in A equals the number of components in $\mathbf{x}$ (both are 3). We get

$$\begin{pmatrix} 2 & 1 & -2 \\ 3 & 0 & -1 \\ 7 & -4 & 2 \end{pmatrix} \begin{pmatrix} 3 \\ -1 \\ -2 \end{pmatrix} = \begin{pmatrix} 2 \cdot 3 + 1 \cdot (-1) + (-2) \cdot (-2) \\ 3 \cdot 3 + 0 \cdot (-1) + (-1) \cdot (-2) \\ 7 \cdot 3 + (-4) \cdot (-1) + 2 \cdot (-2) \end{pmatrix} = \begin{pmatrix} 6 - 1 + 4 \\ 9 + 0 + 2 \\ 21 + 4 - 4 \end{pmatrix} = \begin{pmatrix} 9 \\ 11 \\ 21 \end{pmatrix}$$

♣

Suppose we want to solve Equation 2.2 for different right hand sides $\mathbf{b}_1, \dots, \mathbf{b}_k$:

$$A\mathbf{x}_1 = \mathbf{b}_1 \quad A\mathbf{x}_2 = \mathbf{b}_2 \quad \cdots \quad A\mathbf{x}_k = \mathbf{b}_k$$

We seek a compact notation also for the collection of these systems. For this the following definition is useful.

Definition 2.2 (Matrix multiplication). If A is an $m \times n$ matrix, and X an $n \times k$ -matrix, we define AX as the $m \times k$ -matrix

$$AX = (AX_{:,1} \quad AX_{:,2} \quad \cdots \quad AX_{:,k}) \quad (2.3)$$

If we now set

$$X = (\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_k) \quad B = (\mathbf{b}_1 \quad \mathbf{b}_2 \quad \cdots \quad \mathbf{b}_k),$$

Column j in the equation $AX = B$ is $A\mathbf{x}_j = \mathbf{b}_j$. $AX = B$ can also be interpreted as the assembly of k systems. AX is only defined when the number of columns in A equals the number of rows in X .

Let us see how a matrix product can be computed. From the definition of the matrix-vector product we know that $AX_{:,j} = \sum_{r=1}^n a_{ir}x_{rj}$. Equation 2.3 now says that this equals column j in AX , and it follows that

$$(AX)_{ij} = \sum_{r=1}^n a_{ir}x_{rj}. \quad (2.4)$$

This is also called the *inner form* of a matrix product.

Example 2.8. Let us compute the matrix product

$$\begin{pmatrix} 2 & 4 & 3 \\ 1 & -1 & 3 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ 4 & 1 \\ 2 & -1 \end{pmatrix}$$

The product should be a 2×2 -matrix. If we use Equation 2.4 to compute component $(1,1)$ in this we get

$$2 \cdot 2 + 4 \cdot 4 + 3 \cdot 2 = 4 + 16 + 6 = 26$$

If we do the same for component $(1,2)$ we get

$$2 \cdot (-1) + 4 \cdot 1 + 3 \cdot (-1) = -2 + 4 - 3 = -1.$$

We have thus computed the first row in the matrix product. It is common to write out these calculations as follows.

$$\begin{aligned} \begin{pmatrix} 2 & 4 & 3 \\ 1 & -1 & 3 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ 4 & 1 \\ 2 & -1 \end{pmatrix} &= \begin{pmatrix} 2 \cdot 2 + 4 \cdot 4 + 3 \cdot 2 & 2 \cdot (-1) + 4 \cdot 1 + 3 \cdot (-1) \\ 1 \cdot 2 - 1 \cdot 4 + 3 \cdot 2 & 1 \cdot (-1) - 1 \cdot 1 + 3 \cdot (-1) \end{pmatrix} \\ &= \begin{pmatrix} 4 + 16 + 6 & -2 + 4 - 3 \\ 2 - 4 + 6 & -1 - 1 - 3 \end{pmatrix} = \begin{pmatrix} 26 & -1 \\ 4 & -5 \end{pmatrix} \end{aligned}$$

We can verify this with the built-in method for matrix multiplication.

```
A = np.array([[2,4,3],[1,-1,3]])
B = np.array([[2,-1],[4,1],[2,-1]])
A @ B
```

```
array([[26, -1],
       [ 4, -5]])
```

Matrix multiplication can also be interpreted in terms of the scalar product: Above we computed 4 scalar products: If the left hand side matrix has rows $\mathbf{a}_1$ and $\mathbf{a}_2$, and the right hand side matrix has columns $\mathbf{b}_1$ and $\mathbf{b}_2$, we computed

$$\mathbf{a}_1 \cdot \mathbf{b}_1 = 26 \quad \mathbf{a}_1 \cdot \mathbf{b}_2 = -1 \quad \mathbf{a}_2 \cdot \mathbf{b}_1 = 4 \quad \mathbf{a}_2 \cdot \mathbf{b}_2 = -5.$$

♣

Example 2.9 (The outer product). Let $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$ be column vectors. We have that

$$\mathbf{a}\mathbf{x}^T = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} = \begin{pmatrix} a_1x_1 & a_1x_2 & \cdots & a_1x_n \\ a_2x_1 & a_2x_2 & \cdots & a_2x_n \\ \vdots & \vdots & \ddots & \vdots \\ a_mx_1 & a_mx_2 & \cdots & a_mx_n \end{pmatrix}$$

$\mathbf{a}\mathbf{x}^T$ is also called the *outer product* of $\mathbf{a}$ and $\mathbf{x}$. ♣

Matrix multiplication satisfies several known properties:

Proposition 2.1 (Properties of matrix multiplication). *Let A , B , and C be matrices. The following hold*

1. $(AB)C = A(BC)$
2. $A(B + C) = AB + AC$
3. $(B + C)A = BA + CA$
4. For all scalars s we have that $s(AB) = (sA)B = A(sB)$.
5. If A is $m \times n$ then $AI_n = A$ and $I_m A = A$ (so that the identity matrix behaves in the same way as the number 1 when multiplying real numbers).

It is required that the matrices have matching dimensions, so that the products are defined.

If we multiply many matrices, 1. here implies that it does not matter which ‘neighbouring matrices’ we multiply first. It therefore does not matter where we set the parenthesis when performing matrix multiplication. Therefore, the parenthesis are usually omitted, i.e., one writes ABC instead of $(AB)C = A(BC)$. This, however, does not mean that we can change the order of the matrices: If A is $m \times n$ and B is $n \times m$, it is clear that AB and BA both are defined, but they have different dimensions when $m \neq n$. Even if $m = n$, we have usually that $AB \neq BA$, as the following examples shows:

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

Thus, while the order does not matter when multiplying scalars, this is not the case for matrices!

Proof.

1. is perhaps the most difficult one to prove, since one obtains a “double sum” when multiplying three matrices. Let us therefore take a closer look at this. If A is $m \times k$, B $k \times l$, C $l \times n$, then both $(AB)C$ and $A(BC)$ are well-defined. Furthermore, AB is $m \times l$, and BC is $k \times n$, and both $(AB)C$ and $A(BC)$ are $m \times n$. The definition of matrix multiplication gives

$$\begin{aligned} ((AB)C)_{ij} &= \sum_{s=1}^l (AB)_{is} c_{sj} = \sum_{s=1}^l \left(\sum_{r=1}^k a_{ir} b_{rs} \right) c_{sj} = \sum_{s=1}^l \sum_{r=1}^k a_{ir} b_{rs} c_{sj} \\ (A(BC))_{ij} &= \sum_{r=1}^k a_{ir} (BC)_{rj} = \sum_{r=1}^k a_{ir} \left(\sum_{s=1}^l b_{rs} c_{sj} \right) = \sum_{r=1}^k \sum_{s=1}^l a_{ir} b_{rs} c_{sj}. \end{aligned}$$

A couple of comments are in place:

- in the first equality on the first line the definition of the matrix product of AB and C was used,
- in the first equality on the second line the definition of the matrix product of A and BC was used,
- in the second equality on the first line the definition of the matrix product AB was used,
- in the second equality on the second line the definition of the matrix product BC was used.

The order of summation here does not matter, so that we get the same result. Therefore, all the components in $(AB)C$ and $A(BC)$ are equal, so that $(AB)C = A(BC)$.

2.-4. is somewhat simpler to prove, and is given as an exercise.

5.: We have that

$$(AI_n)_{ij} = \sum_{r=1}^n a_{ir} (I_n)_{rj} = a_{ij} (I_n)_{jj} = a_{ij},$$

where we used that I_n has only one non-zero in every column. Since all components are equal it follows that $AI_n = A$. In the same way one shows that $I_m A = A$. $\square$

There is also another useful form for a matrix product, which we also will return to. This form uses the notation $\mathbf{a}_{\cdot r}$ for column r in A , and $\mathbf{x}_r$ for row r in X :

Proposition 2.2 (Outer form for a matrix product). *We have that*

$$AX = \sum_{r=1}^n \mathbf{a}_{:r} \mathbf{x}_{r:}.$$

This is also called the outer form of a matrix product.

Proof. Let $\mathbf{e}_i$ be the column vector which has 1 in component i , and 0 elsewhere. From Example 2.9 it follows that

- $\mathbf{a}_{:j} \mathbf{e}_j^T$ is the matrix of the same size as A , where column j equals column j in A , and where all other columns are $\mathbf{0}$.
- $\mathbf{e}_i \mathbf{x}_{i:}$ is the matrix with the same size as X , where row i equals row i in X , and where all other rows are $\mathbf{0}$.

This means that we can write $A = \sum_{j=1}^n \mathbf{a}_{:j} \mathbf{e}_j^T$, and $X = \sum_{i=1}^n \mathbf{e}_i \mathbf{x}_{i:}$. If we use this we get

$$AX = \left(\sum_{j=1}^n \mathbf{a}_{:j} \mathbf{e}_j^T \right) \left(\sum_{i=1}^n \mathbf{e}_i \mathbf{x}_{i:} \right) = \sum_{i=1}^n \sum_{j=1}^n \mathbf{a}_{:j} \mathbf{e}_j^T \mathbf{e}_i \mathbf{x}_{i:} = \sum_{r=1}^n \mathbf{a}_{:r} \mathbf{x}_{r:}.$$

where we used that $\mathbf{e}_j^T \mathbf{e}_i = 1$ if $i = j$, and 0 otherwise. This completes the proof. $\square$

In the proof above several of the properties from Proposition 2.1 were used. Which ones?

2.3 Linear transformations and matrix multiplication

A mapping $\mathbf{F}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$ is called *linear* if we have that, for all $\mathbf{a}, \mathbf{a}_1, \mathbf{a}_2 \in \mathbb{R}^n$, and all scalars c ,

1. $\mathbf{F}(\mathbf{a}_1 + \mathbf{a}_2) = \mathbf{F}(\mathbf{a}_1) + \mathbf{F}(\mathbf{a}_2)$
2. $\mathbf{F}(c\mathbf{a}) = c\mathbf{F}(\mathbf{a})$

From Proposition 2.1 it follows that the mapping $\mathbf{x} \rightarrow A\mathbf{x}$, where A is a matrix, is linear. The other way, we will now show that all linear transformations can be written on this form:

Proposition 2.3. *Suppose $\mathbf{F}$ is a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$. There exists an $m \times n$ -matrix A so that $\mathbf{F}(\mathbf{x}) = A\mathbf{x}$ for all $\mathbf{x} \in \mathbb{R}^n$. More precisely,*

$$A = (\mathbf{F}(\mathbf{e}_1) \quad \mathbf{F}(\mathbf{e}_2) \quad \cdots \quad \mathbf{F}(\mathbf{e}_n)), \quad (2.5)$$

where $\mathbf{e}_i$ was the column vector with 1 in component i , and 0 otherwise.

Proof. For the proof we will use that

$$(x_1, x_2, \dots, x_n) = x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + \dots + x_n \mathbf{e}_n,$$

which is easy to show. This gives

$$\begin{aligned} \mathbf{F}(x_1, x_2, \dots, x_n) &= \mathbf{F}(x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + \dots + x_n \mathbf{e}_n) \\ &= x_1 \mathbf{F}(\mathbf{e}_1) + x_2 \mathbf{F}(\mathbf{e}_2) + \dots + x_n \mathbf{F}(\mathbf{e}_n). \end{aligned}$$

By the definition of matrix-vector multiplication this can be written

$$(\mathbf{F}(\mathbf{e}_1) \quad \mathbf{F}(\mathbf{e}_2) \quad \dots \quad \mathbf{F}(\mathbf{e}_n)) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

It follows that all linear transformations can be written on the form $\mathbf{x} \rightarrow A\mathbf{x}$, with A given by Equation 2.5. $\square$

Thus, one only needs compute $\mathbf{F}(\mathbf{e}_i)$ to find the matrix of a linear transformation. Let us take a look at some useful linear transformations, and find their matrices.

Example 2.10 (Projections). From the properties of the scalar product it follows that the projection operator $\text{proj}_{\mathbf{b}}(\mathbf{a}) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b}$ is linear. The matrix of $\text{proj}_{\mathbf{b}}$ is

$$\begin{aligned} &(\text{proj}_{\mathbf{b}}(\mathbf{e}_1) \quad \text{proj}_{\mathbf{b}}(\mathbf{e}_2) \quad \dots \quad \text{proj}_{\mathbf{b}}(\mathbf{e}_n)) \\ &= \left(\frac{\mathbf{e}_1 \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b} \quad \frac{\mathbf{e}_2 \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b} \quad \dots \quad \frac{\mathbf{e}_n \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b} \right) = \left(\frac{b_1}{|\mathbf{b}|} \frac{\mathbf{b}}{|\mathbf{b}|} \quad \frac{b_2}{|\mathbf{b}|} \frac{\mathbf{b}}{|\mathbf{b}|} \quad \dots \quad \frac{b_n}{|\mathbf{b}|} \frac{\mathbf{b}}{|\mathbf{b}|} \right) \\ &= \left(\frac{\mathbf{b}}{|\mathbf{b}|} \frac{b_1}{|\mathbf{b}|} \quad \frac{\mathbf{b}}{|\mathbf{b}|} \frac{b_2}{|\mathbf{b}|} \quad \dots \quad \frac{\mathbf{b}}{|\mathbf{b}|} \frac{b_n}{|\mathbf{b}|} \right) = \frac{\mathbf{b}}{|\mathbf{b}|} \begin{pmatrix} b_1 & b_2 & \dots & b_n \end{pmatrix} = \frac{\mathbf{b}}{|\mathbf{b}|} \left(\frac{\mathbf{b}}{|\mathbf{b}|} \right)^T. \end{aligned}$$

The second last equality used the definition of matrix multiplication, while the

last equality used that $\frac{\mathbf{b}}{|\mathbf{b}|} = \begin{pmatrix} b_1/|\mathbf{b}| \\ b_2/|\mathbf{b}| \\ \vdots \\ b_n/|\mathbf{b}| \end{pmatrix}$, so that

$\left(\frac{\mathbf{b}}{|\mathbf{b}|} \right)^T = \left(\frac{b_1}{|\mathbf{b}|} \quad \frac{b_2}{|\mathbf{b}|} \quad \dots \quad \frac{b_n}{|\mathbf{b}|} \right)$. It follows that, if $\mathbf{b}$ is a vector with length 1, the matrix $\mathbf{b}\mathbf{b}^T$ computes the projection onto the line through the origin, with direction vector $\mathbf{b}$. Such matrices are therefore called *projection matrices*. ♣

Example 2.11 (Rotations). Consider the mapping $\mathbf{F}$ which rotates a vector in $\mathbb{R}^2$ with angle θ . If we use polar form we get

$$\begin{aligned} \mathbf{F} \begin{pmatrix} r \cos \alpha \\ r \sin \alpha \end{pmatrix} &= \begin{pmatrix} r \cos(\alpha + \theta) \\ r \sin(\alpha + \theta) \end{pmatrix} = \begin{pmatrix} \cos \theta r \cos \alpha - \sin \theta r \sin \alpha \\ \sin \theta r \cos \alpha + \cos \theta r \sin \alpha \end{pmatrix} \\ &= \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} r \cos \alpha \\ r \sin \alpha \end{pmatrix} \end{aligned}$$

It follows that $\mathbf{F}$ is linear, with matrix $\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$. Assuming that $\mathbf{F}$ is linear, it is easy to find the matrix of $\mathbf{F}$:

$$\mathbf{F}(\mathbf{e}_1) = \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} \quad \mathbf{F}(\mathbf{e}_2) = \begin{pmatrix} \cos(\theta + \pi/2) \\ \sin(\theta + \pi/2) \end{pmatrix} = \begin{pmatrix} -\sin \theta \\ \cos \theta \end{pmatrix}.$$



2.4 Plotting polynomials

Matrix multiplication can be used naturally to plot polynomials. The function `plt.plot` can be used to plot curves in the plane. The common syntax for `plt.plot` takes two arguments `x` and `y`, which contain the x - and y -coordinates of the plot points, respectively. The `plt.plot` function accepts many variants. For instance, if we want to plot the polynomials

$$q_1(x) = x \quad q_2(x) = x^2 \quad q_3(x) = x^3$$

together, with m points distributed over the interval $[0, 1]$, this can be achieved as follows:

```
x = np.linspace(0,1,m).reshape((-1,1))
plt.plot(x, x)
plt.plot(x, x**2)
plt.plot(x, x**3)
plt.legend(['q_1', 'q_2', 'q_3'])
```

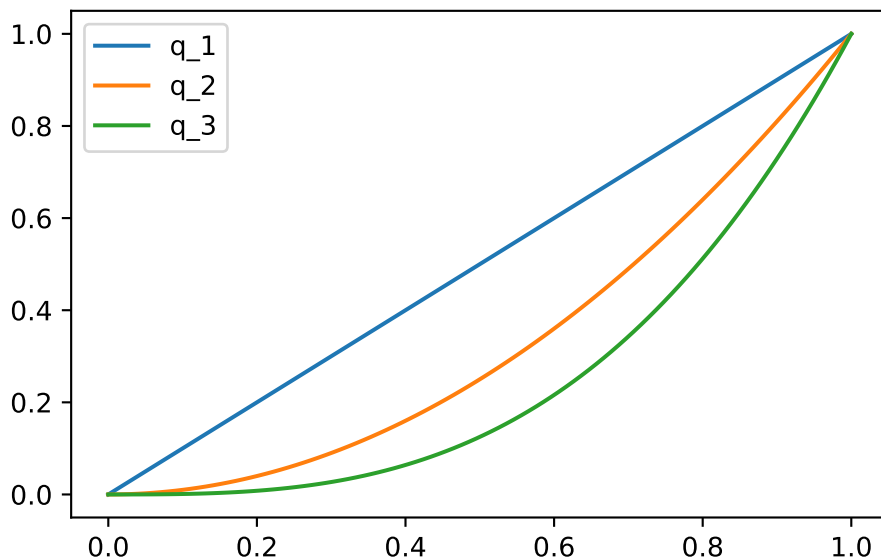


Figure 2.1: Plot of the polynomials $q_1(x) = x$, $q_2(x) = x^2$, and $q_3(x) = x^3$.

The `plt.plot` commands above can also be assembled into one call as follows.

```
plt.plot(x, x, x, x**2, x, x**3)
```

where 6 arguments are given, and where each pair represents a plot. The result can be seen in Figure 2.1. If the polynomials are of higher degree, and with many terms, things get more complicated. Let us take a look at how we can do this systematically by assembling the polynomial coefficients in vectors. Let the first polynomial be

$$q_1(x) = c_0 + c_1x + \cdots + c_nx^n,$$

and let the plot points be $x_1, \dots, x_m$. We write

$$\begin{aligned} q_1(x_1) &= c_0 + x_1c_1 + \cdots + x_1^nc_n \\ q_1(x_2) &= c_0 + x_2c_1 + \cdots + x_2^nc_n \\ &\vdots \\ q_1(x_m) &= c_0 + x_mc_1 + \cdots + x_m^nc_n. \end{aligned}$$

By the definition of matrix-vector-multiplication, this can be written

$$\begin{pmatrix} q_1(x_1) \\ q_1(x_2) \\ \vdots \\ q_1(x_m) \end{pmatrix} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix}, \quad (2.6)$$

where we recognise the Vandermonde matrix from Example 2.5. If we now have another polynomial, $q_2(x) = d_0 + d_1x + \cdots + d_nx^n$, we can in the same way write

$$\begin{pmatrix} q_2(x_1) \\ q_2(x_2) \\ \vdots \\ q_2(x_m) \end{pmatrix} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^n \end{pmatrix} \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_n \end{pmatrix}. \quad (2.7)$$

By the definition of matrix multiplication Equation 2.6 and Equation 2.7 can be assembled into

$$\begin{pmatrix} q_1(x_1) & q_2(x_1) \\ q_1(x_2) & q_2(x_2) \\ \vdots & \vdots \\ q_1(x_m) & q_2(x_m) \end{pmatrix} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^n \end{pmatrix} \begin{pmatrix} c_0 & d_0 \\ c_1 & d_1 \\ \vdots & \vdots \\ c_n & d_n \end{pmatrix}, \quad (2.8)$$

and in the same way if we have even more polynomials. The Vandermonde matrix on the right side can be computed as follows:

```
p = np.arange(0,n+1).reshape((1,-1))
V = x**p
```

We collect the coefficients of the three polynomials q_1 , q_2 , and q_3 , in column vectors `q1`, `q2`, and `q3`, and compute Equation 2.8:

```
q1 = np.array([[0],[1],[0],[0]])
q2 = np.array([[0],[0],[1],[0]])
q3 = np.array([[0],[0],[0],[1]])
y = V @ np.block([q1,q2,q3])
```

This gives a matrix with three columns, where column 1 contains values for q_1 , column 2 for q_2 , and column 3 for q_3 . The polynomials can now be plotted together by writing

```
plt.plot(x, y)
plt.legend(['q_1', 'q_2', 'q_3'])
```

This is a new syntax for the `plt.plot`-function, where

- the first argument is the column vectors with plot points, and
- the second argument is the matrix with as many rows as in the first argument, and 3 columns.

If $\mathbf{x}$ is a column vector and $\mathbf{Y}$ a matrix with as many rows as $\mathbf{x}$ and 3 columns, then `plt.plot(x,Y)` is the same as writing

```
plt.plot(x, V @ q1, x, V @ q2, x, V @ q3)
```

This also applies more generally if we have more columns with polynomials: `plt.plot(x,Y)` plots all columns of $\mathbf{Y}$ against the vector $\mathbf{x}$.

2.5 Block matrices

Matrices can also be used as building blocks for bigger matrices. If we have four matrices A , B , C , D , we write

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right) \quad (2.9)$$

for the matrix where the components in these four matrices are placed over, under, to the left, or to the right of each other as indicated. Lines are used to separate the blocks. We call this a *block matrix*. Block matrices can be created as follows:

```
np.block([[A,B],[C,D]])
```

Equation 2.9 only gives meaning if

- A and B have equally many rows,
- C and D have equally many rows,

for then these can be put next to each other. In the same way A and C must have equally many columns, B and D have equally many columns, for then they can be placed under one-another. It is easy to generalise to block matrices with more than four blocks.

Example 2.12. If

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} \quad C = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{pmatrix} \quad D = \begin{pmatrix} 4 & 4 \\ 4 & 4 \\ 4 & 4 \end{pmatrix}$$

then

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right) = \begin{pmatrix} 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 \\ 3 & 3 & 3 & 4 & 4 \\ 3 & 3 & 3 & 4 & 4 \\ 3 & 3 & 3 & 4 & 4 \end{pmatrix}$$

This is easily verified:

```
A = np.array([[1,1,1],[1,1,1]])
B = np.array([[2,2],[2,2]])
C = np.array([[3,3,3],[3,3,3],[3,3,3]])
D = np.array([[4,4],[4,4],[4,4]])
np.block([[A,B],[C,D]])
```

```
array([[1, 1, 1, 2, 2],
       [1, 1, 1, 2, 2],
       [3, 3, 3, 4, 4],
       [3, 3, 3, 4, 4],
       [3, 3, 3, 4, 4]])
```



The product

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right) \left(\begin{array}{c|c} E & F \\ \hline G & H \end{array} \right) \quad (2.10)$$

of two block matrices give meaning if

- the number of columns in A equals the number of rows in E ,
- the number of columns in B equals the number of rows in G .

Then the product can be computed as

```
np.block([[A,B],[C,D]]) @ np.block([[E,F],[G,H]])
```

Equation 2.10 can also be computed in terms of ordinary matrix multiplication:

$$\left(\begin{array}{c|c} AE + BG & AF + BH \\ \hline CE + DG & CF + DH \end{array} \right)$$

The code then becomes

```
np.block([[A@E + B@G, A@F + B@H], [C@E + D@G, C@F + D@H]])
```

Later we will in particular use the following, where the order of the matrices is switched:

$$\left(\begin{array}{c|c} A & B \end{array} \right) \begin{pmatrix} E \\ G \end{pmatrix} = \left(\begin{array}{c|c} B & A \end{array} \right) \begin{pmatrix} G \\ E \end{pmatrix} = AE + BG \quad (2.11)$$

Exercises

Exercise 2.1. Compute

$$(a) \begin{pmatrix} 2 & 4 & 3 \\ -1 & 2 & 4 \\ 4 & 1 & -1 \end{pmatrix} - \begin{pmatrix} -1 & 2 & 8 \\ -1 & -2 & -5 \\ 2 & 4 & -2 \end{pmatrix}$$

! Solution

We have that

$$\begin{aligned} & \begin{pmatrix} 2 & 4 & 3 \\ -1 & 2 & 4 \\ 4 & 1 & -1 \end{pmatrix} - \begin{pmatrix} -1 & 2 & 8 \\ -1 & -2 & -5 \\ 2 & 4 & -2 \end{pmatrix} \\ &= \begin{pmatrix} 2 - (-1) & 4 - 2 & 3 - 8 \\ -1 - (-1) & 2 - (-2) & 4 - (-5) \\ 4 - 2 & 1 - 4 & -1 - (-2) \end{pmatrix} \begin{pmatrix} 3 & 2 & -5 \\ 0 & 4 & 9 \\ 2 & -3 & 1 \end{pmatrix} \end{aligned}$$

```
A = np.array([[2,4,3],[-1,2,4],[4,1,-1]])
B = np.array([[-1,2,8],[-1,-2,-5],[2,4,-2]])
A - B
array([[ 3,  2, -5],
       [ 0,  4,  9],
       [ 2, -3,  1]])
```

$$(b) \ 2 \begin{pmatrix} 3 & 2 \\ -1 & -2 \end{pmatrix} + 4 \begin{pmatrix} -3 & 3 \\ -1 & -1 \end{pmatrix}.$$

! Solution

$$\begin{aligned}
& 2 \begin{pmatrix} 3 & 2 \\ -1 & -2 \end{pmatrix} + 4 \begin{pmatrix} -3 & 3 \\ -1 & -1 \end{pmatrix} \\
&= \begin{pmatrix} 2 \cdot 3 & 2 \cdot 2 \\ 2 \cdot (-1) & 2 \cdot (-2) \end{pmatrix} + \begin{pmatrix} 4 \cdot (-3) & 4 \cdot 3 \\ 4 \cdot (-1) & 4 \cdot (-1) \end{pmatrix} \\
&= \begin{pmatrix} 6 & 4 \\ -2 & -4 \end{pmatrix} + \begin{pmatrix} -12 & 12 \\ -4 & -4 \end{pmatrix} \\
&= \begin{pmatrix} 6-12 & 4+12 \\ -2-4 & -4-4 \end{pmatrix} = \begin{pmatrix} -6 & 16 \\ -6 & -8 \end{pmatrix}
\end{aligned}$$

```

A = np.array([[3,2],[-1,-2]])
B = np.array([[-3,3],[-1,-1]])
2*A + 4*B
array([[ -6, 16],
       [ -6, -8]])

```

Exercise 2.2. Transpose the following matrices:

(a) $\begin{pmatrix} 2 & 4 \\ 3 & -1 \\ 0 & 1 \end{pmatrix}$

! Solution

$$\begin{pmatrix} 2 & 4 \\ 3 & -1 \\ 0 & 1 \end{pmatrix}^T = \begin{pmatrix} 2 & 3 & 0 \\ 4 & -1 & 1 \end{pmatrix}$$

```

np.array([[2,4],[3,-1],[0,1]]).T
array([[ 2,  3,  0],
       [ 4, -1,  1]])

```

(b) $\begin{pmatrix} 1 & 4 & 7 \\ 2 & 4 & 3 \\ -1 & -2 & 1 \end{pmatrix}.$

! Solution

$$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 4 & 3 \\ -1 & -2 & 1 \end{pmatrix}^T = \begin{pmatrix} 1 & 2 & -1 \\ 4 & 4 & -2 \\ 7 & 3 & 1 \end{pmatrix}$$

```
np.array([[1,4,7],[2,4,3],[-1,-2,1]]).T
```

```
array([[ 1,  2, -1],
       [ 4,  4, -2],
       [ 7,  3,  1]])
```

Exercise 2.3. Compute the following matrix products.

(a) $\begin{pmatrix} 2 & 4 & 3 \\ 1 & -1 & -2 \\ -3 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 5 \\ 3 \end{pmatrix}$

! Solution

$$\begin{pmatrix} 2 & 4 & 3 \\ 1 & -1 & -2 \\ -3 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 5 \\ 3 \end{pmatrix} = \begin{pmatrix} 4 + 20 + 9 \\ 2 - 5 - 6 \\ -6 - 5 + 3 \end{pmatrix} = \begin{pmatrix} 33 \\ -9 \\ -8 \end{pmatrix}$$

```
A = np.array([[2,4,3],[1,-1,-2],[-3,-1,1]])
```

```
B = np.array([[2],[5],[3]])
```

```
A @ B
```

```
array([[33],
       [-9],
       [-8]])
```

(b) $\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 3 & 2 \end{pmatrix}$

! Solution

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 3 & 2 \end{pmatrix} = \begin{pmatrix} -1 + 9 & 0 + 6 \\ -2 + 12 & 0 + 8 \end{pmatrix} = \begin{pmatrix} 8 & 6 \\ 10 & 8 \end{pmatrix}$$

```
A = np.array([[1,3],[2,4]])
```

```
B = np.array([[-1,0],[3,2]])
```

```
A @ B
```

```
array([[ 8,  6],
       [10,  8]])
```

$$(c) \begin{pmatrix} -1 & 2 & 4 & 1 \\ 1 & 3 & 2 & 4 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 2 & 0 \\ 1 & 0 \\ 3 & 4 \end{pmatrix}$$

! Solution

$$\begin{pmatrix} -1 & 2 & 4 & 1 \\ 1 & 3 & 2 & 4 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 2 & 0 \\ 1 & 0 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} -2+4+4+3 & -1+0+0+4 \\ 2+6+2+12 & 1+0+0+16 \end{pmatrix} = \begin{pmatrix} 9 & 3 \\ 22 & 17 \end{pmatrix}$$

```
A = np.array([[ -1, 2, 4, 1], [1, 3, 2, 4]])
B = np.array([[2, 1], [2, 0], [1, 0], [3, 4]])
A @ B
array([[ 9,  3],
       [22, 17]])
```

Exercise 2.4. Let A be an $n \times n$ -matrix. Show that $\mathbf{x}^T A \mathbf{y} = 0$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ if and only if $A = 0$.

! Solution

If $A = 0$ it is clear that $\mathbf{x}^T A \mathbf{y} = 0$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. The other way, assume contrapositively that $A \neq 0$. Then there is some component $a_{ij} \neq 0$ of A . We have that $\mathbf{e}_i^T A \mathbf{e}_j = a_{ij} \neq 0$, so that we do not have that $\mathbf{x}^T A \mathbf{y} = 0$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. This concludes the proof.

Exercise 2.5. Let A be an $n \times n$ -matrix. Show that $(A\mathbf{x}) \cdot \mathbf{y} = \mathbf{x} \cdot (A\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ if and only if A is symmetric.

! Solution

We have that

$$(A\mathbf{x}) \cdot \mathbf{y} = (A\mathbf{x})^T \mathbf{y} = \mathbf{x}^T A^T \mathbf{y} \quad \mathbf{x} \cdot (A\mathbf{y}) = \mathbf{x}^T A \mathbf{y}.$$

If A is symmetric we see that these two expressions are equal for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. The other way, assume contrapositively that A is not symmetric. We can then find indices i, j so that $a_{ij} \neq a_{ji}$. Then we have that $\mathbf{e}_i^T A \mathbf{e}_j = a_{ij}$, and $\mathbf{e}_i^T A^T \mathbf{e}_j = (A^T)_{ij} = a_{ji}$. Since $a_{ij} \neq a_{ji}$ we do not have that $(A\mathbf{x}) \cdot \mathbf{y} = \mathbf{x} \cdot (A\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. This concludes the proof.

Exercise 2.6. The function `np.random.rand(m,n)` returns an $m \times n$ -matrix

with random components between 0 and 1, Consider the code

```
m = 10
n = 7
A = np.random.rand(m,n)
B = np.random.rand(m,n)
C = np.random.rand(m,n)
print( ((A+B)+C) - (A+(B+C)) )
```

Which matrix property does this code verify?

! Solution

The code checks whether $(A + B) + C = A + (B + C)$ is correctly for three given matrices. If the number 0 is displayed, the computer thinks the two are equal. In practice this will not happen. Due to round-off errors on the computer, the computer will not arrive at 0 exactly.

Exercise 2.7. Show that the properties in Proposition 1.1 also hold if we replace the vectors $\mathbf{a}$ and $\mathbf{b}$ with matrices.

! Solution

1. $A + B$ has components $a_{ij} + b_{ij}$, and $B + A$ has components $b_{ij} + a_{ij}$. Again we use that the order of addition of numbers doesn't matter. All components in $A + B$ and $B + A$ are thus equal, so that also $A + B = B + A$.
2. $(A + B) + C$ has components $(a_{ij} + b_{ij}) + c_{ij}$, and $A + (B + C)$ has components $a_{ij} + (b_{ij} + c_{ij})$. Again we can use a known property of numbers, so that the components in the left and right sides are equal, so that $(A + B) + C = A + (B + C)$.
3. $s(A + B)$ has components $s(a_{ij} + b_{ij})$, and $sA + sB$ has components $sa_{ij} + sb_{ij}$. Also now we can use a known property for numbers to conclude that the left and right hand sides are equal.

Exercise 2.8. Prove property 2.-4. in Proposition 2.1.

! Solution

We prove property 2. first. Component (i, j) in $A(B + C)$ is

$$\sum_k a_{ik}(B + C)_{kj} = \sum_k a_{ik}(b_{kj} + c_{kj}). \quad (2.12)$$

Component (i, j) in AB is $\sum_k a_{ik}b_{kj}$, and component (i, j) in AC is $\sum_k a_{ik}c_{kj}$. By the definition of matrix addition component (i, j) in $AB +$

AC is then

$$\sum_k a_{ik}b_{kj} + \sum_k a_{ik}c_{kj} \quad (2.13)$$

From known properties of numbers we know that Equation 2.12 and Equation 2.13 are equal. $A(B + C)$ and $AB + AC$ therefore have equal components, so that $A(B + C) = AB + AC$. Property 3. follows in the same way.

Finally we show property 4..

- Component (i, j) in $s(AB)$ is $s \sum_k a_{ik}b_{kj}$.
- Component (i, j) in $(sA)B$ is $\sum_k (sa_{ik})b_{kj}$.
- Component (i, j) in $A(sB)$ is $\sum_k a_{ik}(sb_{kj})$.

That these three are equal now follows from properties of real numbers.

Exercise 2.9. How many arithmetic operations are required by matrix multiplication in general?

! Solution

Let A be an $m \times n$ -matrix, and B an $n \times k$ -matrix. The matrix AB will then have mk entries, and each entry is computed as a scalar product of vectors with n components. Each such scalar product requires n multiplications and $n - 1$ additions, so a total of $2n - 1$ operations. The total is thus $(2n - 1)mk$ operations. For matrix-vector multiplication $k = 1$, so that $(2n - 1)m$ operations are required.

Exercise 2.10. We write $A^n = AA \cdots A$, i.e., the matrix where we multiply in A n times. Compute A^2 , A^3 and A^4 when $A = \begin{pmatrix} 1 & 4 \\ -2 & -1 \end{pmatrix}$.

! Solution

We have

$$A^2 = \begin{pmatrix} 1 & 4 \\ -2 & -1 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ -2 & -1 \end{pmatrix} = \begin{pmatrix} 1-8 & 4-4 \\ -2+2 & -8+1 \end{pmatrix} = \begin{pmatrix} -7 & 0 \\ 0 & -7 \end{pmatrix}$$

$$A^3 = A^2A = \begin{pmatrix} -7 & 0 \\ 0 & -7 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ -2 & -1 \end{pmatrix} = \begin{pmatrix} -7 & -28 \\ 14 & 7 \end{pmatrix}$$

$$A^4 = A^2A^2 = \begin{pmatrix} -7 & 0 \\ 0 & -7 \end{pmatrix} \begin{pmatrix} -7 & 0 \\ 0 & -7 \end{pmatrix} = \begin{pmatrix} 49 & 0 \\ 0 & 49 \end{pmatrix}.$$

```
A = np.array([[1,4],[-2,-1]])
for k in range(2,5):
    print(np.linalg.matrix_power(A,k))
```

```
[[ -7   0]
```

```

[ 0 -7]]
[[ -7 -28]
 [ 14  7]]
[[49  0]
 [ 0 49]]

```

Exercise 2.11 (Mirroring). Let $\mathbf{F}$ be the linear transformation from $\mathbb{R}^2$ to $\mathbb{R}^2$ which mirrors a point about the line $y = x$. Find the matrix of $\mathbf{F}$.

! Solution

We have that $\mathbf{F}(\mathbf{e}_1) = \mathbf{e}_2$, and that $\mathbf{F}(\mathbf{e}_2) = \mathbf{e}_1$. It follows that the matrix of $\mathbf{F}$ is

$$(\mathbf{F}(\mathbf{e}_1) \quad \mathbf{F}(\mathbf{e}_2)) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Exercise 2.12. Show that $(AB)^T = B^T A^T$, given that A has as many columns as B has rows.

! Solution

We have that component (i, j) in $(AB)^T$ equals component (j, i) in AB , which is $\sum_k a_{jk} b_{ki}$.

We also have that component (i, j) in $B^T A^T$ is

$$\sum_k (B^T)_{ik} (A^T)_{kj} = \sum_k b_{ki} a_{jk} = \sum_k a_{jk} b_{ki}.$$

It follows that the matrices $(AB)^T$ and $B^T A^T$ are equal in all components, so that $(AB)^T = B^T A^T$.

Exercise 2.13. Assume that $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})$ is a real vector, and let $\mathbf{y} = F_N \mathbf{x}$. Show that $y_{N-n} = \overline{y_n}$ for $1 \leq n \leq N-1$.

! Solution

We compute

$$\begin{aligned} \overline{y_n} &= \overline{\sum_{k=0}^{N-1} x_k e^{-2\pi i k n / N}} = \sum_{k=0}^{N-1} \overline{x_k e^{-2\pi i k n / N}} = \sum_{k=0}^{N-1} \overline{x_k} e^{2\pi i k n / N} \\ y_{N-n} &= \sum_{k=0}^{N-1} x_k e^{-2\pi i k (N-n) / N} = \sum_{k=0}^{N-1} x_k e^{2\pi i k n / N} \end{aligned}$$

where we used that $\mathbf{x}$ was real.

Chapter 3

Gaussian elimination and reduced echelon form

A general systems with m equations and n unknowns has the form

$$\begin{array}{cccccc} a_{11}x_1 & +a_{12}x_2 & +\cdots & +a_{1n}x_n & = & b_1 \\ a_{21}x_1 & +a_{22}x_2 & +\cdots & +a_{2n}x_n & = & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ a_{m1}x_1 & +a_{m2}x_2 & +\cdots & +a_{mn}x_n & = & b_m \end{array}$$

Gaussian elimination solves such a system by systematically eliminating variables from left to right. To explain this further, suppose that the variables $x_1, \dots, x_n$ already have been eliminated. There are now two possibilities:

1. x_{n+1} does not appear in the remaining equations. Then x_{n+1} has been eliminated as well, and we proceed with the elimination of x_{n+2} .
2. x_{n+1} appears in at least one of the remaining equations. We do the following:
 - (a) Take one of the equations where x_{n+1} appears, and place this first among the remaining equations.
 - (b) Scale this equation so that there is a 1 in front of x_{n+1} . We shall call this a leading one *leading one* (there should be no contribution left of x_{n+1} in this equation).
 - (c) Add multiples of this equation to the other remaining equations so that x_{n+1} gets eliminated.

We have now eliminated x_{n+1} , and proceed with the elimination of x_{n+2} .

Some concrete examples will clarify this

Example 3.1 (A system with a unique solution). Consider the system

$$\begin{array}{rrcr} & -8x_2 & +3x_3 & = & 6 \\ 4x_1 & -3x_2 & +2x_3 & = & 8 \\ x_1 & -2x_2 & +x_3 & = & 3 \end{array}$$

First we eliminate x_1 . Since x_1 does not contribute in the first equation we swap the first and third equation and get

$$\begin{array}{rrcr} x_1 & -2x_2 & +x_3 & = & 3 \\ \textcolor{red}{4}x_1 & -3x_2 & +2x_3 & = & 8 \\ & -8x_2 & +3x_3 & = & 6 \end{array}$$

We also need to ensure that there is a leading one in the first equation, but this is already the case. We now eliminate x_1 in the second equation by subtracting 4 times the first equation (vi mark in red what we eliminate). Then we get

$$\begin{array}{rrcr} x_1 & -2x_2 & +x_3 & = & 3 \\ & 5x_2 & -2x_3 & = & -4 \\ & -8x_2 & +3x_3 & = & 6 \end{array}$$

We then make a leading one in the second equation:

$$\begin{array}{rrcr} x_1 & -2x_2 & +x_3 & = & 3 \\ & x_2 & -\frac{2}{5}x_3 & = & -\frac{4}{5} \\ & \textcolor{red}{-8}x_2 & +3x_3 & = & 6 \end{array}$$

We eliminate x_2 in the third equation:

$$\begin{array}{rrcr} x_1 & -2x_2 & +x_3 & = & 3 \\ & x_2 & -\frac{2}{5}x_3 & = & -\frac{4}{5} \\ & & -\frac{1}{5}x_3 & = & -\frac{2}{5} \end{array}$$

We make a leading one in the third equation:

$$\begin{array}{rrcr} x_1 & -2x_2 & +\textcolor{red}{1}x_3 & = & 3 \\ & x_2 & -\frac{2}{5}x_3 & = & -\frac{4}{5} \\ & & x_3 & = & 2 \end{array}$$

We now say that the system is on *echelon form*, meaning that:

- each equation is either zero or has a leading one,
- each equation begins with at least one more 0-coefficient than the equation above.

When we have arrived at an echelon form, we are finished eliminating variables left to right. We proceed by eliminating variables from right to left: x_3 can be eliminated in the first two equations by adding multiples of the third equation. If we perform both these operations we get

$$\begin{array}{rrcr} x_1 & \textcolor{red}{-2}x_2 & & = & 1 \\ & x_2 & & = & 0 \\ & & x_3 & = & 2 \end{array}$$

Finally we eliminate x_2 in the first equation by adding a multiple of the second equation:

$$\begin{array}{rcl} x_1 & = & 1 \\ x_2 & = & 0 \\ x_3 & = & 2 \end{array}$$

We thus have the unique solution $\mathbf{x} = (x_1, x_2, x_3) = (1, 0, 2)$. The system is now on what we call *reduced echelon form*: It is on echelon form, and in addition each variable with a leading one appears in just one equation. These are also called *pivot variables*. We also call the position of a leading one a *pivot element*.

♣

To simplify what we did above it can be useful to gather the coefficients and the right hand side in a matrix - without writing down equality signs and variable names. The matrix $(A \quad \mathbf{b})$ is called the *augmented matrix* of the system $A\mathbf{x} = \mathbf{b}$. From the example above we have that

$$A = \begin{pmatrix} 0 & -8 & 3 \\ 4 & -3 & 2 \\ 1 & -2 & 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 6 \\ 8 \\ 3 \end{pmatrix} \quad (A \quad \mathbf{b}) = \begin{pmatrix} 0 & -8 & 3 & 6 \\ 4 & -3 & 2 & 8 \\ 1 & -2 & 1 & 3 \end{pmatrix}$$

An equation can be thought of as a row, a column as the coefficients associated with a variable. The columns in A which have a pivot element are called *pivot columns*. In the same way, the rows in A which have a pivot element are called *pivot rows*. If we let $\sim$ denote the operations we did above, everything can be written more compactly as:

$$\begin{aligned} \begin{pmatrix} 0 & -8 & 3 & 6 \\ 4 & -3 & 2 & 8 \\ 1 & -2 & 1 & 3 \end{pmatrix} &\sim \begin{pmatrix} 1 & -2 & 1 & 3 \\ \color{red}{4} & -3 & 2 & 8 \\ 0 & -8 & 3 & 6 \end{pmatrix} \sim \begin{pmatrix} 1 & -2 & 1 & 3 \\ 0 & 5 & -2 & -4 \\ 0 & -8 & 3 & 6 \end{pmatrix} \\ &\sim \begin{pmatrix} 1 & -2 & 1 & 3 \\ 0 & 1 & -\frac{2}{5} & -\frac{4}{5} \\ 0 & \color{red}{-8} & 3 & 6 \end{pmatrix} \sim \begin{pmatrix} 1 & -2 & 1 & 3 \\ 0 & 1 & -\frac{2}{5} & -\frac{4}{5} \\ 0 & 0 & -\frac{1}{5} & -\frac{2}{5} \end{pmatrix} \\ &\sim \begin{pmatrix} 1 & -2 & \color{red}{1} & 3 \\ 0 & 1 & -\frac{2}{5} & -\frac{4}{5} \\ 0 & 0 & 1 & 2 \end{pmatrix} \sim \begin{pmatrix} 1 & \color{red}{-2} & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{pmatrix} \\ &\sim \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{pmatrix} \end{aligned}$$

After the first five operations the matrix is on echelon form. The last matrix is on reduced echelon form.

We can bring a general matrix to reduced echelon form with the function `rref` in the class `sym.Matrix`. If we test this on the matrix above we get

```
C, cols = sym.Matrix([[0,-8,3,6],[4,-3,2,8],[1,-2,1,3]]).rref()
C
```

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

We see that we obtained the same matrix. This is not obvious, since there are many different ways to bring a matrix to reduced echelon form, and each such can in principle lead to different reduced forms.

We shall later see that the reduced echelon form is unique - regardless of the operations, we will always end up with the same reduced echelon form.

In Equation 3.1 we indicated the operations we did. Let us establish some notation for the first operations.

1. The first operation swapped row 1 and 3. We will write this as $I \leftrightarrow III$.
2. The second operation subtracted 4 times row 1 in row 2. We will write this as $II - 4I$.
3. The third operation multiplied row 2 with $1/5$. We will write this as $II \cdot (1/5)$.

With the help of this notation Equation 3.1 can be rewritten as follows:

$$\begin{aligned} \begin{pmatrix} 0 & -8 & 3 & 6 \\ 4 & -3 & 2 & 8 \\ 1 & -2 & 1 & 3 \end{pmatrix} &\stackrel{I \leftrightarrow III}{\sim} \begin{pmatrix} 1 & -2 & 1 & 3 \\ 4 & -3 & 2 & 8 \\ 0 & -8 & 3 & 6 \end{pmatrix} \stackrel{II - 4I}{\sim} \begin{pmatrix} 1 & -2 & 1 & 3 \\ 0 & 5 & -2 & -4 \\ 0 & -8 & 3 & 6 \end{pmatrix} \\ &\stackrel{II \cdot (1/5)}{\sim} \begin{pmatrix} 1 & -2 & 1 & 3 \\ 0 & 1 & -\frac{2}{5} & -\frac{4}{5} \\ 0 & -8 & 3 & 6 \end{pmatrix} \stackrel{III + 8II}{\sim} \begin{pmatrix} 1 & -2 & 1 & 3 \\ 0 & 1 & -\frac{2}{5} & -\frac{4}{5} \\ 0 & 0 & -\frac{1}{5} & -\frac{2}{5} \end{pmatrix} \\ &\stackrel{III \cdot (-5)}{\sim} \begin{pmatrix} 1 & -2 & 1 & 3 \\ 0 & 1 & -\frac{2}{5} & -\frac{4}{5} \\ 0 & 0 & 1 & 2 \end{pmatrix} \stackrel{II + \frac{2}{5}III, I - III}{\sim} \begin{pmatrix} 1 & -2 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{pmatrix} \\ &\stackrel{I + 2II}{\sim} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{pmatrix} \end{aligned}$$

The system was thus brought to reduced row echelon form with help of three types of operations:

1. Add a multiple of one row to another,
2. multiply a row with a constant $\neq 0$,
3. Swap two rows.

These are called *elementary row operations*, and are easily implemented:

```
A[j,:] = A[j,:] + c*A[i,:]
A[i,:] = c*A[i,:]
A[[i,j],:] = A[[j,i],:]
```

The 3 first operations from Example 3.1 can in particular be programmed as follows:

```
A=np.array([[0,-8,3,6],[4,-3,2,8],[1,-2,1,3]])
A[[0,2],:] = A[[2,0],:]
A[1,:] = A[1,:] - 4*A[0,:]
A[1,:] = A[1,]/5
```

In practice Gaussian elimination is not implemented this way: The program itself find the row operations to be applied (see Section 3.3).

We also say that A and B are *row equivalent* if we can transform A to B through a series of elementary row operations. From the discussion above we get

$$\begin{pmatrix} 0 & -8 & 3 & 6 \\ 4 & -3 & 2 & 8 \\ 1 & -2 & 1 & 3 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{pmatrix}.$$

Every matrix is row equivalent to a matrix on reduced echelon form. Elementary row operations can also be expressed in terms of matrices:

Proposition 3.1. *Suppose B is obtained by applying an elementary row operation to the $m \times n$ -matrix A . Then $B = EA$ where E is the matrix obtained by applying the same row operation to the identity matrix I_m .*

Proof. Let $F(A)$ be the matrix obtained by applying the row operation to the matrix A . If we apply the row operation column-wise we get

$$F(A) = (F(\mathbf{a}_1) \quad F(\mathbf{a}_2) \quad \cdots \quad F(\mathbf{a}_n)).$$

Furthermore $\mathbf{a} \rightarrow F(\mathbf{a})$ is a linear transformation from $\mathbb{R}^m$ to $\mathbb{R}^m$ with matrix

$$(F(\mathbf{e}_1) \quad F(\mathbf{e}_2) \quad \cdots \quad F(\mathbf{e}_m)) = F(I_m).$$

It follows that

$$\begin{aligned} F(A) &= (F(I_m)\mathbf{a}_1 \quad F(I_m)\mathbf{a}_2 \quad \cdots \quad F(I_m)\mathbf{a}_n) \\ &= F(I_m)(\mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_n) = F(I_m)A, \end{aligned}$$

which is what we wanted to show. $\square$

A matrix is called *elementary* if it is obtained by applying an elementary row operation to the identity matrix. The following matrices are elementary, and correspond to the row operations where we add 2 times row 2 to row 1, multiply row 3 with 3, and swap rows 1 and 2, respectively:

$$E_1 = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad E_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix} \quad E_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (3.1)$$

Row operations can be reversed in the sense that one can find a row operation reversing what the first one did. The reverse row operations of Equation 3.1 are adding -2 times row 2 to row 1, multiplying row 3 with $1/3$, and swapping rows 1 and 2, respectively. The elementary matrices for these are

$$F_1 = \begin{pmatrix} 1 & -2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad F_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/3 \end{pmatrix} \quad F_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

That F_i reverses E_i means that $F_1E_1 = I_n$, $F_2E_2 = I_n$, $F_3E_3 = I_n$, which also can be verified by multiplying the matrices. From Proposition 3.1 we get the following.

Proposition 3.2. *For every matrix A we can write*

$$C = E_k \cdots E_2 E_1 A$$

where C is a matrix on reduced echelon form, and where $E_1, E_2, \dots, E_k$ are elementary matrices.

We also have the following result.

Proposition 3.3. *Assume a system is obtained from another by applying an elementary row operation. Then the two systems have the same solutions.*

Proof. Let E be the elementary matrix associated with the row operation, and F the elementary matrix associated with the reverse row operation. When we apply an elementary row operation to the original system $A\mathbf{x} = \mathbf{b}$ we get the new system $E A \mathbf{x} = E \mathbf{b}$.

- If $A\mathbf{x} = \mathbf{b}$ it is clear that also $E A \mathbf{x} = E \mathbf{b}$, so that $\mathbf{x}$ also solves the new system.
- If $E A \mathbf{x} = E \mathbf{b}$ we get by applying F on both sides that $A\mathbf{x} = \mathbf{b}$, so that $\mathbf{x}$ also solves the original system.

It follows that the original- and the new system have the same solutions. $\square$

Above we row reduced a system, and obtained a unique solution. Let us do the same for systems with no solution, and with infinitely many solutions.

Example 3.2 (A system with infinitely many solutions). Consider the system

$$\begin{array}{rrcr} x_1 & +x_2 & +x_3 & = & 4 \\ 2x_1 & & -x_3 & = & 2. \end{array}$$

We write down the augmented matrix, and apply elementary row operations:

$$\begin{pmatrix} 1 & 1 & 1 & 4 \\ 2 & 0 & -1 & 2 \end{pmatrix} \xrightarrow{II-2I} \begin{pmatrix} 1 & 1 & 1 & 4 \\ 0 & -2 & -3 & -6 \end{pmatrix} \xrightarrow{II \cdot (-1/2)} \begin{pmatrix} 1 & 1 & 1 & 4 \\ 0 & 1 & 3/2 & 3 \end{pmatrix} \xrightarrow{I-II} \begin{pmatrix} 1 & 0 & -1/2 & 1 \\ 0 & 1 & 3/2 & 3 \end{pmatrix}$$

Again we ended up with a reduced echelon form. In equation form this says

$$\begin{array}{rcl} x_1 & -\frac{1}{2}x_3 & = 1 \\ x_2 & +\frac{3}{2}x_3 & = 3 \end{array}$$

Which also can be written $x_1 = 1 + \frac{1}{2}x_3$, $x_2 = 3 - \frac{3}{2}x_3$. We see that x_3 can be chosen freely, and that a general solution can be written as

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 + \frac{1}{2}x_3 \\ 3 - \frac{3}{2}x_3 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 0 \end{pmatrix} + x_3 \begin{pmatrix} \frac{1}{2} \\ -\frac{3}{2} \\ 1 \end{pmatrix}$$

We see that there infinitely many solutions. x_1 and x_2 are pivot variables, and we see that these can be expressed with x_3 . x_3 is not a pivot variable since column 3 does not have a leading one. x_3 can be set to anything, and is therefore called a *free variable*. ♣

Example 3.3 (A system without solutions). Consider the system

$$\begin{array}{rcl} x_1 & +x_2 & = 2 \\ 3x_1 & +3x_2 & = 5 \end{array}$$

Row reduction of the augmented matrix gives

$$\begin{pmatrix} 1 & 1 & 2 \\ 3 & 3 & 5 \end{pmatrix} \xrightarrow{II-3I} \begin{pmatrix} 1 & 1 & 2 \\ 0 & 0 & -1 \end{pmatrix} \xrightarrow{II \cdot (-1)} \begin{pmatrix} 1 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} \xrightarrow{I-2II} \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Again we arrived at a reduced echelon form. Note here that, when eliminating x_1 , x_2 also got eliminated. In equation form this says that

$$\begin{array}{rcl} x_1 & +x_2 & = 0 \\ & & 0 = 1 \end{array}$$

$0 = 1$ is impossible, so that there are no solutions. ♣

The three examples above illustrate three possibilities: A system can have

1. a unique solution, as in Example 3.1. Here we had only pivot variables, and no pivot element in the last column.
2. infinitely many solutions, as in Example 3.2. Here we had a free variable, and no pivot element in the last column.
3. no solutions, as in Example 3.3. Here we had a pivot element in the last column.

Furthermore, we arrived at a reduced echelon form in all examples. These observations turn out to hold more generally:

Proposition 3.4. *Any system can be brought to reduced echelon form, and we have the following possibilities:*

1. *The system has no solutions. In this case the last column is a pivot column,*

2. The system has solutions. In this case the last column is not a pivot column and we have that
- (a) when there are free variables the system has infinitely many solutions (any value of the free variables give a solution),
 - (b) when there are no free variables (all columns except the last is a pivot column) the system has a unique solution.

From the discussion above we also have the following.

Theorem 3.1. *The following are equivalent for an $n \times n$ -matrix A :*

- 1. A is row equivalent with I_n .
- 2. $A\mathbf{x} = \mathbf{0}$ has $\mathbf{x} = \mathbf{0}$ as the only solution.
- 3. $A\mathbf{x} = \mathbf{b}$ has a unique solution for all right hand sides $\mathbf{b}$.
- 4. A can be written as the product of elementary matrices.

Later we will extend this result with more equivalences.

Proof. 1. $\Rightarrow$ 3.: Suppose that A can be row reduced to the identity matrix. We can solve $A\mathbf{x} = \mathbf{b}$ by row reducing the augmented matrix $(A \mid \mathbf{b})$, using the same row operations. At the end we then arrive at equations of the form $x_1 = b'_1, x_2 = b'_2, \dots, x_n = b'_n$, so that the system has a unique solution for all right hand sides $\mathbf{b}$.

3. $\Rightarrow$ 2. is obvious.

2. $\Rightarrow$ 1.: We solve $A\mathbf{x} = \mathbf{0}$ by row reducing the augmented matrix $(A \mid \mathbf{0})$. Since the system only has one solution, there can be no free variables. But then all columns in A are pivot columns. Since A is square, all rows must also be pivot rows. It follows that A is row equivalent with I_n .

1. $\Rightarrow$ 4.: That $A \sim I_n$ means that we can write $I_n = E_k \cdots E_1 A$, where E_i are elementary matrices. Let F_i be the elementary matrix which corresponds to the reverse row operation of E_i . If we multiply with F_k to the left on both sides we get that $F_k = E_{k-1} \cdots E_1 A$. If we then multiply with F_{k-1} to the left on both sides we get that $F_{k-1} F_k = E_{k-2} \cdots E_1 A$. If we continue this way we arrive in the end at $A = F_1 \cdots F_k$, so that A can be written as a product of elementary matrices.

4. $\Rightarrow$ 1.: If $A = F_1 \cdots F_k$ we can as above show that $I_n = E_k \cdots E_1 A$ (where again F_i represents the reverse row operation of E_i), so that $A \sim I_n$. $\square$

3.1 Gaussian elimination and complexity

Suppose we have a system with n equations and n unknowns. How many operations (additions, multiplications, etc.) are needed to solve the system, i.e., bring it to reduced echelon form? Suppose for simplicity that the coefficient matrix A is row equivalent with the identity matrix. Let us also assume that row interchanges do not cost anything (in terms of operations).

- In Gaussian elimination we ensure leading ones in the rows by dividing with the first non-zero. This requires

$$(n-1) + (n-2) + \cdots + 1 = (n-1)n/2$$

divisions.

- In the first part of Gaussian elimination we zero out the components below the diagonal, column by column. Under diagonal element k in column k there are $n-k$ rows. For each of these we multiply row k with a number (this corresponds to $n-k$ multiplications, since the row starts with $k-1$ zeroes), and add this to the row (this gives $n-k$ additions). Together we get $(n-k)((n-k) + (n-k)) = 2(n-k)^2$ operations here. Summing over all k we get

$$2 \sum_{k=1}^{n-1} (n-k)^2 = 2 \sum_{k=0}^n k^2 = n(n+1)(2n+1)/6 \approx 2n^3/3,$$

since $n^3/3$ is the dominating term when multiplying out $n(n+1)(2n+1)/6$ (which is the formula for the sum of the n første squares). The interest in the dominating term only comes from that we are usually interested in counting operations when n gets big - we are not interested in the exact number of operations, which typically is an uglier expression.

- In the second part of Gaussian elimination we zero out the components over the diagonal, column by column. No extra operations are needed for this.

Since $(n-1)n/2 \approx n^2/2$ is dominated by $2n^3/3$, we have in total about $2n^3/3$ operations. We have shown the following.

Proposition 3.5 (Number of operations in Gaussian elimination). *Suppose that $A \sim I_n$. About $2n^3/3$ operations are required to bring A to reduced echelon form.*

3.2 Simultaneous solution of systems

Suppose we want to solve two systems with the same coefficient matrix, $A\mathbf{x} = \mathbf{b}_1$ and $A\mathbf{x} = \mathbf{b}_2$. We bring A to reduced echelon form, C , and apply the same row operations to $\mathbf{b}_1$ and $\mathbf{b}_2$, and get $\mathbf{b}'_1$ and $\mathbf{b}'_2$. This corresponds to

$$(A \quad \mathbf{b}_1 \quad \mathbf{b}_2) \sim (C \quad \mathbf{b}'_1 \quad \mathbf{b}'_2),$$

since row reducing A , $\mathbf{b}_1$, and $\mathbf{b}_2$ separately can also be done by row reducing $(A \quad \mathbf{b}_1 \quad \mathbf{b}_2)$ as a whole. From $(C \quad \mathbf{b}'_1 \quad \mathbf{b}'_2)$ we can read out the solutions to the two systems. Let us consider an example.

Example 3.4. In Example 3.1 we solved the system $A\mathbf{x} = \mathbf{b}_1$ where $\mathbf{b}_1 = (6, 8, 3)$. By using the row operations from this example we can simultaneously solve the system $A\mathbf{x} = \mathbf{b}_2$ where $\mathbf{b}_2 = (1, 8, 2)$ as follows:

$$\begin{aligned}
(A \quad \mathbf{b}_1 \quad \mathbf{b}_2) &= \begin{pmatrix} 0 & -8 & 3 & 6 & 1 \\ 4 & -3 & 2 & 8 & 8 \\ 1 & -2 & 1 & 3 & 2 \end{pmatrix} \xrightarrow{I \leftrightarrow III} \begin{pmatrix} 1 & -2 & 1 & 3 & 2 \\ 4 & -3 & 2 & 8 & 8 \\ 0 & -8 & 3 & 6 & 1 \end{pmatrix} \xrightarrow{II - 4I} \begin{pmatrix} 1 & -2 & 1 & 3 & 2 \\ 0 & 5 & -2 & -4 & 0 \\ 0 & -8 & 3 & 6 & 1 \end{pmatrix} \\
&\xrightarrow{II \cdot (1/5)} \begin{pmatrix} 1 & -2 & 1 & 3 & 2 \\ 0 & 1 & -2/5 & -4/5 & 0 \\ 0 & -8 & 3 & 6 & 1 \end{pmatrix} \xrightarrow{III + 8II} \begin{pmatrix} 1 & -2 & 1 & 3 & 2 \\ 0 & 1 & -2/5 & -4/5 & 0 \\ 0 & 0 & -1/5 & -2/5 & 1 \end{pmatrix} \\
&\xrightarrow{III \cdot (-5)} \begin{pmatrix} 1 & -2 & 1 & 3 & 2 \\ 0 & 1 & -2/5 & -4/5 & 0 \\ 0 & 0 & 1 & 2 & -5 \end{pmatrix} \xrightarrow{II + 2/5 III, I - III} \begin{pmatrix} 1 & -2 & 0 & 1 & 7 \\ 0 & 1 & 0 & 0 & -2 \\ 0 & 0 & 1 & 2 & -5 \end{pmatrix} \\
&\xrightarrow{I + 2II} \begin{pmatrix} 1 & 0 & 0 & 1 & 3 \\ 0 & 1 & 0 & 0 & -2 \\ 0 & 0 & 1 & 2 & -5 \end{pmatrix}
\end{aligned}$$

It follows that $A\mathbf{x}_1 = \mathbf{b}_1$ as before has the solution $\mathbf{x}_1 = (1, 0, 2)$, but also that $A\mathbf{x}_2 = \mathbf{b}_2$ has the solution $\mathbf{x}_2 = (3, -2, -5)$. ♣

3.3 Implementation of Gaussian elimination

Let A be an $m \times n$ -matrix. Gaussian elimination is a bit complicated to implement, but consider the following code which is a skeleton for bringing a matrix to echelon form:

```

piv = np.zeros(m).astype(int) # Stores the positions of the pivots.
i, j = 0, 0
while i < m and j < n: # Start scan for next pivot element
    k = i
    while k < m and abs(A[k,j]) < 10**(-6):
        k += 1
    if k == m: # Need to go to the next column
        j += 1
    else: # Found a nonzero in the current column
        # The next pivot element is in row k. If k>i,
        # swap row i and k, so that the next pivot element is (i,j).
        # Create a leading one at pos. (i,j)
        # Zero out col. j below row i.
        piv[i] = j # The i'th leading one is in position (i,j)
        i += 1; j += 1

```

The initial position is $(0, 0)$. The outer **while**-loop scans for a next nonzero entry to be scaled to a leading one/a pivot element. First it looks for this below the current position (i) in the current column (the inner **while**-loop).

When applying row operations roundoff errors occur. Thus, we may get something nonzero, even if exactly zero should appear. The inner **while**-loop remedies this by assuming that a_{kj} is zero when $|a_{kj}| < 10^{-6}$.

If the search in the current column fails (the **if**-statement succeeds), the search moves to the next column (by continuing in the outer **while**-loop).

If the search in the current column succeeds, the **else**-statement is executed. Let us comment on how to implement the next three lines. If the nonzero (which is in row k) was found below the current row (row i), these rows are swapped:

```
A[[k,i]] = A[[i,k]]
```

We then create a leading one at position (i,j) :

```
A[i,j:] = A[i,j:]/A[i,j]
```

(i.e., $\mathbf{a}_i = \mathbf{a}_i/a_{ij}$, where $\mathbf{a}_i$ is row i in A). Note here that we only modified column j and upwards in row i . We did not need to divide with a_{ij} for columns $1, \dots, j-1$, as the entries here are zero. In column j we can zero out component k by subtracting a multiple of row i :

```
A[k,j:] = A[k,j:] - A[k,j] * A[i,j:]
```

(i.e., $\mathbf{a}_k = \mathbf{a}_k - a_{kj}\mathbf{a}_i$). We need to do this for $k = i+1, i+2, \dots, m$, i.e.,

$$\begin{aligned}\mathbf{a}_{i+1} &= \mathbf{a}_{i+1} - a_{(i+1)j}\mathbf{a}_i \\ \mathbf{a}_{i+2} &= \mathbf{a}_{i+2} - a_{(i+2)j}\mathbf{a}_i \\ &\vdots \\ \mathbf{a}_m &= \mathbf{a}_m - a_{mj}\mathbf{a}_i\end{aligned}$$

We here find $A_{(i+1):,j}$ on the left hand side, and on the right hand side the product

$$\begin{pmatrix} a_{(i+1)j} \\ a_{(i+2)j} \\ \vdots \\ a_{mj} \end{pmatrix} \mathbf{a}_i = A_{(i+1):,j} \mathbf{a}_i$$

This means that we can program this as

```
A[(i+1):,j:] = A[(i+1):,j:] - A[(i+1):,j] @ A[i,j:]
```

Once this has been done, the code stores the position of the leading one in row i (**piv**[i] = j), and moves to the next row/column. Once the outer **while**-loop finishes, we arrive at echelon form.

After arriving at echelon form, we can similarly zero out all components above the leading ones to arrive at the reduced echelon form. If the pivot in row i is in column j we get

$$\begin{aligned}\mathbf{a}_1 &= \mathbf{a}_1 - a_{1j}\mathbf{a}_i \\ \mathbf{a}_2 &= \mathbf{a}_2 - a_{2j}\mathbf{a}_i \\ &\vdots \\ \mathbf{a}_{i-1} &= \mathbf{a}_{i-1} - a_{(i-1)j}\mathbf{a}_i\end{aligned}$$

On the right hand side we recognise the product

$$\begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{(i-1)j} \end{pmatrix} \mathbf{a}_i = A_{:, (i-1), j} \mathbf{a}_i$$

so that this can be programmed as

```
j = piv[i]
A[:, (i-1), j:] = A[:, (i-1), j:] - A[:, (i-1), [j]] @ A[[i], j:]
```

We here needed the stored locations of the leading ones.

If we also add some print statements to track the steps in Gaussian elimination, we arrive at the following code which tests everything on the matrix from Example 3.1.

```
import numpy as np

A = np.array([[0., -8., 3., 6.], [4., -3., 2., 8.], [1., -2., 1., 3.]])
m, n = np.shape(A)
piv = np.zeros(m).astype(int)
i, j = 0, 0
while i < m and j < n:
    k = i
    while k < m and abs(A[k, j]) < 10**(-6):
        k += 1
    if k == m:
        j += 1
    else:
        if k > i:
            A[[k, i]] = A[[i, k]]
            print(f"Swapped row {i} and {k}:\n", A)
        A[i, j:] = A[i, j:] / A[i, j]
        print(f"created a leading one at pos. ({i}, {j}):\n", A)
        A[(i+1):, j:] = A[(i+1):, j:] - A[(i+1):, [j]] @ A[[i], j:]
        print(f"zeroed out below row {i} in col. {j}:\n", A)
        piv[i] = j
        i += 1; j += 1
i -= 1
while i > 0:
    j = piv[i]
    A[:, i, j:] = A[:, i, j:] - A[:, i, [j]] @ A[[i], j:]
    print(f"zeroed out above row {i} in col. {j}:\n", A)
    i -= 1
```

In the html-version of these notes you can change the matrix and run the code as you like.

Exercises

Exercise 3.1. Find the solution(s) to the following systems.

(a)

$$\begin{array}{rrcr} 2x_1 & -7x_2 & +x_3 & = & -21 \\ x_1 & -4x_2 & -3x_3 & = & -23 \\ x_1 & -x_2 & +2x_3 & = & 4 \end{array}$$

! Solution

$(2, 4, 3)$

(b)

$$\begin{array}{rrcr} x_1 & -2x_2 & & = & 3 \\ x_1 & -3x_2 & & = & 4 \\ -x_1 & +5x_2 & & = & 2 \end{array}$$

! Solution

The system has no solution.

(c)

$$\begin{array}{rrcr} x_1 & -2x_2 & +2x_3 & = & 2 \\ 2x_1 & +3x_2 & -4x_3 & = & 2 \end{array}$$

! Solution

$\mathbf{x} = (3/2, x_3 - 1/4, x_3) = (3/2, -1/4, 0) + x_3(0, 1, 1).$

Exercise 3.2. We are given the matrix $A = \begin{pmatrix} 1 & -1 & 3 \\ 2 & 1 & 4 \\ -4 & 7 & 2 \end{pmatrix}$. Find the solution to the systems $A\mathbf{x} = \mathbf{b}_1$ and $A\mathbf{x} = \mathbf{b}_2$ where $\mathbf{b}_1 = (-8, -7, -7)$ and $\mathbf{b}_2 = (9, 5, -17)$.

! Solution

$\mathbf{x}_1 = (2, 1, -3)$ and $\mathbf{x}_2 = (0, -3, 2).$

Exercise 3.3. Find another set of row operations which brings the matrix from Example 3.1 to reduced echelon form. Verify that you obtain the same reduced echelon form as in the example.

Exercise 3.4. For which value of a has the following system a solution?

$$2x + 3y + 4z = 5$$

$$2x + 5y + 4z = 7$$

$$4x + 7y + 8z = a$$

! Solution

$$\begin{pmatrix} 2 & 3 & 4 & 5 \\ 2 & 5 & 4 & 7 \\ 4 & 7 & 8 & a \end{pmatrix} \sim \begin{pmatrix} 2 & 3 & 4 & 5 \\ 0 & 2 & 0 & 2 \\ 0 & 1 & 0 & a-10 \end{pmatrix} \sim \begin{pmatrix} 2 & 3 & 4 & 5 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & a-11 \end{pmatrix}$$

We see that it is only for $a = 11$ that the last column is not a pivot column.

Exercise 3.5. Find the reduced echelon form of the following matrices

(a)

$$A = \begin{pmatrix} 1 & 2 & -2 & 2 \\ 2 & 4 & -1 & 16 \\ 3 & 6 & -5 & 11 \end{pmatrix}$$

! Solution

We get

$$\begin{pmatrix} 1 & 2 & -2 & 2 \\ 2 & 4 & -1 & 16 \\ 3 & 6 & -5 & 11 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & -2 & 2 \\ 0 & 0 & 3 & 12 \\ 0 & 0 & 1 & 5 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & -2 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(b)

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 5 & 7 & 13 \\ 3 & 7 & 10 & 18 \end{pmatrix}$$

! Solution

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Exercise 3.6. What are the pivot columns of the following matrix?

$$\begin{pmatrix} 1 & 2 & 4 \\ 2 & 4 & 9 \\ -2 & -4 & 1 \\ -3 & -6 & 0 \end{pmatrix}$$

! Solution

$$\begin{pmatrix} 1 & 2 & 4 \\ 2 & 4 & 9 \\ -2 & -4 & 1 \\ -3 & -6 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 4 \\ 0 & 0 & 1 \\ 0 & 0 & 9 \\ 0 & 0 & 12 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

We see that column 1 and 3 are pivot columns.

Exercise 3.7 (A product of elementary matrices). What is the product

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

! Solution

$$\begin{pmatrix} 1 & 0 & 0 \\ 10 & 3 & 1 \\ 10 & 5 & 0 \end{pmatrix}$$

Exercise 3.8. Let $A \in \mathbb{R}^{3 \times 3}$. Suppose that the system $A\mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix}$ has the solution $\mathbf{x} = \begin{pmatrix} 5 \\ 6 \\ -3 \end{pmatrix}$, and that the system $A\mathbf{y} = \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix}$ has the solution $\mathbf{y} = \begin{pmatrix} 3 \\ -1 \\ 2 \end{pmatrix}$. What is then the solution of the system $A\mathbf{z} = \begin{pmatrix} 2 \\ 8 \\ 4 \end{pmatrix}$?

! Solution

We have that

$$\begin{pmatrix} 2 \\ 8 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix}.$$

The solution of the last system is therefore

$$2 \begin{pmatrix} 5 \\ 6 \\ -3 \end{pmatrix} + 2 \begin{pmatrix} 3 \\ -1 \\ 2 \end{pmatrix} = \begin{pmatrix} 16 \\ 10 \\ -2 \end{pmatrix}.$$

Exercise 3.9. Above we programmed the first three row operations in Example 3.1. Write code that applies the last row operations as well.

! Solution

The last row operations can be programmed as follows

```
A[2,:] = A[2,:] + 8*A[1,:]
A[2,:]=(-5)*A[2,:]
A[0,:]=A[0,:]-A[2,:]
A[1,:] = A[1,:] +(2/5)*A[2,:]
A[0,:] = A[0,:] + 2*A[1,:]
```

Exercise 3.10 (Farkas lemma). Show that the system $A\mathbf{x} = \mathbf{b}$ (with m equations and n unknowns) has a solution if and only if there is no $\mathbf{c} \in \mathbb{R}^m$ so that $\mathbf{c}^T A = \mathbf{0}$ and $\mathbf{c}^T \mathbf{b} = 1$. This result is also called *Farkas lemma*.

! Solution

We prove $1. \Rightarrow 2.$ first. Suppose $\mathbf{x}$ is a solution to the system, and assume for contradiction there exists a $\mathbf{c}$ so that $\mathbf{c}^T A = \mathbf{0}$ and $\mathbf{c}^T \mathbf{b} = 1$. Then $\mathbf{c}^T A\mathbf{x} = \mathbf{c}^T \mathbf{b}$ says that

$$0x_1 + 0x_2 + \cdots + 0x_n = 1,$$

which is impossible. Thus, there can be no such $\mathbf{c}$, and this proves $1. \Rightarrow 2.$. We prove $2. \Rightarrow 1.$ contrapositively. Assume thus that the system does not have a solution. The augmented matrix must then have the last column as a pivot column, and the last row says that

$$0x_1 + 0x_2 + \cdots + 0x_n = 1.$$

This equation can be obtained as a linear combination of the rows of $(A \quad \mathbf{b})$, i.e., there exists a vector $\mathbf{c}$ so that $\mathbf{c}^T A = \mathbf{0}$ and $\mathbf{c}^T \mathbf{b} = 1$, so that such a $\mathbf{c}$ exists. This is the negation of 2., and this proves $2. \Rightarrow 1.$.

Chapter 4

The inverse matrix

We have seen that a system $A\mathbf{x} = \mathbf{b}$ can be solved by Gaussian elimination. If A is square and satisfies 1.-4. in Theorem 3.1, we can consider the mapping $\mathbf{b} \rightarrow \mathbf{x}$, i.e., the mapping which sends a given right hand side to the unique solution $\mathbf{x}$. This is easily seen to be a linear transformation, and its matrix will be denoted by A^{-1} . If we have found A^{-1} , we can solve any system $A\mathbf{x} = \mathbf{b}$ simply by computing $\mathbf{x} = A^{-1}\mathbf{b}$.

In this chapter we shall define the inverse matrix more precisely. We will start by proving the following:

Lemma 4.1. *Let A be a square matrix. A satisfies 1.-4. in Theorem 3.1 if and only if there exists a matrix B so that $AB = I_n$.*

Proof. Suppose that $A = E_k \cdots E_1$ satisfies 1.-4. in Theorem 3.1. If F_i are elementary matrices which represent the reverse operations we have that

$$E_k \cdots E_1 F_1 \cdots F_k = I_n.$$

The matrix $B = F_1 \cdots F_k$ thus satisfies $AB = I_n$. The other way, suppose A and B are square matrices so that $AB = I_n$. Suppose that $B\mathbf{x} = \mathbf{0}$ for an $\mathbf{x} \neq \mathbf{0}$. Then $AB\mathbf{x} = \mathbf{0}$ also, which violates that $AB = I_n$. $B\mathbf{x} = \mathbf{0}$ therefore has the unique solution $\mathbf{x} = \mathbf{0}$, so that B satisfies 1.-4. in Theorem 3.1. But then B can be written as a product of elementary matrices, $B = F_1 \cdots F_k$. From $AB = I_n$ it follows that $A = E_k \cdots E_1$, where E_i represents the reverse row operations. A thus satisfies 1.-4. in Theorem 3.1 også. $\square$

When $AB = I_n$ we say that B is a *right inverse* of A . Due to the lemma above we can add that A has a right inverse as an equivalence in Theorem 3.1. We will do this, but with a small modification: We say that B is a *left inverse* of A if $BA = I_n$. Suppose A has a right inverse B . From the reasoning above it follows

that we can write $A = E_k \cdots E_1$ and $B = F_1 \cdots F_k$, where the E_i are elementary, and where the F_i are the reverse counterparts. But then

$$BA = F_1 \cdots F_k E_k \cdots E_1 = I_n,$$

so that B also is a left inverse of A . It also follows that a left/right inverse is unique. We have shown the following.

Proposition 4.1. *Suppose that B is a right inverse of A . Then B is also a left inverse to A . A left inverse/right inverse of A is unique.*

Due to this we need not differ between right- and left inverses, and it is natural to define that A is *invertible* when there exists a matrix B so that $AB = BA = I_n$. B is called the *inverse matrix* of A , and we write A^{-1} for this.

Some properties of the inverse matrix follow immediately from the definition. Since $A^{-1}A = I_n$, in particular $(A^{-1})^{-1} = A$. Furthermore, if $s \neq 0$ then

$$(sA)(s^{-1}A^{-1}) = AA^{-1} = I_n,$$

so that $(sA)^{-1} = s^{-1}A^{-1}$. In the exercises of this chapter you will be asked to prove other properties of the inverse matrix.

Let us now extend Theorem 3.1 with an equivalence concerning invertibility, and at the same time give it a name.

Theorem 4.1 (Invertible matrix theorem). *The following are equivalent for an $n \times n$ -matrix A :*

1. A is row equivalent with I_n .
2. $A\mathbf{x} = \mathbf{0}$ has $\mathbf{x} = \mathbf{0}$ as the only solution.
3. $A\mathbf{x} = \mathbf{b}$ has a unique solution for all right hand sides $\mathbf{b}$.
4. A can be written as a product of elementary matrices.
5. A is invertible.

From the discussion above it is also clear that if $E_1, E_2, \dots, E_k$ are elementary matrices which row reduce A to the identity matrix (so that $E_k \cdots E_1 A = I_n$), then

$$A = F_1 \cdots F_k \qquad A^{-1} = E_k \cdots E_1$$

where $F_i = E_i^{-1}$. In other words:

- A^{-1} can be obtained by multiplying the elementary matrices,
- A can be obtained by multiplying their inverses in reverse order.

To compute A^{-1} it is useful to apply the row operations not only on A , but also on the augmented matrix $(A \ I_n)$. Then we get

$$E_k \cdots E_1 (A \ I_n) = (I_n \ E_k \cdots E_1) = (I_n \ A^{-1}).$$

From the result we can read out A^{-1} from the last columns. There is an alternative way to interpret this: when row reducing $(A \ I_n)$, we simultaneously solve the systems

$$A\mathbf{x}_1 = \mathbf{e}_1 \quad A\mathbf{x}_2 = \mathbf{e}_2 \quad \dots \quad A\mathbf{x}_n = \mathbf{e}_n,$$

and since $A\mathbf{x}_i = \mathbf{e}_i$ implies that $A^{-1}\mathbf{e}_i = \mathbf{x}_i$, column i in A^{-1} will be $\mathbf{x}_i$. Let us summarise.

Proposition 4.2. *Suppose that A is invertible. The reduced echelon form of $(A \ I_n)$ equals $(I_n \ A^{-1})$.*

Example 4.1. Let us find the inverse matrix of $A = \begin{pmatrix} 2 & 3 & 1 \\ 1 & 4 & 1 \\ 2 & 4 & 2 \end{pmatrix}$:

$$\begin{aligned} (A \ I_n) &= \begin{pmatrix} 2 & 3 & 1 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 & 1 & 0 \\ 2 & 4 & 2 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{I \leftrightarrow II} \begin{pmatrix} 1 & 4 & 1 & 0 & 1 & 0 \\ 2 & 3 & 1 & 1 & 0 & 0 \\ 2 & 4 & 2 & 0 & 0 & 1 \end{pmatrix} \\ &\xrightarrow{II \sim 2I} \begin{pmatrix} 1 & 4 & 1 & 0 & 1 & 0 \\ 0 & -5 & -1 & 1 & -2 & 0 \\ 2 & 4 & 2 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{III \sim 2I} \begin{pmatrix} 1 & 4 & 1 & 0 & 1 & 0 \\ 0 & -5 & -1 & 1 & -2 & 0 \\ 0 & -4 & 0 & 0 & -2 & 1 \end{pmatrix} \\ &\xrightarrow{II \leftrightarrow III} \begin{pmatrix} 1 & 4 & 1 & 0 & 1 & 0 \\ 0 & -4 & 0 & 0 & -2 & 1 \\ 0 & -5 & -1 & 1 & -2 & 0 \end{pmatrix} \xrightarrow{II/(-4)} \begin{pmatrix} 1 & 4 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1/2 & -1/4 \\ 0 & -5 & -1 & 1 & -2 & 0 \end{pmatrix} \\ &\xrightarrow{III + 5II} \begin{pmatrix} 1 & 4 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1/2 & -1/4 \\ 0 & 0 & -1 & 1 & 1/2 & -5/4 \end{pmatrix} \xrightarrow{III(-1)} \begin{pmatrix} 1 & 4 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1/2 & -1/4 \\ 0 & 0 & 1 & -1 & -1/2 & 5/4 \end{pmatrix} \\ &\xrightarrow{I - III} \begin{pmatrix} 1 & 4 & 0 & 1 & 3/2 & -5/4 \\ 0 & 1 & 0 & 0 & 1/2 & -1/4 \\ 0 & 0 & 1 & -1 & -1/2 & 5/4 \end{pmatrix} \xrightarrow{I - 4II} \begin{pmatrix} 1 & 0 & 0 & 1 & -1/2 & -1/4 \\ 0 & 1 & 0 & 0 & 1/2 & -1/4 \\ 0 & 0 & 1 & -1 & -1/2 & 5/4 \end{pmatrix}. \end{aligned}$$

We read out the last 3 columns as $A^{-1} = \begin{pmatrix} 1 & -1/2 & -1/4 \\ 0 & 1/2 & -1/4 \\ -1 & -1/2 & 5/4 \end{pmatrix}$. We can verify

the result with the built-in function for computing the inverse:

```
np.linalg.inv([[2,3,1],[1,4,1],[2,4,2]])
```

```
array([[ 1.   , -0.5  , -0.25],
       [ 0.   ,  0.5  , -0.25],
       [-1.   , -0.5  ,  1.25]])
```

We can also verify the answer symbolically:

```
sym.Matrix([[2,3,1],[1,4,1],[2,4,2]]).inv()
```

$$\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{4} \\ 0 & \frac{1}{2} & -\frac{1}{4} \\ -1 & -\frac{1}{2} & \frac{5}{4} \end{bmatrix}$$

We can also read out the elementary matrices

$$\begin{aligned}
 E_1 &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} & E_2 &= \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & E_3 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix} & E_4 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \\
 E_5 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1/4 & 0 \\ 0 & 0 & 1 \end{pmatrix} & E_6 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 0 & 1 \end{pmatrix} & E_7 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} & E_8 &= \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
 E_9 &= \begin{pmatrix} 1 & -4 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
 \end{aligned}$$

and

$$\begin{aligned}
 F_1 &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} & F_2 &= \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & F_3 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{pmatrix} & F_4 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \\
 F_5 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & 1 \end{pmatrix} & F_6 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -5 & 0 & 1 \end{pmatrix} & F_7 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} & F_8 &= \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
 F_9 &= \begin{pmatrix} 1 & 4 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
 \end{aligned}$$

With these we have that $A^{-1} = E_9 \cdots E_1$, and $A = F_1 \cdots F_9$. ♣

Example 4.2. Consider the system $A\mathbf{x} = \mathbf{b}$ with $A = \begin{pmatrix} 2 & 3 & 1 \\ 1 & 4 & 1 \\ 2 & 4 & 2 \end{pmatrix}$ from the

preceding example, and where $\mathbf{b} = \begin{pmatrix} 19 \\ 21 \\ 26 \end{pmatrix}$. We found A^{-1} above. If we use this

we see that the solution to the system above is

$$\mathbf{x} = A^{-1}\mathbf{b} = \begin{pmatrix} 1 & -1/2 & -1/4 \\ 0 & 1/2 & -1/4 \\ -1 & -1/2 & 5/4 \end{pmatrix} \begin{pmatrix} 19 \\ 21 \\ 26 \end{pmatrix} = \begin{pmatrix} 19 - 21/2 - 13/2 \\ 21/2 - 13/2 \\ -19 - 21/2 + 65/2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 3 \end{pmatrix}.$$

The system can also be solved by Gaussian elimination on the augmented matrix $(A \quad \mathbf{b})$. This is preferable, since Gaussian elimination on $(A \quad I_n)$ requires more computation than that on $(A \quad \mathbf{b})$ (there are fewer columns in this matrix). ♣

Example 4.3 (Square Vandermonde matrices are invertible). Let us prove that the Vandermonde matrix

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \tag{4.1}$$

is invertible if and only if the generators $x_0, x_1, \dots, x_n$ are distinct. One direction of the proof is easy: If two of the generators are equal, then two of the rows are equal, and then A can't be invertible (Exercise 4.4). The other way is a bit more complicated.

Suppose now that all generators are distinct. We will prove by induction on n that the Vandermonde matrix is invertible. It is clear that 1×1 Vandermonde matrices are invertible, since they are simply equal to (1) . Suppose now that all square Vandermonde matrices of dimension $n \times n, (n-1) \times (n-1), \dots, 1 \times 1$ have been shown to be invertible. To show that all $(n+1) \times (n+1)$ -matrices are invertible it is enough to show that their transposes are invertible (Exercise 4.3). So, consider the matrix

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_n \\ x_0^2 & x_1^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_0^n & x_1^n & \cdots & x_n^n \end{pmatrix}.$$

We now subtract x_0 times row $i-1$ from row i (for $i = n, n-1, \dots, 2$). This zeros everything under the leading one in the upper left corner, and does not change whether the matrix is invertible or not. It therefore holds to show that

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & x_1 - x_0 & \cdots & x_n - x_0 \\ 0 & x_1(x_1 - x_0) & \cdots & x_n(x_n - x_0) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & x_1^{n-1}(x_1 - x_0) & \cdots & x_n^{n-1}(x_n - x_0) \end{pmatrix}$$

is invertible. But this is the same as that

$$\begin{pmatrix} x_1 - x_0 & \cdots & x_n - x_0 \\ x_1(x_1 - x_0) & \cdots & x_n(x_n - x_0) \\ \vdots & \ddots & \vdots \\ x_1^{n-1}(x_1 - x_0) & \cdots & x_n^{n-1}(x_n - x_0) \end{pmatrix}$$

is invertible, which is the same as that

$$\begin{pmatrix} x_1 - x_0 & x_1(x_1 - x_0) & \cdots & x_1^{n-1}(x_1 - x_0) \\ \vdots & \vdots & \ddots & \vdots \\ x_n - x_0 & x_n(x_n - x_0) & \cdots & x_n^{n-1}(x_n - x_0) \end{pmatrix}$$

is invertible (Exercise 4.3 again). We now apply more row operations: The first row is multiplied with $1/(x_1 - x_0)$ the second row with $1/(x_2 - x_0)$, and so on. After having done this we see that it is enough to show that

$$\begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{pmatrix}$$

is invertible. But this is an $n \times n$ Vandermonde matrix, which we assumed to be invertible. This completes the proof. ♣

Example 4.4 (The Fourier matrix). The Fourier matrix is also a Vandermonde matrix, so that this is invertible as well. While we don't have a general expression for the inverse of a Vandermonde matrix, the situation is different for the Fourier matrix: We claim that the inverse matrix has components $\frac{1}{N}e^{2\pi i kn/N}$. We have that

$$\sum_{n=0}^{N-1} e^{-2\pi i kn/N} \frac{1}{N} e^{2\pi i nl/N} = \frac{1}{N} \sum_{n=0}^{N-1} e^{2\pi i n(l-k)/N}.$$

If $l = k$ this is 1. If $l \neq k$ we can use the sum formula for a geometric series to get

$$\frac{1 - e^{2\pi i(l-k)}}{1 - e^{2\pi i(l-k)/N}} = 0.$$

It follows that the inverse matrix has components $\frac{1}{N}e^{2\pi i kn/N}$, so that $(F_N)^{-1} = \frac{1}{N}\overline{F_N}$. ♣

Exercises

Exercise 4.1. In this exercise we set $A = \begin{pmatrix} 2 & -1 & 1 \\ 0 & 1 & 4 \\ 1 & -2 & -6 \end{pmatrix}$

(a) Find the inverse matrix of A

! Solution

Gaussian elimination gives:

$$\begin{pmatrix} 2 & -1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 4 & 0 & 1 & 0 \\ 1 & -2 & -6 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{I \leftrightarrow III} \begin{pmatrix} 1 & -2 & -6 & 0 & 0 & 1 \\ 0 & 1 & 4 & 0 & 1 & 0 \\ 2 & -1 & 1 & 1 & 0 & 0 \end{pmatrix} \xrightarrow{III \sim -2I} \begin{pmatrix} 1 & -2 & -6 & 0 & 0 & 1 \\ 0 & 1 & 4 & 0 & 1 & 0 \\ 0 & 3 & 13 & 1 & 0 & -2 \end{pmatrix} \\ \xrightarrow{III \sim -3II} \begin{pmatrix} 1 & -2 & -6 & 0 & 0 & 1 \\ 0 & 1 & 4 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & -3 & -2 \end{pmatrix} \xrightarrow{II \sim -4III} \begin{pmatrix} 1 & -2 & -6 & 0 & 0 & 1 \\ 0 & 1 & 0 & -4 & 13 & 8 \\ 0 & 0 & 1 & 1 & -3 & -2 \end{pmatrix} \\ \xrightarrow{I \sim +6III} \begin{pmatrix} 1 & -2 & 0 & 6 & -18 & -11 \\ 0 & 1 & 0 & -4 & 13 & 8 \\ 0 & 0 & 1 & 1 & -3 & -2 \end{pmatrix} \xrightarrow{I \sim +2II} \begin{pmatrix} 1 & 0 & 0 & -2 & 8 & 5 \\ 0 & 1 & 0 & -4 & 13 & 8 \\ 0 & 0 & 1 & 1 & -3 & -2 \end{pmatrix}$$

It follows that $A^{-1} = \begin{pmatrix} -2 & 8 & 5 \\ -4 & 13 & 8 \\ 1 & -3 & -2 \end{pmatrix}$.

(b) Write A and A^{-1} as products of elementary matrices.

! Solution

From the row operations above it follows that we can write

$$\begin{aligned}
 A^{-1} &= \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 6 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{pmatrix} \\
 &\quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \\
 A &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{pmatrix} \\
 &\quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -6 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
 \end{aligned}$$

For A we applied the reverse row operations in reverse order.

Exercise 4.2 (The inverse of 2×2 -matrices). Show that a general 2×2 -matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is invertible if and only if $ad - bc \neq 0$, and that the inverse is $\frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.

! Solution

We have that

$$\begin{aligned}
 &\begin{pmatrix} a & b \\ c & d \end{pmatrix} \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \\
 &= \frac{1}{ad-bc} \begin{pmatrix} ad-bc & -ab+ab \\ cd-cd & -bc+ad \end{pmatrix} = \frac{1}{ad-bc} \begin{pmatrix} ad-bc & 0 \\ 0 & ad-bc \end{pmatrix} = I_n.
 \end{aligned}$$

It follows that $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$ when $ad - bc \neq 0$. When $ad - bc = 0$ there are two possibilities:

1. $ad = bc = 0$: A must then have a row or column which is 0, and we know that it then can't be row reduced to I_n , so that it is not invertible.

2. $ad = bc \neq 0$: The all four numbers are $\neq 0$ and we get

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \sim \begin{pmatrix} 1 & b/a \\ 0 & d - bc/a \end{pmatrix} = \begin{pmatrix} 1 & b/a \\ 0 & (ad - bc)/a \end{pmatrix} = \begin{pmatrix} 1 & b/a \\ 0 & 0 \end{pmatrix},$$

which also can't be row reduced to I_n .

Exercise 4.3 (The transpose of an invertible matrix is invertible). Show that A is invertible if and only if A^T is invertible. Show that then $(A^T)^{-1} = (A^{-1})^T$.

! Solution

Suppose that A is invertible. Then

$$(A^{-1})^T A^T = (AA^{-1})^T = I^T = I,$$

so that A^T is invertible with inverse $(A^T)^{-1} = (A^{-1})^T$. The other way follows by replacing A with A^T .

Exercise 4.4. Let A be a matrix with either a row or a column of zeros. Show that A is not invertible. Show also that the same conclusion holds if A has two equal rows or columns.

! Solution

If row i in A is zero we have that $A\mathbf{e}_i = \mathbf{0}$, and it follows that A is not invertible. If A has two equal rows we can apply an elementary row operation to obtain a matrix with a row of zeros. This is then not invertible, and A is then not invertible either.

In the case when A has a column of zeros or two equal columns, A^T has a row of zeros or two equal rows. From what we just have shown, A^T is not invertible.

From Exercise 4.3 it follows that A is not invertible either.

Exercise 4.5 (Products of invertible matrices are invertible). Suppose that A and B both are invertible $n \times n$ -matrices. Show that AB also is invertible, and that $(AB)^{-1} = B^{-1}A^{-1}$.

! Solution

Assume that $AB\mathbf{x} = \mathbf{0}$. Since A is invertible it follows that $B\mathbf{x} = \mathbf{0}$. Since B is invertible we must also have that $\mathbf{x} = \mathbf{0}$. It follows that AB is invertible.

Since

$$B^{-1}A^{-1}AB = B^{-1}B = I_n,$$

it also follows that $(AB)^{-1} = B^{-1}A^{-1}$.

Exercise 4.6 (Invertible matrices are isomorphisms). Let $T : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a linear transformation. We say that T is $1 \rightarrow 1$ if $T(\mathbf{x}) \neq T(\mathbf{y})$ whenever $\mathbf{x} \neq \mathbf{y}$. We also say that T is onto if, for any $\mathbf{y} \in V$, there exists a $\mathbf{x} \in U$ so that $T(\mathbf{x}) = \mathbf{y}$. Show that the matrix of T is invertible if and only if T is onto and $1 \rightarrow 1$. Linear transformations that are onto and $1 \rightarrow 1$ are also called isomorphisms.

! Solution

Let A be the matrix of T , so that $T(\mathbf{x}) = A\mathbf{x}$. If A is invertible, $A\mathbf{x} = \mathbf{b}$ has a unique solution for any $\mathbf{b}$, so that T is onto. Also, if $A\mathbf{x}_1 = A\mathbf{x}_2$, we have that $A(\mathbf{x}_2 - \mathbf{x}_1) = \mathbf{0}$, so that $\mathbf{x}_2 - \mathbf{x}_1 = \mathbf{0}$ (since A is invertible), so that $\mathbf{x}_2 = \mathbf{x}_1$. It follows that T also is $1 \rightarrow 1$.

The other way, Assume that T is onto. Then $A\mathbf{x} = \mathbf{b}$ always has a solution. But, since T also is $1 \rightarrow 1$, if $A\mathbf{x}_1 = A\mathbf{x}_2$ we must have that $\mathbf{x}_1 = \mathbf{x}_2$. It follows that $A\mathbf{x} = \mathbf{b}$ always has a unique solution. It follows that A is invertible.

Exercise 4.7 (Nilpotent matrices are not invertible). A square matrix A is called *nilpotent* if there exists a $k > 1$ so that $A^k = 0$. The smallest such k is called the *index of nilpotence* of A .

(a) Run the following code:

```
import numpy as np

n = 6
A = np.zeros((n,n))
A[:(-1),1:] = np.eye(n-1)
for k in range(1,n+1):
    print( np.linalg.matrix_power(A,k))
```

Is A nilpotent? If so, what is the index of nilpotence?

! Solution

The index of nilpotence seems to be n .

(b) Show that nilpotent matrices are not invertible.

! Solution

Since A^k is not invertible, and since $A = A^{k-1}A$, it follows from Exercise 4.5 that one of A and A^{k-1} can not be invertible. If A is not invertible we are done, so assume that A^{k-1} is not invertible. Continuing in the same way we can assume that A^{k-2} , A^{k-3} , and so on, are all not invertible. At the end we have that A^2 is not invertible, from which it follows that A is not invertible.

Exercise 4.8 (Triangular matrices and invertibility). A square matrix A is called upper-triangular if everything below the diagonal of A is 0, and lower-triangular if everything above the diagonal of A is 0. A is called triangular if it is either upper- or lower-triangular.

- (a) Show that a triangular matrix is invertible if and only if all elements on the diagonal are $\neq 0$.

! Solution

Suppose A is upper triangular. If A has no zero entries on the diagonal it is easy to row-reduce the matrix to the identity matrix, and it follows that A is invertible. Otherwise, one can row-reduce so that the nonzero entries on the diagonal are 1, with zeros above them. In this echelon form the entries with zero on the diagonal correspond to free variables, so that A is not invertible.

- (b) Suppose that A is invertible and upper/lower-triangular. Show that A^{-1} also is upper/lower-triangular.

! Solution

Suppose A has no zeros on the diagonal. Column i in A^{-1} is the solution to $A\mathbf{x} = \mathbf{e}_i$. Clearly $x_n = x_{n-1} = \dots = x_{i+1} = 0$. It follows that the entries below entry i in column i of A^{-1} are all zero, so that A^{-1} is upper triangular as well.

Exercise 4.9. Let A be an invertible matrix. Write code which verifies that, when $\mathbf{b}$ is a column vector, the commands `np.linalg.solve(A,b)` and `np.linalg.inv(A)*b` return the same.

! Solution

The following code generates a random A and $\mathbf{b}$, and compares the two expressions.

```
A = np.random.rand(5, 5)
b = np.random.rand(5,1)
np.linalg.solve(A,b) - np.linalg.inv(A)*b
```


Chapter 5

Linear independence and bases

Let $\{\mathbf{a}_i\}_{i=1}^n$ be n vectors in $\mathbb{R}^m$. If $c_1, c_2, \dots, c_n$ are scalars, we call

$$c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \dots + c_n\mathbf{a}_n$$

a *linear combination* of $\{\mathbf{a}_i\}_{i=1}^n$. The *Span* of $\{\mathbf{a}_i\}_{i=1}^n$, written

$$\text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n),$$

is the collection of all linear combinations of $\{\mathbf{a}_i\}_{i=1}^n$. We say that

- the $\mathbf{a}_i$ *span* $\mathbb{R}^m$ if $\text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n) = \mathbb{R}^m$,
- the $\mathbf{a}_i$ are *linearly independent* if every $\mathbf{b} \in \text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$ can be written as a linear combination of the $\mathbf{a}_i$ in a unique way.

The following result is often useful when it comes to proving linear independence for a set of vectors.

Proposition 5.1. *The following are equivalent:*

1. $\{\mathbf{a}_i\}_{i=1}^n$ are linearly independent
2. $c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \dots + c_n\mathbf{a}_n = \mathbf{0}$ always implies that $c_1 = c_2 = \dots = c_n = 0$.

A relation as in 2. is also called a *linear dependence relation*. In other words, if $\mathbf{c} = (c_1, \dots, c_n)$, then $A\mathbf{c} = \mathbf{0}$ is a linear dependence relation between the columns of A .

Proof. We first prove $1. \Rightarrow 2.$. Assume that the $\mathbf{a}_i$ are linearly independent, and that

$$c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \dots + c_n\mathbf{a}_n = \mathbf{0} \tag{5.1}$$

One way of satisfying Equation 5.1 is by setting $c_1 = c_2 = \dots = c_n = 0$. Since $\mathbf{0}$ can be written uniquely in this way (due to linear independence of the $\mathbf{a}_i$), it follows that in fact $c_1 = c_2 = \dots = c_n = 0$, so that 2. holds.

We prove $2. \Rightarrow 1.$ contrapositively. Assume thus that the $\mathbf{a}_i$ are not linearly independent. Then there exists a vector $\mathbf{b}$ which can be written in two different ways:

$$\begin{aligned}\mathbf{b} &= c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \dots + c_n\mathbf{a}_n \\ \mathbf{b} &= d_1\mathbf{a}_1 + d_2\mathbf{a}_2 + \dots + d_n\mathbf{a}_n\end{aligned}$$

If we subtract these we obtain

$$(c_1 - d_1)\mathbf{a}_1 + (c_2 - d_2)\mathbf{a}_2 + \dots + (c_n - d_n)\mathbf{a}_n = \mathbf{0}$$

Setting $t_k = c_k - d_k$ we get that

$$t_1\mathbf{a}_1 + t_2\mathbf{a}_2 + \dots + t_n\mathbf{a}_n = \mathbf{0},$$

where not all t_k are zero. Thus, there exists some nonzero $(c_1, \dots, c_n)$ so that $c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \dots + c_n\mathbf{a}_n = \mathbf{0}$. This is the negated statement of 2., and this completes the proof. $\square$

Orthogonal sets of vectors are linearly independent. To see this, consider a linear dependence relation

$$c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \dots + c_n\mathbf{a}_n = \mathbf{0}.$$

Taking the scalar product with $\mathbf{a}_i$ on each side we get

$$c_1(\mathbf{a}_1 \cdot \mathbf{a}_i) + c_2(\mathbf{a}_2 \cdot \mathbf{a}_i) + \dots + c_n(\mathbf{a}_n \cdot \mathbf{a}_i) = 0,$$

which simplifies to $c_i(\mathbf{a}_i \cdot \mathbf{a}_i) = 0$, so that $c_i = 0$ (since $\mathbf{a}_i \cdot \mathbf{a}_i \neq 0$ when $\mathbf{a}_i \neq \mathbf{0}$). It follows from Proposition 5.1 that $\{\mathbf{a}_i\}_{i=1}^n$ are linearly independent.

Below we shall write $\mathbf{a}_i$ for the columns of the matrix A . We will have use for the following result about linear dependence relations.

Proposition 5.2. *Row equivalent matrices share the same linear dependence relations between columns. In particular, linear dependence relations between columns in a matrix are preserved by elementary row operations.*

Proof. For the first part, assume that $B = EA$ where E is elementary. If $A\mathbf{c} = \mathbf{0}$ we have that $B\mathbf{c} = EAc = \mathbf{0}$, so that any linear dependence relation between columns in A also is a linear dependence relation between columns in B . The other way, any elementary matrix is invertible, so that $A = E^{-1}B$. From this it follows in the same way that any linear dependence relation between columns in B also is a linear dependence relation between columns in A . The two thus share the same linear dependence relations.

For the last part, if A and B are row equivalent, we have that $B = E_1 \cdots E_k A$, where the E_i are elementary. By iterating on the first part it follows that A and B have the same linear dependence relations. $\square$

The concepts linear independence and span can be interpreted by means of pivot elements:

- That $\mathbf{b}$ is a linear combination of $\{\mathbf{a}_i\}_{i=1}^n$ is the same as that we can find $x_1, x_2, \dots, x_n$ so that

$$x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \cdots x_n \mathbf{a}_n = \mathbf{b},$$

which is the same as that $A\mathbf{x} = \mathbf{b}$ has a solution, which again is the same as that the last column in the augmented matrix $(A \mid \mathbf{b})$ is not a pivot column.

- That $\mathbf{b}$ can be written as a linear combination of $\{\mathbf{a}_i\}_{i=1}^n$ in a unique way requires in addition that all columns in A are pivot columns.
- That $\{\mathbf{a}_i\}_{i=1}^n$ span $\mathbb{R}^m$ is the same as that $A\mathbf{x} = \mathbf{b}$ always has a solution, which is the same as that all rows in A are pivot rows. (since this implies that the last column in $(A \mid \mathbf{b})$ never becomes a pivot column).
- In a reduced echelon form the pivot columns are linearly independent, and the other columns can be expressed in terms of these. This therefore also applies for the matrix itself - the pivot columns are linearly independent, and the other columns are linear combinations of these.

From this we will prove the following:

Proposition 5.3. *The following hold*

- If $\{\mathbf{a}_i\}_{i=1}^n$ span $\mathbb{R}^m$ then we have that $n \geq m$.
- If $\{\mathbf{a}_i\}_{i=1}^n$ are linearly independent we have that $n \leq m$.

Proof. If $\{\mathbf{a}_i\}_{i=1}^n$ span $\mathbb{R}^m$ we know that all the rows in A are pivot rows. This gives that

$$m = \# \text{rows} = \# \text{pivot rows} = \# \text{pivot columns} \leq \# \text{columns} = n.$$

If $\{\mathbf{a}_i\}_{i=1}^n$ are linearly independent we know that all columns in A are pivot columns. This gives that

$$n = \# \text{columns} = \# \text{pivot columns} = \# \text{pivot rows} \leq \# \text{rows} = m.$$

$\square$

A *basis* for $\mathbb{R}^m$ is defined as a linearly independent set of vectors which span $\mathbb{R}^m$. We will denote bases by $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and so on. From the result above it follows

that a basis for $\mathbb{R}^m$ has exactly m elements. $\mathbb{R}^m$ has in particular the simple basis consisting of

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \dots, \mathbf{e}_m = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

We will denote this by $\mathcal{E}$, and call it the *standard basis*.

We can now expand the theorem for invertible matrices with another equivalence:

Proposition 5.4 (Invertible matrix theorem). *The following are equivalent for an $n \times n$ -matrix A :*

1. A is row equivalent with I_n .
2. $A\mathbf{x} = \mathbf{0}$ has the unique solution $\mathbf{x} = \mathbf{0}$.
3. $A\mathbf{x} = \mathbf{b}$ has a unique solution for all right hand sides $\mathbf{b}$.
4. A can be written as a product of elementary matrices.
5. A is invertible.
6. The columns of A are a basis for $\mathbb{R}^n$.

Proof. That the columns of A is a basis for $\mathbb{R}^n$ is the same as that all rows and columns have pivot elements. But this is the same as that A is row equivalent to I_n . $\square$

Thus, the columns of any invertible matrix give a basis for $\mathbb{R}^m$. If we consider the DFT matrix, we conclude that the vectors

$$\mathcal{F} = \{(e^{2\pi i n \cdot 0}, e^{2\pi i n \cdot 1}, e^{2\pi i n \cdot 2}, \dots, e^{2\pi i n (N-1)})\}_{n=0}^{N-1}$$

is a basis for $\mathbb{R}^m$. This is also called the *Fourier basis*.

Example 5.1 (Find a linearly independent subset). We have the vectors

$$\mathbf{a}_1 = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} \quad \mathbf{a}_2 = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \quad \mathbf{a}_3 = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \quad \mathbf{a}_4 = \begin{pmatrix} -2 \\ 3 \\ 4 \end{pmatrix}.$$

Let us first show that these span $\mathbb{R}^3$. We row-reduce to echelon form:

$$\begin{pmatrix} 0 & 1 & 1 & -2 \\ 2 & 1 & 2 & 3 \\ 1 & 2 & 0 & 4 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 0 & 4 \\ 2 & 1 & 2 & 3 \\ 0 & 1 & 1 & -2 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 0 & 4 \\ 0 & -3 & 2 & -5 \\ 0 & 1 & 1 & -2 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 0 & 4 \\ 0 & 1 & 1 & -2 \\ 0 & 0 & 5 & -11 \end{pmatrix}$$

From the echelon form it is clear that all rows are pivot rows - we don't need to bring the matrix to reduced echelon form to see this. It follows that the four

vectors span $\mathbb{R}^3$. Further we see that the first three columns are pivot columns, so that these are linearly independent. The fourth vector can be expressed with the other three columns since it is not a pivot column. That the four vector can't be linearly independent can also be inferred from that linear independence would require $m \geq n$, which is clearly not the case. The three columns $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ are therefore both linearly independent and span $\mathbb{R}^3$, since all rows and columns have pivot elements in this 3×3 -matrix. ♣

If the columns in a matrix do not span $\mathbb{R}^m$ we now get a simple procedure for expanding a set of pivot columns to a basis for the whole of $\mathbb{R}^m$: We modify the row-reduced echelon form of A by adding columns which are standard basis vectors, so that we obtain a new matrix with a leading one in each row. The columns in this new matrix provide a basis for $\mathbb{R}^m$. If we row reduce backwards we get m vectors which also are a basis for $\mathbb{R}^m$, and which contain the original vectors. This is easiest to illustrate with an example.

Example 5.2 (Expand to a basis). Let us see how we can expand

$$\mathbf{a}_1 = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} \quad \mathbf{a}_2 = \begin{pmatrix} 6 \\ -1 \\ 3 \end{pmatrix}$$

to a basis for $\mathbb{R}^3$. We row-reduce

$$\begin{pmatrix} 3 & 6 \\ 2 & -1 \\ 1 & 3 \end{pmatrix} \xrightarrow{I \leftrightarrow III} \begin{pmatrix} 1 & 3 \\ 2 & -1 \\ 3 & 6 \end{pmatrix} \xrightarrow{II-2I} \begin{pmatrix} 1 & 3 \\ 0 & -7 \\ 3 & 6 \end{pmatrix} \xrightarrow{III-3I} \begin{pmatrix} 1 & 3 \\ 0 & -7 \\ 0 & -3 \end{pmatrix} \\ \xrightarrow{II/(-7)} \begin{pmatrix} 1 & 3 \\ 0 & 1 \\ 0 & -3 \end{pmatrix} \xrightarrow{III+3II} \begin{pmatrix} 1 & 3 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

Since both columns are pivot columns it follows that the two vectors are linearly independent. To expand to a basis for $\mathbb{R}^3$ we add a third vector to the echelon form so that we get a third pivot column, and row-reduce backwards:

$$\begin{pmatrix} 1 & 3 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \xrightarrow{III-3II} \begin{pmatrix} 1 & 3 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix} \xrightarrow{II/(-7)} \begin{pmatrix} 1 & 3 & 0 \\ 0 & -7 & 0 \\ 0 & -3 & 1 \end{pmatrix} \xrightarrow{III+3I} \begin{pmatrix} 1 & 3 & 0 \\ 0 & -7 & 0 \\ 3 & 6 & 1 \end{pmatrix} \\ \xrightarrow{II+2I} \begin{pmatrix} 1 & 3 & 0 \\ 2 & -1 & 0 \\ 3 & 6 & 1 \end{pmatrix} \xrightarrow{I \leftrightarrow III} \begin{pmatrix} 3 & 6 & 1 \\ 2 & -1 & 0 \\ 1 & 3 & 0 \end{pmatrix}$$

It follows that we obtain a basis for $\mathbb{R}^3$ by adding $\mathbf{a}_3 = (1, 0, 0)$. ♣

5.1 Subspaces

By a *subspace* (of $\mathbb{R}^m$) we mean a set $U \subseteq \mathbb{R}^m$ so that

1. if $\mathbf{u}_1, \mathbf{u}_2 \in U$, then also $\mathbf{u}_1 + \mathbf{u}_2 \in U$
2. if $\mathbf{u} \in U$, then also $c\mathbf{u} \in U$ for all scalars c .

In particular, $\mathbb{R}^m$ is a subspace. It is an exercise to show that, more generally, $\text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$ is a subspace. In particular,

- the *column space* of a matrix (the span of all the columns in the matrix) is a subspace (of $\mathbb{R}^m$), and
- the *row space* of a matrix (the span of all the rows in the matrix) is a subspace (of $\mathbb{R}^n$).

We will write $\text{Col}(A)$ and $\text{Row}(A)$ for the column- and row space of A , respectively. It is an exercise to show that, If B is obtained from A by means of elementary row operations, then A and B have the same row space.

The *Null space* of a matrix A , written $\text{Nul}(A)$, is the collection of all vectors $\mathbf{x}$ so that $A\mathbf{x} = \mathbf{0}$. Let us show that a null space actually is a subspace:

1. if $\mathbf{u}_1, \mathbf{u}_2 \in \text{Nul}(A)$, then $A\mathbf{u}_1 = A\mathbf{u}_2 = \mathbf{0}$. But then $A(\mathbf{u}_1 + \mathbf{u}_2) = A\mathbf{u}_1 + A\mathbf{u}_2 = \mathbf{0} + \mathbf{0} = \mathbf{0}$, so that also $\mathbf{u}_1 + \mathbf{u}_2 \in \text{Nul}(A)$.
2. if $\mathbf{u} \in \text{Nul}(A)$, then $A\mathbf{u} = \mathbf{0}$. But then $A(c\mathbf{u}) = cA\mathbf{u} = c\mathbf{0} = \mathbf{0}$.

It is an exercise to show that all subspaces can be written as the null space of some matrix. ♣

More generally we define a *basis* of a subspace as a linearly independent set of vectors which spans the whole subspace. Any subspace of $\mathbb{R}^m$ has a basis. To see this one can step-by-step expand a set of vectors by adding another vector, as long as one can find a new vector which is linearly independent from the previous ones. This process will stop at latest at m vectors. When it stops we know that the subspace is the span of the (linearly independent) vectors we incrementally added.

From the reduced echelon form of A it follows that

- the pivot rows of A (the rows with a leading one) are a basis for $\text{Row}(A)$,
- the pivot columns of A are a basis for $\text{Col}(A)$.

Since bases of $\mathbb{R}^m$ have exactly m vectors, a subspace of $\mathbb{R}^m$ can't have bases with more than m vectors.

Example 5.3. Let us find bases for the row space, column space, and null space of

$$A = \begin{pmatrix} 2 & 1 & 3 & 8 \\ 1 & 3 & -1 & 4 \\ 4 & -3 & 11 & 16 \end{pmatrix}. \text{ Row reduction first gives}$$

$$A \sim \begin{pmatrix} 1 & 0 & 2 & 4 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The two first rows in the reduced echelon form are pivot rows, so that $(1 \ 0 \ 2 \ 4)$ and $(0 \ 1 \ -1 \ 0)$ provide a basis for $\text{Row}(A)$.

Since the two first columns are pivot columns, the column vectors $(2, 1, 4)$ and $(1, 3, -3)$ in A provide a basis for $\text{Col}(A)$.

The reduced echelon form says that a vector in the null space must satisfy $x_1 + 2x_3 + 4x_4 = 0$ and $x_2 - x_3 = 0$, so that $x_1 = -2x_3 - 4x_4$ and $x_2 = x_3$. It follows that a vector in the null space must be on the form

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -2x_3 - 4x_4 \\ x_3 \\ x_3 \\ x_4 \end{pmatrix} = x_3 \begin{pmatrix} -2 \\ 1 \\ 1 \\ 0 \end{pmatrix} + x_4 \begin{pmatrix} -4 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

It follows that $(-2, 1, 1, 0)$ and $(-4, 0, 0, 1)$ provide a basis for $\text{Nul}(A)$. ♣

In this example there were two free variables, and a null space of dimension two. More generally it is the case that the number of free variables always equals the dimension of the null space. This is best demonstrated with another example. Suppose A has the reduced echelon form

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

There are three free variables, x_2 , x_3 , and x_4 . The first row then says that $x_1 + 2x_2 + 3x_3 + 4x_4 = 0$, which says that $x_1 = -2x_2 - 3x_3 - 4x_4$. A general vector in the null space is therefore

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -2x_2 - 3x_3 - 4x_4 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = x_2 \begin{pmatrix} -2 \\ 1 \\ 0 \\ 0 \end{pmatrix} + x_3 \begin{pmatrix} -3 \\ 0 \\ 1 \\ 0 \end{pmatrix} + x_4 \begin{pmatrix} -4 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad (5.2)$$

so that

$$\left\{ \begin{pmatrix} -2 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} -3 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -4 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right\}$$

span the null space. These are clearly linearly independent, so that they are a basis for the null space: If the linear combination to the right in Equation 5.2 is 0, the preceding expression there says that $x_2 = x_3 = x_4 = 0$. All in all the dimension of the null space is three, which also was the number of free variables.

From the above it follows more generally that all null spaces can be written as the span of a set of vectors. It also was an exercises that all subspaces could be written as a null space. And since any span also is a subspace, we can think of span, null space, and subspace, as the same thing. The following also holds.

Proposition 5.5. *All subspaces have an orthogonal basis.*

This follows in the same way we proved that all subspaces have a basis - we step-by-step expand a set of orthogonal vectors with new ones - as long as we can find new vectors orthogonal to the ones already added.

A subspace has many different bases, but they all have the same number of vectors:

Proposition 5.6. *All bases for a subspace have the same number of vectors.*

Proof. Suppose that

$$\text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n) = \text{Sp}(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_k)$$

and assume for contradiction that $n > k$. We have a mapping $\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^k$ given by

$$\mathbf{F}(c_1, c_2, \dots, c_n) = (d_1, d_2, \dots, d_k) \quad (5.3)$$

when

$$c_1 \mathbf{a}_1 + c_2 \mathbf{a}_2 + \dots + c_n \mathbf{a}_n = d_1 \mathbf{b}_1 + d_2 \mathbf{b}_2 + \dots + d_k \mathbf{b}_k.$$

$\mathbf{F}$ is defined on all of $\mathbb{R}^n$, and is well-defined. Furthermore, it is $1 \rightarrow 1$ and linear, and can therefore be expressed by a $k \times n$ -matrix which we will denote by P . Since $n > k$ there must at least be one column in P which is not a pivot column. But then we must have a linear dependence relation

$$x_1 \mathbf{p}_1 + x_2 \mathbf{p}_2 + \dots + x_n \mathbf{p}_n = \mathbf{0}.$$

between the columns of P , so that $P\mathbf{x} = \mathbf{0}$. This contradicts that $\mathbf{F}(\mathbf{x}) = P\mathbf{x}$ is $1 \rightarrow 1$, since also $\mathbf{F}(\mathbf{0}) = P\mathbf{0} = \mathbf{0}$. It follows that $n = k$, so that the number of vectors in the two bases is equal. $\square$

The number of vectors in a basis is called the *dimension* of the subspace. When a matrix is on reduced echelon form we see that the dimension of both the row space and the column space equals the number of pivot elements. This common dimension of $\text{Col}(A)$ and $\text{Row}(A)$ is called the *rank* of the matrix, written $\text{rank}(A)$.

Example 5.4 (Rank one matrices). Let $\mathbf{a}$ and $\mathbf{b}$ be column vectors $\neq \mathbf{0}$, with m and n components, respectively. We have that

$$\mathbf{a}\mathbf{b}^T = \begin{pmatrix} a_1 \mathbf{b}^T \\ a_2 \mathbf{b}^T \\ \vdots \\ a_m \mathbf{b}^T \end{pmatrix} = (b_1 \mathbf{a} \quad b_2 \mathbf{a} \quad \dots \quad b_n \mathbf{a}).$$

It follows that $\{\mathbf{a}\}$ is a basis for the column space of $\mathbf{a}\mathbf{b}^T$, and that $\{\mathbf{b}^T\}$ is a basis for the row space of $\mathbf{a}\mathbf{b}^T$. $\mathbf{a}\mathbf{b}^T$ thus has rank 1. It is also clear that all matrices with rank one can be written on the form $\mathbf{a}\mathbf{b}^T$, where $\mathbf{a}$ and $\mathbf{b}$ are column vectors $\neq \mathbf{0}$.

By adding multiples of the first row to the other rows it is clear that $\mathbf{ab}^T$ can be row-reduced to $\begin{pmatrix} \mathbf{b}^T \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}$. It follows that we have $n - 1$ free variables when we solve $\mathbf{ab}^T \mathbf{x} = \mathbf{0}$, so that the null space has dimension $n - 1$. ♣

Rank one matrices can be useful building blocks. For example, the outer form for matrices (Proposition 2.2) said that a product of two matrices can be written as a sum of rank one matrices. We shall return to this when we discuss diagonalisation.

Proposition 5.7. *Let A be an $m \times n$ -matrix. Then we have that*

$$\text{rank}(A) + \dim(\text{Nul}(A)) = n.$$

In Example 5.3 we found that both the column space and the row space had dimension 2, so that the rank is 2. The dimension of the null space was 1. This agrees with the proposition since $2 + 1 = 3 = n$. In Example 5.4 we found that the rank was 1, and the dimension of the null space was $n - 1$. This also agrees with the proposition, since $1 + (n - 1) = n$.

Proof. This follows from that the dimension of the null space equals the number of free variables, that the rank equals the number of pivot columns, and that the sum of the number of pivot columns and the number of free variables equals the number of columns in the matrix. $\square$

We round off this section with the following result.

Proposition 5.8. *The reduced echelon form is unique.*

Proof. Suppose that a matrix A can be row-reduced to two reduced echelon forms C_1 and C_2 . We need to show that $C_1 = C_2$. We first show that C_1 and C_2 have the same pivot columns (at the same column indices). Assume by contradiction that this is not the case. Then there exists a column index k so that

- the columns $1, \dots, k - 1$ in both C_1 and C_2 have the same columns with leading ones,
- column k in C_1 have a leading one, while column k in C_2 does not.

Then column k in C_1 can not be expressed by columns $1, \dots, k - 1$ in C_1 , while column k in C_2 , can be expressed by columns $1, \dots, k - 1$ in C_2 . We thus have a linear dependence relation between the stated columns in C_2 , which is not such a relation for C_1 . This is a contradiction, since A , C_1 , and C_2 have the same linear dependence relations. Thus, C_1 and C_2 have the same columns with leading ones (at the same indices). This also means that these columns must be equal, since the leading ones must come in row 1,2,3, and so on.

The remaining columns (those without a leading one) can be written as linear combinations of the columns with leading ones. Since these are equal, and the linear dependence relations are the same, it follows that the remaining columns in C_1 and C_2 also are equal. $\square$

Note that the concept “pivot column” was not used for the reduced echelon forms above. In fact, it is first after this proof that the concept of pivot column for general matrices gives meaning - since we now have proved that they are the same, regardless of how we row-reduce the matrix.

Previously we considered linear transformations from $\mathbb{R}^n$ to $\mathbb{R}^m$. In the same way we say that a mapping T between subspaces U and V is linear if $T(\mathbf{u}_1 + \mathbf{u}_2) = T(\mathbf{u}_1) + T(\mathbf{u}_2)$, and $T(\alpha\mathbf{u}) = \alpha T(\mathbf{u})$ for all scalars α , and vectors $\mathbf{u}_i, \mathbf{u} \in U$. We say that

- T is $1 \rightarrow 1$ if $T(\mathbf{x}) \neq T(\mathbf{y})$ whenever $\mathbf{x} \neq \mathbf{y}$,
- T is onto if, for any $\mathbf{y} \in V$, there exists a $\mathbf{x} \in U$ so that $T(\mathbf{x}) = \mathbf{y}$,
- T is an *isomorphism* if it is linear, $1 \rightarrow 1$, and onto.

It is an exercise to prove the following.

Proposition 5.9 (The inverse of an isomorphism is an isomorphism). *Assume that $T : U \rightarrow V$ is an isomorphism. The mapping $S : V \rightarrow U$ defined by $S(\mathbf{y}) = \mathbf{x}$ when $T(\mathbf{x}) = \mathbf{y}$ is well-defined and satisfies $S(T(\mathbf{u})) = \mathbf{u}$ and $T(S(\mathbf{v})) = \mathbf{v}$ for all $\mathbf{u} \in U, \mathbf{v} \in V$. Moreover, S is an isomorphism, called the inverse of T .*

It is also an exercise to prove that isomorphisms preserve linear dependence relations, spanning sets, and bases. In particular, any invertible matrix preserve linear dependence relations.

5.2 Coordinate mappings and change of coordinates

Let U be a subspace of $\mathbb{R}^m$, and let $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$ be a basis for U . Any $\mathbf{u} \in U$ can be written uniquely on the form $\mathbf{u} = c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \dots + c_n\mathbf{a}_n$. The mapping from U to $\mathbb{R}^n$ defined by

$$\mathbf{u} \rightarrow (c_1, c_2, \dots, c_n)$$

is called the *coordinate mapping* of $\mathcal{A}$. We also write $[\mathbf{u}]_{\mathcal{A}}$ for $(c_1, c_2, \dots, c_n)$ and call this the *coordinate vector* of $\mathbf{u}$ relative to $\mathcal{A}$. The coordinate mapping is

- well-defined since $\mathbf{u}$ can be written uniquely on the given form,
- $1 \rightarrow 1$, since if $[\mathbf{x}]_{\mathcal{A}} = [\mathbf{y}]_{\mathcal{A}} = (c_1, c_2, \dots, c_n)$, we must have that $\mathbf{x} = \mathbf{y} = c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \dots + c_n\mathbf{a}_n$.
- onto, since if $(c_1, c_2, \dots, c_n) \in \mathbb{R}^n$, then $\mathbf{x} = c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \dots + c_n\mathbf{a}_n \in U$, has coordinate vector $[\mathbf{x}]_{\mathcal{A}} = (c_1, c_2, \dots, c_n)$.

We also have the following result:

Proposition 5.10. *The coordinate mapping of $\mathcal{A}$ is linear.*

The coordinate mapping is thus an isomorphism. Thus, we know that it preserves linear dependence relations, spanning sets, and bases.

Proof. The coordinate mapping is $1 \rightarrow 1$ and onto since the vectors in $\mathcal{A}$ are linearly independent and span U , ensuring a unique $(c_1, \dots, c_n)$ so that $\mathbf{x} = c_1 \mathbf{a}_1 + \dots + c_n \mathbf{a}_n$ for any $\mathbf{x} \in U$. To prove linearity we need to show that

$$[\mathbf{u}_1 + \mathbf{u}_2]_{\mathcal{A}} = [\mathbf{u}_1]_{\mathcal{A}} + [\mathbf{u}_2]_{\mathcal{A}},$$

and that $[c\mathbf{u}]_{\mathcal{A}} = c[\mathbf{u}]_{\mathcal{A}}$ for any scalar c . Let us show only the first statement. Write

$$[\mathbf{u}_1]_{\mathcal{A}} = (c_1, c_2, \dots, c_n) \quad [\mathbf{u}_2]_{\mathcal{A}} = (d_1, d_2, \dots, d_n)$$

then

$$\mathbf{u}_1 = c_1 \mathbf{b}_1 + c_2 \mathbf{b}_2 + \dots + c_n \mathbf{b}_n \quad \mathbf{u}_2 = d_1 \mathbf{b}_1 + d_2 \mathbf{b}_2 + \dots + d_n \mathbf{b}_n,$$

so that

$$\mathbf{u}_1 + \mathbf{u}_2 = (c_1 + d_1) \mathbf{b}_1 + (c_2 + d_2) \mathbf{b}_2 + \dots + (c_n + d_n) \mathbf{b}_n,$$

so that

$$\begin{aligned} [\mathbf{u}_1 + \mathbf{u}_2]_{\mathcal{A}} &= (c_1 + d_1, c_2 + d_2, \dots, c_n + d_n) \\ &= (c_1, c_2, \dots, c_n) + (d_1, d_2, \dots, d_n) = [\mathbf{u}_1]_{\mathcal{A}} + [\mathbf{u}_2]_{\mathcal{A}}. \end{aligned}$$

□

Example 5.5. Let us find the coordinate vector of $\mathbf{x} = (2, 4)$ relative to the basis $\mathcal{B} = \{(-1, 3), (3, 1)\}$ for $\mathbb{R}^2$. We need to solve

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

We row-reduce

$$\begin{pmatrix} -1 & 3 & 2 \\ 3 & 1 & 4 \end{pmatrix} \sim \begin{pmatrix} 1 & -3 & -2 \\ 0 & 10 & 10 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

so that

$$\begin{pmatrix} 2 \\ 4 \end{pmatrix} = \begin{pmatrix} -1 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 1 \end{pmatrix}.$$

It follows that the coordinate vector of $\mathbf{x}$ relative to $\mathcal{B}$ is $[\mathbf{x}]_{\mathcal{B}} = (1, 1)$. ♣

From Proposition 5.10 it follows that, if $\mathcal{A} = \{\mathbf{a}_i\}_{i=1}^n$ and $\mathcal{B} = \{\mathbf{b}_i\}_{i=1}^n$ are bases for the same subspace, the mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$ defined by $[\mathbf{x}]_{\mathcal{A}} \rightarrow [\mathbf{x}]_{\mathcal{B}}$ (where $\mathbf{x} \in U$) is linear, well-defined, $1 \rightarrow 1$, and onto (as it is a composition of two such mappings), so that it is given by an invertible matrix (Exercise 4.6).

This matrix is called the *change of coordinate matrix* from $\mathcal{A}$ to $\mathcal{B}$, and is denoted $P_{\mathcal{B} \leftarrow \mathcal{A}}$. Thus,

$$[\mathbf{x}]_{\mathcal{B}} = P_{\mathcal{B} \leftarrow \mathcal{A}}[\mathbf{x}]_{\mathcal{A}}.$$

Note that the mapping $\mathbf{F}$ from the proof of Proposition 5.6 can be written as $[\mathbf{x}]_{\mathcal{B}} = \mathbf{F}([\mathbf{x}]_{\mathcal{A}})$, so that the matrix of $\mathbf{F}$ is $P_{\mathcal{B} \leftarrow \mathcal{A}}$. Note that

$$[\mathbf{a}_i]_{\mathcal{B}} = P_{\mathcal{B} \leftarrow \mathcal{A}}[\mathbf{a}_i]_{\mathcal{A}} = P_{\mathcal{B} \leftarrow \mathcal{A}}\mathbf{e}_i,$$

and the last expression is simply column i in $P_{\mathcal{B} \leftarrow \mathcal{A}}$. Thus

$$P_{\mathcal{B} \leftarrow \mathcal{A}} = ([\mathbf{a}_1]_{\mathcal{B}} \quad [\mathbf{a}_2]_{\mathcal{B}} \quad \cdots \quad [\mathbf{a}_n]_{\mathcal{B}}).$$

Clearly $P_{\mathcal{B} \leftarrow \mathcal{A}}$ is invertible with inverse $P_{\mathcal{B} \leftarrow \mathcal{A}}^{-1} = P_{\mathcal{A} \leftarrow \mathcal{B}}$ (the inverse is the change of coordinates the opposite way).

Example 5.6. Let us find the change of coordinate matrix from $\mathcal{A} = \{(2, 1), (1, -2)\}$ to $\mathcal{B} = \{(-1, 3), (3, 1)\}$. We can express $\mathbf{a}_1$ and $\mathbf{a}_2$ in terms of $\mathbf{b}_1$ and $\mathbf{b}_2$ by row-reducing

$$\begin{aligned} \begin{pmatrix} -1 & 3 & 2 & 1 \\ 3 & 1 & 1 & -2 \end{pmatrix} &\sim \begin{pmatrix} 1 & -3 & -2 & -1 \\ 0 & 10 & 7 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & -3 & -2 & -1 \\ 0 & 1 & 7/10 & 1/10 \end{pmatrix} \\ &\sim \begin{pmatrix} 1 & 0 & 1/10 & -7/10 \\ 0 & 1 & 7/10 & 1/10 \end{pmatrix} \end{aligned}$$

It follows that

$$[\mathbf{a}_1]_{\mathcal{B}} = (1/10, 7/10) \quad [\mathbf{a}_2]_{\mathcal{B}} = (-7/10, 1/10),$$

so that

$$P_{\mathcal{B} \leftarrow \mathcal{A}} = \frac{1}{10} \begin{pmatrix} 1 & -7 \\ 7 & 1 \end{pmatrix}.$$

To find $P_{\mathcal{A} \leftarrow \mathcal{B}}$ we can simply invert $P_{\mathcal{B} \leftarrow \mathcal{A}}$. We can also, similarly to as above, express $\mathbf{b}_1$ and $\mathbf{b}_2$ in terms of $\mathbf{a}_1$ and $\mathbf{a}_2$ by row-reducing

$$\begin{aligned} \begin{pmatrix} 2 & 1 & -1 & 3 \\ 1 & -2 & 3 & 1 \end{pmatrix} &\sim \begin{pmatrix} 1 & -2 & 3 & 1 \\ 0 & 5 & -7 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & -2 & 3 & 1 \\ 0 & 1 & -7/5 & 1/5 \end{pmatrix} \\ &\sim \begin{pmatrix} 1 & 0 & 1/5 & 7/5 \\ 0 & 1 & -7/5 & 1/5 \end{pmatrix} \end{aligned}$$

It follows that

$$[\mathbf{b}_1]_{\mathcal{A}} = (1/5, -7/5) \quad [\mathbf{b}_2]_{\mathcal{A}} = (7/5, 1/5),$$

so that

$$P_{\mathcal{A} \leftarrow \mathcal{B}} = \frac{1}{5} \begin{pmatrix} 1 & 7 \\ -7 & 1 \end{pmatrix}.$$

♣

A particular case is when the subspace is all of $\mathbb{R}^m$. Then the standard basis can be one of the bases. The change of coordinate matrix from $\mathcal{A}$ to the standard basis is simply the matrix with $\mathbf{a}_i$ as columns, since $[\mathbf{a}_i]_{\mathcal{E}} = \mathbf{a}_i$.

Example 5.7 (Change of coordinates from the Fourier basis). The change of coordinate matrix from the Fourier basis to the standard basis has in particular the Fourier basis vectors $(e^{2\pi i n \cdot 0}, e^{2\pi i n \cdot 1}, e^{2\pi i n \cdot 2}, \dots, e^{2\pi i n (N-1)})$ as columns. We saw in Example 4.4 that the inverse of this has columns $(e^{-2\pi i k \cdot 0}, e^{-2\pi i k \cdot 1}, e^{-2\pi i k \cdot 2}, \dots, e^{-2\pi i k (N-1)})/N$, so that the change of coordinates from the standard basis to the Fourier basis is given by the $1/N$ times the DFT matrix. ♣

Let $U \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}^m$ be subspaces, and T a linear transformation from U to V . Let also $\mathcal{A}$ and $\mathcal{B}$ be bases for U and V , respectively. The mapping

$$[\mathbf{x}]_{\mathcal{A}} \rightarrow [T(\mathbf{x})]_{\mathcal{B}}.$$

is well-defined and linear (as a composition of three such mappings), but may not be $1 \rightarrow 1$ and onto (since T may not be $1 \rightarrow 1$ or onto). The mapping can thus be represented by an $m \times n$ -matrix, which is denoted by $[T]_{\mathcal{B} \leftarrow \mathcal{A}}$. It is also called *the matrix of T relative to the bases $\mathcal{A}$ and $\mathcal{B}$* . This satisfies

$$[T(\mathbf{x})]_{\mathcal{B}} = [T]_{\mathcal{B} \leftarrow \mathcal{A}} [\mathbf{x}]_{\mathcal{A}}.$$

The columns in $[T]_{\mathcal{B} \leftarrow \mathcal{A}}$ are $[T(\mathbf{a}_i)]_{\mathcal{B}}$ since

$$[T]_{\mathcal{B} \leftarrow \mathcal{A}} \mathbf{e}_i = [T]_{\mathcal{B} \leftarrow \mathcal{A}} [\mathbf{a}_i]_{\mathcal{A}} = [T(\mathbf{a}_i)]_{\mathcal{B}}.$$

When T is a mapping of U to itself, and $\mathcal{A} = \mathcal{B}$, we denote the matrix $[T]_{\mathcal{B} \leftarrow \mathcal{A}}$ simply by $[T]_{\mathcal{B}}$, and call it the matrix of T relative to $\mathcal{B}$.

Example 5.8. Consider the linear transformation $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ with matrix $\begin{pmatrix} 5 & -1 \\ -1 & 5 \end{pmatrix}$. Let us find the matrix of T relative to $\mathcal{B} = \{(1, 1), (-1, 1)\}$, i.e., $[T]_{\mathcal{B}}$. We compute

$$\begin{aligned} T(\mathbf{b}_1) &= \begin{pmatrix} 5 & -1 \\ -1 & 5 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix} = 4\mathbf{b}_1 \\ T(\mathbf{b}_2) &= \begin{pmatrix} 5 & -1 \\ -1 & 5 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} -6 \\ 6 \end{pmatrix} = 6\mathbf{b}_2 \end{aligned}$$

It follows that $[T(\mathbf{b}_1)]_{\mathcal{B}} = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$ and $[T(\mathbf{b}_2)]_{\mathcal{B}} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$, so that

$$[T]_{\mathcal{B}} = ([T(\mathbf{b}_1)]_{\mathcal{B}} \quad [T(\mathbf{b}_2)]_{\mathcal{B}}) = \begin{pmatrix} 4 & 0 \\ 0 & 6 \end{pmatrix}.$$

The matrix of T relative to this basis was thus diagonal. Since multiplying with a diagonal matrix is simpler computationally, linear transformations may be more easily computed in suitable bases. ♣

Proposition 5.11. *The following hold:*

$$[T]_{\mathcal{B}} = P_{\mathcal{B} \leftarrow \mathcal{A}} [T]_{\mathcal{A}} (P_{\mathcal{B} \leftarrow \mathcal{A}})^{-1} \quad (5.4)$$

Proof. We have that

$$[T(\mathbf{x})]_{\mathcal{A}} = [T]_{\mathcal{A}} [\mathbf{x}]_{\mathcal{A}}.$$

If we apply change of coordinates on both sides we get

$$[T(\mathbf{x})]_{\mathcal{B}} = P_{\mathcal{B} \leftarrow \mathcal{A}} [T]_{\mathcal{A}} [\mathbf{x}]_{\mathcal{A}},$$

which also can be written as

$$[T(\mathbf{x})]_{\mathcal{B}} = P_{\mathcal{B} \leftarrow \mathcal{A}} [T]_{\mathcal{A}} P_{\mathcal{A} \leftarrow \mathcal{B}} [\mathbf{x}]_{\mathcal{B}},$$

which reads as Equation 5.4 □

Exercises

Exercise 5.1 (Row equivalent matrices have the same row space). Suppose that A and B are row equivalent. Show that $\text{Row}(A) = \text{Row}(B)$.

! Solution

Since the order of the vectors $\mathbf{a}_i$ in $\text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$ does not matter, it is clear that $\text{Row}(A) = \text{Row}(B)$ if B is obtained from A by a row exchange. Since

$$\text{Sp}(s\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n) = \text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$$

it is also clear that $\text{Row}(A) = \text{Row}(B)$ if B is obtained from A by multiplying a row with a scalar $\neq 0$.

Finally it is clear that

$$\text{Sp}(\mathbf{a}_1 + c\mathbf{a}_2, \mathbf{a}_2, \dots, \mathbf{a}_n) \subseteq \text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n).$$

Since also $\mathbf{a}_1 = (\mathbf{a}_1 + c\mathbf{a}_2) - c\mathbf{a}_2$ it is also clear that

$$\text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n) \subseteq \text{Sp}(\mathbf{a}_1 + c\mathbf{a}_2, \mathbf{a}_2, \dots, \mathbf{a}_n)$$

It follows that

$$\text{Sp}(\mathbf{a}_1 + c\mathbf{a}_2, \mathbf{a}_2, \dots, \mathbf{a}_n) = \text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n).$$

Therefore it is clear that $\text{Row}(A) = \text{Row}(B)$ also if B is obtained from A by adding a multiple of one row to another. Therefore, no elementary row operation changes the row space of the matrix, so that row equivalent matrices also have the same row space.

Exercise 5.2. We have the matrix $A = \begin{pmatrix} 2 & 4 & 1 & -1 \\ 1 & -1 & 2 & 3 \\ -1 & -11 & 4 & 11 \end{pmatrix}$.

- (a) Find bases for the row- and column space of A . What is the rank of A ?

! Solution

Row reduction of A gives $A \sim \begin{pmatrix} 1 & 0 & 3/2 & 11/6 \\ 0 & 1 & -1/2 & -7/6 \\ 0 & 0 & 0 & 0 \end{pmatrix}$. We see that the first two rows are pivot rows, so that these are a basis for the row space of A , which therefore has dimension 2. We also see that the first two columns in A are pivot columns, so that these are a basis for the column space of A , which also has dimension 2. The rank of A is therefore 2.

- (b) Find a basis for the null space of A . What is the dimension of the null space?

! Solution

A vector in the null space must satisfy

$$x_1 + \frac{3}{2}x_3 + \frac{11}{6}x_4 = 0 \qquad x_2 - \frac{1}{2}x_3 - \frac{7}{6}x_4 = 0,$$

so that

$$x_1 = -\frac{3}{2}x_3 - \frac{11}{6}x_4 \qquad x_2 = \frac{1}{2}x_3 + \frac{7}{6}x_4,$$

so that

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -\frac{3}{2}x_3 - \frac{11}{6}x_4 \\ \frac{1}{2}x_3 + \frac{7}{6}x_4 \\ x_3 \\ x_4 \end{pmatrix} = x_3 \begin{pmatrix} -\frac{3}{2} \\ \frac{1}{2} \\ 1 \\ 0 \end{pmatrix} + x_4 \begin{pmatrix} -\frac{11}{6} \\ \frac{7}{6} \\ 0 \\ 1 \end{pmatrix},$$

so that $(-3, 1, 2, 0)$ and $(-11, 7, 0, 6)$ is a basis for the null space of A , which therefore has dimension 2. The last could have been concluded from that the rank plus the dimension of the null space equals the number of columns in the matrix, which is four.

Exercise 5.3. Consider the matrix $A = \begin{pmatrix} 2 & 8 & 6 & 1 \\ -4 & -16 & -12 & 2 \\ -1 & -4 & -3 & 1 \end{pmatrix}$

- (a) Find a basis for the column space of A .

! Solution

We row reduce

$$\begin{aligned}
 \begin{pmatrix} 2 & 8 & 6 & 1 \\ -4 & -16 & -12 & 2 \\ -1 & -4 & -3 & 1 \end{pmatrix} &\xrightarrow{I \leftrightarrow III} \begin{pmatrix} -1 & -4 & -3 & 1 \\ -4 & -16 & -12 & 2 \\ 2 & 8 & 6 & 1 \end{pmatrix} \\
 &\xrightarrow{I \cdot (-1)} \begin{pmatrix} 1 & 4 & 3 & -1 \\ -4 & -16 & -12 & 2 \\ 2 & 8 & 6 & 1 \end{pmatrix} \\
 &\xrightarrow{II+4I, III-2I} \begin{pmatrix} 1 & 4 & 3 & -1 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 3 \end{pmatrix} \\
 &\xrightarrow{II/(-2)} \begin{pmatrix} 1 & 4 & 3 & -1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 3 \end{pmatrix} \\
 &\xrightarrow{III-3II, I+II} \begin{pmatrix} 1 & 4 & 3 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}
 \end{aligned}$$

We see that column 1 and 4 in A are pivot columns, so that these provide a basis for the column space of A .

- (b) Expand the basis you found in (a) to a basis for $\mathbb{R}^3$.

! Solution

We insert an extra column so that all rows have a pivot element, and

row-reduce backwards:

$$\begin{aligned}
 \begin{pmatrix} 1 & 4 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} &\stackrel{III+3II, I-II}{\sim} \begin{pmatrix} 1 & 4 & 3 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 & 1 \end{pmatrix} \\
 &\stackrel{II(-2)}{\sim} \begin{pmatrix} 1 & 4 & 3 & -1 & 0 \\ 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 3 & 1 \end{pmatrix} \\
 &\stackrel{II-4I, III+2I}{\sim} \begin{pmatrix} 1 & 4 & 3 & -1 & 0 \\ -4 & -16 & -12 & 2 & 0 \\ 2 & 8 & 6 & 1 & 1 \end{pmatrix} \\
 &\stackrel{I(-1)}{\sim} \begin{pmatrix} -1 & -4 & -3 & 1 & 0 \\ -4 & -16 & -12 & 2 & 0 \\ 2 & 8 & 6 & 1 & 1 \end{pmatrix} \\
 &\stackrel{I \leftrightarrow III}{\sim} \begin{pmatrix} 2 & 8 & 6 & 1 & 1 \\ -4 & -16 & -12 & 2 & 0 \\ -1 & -4 & -3 & 1 & 0 \end{pmatrix}
 \end{aligned}$$

We can therefore expand with the vector $(1, 0, 0)$.

Exercise 5.4. Suppose that $m > n$, and that the columns $\mathbf{a}_1, \dots, \mathbf{a}_n$ in A are linearly independent. Suppose also that the first n rows in A are linearly independent. Explain that

$$\{\mathbf{a}_1, \dots, \mathbf{a}_n, \mathbf{e}_{n+1}, \dots, \mathbf{e}_m\}$$

is a basis for $\mathbb{R}^m$.

! Solution

Do as follows:

1. Row-reduce the n first rows in A to the $n \times n$ identity matrix.
2. Null out the $m - n$ last rows with help of elementary row operations.
3. Insert $\mathbf{e}_{n+1}, \dots, \mathbf{e}_{n+m}$, so that we obtain the $m \times m$ identity matrix.
4. Row-reduce backwards.

After this we will still have zeros above the standard basis vectors we inserted, since the $m - n$ last rows never will affect the n first rows in the operations we did. The vectors $\mathbf{e}_{n+1}, \dots, \mathbf{e}_{n+m}$ which we inserted will also not be affected by adding the rows above to these. After backwards row-reduction we therefore get

$$\{\mathbf{a}_1, \dots, \mathbf{a}_n, \mathbf{e}_{n+1}, \dots, \mathbf{e}_m\},$$

which thus also must be a basis for $\mathbb{R}^m$.

Exercise 5.5 (Spans are subspaces). Show that $\text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$ is a subspace of $\mathbb{R}^m$

Exercise 5.6 (The orthogonal complement is a subspace). Let A be a set in $\mathbb{R}^n$. The *orthogonal complement* of A , $A^\perp$, is defined as the set of all vectors in $\mathbb{R}^n$ that are orthogonal to all vectors in A , meaning that

$$A^\perp = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \cdot \mathbf{a} = 0 \forall \mathbf{a} \in A\}.$$

Show that $A^\perp$ is a subspace of $\mathbb{R}^n$.

! Solution

Let $\mathbf{x}, \mathbf{y} \in A^\perp$, and let $\mathbf{a} \in A$. Then

$$\begin{aligned} (\mathbf{x} + \mathbf{y}) \cdot \mathbf{a} &= \mathbf{x} \cdot \mathbf{a} + \mathbf{y} \cdot \mathbf{a} = 0 + 0 = 0 \\ (c\mathbf{x}) \cdot \mathbf{a} &= c(\mathbf{x} \cdot \mathbf{a}) = c0 = 0. \end{aligned}$$

This shows that both $\mathbf{x} + \mathbf{y} \in A^\perp$, and that $c\mathbf{x} \in A^\perp$. It follows that $A^\perp$ is a subspace.

Exercise 5.7 (A sum of subspaces is a subspace). Let U and V be subspaces of $\mathbb{R}^n$. Show that

$$\{\mathbf{u} + \mathbf{v} : \mathbf{u} \in U, \mathbf{v} \in V\}$$

also is a subspace of $\mathbb{R}^n$.

Exercise 5.8. Prove Proposition 5.9 (i.e., the inverse of an isomorphism is also an isomorphism).

! Solution

Let $T : U \rightarrow V$ be an isomorphism, and let $S : V \rightarrow U$ be defined by $S(\mathbf{v}) = \mathbf{u}$ when $T(\mathbf{u}) = \mathbf{v}$. S is

- well-defined on all of V since T is onto,
- onto since T is defined on all of U ,
- $1 \rightarrow 1$ since T is well-defined.

For linearity of S , assume that $S(\mathbf{v}_1) = \mathbf{u}_1$ and $S(\mathbf{v}_2) = \mathbf{u}_2$. Then $T(\mathbf{u}_1) = \mathbf{v}_1$ and $T(\mathbf{u}_2) = \mathbf{v}_2$, so that

$$T(\mathbf{u}_1 + \mathbf{u}_2) = T(\mathbf{u}_1) + T(\mathbf{u}_2) = \mathbf{v}_1 + \mathbf{v}_2,$$

so that

$$S(\mathbf{v}_1 + \mathbf{v}_2) = \mathbf{u}_1 + \mathbf{u}_2 = S(\mathbf{v}_1) + S(\mathbf{v}_2).$$

Similarly, if $T(\mathbf{u}) = \mathbf{v}$ then $T(\alpha\mathbf{u}) = \alpha\mathbf{v}$, so that

$$S(\alpha\mathbf{v}) = T(\alpha\mathbf{u}) = \alpha T(\mathbf{u}) = \alpha S(\mathbf{v}).$$

Exercise 5.9. Show that any subspace can be written as the null space of a matrix.

 Hint

A general subspace can be written as $\text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$, where the vectors are linearly independent. Expand this to a basis for $\mathbb{R}^m$, and let A be the matrix with these as columns. Then consider the null space of the matrix

$$C = \left(\begin{array}{c|c} \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & I \end{array} \right) A^{-1}.$$

 Solution

Suppose that the subspace is $\text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$, where the vectors are linearly independent. Expand these to a basis for $\mathbb{R}^m$, and let A be the matrix with these as columns. Then we know that A is invertible, and that $A\mathbf{e}_i = \mathbf{a}_i$ for all i . Then we also have that $A^{-1}\mathbf{a}_i = \mathbf{e}_i$. Now set

$$C = \left(\begin{array}{c|c} \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & I \end{array} \right) A^{-1},$$

where the upper left block is $n \times n$. Then $C\mathbf{a}_1 = C\mathbf{a}_2 = \dots = C\mathbf{a}_n = \mathbf{0}$, so that

$$\text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n) \subseteq \text{Nul}(C).$$

We also have that

$$C\mathbf{a}_{n+1} = \mathbf{e}_{n+1}, C\mathbf{a}_{n+2} = \mathbf{e}_{n+2}, \dots, C\mathbf{a}_m = \mathbf{e}_m$$

If $\mathbf{x} \notin \text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$ then we can write

$$\mathbf{x} = c_1\mathbf{a}_1 + \dots + c_n\mathbf{a}_n + c_{n+1}\mathbf{a}_{n+1} + \dots + c_m\mathbf{a}_m,$$

where at least one of $c_{n+1}, \dots, c_m \neq 0$. We now get

$$C\mathbf{x} = c_{n+1}\mathbf{e}_{n+1} + \dots + c_m\mathbf{e}_m,$$

which then is $\neq \mathbf{0}$. It follows that $\mathbf{x} \notin \text{Nul}(C)$, so that $\text{Nul}(C) \subseteq \text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$. It follows that $\text{Nul}(C) = \text{Sp}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$.

Exercise 5.10. Let $U \subseteq \mathbb{R}^m$, $V \subseteq \mathbb{R}^m$ be subspaces, and $T : U \rightarrow V$ a linear transformation between them. Let also $\{\mathbf{x}_i\}_{i=1}^k$ be vectors in U .

- (a) Assume that $\{T(\mathbf{x}_i)\}_{i=1}^k$ are linearly independent. Show that $\{\mathbf{x}_i\}_{i=1}^k$ also are linearly independent.

! Solution

Assume that $c_1\mathbf{x}_1 + \cdots + c_k\mathbf{x}_k = \mathbf{0}$. Since T is linear we have that $c_1T(\mathbf{x}_1) + \cdots + c_kT(\mathbf{x}_k) = \mathbf{0}$, from which it follows that $c_1 = \cdots = c_k = 0$ due to linear independence of the $\{T(\mathbf{x}_i)\}_{i=1}^k$. Thus, $\{\mathbf{x}_i\}_{i=1}^k$ are also linearly independent.

- (b) Assume that T is $1 \rightarrow 1$, and that $\{\mathbf{x}_i\}_{i=1}^k$ are linearly independent. Show that $\{T(\mathbf{x}_i)\}_{i=1}^k$ are also linearly independent.

! Solution

Assume that $c_1T(\mathbf{x}_1) + \cdots + c_kT(\mathbf{x}_k) = \mathbf{0}$. Then we also have that $T(c_1\mathbf{x}_1 + \cdots + c_k\mathbf{x}_k) = \mathbf{0}$. Since T is $1 \rightarrow 1$ we must have that $c_1\mathbf{x}_1 + \cdots + c_k\mathbf{x}_k = \mathbf{0}$, so that $c_1 = \cdots = c_k = 0$ due to linear independence of the $\{\mathbf{x}_i\}_{i=1}^k$. It follows that $\{T(\mathbf{x}_i)\}_{i=1}^k$ are linearly independent.

- (c) Assume that T is onto and that $\{\mathbf{x}_i\}_{i=1}^k$ span U . Show that $\{T(\mathbf{x}_i)\}_{i=1}^k$ span V .

! Solution

Let $\mathbf{b} \in V$. Since T is onto there exists a linear combination $c_1\mathbf{x}_1 + \cdots + c_k\mathbf{x}_k$ so that $T(c_1\mathbf{x}_1 + \cdots + c_k\mathbf{x}_k) = \mathbf{b}$. But then $c_1T(\mathbf{x}_1) + \cdots + c_kT(\mathbf{x}_k) = \mathbf{b}$, so that $\{T(\mathbf{x}_i)\}_{i=1}^k$ also span V .

- (d) Assume that T is $1 \rightarrow 1$ and that $\{T(\mathbf{x}_i)\}_{i=1}^k$ span V . Show that $\{\mathbf{x}_i\}_{i=1}^k$ span U .

! Solution

Let $\mathbf{y} \in U$. Since $\{T(\mathbf{x}_i)\}_{i=1}^k$ span V there exist $c_1, \dots, c_k$ so that

$$c_1T(\mathbf{x}_1) + \cdots + c_kT(\mathbf{x}_k) = T(\mathbf{y}).$$

But then $T(c_1\mathbf{x}_1 + \cdots + c_k\mathbf{x}_k) = T(\mathbf{y})$. Since T is $1 \rightarrow 1$ we have that $c_1\mathbf{x}_1 + \cdots + c_k\mathbf{x}_k = \mathbf{y}$, so that $\{\mathbf{x}_i\}_{i=1}^k$ span U .

- (e) Assume that T is $1 \rightarrow 1$ and onto. Show that $\{\mathbf{x}_i\}_{i=1}^k$ is a basis for U if and only if $\{T(\mathbf{x}_i)\}_{i=1}^k$ is a basis for V .

! Solution

This follows from (a)-(d).

Exercise 5.11. Assume that P is invertible and $m \times m$, and that A is $m \times n$. Show that the column spaces of A and PA have the same dimensions.

! Solution

We have that $\text{Col}(PA) = \text{Span}\{P\mathbf{a}_1, \dots, P\mathbf{a}_n\}$. Since P is an isomorphism, linear dependence relations are preserved under P , which in particular means that the dimension will not change.

Exercise 5.12. Let $T : U \rightarrow V$ and $S : V \rightarrow W$ be isomorphisms. Prove that the mapping sending $\mathbf{x}$ to $S(T(\mathbf{x}))$ is an isomorphism from U to W .

! Solution

We have that

$$S(T(\mathbf{u}_1 + \mathbf{u}_2)) = S(T(\mathbf{u}_1) + T(\mathbf{u}_2)) = S(T(\mathbf{u}_1)) + S(T(\mathbf{u}_2)).$$

A similar argument can be made for multiplication with a scalar, so that the composition of S and T is linear.

Assume that $\mathbf{u}_1 \neq \mathbf{u}_2$ are vectors in U . Since T is $1 \rightarrow 1$ we have that $T(\mathbf{u}_1) \neq T(\mathbf{u}_2)$. Since S also is $1 \rightarrow 1$ it follows that $S(T(\mathbf{u}_1)) \neq S(T(\mathbf{u}_2))$, so that the composition of S and T also is $1 \rightarrow 1$.

Finally, let $\mathbf{w} \in W$. Since S is onto there exists a $\mathbf{v} \in V$ so that $S\mathbf{v} = \mathbf{w}$. And since T is onto there exists a $\mathbf{u} \in U$ so that $T(\mathbf{u}) = \mathbf{v}$. But then $S(T(\mathbf{u})) = S(\mathbf{v}) = \mathbf{w}$, so that the composition of S and T also is onto. Since this composition is onto, $1 \rightarrow 1$, and linear, it is also an isomorphism.

Exercise 5.13. Let $T : V \rightarrow W$ be a linear transformation between subspaces, and let $\mathcal{A}$ be a basis for V , and $\mathcal{B}$ a basis for W . Show that T is $1 \rightarrow 1$ /onto/an isomorphism if and only if $[T]_{\mathcal{B} \leftarrow \mathcal{A}}$ is $1 \rightarrow 1$ /onto/an isomorphism.

! Solution

$[T]_{\mathcal{B} \leftarrow \mathcal{A}}$ is the composition of T , one coordinate mapping, and one inverse coordinate mapping. Since the two latter are isomorphisms (between V/W and $\mathbb{R}^m/\mathbb{R}^n$), it follows that $[T]_{\mathcal{B} \leftarrow \mathcal{A}}$ is $1 \rightarrow 1$ /onto/an isomorphism if and only if T is. Since a matrix gives an isomorphism if and only if $m = n$, V and W must have the same dimension when T is an isomorphism.

Exercise 5.14. Find the coordinate vector of $\mathbf{x} = (-10, 20)$ relative to the

basis $\mathcal{B} = \{(2, 4), (-4, 2)\}$ for $\mathbb{R}^2$.

! Solution

We row-reduce

$$\begin{pmatrix} 2 & -4 & -10 \\ 4 & 2 & 20 \end{pmatrix} \sim \begin{pmatrix} 1 & -2 & -5 \\ 0 & 10 & 40 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 4 \end{pmatrix},$$

so that $(-10, 20) = 3(2, 4) + 4(-4, 2)$. It follows that $[\mathbf{x}]_{\mathcal{B}} = (3, 4)$.

Exercise 5.15. Consider the bases $\mathcal{A} = \{(-1, 1), (1, 1)\}$, and $\mathcal{B} = \{(-1, 2), (-1, 3)\}$.

(a) Find the change of coordinate matrix from $\mathcal{A}$ to $\mathcal{B}$.

! Solution

We row-reduce

$$\begin{pmatrix} -1 & -1 & -1 & 1 \\ 2 & 3 & 1 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 1 & -1 \\ 0 & 1 & -1 & 3 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 2 & -4 \\ 0 & 1 & -1 & 3 \end{pmatrix},$$

so that $(-1, 1) = 2(-1, 2) - (-1, 3)$ and $(1, 1) = -4(-1, 2) + 3(-1, 3)$, so that $[\mathbf{a}_1]_{\mathcal{B}} = (2, -1)$ $[\mathbf{a}_2]_{\mathcal{B}} = (-4, 3)$. It follows that

$$P_{\mathcal{B} \leftarrow \mathcal{A}} = ([\mathbf{a}_1]_{\mathcal{B}} \quad [\mathbf{a}_2]_{\mathcal{B}}) = \begin{pmatrix} 2 & -4 \\ -1 & 3 \end{pmatrix}.$$

(b) Let $\mathbf{x} = 4\mathbf{a}_1 - 5\mathbf{a}_2$. Find $[\mathbf{x}]_{\mathcal{B}}$.

! Solution

We have $[\mathbf{x}]_{\mathcal{A}} = (4, -5)$, so that

$$[\mathbf{x}]_{\mathcal{B}} = P_{\mathcal{B} \leftarrow \mathcal{A}}[\mathbf{x}]_{\mathcal{A}} = \begin{pmatrix} 2 & -4 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} 4 \\ -5 \end{pmatrix} = \begin{pmatrix} 28 \\ -19 \end{pmatrix}.$$

Exercise 5.16. Consider the linear transformation $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ with matrix $\begin{pmatrix} 2 & 6 \\ -6 & 2 \end{pmatrix}$. Find the matrix of T relative to $\mathcal{B} = \{(3, 1), (-1, 4)\}$, i.e., $[T]_{\mathcal{B}}$.

! Solution

We first find $T(\mathbf{b}_1) = (12, -16)$, $T(\mathbf{b}_2) = (22, 14)$. We then row-reduce

$$\begin{pmatrix} 3 & -1 & 12 & 22 \\ 1 & 4 & -16 & 14 \end{pmatrix} \sim \begin{pmatrix} 1 & 4 & -16 & 14 \\ 0 & -13 & 60 & -20 \end{pmatrix} \\ \sim \begin{pmatrix} 1 & 4 & -16 & 14 \\ 0 & 1 & -60/13 & 20/13 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 32/13 & 102/13 \\ 0 & 1 & -60/13 & 20/13 \end{pmatrix}$$

It follows that $[T(\mathbf{b}_1)]_{\mathcal{B}} = (32, -60)/13$, $[T(\mathbf{b}_2)]_{\mathcal{B}} = (102, 20)/13$, so that

$$[T]_{\mathcal{B}} = ([T(\mathbf{b}_1)]_{\mathcal{B}} \quad [T(\mathbf{b}_2)]_{\mathcal{B}}) = \frac{2}{13} \begin{pmatrix} 16 & 51 \\ -30 & 10 \end{pmatrix}.$$

Chapter 6

Determinants

In this chapter we will look at *determinants*. These are useful theoretically, with a long history in multivariable analysis. We will start with the general definition of determinants, which looks rather complex at a first glance. After this we will consider how elementary row operations change the determinant. From this it will be clear that the determinant says something about how a matrix transforms a volume in $\mathbb{R}^n$. We all have some intuition about volumes in $\mathbb{R}^3$ and areas in $\mathbb{R}^2$, but when it comes to volumes in $\mathbb{R}^n$ for $n > 3$, things are different. We shall define this, and see that, if A is an $n \times n$ -matrix and X a set in $\mathbb{R}^n$, then the set

$$AX = \{A\mathbf{x} : \mathbf{x} \in X\}$$

has a volume, written $V(AX)$, which depends linearly on the volume $V(X)$ of X - regardless how the set X must look. We thus have that $V(AX) = kV(X)$ for a constant k which only depends on the matrix A . k can be interpreted as a *magnification factor* for volumes for the matrix A , and we shall see that magnification factors change under elementary row operations in the same way as determinants do. From this we deduce that the absolute value of the determinant equals the magnification factor.

One of the things we will show is that the determinant of a product equals the product of the determinants. If we disregard the sign of the determinant, this is perhaps simpler to see when thinking in terms of magnification factors: If A_1 and A_2 have magnification factors k_1 and k_2 , respectively, then A_2X has volume $k_2V(X)$, so that A_1A_2X has volume $k_1(k_2V(X)) = k_1k_2V(X)$, so that A_1A_2 has magnification factor k_1k_2 .

We round off this chapter with a useful geometric interpretation of the determinant.

6.1 The definition of the determinant

The general definition of determinant is as follows

Definition 6.1 (Determinant). The *determinant* of an $n \times n$ -matrix A is defined recursively as follows:

1. The determinant of a 1×1 -matrix A is defined as a_{11} .
2. The determinant of an $n \times n$ -matrix A is defined in terms of determinants of $(n-1) \times (n-1)$ -submatrices of A as

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} & \cdots & a_{2n} \\ a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n2} & a_{n3} & \cdots & a_{nn} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n3} & \cdots & a_{nn} \end{vmatrix} \\ + \cdots + (-1)^{n+1} a_{1n} \begin{vmatrix} a_{21} & a_{22} & \cdots & a_{2(n-1)} \\ a_{31} & a_{32} & \cdots & a_{3(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n(n-1)} \end{vmatrix}$$

Row 1 of A has been left out in all submatrices on the right side. In the first submatrix column 1 of A has also been left out, in the second column 2 of A has been left out, and so on.

Vertical lines are used as notation for determinants above, but it is also common to write $\det(A)$ for the determinant of A . Let us also write $A^{(ij)}$ for the $(n-1) \times (n-1)$ -matrix we obtain by leaving out row i and column j of A . The quantities $\det(A^{(ij)})$ are also called *minors*. With this notation Equation 6.1 can be written as

$$\det(A) = a_{11} \det(A^{(11)}) - a_{12} \det(A^{(12)}) + a_{13} \det(A^{(13)}) - \cdots + a_{1n} (-1)^{n+1} \det(A^{(1n)}),$$

or even more compactly as

$$\det(A) = \sum_{j=1}^n (-1)^{j+1} a_{1j} \det(A^{(1j)}).$$

The determinant of a 1×1 -matrix is per definition the one component in the matrix, and for a 2×2 -matrix it is

$$\det(A) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21},$$

The determinant of a 3×3 -matrix is

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

Example 6.1 (A 3×3 -determinant). Let us compute the determinant of the matrix $\begin{pmatrix} 2 & 1 & 3 \\ 1 & 4 & 2 \\ 3 & 1 & -1 \end{pmatrix}$ using the definition. We get

$$\begin{vmatrix} 2 & 1 & 3 \\ 1 & 4 & 2 \\ 3 & 1 & -1 \end{vmatrix} = 2 \begin{vmatrix} 4 & 2 \\ 1 & -1 \end{vmatrix} - 1 \begin{vmatrix} 1 & 2 \\ 3 & -1 \end{vmatrix} + 3 \begin{vmatrix} 1 & 4 \\ 3 & 1 \end{vmatrix} \\ = 2(-4 - 2) - (-1 - 6) + 3(1 - 12) = -12 + 7 - 33 = -38.$$

We can check the answer with the built-in function for computing the determinant:

```
A=np.array([[2,1,3],[1,4,2],[3,1,-1]])
np.linalg.det(A)
```

```
-37.999999999999999
```



Let $K(n)$ be the number of operations needed to compute the determinant recursively using Equation 6.1. It is clear that

$$K(n) = nK(n-1) + n + n - 1 \geq nK(n-1).$$

We see from this that we must have $K(n) \geq n!$. When n is large this leads to too much computation, even for the most powerful computers. Later in this chapter we will see that one can compute the determinant more efficiently using Gaussian elimination. The definition above is more of theoretical interest - it does not give a good implementation even if it is easily programmed, as seen below.

```
def detdef(A):
    n = np.shape(A)[0]
    if n>1:
        d=0
        for k in range(n):
            colinds = np.arange(0,n)
            colinds=np.delete(colinds, k)
            d += (-1)**k*A[0,k]*detdef( A[1:,colinds] )
        return d
    else:
        return A.item()
```

One could guess that Equation 6.1 is not used to compute the determinant in Example 6.1: The answer there comes with four decimals, so the answer is not -38 exactly. Had Equation 6.1 been used, only integer arithmetic would have been needed, and the answer should be -38 exactly. It is straightforward to verify that this is the case:

```
detdef( np.array([[2,1,3],[1,4,2],[3,1,-1]]) )
```

-38

6.2 Properties of the determinant

Let us prove some useful properties of determinants. We start with the following.

Proposition 6.1. *We have that $\det(A) = 0$ if A has a row with zeros only, or a column with zeros only.*

Proof. This is simplest to see for rows with zeros only: If we use Equation 6.1 so many times that all sub-matrices start with a row of zeros, it is clear that the next application of Equation 6.1 will give 0. Suppose instead that we have a column of zeros. For 1×1 -matrices with a column of zeros it is clear that the determinant is 0. Suppose we have shown that all $(n-1) \times (n-1)$ -matrices with a column of zeros have determinant 0. When we use Equation 6.1 on an $n \times n$ -matrix with a column of zeros, we get a contribution from $n-1$ $(n-1) \times (n-1)$ -sub-matrices with columns of zeros. And the contribution from the remaining column is 0, since we have a zero on the corresponding place in the first row that is multiplied in. It follows that $n \times n$ -matrices with a zero column also have determinant 0. $\square$

Recall that a square matrix is called *upper triangular* if everything below the diagonal is 0, and *lower triangular* if everything above the diagonal is 0.

Proposition 6.2. *The determinant of an upper- or lower triangular matrix equals the product of the diagonal elements.*

Proof. The result is obvious for $n = 1$. If A is lower triangular Equation 6.1 says that $\det(A) = a_{11} \det(A^{(11)})$. Since $A^{(11)}$ also is lower triangular, and with $a_{22}, a_{33}, \dots, a_{nn}$ on the diagonal, it follows by induction that

$$\det(A) = a_{11}(a_{22}a_{33} \cdots a_{nn}) = a_{11}a_{22}a_{33} \cdots a_{nn}.$$

Suppose instead that A is upper triangular. The first column in the minors $A^{(12)}, A^{(13)}, \dots, A^{(1n)}$ are then all zero, and then Proposition 6.1 says that

$$\det(A^{(12)}) = \det(A^{(13)}) = \cdots = \det(A^{(1n)}) = 0.$$

Only the first minor contributes in Equation 6.1, so that $\det(A) = a_{11} \det(A^{(11)})$, which again by induction gives that $\det(A) = a_{11}a_{22}a_{33} \cdots a_{nn}$ $\square$

We have the following result on how elementary row operations change the determinant.

Proposition 6.3. *Let A and B be square matrices of equal dimension. The following hold:*

1. If B is obtained from A by a simple row interchange, then $\det(B) = -\det(A)$.
2. If B is obtained by multiplying a row in A with s , then $\det(B) = s \det(A)$.
3. If B is obtained by adding s times row i to row j in A , then $\det(B) = \det(A)$.

Proof. We prove 1.. Let D_{ij} be the determinant of the matrix obtained by removing row 1 and 2 in A , as well as column i and j in A . We have that

$$\begin{aligned}\det(A) &= \sum_{i < j} ((-1)^{i+1}(-1)^j a_{1i} a_{2j} + (-1)^{j+1}(-1)^{i+1} a_{1j} a_{2i}) D_{ij} \\ &= \sum_{i < j} (-1)^{i+j} (a_{1j} a_{2i} - a_{1i} a_{2j}) D_{ij}\end{aligned}$$

If B is the matrix where row 1 and 2 in A are swapped, we have that $b_{1i} = a_{2i}$ and $b_{1j} = a_{2j}$ and so on, and we get

$$\begin{aligned}\det(B) &= \sum_{i < j} (-1)^{i+j} (b_{1j} b_{2i} - b_{1i} b_{2j}) D_{ij} = \sum_{i < j} (-1)^{i+j} (a_{2j} a_{1i} - a_{2i} a_{1j}) D_{ij} \\ &= - \sum_{i < j} (-1)^{i+j} (a_{2i} a_{1j} - a_{2j} a_{1i}) D_{ij} = -\det(A)\end{aligned}$$

If two other neighbouring rows are swapped it follows by induction, and from the definition $\det(A) = \sum_{j=1}^n (-1)^{j+1} a_{1j} A_{1j}$ that the determinant also changes sign. Further, if row i and j are not neighbouring rows, a row interchange of these can be expressed as $j - i + (j - i - 1) = 2(j - i) - 1$ neighbouring row interchanges. Since this is an odd number the determinant also then will change sign. This proves 1..

The proofs for 2. and 3. are given as exercises. $\square$

If A has two equal rows, we end up with the same matrix if these are interchanged. As a consequence of 1. above we must have that $\det(A) = -\det(A)$, so that $\det(A) = 0$.

Corollary 6.1. $\det(A) = 0$ if A has two identical rows.

This result is useful when it comes to proving 3. above. We can also expand the determinant along any given row (This is an exercise):

Proposition 6.4 (Determinants can be expanded along any row). *For all $1 \leq i \leq n$ we have that*

$$\det(A) = \sum_{j=1}^n (-1)^{i+j} a_{ij} \det(A^{(ij)}).$$

If we in particular set $A = I_n$ (which has determinant 1) and $B = E$ (where E is an elementary matrix) in Proposition 6.3 we get that

$$\det(E) = \begin{cases} -1 & \text{if } E \text{ is a row interchange.} \\ s & \text{if } E \text{ multiplies a row with } s. \\ 1 & \text{if } E \text{ adds a multiple of one row to another.} \end{cases} \quad (6.1)$$

We can now prove the following.

Proposition 6.5. *If A and B both are square we have that $\det(AB) = \det(A)\det(B)$.*

Proof. we can write

$$A = E_1 E_2 \cdots E_k C \qquad B = F_1 F_2 \cdots F_l D$$

where $E_1, F_1, E_2, F_2, \dots$ are elementary matrices, and C and D are reduced echelon forms. If $C = D = I_n$ then

$$\begin{aligned} \det(AB) &= \det(E_1 E_2 \cdots E_k F_1 F_2 \cdots F_l) = \det(E_1) \det(E_2) \cdots \det(E_k) \det(F_1 F_2 \cdots F_l) \\ &= \det(E_1 E_2 \cdots E_k) \det(F_1 F_2 \cdots F_l) = \det(A) \det(B). \end{aligned}$$

Suppose then that one of $C, D \neq I_n$. It follows that one of $\det(A)$ or $\det(B)$ equals 0, so that also $\det(A)\det(B) = 0$. It is therefore enough to show that also $\det(AB) = 0$. It is clear that $\det(AB) = 0$ if and only if $\det(CF_1 F_2 \cdots F_l D) = 0$. We consider two possibilities:

1. If $C = I_n, D \neq I_n$: Then $\det(D) = 0$ (since D has a row of zeros). But then

$$\det(CF_1 F_2 \cdots F_l D) = \det(F_1 F_2 \cdots F_l D) = 0.$$

2. If $C \neq I_n$ then C has a row of zeros. But then also $CF_1 F_2 \cdots F_l D$ has a row of zeros, and then $\det(CF_1 F_2 \cdots F_l D) = 0$

□

In particular we can find the determinant with the help of Gaussian elimination. Let us show this with an example.

Example 6.2 (A 3×3 -determinant using Gaussian elimination). Let us verify the determinant we found in Example 6.1, using Gaussian elimination. We get

$$\begin{aligned} \begin{pmatrix} 2 & 1 & 3 \\ 1 & 4 & 2 \\ 3 & 1 & -1 \end{pmatrix} &\sim \begin{pmatrix} 1 & 4 & 2 \\ 2 & 1 & 3 \\ 3 & 1 & -1 \end{pmatrix} \sim \begin{pmatrix} 1 & 4 & 2 \\ 0 & -7 & -1 \\ 0 & -11 & -7 \end{pmatrix} \sim \begin{pmatrix} 1 & 4 & 2 \\ 0 & 1 & 1/7 \\ 0 & -11 & -7 \end{pmatrix} \\ &\sim \begin{pmatrix} 1 & 4 & 2 \\ 0 & 1 & 1/7 \\ 0 & 0 & -38/7 \end{pmatrix} \end{aligned}$$

The first row operation was a row interchange. This is followed by two operations where we add a multiple of row 1 to the other rows. Then we scale row 2 with $(-1/7)$, before adding a multiple of row 2 to row 3. We ended up with an upper triangular matrix, where we know that the determinant is the product of the diagonal elements. We then get

$$\begin{vmatrix} 1 & 4 & 2 \\ 0 & 1 & 1/7 \\ 0 & 0 & -38/7 \end{vmatrix} = 1 \times (-1/7) \times (1) \times (-1) \begin{vmatrix} 2 & 1 & 3 \\ 1 & 4 & 2 \\ 3 & 1 & -1 \end{vmatrix}$$

so that

$$(-38/7) = (1/7) \begin{vmatrix} 2 & 1 & 3 \\ 1 & 4 & 2 \\ 3 & 1 & -1 \end{vmatrix},$$

so that the determinant is -38 . ♣

For large matrices (which often is the case in applications) it is more efficient to compute determinants by Gaussian elimination as above, when compared with using the definition of the determinant as in Example 6.1: In Section 3.1 we showed that the number of operations needed by Gaussian elimination is about $2n^3/3$, while using the definition needed at least $n!$ operations.

Again we can expand the theorem for invertible matrices with an equivalence. This will be the final version of this theorem:

Proposition 6.6 (Invertible matrix theorem). *The following are equivalent for an $n \times n$ -matrix A :*

1. A is row equivalent with I_n .
2. $A\mathbf{x} = \mathbf{0}$ has the unique solution $\mathbf{x} = \mathbf{0}$.
3. $A\mathbf{x} = \mathbf{b}$ has a unique solution for all right hand sides $\mathbf{b}$.
4. A can be written as a product of elementary matrices.
5. A is invertible.
6. The columns of A provide a basis for $\mathbb{R}^n$.
7. $\det(A) \neq 0$

Proof. 4. $\Rightarrow$ 7.: If $A = E_k \cdots E_1$ then

$$\det(A) = \det(E_k \cdots E_1) = \det(E_k) \cdots \det(E_1) \neq 0,$$

where we used Proposition 6.5, and that elementary matrices have a determinant different from 0.

7. $\Rightarrow$ 1. Suppose that A is not row equivalent with I_n . Then we can write $C = E_k \cdots E_1 A$, and $A = F_1 \cdots F_k C$, where $C \neq I_n$ is the reduced echelon form of A . But then $\det(C) = 0$, and it follows from Proposition 6.5 that $\det(A) = 0$. $\square$

Proposition 6.7. *For all square matrices A we have that $\det(A) = \det(A^T)$.*

Proof. It is clear that for all elementary matrices E , we have that $\det(E) = \det(E^T)$. Suppose $A = E_1 E_2 \cdots E_k C$, where C is the reduced echelon form of A . We have that

$$\begin{aligned}\det(A^T) &= \det(C^T E_k^T E_{k-1}^T \cdots E_1^T) = \det(C^T) \det(E_k^T) \det(E_{k-1}^T) \cdots \det(E_1^T) \\ &= \det(E_1) \det(E_2) \cdots \det(E_k) \det(C) = \det(E_1 E_2 \cdots E_k C) = \det(A).\end{aligned}$$

We used here that $\det(C) = \det(C^T)$. It is clear that this holds if $C = I_n$. If $C \neq I_n$ then C has a row of zeros, so that $\det(C) = 0$. But then C^T has a column of zeros, so that also $\det(C^T) = 0$. $\square$

From this we can deduce that we also can expand the determinant along any column (this is an exercise).

Proposition 6.8 (Determinants can be expanded along any column). *For all $1 \leq j \leq n$ we have that*

$$\det(A) = \sum_{i=1}^n (-1)^{i+j} a_{ij} \det(A^{(ij)}).$$

From Proposition 6.4 and Proposition 6.8 we see that the minor $\det(A^{(ij)})$ should have an alternating sign $(-1)^{i+j}$, regardless of which row or column we expand along. This can also be indicated with the following sign table:

$$\begin{pmatrix} + & - & + & - & \cdots \\ - & + & - & + & \cdots \\ + & - & + & - & \cdots \\ - & + & - & + & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

For practical use we should expand the determinant along the row/column which has the most zeros.

Example 6.3. Let us consider the 4×4 -matrix $A = \begin{pmatrix} 2 & 0 & 2 & 3 \\ 1 & 0 & 0 & 2 \\ 3 & 1 & -1 & 3 \\ -1 & 0 & 2 & 3 \end{pmatrix}$. We

see quickly that the row/column with most zeros is column 2. Expanding along this we get

$$\begin{aligned}\begin{vmatrix} 2 & 0 & 2 & 3 \\ 1 & 0 & 0 & 2 \\ 3 & 1 & -1 & 3 \\ -1 & 0 & 2 & 3 \end{vmatrix} &= (-1)^{3+2} \begin{vmatrix} 2 & 2 & 3 \\ 1 & 0 & 2 \\ -1 & 2 & 3 \end{vmatrix} = - \left((-1)^{2+1} \begin{vmatrix} 2 & 3 \\ 2 & 3 \end{vmatrix} + (-1)^{2+3} 2 \begin{vmatrix} 2 & 2 \\ -1 & 2 \end{vmatrix} \right) \\ &= -(-(-6-6) - 2(4+2)) = 12.\end{aligned}$$

Here we also expanded along row 2, where there also was a zero. ♣

6.3 Magnification factors for matrices(*)

Let us now consider the geometrical interpretation of the determinant as a magnification factor. When we now explain what we mean by the *volume* $V(X)$ of a set X in $\mathbb{R}^n$ for $n > 3$, let us list some properties $V(X)$ intuitively should have:

1. *Unit boxes*

$$B = \{\mathbf{x} \in \mathbb{R}^n : 0 \leq x_i \leq 1 \forall i\}$$

should have volume 1.

2. If X is *translated* with a vector $\mathbf{r}$, then the volume should stay the same. In formula: $V(X + \mathbf{r}) = V(X)$, where $X + \mathbf{r} = \{\mathbf{x} + \mathbf{r} : \mathbf{x} \in X\}$.
3. If X is *stretched* with a factor r in direction $\mathbf{e}_i$, the volume should be multiplied with r . In formula: $V(X_{i,r}) = rV(X)$, where

$$X_{i,r} = \{\mathbf{x} : (x_1, \dots, x_{i-1}, x_i/r, x_{i+1}, \dots, x_n) \in X\}.$$

4. If X_1 and X_2 are disjoint sets then $V(X_1 \cup X_2) = V(X_1) + V(X_2)$.
5. Volume should equal base times height, as we are used to in $\mathbb{R}^2$ and $\mathbb{R}^3$. But now we must think of the base as a volume in $\mathbb{R}^{n-1}$.

If we use 1. 2., and 3. repeatedly on a unit box, the boxes

$$B_{\mathbf{r},\delta} = \{\mathbf{x} \in \mathbb{R}^n : r_i\delta \leq x_i \leq (r_i + 1)\delta \forall i\}$$

should have volume δ^n for all choices of vectors $\mathbf{r}$. This gives us the partition

$$\mathbb{R}^n = \cup_{\mathbf{r}} B_{\mathbf{r},\delta}.$$

Most sets in $\mathbb{R}^n$ can be approximates arbitrarily well with the help of enough, and small enough, such boxes (more generally volume should be defined as a limit where $\delta \rightarrow 0$). Let us therefore restrict ourselves to regions which can be written as a finite union of boxes. Due to 4. we must have

$$V(\cup_i B_{\mathbf{r}_i,\delta}) = \sum_i \delta^n.$$

If $X = \cup_i B_{\mathbf{r}_i,\delta}$, and A is invertible, it is clear that $AX = A(\cup_i B_{\mathbf{r}_i,\delta}) = \cup_i A(B_{\mathbf{r}_i,\delta})$ is a partition of AX into disjoint sets.

Proposition 6.9. *Let $X = \cup_i B_{\mathbf{r}_i,\delta}$ be a disjoint union of boxes. Then $V(AX) = V(AB)V(X)$.*

This means that $V(AB)$ works as a *magnification factor* for A for sets X on the form $\cup_i B_{\mathbf{r}_i,\delta}$. The idea is that, since $V(AX) = V(AB)V(X)$ for such X , and since most sets can be approximated arbitrarily well with such sets, the also $V(AX) = V(AB)V(X)$ for more general sets X . Thus, $V(AB)$ works as a magnification factor more generally sets as well!

Proof. Since $B_{\mathbf{r},\delta} = \delta(B + \mathbf{r})$ we have that

$$AB_{\mathbf{r},\delta} = A(\delta(B + \mathbf{r})) = \delta(AB + A\mathbf{r}).$$

From property 2. and 3. it follows that $V(AB_{\mathbf{r},\delta}) = \delta^n V(AB)$. If we use property 4. we now get

$$\begin{aligned} V(AX) &= V(A(\cup_i B_{\mathbf{r}_i,\delta})) = V(\cup_i AB_{\mathbf{r}_i,\delta}) = \sum_i V(AB_{\mathbf{r}_i,\delta}) \\ &= \sum_i \delta^n V(AB) = V(AB) \sum_i \delta^n = V(AB)V(X). \end{aligned}$$

□

Since $B = \{\sum_{i=1}^n x_i \mathbf{e}_i : 0 \leq x_i \leq 1 \forall i\}$ then

$$AB = \left\{ \sum_{i=1}^n x_i A\mathbf{e}_i : 0 \leq x_i \leq 1 \forall i \right\} = \left\{ \sum_{i=1}^n x_i \mathbf{a}_i : 0 \leq x_i \leq 1 \forall i \right\},$$

where $\mathbf{a}_i$ is column i in A . More generally, if $\mathbf{a}_1, \dots, \mathbf{a}_m$ are vectors in $\mathbb{R}^n$, the set

$$F(\mathbf{a}_1, \dots, \mathbf{a}_m) := \left\{ \sum_{i=1}^m x_i \mathbf{a}_i, 0 \leq x_i \leq 1 \forall i \right\}$$

is called the *parallelepiped* spanned by $\{\mathbf{a}_i\}_{i=1}^m$ (parallelograms correspond to setting $m = 2$). Our magnification factors are thus volumes of parallelepipeds. Figure 6.1 shows a parallelepiped spanned by three vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, in space.

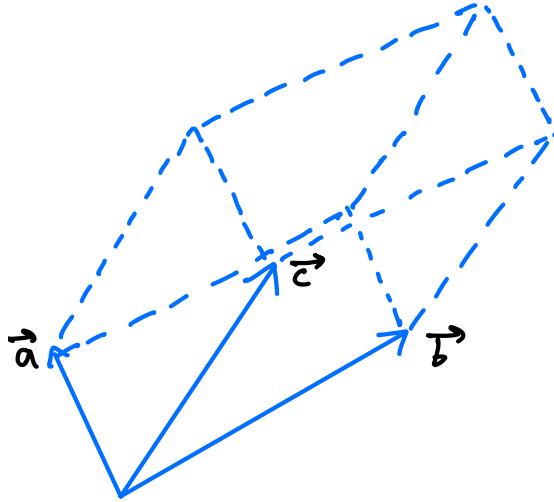


Figure 6.1: A parallelepiped in space.

Since A can be written as $E_k \cdots E_1$, where the E_i are elementary, we shall look at how volumes are magnified for every such. There are 3 different types of elementary matrices we will look at.

Proposition 6.10. *The magnification factor of an elementary matrix E is given by*

$$V(EB) = \begin{cases} 1 & \text{if } E \text{ is a row interchange.} \\ |s| & \text{if } E \text{ multiplies a row with } s. \\ 1 & \text{if } E \text{ adds a multiple of one row to another.} \end{cases}$$

Proof. We go through the three possibilities.

1. Suppose E swaps row i and j . Then $EB = B$, so that $V(EB) = 1$.
2. If E multiplies row i with s then $EB = \{\mathbf{x} : 0 \leq x_i \leq s, 0 \leq x_j \leq 1, j \neq i\}$, so that $V(EB) = s$ due to point 3..
3. Suppose E adds c times row n to row $n - 1$ (other choices of rows are handled similarly), i.e.,

$$E(x_1, \dots, x_{n-1}, x_n) = (x_1, \dots, x_{n-1} + cx_n, x_n).$$

For the cross section where $x_n = k$ we get

$$E(x_1, \dots, x_{n-1}, k) = (x_1, \dots, x_{n-1}, k) + (0, \dots, ck, 0),$$

so that the cross section where $x_n = k$ is translated by E with a constant vector $(0, \dots, ck, 0)$, so that all cross sections of EB on the form $x_n = k$ have the same area as B . This means that both base and height ($x_n = 1$) are equal for B and EB . It follows then from point 5. that $V(EB)$ is 1.

□

Comparing with Equation 6.1 we now see that $|\det(E)| = V(EB)$ for all elementary matrices E , and from Proposition 6.3 applied several times it follows that

$$\det(E_k \cdots E_1) = \det(E_k) \cdots \det(E_1).$$

Since also

$$V(E_k \cdots E_1 B) = V(E_k B) \cdots V(E_1 B)$$

it follows from $|\det(E_i)| = V(E_i B)$ that $|\det(A)| = V(AB)$, as long as A can be written as a product of elementary matrices (i.e., that A is invertible). Vi har dermed vist følgende:

Proposition 6.11. *If A is invertible then the magnification factor $V(AB)$ equals $|\det(A)|$.*

For $n = 2$ this says in particular that $\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$ equals the area of the parallelogram spanned by $\mathbf{a}_1$ and $\mathbf{a}_2$. If we set $\mathbf{a}_1 = |\mathbf{a}_1|(\cos \theta_1, \sin \theta_1)$, and $\mathbf{a}_2 = |\mathbf{a}_2|(\cos \theta_2, \sin \theta_2)$ then

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = \begin{vmatrix} |\mathbf{a}_1| \cos \theta_1 & |\mathbf{a}_1| \sin \theta_1 \\ |\mathbf{a}_2| \cos \theta_2 & |\mathbf{a}_2| \sin \theta_2 \end{vmatrix} = |\mathbf{a}_1||\mathbf{a}_2|(\cos \theta_1 \sin \theta_2 - \sin \theta_1 \cos \theta_2) \\ = |\mathbf{a}_1||\mathbf{a}_2| \sin(\theta_2 - \theta_1)$$

This was also deduced in Exercise 1.9.

The magnification vector $A = E_k \cdots E_1$ is now obtained by multiplying the magnification factors from the elementary matrices.

6.4 Geometrical interpretation of the determinant. The vector product.

If we define

$$\mathbf{b} = (\det(A^{(11)}), -\det(A^{(12)}), \det(A^{(13)}), \dots, (-1)^{n+1} \det(A^{(1n)})), \quad (6.2)$$

then the definition of the determinant given by Equation 6.1 can be written as the scalar product of $\mathbf{a}_1 \cdot \mathbf{b}$ ($\mathbf{a}_i$ is here row i in A). This gives us the following geometrical interpretation.

Proposition 6.12. *We have that*

1. $\mathbf{b}$ is orthogonal to all $\mathbf{a}_2, \dots, \mathbf{a}_n$,
2. $|\mathbf{b}| = V(F(\mathbf{a}_2, \dots, \mathbf{a}_n))$,
3. $|\mathbf{a}_1 \cdot \mathbf{b}| = V(AB) = |\det(A)|$.

Shortly we will look at the proof of this, but let us first take a closer look at the case $n = 3$. Then we have that

$$\mathbf{b} = (a_{22}a_{33} - a_{23}a_{32}, -(a_{21}a_{33} - a_{23}a_{31}), a_{21}a_{32} - a_{22}a_{31}). \quad (6.3)$$

This also has its own name:

Definition 6.2. Equation 6.3 is called the *vector product* of $\mathbf{a}_2$ and $\mathbf{a}_3$. We also write $\mathbf{a}_2 \times \mathbf{a}_3$ for the vector product of $\mathbf{a}_2$ and $\mathbf{a}_3$.

It is common to write the vector product as

$$\begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \\ = (a_{22}a_{33} - a_{23}a_{32})\mathbf{e}_1 - (a_{21}a_{33} - a_{23}a_{31})\mathbf{e}_2 + (a_{21}a_{32} - a_{22}a_{31})\mathbf{e}_3 \\ = (a_{22}a_{33} - a_{23}a_{32}, -(a_{21}a_{33} - a_{23}a_{31}), a_{21}a_{32} - a_{22}a_{31}),$$

i.e., we have inserted vectors instead of scalars in the first row in the matrix.

In particular $\mathbf{a}_2 \times \mathbf{a}_3$ is orthogonal to $\mathbf{a}_2$ and $\mathbf{a}_3$, with length equal to the area of the parallelogram spanned by the two vectors. These two properties do not characterise the vector product completely, since if $\mathbf{b}$ satisfy these two properties, then $-\mathbf{b}$ will as well! It can be shown, however, that $\mathbf{b} = \mathbf{a}_2 \times \mathbf{a}_3$ also satisfies what is called the *right hand rule*, something $-\mathbf{b}$ not does:

Definition 6.3. Let the right index finger point in the same direction as $\mathbf{a}_2$, and let the middle finger point in the same direction as $\mathbf{a}_3$, in such a way that there is less than 180 degrees between the two. The thumb will now point in the direction of the vector product $\mathbf{a}_2 \times \mathbf{a}_3$, perpendicular to both $\mathbf{a}_2$ and $\mathbf{a}_3$.

Proposition 6.13. $\mathbf{b} = \mathbf{a}_2 \times \mathbf{a}_3$ is the unique vector orthogonal to both $\mathbf{a}_2$ and $\mathbf{a}_3$, with length equal to the area of the parallelogram spanned by $\mathbf{a}_2$ and $\mathbf{a}_3$, and with direction given by the right hand rule.

We will not prove that the direction of the vector product abides to the right hand rule. A 3×3 determinant can also be written as $\det(A) = \mathbf{a}_1 \cdot (\mathbf{a}_2 \times \mathbf{a}_3)$, which due to Proposition 6.11 also equals the volume of the parallelepiped spanned by $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$.

Note that $\mathbf{x} \times \mathbf{y} = \mathbf{0}$ if $\mathbf{x}$ and $\mathbf{y}$ are parallel. This can be seen by observing that the parallelogram the two span then has area 0. The length of the vector product is then also 0, and then $\mathbf{x} \times \mathbf{y} = \mathbf{0}$. One can also argue that the vector product then is a determinant where the second and third rows are equal (after a scaling), which is 0 due to Corollary 6.1.

Example 6.4. Let $\mathbf{a}_1 = (1, 3, 2)$, and $\mathbf{a}_2 = (2, -1, 3)$. Then we have that

$$\begin{aligned} \mathbf{a}_1 \times \mathbf{a}_2 &= \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ 1 & 3 & 2 \\ 2 & -1 & 3 \end{vmatrix} \\ &= (3 \cdot 3 - 2 \cdot (-1))\mathbf{e}_1 - (1 \cdot 3 - 2 \cdot 2)\mathbf{e}_2 + (1 \cdot (-1) - 3 \cdot 2)\mathbf{e}_3 \\ &= (9 + 2, -(3 - 4), -1 - 6) = (11, 1, -7). \end{aligned}$$

The vector product can also be computed with help of the command `cross`:

```
a1 = [1,3,2]
a2 = [2,-1,3]
np.cross(a1,a2)
```

```
array([11,  1, -7])
```



Example 6.5. In Example 1.7 we found the expression for a plane through a given point, and with a given normal vector. If we instead have three points, we can use the vector product to find a normal vector, and then also an expression for the plane: If the three points are $\mathbf{x}_0$, $\mathbf{x}_1$, and $\mathbf{x}_2$, then clearly $\mathbf{x}_1 - \mathbf{x}_0$ and

$\mathbf{x}_2 - \mathbf{x}_0$ will be vectors in this plane, and the vector product of these will then give a normal vector.

To illustrate this, consider the three points $\mathbf{a} = (4, -2, -1)$, $\mathbf{b} = (1, 0, 0)$, and $\mathbf{c} = (2, -1, 0)$. We first get that $\mathbf{b} - \mathbf{a} = (-3, 2, 1)$, and $\mathbf{c} - \mathbf{a} = (-2, 1, 1)$. The vector product of these is

$$\begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ -3 & 2 & 1 \\ -2 & 1 & 1 \end{vmatrix} = (2-1)\mathbf{e}_1 - (-3+2)\mathbf{e}_2 + (-3+4)\mathbf{e}_3 = \mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3 = (1, 1, 1).$$

From this it follows that the plane can be written on the form $x + y + z = 1$.

By the properties of the vector product we also have that

- The area of the parallelogram spanned by $\mathbf{a}$ and $\mathbf{b}$ is the length of the vector

$$\begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ 4 & -2 & -1 \\ 1 & 0 & 0 \end{vmatrix} = (0, -1, 2),$$

which is $\sqrt{5}$.

- The volume of the parallelepiped spanned by $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, is

$$\begin{vmatrix} 4 & -2 & -1 \\ 1 & 0 & 0 \\ 2 & -1 & 0 \end{vmatrix} = - \begin{vmatrix} -2 & -1 \\ -1 & 0 \end{vmatrix} = 1.$$



Proof. (for Proposition 6.12) We first prove point 1.. Define the matrix

$$A_0 = \begin{pmatrix} \mathbf{a}_i \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \vdots \\ \mathbf{a}_n \end{pmatrix},$$

where $\mathbf{a}_i$ is inserted in both row 1 and i . Since a matrix with two equal rows has determinant 0 (Corollary 6.1), we have that

$$0 = \det(A_0) = \mathbf{a}_i \cdot \mathbf{b} \text{ for } i = 2, \dots, n,$$

so that $\mathbf{b}$ is orthogonal to all $\{\mathbf{a}_i\}_{i=2}^n$.

To see that $|\mathbf{b}| = V(F(\mathbf{a}_2, \dots, \mathbf{a}_n))$ (punkt 2.), observe first that the projection of $\mathbf{a}_1$ onto $\mathbf{b}$ is $\frac{\mathbf{a}_1 \cdot \mathbf{b}}{|\mathbf{b}|^2} \mathbf{b}$, which has length $\frac{|\mathbf{a}_1 \cdot \mathbf{b}|}{|\mathbf{b}|}$. This equals the distance from $\mathbf{a}_1$ to the plane spanned by $\mathbf{a}_2, \dots, \mathbf{a}_n$. Since volume equals base times height we get that

$$V(AB) = V(F(\mathbf{a}_1, \dots, \mathbf{a}_n)) = \frac{|\mathbf{a}_1 \cdot \mathbf{b}|}{|\mathbf{b}|} V(F(\mathbf{a}_2, \dots, \mathbf{a}_n)).$$

Since $|\det(A)| = V(AB)$ (Proposition 6.11) it follows that

$$|\mathbf{a}_1 \cdot \mathbf{b}| = \frac{|\mathbf{a}_1 \cdot \mathbf{b}|}{|\mathbf{b}|} V(F(\mathbf{a}_2, \dots, \mathbf{a}_n))$$

so that $|\mathbf{b}| = V(F(\mathbf{a}_2, \dots, \mathbf{a}_n))$.

Point 3. follows directly from the definition of the determinant given by Equation 6.1, and the connection we have proved between volume and determinant. $\square$

The result above is perhaps simplest to interpret on (the obviously true) form

$$|\mathbf{a}_1 \cdot \mathbf{b}| = \frac{|\mathbf{a}_1 \cdot \mathbf{b}|}{|\mathbf{b}|} |\mathbf{b}|.$$

On the left side here stands the volume of the parallelepiped spanned by $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$ ($V(AB)$). On the right side $\frac{|\mathbf{a}_1 \cdot \mathbf{b}|}{|\mathbf{b}|}$ is the height over the plane spanned by $\mathbf{a}_2, \mathbf{a}_3, \dots, \mathbf{a}_n$, while $|\mathbf{b}|$ is the area of the parallelepiped spanned by $\mathbf{a}_2, \mathbf{a}_3, \dots, \mathbf{a}_n$ ($V(F(\mathbf{a}_2, \dots, \mathbf{a}_n))$). In other words, volume equals base times height!

Exercises

Exercise 6.1. Compute

- (a) $\begin{vmatrix} 4 & -1 \\ 2 & 3 \end{vmatrix}$
- (b) $\begin{vmatrix} 2 & 0 & 1 \\ 3 & -1 & 7 \\ 0 & 2 & 5 \end{vmatrix}$
- (c) $(2, 0, 4) \times (-1, 2, 1)$.

Exercise 6.2 (Properties of the vector product). Let $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ be vectors in $\mathbb{R}^3$. Show that

- (a) $\mathbf{x} \times \mathbf{y} = -\mathbf{y} \times \mathbf{x}$.
- (b) For all scalars s , $\mathbf{x} \times (s\mathbf{y}) = (s\mathbf{x}) \times \mathbf{y} = s(\mathbf{x} \times \mathbf{y})$.
- (c) $\mathbf{x} \times (\mathbf{y} + \mathbf{z}) = \mathbf{x} \times \mathbf{y} + \mathbf{x} \times \mathbf{z}$.

Exercise 6.3. Let $\mathbf{x}$ be a given vector in $\mathbb{R}^3$. From 2. and 3. in the preceding exercise it follows that the mapping $\mathbf{y} \rightarrow \mathbf{x} \times \mathbf{y}$ (which goes from $\mathbb{R}^3$ to $\mathbb{R}^3$) is linear. What is the matrix of this linear mapping?

! Solution

Define the mapping T by $T(\mathbf{y}) = \mathbf{x} \times \mathbf{y}$. We get that

$$\begin{aligned}
\bullet T(\mathbf{e}_1) &= \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ x_1 & x_2 & x_3 \\ 1 & 0 & 0 \end{vmatrix} = \begin{vmatrix} \mathbf{e}_2 & \mathbf{e}_3 \\ x_2 & x_3 \end{vmatrix} = x_3\mathbf{e}_2 - x_2\mathbf{e}_3 = (0, x_3, -x_2) \\
\bullet T(\mathbf{e}_2) &= \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ x_1 & x_2 & x_3 \\ 0 & 1 & 0 \end{vmatrix} = -\begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_3 \\ x_1 & x_3 \end{vmatrix} = -(x_3\mathbf{e}_1 - x_1\mathbf{e}_3) = \\
&\quad (-x_3, 0, x_1) \\
\bullet T(\mathbf{e}_3) &= \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ x_1 & x_2 & x_3 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 \\ x_1 & x_2 \end{vmatrix} = x_2\mathbf{e}_1 - x_1\mathbf{e}_2 = (x_2, -x_1, 0)
\end{aligned}$$

The matrix of T is thus $\begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}$.

Exercise 6.4. We are given the three points $\mathbf{a} = (-1, 2, -2)$, $\mathbf{b} = (2, -1, 4)$, and $\mathbf{c} = (3, 3, 1)$.

(a) Find an expression for the plane going through the three points.

! Solution

We find first $\mathbf{b} - \mathbf{a} = (3, -3, 6)$ and $\mathbf{c} - \mathbf{a} = (4, 1, 3)$. A normal vector to the plane is

$$(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a}) = \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ 3 & -3 & 6 \\ 4 & 1 & 3 \end{vmatrix} = (-9-6)\mathbf{e}_1 - (9-24)\mathbf{e}_2 + (3-12)\mathbf{e}_3 = (-15, 15, 15)$$

The plane can thus be written $-15x + 15y + 15z = d$ for some constant d . Inserting the point $\mathbf{a} = (-1, 2, -2)$ gives that $d = 15 + 30 - 30 = 15$, so that the plane can be written as $-15x + 15y + 15z = 15$, which can be simplified to $-x + y + z = 1$.

(b) Find the area of the parallelogram spanned by $\mathbf{a}$ and $\mathbf{b}$.

! Solution

We compute

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ -1 & 2 & -2 \\ 2 & -1 & 4 \end{vmatrix} = (8-2)\mathbf{e}_1 - (-4+4)\mathbf{e}_2 + (1-4)\mathbf{e}_3 = (6, 0, -3).$$

The length of this vector is $\sqrt{45} = 3\sqrt{5}$, which also is the area of the parallelogram.

(c) Find the volume of the parallelepiped spanned by **a**, **b**, and **c**.

! Solution

We compute

$$\begin{vmatrix} -1 & 2 & -2 \\ 2 & -1 & 4 \\ 3 & 3 & 1 \end{vmatrix} = - \begin{vmatrix} -1 & 4 \\ 3 & 1 \end{vmatrix} - 2 \begin{vmatrix} 2 & 4 \\ 3 & 1 \end{vmatrix} - 2 \begin{vmatrix} 2 & -1 \\ 3 & 3 \end{vmatrix} \\ = -(-1 - 12) - 2(2 - 12) - 2(6 + 3) = 13 + 20 - 18 = 15,$$

which also is the volume of the parallelepiped.

Exercise 6.5. Compute the determinant of

$$A = \begin{pmatrix} 2 & -1 & 1 \\ 0 & 1 & 4 \\ 1 & -2 & -6 \end{pmatrix}.$$

in two different ways: By help of the steps in Gaussian elimination (those were written down in Exercise 4.1), and by expanding the determinant along the first column.

! Solution

In the steps of Gaussian elimination there is only one row interchange which contributes to changing the determinant, which therefore is -1 . If we expand the determinant along the first column we get

$$\begin{vmatrix} 2 & -1 & 1 \\ 0 & 1 & 4 \\ 1 & -2 & -6 \end{vmatrix} = 2(-6 + 8) - 0 + (-4 - 1) = 4 - 5 = -1.$$

Exercise 6.6. Prove that the determinant of a vandermonde matrix with generators $x_0, \dots, x_n$ is

$$\prod_{i>j} (x_i - x_j).$$

💡 Hint

Write down all row operations which change the determinant in Example 4.3.

Exercise 6.7. Prove Proposition 6.4.

! Solution

First apply $i - 1$ row interchanges so that row i occurs first. After this apply Equation 6.1. The sign becomes $(-1)^{i+j}$.

Exercise 6.8. Prove point 2. and 3. in Proposition 6.3.

💡 Hint

Use point 1. of Proposition 6.3, and Corollary 6.1.

! Solution

We first prove point 2. If the first row is multiplied with s it follows from Equation 6.1 that

$$\begin{aligned}\det(B) &= sa_{11} \det(A^{(11)}) + sa_{12} \det(A^{(12)}) + \cdots + sa_{1n} \det(A^{(1n)}) \\ &= s(a_{11} \det(A^{(11)}) + a_{12} \det(A^{(12)}) + \cdots + a_{1n} \det(A^{(1n)})) = s \det(A).\end{aligned}$$

If one of the other rows is multiplied with s , we combine three operations: First a row interchange with row 1, followed by multiplication of row 1 with s , and finally the same row interchange one more time. Proposition 6.3 then gives

$$\det(B) = -(s(-\det(A))) = s \det(A).$$

Let us now prove point 3.. Let C be the matrix obtained by swapping row 1 and j in B . The first row in C then has components $a_{jk} + sa_{ik}$, so that Equation 6.1 gives

$$\begin{aligned}-\det(B) &= \det(C) = \sum_{k=1}^n (-1)^{k+1} (a_{jk} + sa_{ik}) \det(C_{1k}) \\ &= \sum_{k=1}^n (-1)^{k+1} a_{jk} \det(C_{1k}) + s \sum_{k=1}^n (-1)^{k+1} a_{ik} \det(C_{1k}) \\ &= \sum_{k=1}^n (-1)^{k+1} a_{jk} \det(C_{1k}),\end{aligned}$$

where the other sum became zero since row 1 and i are equal (Corollary 6.1). The last expression is the determinant of the matrix obtained by swapping rows 1 and j in A , and this is $-\det(A)$. It follows that $\det(A) = \det(B)$.

Exercise 6.9. Suppose A is invertible. Show that $\det(A^{-1}) = 1/\det(A)$.

! Solution

Since $1 = \det(I_n) = \det(A^{-1}A) = \det(A^{-1})\det(A)$, it follows that $\det(A^{-1}) = 1/\det(A)$.

Exercise 6.10. Suppose U is orthogonal, i.e., U is real, and $U^T U = I_n$. Show that $\det(U)$ equals 1 or -1 .

Exercise 6.11. Prove Proposition 6.8.

! Solution

We have that

$$\det(A) = \det(A^T) = \sum_{j=1}^n (-1)^{j+i} a_{ji} \det(A^{(ji)}) = \sum_{i=1}^n (-1)^{i+j} a_{ij} \det(A^{(ij)}).$$

where we expanded the determinant of A^T along row i .

Exercise 6.12. Let A and B be square matrices, and set $D = \left(\begin{array}{c|c} A & C \\ \hline \mathbf{0} & B \end{array} \right)$, where $\mathbf{0}$ and C have dimensions so that D is square. Show that $\det(D) = \det(A)\det(B)$.

Exercise 6.13 (Cramer's rule). Suppose A is invertible, and that $\mathbf{x}$ solves the system $A\mathbf{x} = \mathbf{b}$. Let also $A_i(\mathbf{b})$ be the matrix A with column i replaced with $\mathbf{b}$.

(a) Show that

$$A \begin{pmatrix} 1 & 0 & \cdots & 0 & x_1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & x_2 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & x_{n-1} & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & x_n & 0 & \cdots & 0 & 1 \end{pmatrix} = A_i(\mathbf{b})$$

The vector $\mathbf{x}$ here comes in column i .

(b) Explain that the determinant of the large matrix above is x_i . Deduce from this that $x_i = \det(A_i(\mathbf{b}))/\det(A)$.

As a result of this, the solution to a linear system can be expressed in terms of determinants. This is also called *Cramer's rule*.

(c) Let $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ be a general invertible 2×2 -matrix. In a previous exercise it was proved that $A^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$. Prove this also by using Cramer's rule.

! Solution

The inverse matrix of A can be obtained by solving $A\mathbf{x} = \mathbf{e}_1$, and $A\mathbf{x} = \mathbf{e}_2$. Cramer's rule gives that $A\mathbf{x} = \mathbf{e}_1$ has solution

$$\left(\frac{\begin{vmatrix} 1 & b \\ 0 & d \end{vmatrix}}{\begin{vmatrix} a & b \\ c & d \end{vmatrix}}, \frac{\begin{vmatrix} a & 1 \\ c & 0 \end{vmatrix}}{\begin{vmatrix} a & b \\ c & d \end{vmatrix}} \right) = \frac{1}{ad-bc} \begin{pmatrix} d \\ -c \end{pmatrix},$$

and that $A\mathbf{x} = \mathbf{e}_2$ has solution

$$\left(\frac{\begin{vmatrix} 0 & b \\ 1 & d \end{vmatrix}}{\begin{vmatrix} a & b \\ c & d \end{vmatrix}}, \frac{\begin{vmatrix} a & 0 \\ c & 1 \end{vmatrix}}{\begin{vmatrix} a & b \\ c & d \end{vmatrix}} \right) = \frac{1}{ad-bc} \begin{pmatrix} -b \\ a \end{pmatrix}.$$

The inverse matrix is thus $\frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.

- (d) Sometimes we want to restrict ourselves to matrices with integer entries which also have an inverse with integer entries. Explain how Cramer's rule can be used to produce such matrices.

! Solution

If A has integer entries and $\mathbf{b} = \mathbf{e}_i$ for some i , we have that $\det(A_i\mathbf{b})$ also is an integer. if $\det(A) = \pm 1$ we will then have that A^{-1} has integer entries by Cramer's rule. Matrices with determinant 1 are easy to construct - we can take a diagonal matrix with ones on the diagonal. And by adding multiples of rows to other rows the determinant will be unchanged at 1.

Chapter 7

Eigenvalues and eigenvectors

In this chapter we will work with eigenvectors and eigenvalues, which we define as follows.

Definition 7.1 (Eigenvalues and eigenvectors). Let A be an $n \times n$ -matrix. λ is called an *eigenvalue* of A if there exists an $\mathbf{x} \neq \mathbf{0} \in \mathbb{R}^n$ which satisfies $A\mathbf{x} = \lambda\mathbf{x}$. $\mathbf{x}$ is then called an *eigenvector* corresponding to λ .

λ is an eigenvalue if $(A - \lambda I)\mathbf{x} = \mathbf{0}$ has a solution $\mathbf{x} \neq \mathbf{0}$, which is the same as $A - \lambda I$ not being invertible, which again is the same as that

$$\det(A - \lambda I) = 0.$$

This is called the *characteristic equation* of A . If we use the recursive definition of the determinant this gives a polynomial of degree at most n in λ , called the *characteristic polynomial* of A . eigenvalues can thus be computed by finding roots in polynomials. In principle one can find the eigenvalues by row-reducing $A - \lambda I$, but the problem is that the actual row operations will depend on the concrete value of λ . Therefore we will hold on to the recursive definition of the determinant for now.¹

When the eigenvalues have been found, an eigenvector corresponding to a given λ can be found by solving the system $(A - \lambda I)\mathbf{x} = \mathbf{0}$. This can be achieved by row-reducing $A - \lambda I$. Eigenvectors are not unique - if we multiply it with a scalar we get another eigenvector. Sometimes one needs the general expression for the eigenvectors corresponding to a given eigenvalue. Other times one only needs one concrete eigenvector. It is then desirable to pick one on a form which is simplest possible - typically with integer components as small as possible. Some examples will clarify this.

¹In practice the computer finds eigenvalues with help of another algorithm.

Example 7.1. Let us first consider the linear mapping T which reflects about the line $y = 0$ in $\mathbb{R}^2$. We have that $T(\mathbf{e}_1) = \mathbf{e}_1$ and $T(\mathbf{e}_2) = -\mathbf{e}_2$ (we see already here that $\mathbf{e}_1$ and $\mathbf{e}_2$ are eigenvectors), so that the matrix if T is $A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. The characteristic equation is $(1 - \lambda)(-1 - \lambda) = 0$, so that the eigenvalues are $\lambda_1 = -1$, $\lambda_2 = 1$.

Eigenvector for $\lambda_1 = -1$:

$$A + I = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

The first row here says that an eigenvector must satisfy $x_1 = 0$. The general eigenvector is therefore $(0, x_2)$ with x_2 a free variable. In particular we get the simple eigenvector $(0, 1)$.

Eigenvector for $\lambda_2 = 1$:

$$A - I = \begin{pmatrix} 0 & 0 \\ 0 & -2 \end{pmatrix} \sim \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

The first row here says that an eigenvector must satisfy $x_2 = 0$. In particular we get the eigenvector $(1, 0)$. ♣

Example 7.2. Let us consider a bigger example. We set

$$A = \begin{pmatrix} 1 & 2 & 4 \\ -3 & 6 & 7 \\ 2 & -2 & -1 \end{pmatrix}.$$

We first get

$$\begin{aligned} \det(A - \lambda I) &= \begin{vmatrix} 1 - \lambda & 2 & 4 \\ -3 & 6 - \lambda & 7 \\ 2 & -2 & -1 - \lambda \end{vmatrix} \\ &= (1 - \lambda) \begin{vmatrix} 6 - \lambda & 7 \\ -2 & -1 - \lambda \end{vmatrix} - 2 \begin{vmatrix} -3 & 7 \\ 2 & -1 - \lambda \end{vmatrix} + 4 \begin{vmatrix} -3 & 6 - \lambda \\ 2 & -2 \end{vmatrix} \\ &= (1 - \lambda)((6 - \lambda)(-1 - \lambda) + 14) - 2(3 + 3\lambda - 14) + 4(6 - 12 + 2\lambda) \\ &= (1 - \lambda)(\lambda^2 - 5\lambda + 8) - 2(3\lambda - 11) + 4(2\lambda - 6) \\ &= -\lambda^3 + 5\lambda^2 - 8\lambda + \lambda^2 - 5\lambda + 8 - 6\lambda + 22 + 8\lambda - 24 \\ &= -\lambda^3 + 6\lambda^2 - 11\lambda + 6 \end{aligned}$$

Most of us do not remember the formula for the solution to a third degree equation, so it is not obvious how to proceed. One can find the roots of a polynomial with help of the command `roots`. This takes the coefficients of the polynomial as input:

```
np.roots(np.array([-1,6,-11,6]))
```

```
array([3., 2., 1.])
```

Alternatively one can find the roots by trial and error. Then we probably first will find the root $\lambda_1 = 1$. The two other roots can be found by first applying polynomial division

$$\begin{array}{r}
 (-\lambda^3 + 6\lambda^2 - 11\lambda + 6) : (\lambda - 1) = -\lambda^2 + 5\lambda - 6 \\
 \underline{-(-\lambda^3 + \lambda^2)} \\
 5\lambda^2 - 11\lambda \\
 \underline{-(5\lambda^2 - 5\lambda)} \\
 -6\lambda + 6 \\
 \underline{-(-6\lambda + 6)} \\
 0,
 \end{array}$$

so that

$$-\lambda^3 + 24\lambda^2 - 176\lambda + 384 = (\lambda - 1)(-\lambda^2 + 5\lambda - 6),$$

and then using the formula for the solution to the second degree equation to find the roots of $-\lambda^2 + 5\lambda - 6 = 0$. This gives

$$\lambda = \frac{-5 \pm \sqrt{25 - 24}}{-2} = \frac{-5 \pm 1}{-2} = \frac{5 \pm 1}{2}$$

so that we also get the roots $\lambda_2 = 2$ and $\lambda_3 = 3$. Let us turn to find the eigenvectors

- For $\lambda_1 = 1$ we get

$$\begin{aligned}
 A - \lambda_1 I &= \begin{pmatrix} 0 & 2 & 4 \\ -3 & 5 & 7 \\ 2 & -2 & -2 \end{pmatrix} \sim \begin{pmatrix} 1 & -1 & -1 \\ -3 & 5 & 7 \\ 0 & 1 & 2 \end{pmatrix} \sim \begin{pmatrix} 1 & -1 & -1 \\ 0 & 2 & 4 \\ 0 & 1 & 2 \end{pmatrix} \\
 &\sim \begin{pmatrix} 1 & -1 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{pmatrix}
 \end{aligned}$$

The first row says that $x_1 + x_3 = 0$, so that $x_1 = -x_3$. The second row says that $x_2 + 2x_3 = 0$, so that $x_2 = -2x_3$. The general eigenvector is therefore

$$(x_1, x_2, x_3) = (-x_3, -2x_3, x_3) = x_3(-1, -2, 1)$$

Setting $x_3 = 1$ we get the special eigenvector $\mathbf{v}_1 = (-1, -2, 1)$.

- For $\lambda_2 = 2$ we get

$$\begin{aligned}
 A - \lambda_2 I &= \begin{pmatrix} -1 & 2 & 4 \\ -3 & 4 & 7 \\ 2 & -2 & -3 \end{pmatrix} \sim \begin{pmatrix} 1 & -2 & -4 \\ 0 & -2 & -5 \\ 0 & 2 & 5 \end{pmatrix} \sim \begin{pmatrix} 1 & -2 & -4 \\ 0 & 1 & 5/2 \\ 0 & 0 & 0 \end{pmatrix} \\
 &\sim \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 5/2 \\ 0 & 0 & 0 \end{pmatrix}
 \end{aligned}$$

Here we get the special eigenvector $\mathbf{v}_2 = (-2, -5, 2)$.

- For $\lambda_3 = 3$ we get

$$\begin{aligned} A - \lambda_3 I &= \begin{pmatrix} -2 & 2 & 4 \\ -3 & 3 & 7 \\ 2 & -2 & -4 \end{pmatrix} \sim \begin{pmatrix} 1 & -1 & -2 \\ -3 & 3 & 7 \\ 0 & 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & -1 & -2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \\ &\sim \begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \end{aligned}$$

so that we get the special eigenvector $\mathbf{v}_3 = (1, 1, 0)$.

Let us verify this computation:

$$\begin{aligned} \bullet & \begin{pmatrix} 1 & 2 & 4 \\ -3 & 6 & 7 \\ 2 & -2 & -1 \end{pmatrix} \begin{pmatrix} -1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 - 4 + 4 \\ 3 - 12 + 7 \\ -2 + 4 - 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -2 \\ 1 \end{pmatrix} \\ \bullet & \begin{pmatrix} 1 & 2 & 4 \\ -3 & 6 & 7 \\ 2 & -2 & -1 \end{pmatrix} \begin{pmatrix} -2 \\ -5 \\ 2 \end{pmatrix} = \begin{pmatrix} -2 - 10 + 8 \\ 6 - 30 + 14 \\ -4 + 10 - 2 \end{pmatrix} = \begin{pmatrix} -4 \\ -10 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} -2 \\ -5 \\ 2 \end{pmatrix} \\ \bullet & \begin{pmatrix} 1 & 2 & 4 \\ -3 & 6 & 7 \\ 2 & -2 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 + 2 \\ -3 + 6 \\ 2 - 2 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \\ 0 \end{pmatrix} = 3 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}. \end{aligned}$$



The example above is constructed so that it can be solved without too ugly numbers. In practice eigenvalues are neither integers nor fractions. In this case the computation will suffer from roundoff errors. We will mostly not do calculations as above by hand - as it is too time consuming, and error-prone. Instead we can use the command `np.linalg.eig` which takes a matrix as input, and returns two matrices:

- one matrix with the eigenvectors of the input matrix as columns,
- one diagonal matrix where the eigenvalues of the input matrix are on the diagonal (matching eigenvectors are found in the corresponding column of the first matrix).

If we test `np.linalg.eig` on the example above we get

```
A = np.array([[1,2,4],[-3,6,7],[2,-2,-1]])
D, P = np.linalg.eig(A)
D, P
```

```
(array([1., 2., 3.]),
 array([[ 4.08248290e-01, -3.48155312e-01,  7.07106781e-01],
        [ 8.16496581e-01, -8.70388280e-01,  7.07106781e-01],
        [-4.08248290e-01,  3.48155312e-01, -1.24634588e-15]]))
```

We observe several things here. Firstly, decimal numbers are returned, not integers. This is the case since a numerical algorithm is used to find the roots of the characteristic polynomial. Secondly, consider the following:

`P.T@P`

```
array([[ 1.          , -0.99493668,  0.8660254 ],
       [-0.99493668,  1.          , -0.86164044],
       [ 0.8660254 , -0.86164044,  1.          ]])
```

Since $P^T P$ returns all possible scalar products between the columns in P , this shows two things:

- the columns in P are scaled so that they have length 1,
- the eigenvectors are not orthogonal (since there are not only zeros outside the diagonal).

If you compute the reduced echelon form of P you will get the identity matrix, so that the eigenvectors at least are linearly independent. We will soon prove that this is the case also more generally for eigenvectors corresponding to distinct eigenvalues.

We would like to get exact expressions for eigenvalues and eigenvectors as well. This can be achieved by converting to symbolic variables:

```
A = sym.Matrix([[1,2,4],[-3,6,7],[2,-2,-1]])
A.eigenvals(), A.eigenvects()
```

```
({3: 1, 2: 1, 1: 1},
 [(1, 1, [Matrix([
  [-1],
  [-2],
  [ 1]])]),
 (2,
  1,
  [Matrix([
  [-1],
  [-5/2],
  [ 1]])]),
 (3,
  1,
  [Matrix([
  [1],
  [1],
  [0]])])])
```

We see that the components in the eigenvectors are fractions, and that there are no decimal numbers. We see the exact same eigenvalues we found, and if we multiply P with 2 we get the same eigenvectors.

Let us say that we have a large matrix A and a vector $\mathbf{x}_0$, and that we would like to compute $A^{100}\mathbf{x}_0$. A direct computation would be very demanding, but imagine that we already have computed the eigenvectors $\mathbf{v}_i$ and the eigenvalues λ_i of A , and that the eigenvectors provide a basis for $\mathbb{R}^n$. Then we can find values c_i so that $\mathbf{x}_0 = \sum_{i=1}^n c_i \mathbf{v}_i$. From this we can compute

$$A^{100}\mathbf{x}_0 = A^{100} \left(\sum_{i=1}^n c_i \mathbf{v}_i \right) = \sum_{i=1}^n c_i A^{100} \mathbf{v}_i = \sum_{i=1}^n c_i \lambda_i^{100} \mathbf{v}_i,$$

where we used that $A\mathbf{v}_i = \lambda_i \mathbf{v}_i$, and then $A^2 \mathbf{v}_i = \lambda_i A \mathbf{v}_i = \lambda_i^2 \mathbf{v}_i$, and so on. If the eigenvalues here are ≤ 1 we can obtain a limit value. Let us consider an example.

Example 7.3. We study the spread of a disease in a population, and let x_n be the number of sick people after n days, and y_n the number of healthy people after n days. We assume that there are 1000 people all in all, so that $x_n + y_n = 1000$ for all n . We assume also:

1. If a person is sick one day, it is a 40% chance that he will be sick tomorrow as well.
2. If a person is healthy one day, it is a 20% chance that he will be sick tomorrow.

Let us first find a transition matrix A so that $\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = A \begin{pmatrix} x_n \\ y_n \end{pmatrix}$.

- The number of sick people one day is 40% of those sick the day before, plus 20% of those healthy the day before. This gives that $x_{n+1} = 0.4x_n + 0.2y_n$.
- The remaining $0.6x_n + 0.8y_n$ is the number of healthy people one day.

This gives that $A = \begin{pmatrix} 0.4 & 0.2 \\ 0.6 & 0.8 \end{pmatrix}$. Let us find the eigenvalues and eigenvectors of A :

$$\det(A - \lambda I) = \begin{vmatrix} 0.4 - \lambda & 0.2 \\ 0.6 & 0.8 - \lambda \end{vmatrix} = (0.4 - \lambda)(0.8 - \lambda) - 0.12 = \lambda^2 - 1.2\lambda + 0.2.$$

The roots of this polynomial are $\lambda = \frac{1.2 \pm \sqrt{1.44 - 0.8}}{2} = 0.6 \pm 0.4$, so that the eigenvalues are $\lambda_1 = 0.2$ and $\lambda_2 = 1$.

- For $\lambda_1 = 0.2$ we get

$$A - 0.2I = \begin{pmatrix} 0.2 & 0.2 \\ 0.6 & 0.6 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

We get a general eigenvector by setting $x_1 = -x_2$. If we set $x_2 = 1$ we get the particular eigenvector $\mathbf{v}_1 = (-1, 1)$.

- For $\lambda_2 = 1$ we get

$$A - I = \begin{pmatrix} -0.6 & 0.2 \\ 0.6 & -0.2 \end{pmatrix} \sim \begin{pmatrix} 1 & -1/3 \\ 0 & 0 \end{pmatrix}$$

We get a general eigenvector by setting $x_1 = x_2/3$. If we set $x_2 = 3$ we get the particular eigenvector $\mathbf{v}_2 = (1, 3)$.

Let now $\mathbf{x}_0 = \begin{pmatrix} 0 \\ 1000 \end{pmatrix}$ be the initial state where everyone is healthy. . Let us find the limit $\lim_{n \rightarrow \infty} \mathbf{x}_n$. We now express $\mathbf{x}_0$ in terms of $\mathbf{v}_1$ and $\mathbf{v}_2$:

$$(\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{x}_0) = \begin{pmatrix} -1 & 1 & 0 \\ 1 & 3 & 1000 \end{pmatrix} \sim \begin{pmatrix} 1 & 3 & 1000 \\ 0 & 4 & 1000 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 250 \\ 0 & 1 & 250 \end{pmatrix},$$

so that $\mathbf{x}_0 = 250\mathbf{v}_1 + 250\mathbf{v}_2$. Finally we get

$$\mathbf{x}_n = A^n \mathbf{x}_0 = 250 \cdot 0.2^n \mathbf{v}_1 + 250 \cdot 1^n \mathbf{v}_2 = 250 \begin{pmatrix} 1 - 0.2^n \\ 3 + 0.2^n \end{pmatrix}.$$

It follows that $\lim_{n \rightarrow \infty} \mathbf{x}_n = (250, 750)$. This is an *equilibrium*, meaning that

$$A \begin{pmatrix} 250 \\ 750 \end{pmatrix} = \begin{pmatrix} 0.4 & 0.2 \\ 0.6 & 0.8 \end{pmatrix} \begin{pmatrix} 250 \\ 750 \end{pmatrix} = \begin{pmatrix} 100 + 150 \\ 150 + 600 \end{pmatrix} = \begin{pmatrix} 250 \\ 750 \end{pmatrix}.$$



The matrices in the examples above had distinct eigenvalues, and it turned out that $\mathbb{R}^n$ had a basis of eigenvectors. This turns out to hold more generally:

Proposition 7.1. *Let $\mathbf{v}_1, \dots, \mathbf{v}_k$ be eigenvectors corresponding to distinct eigenvalues. Then $\mathbf{v}_1, \dots, \mathbf{v}_k$ are linearly independent.*

Proof. Suppose for contradiction that $\mathbf{v}_1, \dots, \mathbf{v}_k$ are linearly dependent. Then there is a linear independence relation

$$c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 + \dots + c_k \mathbf{v}_k = \mathbf{0}.$$

But then there is such a relation with as few non-zeros as possible, let us say $r \geq 2$ non-zeros. By rearranging the vectors we can assume that the r first vectors have coefficients different from 0, i.e.,

$$c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 + \dots + c_r \mathbf{v}_r = \mathbf{0},$$

where all $c_1, \dots, c_r$ are $\neq 0$. Multiplying this relation with $-\lambda_1$ we obtain

$$-c_1 \lambda_1 \mathbf{v}_1 - c_2 \lambda_1 \mathbf{v}_2 - \dots - c_r \lambda_1 \mathbf{v}_r = \mathbf{0}, \quad (7.1)$$

We also have that

$$A(c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 + \dots + c_r \mathbf{v}_r) = c_1 \lambda_1 \mathbf{v}_1 + c_2 \lambda_2 \mathbf{v}_2 + \dots + c_r \lambda_r \mathbf{v}_r = \mathbf{0} \quad (7.2)$$

If we add Equation 7.1 and Equation 7.2 we get

$$c_2(\lambda_2 - \lambda_1) \mathbf{v}_2 + c_3(\lambda_3 - \lambda_1) \mathbf{v}_3 + \dots + c_r(\lambda_r - \lambda_1) \mathbf{v}_r = \mathbf{0}.$$

This gives us an even shorter linear dependence relation, which is a contradiction. $\mathbf{v}_1, \dots, \mathbf{v}_k$ must thus be linearly independent. $\square$

If a matrix does not have distinct eigenvalues two things can happen:

1. $\mathbb{R}^n$ does not have a basis of eigenvectors for A .
2. $\mathbb{R}^n$ has a basis of eigenvectors for A .

Let us show that both of these can occur.

Example 7.4. Consider $A = \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix}$. Then

$$\det(A - \lambda I) = \begin{vmatrix} 1 - \lambda & -2 \\ 2 & -3 - \lambda \end{vmatrix} = (1 - \lambda)(-3 - \lambda) + 4 = \lambda^2 + 2\lambda + 1 = (\lambda + 1)^2.$$

$\lambda = -1$ is therefore the only eigenvalue. The eigenvectors for this can be found by row reduction:

$$A + I = \begin{pmatrix} 2 & -2 \\ 2 & -2 \end{pmatrix} \sim \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix},$$

so that $\mathbf{v} = (1, 1)$ is a particular eigenvector. This does not span $\mathbb{R}^2$, so that 1. above is not the case. ♣

Example 7.5. Let us set $A = I_2$. The characteristic polynomial is now $(1 - \lambda)^2$, so that $\lambda = 1$ is the only eigenvalue. Since $A - I = 0$, both x_1 and x_2 are free variables, so that all vectors are eigenvectors. In particular $\{\mathbf{e}_1, \mathbf{e}_2\}$ is a basis of eigenvectors, so that 2. above is the case. ♣

A matrix can also have complex eigenvalues and eigenvectors:

Example 7.6. Consider $A = \begin{pmatrix} 2 & -8 \\ 2 & 2 \end{pmatrix}$. Then

$$\det(A - \lambda I) = \begin{vmatrix} 2 - \lambda & -8 \\ 2 & 2 - \lambda \end{vmatrix} = (2 - \lambda)^2 + 16,$$

which is 0 when $2 - \lambda = \pm 4i$, i.e., when $\lambda = 2 \pm 4i$.

Let us find an eigenvector for $\lambda_1 = 2 + 4i$. We have that

$$A - (2 + 4i)I = \begin{pmatrix} -4i & -8 \\ 2 & -4i \end{pmatrix} \sim \begin{pmatrix} 1 & -2i \\ 0 & 0 \end{pmatrix}.$$

This gives the particular eigenvector $\mathbf{v}_1 = (2i, 1)$.

Let us find an eigenvector for $\lambda_1 = 2 - 4i$. We have that

$$A - (2 - 4i)I = \begin{pmatrix} 4i & -8 \\ 2 & 4i \end{pmatrix} \sim \begin{pmatrix} 1 & 2i \\ 0 & 0 \end{pmatrix}.$$

This gives the particular eigenvector $\mathbf{v}_1 = (-2i, 1)$. ♣

In the example above we saw that the eigenvalues and eigenvectors were conjugates of one-another. This is actually the case for all real matrices. To see this, suppose λ is an eigenvalue with corresponding eigenvector $\mathbf{x}$. Then

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

This shows that $\bar{\lambda}$ also is an eigenvalue, with corresponding eigenvector $\bar{\mathbf{x}}$.

Example 7.7. Let us find the eigenvalues of the rotation matrix $A = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$:

$$\det(A - \lambda I) = \begin{vmatrix} \cos \theta - \lambda & -\sin \theta \\ \sin \theta & \cos \theta - \lambda \end{vmatrix} = (\cos \theta - \lambda)^2 + \sin^2 \theta,$$

which is 0 when $\lambda = \cos \theta \pm i \sin \theta$. We then find the eigenvector corresponding to $\lambda = \cos \theta + i \sin \theta$ using row reduction:

$$A - (\cos \theta + i \sin \theta)I = \begin{pmatrix} -i \sin \theta & -\sin \theta \\ \sin \theta & -i \sin \theta \end{pmatrix} \sim \begin{pmatrix} 1 & -i \\ 0 & 0 \end{pmatrix},$$

so that the corresponding eigenvector is $(i, 1)$. Since rotation matrices are real it follows that also $\lambda = (-i, 1)$ is an eigenvector corresponding to $\cos \theta - i \sin \theta$.

♣

7.1 Diagonalisation

Suppose that the $n \times n$ -matrix A is real, and has eigenvectors $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k$ with corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_k$. Since $A\mathbf{p}_i = \lambda_i\mathbf{p}_i$ for $i = 1, \dots, k$ we have that

$$\begin{aligned} A(\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_k) &= (A\mathbf{p}_1 \quad A\mathbf{p}_2 \quad \cdots \quad A\mathbf{p}_k) \\ &= (\lambda_1\mathbf{p}_1 \quad \lambda_2\mathbf{p}_2 \quad \cdots \quad \lambda_k\mathbf{p}_k) \\ &= (\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_k) \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_k \end{pmatrix}. \end{aligned}$$

We can also write this as $AP = PD$, where $P = (\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_k)$, and D is the $k \times k$ diagonal matrix with $\lambda_1, \lambda_2, \dots, \lambda_k$ on the diagonal. If A has n linearly independent eigenvectors, the matrix P is an invertible $n \times n$ -matrix. If we multiply with P^{-1} to the right on both sides we get that

$$A = PDP^{-1}.$$

Matrices that can be factored like this (with D diagonal) are called *diagonalisable*. From this it is clear that matrices that have n linearly independent

eigenvectors are diagonalisable. The other way, if $A = PDP^{-1}$, we can write $AP = PD$, which says that $\{\mathbf{p}_i\}_{i=1}^n$ are linearly independent eigenvectors for A , with $\{\lambda_i\}_{i=1}^n$ as corresponding eigenvalues. We have therefore proved the following:

Proposition 7.2. *A matrix A is diagonalisable if and only if $\mathbb{C}^n$ has a basis consisting of eigenvectors for A .*

While A is supposed to be real, in a diagonalisation we must allow for P and D to be complex - since real matrices can have complex eigenvalues, and complex eigenvalues give rise to complex eigenvectors. This is why we wrote $\mathbb{C}^n$ above, and not $\mathbb{R}^n$.

Why is it desirable for a matrix to be diagonalisable? First of all, powers of a diagonalisable matrix are easily computed, since $A^k = PD^kP^{-1}$ (it is less work to compute D^k than A^k for large matrices). More generally it is easy to show that

$$p(A) = P \begin{pmatrix} p(\lambda_1) & 0 & \cdots & 0 \\ 0 & p(\lambda_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p(\lambda_n) \end{pmatrix} P^{-1}$$

for every polynomial p .

We also say that the matrices A and D are *similar* if $A = PDP^{-1}$ for some invertible matrix P . This definition covers square matrices in general - regardless of whether they are diagonal or not. As an example, from Proposition 5.11 it follows that the matrices $[T]_{\mathcal{A}}$ and $[T]_{\mathcal{B}}$ of linear transformations relative to any bases are similar. Since $A\mathbf{p}_i = \lambda_i\mathbf{p}_i$ it follows that the matrix of $T(\mathbf{x}) = A\mathbf{x}$ relative to a basis of eigenvectors is the diagonal matrix with the diagonal elements being the eigenvalues.

Example 7.8. In Example 7.4 we saw that the matrix $A = \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix}$ did not have a basis of eigenvectors. It is therefore not diagonalisable. ♣

Example 7.9. In Example 7.6 we found that the matrix $A = \begin{pmatrix} 2 & -8 \\ 2 & 2 \end{pmatrix}$ had eigenvalues $\lambda_1 = 2 + 4i$ and $\lambda_2 = 2 - 4i$ with corresponding eigenvectors $\mathbf{p}_1 = (2i, 1)$ and $\mathbf{p}_2 = (-2i, 1)$. We can thus set $P = \begin{pmatrix} 2i & -2i \\ 1 & 1 \end{pmatrix}$. We compute

$$P^{-1} = \frac{1}{2i + 2i} \begin{pmatrix} 1 & 2i \\ -1 & 2i \end{pmatrix} = \frac{1}{4} \begin{pmatrix} -i & 2 \\ i & 2 \end{pmatrix}.$$

A diagonalisation of A is thus

$$A = PDP^{-1} = \frac{1}{4} \begin{pmatrix} 2i & -2i \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 2 + 4i & 0 \\ 0 & 2 - 4i \end{pmatrix} \begin{pmatrix} -i & 2 \\ i & 2 \end{pmatrix}.$$

If we write $2+4i$ on polar form $re^{i\theta}$ then we get $(2+4i)^k = r^k e^{ik\theta}$ and $(2-4i)^k = r^k e^{-ik\theta}$. This gives

$$A^k = PD^k P^{-1} = \frac{r^k}{4} \begin{pmatrix} 2i & -2i \\ 1 & 1 \end{pmatrix} \begin{pmatrix} e^{ik\theta} & 0 \\ 0 & e^{-ik\theta} \end{pmatrix} \begin{pmatrix} -i & 2 \\ i & 2 \end{pmatrix}.$$



Exercise 6.13 explains how we can obtain an invertible matrix P so that both P and P^{-1} have integer entries only. If we let D be a diagonal matrix with integer entries, $A = PDP^{-1}$ will also have integer entries, with eigenvectors and -values also being integers. Exercise 6.13 thus gives a recipe for constructing eigenvector examples with nice numbers in them. Example 7.2 was constructed following this recipe.

Exercises

Exercise 7.1. Find eigenvalues and eigenvectors for the matrix

(a) $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{pmatrix}$

! Solution

We have

$$\det(A - \lambda I) = (1 - \lambda) \begin{vmatrix} 3 - \lambda & 1 \\ 1 & 3 - \lambda \end{vmatrix} = (1 - \lambda)((3 - \lambda)^2 - 1)$$

where we expanded the determinant along the first row. Since $(1 - \lambda)$ was a factor here we need not apply polynomial division. We also need not multiply out $(3 - \lambda)^2 - 1$, since this can be zero only when $\lambda = 3 \pm 1$. The eigenvalues are therefore $\lambda_1 = 1$, $\lambda_2 = 2$, $\lambda_3 = 4$.

Eigenvector for $\lambda_1 = 1$:

$$A - I_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix} \sim \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

so that $(1, 0, 0)$ is a particular eigenvector.

Eigenvector for $\lambda_2 = 2$:

$$A - 2I_3 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

so that $(0, -1, 1)$ is a particular eigenvector.

Eigenvector for $\lambda_3 = 4$:

$$A - 4I_3 = \begin{pmatrix} -3 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix},$$

so that $(0, 1, 1)$ is a particular eigenvector.

```
A = sym.Matrix([[1,0,0],[0,3,1],[0,1,3]])
A.eigenvals(), A.eigenvects()

({1: 1, 4: 1, 2: 1},
 [(1, 1, [Matrix([
 [1],
 [0],
 [0]])]),
 (2,
 1,
 [Matrix([
 [ 0],
 [-1],
 [ 1]])]),
 (4,
 1,
 [Matrix([
 [0],
 [1],
 [1]])])])
```

(b) $A = \begin{pmatrix} 0 & -4 \\ 1 & 0 \end{pmatrix}$

! Solution

We have $\det(A - \lambda I) = \lambda^2 + 4$, and this is 0 for $\lambda = \pm 2i$.

Eigenvector for $\lambda_1 = 2i$:

$$A - 2iI_2 = \begin{pmatrix} -2i & -4 \\ 1 & -2i \end{pmatrix} \sim \begin{pmatrix} 1 & -2i \\ 0 & 0 \end{pmatrix},$$

so that $(2i, 1)$ is a particular eigenvector. But then $(-2i, 1)$ is a particular eigenvector for $\lambda_2 = -2i$.

```
A = sym.Matrix([[0,-4],[1,0]])
A.eigenvals(), A.eigenvects()

({-2*I: 1, 2*I: 1},
 [(-2*I, 1, [Matrix([
```

```

    [-2*I],
    [ 1]]]]),
(2*I,
1,
[Matrix([
    [2*I],
    [ 1]]]]))

```

Exercise 7.2. Define $A = \begin{pmatrix} 0.7 & 0.6 \\ 0.3 & 0.4 \end{pmatrix}$.

(a) Find the eigenvectors and eigenvalues of A .

! Solution

The characteristic polynomial is

$$(0.7 - \lambda)(0.4 - \lambda) - 0.3 \cdot 0.6 = \lambda^2 - 1.1\lambda + 0.1,$$

which has roots $\lambda = \frac{1.1 \pm \sqrt{1.21 - 0.4}}{2} = \frac{1.1 \pm 0.9}{2}$, so that $\lambda_1 = 0.1$, $\lambda_2 = 1$.
Eigenvalue for $\lambda_1 = 0.1$:

$$A - 0.1I_2 = \begin{pmatrix} 0.6 & 0.6 \\ 0.3 & 0.3 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix},$$

so that $(-x_2, x_2)$ is a general eigenvector. A particular eigenvector is $\mathbf{v}_1 = (-1, 1)$.

Eigenvalue for $\lambda_2 = 1$:

$$A - I_2 = \begin{pmatrix} -0.3 & 0.6 \\ 0.3 & -0.6 \end{pmatrix} \sim \begin{pmatrix} 1 & -2 \\ 0 & 0 \end{pmatrix},$$

so that $(2x_2, x_2)$ is a general eigenvector. A particular eigenvector is $\mathbf{v}_2 = (2, 1)$.

```

A = sym.Matrix([[7,6],[3,4]])/10
A.eigenvals(), A.eigenvects()

```

```

({1: 1, 1/10: 1},
 [(1/10, 1, [Matrix([
    [-1],
    [ 1]])]),
 (1,
 1,
 [Matrix([
    [2],
    [1]]]]))

```

- (b) Write the vector $\mathbf{x}_0 = (1, 5)$ as a sum of eigenvectors, and compute the limit $\lim_{n \rightarrow \infty} A^n \mathbf{x}_0$.

! Solution

We row reduce

$$\begin{pmatrix} -1 & 1 & 1 \\ 1 & 1 & 5 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 5 \\ 0 & 2 & 6 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \end{pmatrix},$$

so that $\mathbf{x}_0 = 2\mathbf{v}_1 + 3\mathbf{v}_2$. This gives

$$\lim_{n \rightarrow \infty} A^n \mathbf{x}_0 = \lim_{n \rightarrow \infty} (2A^n \mathbf{v}_1 + 3A^n \mathbf{v}_2) = \lim_{n \rightarrow \infty} (2 \cdot 0.1^n \mathbf{v}_1 + 3\mathbf{v}_2) = 3\mathbf{v}_2 = (3, 3).$$

Exercise 7.3. Find eigenvalues and eigenvectors for $A = \begin{pmatrix} 6 & 2 \\ 0 & 6 \end{pmatrix}$. Is there a basis for $\mathbb{R}^2$ consisting of eigenvectors for A ?

! Solution

We have that $\det(A - \lambda I_2) = (6 - \lambda)^2$, so that $\lambda = 6$ is the only eigenvalue. We find the eigenvectors:

$$A - 6I_2 = \begin{pmatrix} 0 & 2 \\ 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},$$

so that $(1, 0)$ is a particular eigenvector. This vector spans a space of dimension one, so that there is no basis for $\mathbb{R}^2$ consisting of eigenvectors for A .

```
A = sym.Matrix([[6,2],[0,6]])
A.eigenvals(), A.eigenvects()
({6: 2},
 [(6, 2, [Matrix([
 [1],
 [0]])])])])
```

Exercise 7.4. Let A be the matrix $A = \begin{pmatrix} 0.7 & 0.8 \\ 0.3 & 0.2 \end{pmatrix}$.

- (a) Find the eigenvalues and eigenvectors of A .

! Solution

We have that

$$\det(\lambda I - A) = (\lambda - 0.7)(\lambda - 0.2) - 0.24 = \lambda^2 - 0.9\lambda - 0.1.$$

This is 0 when $\lambda = \frac{0.9 \pm 1.1}{2}$. The eigenvalues are thus $\lambda_1 = -0.1$ and $\lambda_2 = 1$.

Eigenvector for $\lambda_1 = -0.1$:

$$-0.1I - A = \begin{pmatrix} -0.8 & -0.8 \\ -0.3 & -0.3 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.$$

$\mathbf{v}_1 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$ is a particular eigenvector.

Eigenvector for $\lambda_2 = 1$:

$$I - A = \begin{pmatrix} 0.3 & -0.8 \\ -0.3 & 0.8 \end{pmatrix} \sim \begin{pmatrix} 1 & -8/3 \\ 0 & 0 \end{pmatrix}$$

$\mathbf{v}_2 = \begin{pmatrix} 8 \\ 3 \end{pmatrix}$ is a particular eigenvector.

```
A = sym.Matrix([[7,8],[3,2]])/10
A.eigenvals(), A.eigenvects()
```

```
{1: 1, -1/10: 1},
[(-1/10, 1, [Matrix([
  [-1],
  [ 1]])]),
(1,
1,
[Matrix([
  [8/3],
  [ 1]])])])
```

- (b) Write down a diagonalisation of the matrix A . Does A have an orthonormal basis of eigenvectors?

! Solution

A matrix A is diagonalisable if it can be written on the form $A = PDP^{-1}$, where D is diagonal and P is invertible. P then has the eigenvectors as columns, and D has the eigenvalues on the diagonal.

We know that eigenvectors corresponding to different eigenvalues are linearly independent. Since the two eigenvalues here are different, we have two linearly independent eigenvectors. P can then be obtained by insert-

ing the two eigenvectors as columns, and the corresponding eigenvalues on the diagonal in D . We therefore have $A = PDP^{-1}$ with $P = \begin{pmatrix} -1 & 8 \\ 1 & 3 \end{pmatrix}$ and $D = \begin{pmatrix} -0.1 & 0 \\ 0 & 1 \end{pmatrix}$, and

$$P^{-1} = \frac{1}{\det(P)} \begin{pmatrix} 3 & -8 \\ -1 & -1 \end{pmatrix} = \frac{1}{11} \begin{pmatrix} -3 & 8 \\ 1 & 1 \end{pmatrix}.$$

It is easily checked that the eigenvectors $(-1, 1)$ and $(8, 3)$ are not orthogonal, so that we do not have an orthonormal basis of eigenvectors. Here one could also argue as follows:

If A has an orthonormal basis of eigenvectors, it can be written on the form $A = PDP^T$, where P has eigenvectors as columns. But then A symmetric since

$$A^T = (PDP^T)^T = PD^T P^T = PDP^T = A.$$

The matrix here was not symmetric, and can then not have an orthonormal basis of eigenvectors.

Exercise 7.5. Suppose both $\mathbf{x}_1$ and $\mathbf{x}_2$ are eigenvectors for A , for the same eigenvalue λ . Show that $\mathbf{x}_1 + \mathbf{x}_2$ also is an eigenvector for A with eigenvalue λ . We also know that $c\mathbf{x}_1$ is an eigenvector for A with eigenvalue λ . From this it follows that all eigenvectors for A corresponding to λ is a subspace of $\mathbb{R}^n$. This subspace is also called an *eigenspace*.

! Solution

We have that $A(\mathbf{x}_1 + \mathbf{x}_2) = A\mathbf{x}_1 + A\mathbf{x}_2 = \lambda\mathbf{x}_1 + \lambda\mathbf{x}_2 = \lambda(\mathbf{x}_1 + \mathbf{x}_2)$. This proves the claim.

Exercise 7.6. Let A be a square matrix. Show that A and A^T have the same eigenvalues.

! Solution

We have that $\det(A - \lambda I_n) = \det(A - \lambda I_n)^T = \det(A^T - \lambda I_n)$. This shows that A and A^T have the same characteristic polynomial, so that they also have the same eigenvalues.

Exercise 7.7 (Nilpotent matrices have 0 as the only eigenvalue). Recall from Exercise 4.7 that a square matrix A is called nilpotent if $A^k = 0$ for some k . Show that all eigenvalues in a nilpotent matrix are 0.

! Solution

Assume that $A\mathbf{x} = \lambda\mathbf{x}$. Applying A several times gives $\mathbf{0} = A^k\mathbf{x} = \lambda^k\mathbf{x}$. It follows that $\lambda^k = 0$, so that $\lambda = 0$ also.

Exercise 7.8 (Nilpotent matrices are not diagonalisable). Prove that a nonzero nilpotent matrix is not diagonalisable.

! Solution

Assume for contradiction that there exists a nilpotent matrix A that is nonzero and diagonalisable. Then $A = PDP^{-1}$ for some diagonal matrix D and invertible matrix P . We have that $D^k = P^{-1}A^kP$. Since $A^k = 0$ for some k this implies that $D^k = 0$. Since D is diagonal this implies that $D = 0$, which is a contradiction.

Exercise 7.9. Show that the rank of a diagonalisable matrix equals the number of nonzero eigenvalues (counted with multiplicity). Show also that A^k has the same rank as A for all k when A is diagonalisable.

! Solution

Let $A = PDP^{-1}$ be diagonalisable. Multiplication with P^{-1} to the right does not change the column space, so that $\text{rank}(DP^{-1}) = \text{rank}(D)$. Similarly, multiplication with P to the left does not change the dimension of the column space, so that

$$\text{rank}(A) = \text{rank}(PDP^{-1}) = \text{rank}(DP^{-1}) = \text{rank}(D),$$

and $\text{rank}(D)$ clearly equals the number of nonzero eigenvalues counted with multiplicity.

The second statement follows from that $A^k = PD^kP^{-1}$, and from that D and D^k have the same positions of zeros on the diagonal.

Exercise 7.10. Let A be an invertible matrix with eigenvalues λ_i . Show that A^{-1} has the eigenvalue $1/\lambda_i$. Explain also that A and A^{-1} have the same eigenvectors.

! Solution

We have that

$$\det(A) \det(A^{-1} - \lambda I_n) = \det(I_n - \lambda A) = (-\lambda)^n \det(A - \frac{1}{\lambda} I_n).$$

From this it follows that λ is an eigenvalue of A^{-1} if and only if $\frac{1}{\lambda}$ is

an eigenvalue of A . If we apply A^{-1} to each side in $A\mathbf{x} = \lambda\mathbf{x}$ we get that $\mathbf{x} = \lambda A^{-1}\mathbf{x}$, so that $A^{-1}\mathbf{x} = \frac{1}{\lambda}\mathbf{x}$, so that A and A^{-1} have the same eigenvectors.

Exercise 7.11. Suppose that $\mathbf{v}$ is an eigenvector for both A and B . Show that $\mathbf{v}$ is an eigenvector for $A + B$. What is the corresponding eigenvalue?

! Solution

Suppose $A\mathbf{v} = \lambda_1\mathbf{v}$ and $B\mathbf{v} = \lambda_2\mathbf{v}$. Then

$$(A + B)\mathbf{v} = A\mathbf{v} + B\mathbf{v} = \lambda_1\mathbf{v} + \lambda_2\mathbf{v} = (\lambda_1 + \lambda_2)\mathbf{v}.$$

$\mathbf{v}$ is therefore also an eigenvector for $A + B$, with $\lambda_1 + \lambda_2$ as corresponding eigenvalue.

Exercise 7.12. Suppose that A is diagonalisable, so that we can write $A = PDP^{-1}$. Since

$$PD = (\lambda_1\mathbf{p}_1 \quad \lambda_2\mathbf{p}_2 \quad \cdots \quad \lambda_n\mathbf{p}_n)$$

the outer form for matrix multiplication gives

$$PDP^{-1} = \sum_{k=1}^n \lambda_k P_{:k} (P^{-1})_{k:}. \quad (7.3)$$

This splits A into a sum of simpler matrices $Q_k = P_{:k} (P^{-1})_{k:}$. We can thus write $A = \sum_{k=1}^n \lambda_k Q_k$. In this exercise we will look at the matrices Q_k more closely.

(a) Show that $Q_k^2 = Q_k$. Matrices satisfying this are also called *idempotent*.

! Solution

We have that

$$Q_k^2 = P_{:k} (P^{-1})_{k:} P_{:k} (P^{-1})_{k:} = P_{:k} (P^{-1}P)_{kk} (P^{-1})_{k:} = P_{:k} (P^{-1})_{k:} = Q_k$$

We here used that $(P^{-1})_{k:} P_{:k} = (P^{-1}P)_{kk} = 1$.

(b) Show that $Q_r Q_s = 0$ when $r \neq s$.

! Solution

We have that $Q_r Q_s = P_{:r} (P^{-1})_{r:} P_{:s} (P^{-1})_{s:} = P_{:r} (P^{-1}P)_{rs} (P^{-1})_{s:} = 0$ when $r \neq s$, since $P^{-1}P = I$ has zeros outside the diagonal.

- (c) Show that $p(A)$ can be written as a linear combination of $\{Q_k\}_{k=1}^n$, when p is a polynomial.
- (d) Show that Q_k can be written as a polynomial in A .

! Solution

It is clear that

$$p(A) = P \begin{pmatrix} p(\lambda_1) & 0 & \cdots & 0 \\ 0 & p(\lambda_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p(\lambda_n) \end{pmatrix} P^{-1}$$

for all polynomials p . Choose now a polynomial p which is 1 in λ_i , and 0 in the other eigenvalues. Then we get that $p(A) = \lambda_i \sum Q_k$.

Exercise 7.13. We shall consider matrices where the column sums all are 1. A general such 2×2 -matrix can be written $A = \begin{pmatrix} d & e \\ 1-d & 1-e \end{pmatrix}$.

- (a) Show that the eigenvalues of A are $d - e$ and 1.

! Solution

We compute

$$\begin{aligned} \det(A - \lambda I) &= \begin{vmatrix} d - \lambda & e \\ 1 - d & 1 - e - \lambda \end{vmatrix} = (d - \lambda)(1 - e - \lambda) - e(1 - d) \\ &= \lambda^2 - (d - e + 1)\lambda + d - e. \end{aligned}$$

This is 0 for

$$\begin{aligned} \lambda &= \frac{d - e + 1 \pm \sqrt{(d - e + 1)^2 - 4(d - e)}}{2} = \frac{d - e + 1 \pm \sqrt{(d - e - 1)^2}}{2} \\ &= \frac{d - e + 1 \pm (d - e - 1)}{2}, \end{aligned}$$

which gives $\lambda_1 = d - e$, $\lambda_2 = 1$.

- (b) Find the eigenvectors of A .

! Solution

We have

$$A - (d - e)I_2 = \begin{pmatrix} e & e \\ 1 - d & 1 - d \end{pmatrix} \sim \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

We therefore have $x_1 = -x_2$, so that $(-1, 1)$ is a particular eigenvector.

$$A - I_2 = \begin{pmatrix} d - 1 & e \\ 1 - d & -e \end{pmatrix} \sim \begin{pmatrix} 1 & \frac{e}{d-1} \end{pmatrix}$$

A general eigenvector is $x_2 \begin{pmatrix} -\frac{e}{d-1} \\ 1 \end{pmatrix}$. If we set $x_2 = 1 - d$ we get the particular eigenvector $\begin{pmatrix} e \\ 1 - d \end{pmatrix}$.

Chapter 8

Diagonalisation and Systems of differential equations.

With a *linear first order system of differential equations with constant coefficients* we mean a system of the form

$$\begin{aligned}x'_1(t) &= a_{11}x_1(t) + a_{12}x_2(t) + \cdots + a_{1n}x_n(t) \\x'_2(t) &= a_{21}x_1(t) + a_{22}x_2(t) + \cdots + a_{2n}x_n(t) \\&\vdots \\x'_n(t) &= a_{n1}x_1(t) + a_{n2}x_2(t) + \cdots + a_{nn}x_n(t),\end{aligned}$$

where $x_1(t), x_2(t), \dots, x_n(t)$ are unknown differentiable functions. If we write

$$\mathbf{x}(t) = (x_1(t), x_2(t), \dots, x_n(t)) \quad \mathbf{x}'(t) = (x'_1(t), x'_2(t), \dots, x'_n(t)),$$

and define the matrix A to have components a_{ij} , then this can be written more compactly as

$$\mathbf{x}'(t) = A\mathbf{x}(t).$$

Diagonalisation is very useful when it comes to solving such systems:

Proposition 8.1. *Suppose A is diagonalisable, with $A = PDP^{-1}$. Then we have that $\mathbf{x}(t)$ solves the system $\mathbf{x}'(t) = A\mathbf{x}(t)$ if and only if $\mathbf{y}(t) = P^{-1}\mathbf{x}(t)$ solves the system $\mathbf{y}'(t) = D\mathbf{y}(t)$.*

Proof. Suppose first that $\mathbf{x}(t)$ solves $\mathbf{x}'(t) = A\mathbf{x}(t)$. With $\mathbf{y}(t) = P^{-1}\mathbf{x}(t)$ we get that

$$\mathbf{y}'(t) = (P^{-1}\mathbf{x})' = P^{-1}\mathbf{x}' = P^{-1}A\mathbf{x}(t) = P^{-1}AP\mathbf{y}(t) = D\mathbf{y}(t),$$

so that $\mathbf{y}(t)$ løser $\mathbf{y}'(t) = D\mathbf{y}(t)$. Suppose then that $\mathbf{y}(t)$ løser $\mathbf{y}'(t) = D\mathbf{y}(t)$. We then get

$$\mathbf{x}'(t) = (P\mathbf{y}(t))' = P\mathbf{y}'(t) = PD\mathbf{y}(t) = PDP^{-1}\mathbf{x}(t) = A\mathbf{x}(t),$$

so that $\mathbf{x}(t)$ solves $\mathbf{x}'(t) = A\mathbf{x}(t)$. □

The point is that the system $\mathbf{y}'(t) = D\mathbf{y}(t)$ can be split into

$$y_1'(t) = \lambda_1 y_1(t) \quad y_2'(t) = \lambda_2 y_2(t) \quad \dots \quad y_n'(t) = \lambda_n y_n(t)$$

so that the component functions can be found independently of one-another as $y_i(t) = C_i e^{\lambda_i t}$, where C_i is a constant that can be complex, and

$$\mathbf{y}(t) = (C_1 e^{\lambda_1 t}, C_2 e^{\lambda_2 t}, \dots, C_n e^{\lambda_n t}).$$

It is interesting to interpret solutions of different systems when A is 2×2 , since we then can plot x_1 against x_2 in the plane. The general solution when $n = 2$ is

$$\mathbf{x}(t) = P\mathbf{y}(t) = (\mathbf{p}_1 \quad \mathbf{p}_2) \begin{pmatrix} C_1 e^{\lambda_1 t} \\ C_2 e^{\lambda_2 t} \end{pmatrix} = C_1 e^{\lambda_1 t} \mathbf{p}_1 + C_2 e^{\lambda_2 t} \mathbf{p}_2,$$

Let us first consider the case where A has real eigenvalues. Recall that A may not be diagonalisable when A has equal eigenvalues. We have the following result.

Proposition 8.2. *Suppose that A has real eigenvalues and that $\mathbf{x}(t_0)$ is known, and real. Then there exist unique real constants C_1 and C_2 so that*

$$\mathbf{x}(t) = C_1 e^{\lambda_1 t} \mathbf{p}_1 + C_2 e^{\lambda_2 t} \mathbf{p}_2. \quad (8.1)$$

The other way it is clear that Equation 8.1 is real when C_1 and C_2 are real.

Proof. The system $\mathbf{x}(t_0) = (\mathbf{p}_1 \quad \mathbf{p}_2) \begin{pmatrix} C_1 e^{\lambda_1 t_0} \\ C_2 e^{\lambda_2 t_0} \end{pmatrix}$ is in this case real. Since P is invertible it follows that also the solution $\begin{pmatrix} C_1 e^{\lambda_1 t_0} \\ C_2 e^{\lambda_2 t_0} \end{pmatrix}$ is unique and real. But then C_1 and C_2 are also unique and real. □

Example 8.1 (reelle egenverdier). Set $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. A has eigenvalues $\lambda_1 = -1$ and $\lambda_2 = 1$, with corresponding eigenvectors $\mathbf{p}_1 = (-1, 1)$ and $\mathbf{p}_2 = (1, 1)$. The general solution Equation 8.1 is thus

$$C_1 e^{-t} \begin{pmatrix} -1 \\ 1 \end{pmatrix} + C_2 e^t \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -C_1 e^{-t} + C_2 e^t \\ C_1 e^{-t} + C_2 e^t \end{pmatrix},$$

and it is clear that these have $f(x) = x$ and $f(x) = -x$ as asymptotes when $t \rightarrow \infty$ and $t \rightarrow -\infty$. We can plot more solutions $C_1 e^{\lambda_1 t} \mathbf{p}_1 + C_2 e^{\lambda_2 t} \mathbf{p}_2$ in

the same systems by varying C_1 and C_2 . The solutions we get by choosing $C_i \in \{1.5, 2, 2.5\}$ (9 solutions) are shown in Figure 8.1.

```
t = np.linspace(-2,2,100)
for C1 in [1.5,2,2.5]:
    for C2 in [1.5,2,2.5]:
        plt.plot( -C1*np.exp(-t) + C2*np.exp(t) , \
                  C1*np.exp(-t) + C2*np.exp(t) )
```

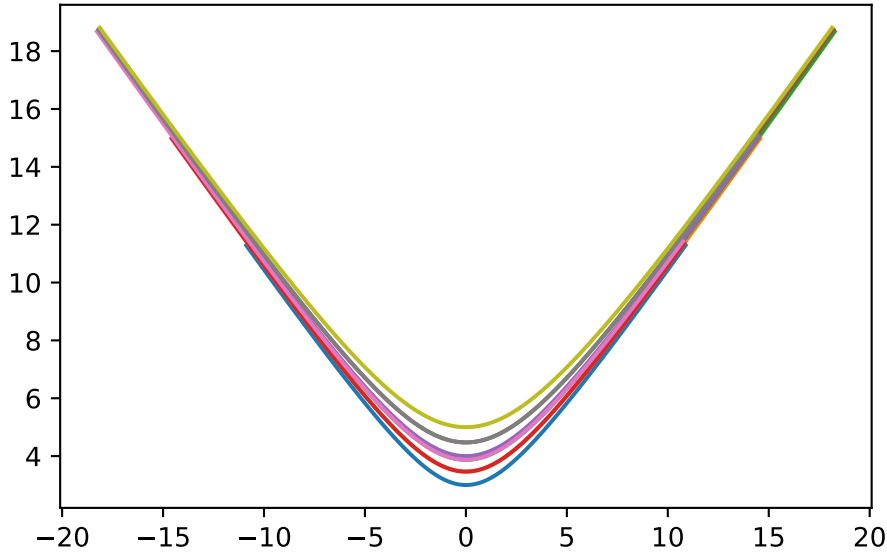


Figure 8.1: Plot of nine solutions to the system with matrix $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.



Let us now consider the case where A has complex eigenvalues. We then know that A is diagonalisable, and that eigenvalues and eigenvectors come in conjugate pairs.

Proposition 8.3. *Suppose that A has complex eigenvalues λ and $\bar{\lambda}$ (with corresponding eigenvectors $\mathbf{p}$ and $\bar{\mathbf{p}}$). Suppose also that $\mathbf{x}(t_0)$ is known, and real. Then there exists a constant C (which can be complex) so that*

$$\mathbf{x}(t) = 2\Re(Ce^{\lambda t}\mathbf{p}). \quad (8.2)$$

The other way it is clear that Equation 8.2 is real for all values of C .

Proof. We have that

$$\mathbf{x}(t) = P\mathbf{y}(t) = (\mathbf{p} \quad \bar{\mathbf{p}}) \begin{pmatrix} C_1 e^{\lambda t} \\ C_2 e^{\bar{\lambda} t} \end{pmatrix}$$

for constants C_1 and C_2 . If we set $t = t_0$ we get

$$\begin{aligned} \mathbf{x}(t_0) &= (\mathbf{p} \quad \bar{\mathbf{p}}) \begin{pmatrix} C_1 e^{\lambda t_0} \\ C_2 e^{\bar{\lambda} t_0} \end{pmatrix} \\ \mathbf{x}(t_0) &= \overline{\mathbf{x}(t_0)} = (\bar{\mathbf{p}} \quad \mathbf{p}) \begin{pmatrix} \overline{C_1} e^{\bar{\lambda} t_0} \\ \overline{C_2} e^{\lambda t_0} \end{pmatrix} = (\mathbf{p} \quad \bar{\mathbf{p}}) \begin{pmatrix} \overline{C_2} e^{\lambda t_0} \\ \overline{C_1} e^{\bar{\lambda} t_0} \end{pmatrix}, \end{aligned} \quad (8.3)$$

where we changed the order of the first two components. Since P is invertible we get by comparing the two lines in Equation 8.3 that $C_2 = \overline{C_1}$, and the solution becomes

$$\mathbf{x}(t) = (\mathbf{p} \quad \bar{\mathbf{p}}) \begin{pmatrix} C_1 e^{\lambda t} \\ \overline{C_1} e^{\bar{\lambda} t} \end{pmatrix} = C_1 e^{\lambda t} \mathbf{p} + \overline{C_1} e^{\bar{\lambda} t} \bar{\mathbf{p}} = 2\Re(C_1 e^{\lambda t} \mathbf{p}),$$

so that we can set $C = C_1$. □

The expression $2\Re(C_1 e^{\lambda t} \mathbf{p})$ involves complex numbers, even though the answer is real. But it is easy to rewrite this to a form not involving complex arithmetic. Let $\mathbf{c}$ and $\mathbf{d}$ be the real and imaginary parts of the complex vector $C\mathbf{p}$, and let $\lambda = a + ib$. Then $C\mathbf{p} = \mathbf{c} + i\mathbf{d}$ and $e^{\lambda t} = e^{at}(\cos(bt) + i\sin(bt))$, and we get

$$\Re(C e^{\lambda t} \mathbf{p}) = \Re(e^{at}(\mathbf{c} + i\mathbf{d})(\cos(bt) + i\sin(bt))) = e^{at}(\cos(bt)\mathbf{c} - \sin(bt)\mathbf{d}),$$

so that

$$\mathbf{x}(t) = 2e^{at}(\cos(bt)\mathbf{c} - \sin(bt)\mathbf{d}). \quad (8.4)$$

From this form it is clear that

1. For $a > 0$: $x(t) \rightarrow \infty$ when $t \rightarrow \infty$ (spiral away from the origin),
2. For $a < 0$: $x(t) \rightarrow 0$ when $t \rightarrow \infty$ (spiral towards the origin),
3. For $a = 0$: $x(t)$ is periodic.

Since $\cos(bt)$ and $\sin(bt)$ are periodic with period $2\pi/b$, a plot over $[0, 2\pi/b]$ should show one arm of a spiral in 1. and 2., or one whole period in 3..

The following function takes a real matrix A with complex eigenvalues, together with an initial value $x(t_0) = x_0$, as parameters. The code computes the eigenvalues and eigenvectors of A , solves the system

$$(\mathbf{p} \quad \bar{\mathbf{p}}) \begin{pmatrix} C e^{\lambda t_0} \\ \overline{C} e^{\bar{\lambda} t_0} \end{pmatrix} = \mathbf{x}_0,$$

finds C , computes $C\mathbf{p} = \mathbf{c} + i\mathbf{d}$, and then the solution Equation 8.4, which is returned:

```

def findcomps(A,t_0,x_0):
    D,P=np.linalg.eig(A)
    C = np.linalg.solve(P, x_0)
    C = C[0]/np.exp(D[0]*t_0)
    c = np.real(C*P[:,0])
    d = np.imag(C*P[:,0])
    a = np.real(D[0])
    b = np.imag(D[0])

    t=np.linspace(t_0,t_0+4*np.pi/b,100).reshape((1,-1))
    res = ( c@(np.cos(b*t)*np.exp(a*t)) - d@(np.sin(b*t)*np.exp(a*t)) )*2
    return t, res[[0],:], res[[1],:], a, c[0], d[0]

```

The interval $[t_0, t_0 + 4\pi/b]$ includes two arms of the spiral.

Example 8.2 (Complex eigenvalues). Consider the three matrices

$$A_1 = \begin{pmatrix} 0.1 & 1 \\ -1 & 0.1 \end{pmatrix} \quad A_2 = \begin{pmatrix} -0.1 & 1 \\ -1 & -0.1 \end{pmatrix} \quad A_3 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

These have eigenvalues $0.1 \pm i$, $-0.1 \pm i$, and $\pm i$, respectively. We also have that $a > 0$, $a < 0$, and $a = 0$, respectively, so that the solutions to the systems will demonstrate the three possibilities above. The first solution will in particular show a spiral moving away from the origin, the second a spiral towards the origin. The solutions are plotted in Figure 8.2, Figure 8.3, Figure 8.4. We have used the initial value $\mathbf{x}(2) = (2, 3)$, so that all solutions start in the point $(2, 3)$.

```

t_0 = 2
x_0 = np.array([2,3])
A_1 = np.array([0.1,1],[-1,0.1])
t, x, y, a, c_1, d_1 = findcomps(A_1,t_0,x_0)
plt.plot(x.T,y.T)

```

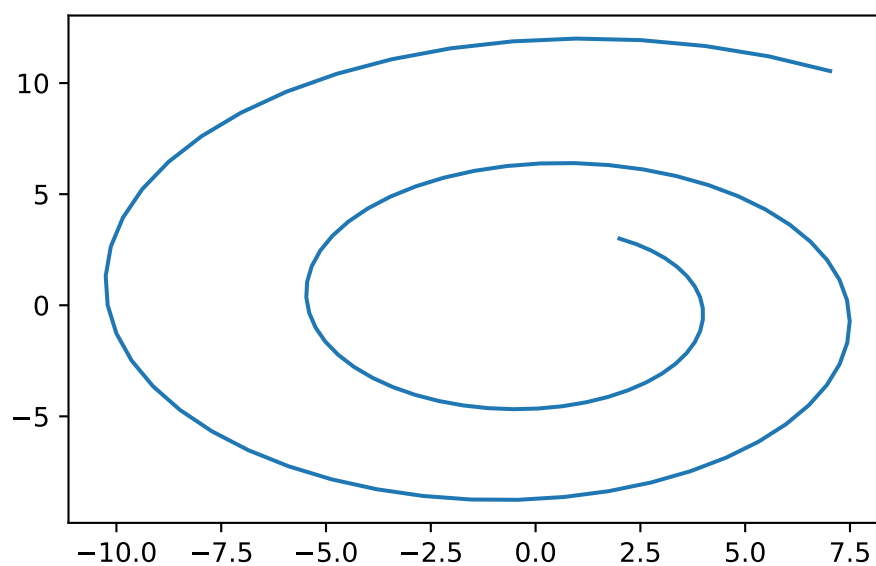


Figure 8.2

```
A_2 = np.array([[1,1],[-1,-0.1]])
t, x, y, a, c_1, d_1 = findcomps(A_2,t_0,x_0)
plt.plot(x.T,y.T)
```

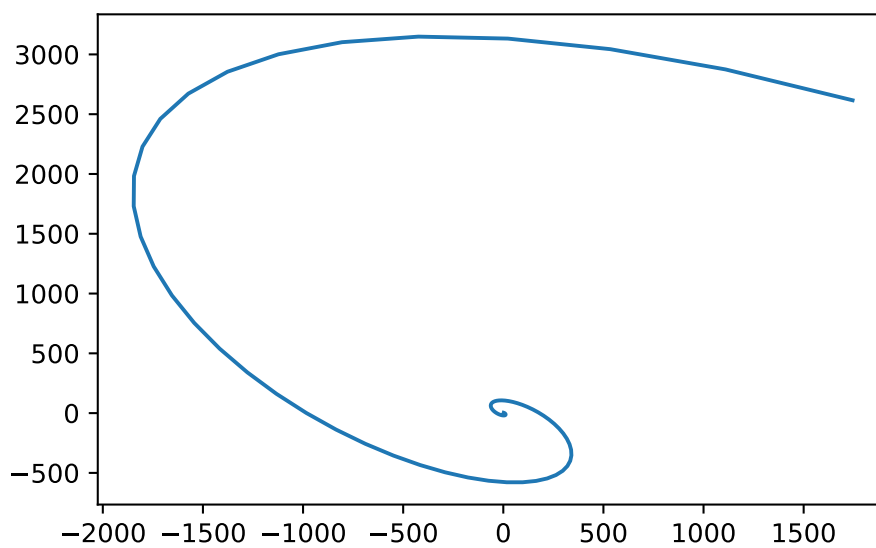


Figure 8.3

```
A_3 = np.array([[0,1],[-1,0]])
t, x, y, a, c_1, d_1 = findcomps(A_3,t_0,x_0)
plt.plot(x.T,y.T)
```

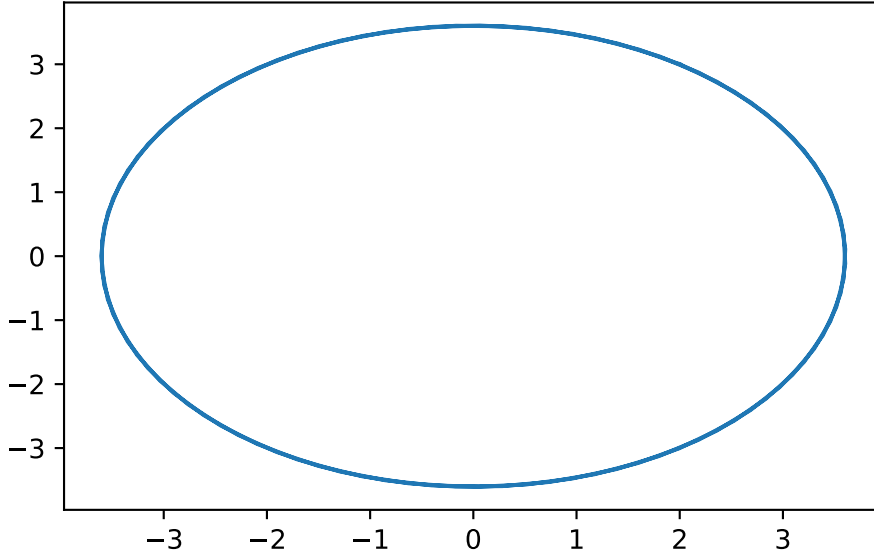


Figure 8.4

One can also plot t against $x_i(t)$. Doing this one can visualise all components in the same plot, also when $n > 2$. In Figure 8.5 we plot only the first component $x_1(t)$, over the same time interval, and with the same initial condition as above. Since $c_1 \cos(bt) - d_1 \sin(bt)$ has amplitude $\sqrt{c_1^2 + d_1^2}$ (Exercise 8.1) we have upper and lower bounds

$$-2\sqrt{c_1^2 + d_1^2}e^{at} \leq x_1(t) \leq 2\sqrt{c_1^2 + d_1^2}e^{at} \quad (8.5)$$

These are also included in the plot.

```
t, x, y, a, c_1, d_1 = findcomps(A_1,t_0,x_0)
plt.plot(t.T,x.T)
plt.plot(t.T,2*np.sqrt(c_1**2 + d_1**2)*np.exp(a*t.T))
plt.plot(t.T,-2*np.sqrt(c_1**2 + d_1**2)*np.exp(a*t.T))
```

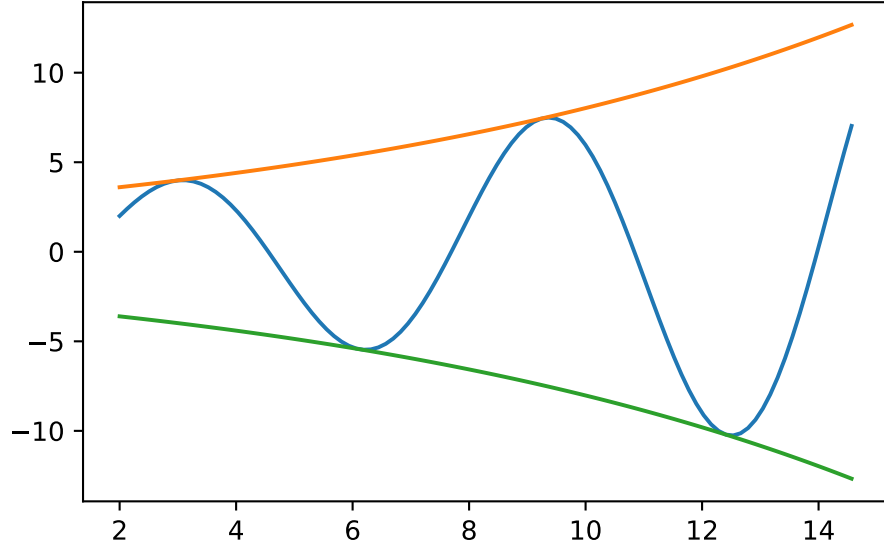


Figure 8.5: Plot of $x_1(t)$ for the system with matrix A_1 . The upper and lower bounds from Equation 8.5 are also shown.



It turns out that one can write down solutions for systems with $n > 2$, based on what we did above. In the general case we have a mix of real and complex eigenvalues and eigenvectors:

$$\lambda_1, \overline{\lambda_1}, \dots, \lambda_m, \overline{\lambda_m}, \lambda_{2m+1}, \dots, \lambda_n$$

$$\mathbf{p}_1, \overline{\mathbf{p}_1}, \dots, \mathbf{p}_m, \overline{\mathbf{p}_m}, \mathbf{p}_{2m+1}, \dots, \mathbf{p}_n,$$

where the real ones were listed at the end. Exercise 8.2 explains how one can solve the case with two complex and one real eigenvalue.

8.1 Higher order linear differential equations

A type of system related to the systems given in Equation 8.1 are *higher order* linear differential equations with constant coefficients. These are on the form

$$y^{(n)}(t) + a_1 y^{(n-1)}(t) + a_2 y^{(n-2)}(t) + \dots + a_n y(t) = 0, \quad (8.6)$$

where $y(t)$ is the unknown function. We shall limit ourselves to second order equations, and write these on the form

$$y'' + py' + qy = 0$$

where p and q are constants. By setting $x_0 = y$, $x_1 = y'$ we can write this as the first order linear system

$$\begin{aligned} x_0' &= x_1 \\ x_1' &= -qx_0 - px_1 \end{aligned}$$

which has coefficient matrix $A = \begin{pmatrix} 0 & 1 \\ -q & -p \end{pmatrix}$. We have that

$$\det(A - \lambda I) = \lambda(\lambda + p) + q = \lambda^2 + p\lambda + q.$$

Assume that the eigenvalues λ_1 and λ_2 are different (this is the same as that $p^2 \neq 4q$). Then A is diagonalisable, and we have that

$$A - \lambda I_n = \begin{pmatrix} -\lambda & 1 \\ -q & -\lambda - p \end{pmatrix} \sim \begin{pmatrix} -\lambda & 1 \\ 0 & 0 \end{pmatrix},$$

so that $(1, \lambda_1)$ and $(1, \lambda_2)$ are two particular eigenvectors. We can therefore set

$$P = \begin{pmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{pmatrix}.$$

Let us first assume real eigenvalues. We have shown that the general solution is

$$\begin{pmatrix} y(t) \\ y'(t) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{pmatrix} \begin{pmatrix} C_1 e^{\lambda_1 t} \\ C_2 e^{\lambda_2 t} \end{pmatrix}, \quad (8.7)$$

where one can find C_1 and C_2 with help of initial values $y(t_0)$, $y'(t_0)$.

Example 8.3 (Distinct real eigenvalues). Consider the equation

$$y'' - 5y' + 6y = 0,$$

with initial values $y(0) = 1$, $y'(0) = 0$. The eigenvalues of A are found by solving the characteristic equation $\lambda^2 - 5\lambda + 6 = 0$, which gives $\lambda_1 = 2$, $\lambda_2 = 3$. If we insert $t = 0$ in Equation 8.7 we get

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix},$$

which has the solution $C_1 = 3$, $C_2 = -2$. The solution is thus

$$y(t) = C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t} = 3e^{2t} - 2e^{3t}.$$



The case with complex eigenvalues is a bit more complicated. We shall first prove the following.

Proposition 8.4.

$$y(t) = 2e^{at} (\cos(bt)c_1 - \sin(bt)d_1), \quad (8.8)$$

where c_1 and d_1 are arbitrarily chosen real numbers (recall that $\lambda = a + ib$).

Proof. Equation 8.4 said that the general solution is

$$\begin{pmatrix} y(t) \\ y'(t) \end{pmatrix} = 2e^{at} (\cos(bt)\mathbf{c} - \sin(bt)\mathbf{d}), \quad (8.9)$$

where $\mathbf{c} = \Re(C\mathbf{p})$, $\mathbf{d} = \Im(C\mathbf{p})$, and where C is an arbitrarily chosen complex number.

Let now c_1, d_1 be given numbers, and define $C = c_1 + id_1$. Then

$$C\mathbf{p} = \begin{pmatrix} c_1 + id_1 \\ \cdot \end{pmatrix} \quad \mathbf{c} = \Re(C\mathbf{p}) = \begin{pmatrix} c_1 \\ \cdot \end{pmatrix} \quad \mathbf{d} = \Im(C\mathbf{p}) = \begin{pmatrix} d_1 \\ \cdot \end{pmatrix},$$

where $\cdot$ indicate a concrete number. Evaluating the first row in Equation 8.9 with this we get that Equation 8.8 is a solution for all choices of c_1 and d_1 , and because of Equation 8.9 all solutions must be on this form. $\square$

Example 8.4 (Complex eigenvalues). Consider the equation

$$y'' + 4y' + 5y = 0.$$

The characteristic equation now becomes $\lambda^2 + 4\lambda + 5 = 0$, which has the solution $\frac{-4 \pm \sqrt{16-20}}{2} = -2 \pm i$. The general solution thus becomes

$$y(t) = 2e^{-2t} (c_1 \cos t - d_1 \sin t).$$



So far we have avoided the case where $\lambda_1 = \lambda_2$. In that case we have that $\lambda_1 = \lambda_2 = -p/2$, and that $p^2 = 4q$. We then get

$$A - \lambda I = \begin{pmatrix} -\lambda & 1 \\ -q & -\lambda - p \end{pmatrix} = \begin{pmatrix} p/2 & 1 \\ -p^2/4 & -p/2 \end{pmatrix} \sim \begin{pmatrix} p/2 & 1 \\ 0 & 0 \end{pmatrix}.$$

It follows that the eigenspace has dimension one, so that A is not diagonalisable. We can therefore not use what we have done to find the solutions, but it is an exercise to show by direct substitution that $y(t) = e^{\lambda t}$ is a solution, and also that $y(t) = te^{\lambda t}$ is a solution. It follows that all functions on the form $y(t) = Ce^{\lambda t} + Dte^{\lambda t}$ also are solutions, and one can show that there is a unique such solution satisfying the initial conditions for $y(t_0)$ and $y'(t_0)$. We will not go into how we “guessed” the solution $te^{\lambda t}$, and we will also not prove that all functions can be expressed in terms of the functions $e^{\lambda t}$ and $te^{\lambda t}$.

Example 8.5 (Double root). Consider the equation $y'' + 6y' + 9y = 0$ with initial conditions $y(0) = 4$, $y'(0) = -13$. The characteristic equation becomes $\lambda^2 + 6\lambda + 9 = 0$, which has the solution $\lambda_1 = \lambda_2 = -3$. The general solution thus becomes $y(t) = Ce^{-3t} + Dte^{-3t}$, and we also have that $y'(t) = -3Ce^{-3t} + D(1 - 3t)e^{-3t}$. The initial conditions give the equations

$$\begin{aligned} C &= 4 \\ -3C + D &= -13, \end{aligned}$$

which have the solution $C = 4$, $D = -1$. The solution is thus $y(t) = 4e^{-3t} - te^{-3t}$.



Exercises

Exercise 8.1. Let b , c , and d be real numbers. Show that $|c \cos(bt) - d \sin(bt)| \leq \sqrt{c^2 + d^2}$ for all t . For which t do you have equality here?

! Solution

Consider the vector $\left(\frac{c}{\sqrt{c^2+d^2}}, \frac{d}{\sqrt{c^2+d^2}}\right)$. It is easy to verify that this has length one. Then there is an angle θ so that

$$\left(\frac{c}{\sqrt{c^2+d^2}}, \frac{d}{\sqrt{c^2+d^2}}\right) = (\cos \theta, \sin \theta).$$

But then $c = \sqrt{c^2 + d^2} \cos \theta$ and $d = \sqrt{c^2 + d^2} \sin \theta$, so that

$$\begin{aligned} |c \cos(bt) - d \sin(bt)| &= |\sqrt{c^2 + d^2} \cos \theta \cos(bt) - \sqrt{c^2 + d^2} \sin \theta \sin(bt)| \\ &= \sqrt{c^2 + d^2} |\cos \theta \cos(bt) - \sin \theta \sin(bt)| \\ &= \sqrt{c^2 + d^2} |\cos \theta \cos(bt + \theta)| \leq \sqrt{c^2 + d^2}, \end{aligned}$$

since $0 \leq |\cos \theta| \leq 1$ for all θ . Here we also used the formula for the cosine of the sum of two angles.

Exercise 8.2. Consider the system

$$\begin{aligned} x_1'(t) &= x_1(t) + x_2(t) \\ x_2'(t) &= -x_1(t) + x_2(t) \\ x_3'(t) &= x_3(t) \end{aligned}$$

(a) Find the general solution of the system.

! Solution

We here have $A = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. We get that

$$A - \lambda I = \begin{pmatrix} 1-\lambda & 1 & 0 \\ -1 & 1-\lambda & 0 \\ 0 & 0 & 1-\lambda \end{pmatrix}.$$

The determinant of this is $(1-\lambda)((1-\lambda)^2 + 1)$, so that the eigenvalues are $1 \pm i$, and 1. The eigenvectors are

$$\mathbf{p}_1 = \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix} \quad \mathbf{p}_2 = \begin{pmatrix} 1 \\ -i \\ 0 \end{pmatrix} \quad \mathbf{p}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

so that the general solution is

$$C e^{(1+i)t} \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix} + D e^{(1-i)t} \begin{pmatrix} 1 \\ -i \\ 0 \end{pmatrix} + E e^t \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

- (b) Find the solution of the system, given the initial condition $\mathbf{x}(0) = (1, 1, 1)$. What happens with the solution when $t \rightarrow -\infty$, and when $t \rightarrow \infty$?

! Solution

The initial condition gives the equation

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ i & -i & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} C \\ D \\ E \end{pmatrix},$$

which has the solution $C = (1-i)/2$, $D = (1+i)/2 = \overline{C}$, $E = 1$. This

gives

$$\begin{aligned}
 & Ce^{(1+i)t} \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix} + De^{(1-i)t} \begin{pmatrix} 1 \\ -i \\ 0 \end{pmatrix} + Ee^t \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
 &= \frac{1-i}{2} e^{(1+i)t} \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix} + \frac{\overline{1-i}}{2} e^{(1-i)t} \overline{\begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix}} + e^t \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
 &= 2\Re \left(\frac{1-i}{2} e^{(1+i)t} \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix} \right) + e^t \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
 &= e^t \Re \left((\cos t + i \sin t) \left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + i \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \right) \right) + e^t \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
 &= e^t \left(\begin{pmatrix} \cos t \\ \cos t \end{pmatrix} + \begin{pmatrix} \sin t \\ -\sin t \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right) \\
 &= e^t \begin{pmatrix} \cos t + \sin t \\ \cos t - \sin t \\ 1 \end{pmatrix},
 \end{aligned}$$

where we computed

$$\frac{1-i}{2} \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix} = \begin{pmatrix} 1/2 \\ 1/2 \\ 0 \end{pmatrix} + i \begin{pmatrix} -1/2 \\ 1/2 \\ 0 \end{pmatrix}$$

Exercise 8.3. Find the solution of

(a) $y'' - 5y' + 4y = 0$ with initial conditions $y(0) = 2$, $y'(0) = 5$.

! Solution

$$y(t) = e^t + e^{4t}.$$

(b) $y'' + 2y' + 2y = 0$.

! Solution

$$e^{-t}(c \cos t + d \sin t).$$

(c) $y'' - 4y' + 4y = 0$ with initial conditions $y(0) = 3$, $y'(0) = 8$.

! Solution

$$y(t) = 3e^{2t} + 2te^{2t}.$$

Exercise 8.4. Consider the equation $y'' + py' + qy$, and assume that the characteristic equation of this has a double root λ .

(a) Show that $y(t) = e^{\lambda t}$ solves the equation.

! Solution

We have that

$$y'' + py' + qy = (\lambda^2 + p\lambda + q)e^{\lambda t} = 0,$$

since λ solves the characteristic equation.

(b) Show that $y(t) = te^{\lambda t}$ solves the equation (you will here need that $\lambda_1 = \lambda_2 = -p/2$, and that $p^2 = 4q$ when we have a double root).

! Solution

We first get that

$$y' = (1 + \lambda t)e^{\lambda t} \qquad y'' = (2\lambda + \lambda^2 t)e^{\lambda t},$$

and then

$$\begin{aligned} y'' + py' + qy &= (2\lambda + \lambda^2 t + p(1 + \lambda t) + qt) e^{\lambda t} \\ &= \left(-p + \frac{p^2}{4}t + p\left(1 - \frac{p}{2}t\right) + \frac{p^2}{4}t \right) e^{\lambda t} \\ &= \left(-p + \frac{p^2}{4}t + p - \frac{p^2}{2}t + \frac{p^2}{4}t \right) e^{\lambda t} = 0. \end{aligned}$$

(c) Show that there is a unique choice of C and D so that $y(t) = Ce^{\lambda t} + Dte^{\lambda t}$ satisfies the initial values for $y(t_0)$ and $y'(t_0)$.

! Solution

We have that

$$y(t_0) = Ce^{\lambda t_0} + Dt_0e^{\lambda t_0} \qquad y'(t_0) = C\lambda e^{\lambda t_0} + D(\lambda t_0 + 1)e^{\lambda t_0},$$

which can be written as

$$e^{\lambda t_0} \begin{pmatrix} 1 & t_0 \\ \lambda & \lambda t_0 + 1 \end{pmatrix} \begin{pmatrix} C \\ D \end{pmatrix} = \begin{pmatrix} y(t_0) \\ y'(t_0) \end{pmatrix}.$$

It is easy to see that the matrix here is invertible, so that there is a unique choice for C and D .

Chapter 9

Orthogonally diagonalisable matrices. Condition numbers.

When solving the system $A\mathbf{x} = \mathbf{b}$ on a computer, the matrix A and the right hand side $\mathbf{b}$ will typically be subject to measurement- and round-off errors. The computer will therefore actually be served a problem on the form

$$(A + E)\mathbf{y} = \mathbf{b} + \mathbf{e},$$

where $E/\mathbf{e}$ is a matrix/vector with small components, and which represent a deviation from the actual coefficient matrix/right hand side. In this chapter we will assume for simplicity that there is no round-off errors in the coefficient matrix, so that $E = 0$. We will ask ourselves the following question: How does the error $\mathbf{e}$ in the right hand side propagate to the solution of the system? Are there any cases where even a small error $\mathbf{e}$ will propagate to a large error between the solutions $\mathbf{x}$ and $\mathbf{y}$? Such behaviour is clearly not desirable, since it would make us vulnerable to small errors.

To illustrate that such a behaviour actually is possible, set

$$A = \frac{1}{2} \begin{pmatrix} 1 + \epsilon & 1 - \epsilon \\ 1 - \epsilon & 1 + \epsilon \end{pmatrix},$$

where ϵ is a number we will experiment with. Consider the system $A\mathbf{x} = \mathbf{b}$, where $\mathbf{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. It is easy to show that the exact solution is $\mathbf{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ for all ϵ . If there is a small error $\mathbf{e}$, we would expect the solution of $A\mathbf{y} = \mathbf{b} + \mathbf{e}$ to be near $\mathbf{x} = (1, 1)$, regardless of what ϵ is. Let us check this with $\mathbf{e} = 10^{-3} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

In Exercise 9.4 you are asked to check that $A\mathbf{y} = \mathbf{b} + \epsilon$ has solution

$$\mathbf{y} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + 10^{-3}\epsilon^{-1} \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Something unexpected occurs here: When $\epsilon \rightarrow 0$ we get very large solutions $\mathbf{y}$, which get further and further away from $\mathbf{x}$. What goes wrong here? The explanation we shall give is that A has a high *condition number* when $\epsilon \rightarrow 0$. Put differently, when $\epsilon \rightarrow 0$ the matrix A will be close to the matrix $\frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, which is not invertible. This is problematic since a system with a non-invertible matrix does not have a unique solution in general.

The behaviour above is best explained if we restrict ourselves to symmetric matrices. In the next section we will explain that all such matrices are what we call orthogonally diagonalisable, and this property will help us in our analysis.

9.1 Orthogonally diagonalisable matrices

In the factorisation of a diagonalisable matrix, one has the term P^{-1} . If P also is orthogonal (i.e., P is real and $P^{-1} = P^T$) we have that

$$A = PDP^T.$$

Real matrices which can be written in this way are called *orthogonally diagonalisable*. Such matrices must be symmetric since

$$A^T = (PDP^T)^T = PD^T P^T = PDP^T = A.$$

Since $D = P^T A P$ and A and P are real, D is also real.

When $P^{-1} = P^T$, the outer form of matrix multiplication gives

$$PDP^T = \sum_{k=1}^n \lambda_k P_{:k} (P_{:k})^T$$

Since the columns in P have length 1, this says that A can be written as a (weighted) sum of projection matrices (see Example 2.10).

The *spectral theorem* says that, the other way, all real symmetric matrices are orthogonally diagonalisable. To prove this is more difficult. First we prove the following.

Proposition 9.1. *A real, symmetric $n \times n$ -matrix A has only real eigenvalues, and all eigenvectors can be written on real form.*

Proof. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. With the definition of the complex scalar product we get

$$\begin{aligned} (A\mathbf{x}) \cdot \mathbf{y} &= (A\mathbf{x})^T \bar{\mathbf{y}} = \mathbf{x}^T A^T \bar{\mathbf{y}} \\ \mathbf{x} \cdot (A\mathbf{y}) &= \mathbf{x}^T \overline{(A\mathbf{y})} = \mathbf{x}^T \bar{A} \bar{\mathbf{y}} = \mathbf{x}^T A \bar{\mathbf{y}}. \end{aligned}$$

Since $A = A^T$ we have equality here for all $\mathbf{x}, \mathbf{y}$, i.e.,

$$(\mathbf{Ax}) \cdot \mathbf{y} = \mathbf{x} \cdot (\mathbf{Ay}). \quad (9.1)$$

If here $\mathbf{x} = \mathbf{y}$ and $\mathbf{Ax} = \lambda\mathbf{x}$ with $\mathbf{x} \neq \mathbf{0}$, substitution in Equation 9.1 gives that

$$\lambda|\mathbf{x}|^2 = \bar{\lambda}|\mathbf{x}|^2.$$

These two are equal if and only if $\lambda = \bar{\lambda}$, so that λ is real. Row reduction of $A - \lambda I$ now gives us that all eigenvectors can be written on real form. $\square$

We now need the following result, which is the one most difficult to prove

Theorem 9.1. *A real, symmetric $n \times n$ -matrix can be written on the form*

$$A = PRP^T,$$

with P orthogonal, and R upper triangular.

The factorisation $A = PRP^T$ with P orthogonal and R upper triangular is also called a *Schur-factorisation*. The theorem holds more generally, but we will only need the result for symmetric matrices.

Proof. For $n = 1$ this is obvious (choose $P = 1$). Suppose now we have shown that all $(n - 1) \times (n - 1)$ -matrices have a Schur-factorisation. Let $\mathbf{x}_1$ be an eigenvector for A corresponding to eigenvalue λ_1 . Let $\mathbf{x}_1$ have length 1. We can assume that both $\mathbf{x}_1$ and λ_1 are real since A is symmetric. Extend $\mathbf{x}_1$ to an orthonormal basis $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ for $\mathbb{R}^n$ (see Proposition 5.5), and set

$$P_1 = (\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \mathbf{x}_n).$$

P_1 is orthogonal. We have that

$$P_1^T A P_1 \mathbf{e}_1 = P_1^T A \mathbf{x}_1 = \lambda_1 P_1^T \mathbf{x}_1 = \lambda_1 \mathbf{e}_1,$$

so that $P_1^T A P_1 = \left(\begin{array}{c|c} \lambda_1 & \mathbf{a} \\ \mathbf{0} & M \end{array} \right)$, where $\mathbf{a}$ is a row vector with $n - 1$ components, and M is an $(n - 1) \times (n - 1)$ matrix. We then have that

$$A = P_1 \left(\begin{array}{c|c} \lambda_1 & \mathbf{a} \\ \mathbf{0} & M \end{array} \right) P_1^T.$$

The matrix M is symmetric since A is, and by induction it is assumed to have a Schur factorisation $M = P_0 R_0 P_0^T$, where P_0 is orthogonal and R_0 upper triangular. If we insert this we get

$$\begin{aligned} A &= P_1 \left(\begin{array}{c|c} \lambda_1 & \mathbf{a} \\ \mathbf{0} & P_0 R_0 P_0^T \end{array} \right) P_1^T \\ &= P_1 \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \mathbf{0} & P_0 \end{array} \right) \left(\begin{array}{c|c} \lambda_1 & \mathbf{a} P_0 \\ \mathbf{0} & R_0 \end{array} \right) \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \mathbf{0} & P_0^T \end{array} \right) P_1^T \\ &= P_1 \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \mathbf{0} & P_0 \end{array} \right) \left(\begin{array}{c|c} \lambda_1 & \mathbf{a} P_0 \\ \mathbf{0} & R_0 \end{array} \right) \left(P_1 \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \mathbf{0} & P_0 \end{array} \right) \right)^T. \end{aligned}$$

Since $P_1 \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \hline \mathbf{0} & P_0 \end{array} \right)$ is orthogonal (this is an exercise) and $\left(\begin{array}{c|c} \lambda_1 & \mathbf{a}P_0 \\ \hline \mathbf{0} & R_0 \end{array} \right)$ is upper triangular, this is a Schur factorisation of A . $\square$

If $A = PRP^T$ is symmetric we have that $PRP^T = PR^TP^T$, so that $R = R^T$. But an upper triangular matrix which is symmetric must be diagonal, so that the Schur-factorisation in fact is an orthogonal diagonalisation. Real, symmetric matrices are also orthogonally diagonalisable, and since we started by saying that orthogonally diagonalisable matrices are symmetric, we have shown the following.

Theorem 9.2 (The spectral theorem). *A real $n \times n$ -matrix is orthogonally diagonalisable if and only if it is symmetric.*

Let us summarise some other things as well. We know that, if $A = PDP^T$, then column i in P is an eigenvector for A corresponding to λ_i . Since the columns in P are orthonormal, it follows that the eigenspaces are orthogonal, and that the dimension of the eigenspace equals the multiplicity of the eigenvalue.

Theorem 9.3. *A real symmetric $n \times n$ -matrix*

1. *has n real eigenvalues,*
2. *has mutually orthogonal eigenspaces,*
3. *has dimension of the eigenspace corresponding to λ equal to the multiplicity of λ as root in the characteristic equation.*

We will have use for the following result.

Lemma 9.1. *Suppose that A is real, symmetric, and invertible, and let*

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n| > 0$$

be the eigenvalues of A listed by descending absolute value. Then we have that

$$|\lambda_n||\mathbf{x}| \leq |A\mathbf{x}| \leq |\lambda_1||\mathbf{x}| \quad (9.2)$$

for all $\mathbf{x}$.

Since symmetric matrices are orthogonally diagonalisable we can write $A = PDP^T$, where P is orthogonal. We have that

$$|A\mathbf{x}|^2 = |PDP^T\mathbf{x}|^2 = \mathbf{x}^T PDP^T PDP^T \mathbf{x} = (P^T\mathbf{x})^T D^2 P^T \mathbf{x} = \mathbf{u}^T D^2 \mathbf{u},$$

where we put $\mathbf{u} = P^T\mathbf{x}$. We then get

$$\begin{aligned} \mathbf{u}^T D^2 \mathbf{u} &= \sum_{i=1}^n \lambda_i^2 u_i^2 \leq \lambda_1^2 \sum_{i=1}^n u_i^2 = \lambda_1^2 |\mathbf{u}|^2 = \lambda_1^2 |\mathbf{x}|^2 \\ \mathbf{u}^T D^2 \mathbf{u} &= \sum_{i=1}^n \lambda_i^2 u_i^2 \geq \lambda_n^2 \sum_{i=1}^n u_i^2 = \lambda_n^2 |\mathbf{u}|^2 = \lambda_n^2 |\mathbf{x}|^2, \end{aligned}$$

where we used that $|\mathbf{u}|^2 = \mathbf{x}^T P P^T \mathbf{x} = \mathbf{x}^T \mathbf{x} = |\mathbf{x}|^2$. By taking square roots it follows that $|\lambda_n||\mathbf{x}| \leq |A\mathbf{x}| \leq |\lambda_1||\mathbf{x}|$.

9.2 Relative and absolute error

The discussion we started this chapter with had to do with the error we get when solving systems. It is therefore important to be precise with the concept of error. When talking about the error between two quantities it is natural to use the concept of relative error:

Definition 9.1. Suppose a and b are numbers. $\left|\frac{b-a}{a}\right|$ is called the *relative error* between a and b . If $\mathbf{a}$ and $\mathbf{b}$ more generally are vectors, we define the relative error between these as $\frac{|\mathbf{b}-\mathbf{a}|}{|\mathbf{a}|}$.

Relative error says more about the size ratio between a and b than what the *absolute error* $|b-a|$ does, since it does not change if we scale the numbers: Since $\left|\frac{cb-ca}{ca}\right| = \left|\frac{b-a}{a}\right|$ for any $c \neq 0$, the relative error between a and b equals the relative error between ca and cb . The same can not be said for the absolute error. Relative error also says something about how many decimals the numbers a and b have in common: If the relative error is about 10^{-k} , this means that approximately the k first decimal digits in a and b agree.

For the systems $\mathbf{A}\mathbf{x} = \mathbf{b}$ and $\mathbf{A}\mathbf{y} = \mathbf{b} + \mathbf{e}$ above there are two different relative errors:

1. The relative error between the right hand sides of the system: $|\mathbf{b} + \mathbf{e} - \mathbf{b}|/|\mathbf{b}| = |\mathbf{e}|/|\mathbf{b}|$.
2. The relative error between the solutions: $|\mathbf{y} - \mathbf{x}|/|\mathbf{x}|$.

The relation

$$\frac{|\mathbf{y} - \mathbf{x}|/|\mathbf{x}|}{|\mathbf{e}|/|\mathbf{b}|} \quad (9.3)$$

between these say how much the relative error between the right hand sides will propagate to the solutions. If Equation 9.3 is about 10^{-n} , this means that approximately n decimal digits in precision can be lost when solving the perturbed system.

In the example above the relative error in 1. equals $|\mathbf{e}|/|\mathbf{b}| = 10^{-3}$. For the relative error in 2. we first obtain

$$\mathbf{y} - \mathbf{x} = \begin{pmatrix} 1 + 10^{-3}\epsilon^{-1} \\ 1 - 10^{-3}\epsilon^{-1} \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 10^{-3}\epsilon^{-1} \begin{pmatrix} 1 \\ -1 \end{pmatrix},$$

and then $|\mathbf{y} - \mathbf{x}|/|\mathbf{x}| = 10^{-3}\epsilon^{-1}$. The ‘‘Propagation factor’’ Equation 9.3 is thus ϵ^{-1} . Since this number is large when ϵ is small, the relative error in the right hand side of the system gets largely magnified when we compare the solutions.

9.3 Definition of condition numbers

In this section we will deduce a result which says how large the propagation factor Equation 9.3 can be. We will show the following.

Proposition 9.2. Suppose that A is symmetric and invertible, and let

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n| > 0$$

be the eigenvalues of A , listed by descending absolute value. Then we have that

$$\left| \frac{\lambda_n}{\lambda_1} \right| \frac{|\mathbf{e}|}{|\mathbf{b}|} \leq \frac{|\mathbf{y} - \mathbf{x}|}{|\mathbf{x}|} \leq \left| \frac{\lambda_1}{\lambda_n} \right| \frac{|\mathbf{e}|}{|\mathbf{b}|} \quad (9.4)$$

Further we have that

1. equality in the right inequality when $\mathbf{e}$ is an eigenvector for λ_n , and $\mathbf{b}$ for λ_1 ,
2. equality in the left inequality when $\mathbf{e}$ is an eigenvector for λ_1 , and $\mathbf{b}$ for λ_n .

If we define the *condition number* of A as $K(A) = |\lambda_1/\lambda_n|$ then Equation 9.4 can be written

$$\frac{1}{K(A)} \frac{|\mathbf{e}|}{|\mathbf{b}|} \leq \frac{|\mathbf{y} - \mathbf{x}|}{|\mathbf{x}|} \leq K(A) \frac{|\mathbf{e}|}{|\mathbf{b}|} \quad (9.5)$$

In Exercise 9.4 you found that the eigenvalues of A are $\epsilon < 1$ and 1 , so that the condition number is $K(A) = \epsilon^{-1}$. Note that

1. the right inequality says that the relative error in the right hand side propagates maximally with a factor $K(A)$ to the solutions,
2. the left inequality can be rewritten as $\frac{|\mathbf{e}|}{|\mathbf{b}|} \leq K(A) \frac{|\mathbf{y} - \mathbf{x}|}{|\mathbf{x}|}$. On this form it says that the relative error in $\mathbf{x}$ propagates maximally with a factor $K(A)$ under the matrix multiplication $\mathbf{b} = A\mathbf{x}$.

Proof. of Proposition 9.2. We first get that $A(\mathbf{y} - \mathbf{x}) = \mathbf{e}$, so that $\mathbf{y} - \mathbf{x} = A^{-1}\mathbf{e}$. If we apply Lemma 9.1 on this, and on $A\mathbf{x} = \mathbf{b}$ we get that

$$|\lambda_n||\mathbf{x}| \leq |\mathbf{b}| \leq |\lambda_1||\mathbf{x}| \quad \frac{1}{|\lambda_1|}|\mathbf{e}| \leq |\mathbf{y} - \mathbf{x}| \leq |\mathbf{e}| \frac{1}{|\lambda_n|}$$

Here we used that A^{-1} also is a symmetric matrix, with eigenvalues $1/\lambda_n, \dots, 1/\lambda_1$, listed by descending absolute value, see Exercise 9.6. If we combine these we get that

$$\left| \frac{\lambda_n}{\lambda_1} \right| \frac{|\mathbf{e}|}{|\mathbf{b}|} \leq \frac{|\mathbf{y} - \mathbf{x}|}{|\mathbf{x}|} \leq \left| \frac{\lambda_1}{\lambda_n} \right| \frac{|\mathbf{e}|}{|\mathbf{b}|}$$

Further, there is equality in the right inequality when $\mathbf{e}$ is an eigenvector for λ_n , and $\mathbf{b}$ is an eigenvector for λ_1 , and equality in the left inequality when $\mathbf{e}$ is an eigenvector for λ_1 , and $\mathbf{b}$ is an eigenvector for λ_n . $\square$

The last part of Proposition 9.2 says that we can have equality in both inequalities by choosing concrete vectors/perturbations. We have already considered the choice in 1.. The choice in 2. corresponds to changing the right hand side

to $\mathbf{b} = (1, -1)$ (the eigenvector for the smallest eigenvalue), and the error to $\mathbf{e} = 10^{-3}(1, 1)$ (the eigenvector for the largest eigenvalue). With

$$\mathbf{x} = \epsilon^{-1} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad \mathbf{y} = \epsilon^{-1} \begin{pmatrix} 1 \\ -1 \end{pmatrix} + 10^{-3} \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

we now get

$$\begin{aligned} A\mathbf{x} &= \epsilon^{-1}A\mathbf{v}_2 = \mathbf{v}_2 = \mathbf{b} \\ A\mathbf{y} &= \epsilon^{-1}A\mathbf{v}_2 + 10^{-3}A\mathbf{v}_1 = \mathbf{v}_2 + 10^{-3}\mathbf{v}_1 = \mathbf{b} + \mathbf{e}. \end{aligned}$$

This gives

$$\begin{aligned} |\mathbf{y} - \mathbf{x}|/|\mathbf{x}| &= 10^{-3}\sqrt{2}/(\epsilon^{-1}\sqrt{2}) = 10^{-3}\epsilon \\ |\mathbf{e}|/|\mathbf{b}| &= 10^{-3}\sqrt{2}/\sqrt{2} = 10^{-3}. \end{aligned}$$

The propagation factor in Equation 9.3 thus becomes ϵ , which gives equality in the left inequality. This choice thus gives maximal propagation of relative error under the matrix multiplication $\mathbf{b} = A\mathbf{x}$.

The example above can seem trivial, since A is near a non-invertible matrix when ϵ is small. The Hilbert matrix from Example 2.4 is another example of a matrix with high condition number. This is not quite as obvious, and here one also gets matrices of all sizes. The code below plots condition numbers for $n \times n$ Hilbert matrices, for different values of n .

```
nvals = np.arange(1, 21)
ch = np.zeros(20)
for k in nvals:
    ch[k-1] = np.linalg.cond(hilbert(k))
plt.semilogy(nvals, ch)
```

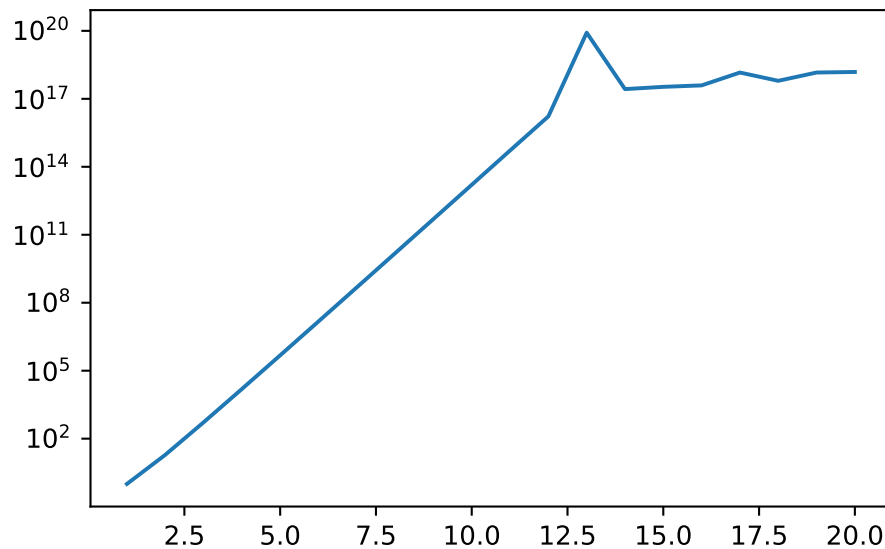


Figure 9.1: Condition numbers of $n \times n$ Hilbert matrices, for different values of n

Here we used the command `cond(A)`, which returns the condition number of a matrix. We have n along the x -axis. The plot is logarithmic (see Exercise 9.8 when it comes to the command `semilogy`), so that we along the y -axis have the number of digits we lose. We see that Hilbert matrices lose about one digit in precision every time we increase n . They are thus difficult to use numerically, even for small matrices and small perturbations.

Condition numbers can be defined more generally for all matrices, not just for the symmetric ones. We will not consider the more general definition, however.

Exercises

Exercise 9.1. Find an orthogonal diagonalisation of the matrix $A = \begin{pmatrix} 14 & -2 \\ -2 & 11 \end{pmatrix}$.

! Solution

The characteristic equation is $(14 - \lambda)(11 - \lambda) - 4 = 0$, which gives $\lambda^2 - 25\lambda + 150 = 0$. The eigenvalues are $\frac{25 \pm \sqrt{625 - 600}}{2} = \frac{25 \pm 5}{2}$, which are $\lambda_1 = 10$ and $\lambda_2 = 15$.

Eigenvectors for $\lambda_1 = 10$:

$$\begin{pmatrix} 14 & -2 \\ -2 & 11 \end{pmatrix} - 10I = \begin{pmatrix} 4 & -2 \\ -2 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & -1/2 \\ 0 & 0 \end{pmatrix}$$

$(1, 2)$ is a particular eigenvector. An eigenvector with length 1 is $(1, 2)/\sqrt{5}$.

Eigenvectors for $\lambda_2 = 15$:

$$\begin{pmatrix} 14 & -2 \\ -2 & 11 \end{pmatrix} - 15I = \begin{pmatrix} -1 & -2 \\ -2 & -4 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix}$$

$(-2, 1)$ is a particular eigenvector. An eigenvector with length 1 is $(-2, 1)/\sqrt{5}$.

An orthogonal diagonalisation is thus

$$A = \frac{1}{\sqrt{5}} \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 10 & 0 \\ 0 & 15 \end{pmatrix} \left(\frac{1}{\sqrt{5}} \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix} \right)^T.$$

Exercise 9.2. Show that $U_1 U_2$ is orthogonal when U_1 and U_2 are orthogonal.

Exercise 9.3. Show that $\left(\begin{array}{c|c} U_1 & \mathbf{0} \\ \hline \mathbf{0} & U_2 \end{array} \right)$ is orthogonal when U_1 and U_2 are orthogonal (here zeros are inserted so that the resulting matrix is square).

Exercise 9.4. As before we set $A = \frac{1}{2} \begin{pmatrix} 1+\epsilon & 1-\epsilon \\ 1-\epsilon & 1+\epsilon \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

(a) Find the eigenvalues of A and the corresponding eigenvectors.

! Solution

We have that

$$\det(\lambda I - A) = (\lambda - (1+\epsilon)/2)^2 - (1-\epsilon)^2/4 = \lambda^2 - (1+\epsilon)\lambda + \epsilon.$$

This has roots

$$\frac{1+\epsilon \pm \sqrt{(1+\epsilon)^2 - 4\epsilon}}{2} = \frac{1+\epsilon \pm (1-\epsilon)}{2}.$$

The largest eigenvalue is $\lambda_1 = 1$, the smallest is $\lambda_2 = \epsilon$. Let us find the eigenvectors.

- An eigenvector for $\lambda_1 = 1$:

$$I - A = \begin{pmatrix} 1 - (1+\epsilon)/2 & -(1-\epsilon)/2 \\ -(1-\epsilon)/2 & 1 - (1+\epsilon)/2 \end{pmatrix} = \begin{pmatrix} (1-\epsilon)/2 & -(1-\epsilon)/2 \\ -(1-\epsilon)/2 & (1-\epsilon)/2 \end{pmatrix} \sim \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix},$$

so that we can set $\mathbf{v}_1 = (1, 1)$.

- An eigenvector for $\lambda_2 = \epsilon$:

$$\epsilon I - A = \begin{pmatrix} \epsilon - (1 + \epsilon)/2 & -(1 - \epsilon)/2 \\ -(1 - \epsilon)/2 & \epsilon - (1 + \epsilon)/2 \end{pmatrix} = \begin{pmatrix} -(1 - \epsilon)/2 & -(1 - \epsilon)/2 \\ -(1 - \epsilon)/2 & -(1 - \epsilon)/2 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix},$$

so that we can set $\mathbf{v}_2 = (1, -1)$.

- (b) Set $\mathbf{e} = 10^{-3} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$. Show that the solution of $A\mathbf{y} = \mathbf{b} + \mathbf{e}$ is

$$\mathbf{y} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + 10^{-3}\epsilon^{-1} \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

! Solution

We have that $\mathbf{y} = \mathbf{v}_1 + 10^{-3}\epsilon^{-1}\mathbf{v}_2$, so that

$$A\mathbf{y} = A\mathbf{v}_1 + 10^{-3}\epsilon^{-1}A\mathbf{v}_2 = \mathbf{v}_1 + 10^{-3}\mathbf{v}_2 = \mathbf{b} + \mathbf{e}.$$

Exercise 9.5. Let A be the matrix $A = \begin{pmatrix} 0.2 & 0.8 \\ 0.8 & 0.2 \end{pmatrix}$.

- (a) Find the eigenvalues and eigenvectors of A .

! Solution

We have that

$$\det(\lambda I - A) = (\lambda - 0.2)^2 - 0.64,$$

and this is 0 when $\lambda - 0.2 = \pm 0.8$. The eigenvalues are thus $\lambda_1 = -0.6$ and $\lambda_2 = 1$.

Eigenvector for $\lambda_1 = -0.6$:

$$-0.6I - A = \begin{pmatrix} -0.8 & -0.8 \\ -0.8 & -0.8 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.$$

$\mathbf{v}_1 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$ is a particular eigenvector.

Eigenvector for $\lambda_2 = 1$:

$$I - A = \begin{pmatrix} 0.8 & -0.8 \\ -0.8 & 0.8 \end{pmatrix} \sim \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}$$

$\mathbf{v}_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ is a particular eigenvector.

- (b) What is the condition number $K(A)$?

! Solution

The matrix is symmetric. It is also invertible, since both eigenvalues are $\neq 0$. The condition number is thus the ratio between the largest and smallest eigenvalue (in absolute value), which gives $K(A) = 1/0.6 = 5/3$.

Exercise 9.6. Suppose A is invertible, symmetric, and $n \times n$, with eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$. Prove that A^{-1} also is symmetric, with eigenvalues $1/\lambda_1, \dots, 1/\lambda_n$.

! Solution

Since A is invertible, all $\lambda_i \neq 0$. If we transpose both sides in $A^{-1}A = I$ we get that $A^T(A^{-1})^T = I$. Since A is symmetric this means that $A(A^{-1})^T = I$, so that $A^{-1} = (A^{-1})^T$. It follows that A^{-1} also is symmetric.

Suppose $A\mathbf{v} = \lambda\mathbf{v}$. Then $\mathbf{v} = A^{-1}(\lambda\mathbf{v})$, so that $A^{-1}\mathbf{v} = \frac{1}{\lambda}\mathbf{v}$. It follows that $1/\lambda$ is an eigenvalue for A^{-1} when λ is an eigenvalue for A . A and A^{-1} also have the same eigenvectors.

This also follows easily from that A is orthogonally diagonalisable, so that we can write $A = PDP^{-1}$, where D is the diagonal matrix with eigenvalues λ_i on the diagonal. From this it follows that $A^{-1} = PD^{-1}P^T$, where D^{-1} is the diagonal matrix with $1/\lambda_i$ on the diagonal.

Exercise 9.7. An $n \times n$ -matrix is called orthogonal if $U^{-1} = U^T$. Explain that the condition number of U is 1. This is the case regardless of whether U is symmetric or not. What does this tell us about such matrices when it comes to solving a perturbed system?

! Solution

It is clear that

$$|U\mathbf{x}|^2 = \mathbf{x}^T U^T U \mathbf{x} = |\mathbf{x}|^2.$$

It follows that, if $\mathbf{x}$ is an eigenvector, then $|U\mathbf{x}| = |\lambda||\mathbf{x}| = |\mathbf{x}|$, so that $|\lambda| = 1$.

Since all eigenvalues are equal in absolute value it follows that $K(A) = 1$. Note that we do not appeal to orthogonal diagonalisability here.

Exercise 9.8. If $y = Ke^{kx}$, we say that y is *exponential* in x , with constant K and exponent k .

- (a) The command `plt.semilogx(x,y)` plots $\log x$ against y . Suppose that `plt.semilogx(x,y)` displays a line. Show that then x must be exponential in y . How can we find the constant K and the exponent k ?

! Solution

A line in such a plot means that $y = a \log x + b$, where a and b are constants we can read out from the line (a is the slope, b is the intersection with the y -axis). $y = a \log x + b$ gives that $x = e^{(y-b)/a} = e^{-b/a} e^{y/a}$, so that x is exponential in y with constant $K = e^{-b/a}$ and exponent $1/a$.

- (b) The command `plt.semilogy(x,y)` in the same way plots x against $\log y$. Suppose that `plt.semilogy(x,y)` displays a line. Show that then y must be exponential in x . How can we find the constant and the exponent now?

! Solution

A line in such a plot means that $\log y = ax + b$. This gives that $y = e^b e^{ax}$, so that y is exponential in x with constant $K = e^b$ and exponent a .

Part II

Numerical methods

Chapter 10

Interpolation

A basic idea in mathematics is to approximate something complicated with something simpler, in the hope that the approximation captures some of the most important information. The theory of *approximation* has been central in mathematics since the dawn of calculus, and is often useful in the study of functions. Most functions in mathematics can't be expressed as combinations of the elementary functions we had in school. Nevertheless, so to say all functions of practical interest can be approximated arbitrarily well with simple functions like polynomials, trigonometric functions, or exponential functions. In computational mathematics polynomials and trigonometric functions in particular are used much.

Interpolation is one particular branch of approximation theory. To explain the setting, consider a function f which may only be known at a set of points $x_0, x_1, \dots, x_m \in [a, b]$ (which are assumed to be distinct). We say that the function $p: [a, b] \rightarrow \mathbb{R}$ *interpolates* f in these points if

$$f(x_k) = p(x_k) \quad \text{for } k = 0, 1, \dots, m. \quad (10.1)$$

The $x_0, x_1, \dots, x_m$ are called *interpolation points*. The basic question in interpolation theory is the following:

Given functions $\phi_0(x), \phi_1(x), \dots, \phi_n(x)$. How can we find scalars $c_0, c_1, \dots, c_n$ so that the linear combination

$$p(x) = c_0\phi_0(x) + c_1\phi_1(x) + \dots + c_n\phi_n(x) \quad (10.2)$$

interpolates f in $x_0, x_1, \dots, x_m$?

The functions $\phi_i(x)$ will be called *basis functions*. The above leads to the system

$$\begin{aligned} c_0\phi_0(x_0) + c_1\phi_1(x_0) + \cdots + c_n\phi_n(x_0) &= f(x_0) \\ c_0\phi_0(x_1) + c_1\phi_1(x_1) + \cdots + c_n\phi_n(x_1) &= f(x_1) \\ &\vdots \\ c_0\phi_0(x_m) + c_1\phi_1(x_m) + \cdots + c_n\phi_n(x_m) &= f(x_m). \end{aligned} \quad (10.3)$$

Setting $\mathbf{c} = (c_0, c_1, \dots, c_n)$ and $\mathbf{f} = (f(x_0), f(x_1), \dots, f(x_m))$, This gives the matrix equation

$$\begin{pmatrix} \phi_0(x_0) & \phi_1(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_m) & \phi_1(x_m) & \cdots & \phi_n(x_m) \end{pmatrix} \mathbf{c} = \mathbf{f} \quad (10.4)$$

which has a unique solution $\mathbf{c}$ for all vectors $\mathbf{f}$ if and only if the coefficient matrix (which has components $\phi_j(x_i)$) is invertible. It must thus be square, i.e., the number of interpolation points must equal the number of basis functions. This will always be the case in what follows. Once Equation 10.4 has been solved for the column vector $\mathbf{c}$, the interpolant can easily be plotted: For any x Equation 10.2 can be rewritten as

$$p(x) = (\phi_0(x) \quad \phi_1(x) \quad \cdots \quad \phi_n(x)) \mathbf{c}$$

which means that we obtain a vector of plot points¹ as the matrix-vector product

$$\begin{pmatrix} p(x_0) \\ p(x_1) \\ p(x_2) \\ \vdots \end{pmatrix} = \begin{pmatrix} \phi_0(x_0) & \phi_1(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} \quad (10.5)$$

Our plots of the interpolant will use this expression.

We start this chapter with *interpolation with polynomials*, where the basis functions ϕ_j are assumed to be polynomials of degree j . When we choose the basis functions as $\{1, x^1, \dots, x^n\}$ we will get what we call the *canonical form* of the interpolation polynomial (Section 10.1). When we choose the basis functions to be

$$\{1, x - x_0, (x - x_0)(x - x_1), \dots, (x - x_0)(x - x_1) \cdots (x - x_{n-1})\}$$

we get what we call the *Newton form* of the interpolating polynomial (Section 10.2). Other choices for $\{\phi_k\}_{k=0}^n$ are also possible - the point is that all polynomials are expressible in terms of them.

We then consider the error in polynomial interpolation (Section 10.3), and find an expression for this similar to that of the remainder term in Taylor series.

¹the vector of plot points is usually much longer than the vector of interpolation points, although they here are denoted in the same way.

We end the chapter with interpolation with trigonometric functions in Section 10.4. In this setting the basis functions are on the form $\sin(2\pi nt/T)$, or $\cos(2\pi nt/T)$, and defined on an interval $[0, T]$.

In the next chapters we will use interpolating polynomials to derive numerical methods for the derivatives, and integrals, of functions.

10.1 Interpolation on canonical form

When the basis functions are $\phi_i(x) = x^i$ the interpolating polynomial takes the form

$$p_n(x) = c_0 + c_1x + c_2x^2 + \dots + c_nx^n. \quad (10.6)$$

This is called the *canonical form* of the interpolating polynomial. The matrix form in Equation 10.4 now takes the form

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix} \quad (10.7)$$

The left side matrix is a square vandermonde matrix. From Example 4.3 we know that these are invertible. Equation 10.7 therefore has a unique solution for $\mathbf{c}$, and we can conclude that there is a unique interpolating polynomial. The function in Listing 10.1 plots this: It takes the vectors $\mathbf{x} = (x_0, x_1, \dots, x_n)$ and $\mathbf{f} = (y_0, y_1, \dots, y_n)$ as input, finds the coefficients $\mathbf{c} = (c_0, c_1, \dots, c_n)$ of the polynomial interpolating in these points, and repeats what we did in Section 2.4 to plot this. Since we are only interested in the solution to the system $V\mathbf{c} = \mathbf{f}$, it is more efficient to do as in the code, instead of computing $\mathbf{c} = V^{-1}\mathbf{f}$.

Listing 10.1 Code for finding points of the interpolating polynomial, adapted to canonical form

```
def points_canonical(xsp, fsp):
    numpoints = xsp.shape[0] - 1
    n = np.arange(0, numpoints+1).reshape((1, -1))
    c = np.linalg.solve(xsp**n, fsp)
    x = np.linspace(xsp[0,0], xsp[-1,0], 100).reshape((-1, 1))
    p = x**n @ c
    return x, p
```

Some comments on this code are needed:

- The code assumes that $\mathbf{xsp}$ and $\mathbf{fsp}$ are column vectors, and have equal number of components.
- $\mathbf{x**n}$ computes a vandermonde matrix where the generators are listed in the column vector $\mathbf{x}$, and the powers in the row vector $\mathbf{n}$.

- The coefficients `c` of the interpolating polynomial are found by solving Equation 10.7.
- `x` is a column vector with plot points, and which covers the interpolation points.
- The code `x**n @ c` computes the expression in Equation 10.5, returning a column vector with values of the interpolating polynomial.

The code above makes priority of simplicity rather than efficiency and memory usage. A Vandermonde matrix can be computed with about n^2 operations on a computer (see Exercise 10.1). In Section 3.1 we saw that about $\approx \frac{2}{3}n^3$ operations are required in order to solve a general system with n equations and n unknowns. In other words: It is more demanding to solve the system than computing the matrix for the system.

Once the plot points have been obtained it is easy to plot the interpolating polynomial for a function over a given interval.

Example 10.1. Suppose we want to find the polynomial passing through the three points $(0, 1)$, $(1, 3)$, and $(2, 2)$. In other words, we want to find a polynomial p so that

$$p(0) = 1, \quad p(1) = 3, \quad p(2) = 2. \quad (10.8)$$

Since there are three points we assume a quadratic polynomial $p(x) = c_0 + c_1x + c_2x^2$. Inserting in Equation 10.8 we get the three equations.

$$\begin{aligned} 1 &= p(0) = c_0, \\ 3 &= p(1) = c_0 + c_1 + c_2, \\ 2 &= p(2) = c_0 + 2c_1 + 4c_2. \end{aligned}$$

Solving these gives $c_0 = 1$, $c_1 = 7/2$, and $c_2 = -3/2$, so that p is given by

$$p(x) = 1 + \frac{7}{2}x - \frac{3}{2}x^2.$$

A plot of this polynomial is shown in Figure 10.1. The code verifies that the computation above gives the same plot as `points_canonical`.

```
xsp = np.array([[0],[1],[2]])
fsp = np.array([[1],[3],[2]])
x, p = points_canonical(xsp, fsp)
plt.plot(x,p)
plt.plot(x,1+7*x/2-3*x**2/2,'--')
```

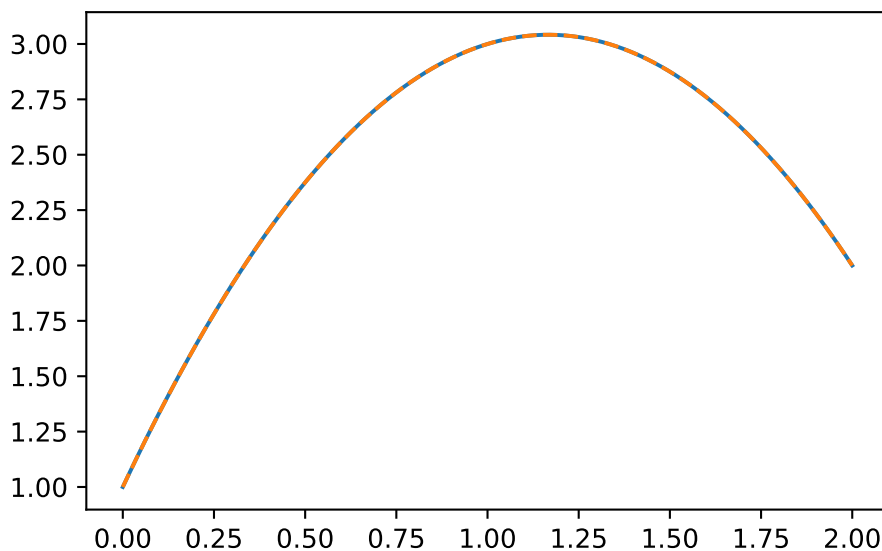


Figure 10.1: Three interpolation point and the corresponding quadratic interpolation polynomial.



If we instead of the interpolation points are given the function itself, we can plot one or several interpolating polynomials together with the help of Listing 10.2. The code takes as input a function `points_p_n` which computes the plot points. At start we set this to `points_canonical`. In the next section we will replace this with another method for computing the plot points.

Listing 10.2 Code for the plotting the interpolating polynomials

```
def plot_p_n(f, a, b, nvals, points_p_n):
    x = []
    for n in nvals:
        xsp = np.linspace(a, b, n + 1).reshape((-1, 1))
        x, p = points_p_n(xsp, f(xsp))
        plt.plot(x, p, '--', label = f'p_{n}')
    plt.plot(x, f(x), '-k', label = 'f')
    plt.legend()
```

In Figure 10.2 we have used this code to plot the interpolating polynomials p_2 , p_3 , and p_4 , for $f(x) = \cos(2\pi x)$, given a uniform partition of $[0, 1]$.

```
plot_p_n( lambda x: np.cos(2*np.pi*x),
          0, 1, [2,3,4,5], points_canonical)
```

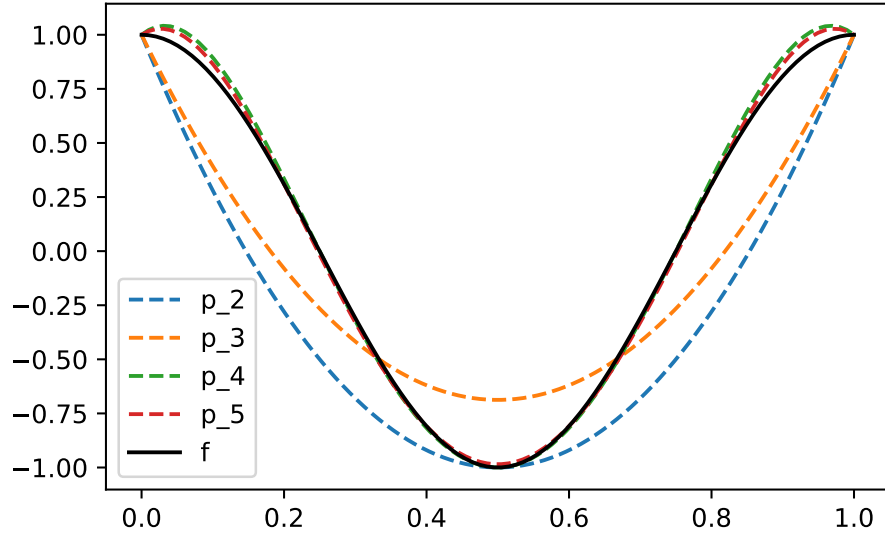


Figure 10.2: The interpolating polynomials p_2 , p_3 , and p_4 , for $f(x) = \cos(2\pi x)$. f is also shown.

10.2 Interpolation on Newton form

Let again $x_0, x_1, \dots, x_n$ be interpolation points. We say that the polynomial p_n is on *Newton form* if it is written on the form.

$$p_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \dots + c_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}). \quad (10.9)$$

The Newton form has some advantages compared to the canonical one. The interpolation conditions now take the form

$$\begin{aligned} f(x_0) &= c_0 \\ f(x_1) &= c_0 + c_1(x_1 - x_0) \\ &\vdots \\ f(x_n) &= c_0 + c_1(x_n - x_0) + \dots + c_n(x_n - x_0) \cdots (x_n - x_{n-1}), \end{aligned} \quad (10.10)$$

or on matrix form:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x_0 & \cdots & (x_n - x_0) \cdots (x_n - x_{n-1}) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}. \quad (10.11)$$

This is a *triangular system*, and if the interpolation points are distinct, all components on the diagonal will be nonzero. The determinant of the matrix is then also nonzero, so that the matrix is invertible. Equation 10.11 thus has a unique solution $(c_0, c_1, \dots, c_n)$. In other words, the Newton form is unique.

Triangular systems are simpler to solve than general systems. To see this, denote the triangular matrix by A , let $A = (a_{ij})_{i,j=0}^n$ be the components of A , and let $A\mathbf{c} = \mathbf{f}$ be the system we want to solve. It is clear that we can find c_0 from the first equation, since this only says that $c_0 = f_0$ (recall that $a_{00} = 1$). Suppose we have found $c_0, c_1, \dots, c_{k-1}$ from the k first equations. Equation $k+1$ says that

$$a_{k0}c_0 + a_{k1}c_1 + \dots + a_{k(k-1)}c_{k-1} + a_{kk}c_k = f_k.$$

We can write this as

$$c_k = \frac{1}{a_{kk}} (f_k - a_{k0}c_0 - a_{k1}c_1 - \dots - a_{k(k-1)}c_{k-1}) \quad (10.12)$$

(the diagonal element $a_{kk} = x_k - x_0$ is nonzero, so this expression gives meaning). We can therefore find c_k from equation $k+1$ in the system. This method to solve a triangular system is called *forward substitution*: We first find c_0 , use this to find c_1 , then use these to find c_2 , and so on.

Using Equation 10.12 to find c_k requires k subtractions and k multiplications. If we sum over all k we get

$$2 + 4 + 6 + \dots + 2n = 2(1 + 2 + \dots + n) = n(n+1) \approx n^2$$

operations (we approximate with n^2 since n^2 is the dominating term here). In addition we have $\approx n^2/2$ operations to find the numbers $x_i - x_j$. Together this gives $3n^2/2$ operations. This is a great reduction when we compare with $\approx \frac{2}{3}n^3$ operations for the canonical system. By adapting the form of the interpolating polynomials, we thus get computation reduction for our method. The learning outcome for the newton form is thus that it can be smart to look at other ways to represent polynomials.

In Listing 10.3 we computed values of the interpolating polynomial using the Newton form, analogous to Listing 10.1 for the canonical form. The first `for`-loop applies forward substitution to find $c_0, c_1, \dots, c_n$.

Example 10.2 (The Newton form for $n = 0, 1$, and 2). If there is only one interpolation point x_0 , the Newton form becomes $p_0(x) = c_0$. We get that $c_0 = f(x_0)$, so that

$$p_0(x) = f(x_0).$$

With two points x_0 and x_1 the newton form becomes $p_1(x) = c_0 + c_1(x - x_0)$. We get the equations

$$\begin{aligned} f(x_0) &= c_0 \\ f(x_1) &= c_0 + c_1(x_1 - x_0) \end{aligned}$$

Listing 10.3 Code to find values of the interpolating polynomial, adapted to Newton form

```
def points_newton(xsp, fsp):
    numpoints = np.shape(xsp)[0] - 1
    x = np.linspace(xsp[0], xsp[-1], 100).reshape((-1,1))
    V = np.ones((100, numpoints + 1))
    c = np.zeros((numpoints+1,1))
    c[0,0] = fsp[0,0]
    for k in range(1,numpoints+1):
        s = fsp[k,0]
        p = 1
        for r in range(0,k):
            s = s - c[r,0]*p
            p = p*(xsp[k,0] - xsp[r,0])
        c[k,0] = s/p
        V[:, [k]] = V[:, [k-1]]*(x - xsp[k-1,0])
    p = V @ c
    return x, p
```

which have the solution

$$c_0 = f(x_0), \quad c_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}. \quad (10.13)$$

This gives

$$p_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0),$$

which simply is the secant of f through x_0 and x_1 , see Figure 10.3. For comparison, the first order Taylor polynomial of f gives the tangent at the point.

If we add yet another point x_2 we get a quadratic interpolation polynomial:

$$p_2(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1)$$

If we insert $x = x_0$ and $x = x_1$ we get the equations

$$\begin{aligned} f(x_0) &= c_0, \\ f(x_1) &= c_0 + c_1(x_1 - x_0), \end{aligned}$$

These are the same equations we solved for $n = 1$, so c_0 and c_1 must be given by Equation 10.13 also when $n = 2$. If we insert $x = x_2$ we get

$$f(x_2) = c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1),$$

which gives

$$c_2 = \frac{f(x_2) - f(x_0) - \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)}.$$

After some simplification this can be written

$$c_2 = \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2-x_0}. \quad (10.14)$$

The interpolating polynomial is now a parabola, see Figure 10.3.

```
plot_p_n(np.exp, 0, 2, [1,2], points_newton)
```

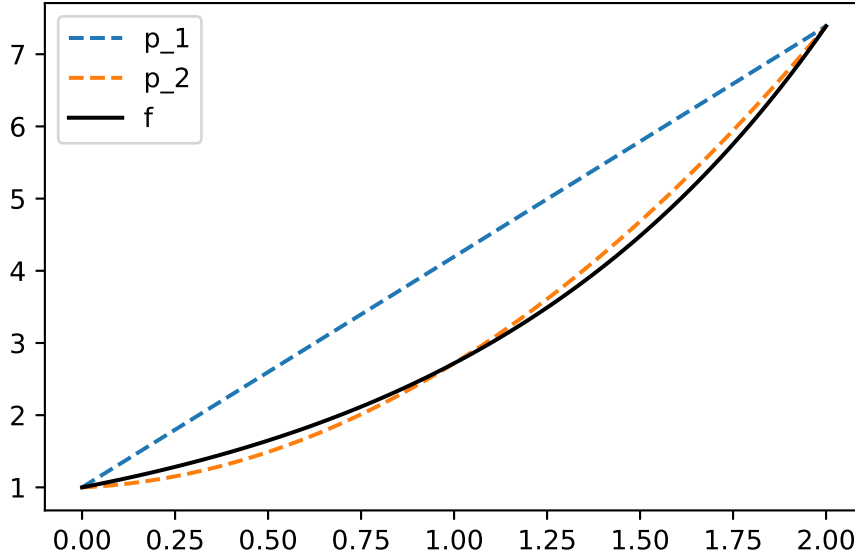


Figure 10.3



Example 10.3. Let us return to Example 10.1, where the interpolation points were $(0, 1)$, $(1, 3)$, and $(2, 2)$, and make the computations on Newton form. We write the interpolating polynomial as

$$p_2(x) = c_0 + c_1x + c_2x(x-1).$$

If we insert the points we get the three equations

$$\begin{aligned} 1 &= p(0) = c_0, \\ 3 &= p(1) = c_0 + c_1, \\ 2 &= p(2) = c_0 + 2c_1 + 2c_2, \end{aligned}$$

which have the solution $c_0 = 1$, $c_1 = 3 - c_0 = 2$, and $c_2 = (2 - c_0 - 2c_1)/2 = -3/2$, so that the Newton form of the interpolating polynomial is

$$p_2(x) = 1 + 2x - \frac{3}{2}x(x-1)$$



More generally we see from the triangular system in Equation 10.10 that the equation obtained from the insertion $x = x_k$ only involves the points $(x_0, f(x_0))$, $(x_1, f(x_1))$, ..., $(x_k, f(x_k))$. Thus, c_k will only depend on these points. It is therefore usual to write

$$c_k = f[x_0, \dots, x_k] \quad (10.15)$$

for $k = 0, 1, \dots, n$, so that the Newton form

$$p_n(x) = c_0 + c_1(x - x_0) + \dots + c_n(x - x_0)(x - x_1) \dots (x - x_{n-1}) \quad (10.16)$$

can be written

$$\begin{aligned} p_n(x) = & f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ & + \dots + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \dots (x - x_{n-1}). \end{aligned} \quad (10.17)$$

Note that the interpolating polynomials p_n and p_{n-1} satisfy

$$p_n(x) = p_{n-1}(x) + f[x_0, \dots, x_n](x - x_0) \dots (x - x_{n-1}).$$

Figure 10.4 shows the interpolating polynomial p_n for different values of n for the function $f(x) = \sin(x)$. We see that the approximation becomes better when we increase n .

```
plot_p_n(np.sin, 0, 4, [1,2,3,4], points_newton)
```

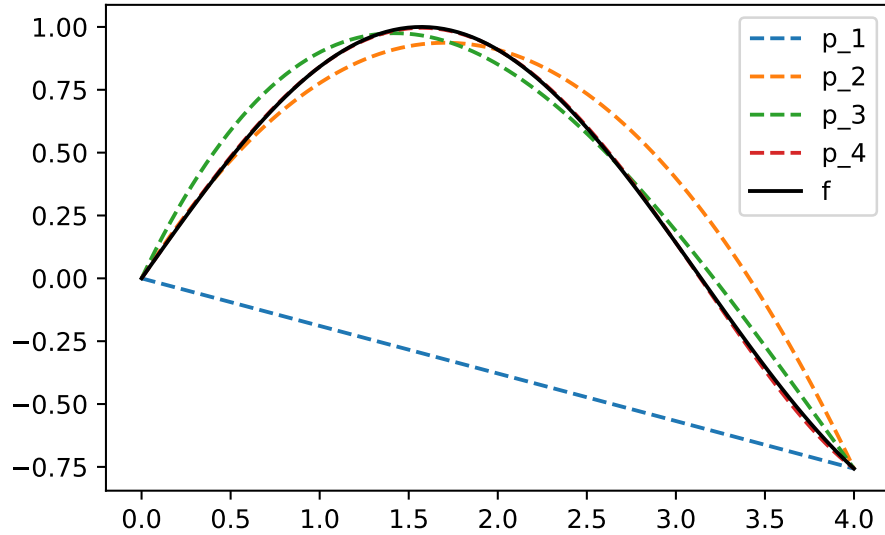


Figure 10.4: Interpolation of $f(x) = \sin(x)$ with polynomials of different degrees.

Example 10.4. We interpolate the function $f(x) = \sqrt{x}$ in the points $x_i = i$, for $i = 0, 1, 2, 3$. The Newton form becomes

$$p_3(x) = c_0 + c_1x + c_2x(x-1) + c_3x(x-1)(x-2),$$

and the interpolation conditions give

$$\begin{aligned} 0 &= c_0, \\ 1 &= c_0 + c_1, \\ \sqrt{2} &= c_0 + 2c_1 + 2c_2, \\ \sqrt{3} &= c_0 + 3c_1 + 6c_2 + 6c_3. \end{aligned}$$

If we solve this system we get

$$c_0 = 0, \quad c_1 = 1, \quad c_2 = -(1 - \sqrt{2})/2, \quad c_3 = (3 + \sqrt{3} - 3\sqrt{2})/6.$$

Figure 10.5 shows the interpolating polynomial.

```
plot_p_n(np.sqrt, 0, 3, [3], points_newton)
```

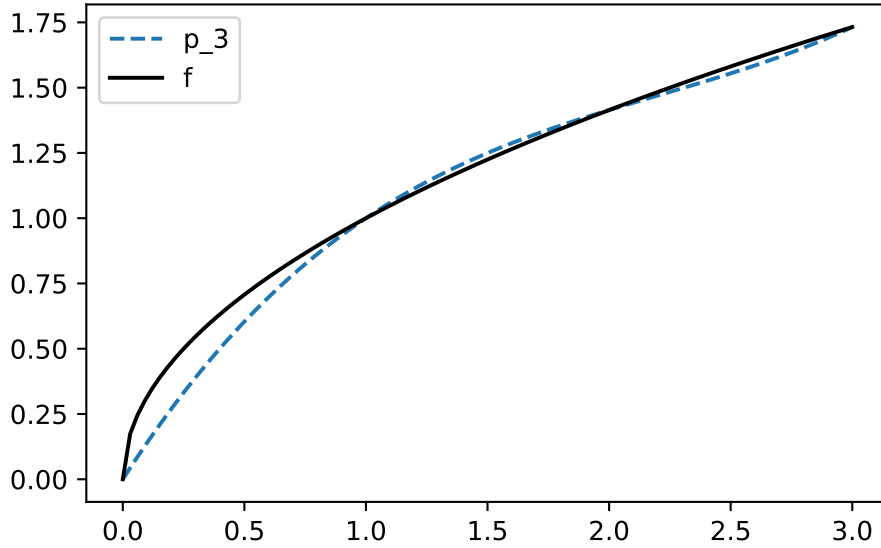


Figure 10.5: The function $f(x) = \sqrt{x}$ (solid line) and the cubic interpolant in the four points 0, 1, 2, and 3.



If interpolation is used the right way, it will usually give a good approximation to the underlying function. As the distance between neighbouring points decrease

(either by increasing the number of points, or by moving the points closer to each other), one can expect that the approximation gets better. But we know that there exist functions for which Taylor polynomials are bad approximations, and the same thing can happen for interpolation. We can get this problem when the derivative of the function gets very large, as the following example shows.

Example 10.5. The *Runge function* is defined by $f(x) = 1/(1 + x^2)$ on the interval $[-5, 5]$. The interpolants of degree 10 and 20 are shown in Figure 10.6.

```
plot_p_n(lambda x: 1/(1+x**2), -5, 5, [10,20], points_newton)
plt.axis([-5,5,-0.5,2])
```

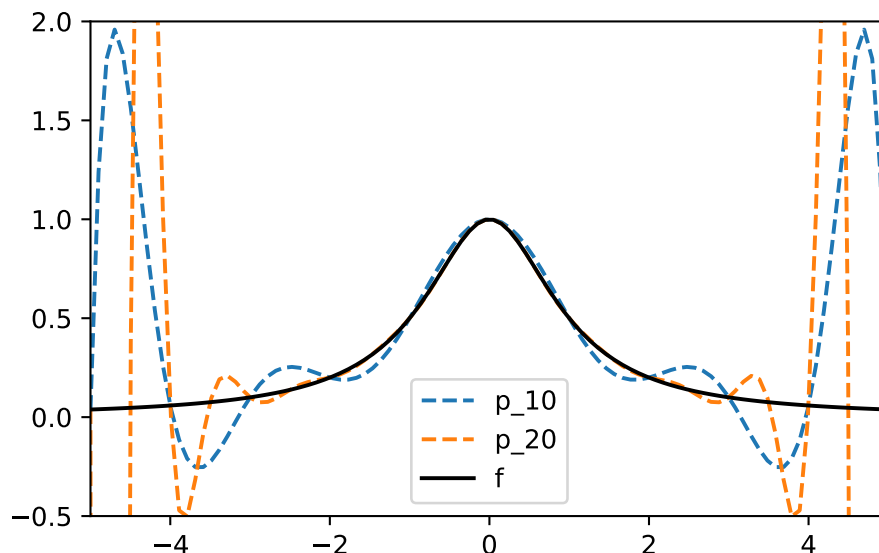


Figure 10.6: Interpolation of the Runge function $f(x) = 1/(1 + x^2)$ on the interval $[-5, 5]$.

Observe what happens when the degree increases: The error then gets less in the middle of the interval, but larger near the end points. We will analyse what goes wrong in Section 10.3.



The Newton- and canonical forms are just two ways to write the interpolating polynomial - many other variants exist also. One of these is the *Lagrange form*, which we will consider in Exercise 10.5.

10.3 The interpolation error

In the previous section we saw examples where the interpolating polynomial p_n becomes a better approximation when we increase n , but we also saw examples where the opposite happens. We will now give a mathematical explanation of why this happens.

We shall explain that the error in polynomial interpolation can be expressed in a way similar to the remainder term in Taylor series, which says that if f is $n+1$ times continuously differentiable, then for any x there is a number $\xi \in [a, x]$ so that

$$f(x) = p(x) + \frac{(x-a)^{n+1}}{(n+1)!} f^{(n+1)}(\xi). \quad (10.18)$$

If x is near a , and the derivative $f^{(n+1)}$ is not very large, the remainder will be small, and p is therefore near f . In other words: If the derivatives of f are not very large, the Taylor polynomial p will be a good approximation to f , at least near a .

Theorem 10.1. *Let $x_0, x_1, \dots, x_n \in [a, b]$ be distinct interpolation points, and suppose that the function $f: [a, b] \rightarrow \mathbb{R}$ is (at least) $n+1$ times continuously differentiable. Let p_n be the unique interpolating polynomial of degree $\leq n$ in the points $x_0, x_1, \dots, x_n$. Then*

$$|f(x) - p_n(x)| \leq \frac{M_{n+1}(b-a)^{n+1}}{(n+1)!} \quad (10.19)$$

for all $x \in [a, b]$, where $M_{n+1} = \max_{[a, b]} |f^{(n+1)}|$.

Proof. Let $x \in [a, b]$. We claim that there is a point $y \in [a, b]$ so that

$$f(x) - p_n(x) = \frac{f^{(n+1)}(y)}{(n+1)!} Q_n(x), \quad (10.20)$$

where Q_n is the degree $(n+1)$ polynomial

$$Q_n(x) = (x-x_0)(x-x_1)\cdots(x-x_n).$$

If x is one of the points $x_0, x_1, \dots, x_n$, then both sides of Equation 10.20 are 0, so that the identity is automatically true. Suppose then that x is NOT one of the points $x_0, x_1, \dots, x_n$, and define the function

$$\phi(t) = f(t) - p_n(t) - \frac{f(x) - p_n(x)}{Q_n(x)} Q_n(t) \quad \text{for } t \in [a, b].$$

Then ϕ has zeros in the $n+2$ distinct points $x, x_0, x_1, \dots, x_n$ (this is Exercise 10.11). By the mean value theorem it follows then that there exist $n+1$ points in the interval (a, b) where ϕ' equals 0. By the same arguments, there must exist n points where $(\phi')' = \phi''$ equals 0. Continuing this way, we find

that the function $\phi^{(n+1)}$ has at least one zero $y \in (a, b)$. Since $p_n^{(n+1)} \equiv 0$ and $Q^{(n+1)} \equiv (n+1)!$ (this is Exercise 10.12), we get

$$0 = \phi^{(n+1)}(y) = f^{(n+1)}(y) - \frac{f(x) - p_n(x)}{Q_n(x)}(n+1)!.$$

Reorganising this equation we get Equation 10.20.

If we replace $|f^{(n+1)}(y)|$ on the right hand side in Equation 10.20 with its maximum, we get

$$|f(x) - p_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |Q_n(x)|.$$

Since each of the terms $(x - x_0), \dots, (x - x_n)$ can be at most $b - a$ (because the points $x, x_0, x_1, \dots, x_n$ all lie in $[a, b]$), we see that $|Q_n(x)| \leq (b - a)^{n+1}$ for all x . This proves Equation 10.19. $\square$

Since the faculty $(n+1)!$ gets very large when n increases, we see that the interpolation error decreases when n grows, AS LONG AS M_{n+1} does not grow too fast!

Example 10.6. In Figure 10.4 we interpolated the function $f(x) = \sin(x)$ on the interval $[a, b] = [0, 2\pi]$. If we choose $n+1$ uniformly distributed interpolation points $x_0, x_1, \dots, x_n$, we get an interpolating polynomial of degree $\leq n$. We know that the derivatives of f are either $\pm \cos(x)$ or $\pm \sin(x)$, and none of these functions obtain values larger than 1. Therefore, $M_{n+1} \leq 1$ for all n . We therefore get

$$|f(x) - p_n(x)| \leq \frac{(2\pi)^{n+1}}{(n+1)!}.$$

Since $(n+1)!$ grows much faster than $(2\pi)^{n+1}$ when n grows, the error will go quickly to 0 when n grows. ♣

Example 10.7. Let f be the Runge function from Example 10.5, given by $f(x) = 1/(1+x^2)$ for $x \in [-5, 5]$. A slightly technical computation shows that $M_{n+1} \approx (n+1)!$ on the interval $[-5, 5]$. The error estimate in Equation 10.19 therefore says that

$$|f(x) - p_n(x)| \leq \frac{M_{n+1}(b-a)^{n+1}}{(n+1)!} \approx (b-a)^{n+1} = 10^{n+1},$$

which is a very large number when n is large. The error estimate thus DOES NOT guarantee that the interpolation error is small when n is large. And, as we saw in Example 10.5, the error is actually quite large. ♣

10.4 Interpolation with trigonometric functions

We shall now interpolate with trigonometric functions, rather than with polynomials. Trigonometric functions can be thought of as the building blocks of

sound. Think of a sound as a mathematical function which returns its amplitude at a given instance in time. A sine wave on the form $\sin(2\pi t/T)$ (T is the period of the sine wave) in particular makes a distinct sound it may be possible to match with a key on a piano. And adding many sine waves with different periods together, it turns out that more complex sounds can be obtained. We will therefore illustrate interpolation with trigonometric functions in terms of applications to sound.

Sound on a computer is represented as numbers between -1 and 1 , called *sound samples*. In many applications sound is represented by 44100 samples for each second, also called the *sampling frequency*, denoted `freq_s`.

The sounds we consider will be periodic. If a sound repeats 440 times per second we say that its *frequency* is $f = 440\text{Hz}$. The period is then $T=1/\text{freq}$, and the number of samples per period is `freq_s/freq` (which may not be an integer). In the following we will plot the sound samples against time. First we will plot sounds over one period.

```
freq_s = 44100
freq = 440
T= 1/freq
samples_per_period = int(freq_s/freq)
t = np.linspace(0,T,samples_per_period).reshape((-1,1))
```

The interpolating function p will agree with the function (denoted as before by f) at a subset of the plot points. We pick $M = 2N + 1$ function values uniformly on $[0, T]$, so that $t_k = kT/M$, $0 \leq k < M$. We set

$$\mathbf{f} = (f(0 \cdot T/M), f(1 \cdot T/M), \dots, f((M-1)T/M)).$$

```
N=20
M=2*N+1
tsp = np.linspace(0,T-T/M,M).reshape((-1,1))
```

We will consider the square wave defined by

$$f_s(t) = \begin{cases} 1 & \text{for } 0 \leq t < T/2 \\ -1 & \text{for } T/2 \leq t < T. \end{cases}$$

The following function returns samples of the square wave at concrete instances in time in one period.

```
def square_wave(t, T):
    return 2*(t < T/2).astype(int) - 1
```

We obtain the samples of the square wave at the interpolation points and the plot points as follows.

```
fsp = square_wave(tsp, T) # samples from one period
f = square_wave(t, T) # plot points
```

Let us now use the $M = 2N + 1$ trigonometric functions

$$\{1\} \cup \{\cos(2\pi nt/T), \sin(2\pi nt/T)\}_{n=1}^N$$

as a basis for interpolation. Since

$$\begin{aligned} e^{it} &= \cos t + i \sin t & e^{-it} &= \cos t - i \sin t \\ \cos t &= \frac{1}{2}(e^{it} + e^{-it}) & \sin t &= \frac{1}{2i}(e^{it} - e^{-it}) \end{aligned}$$

we may as well use

$$\{e^{2\pi int/T}\}_{n=-N}^N \quad (10.21)$$

as a basis for interpolation instead. It turns out that these functions, which are complex, will simplify things a lot. We have the following result:

Theorem 10.2. *The unique function interpolating f at the points $\{kT/M\}_{k=0}^{M-1}$ using the basis from Equation 10.21 is*

$$p(t) = \sum_{n=-N}^N c_n e^{2\pi int/T}, \quad (10.22)$$

where the c_n can be found from

$$(c_0, \dots, c_N, c_{-N}, c_{-1}) = \frac{1}{M} F_M \mathbf{f}, \quad (10.23)$$

where F_M is the DFT matrix from Example 2.6. Moreover, $p(t)$ is a real function.

The proof is a bit technical, so we defer it to the end of the section. Let us first consider how this result can be implemented. The theorem says that the coefficients of the interpolating function can be obtained as

```
c = np.fft.fft(fsp, axis = 0)/M
```

We want the components in $\mathbf{c}$ to come in the same order as $(c_{-N}, \dots, c_{-1}, c_0, \dots, c_N)$, so we swap the two halves of this vector:

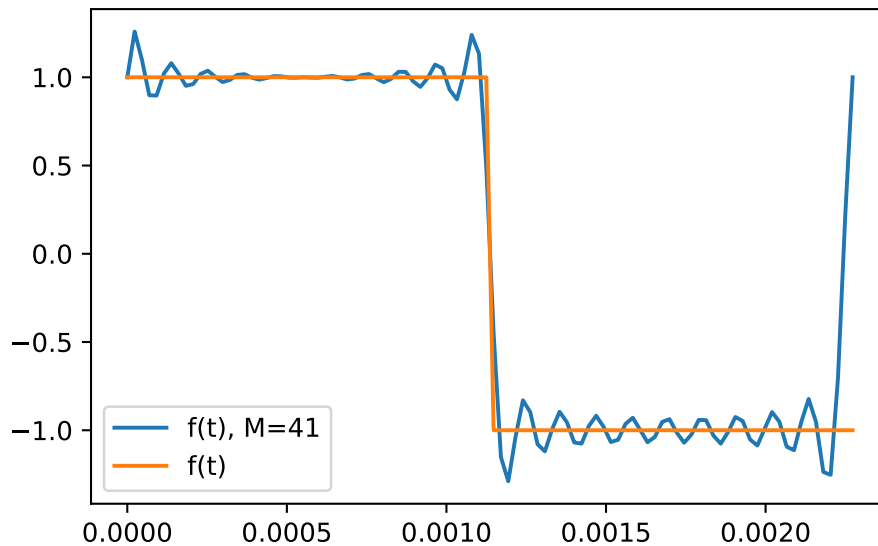
```
c = np.block([ c[(N+1):, :] , [c[: (N+1)], :] ])
```

Equation 10.22 can now be written as a matrix-vector product, and we obtain plot points of p with the following implementation.

```
n = np.arange(-N, N+1).reshape((1, -1))
p = np.real( np.exp(2*np.pi*1j*n*t/T) @ c )
```

We can plot this together with the square wave itself as follows.

```
plt.plot( t, p)
plt.plot( t, f)
plt.legend([f"f(t), M={M}", 'f(t) '])
```



In order to listen to a set of samples as a sound we need the following function

```
def audiowrite(filename, f, freq_s):
    ofile = wave.open(filename, 'w')
    ofile.setsampwidth(2)
    ofile.setframerate(freq_s)
    ofile.setnchannels(1)
    max_amplitude = 2**15-1
    f=max_amplitude*f
    f=f.astype(np.uint16)
    ofile.writeframesraw(f.astype(np.uint16).tobytes())
    ofile.close()
```

We now generate sounds corresponding to a sine, a square wave and its interpolant (we set the sounds to last three seconds).

```
numsec=3

t = np.linspace(0,numsec,samples_per_period*numsec*freq)
sine = np.sin(2*np.pi*freq*t)

f = f.reshape(-1)
f = np.tile(f, numsec*freq)
```

```
p = p.reshape(-1)
p = np.tile(p, numsec*freq)
p = p/np.abs(p).max()
```

The periods of the square wave and its interpolant were tiled so many times that it lasts the given number of seconds. The sound samples of the interpolant were scaled so that they are between -1 and 1, as is required for sound samples on a computer. This was necessary since the plot showed that the values of the interpolant exceeded 1, although the square wave did not.

Finally we write sound data to file:

```
audiowrite('sine.wav', sine, freq_s)
audiowrite('square.wav', f, freq_s)
audiowrite('interpolant.wav', p, freq_s)
```

You can now listen to the three generated files in your favourite audio player.

Proof. (of Theorem 10.2). The interpolation conditions are

$$\sum_{n=-N}^N c_n e^{2\pi i n k / M} = f(kT/M), \quad (10.24)$$

where $0 \leq k < N$. This corresponds to the matrix equation

$$\left(\begin{array}{ccc|ccc} e^{2\pi i 0(-N)/M} & \dots & e^{2\pi i 0(-1)/M} & e^{2\pi i 0 0/M} & \dots & e^{2\pi i 0 N/M} \\ e^{2\pi i 1(-N)/M} & \dots & e^{2\pi i 1(-1)/M} & e^{2\pi i 1 0/M} & \dots & e^{2\pi i 1 N/M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ e^{\frac{2\pi i (M-1)(-N)}{M}} & \dots & e^{\frac{2\pi i (M-1)(-1)}{M}} & e^{\frac{2\pi i (M-1) 0}{M}} & \dots & e^{\frac{2\pi i (M-1) N}{M}} \end{array} \right) \begin{pmatrix} c_{-N} \\ \vdots \\ c_{-1} \\ c_0 \\ \vdots \\ c_N \end{pmatrix} = \mathbf{f}.$$

Using Equation 2.11 we get

$$\left(\begin{array}{ccc|ccc} e^{2\pi i 0 0/M} & \dots & e^{2\pi i 0 N/M} & e^{2\pi i 0 \dots (-N)/M} & \dots & e^{2\pi i 0(-1)/M} \\ e^{2\pi i 1 0/M} & \dots & e^{2\pi i 1 N/M} & e^{2\pi i 1(-N)/M} & \dots & e^{2\pi i 1(-1)/M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ e^{\frac{2\pi i (M-1) 0}{M}} & \dots & e^{\frac{2\pi i (M-1) N}{M}} & e^{\frac{2\pi i (M-1)(-N)}{M}} & \dots & e^{\frac{2\pi i (M-1)(-1)}{M}} \end{array} \right) \begin{pmatrix} c_0 \\ \vdots \\ c_N \\ c_{-N} \\ \vdots \\ c_{-1} \end{pmatrix} = \mathbf{f}.$$

Using that $e^{2\pi i (-n)k/M} = e^{2\pi i (M-n)k/M}$ we get

$$\left(\begin{array}{ccc|ccc} e^{2\pi i 0 0/M} & \dots & e^{2\pi i 0 N/M} & e^{2\pi i 0 (M-N)/M} & \dots & e^{2\pi i 0 (M-1)/M} \\ e^{2\pi i 1 0/M} & \dots & e^{2\pi i 1 N/M} & e^{2\pi i 1 (M-N)/M} & \dots & e^{2\pi i 1 (M-1)/M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ e^{\frac{2\pi i (M-1) 0}{M}} & \dots & e^{\frac{2\pi i (M-1) N}{M}} & e^{\frac{2\pi i (M-1)(M-N)}{M}} & \dots & e^{\frac{2\pi i (M-1)(M-1)}{M}} \end{array} \right) \begin{pmatrix} c_0 \\ \vdots \\ c_N \\ c_{-N} \\ \vdots \\ c_{-1} \end{pmatrix} = \mathbf{f}.$$

$M = 2N + 1$ implies that $M - N = N + 1$, so that this can be written as

$$\begin{pmatrix} e^{2\pi i 0 \cdot 0/M} & e^{2\pi i 0 \cdot 1/M} & \dots & e^{2\pi i 0(M-1)/M} \\ e^{2\pi i 1 \cdot 0/M} & e^{2\pi i 1 \cdot 1/M} & \dots & e^{2\pi i 1(M-1)/M} \\ \vdots & \vdots & \ddots & \vdots \\ e^{2\pi i (M-1) \cdot 0/M} & e^{2\pi i (M-1) \cdot 1/M} & \dots & e^{2\pi i (M-1)(M-1)/M} \end{pmatrix} \begin{pmatrix} c_0 \\ \vdots \\ c_N \\ c_{-N} \\ \vdots \\ c_{-1} \end{pmatrix} = \mathbf{f}.$$

From Example 4.4 we see that the matrix on the left side is MF_M^{-1} . Multiplying with F_M on both sides we get Equation 10.23.

Finally, let us comment on why $p(t)$ must be real. Since $\mathbf{f}$ is real it follows that $c_{-n} = \overline{c_n}$ (see Exercise 2.13), so that

$$c_n e^{2\pi i n t/T} + c_{-n} e^{2\pi i (-n) t/T} = 2\Re(c_n e^{2\pi i n t/T}).$$

Since also c_0 is real, it follows that Equation 10.22 also evaluates to something real. $\square$

Exercises

Exercise 10.1. In this exercise we will look at code which computes the matrix we need to find the canonical form of the interpolating polynomial. Listing 10.2 computed the Vandermonde matrix with the code

```
p = np.arange(0,n+1).reshape((1,-1))
A = x**p
```

Compare this with the following alternative code:

```
A[:,0] = np.ones((n+1,1))
for k in range(1,n+1):
    A[:,k] = A[:,k-1]*x
```

Which of the two alternatives is to prefer? Justify your answer, and what assumptions you make.

! Solution

If we assume that the computer finds x^n by multiplying in x $n-1$ times, the first code segment will use $n(1 + \dots + n-2) = n(n-1)(n-2)/2$ operations, while the second code segment will use $n^2 - n$ operations, and therefore is to prefer.

It is not for granted, however, that the computer finds x^n by means of $n-1$ multiplications. The computer probably uses a more efficient algorithm. In particular, note that $x^n = e^{n \ln x}$, which the computer can exploit to find the powers, as long as it has efficient algorithms to compute logarithms

and the exponential functions.

Exercise 10.2. In this exercise we will consider another proof of the fact that the interpolating polynomial is unique .

- (a) Suppose that there exist two quadratic polynomials p_1 and p_2 which interpolate f in the three points x_0 , x_1 and x_2 . Consider the difference $p = p_2 - p_1$. What are the values of p in the interpolation points?

! Solution

Since $p_1(x_0) = p_2(x_0) = f(x_0)$, $p_1(x_1) = p_2(x_1) = f(x_1)$, $p_1(x_2) = p_2(x_2) = f(x_2)$, we get that

$$p(x_0) = p_2(x_0) - p_1(x_0) = f(x_0) - f(x_0) = 0$$

$$p(x_1) = p_2(x_1) - p_1(x_1) = f(x_1) - f(x_1) = 0$$

$$p(x_2) = p_2(x_2) - p_1(x_2) = f(x_2) - f(x_2) = 0.$$

The values in the interpolation points are thus 0.

- (b) Use the observation in (a) to show that p_1 and p_2 must be equal.

! Solution

The second degree polynomial p has three distinct zeros, but a second degree polynomial different from 0 can not have more than two zeros. We therefore must have that $p = 0$, so that $p_1 = p_2$, so that the interpolating polynomial is unique.

- (c) Generalise the results from (a) and (b) to polynomials of degree n .

! Solution

If we define $p = p_1 - p_2$ as above, we see as before that $p(x_0) = p(x_1) = \dots = p(x_n) = 0$, so that p is a polynomial of degree n with $n + 1$ distinct zeros. From the fundamental theorem of algebra (i.e., that a polynomial of degrees n can't have more than n roots) it follows that $p = 0$, so that $p_1 = p_2$, so that the interpolating polynomial is unique also in this case.

Exercise 10.3 (Horner's rule). A direct computation of Equation 10.6 in a point x requires n multiplications and n additions, as well as computation of the powers ($n - 1$ multiplikasjoner). In total this is $3n - 1$ operations. Let us rewrite Equation 10.6 to

$$p_n(x) = c_0 + x(c_1 + c_2x + c_3x^2 + \dots c_nx^{n-1}). \quad (10.25)$$

Inside the parentheses on the right side here there is a polynomial of degree $n - 1$ (which we call p_{n-1}), which in the same way can be expressed in terms of a polynomial of degree $n - 2$ (p_{n-2}), and so on. We have that

$$\begin{array}{rcl} p_n(x) & = & c_0 + xp_{n-1}(x) \\ p_{n-1}(x) & = & c_1 + xp_{n-2}(x) \\ p_{n-2}(x) & = & c_2 + xp_{n-3}(x) \\ \vdots & & \vdots \\ p_1(x) & = & c_{n-1} + xp_0(x) \\ p_0(x) & = & c_n \end{array}$$

To compute $p_n(x)$ we start from the last equation, and continue upwards to the first equation. This algorithm is also called *Horner's rule*.

- (a) Write an implementation of Horner's rule.

! Solution

If we denote the computation of the right hand side by s , and start with the last equation, the steps become

$$\begin{array}{rcl} s & = & c_n \\ s & = & c_{n-1} + xs \\ \vdots & & \vdots \\ s & = & c_2 + xs \\ s & = & c_1 + xs \\ s & = & c_0 + xs. \end{array}$$

After this we get that $s = p_n(x)$. The code can look as follows.

```
s = c[n]
for k in range(n-1,-1,-1):
    s = c[k] + x*s
```

- (b) How many arithmetic operations are required by Horner's rule?

! Solution

In each iteration of the loop an addition and a multiplication are computed. The total number of operations is therefore $2n$, which is an improvement from $3n - 1$. Horner's rule in fact turns out to be optimal computationally when it comes to computing values of a polynomial. It is also optimal when it comes to memory usage - one only needs the variable s , in addition to x and $\mathbf{c}$. In order to integrate Horner's rule in Listing 10.2, you should replace $\mathbf{Vp}*\mathbf{c}$ with what you got from the code above.

- (c) Explain how you can adapt Horner's rule to the Newton form in Equation 10.9.

! Solution

Equation 10.9 is now rewritten to

$$\begin{aligned}
 p_n(x) &= c_0 + (x - x_0)p_{n-1}(x) \\
 p_{n-1}(x) &= c_1 + (x - x_1)p_{n-2}(x) \\
 &\vdots \\
 p_1(x) &= c_{n-1} + (x - x_{n-1})p_0(x) \\
 p_0(x) &= c_n
 \end{aligned}$$

The number of operations will be the same as above, with the exception that we now also must compute the differences $x - x_0, x - x_1, \dots, x - x_{n-1}$.

Exercise 10.4. We interpolate the function $f(x) = x^2$ with a polynomial p_3 of degree ≤ 3 , in the points 0, 1, 2, and 3. What is $p_3(4)$?

! Solution

16

Exercise 10.5 (Lagrange form).

We have the data

x	0	1	3	4
$f(x)$	1	0	2	1

(a) Write the cubic polynomial which interpolate in these points as

$$\begin{aligned}
 p_3(x) &= c_0(x-1)(x-3)(x-4) + c_1x(x-3)(x-4) \\
 &\quad + c_2x(x-1)(x-4) + c_3x(x-1)(x-3),
 \end{aligned}$$

and find the coefficients from the interpolation conditions. Dette This is called the *Lagrange form* of the interpolating polynomial.

! Solution

If we set $x = 0$ we get $1 = c_0(-1)(-3)(-4)$, so that $c_0 = -1/12$.

If we set $x = 1$ we get $0 = c_1(1)(1-3)(1-4)$, so that $c_1 = 0$.

If we set $x = 3$ we get $2 = c_2(3)(3-1)(3-4)$, so that $c_2 = -1/3$.

If we set $x = 4$ we get $1 = c_3(4)(4-1)(4-3)$, so that $c_3 = 1/12$.

We thus get

$$p_3(x) = -\frac{(x-1)(x-3)(x-4)}{12} - \frac{x(x-1)(x-4)}{3} + \frac{x(x-1)(x-3)}{12}.$$

(b) Find the Newton form of the interpolating polynomial .

! Solution

If we set $p_3(x) = c_0 + c_1x + c_2x(x-1) + c_3x(x-1)(x-3)$, we get the equations

$$\begin{aligned} 1 &= c_0 \\ 0 &= c_0 + c_1 \\ 2 &= c_0 + 3c_1 + 6c_2 \\ 1 &= c_0 + 4c_1 + 12c_2 + 12c_3. \end{aligned}$$

The first two equations give that $c_0 = 1$ and $c_1 = -1$. Inserting this in the third equation gives that $c_2 = (2 - 1 + 3)/6 = 2/3$. Inserting this in the fourth equation gives that $c_3 = (1 - 1 + 4 - 8)/12 = -1/3$. This gives

$$p_3(x) = 1 - x + \frac{2}{3}x(x-1) - \frac{1}{3}x(x-1)(x-3).$$

(c) Verify that the solutions you found in (a) and (b) are equal.

Exercise 10.6 (Interpolation with a set of data).

(a) We have the data

x	0	1	2
$f(x)$	2	1	0

These are points on the line $y = 2 - x$. Find the Newton form of the quadratic interpolating polynomial and compare. What is the difference?

! Solution

The Newton form of the interpolating second degree polynomial is

$$p_2(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1).$$

We have that

$$\begin{aligned} f[x_0, x_1] &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{1 - 2}{1 - 0} = -1 \\ f[x_1, x_2] &= \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{0 - 1}{2 - 1} = -1 \\ f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{-1 - (-1)}{2 - 0} = 0. \end{aligned}$$

If we insert this in $p_2(x)$ we get that $p_2(x) = 2 - (x - x_0) = 2 - x$.

(b) Suppose we interpolate f with polynomials of degree $\leq n$ in the points

$x_0, \dots, x_n$. Show that if f is a polynomial of degree $\leq n$, then the interpolant p_n will be identical to f . Why does this explain the result in (a)?

! Solution

Both f and p_n are interpolating polynomials of degree $\leq n$, and then we know from Exercise 10.2 that they must be equal. In (a) the function already was a second degree polynomial, so that it must be equal to its interpolant as well. Therefore we obtained that p_n was equal to the function itself.

Exercise 10.7. Suppose we have the data

$$(0, y_0), \quad (1, y_1), \quad (2, y_2), \quad (3, y_3) \quad (10.26)$$

where $y_i = f(i)$ are values of an unknown function f . In this exercise we will find expressions for approximations to f in different points by using cubic interpolation.

- (a) Find the line p_1 which interpolates the two midpoints in Equation 10.26, and use $p_1(3/2)$ as an approximation to $f(3/2)$. Show that

$$p_1(3/2) = \frac{1}{2}(y_1 + y_2).$$

! Solution

Linear interpolant p_1 :

$$p_1(x) = y_1 + (y_2 - y_1)(x - 1).$$

This gives $p_1(3/2) = y_1 + \frac{1}{2}(y_2 - y_1) = \frac{1}{2}(y_1 + y_2)$.

- (b) Find the cubic polynomial p_3 which interpolates the data Equation 10.26, and use $p_3(3/2)$ as an approximation to $f(3/2)$. Show that

$$p_3(3/2) = \frac{-y_0 + 9y_1 + 9y_2 - y_3}{16}.$$

! Solution

The Newton form for the polynomial interpolating in these four points

can be found by solving

$$\begin{aligned}y_0 &= c_0 \\y_1 &= c_0 + c_1 \\y_2 &= c_0 + 2c_1 + 2c_2 \\y_3 &= c_0 + 3c_1 + 6c_2 + 6c_3.\end{aligned}$$

This first gives $c_0 = y_0$, then $c_1 = y_1 - y_0$, then

$$\begin{aligned}c_2 &= \frac{y_2 - y_0 - 2(y_1 - y_0)}{2} = \frac{y_2 - 2y_1 + y_0}{2} \\c_3 &= \frac{y_3 - y_0 - 3(y_1 - y_0) - 6(y_2 - 2y_1 + y_0)/2}{6} = \frac{y_3 - 3y_2 + 3y_1 - y_0}{6},\end{aligned}$$

so that the interpolating cubic polynomial becomes

$$p_3(x) = y_0 + (y_1 - y_0)x + \frac{y_2 - 2y_1 + y_0}{2}x(x-1) + \frac{y_3 - 3y_2 + 3y_1 - y_0}{6}x(x-1)(x-2).$$

This gives finally

$$\begin{aligned}p_3(3/2) &= y_0 + (y_1 - y_0)\frac{3}{2} + \frac{y_2 - 2y_1 + y_0}{2}\frac{3}{4} - \frac{y_3 - 3y_2 + 3y_1 - y_0}{6}\frac{3}{8} \\&= \frac{16y_0 + 24y_1 - 24y_0 + 6y_2 - 12y_1 + 6y_0 - y_3 + 3y_2 - 3y_1 + y_0}{16} \\&= \frac{-y_0 + 9y_1 + 9y_2 - y_3}{16}.\end{aligned}$$

Exercise 10.8 (Taylor series and interpolation). Our basis for interpolation has been the knowledge of function values $f(x_0), \dots, f(x_m)$. *Taylor polynomials* can be seen as an alternative to this, with this knowledge replaced by $f(a), f'(a), \dots, f^{(n)}(a)$. Let the basis functions $\phi_i(x)$ be

$$\{1, x - a, (x - a)^2, \dots, (x - a)^n\}$$

The polynomial we seek is then

$$p(x) = c_0 + c_1(x - a) + \dots + c_n(x - a)^n,$$

but now the interpolation conditions instead take the form

$$p(a) = f(a) \quad p'(a) = f'(a) \quad p''(a) = f''(a) \quad \dots \quad p^{(n)}(a) = f^{(n)}(a). \quad (10.27)$$

Show that $c_k = f^{(k)}(a)/k!$, so that we obtain the Taylor polynomial

$$f(a) + f'(a)(x - a)/1! + f''(a)(x - a)^2/2! + \dots + f^{(n)}(a)(x - a)^n/n!$$

! Solution

Since the k 'th derivative of $(x - a)^k$ is $k!$, and all its other derivatives are zero, we have that $p^{(k)}(a) = c_k k!$. Equation 10.27 thus takes the form

$$c_0 = f(a) \quad c_1 1! = f'(a) \quad c_2 2! = f''(a) \quad \cdots \quad c_n n! = f^{(n)}(a),$$

so that $c_k = f^{(k)}(a)/k!$ for all k .

Exercise 10.9. The previous exercise shows that the interpolation conditions can be values of derivatives, rather than a set of function values. This exercise will show that we also can have a mix of the two.

- (a) Suppose that f is differentiable in x_1 , and that x_0, x_1 , and x_2 are distinct. Find out if there exists a unique cubic polynomial p_3 so that

$$p_3(x_0) = f(x_0), \quad p_3(x_1) = f(x_1), \quad p'_3(x_1) = f'(x_1), \quad p_3(x_2) = f(x_2).$$

! Solution

Let us try with a cubic polynomial on the form

$$p_3(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + c_3(x - x_0)(x - x_1)^2.$$

We have that $p'_3(x) = c_1 + c_2(2x - x_0 - x_1) + c_3(x - x_0)(3x - 2x_0 - x_1)$, and the interpolation conditions give

$$\begin{aligned} f(x_0) &= c_0 \\ f(x_1) &= c_0 + c_1(x_1 - x_0) \\ f'(x_1) &= c_1 + c_2(x_1 - x_0) \\ f(x_2) &= c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) + c_3(x_2 - x_0)(x_2 - x_1)^2 \end{aligned}$$

This gives the matrix equation

$$\begin{pmatrix} f(x_0) \\ f(x_1) \\ f'(x_1) \\ f(x_2) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & x_1 - x_0 & 0 & 0 \\ 0 & 1 & x_1 - x_0 & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & (x_2 - x_0)(x_2 - x_1)^2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

Again the coefficient matrix here is lower triangular with non-zeros on the diagonal, so that it is invertible. It follows that the coefficients are unique, so that the polynomial is unique.

- (b) Can you generalise this result to cases where some of the derivatives in the interpolation points also are known? Can you relate this to Taylor polynomials?

! Solution

Yes, it is possible to generalise this. Suppose that $f(x_i)$, $f'(x_i)$, ..., $f^{(k-1)}(x_i)$ are known. We have contribution from $r_n(x)(x-x_i)$, $r_n(x)(x-x_i)^2$, ..., $r_n(x)(x-x_i)^k$, $r_n(x)(x-x_i)^k s_n(x)$, where r_n represents the contribution from $x_0, \dots, x_{i-1}$, s_n the contribution from $x_{i+1}, \dots, x_n$.

The value of the derivative of $r_n(x)(x-x_i)$ in x_i is $r_n(x_i)$. The derivative of $r_n(x)(x-x_i)^2$ and so in x_i is 0.

The value of derivative number $k-1$ of $r_n(x)(x-x_i)^{k-1}$ is $(k-1)!r_n(x)$, which is $\neq 0$.

Again we get a triangular system with nonzeros on the diagonal.

Exercise 10.10. We approximate $f(x) = e^x$ with an interpolating polynomial p_n in the points

$$0, 1/n, 2/n, \dots, (n-1)/n, 1.$$

How large must n be in order for the error $|f(x) - p_n(x)|$ to be less than 0.001 for all $x \in [0, 1]$?

! Solution

We have that $f^{(n+1)}(x) = e^x$, so that $M_{n+1} = e$. Equation 10.19 then gives that, for all $x \in [0, 1]$, we have

$$|f(x) - p_n(x)| \leq \frac{e \cdot (1-0)^{n+1}}{(n+1)!} = \frac{e}{(n+1)!}$$

We thus demand that $\frac{e}{(n+1)!} \leq 10^{-3}$, i.e., that $(n+1)! \geq 1000e$. This can be achieved by choosing $n \geq 6$.

Exercise 10.11. Show that $\phi(t)$ in the proof for Equation 10.19 fulfills

$$\phi(x) = \phi(x_0) = \phi(x_1) = \dots = \phi(x_n) = 0$$

Exercise 10.12. We have seen that

$$Q_n(x) = (x-x_0)(x-x_1)\dots(x-x_n)$$

is central in the Newton form, and in the proof of Equation 10.19. In the next chapters we will differentiate and integrate this interpolating polynomial, and we will therefore need to differentiate and integrate Q_n as well.

- (a) Show that $Q^{(n+1)}/x = (n+1)!$
- (b) A special case we will consider is when the x_i 's are a uniform partition, i.e., $x_i - x_{i-1} = h$ for all i . Compute $\int_{x_0}^{x_2} Q_1(x) dx$ and $\int_{x_0}^{x_3} Q_2(x) dx$ when the partition is uniform.

 Hint

Integration by parts will be of help here.

 Solution

We get

$$\begin{aligned}\int_{x_0}^{x_2} (x - x_0)(x - x_1) dx &= \left[\frac{1}{2}(x - x_0)^2(x - x_1) \right]_{x_0}^{x_2} - \frac{1}{2} \int_{x_0}^{x_2} (x - x_0)^2 dx \\ &= 2h^3 - \frac{8}{6}h^3 = \frac{2}{3}h^3.\end{aligned}$$

and

$$\begin{aligned}\int_{x_0}^{x_3} (x - x_0)(x - x_1)(x - x_2) dx &= \left[\frac{1}{2}(x - x_0)^2(x - x_1)(x - x_2) \right]_{x_0}^{x_3} - \frac{1}{2} \int_{x_0}^{x_3} (x - x_0)^2((x - x_1) + (x - x_2)) dx \\ &= 9h^4 - \frac{1}{6}[(x - x_0)^3((x - x_1) + (x - x_2))]_{x_0}^{x_3} + \frac{1}{6} \int_{x_0}^{x_3} 2(x - x_0)^3 dx \\ &= 9h^4 - \frac{27}{2}h^4 + \frac{1}{24}[(x - x_0)^4]_{x_0}^{x_3} \\ &= -\frac{9}{2}h^4 + \frac{27}{8}h^4 = -\frac{9}{8}h^4\end{aligned}$$

Exercise 10.13. Fill out the code below so that it generalises the code developed in Section 10.4.

```
import numpy as np
import matplotlib.pyplot as plt

def plot_trig_approx(f, Nvals):
    ...

def triangle_wave(tvals, T):
    return np.abs(t - T/2)*4/T - 1

plt.subplots(1,1)
plot_trig_approx(triangle_wave, [5,10,20])
plt.legend()
plt.show()
```

The input should be a function `func`, and a vector `Nvals` containing different

values for N . The function and all the interpolants should all be plotted all together. Run the code for the listed code for the *triangle wave*, and compare the resulting plot with that of the square wave

! Solution

The code can be as follows

```
import numpy as np
import matplotlib.pyplot as plt

def plot_trig_approx(func, Nvals):
    freq_s = 44100
    freq = 440
    T = 1/freq
    samples_per_period = int(freq_s/freq)
    t = np.linspace(0,T,samples_per_period).reshape((-1,1))
    f = func(t, T) # plot points
    plt.plot( t, f, '-k', label = 'f(t)')

    for N in Nvals:
        M=2*N+1
        tsp = np.linspace(0,T-T/M,M).reshape((-1,1))
        fsp = func(tsp, T) # samples from one period

        c = np.fft.fft(fsp,axis=0)/M
        c = np.block([ [c[(N+1):,:]] , [c[: (N+1),:]] ])
        n = np.arange(-N,N+1).reshape((1,-1))
        p = np.real(np.exp(2*np.pi*1j*n*t/T)*c)
        plt.plot(t, p, label=f"p(t), M={M}.")

def triangle_wave(t, T):
    return np.abs(t - T/2)*4/T - 1

plt.subplots(1,1)
plot_trig_approx(triangle_wave,[5,10,20])
plt.legend()
plt.show()
```

The approximations seem closer to that of the triangle wave, when compared with the case of the square wave.

Chapter 11

Numerical differentiation

Differentiation is a basic operation in mathematics, with many applications. It is therefore important to have good methods for computing and applying derivatives. You have probably learnt the basic rules of differentiation in school - symbolic methods suitable for pen and paper calculations. These are of limited value on a computer, since many programming languages do not support symbolic computation.

Another thing is that a function may only be known in an isolated set of points, such as measurements of the position of a car with a GPS unit. Suppose we want to find the speed of the car. If the position of the car is known at all time instances (as a mathematical function), then this can be found by means of symbolic differentiation. But if the position is known for some points in time we have to appeal to other methods.

The solution is numerical methods which approximate the derivative by means of suitable combinations of known function values. Let us start by formulating this as a general algorithm: Suppose f is known at the points

$$\dots, a - 2h, a - h, a, a + h, a + 2h, \dots$$

(so, h is the “resolution” for known values of f).

1. Pick a subset of $n + 1$ of these points near a , and write these as

$$a + m_0h, a + m_1h, a + m_2h, \dots, a + m_nh, \quad (11.1)$$

where $m_0, \dots, m_n$ are integers.

2. Find the polynomial $p_n(x)$ interpolating f in the $n + 1$ points in Equation 11.1.
3. Use $p'_n(a)$ as an approximation to $f'(a)$.

We will also call this the *point method* based on the points in Equation 11.1. Point methods can be used for higher order derivatives as well, by using $p_n^{(k)}(a)$

as an approximation to $f^{(k)}(a)$. We define the *truncation error* in this approximation as

$$E_k(f; a, h) = f^{(k)}(a) - p_n^{(k)}(a),$$

i.e., the error resulting from differentiating the interpolating polynomial rather than the function itself. The goal is to get this error as small as possible.

The simplest point methods base themselves on two known function values only, and in Section 11.1 we will see how this has a natural relation to the definition of the derivative. After this we will see how point methods with more points can produce better approximations. The methods for numerical integration in Chapter 12 will be derived and analysed the same way as the differentiation methods in this chapter.

In this chapter we will write $M_k = \max_x |f^{(k)}(x)|$ (when f is k times differentiable), where the maximum is taken over an interval that should be clear from the context. In the notation p_n and $E_k(f; a, h)$ it should be clear from the context what the numbers m_i are, even if they are not part of the notation.

11.1 The Newton quotient

The derivative of f in a point a is defined as

$$f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}. \quad (11.2)$$

We will assume that f is differentiable in a , i.e., this limit exists. For a fixed $h \neq 0$ the approximation

$$\frac{f(a+h) - f(a)}{h} \quad (11.3)$$

to $f'(a)$ is called the *Newton difference quotient*, or simply the *Newton quotient*.

If we use the point method based on the points a and $a+h$ we get the line p_1 through $(a, f(a))$ and $(a+h, f(a+h))$, i.e., the secant through these. This can be written

$$p_1(x) = f(a) + \frac{f(a+h) - f(a)}{h}(x-a).$$

The derivative of p_1 , i.e., the slope of the secant, is precisely the right hand side in Equation 11.3. The Newton quotient thus coincides with the point method based on a and $a+h$ (see Figure 11.1).

The truncation error for the Newton quotient is

$$E_1(f; a, h) = f'(a) - \frac{f(a+h) - f(a)}{h}.$$

Let us take a closer look at this. From Taylor's formula with remainder (Equation 10.18), we get that

$$f(a+h) = f(a) + hf'(a) + \frac{h^2}{2}f''(\xi_h), \quad (11.4)$$

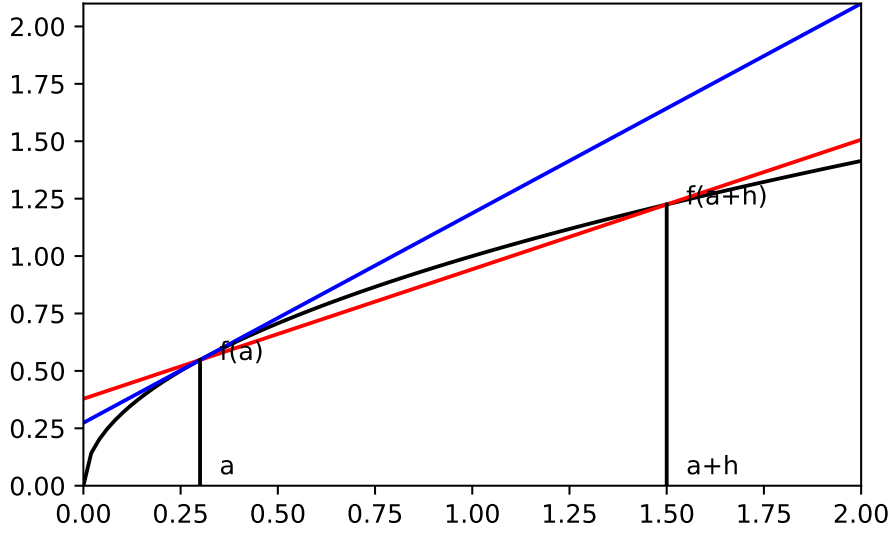


Figure 11.1: The secant (in red) of a function (in black) in the points a and $a + h$, as well as the tangent in a (blue).

where $\xi_h \in [a, a + h]$. Reorganising this we get the following expression for the truncation error:

$$f'(a) - \frac{f(a+h) - f(a)}{h} = -\frac{h}{2}f''(\xi_h). \quad (11.5)$$

If the second derivative is continuous, the extreme value theorem says that the function $|f''|$ has a maximum on the interval $[a, a + h]$, which we denote by M_2 . From this we get the following error estimate.

Proposition 11.1. *Suppose f has continuous second order derivatives near a . Then the truncation error of the Newton quotient satisfies*

$$|E_1(f; a, h)| \leq M_2 \frac{h}{2}, \quad (11.6)$$

where $M_2 = \max_x |f''(x)|$.

When h is small, $\max_{x \in [a, a+h]} |f''(x)|$ is close to $|f''(a)|$, so that the actual error is close to $|f''(a)| \frac{h}{2}$. Consider the function $f(x) = \sin x$ at the point $a = 0.5$. We have $f''(x) = -\sin(x)$, so that for small h the error in the Newton quotient is about $\frac{\sin 0.5}{2}h \approx 0.24h$.

It is common also to use the worse bound $M_2 = \max_{x \in \mathbb{R}} |f''(x)| = 1$, rather than $\max_{x \in [a, a+h]} |f''(x)|$. This gives the bound $|E_1(f; a, h)| \leq 0.5h$ for the Newton quotient, which is an *overestimate* of the error with a factor of about 2. In error estimates it is common to miss a bit, and “exaggerate” how large the

error can become. The central point here is that we have estimated something unknown (the error E_1) with something computable (the right hand side in Equation 11.6).

The error e in our numerical methods will usually be on the form $e = Kh^k$, where h is the precision in the method, and the constant K and the exponent k are unknown. We then say that y is *polynomial* in x , with constant K and exponent k . A *loglog plot* can help us find out if there are such relations between h and e , and if so help us estimate the concrete values of K and k . If we plot h against e it is not so easy to grasp K and k from the plot, since the curves ascend quickly. A trick can then be to take logarithms, which gives

$$\log e = \log K + k \log h.$$

In other words, if we plot the logarithm of the error against the logarithm of h , we obtain a line where

- the slope of the curve gives us the exponent k , and
- the intersection with the y -axis is in $\log K$, so that we can find K from this intersection.

We write `plt.loglog(h,e)` to plot $\log h$ against $\log e$. `plt.loglog` has a syntax similar to that of the `plt.plot`-command.

Figure 11.2 shows a loglog plot of h against the truncation error, together with the upper bound $M_2h/2$ for the error. Here we set $M_2 = 1$, so that we again produce an overestimate. A better estimate could be obtained by finding $\max_x |f''(x)|$ over a small interval near a instead, but if you do this you will see that the two curves lie on top of one-another.

```
f = np.sin
fd = np.cos
a = 0.5
h = 10**(-np.arange(0,6,0.1).reshape(-1,1))
efh = fd(a) - (f(a+h)-f(a))/h
plt.loglog(h, np.abs(efh), 'k.', h, h/2, 'r')
plt.legend(['Error', 'Bound on error'])
```

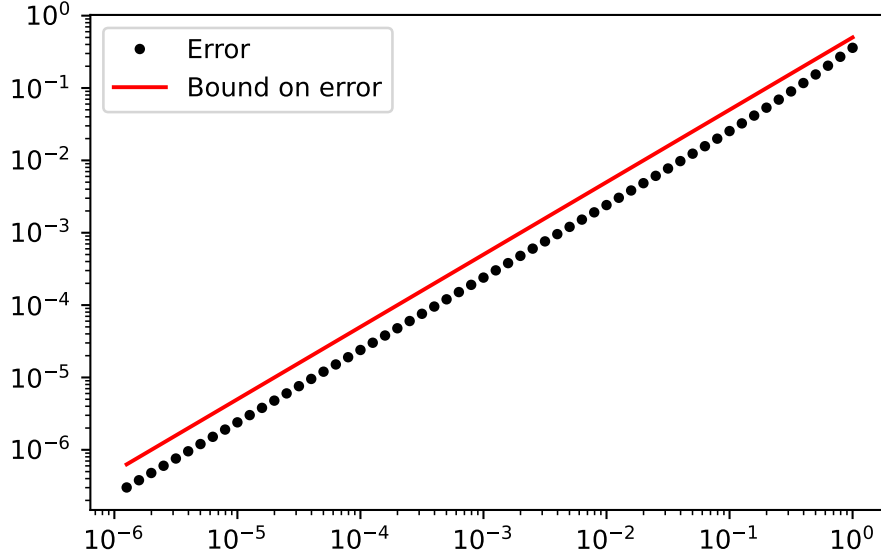


Figure 11.2: The truncation error of the Newton quotient when $f(x) = \sin x$ at the point $x = 0.5$, for $h \in [10^{-6}, 10^0]$. The plot is logarithmic.

The plot looks like a line with slope 1. Since the plot is logarithmic, this agrees with the approximation $|E_1(f; a, h)| \approx \frac{h}{2} |f''(a)|$ for the truncation error, since we obtain by taking logarithms in this that

$$\begin{aligned} \log |E_1(f; a, h)| &\approx \log h + \log(0.5) + \log(\sin(0.5)) \\ &\approx \log h - 0.6203. \end{aligned}$$

This agrees well with the point where the line hits the y -axis. ♣

11.2 The symmetric Newton quotient

The Newton quotient was based on the points $f(a)$ and $f(a+h)$. Let us instead consider the points $f(a)$, $f(a-h)$, and $f(a+h)$. Using the Newton form of the interpolating polynomial we get from Equation 10.13 and Equation 10.14 that

$$\begin{aligned} p_2(x) = f(a-h) &+ \frac{f(a) - f(a-h)}{h}(x - (a-h)) \\ &+ \frac{f(a+h) - 2f(a) + f(a-h)}{2h^2}(x - (a-h))(x - a). \end{aligned} \quad (11.7)$$

After some computation we get

$$p_2'(a) = \frac{f(a+h) - f(a-h)}{2h} \quad (11.8)$$

We leave the details in Equation 11.7 and Equation 11.8 to Exercise 11.4. Equation 11.8 gives the slope of the secant through the points $a - h$ and $a + h$, and is called the *symmetric Newton quotient*. This is illustrated in Figure 11.3,

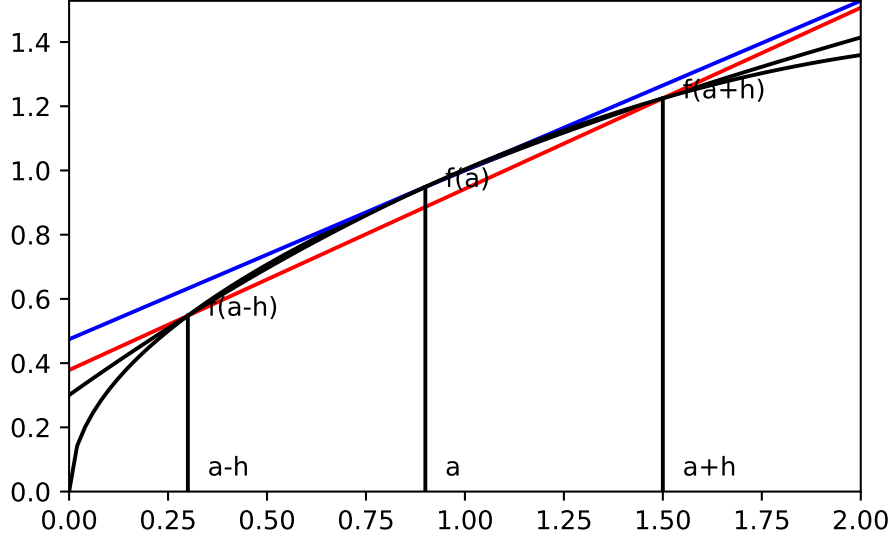


Figure 11.3: The secant based on $a - h$ and $a + h$ (in red) of a function f (in black), as well as the tangent in a (in blue). The parabola interpolating f in $a - h$, a , and $a + h$ are also shown (in black)

The truncation error of the symmetric Newton quotient can be analysed in the same way as the Newton quotient. We now evaluate the Taylor polynomial at two points, and include a quadratic term instead:

$$\begin{aligned} f(a+h) &= f(a) + f'(a)h + \frac{1}{2}f''(a)h^2 + \frac{1}{6}f'''(c_1)h^3 \\ f(a-h) &= f(a) - f'(a)h + \frac{1}{2}f''(a)h^2 - \frac{1}{6}f'''(c_2)h^3, \end{aligned}$$

where $c_1 \in [a, a+h]$, $c_2 \in [a-h, a]$. Subtracting these we get

$$\frac{f(a+h) - f(a-h)}{2h} - f'(a) = \frac{1}{12}h^2(f'''(c_1) + f'''(c_2)).$$

If we take the absolute value of both sides and use the triangle inequality we get that

$$|E_1(f; a, h)| = \left| \frac{1}{12}h^2(f'''(c_1) + f'''(c_2)) \right| \leq \frac{h^2}{12}M_3 + \frac{h^2}{12}M_3 = M_3 \frac{h^2}{6}.$$

The approximation will in particular be exact for quadratic polynomials, since these have a third derivative equal to 0 (so that $M_3 = 0$). The truncation error is therefore 0 for these. The approximation is not exact for cubic polynomials.

11.3 A point method for the second derivative

To derive a point method for the second derivative, we will use the same points as for the symmetric Newton quotient, This gives the same interpolating polynomial as in Equation 11.7. After some computation we find that

$$p_2''(a) = \frac{f(a+h) - 2f(a) + f(a-h)}{h^2} \quad (11.9)$$

As above we can find the truncation error by evaluating the Taylor polynomial in the same points, but we now also include a cubic term,

$$\begin{aligned} f(a+h) &= f(a) + f'(a)h + \frac{1}{2}f''(a)h^2 + \frac{1}{6}f'''(a)h^3 + \frac{1}{24}f^{(iv)}(c_1)h^4 \\ f(a-h) &= f(a) - f'(a)h + \frac{1}{2}f''(a)h^2 - \frac{1}{6}f'''(a)h^3 + \frac{1}{24}f^{(iv)}(c_2)h^4, \end{aligned}$$

and instead of subtracting the two we now add them together. We then get

$$f(a-h) - 2f(a) + f(a+h) - f''(a)h^2 = \frac{1}{24}f^{(iv)}(c_1)h^4 + \frac{1}{24}f^{(iv)}(c_2)h^4.$$

Dividing with h^2 on both sides we get the truncation error

$$|E_2f(;a,h)| = \left| \frac{f(a-h) - 2f(a) + f(a+h)}{h^2} - f''(a) \right|$$

on the left side, and on the right side we get

$$M_4 \frac{h^2}{24} + M_4 \frac{h^2}{24} = M_4 \frac{h^2}{12}.$$

11.4 More general point methods

For point methods with more points one can also do as above - i.e., combining Taylor series evaluated in the known points so that one on the left side gets an expression for the truncation error. We need the following result, which uses linear algebra to derive the general point method based on $f(a+m_0h)$, $f(a+m_1h), \dots, f(a+m_nh)$.

Theorem 11.1. *Suppose we know $n+1$ function values $\{f(a+m_ih)\}_{i=0}^n$, where $m_0 = 0$ and $m_0, m_1, \dots, m_n$ are distinct integers (we assume in particular that $f(a)$ should contribute). Let p_n be the interpolating polynomial of f of degree $\leq n$ in the points $\{a+m_ih\}_{i=0}^n$. Then we have that*

$$\begin{pmatrix} h^1 p_n^{(1)}(a)/1! \\ h^2 p_n^{(2)}(a)/2! \\ \vdots \\ h^n p_n^{(n)}(a)/n! \end{pmatrix} = \begin{pmatrix} m_1 & m_1^2 & \cdots & m_1^n \\ m_2 & m_2^2 & \cdots & m_2^n \\ \vdots & \vdots & \ddots & \vdots \\ m_n & m_n^2 & \cdots & m_n^n \end{pmatrix}^{-1} \begin{pmatrix} f(a+m_1h) - f(a) \\ f(a+m_2h) - f(a) \\ \vdots \\ f(a+m_nh) - f(a) \end{pmatrix}. \quad (11.10)$$

You are not supposed to remember this result! The point is that we from this actually can read out, and write code to code to find the approximations $p_n^{(k)}(a)$ to all the derivatives! We obtain the approximations by solving a system involving a Vandermonde matrix.

Proof. p_n equals its own Taylor series of degree $\leq n$ om a , i.e., that

$$p_n(x) = p_n(a) + p'_n(a)(x-a) + \dots + \frac{p_n^{(n)}(a)}{n!}(x-a)^n.$$

Evaluating this in $x = a + m_i h$ we get that

$$f(a + m_i h) = f(a) + p'_n(a)m_i h + \dots + \frac{p_n^{(n)}(a)}{n!}m_i^n h^n,$$

which can be written

$$f(a + m_i h) - f(a) = (m_i \quad m_i^2 \quad \dots \quad m_i^n) \begin{pmatrix} hp'_n(a) \\ h^2 p_n^{(2)}(a)/2! \\ \vdots \\ h^n p_n^{(n)}(a)/n! \end{pmatrix}$$

for $i = 1, 2, \dots, n$ qv assemble this to

$$\begin{pmatrix} m_1 & m_1^2 & \dots & m_1^n \\ m_2 & m_2^2 & \dots & m_2^n \\ \vdots & \vdots & \ddots & \vdots \\ m_n & m_n h^2 & \dots & m_n^n \end{pmatrix} \begin{pmatrix} h^1 p_n^{(1)}(a)/1! \\ h^2 p_n^{(2)}(a)/2! \\ \vdots \\ h^n p_n^{(n)}(a)/n! \end{pmatrix} = \begin{pmatrix} f(a + m_1 h) - f(a) \\ f(a + m_2 h) - f(a) \\ \vdots \\ f(a + m_n h) - f(a) \end{pmatrix},$$

The matrix on the left side is an invertible Vandermonde matrix. Multiplying with its inverse on both sides gives the needed expression. $\square$

Let us consider some examples.

Example 11.1. Let us consider point methods based on the three points a , $a - h$, and $a + h$. This corresponds to setting $n = 2$, $m_1 = -1$, $m_2 = 1$. Equation 11.10 gives

$$\begin{aligned} \begin{pmatrix} hp'_2(a) \\ h^2 p''_2(a)/2 \end{pmatrix} &= \begin{pmatrix} m_1 & m_1^2 \\ m_2 & m_2^2 \end{pmatrix}^{-1} \begin{pmatrix} f(a-h) - f(a) \\ f(a+h) - f(a) \end{pmatrix} = \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} f(a-h) - f(a) \\ f(a+h) - f(a) \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} f(a-h) - f(a) \\ f(a+h) - f(a) \end{pmatrix} = \begin{pmatrix} \frac{f(a+h) - f(a-h)}{2} \\ \frac{f(a-h) - 2f(a) + f(a+h)}{2} \end{pmatrix}. \end{aligned}$$

This can be rewritten to

$$\begin{aligned} p'_2(a) &= \frac{f(a+h) - f(a-h)}{2h} \\ p''_2(a) &= \frac{f(a-h) - 2f(a) + f(a+h)}{h^2} \end{aligned}$$

We see that this coincides with the point methods from Equation 11.8 and Equation 11.9. ♣

Example 11.2. Let us consider point methods based on the five points $a, a-2h, a-h, a+h$, and $a+2h$. We set $n = 4$, $m_1 = -2$, $m_2 = -1$, $m_3 = 1$, $m_4 = 2$. Equation 11.10 gives

$$\begin{aligned} \begin{pmatrix} h^1 p_4^{(1)}(a)/1! \\ h^2 p_4^{(2)}(a)/2! \\ h^3 p_4^{(3)}(a)/3! \\ h^4 p_4^{(4)}(a)/4! \end{pmatrix} &= \begin{pmatrix} m_1 & m_1^2 & m_1^3 & m_1^4 \\ m_2 & m_2^2 & m_2^3 & m_2^4 \\ m_3 & m_3^2 & m_3^3 & m_3^4 \\ m_4 & m_4^2 & m_4^3 & m_4^4 \end{pmatrix}^{-1} \begin{pmatrix} f(a-2h) - f(a) \\ f(a-h) - f(a) \\ f(a+h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix} \\ &= \begin{pmatrix} -2 & 4 & -8 & 16 \\ -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 4 & 8 & 16 \end{pmatrix}^{-1} \begin{pmatrix} f(a-2h) - f(a) \\ f(a-h) - f(a) \\ f(a+h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix} \end{aligned}$$

It is smart to find the inverse matrix symbolically here:

```
V = sym.Matrix([[-2,4,-8,16],[-1,1,-1,1],[1,1,1,1],[2,4,8,16]])
V.inv()
```

$$\begin{bmatrix} \frac{1}{12} & -\frac{2}{3} & \frac{2}{3} & -\frac{1}{12} \\ -\frac{1}{24} & \frac{2}{3} & \frac{2}{3} & -\frac{1}{24} \\ -\frac{1}{12} & \frac{1}{6} & -\frac{1}{6} & \frac{1}{12} \\ \frac{1}{24} & -\frac{1}{6} & -\frac{1}{6} & \frac{1}{24} \end{bmatrix}$$

The expression above therefore becomes

$$\begin{pmatrix} 1/12 & -2/3 & 2/3 & -1/12 \\ -1/24 & 2/3 & 2/3 & -1/24 \\ -1/12 & 1/6 & -1/6 & 1/12 \\ 1/24 & -1/6 & -1/6 & 1/24 \end{pmatrix} \begin{pmatrix} f(a-2h) - f(a) \\ f(a-h) - f(a) \\ f(a+h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix}$$

Evaluating the first row we get

$$\begin{aligned} h p_4'(a) &= (1/12 \quad -2/3 \quad 2/3 \quad -1/12) \begin{pmatrix} f(a-2h) - f(a) \\ f(a-h) - f(a) \\ f(a+h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix} \\ &= \frac{f(a-2h)}{12} - \frac{2f(a-h)}{3} + \frac{2f(a+h)}{3} - \frac{f(a+2h)}{12} - \left(\frac{1}{12} - \frac{2}{3} + \frac{2}{3} - \frac{1}{12} \right) f(a) \\ &= \frac{f(a-2h) - 8f(a-h) + 8f(a+h) - f(a+2h)}{12}. \end{aligned}$$

Dividing by h we get

$$p_4'(a) = \frac{f(a-2h) - 8f(a-h) + 8f(a+h) - f(a+2h)}{12h}.$$

Table 11.1: Truncation errors for different h for the three methods we have considered.

h	Newton	Symmetric Newton	Four point method
0.100000	2.541321e-02	1.461906e-03	2.921795e-06
0.010000	2.411734e-03	1.462630e-05	2.925195e-10
0.001000	2.398590e-04	1.462637e-07	5.773160e-15
0.000100	2.397274e-05	1.462736e-09	7.516210e-14
0.000010	2.397147e-06	1.750344e-11	5.476064e-12
0.000001	2.396897e-07	7.476686e-12	7.476686e-12

11.5 Summary

Let us return to the function $f(x) = \sin(x)$, at the point $a = 0.5$. Since the derivative is $f'(x) = \cos(x)$, and $f'(a) \approx 0.8775825619$, it is easy to check the accuracy in the methods we have derived. The errors with $h = 10^{-1}, 10^{-2}, \dots, 10^{-6}$ for three of the methods are shown in Table 11.1.

Each time h is divided by 10 we see that

- for the Newton quotient, the truncation error is reduced by a factor of about 10. This is as expected since the truncation error was shown to be $M_2 \frac{h}{2}$.
- for the symmetric Newton quotient, the error gets reduced by a factor of about 100. This is as also expected since the truncation error was shown to be $M_3 \frac{h^2}{6}$.
- for the four point method, the error gets reduced by a factor of about 10000. This follows from that the truncation error satisfies $|E_1(f; a, h)| \leq \frac{M_5}{18} h^4$, which we state without proof.

Let us also make loglog plots of the truncation errors. For this we will also include the second derivative, which had a truncation error bounded by $M_4 \frac{h^2}{12}$. We will also use that $M_k \approx (\sin a)^{(k)}$ near a . The truncation error bounds are plotted against each other as follows.

```

a=0.5
h = 10**(-np.arange(0,6,0.1))
plt.loglog(h, np.abs(np.sin(a)*h/2),
            h, np.abs(np.cos(a)*h**2/6),
            h, np.abs(18*np.cos(a)*h**4),
            h, np.abs(np.sin(a)*h**2/12))
plt.legend(['Newton', 'Symmetric Newton',
            'Four points', 'second derivative'])

```

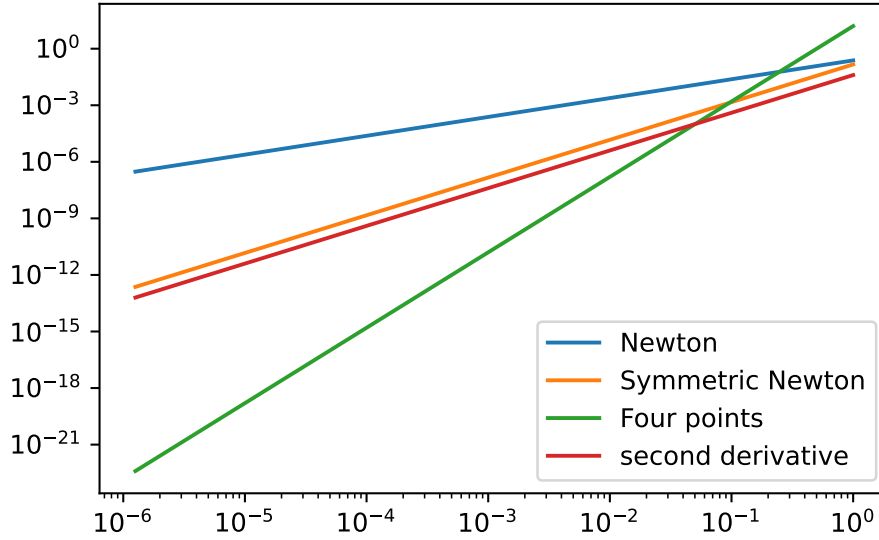


Figure 11.4: loglog plot of the estimates of the truncation errors for the four methods we have considered, for $f(x) = \sin x$ in $x = 0.5$, and for $h \in [10^{-6}, 10^0]$.

Note that, if $|E_1(f; a, h)| \approx Ch^p$, we have that

$$\log |E_1(f; a, h)| \approx p \log h + \log C.$$

Our methods have truncation errors proportional to h , h^2 , and h^4 . This is why we see lines with slopes 1, 2, and 4 in the loglog plot.

Exercises

Exercise 11.1 (Left sided Newton quotient). An alternative to the Newton quotient in Equation 11.3 is the left sided variant

$$f'(a) \approx \frac{f(a) - f(a-h)}{h}.$$

Show that this coincides with the point method based on a and $a-h$.

! Solution

The polynomial interpolating f in a and $a-h$ is the line $p_1(x) = f(a) + \frac{f(a)-f(a-h)}{h}(x-a)$. This has derivative $(f(a) - f(a-h))/h$, which is the left sided variant of the Newton quotient.

Exercise 11.2. Write a program which computes the Newton quotient for the function $f(x) = e^x$ in $a = 1$, and uses $h = 10^{-k}$ for $k = 1, 2, \dots, 14$. Also compute the error. Does the approximation and its error behave as Proposition 11.1 predicts?

Exercise 11.3. When deriving the truncation error in Equation 11.6, it was not obvious what the degree of the Taylor polynomial in Equation 11.4 should be. In this exercise you will see what happens when one decreases the degree of the Taylor polynomial.

- (a) Redo the Taylor expansion in Equation 11.4 again, but this time use a Taylor polynomial of degree 2. Try from this to derive a formula for the error corresponding to Equation 11.5.

! Solution

We now write

$$f(a+h) = f(a) + hf'(a) + \frac{h^2}{2}f''(a) + \frac{h^3}{6}f'''(\xi_h)$$

for an ξ_h in the interval from a to $a+h$. We can now write

$$f'(a) - \frac{f(a+h) - f(a)}{h} = -\frac{h}{2}f''(a) - \frac{h^2}{6}f'''(\xi_h).$$

After taking absolute values the error estimate becomes

$$\frac{h}{2}|f''(a)| + \frac{h^2}{6} \max_{x \in [a, a+h]} |f'''(x)|.$$

This error estimate resembles the one we obtained when using a Taylor expansion of order one, but we also need a general estimate for the third derivative. We still need an estimate for the second derivative, but only in the point a .

- (b) Repeat (a), but use a Taylor series of degree 0, i.e., a constant.

! Solution

We now write $f(a+h) = f(a) + f'(\xi_h)h$ for a ξ_h in the interval between a and $a+h$. In this expression $f'(a)$ is not present, so that it is not possible to obtain an estimate for the truncation error with such a Taylor expansion.

- (c) Why can you conclude that the first order Taylor polynomial and the formula for the error in Equation 11.5 is best?

! Solution

The linear Taylor polynomial is best since it is the shortest possible Taylor series which can give an estimate for $f'(a)$ (as shown in (b)), and also that which gives the simplest expression for the truncation error, in the sense that it does not depend on derivatives of higher order than two (as shown in (a)).

Exercise 11.4. Show that Equation 11.7 Equation 11.8 are correct.

Exercise 11.5. Show that the symmetric Newton quotient in $x = a$

(a) equals $f'(a)$ when f is a quadratic polynomial $f(x) = bx^2 + cx + d$.

! Solution

It is perhaps simplest to show first that this holds for the three functions $f(x) = 1$, $f(x) = x$, and $f(x) = x^2$:

- For $f(x) = 1$ we have that $f'(a) = 0$, and the symmetric Newton quotient is

$$\frac{1 - 1}{2h} = 0.$$

- For $f(x) = x$ we have that $f'(a) = 1$, and the symmetric Newton quotient is

$$\frac{a + h - (a - h)}{2h} = 1.$$

- For $f(x) = x^2$ we have that $f'(a) = 2a$, and the symmetric Newton quotient is

$$\frac{(a + h)^2 - (a - h)^2}{2h} = \frac{4ah}{2h} = 2a.$$

The symmetric Newton quotient will then be exact also for all quadratic polynomials, since all such can be written as linear combinations of 1, x , and x^2 , and since differentiation and the symmetric Newton quotient are linear operations.

(b) equals the average of the two asymmetric quotients

$$f_r(a) = \frac{f(a + h) - f(a)}{h} \quad f_l(a) = \frac{f(a) - f(a - h)}{h}. \quad (11.11)$$

! Solution

We have that

$$\begin{aligned}\frac{f_r(a) + f_l(a)}{2} &= \frac{\frac{f(a+h)-f(a)}{h} + \frac{f(a)-f(a-h)}{h}}{2} \\ &= \frac{f(a+h) - f(a) + f(a) - f(a-h)}{2h} = \frac{f(a+h) - f(a-h)}{2h},\end{aligned}$$

which is the symmetric Newton quotient.

Exercise 11.6. In this exercise we consider some other possible approximations to the derivative.

! Solution

We will solve (a) and (b) in three different ways. First we will find the approximations by differentiating the interpolating polynomial. Then we will use Theorem 11.1. Finally we derive the same approximations by combining Taylor series of fitting order in the given points

- (a) Find an approximation to $f'(a)$ which only uses the values $f(a-h)$, $f(a)$, and $f(a+2h)$.

! Solution

The Newton form of the polynomial interpolating in $a-h$, a , and $a+2h$ is

$$p_2(x) = c_0 + c_1(x - (a-h)) + c_2(x-a)(x-(a-h)).$$

We evaluate this in the three points, and solve for c_0, c_1, c_2 :

$$\begin{aligned}f(a-h) &= c_0 \\ f(a) &= c_0 + c_1 h \\ f(a+2h) &= c_0 + 3c_1 h + 6c_2 h^2.\end{aligned}$$

If we solve this we first get $c_0 = f(a-h)$, $c_1 = \frac{f(a)-f(a-h)}{h}$, and then

$$c_2 = \frac{f(a+2h) - f(a-h) - 3(f(a) - f(a-h))}{6h^2} = \frac{f(a+2h) - 3f(a) + 2f(a-h)}{6h^2}.$$

We thus get

$$p_2(x) = f(a-h) + \frac{f(a) - f(a-h)}{h}(x - (a-h)) + \frac{f(a+2h) - 3f(a) + 2f(a-h)}{6h^2}(x-a)(x-(a-h)),$$

and then

$$p'_2(x) = \frac{f(a) - f(a-h)}{h} + \frac{f(a+2h) - 3f(a) + 2f(a-h)}{6h^2}((x-a) + (x-(a-h))),$$

and finally

$$p_2'(a) = \frac{f(a) - f(a-h)}{h} + \frac{f(a+2h) - 3f(a) + 2f(a-h)}{6h} = \frac{f(a+2h) + 3f(a) - 4f(a-h)}{6h}.$$

Theorem 11.1 gives with $m_1 = -1$, $m_2 = 2$.

$$\begin{pmatrix} hp_2^{(1)}/1! \\ h^2 p_2^{(2)}/2! \end{pmatrix} = \begin{pmatrix} -1 & (-1)^2 \\ 2 & 2^2 \end{pmatrix}^{-1} \begin{pmatrix} f(a-h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix} = -\frac{1}{6} \begin{pmatrix} 4 & -1 \\ -2 & -1 \end{pmatrix} \begin{pmatrix} f(a-h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix}$$

The first row here says that

$$\begin{aligned} f'(a) &\approx p_2^{(1)} = -\frac{1}{6h} (4(f(a-h) - f(a)) - (f(a+2h) - f(a))) \\ &= \frac{-4f(a-h) + 3f(a) + f(a+2h)}{6h}. \end{aligned}$$

Alternatively we can write

$$\begin{aligned} f(a-h) &= f(a) - f'(a)h + \frac{f''(a)}{2}h^2 + \frac{f'''(c_1)}{6}h^3 \\ f(a+2h) &= f(a) + 2f'(a)h + 2f''(a)h^2 + 4\frac{f'''(c_2)}{3}h^3 \end{aligned}$$

If we multiply the first equation with 4 and subtract the two (this eliminates $f''(a)$) we get that

$$4f(a-h) - f(a+2h) = 3f(a) - 6f'(a)h + CM_3h^3,$$

where C is a number. This gives the approximation $f'(a) \approx \frac{3f(a) - 4f(a-h) + f(a+2h)}{6h}$, where the error is $\approx CM_3h^2$.

- (b) Find an approximation to $f'(a)$ which only uses the values $f(a)$, $f(a+h)$, and $f(a+2h)$.

! Solution

The Newton form of the polynomial interpolating in a , $a+h$, and $a+2h$ is

$$p_2(x) = c_0 + c_1(x-a) + c_2(x-a)(x-(a+h)).$$

We evaluate this in the three points, and solve for c_0, c_1, c_2 :

$$\begin{aligned} f(a) &= c_0 \\ f(a+h) &= c_0 + c_1hf(a+2h) &= c_0 + 2c_1h + 2c_2h^2. \end{aligned}$$

If we solve this we first get $c_0 = f(a)$, $c_1 = \frac{f(a+h)-f(a)}{h}$, and then

$$c_2 = \frac{f(a+2h) - f(a) - 2f(a+h) + 2f(a)}{2h^2} = \frac{f(a+2h) - 2f(a+h) + f(a)}{2h^2}.$$

We thus get

$$p_2(x) = f(a) + \frac{f(a+h) - f(a)}{h}(x-a) + \frac{f(a+2h) - 2f(a+h) + f(a)}{2h^2}(x-a)(x-(a+h)),$$

and then

$$p_2'(x) = \frac{f(a+h) - f(a)}{h} + \frac{f(a+2h) - 2f(a+h) + f(a)}{2h^2}((x-a) + (x-(a+h))),$$

and finally

$$p_2'(a) = \frac{f(a+h) - f(a)}{h} - \frac{f(a+2h) - 2f(a+h) + f(a)}{2h} = \frac{-f(a+2h) + 4f(a+h) - 3f(a)}{2h}$$

Theorem 11.1 gives with $m_1 = 1$, $m_2 = 2$.

$$\begin{pmatrix} hp_2^{(1)}/1! \\ h^2 p_2^{(2)}/2! \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 4 \end{pmatrix}^{-1} \begin{pmatrix} f(a+h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 4 & -1 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} f(a+h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix}$$

The first row here says that

$$\begin{aligned} f'(a) &\approx p_2^{(1)} = \frac{1}{2h} (4(f(a+h) - f(a)) - (f(a+2h) - f(a))) \\ &= \frac{-3f(a) + 4f(a+h) - f(a+2h)}{2h}. \end{aligned}$$

Alternatively we can write

$$\begin{aligned} f(a+h) &\approx f(a) + f'(a)h + \frac{f''(a)}{2}h^2 + \frac{f'''(c_1)}{6}h^3 \\ f(a+2h) &\approx f(a) + 2f'(a)h + 2f''(a)h^2 + 4\frac{f'''(c_2)}{3}h^3 \end{aligned}$$

If we multiply the first equation with 4 and subtract the two (this eliminates $f''(a)$) we get that

$$4f(a+h) - f(a+2h) = 3f(a) + 2f'(a)h + CM_3h^3,$$

where C is a number. This gives the approximation $f'(a) \approx \frac{-3f(a) + 4f(a+h) - f(a+2h)}{2h}$, where the error is $\approx CM_3h^2$ also here.

Exercise 11.7. This exercise illustrates another method way to derive methods

for numerical differentiation.

- (a) Suppose we want to derive a method for approximating $f'(a)$ on the form

$$f'(a) \approx c_1 f(a-h) + c_2 f(a+h), \quad c_1, c_2 \in \mathbb{R}.$$

We want the method to be exact when $f(x) = 1$ and $f(x) = x$. Use these requirements to compute c_1 and c_2 .

! Solution

If the approximation method $f'(a) \approx c_1 f(a-h) + c_2 f(a+h)$ should be exact for $f(x) = 1$, we must have that $0 = c_1 + c_2$, since $f(a-h) = f(a+h) = 1$, and since $f'(x) = 0$. we must thus have that $c_2 = -c_1$. If the method is to be exact for $f(x) = x$ we must in the same way have that

$$1 = c_1(a-h) + c_2(a+h) = c_1(a-h) - c_1(a+h) = -2c_1 h,$$

so that $c_1 = -\frac{1}{2h}$, so that also $c_2 = \frac{1}{2h}$. The method therefore becomes $-\frac{1}{2h} f(a-h) + \frac{1}{2h} f(a+h) = \frac{f(a+h)-f(a-h)}{2h}$

- (b) Show that the method in (a) is exact for all polynomials of degree one, and compare it with the methods we have derived in this chapter.

! Solution

If $f(x) = cx + d$ we have that $f'(x) = c$, and the method becomes

$$\frac{f(a+h) - f(a-h)}{2h} = \frac{c(a+h) + d - (c(a-h) + d)}{2h} = \frac{2ch}{2h} = c,$$

so that the method is exact for all polynomials of degree ≤ 1 . We see that the method coincides with the symmetric Newton quotient, and it therefore has an error of order h^2 , which is better than the Newton quotient (which has an error of order h). It is worse than the four point method, which is of order h^4 .

The method is also exact for polynomials of degree ≤ 2 . There are several ways to see this. Firstly, the error estimate from this section uses $f^{(3)}(x)$, and since all quadratic polynomials have a third derivative of 0, the error must be 0. One can also as above substitute $f(x) = x^2$ in the formula:

$$\frac{f(a+h) - f(a-h)}{2h} = \frac{(a+h)^2 - (a-h)^2}{2h} = \frac{4ah}{2h} = 2a,$$

which also is $f'(a)$. Finally, the symmetric Newton quotient was defined as the derivative in a of the unique parabola interpolating f in $a-h$, a , and $a+h$. If f itself is a parabola, it is equal to its interpolant since

this is unique, so that the symmetric Newton quotient must return the derivative.

- (c) Use the methods in (a) and (b) to derive a method for approximating $f''(a)$,

$$f''(a) \approx c_1 f(a-h) + c_2 f(a) + c_3 f(a+h), \quad c_1, c_2, c_3 \in \mathbb{R},$$

demanding that the method should be exact for $f(x) = 1$, x , and x^2 . Do you recognise this method?

! Solution

If the approximation method $f''(a) \approx c_1 f(a-h) + c_2 f(a) + c_3 f(a+h)$ is exact for $f(x) = 1$, we must have that $0 = c_1 + c_2 + c_3$. If it is exact for $f(x) = x$ we must have that

$$0 = c_1(a-h) + c_2 a + c_3(a+h) = a(c_1 + c_2 + c_3) + h(-c_1 + c_3) = h(-c_1 + c_3),$$

which gives that $c_1 = c_3$. If it is exact for $f(x) = x^2$ we must have that

$$\begin{aligned} 2 &= c_1(a-h)^2 + c_2 a^2 + c_3(a+h)^2 \\ &= a^2(c_1 + c_2 + c_3) - 2ahc_1 + 2ahc_3 + h^2(c_1 + c_3) = 2c_1 h^2, \end{aligned}$$

which gives that $c_1 = \frac{1}{h^2}$. We thus also get $c_3 = \frac{1}{h^2}$, and that $c_2 = -c_1 - c_3 = -\frac{2}{h^2}$, so that the method becomes

$$\frac{1}{2h} f(a-h) - \frac{1}{h} f(a) + \frac{1}{2h} f(a+h) = \frac{f(a-h) - 2f(a) + f(a+h)}{h^2}.$$

We see that this coincides with the three point method for the second derivative which we have already considered.

- (d) Show that the method in (c) is exact for all cubic polynomials.

! Solution

All cubic polynomials have a fourth derivative of 0, and the truncation error is thus 0 ($M_4 = 0$). Alternatively we can substitute $f(x) = x^3$ in

the formula:

$$\begin{aligned}
 & \frac{f(a-h) - 2f(a) + f(a+h)}{2h} \\
 &= \frac{(a-h)^3 - 2a^3 + (a+h)^3}{h^2} \\
 &= \frac{a^3 - 3a^2h + 3ah^2 - h^3 - 2a^3 + a^3 + 3a^2h + 3ah^2 + h^3}{h^2} \\
 &= \frac{6ah^2}{h^2} = 6a,
 \end{aligned}$$

which coincides with the second derivative of f in a .

Exercise 11.8. In this exercise we will compare the symmetric Newton quotient with the two asymmetric Newton quotients.

(a) Plot the graph of the function

$$f(x) = \frac{-x^2 + 10x - 5}{4}$$

over the interval $[0, 6]$, together with the three secants corresponding to the three approximations to the derivative from (a) (use $a = 3$ and $h = 2$). Can you from this decide which approximation is best?

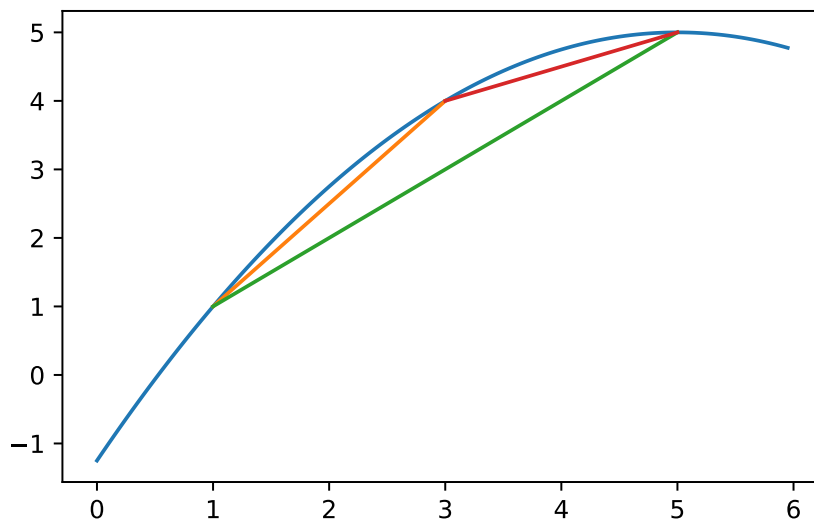
! Solution

We can draw the graph together with the secants as follows:

```

x = np.arange(0,6,0.05)
f = lambda x: (-x**2+10*x-5)/4
plt.plot(x,f(x))
plt.plot([1,3],[f(1),f(3)])
plt.plot([1,5],[f(1),f(5)])
plt.plot([3,5],[f(3),f(5)])
plt.show()

```



- (b) Compute the three approximations to the derivatives in (a) numerically for $f(x)$ with $a = 3$ and $h_1 = 0.1$ and $h_2 = 0.001$. What are the relative errors?

! Solution

With 6 digits:

$$(f(a+h) - f(a))/h = 0.975, \text{ relative error: } 0.025.$$

$$(f(a) - f(a-h))/h = 1.025, \text{ relative error: } 0.025.$$

$$(f(a+h) - f(a-h))/(2h) = 1, \text{ relative error: } 8.88178 \times 10^{-16}.$$

Chapter 12

Numerical integration

In numerical differentiation known function values are used to find approximations to the derivative. In *numerical integration* the same information is used to find approximations to the *integral*. These methods are useful when the function is only known at isolated points, or when it is difficult or impossible to compute the integral by hand.

In Chapter 11 we explained a general algorithm for deriving methods of numerical differentiation. The corresponding algorithm for numerical integration is as follows:

- Find the polynomial interpolating the function in a suitable set of points,
- use the integral of this polynomial as an approximation to the integral of the function.
- analyse the truncation error in a way similar to that of numerical differentiation.

We start this chapter by revisiting the definition of the integral (Section 12.1). Then we look at the simplest method for numerical integration, the midpoint method (Section 12.2). In Section 12.3 we will then formulate the general algorithm stated above in more detail, and look at some important special cases.

12.1 Definition of the integral

The integral of a function $f: [a, b] \rightarrow \mathbb{R}$ is interpreted as the area under the graph, where the positive values of f have a positive contribution to the area, and the negative values have a negative contribution (see Figure 12.1). The standard notation for the integral is

$$\int_a^b f(x) dx.$$

Before we continue we must introduce a concept which we will use several times in our definition of the integral.

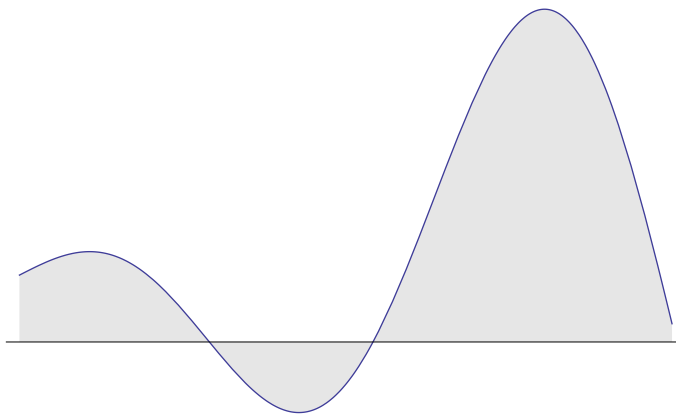


Figure 12.1: The area under the graph of a function.

Definition 12.1 (Partition). Let a and b be two real numbers with $a < b$. A *partition* of $[a, b]$ is a collection of numbers $\{x_0, x_1, \dots, x_n\}$ so that

$$a = x_0 < x_1 < x_2 \cdots < x_{n-1} < x_n = b.$$

The partition is said to be *uniform* if $x_i - x_{i-1} = (b - a)/n$ for $i = 1, \dots, n$. The distance between the points, $h = (b - a)/n$, is then called the *step length* of the partition.

If $\Pi = \{x_0, x_1, \dots, x_n\}$ is a partition, we let the *size* of the partition be $|\Pi| = \max_{i=1, \dots, n} |x_i - x_{i-1}|$, i.e. the largest distance between neighbouring points. If Π is uniform, $|\Pi| = (b - a)/n$.

To define the integral one uses so-called *Riemann sums*, which are approximations to the area under the graph. We choose a partition $\Pi = \{x_0, x_1, \dots, x_n\}$ of $[a, b]$, and find the maximum and minimum of f on each sub-interval $[x_{i-1}, x_i]$,

$$m_i = \min_{x \in [x_{i-1}, x_i]} f(x), \quad M_i = \max_{x \in [x_{i-1}, x_i]} f(x) \quad \text{for } i = 1, 2, \dots, n.$$

The min and the max here may or may not exist. Continuity of f will secure this, however. Approximations to the integral can be obtained from two different piecewise constant functions, both being constant on each sub-interval $[x_{i-1}, x_i]$: The first equals m_i on this sub-interval, the second equals M_i (see Figure 12.2). We sum up the areas under the two, and end up with the two approximations

$$\underline{I}(\Pi) = \sum_{i=1}^n m_i(x_i - x_{i-1}), \quad \bar{I}(\Pi) = \sum_{i=1}^n M_i(x_i - x_{i-1}). \quad (12.1)$$

The first approximation is in general too small, the second too large. The integral is obtained by letting the size of the partition go to zero.

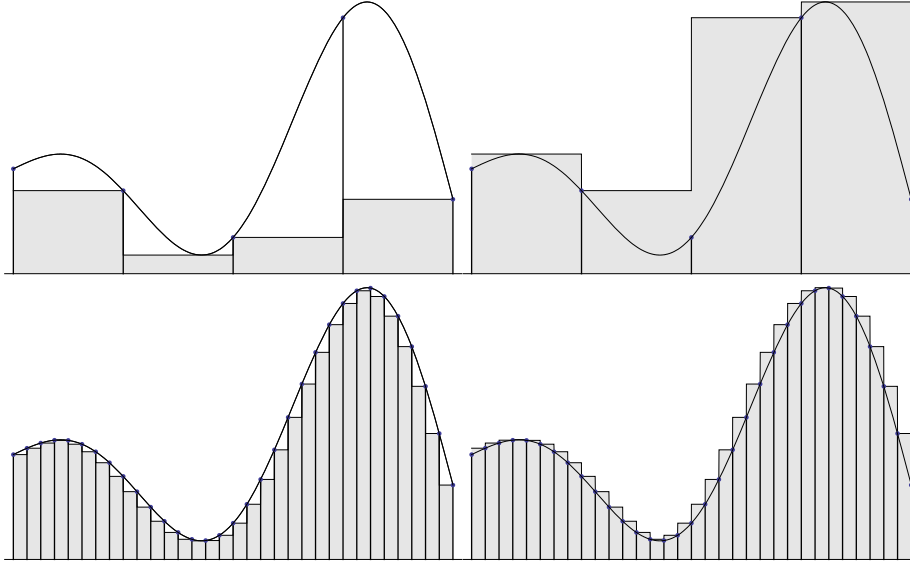


Figure 12.2: Inscribed (left) and circumscribed (right) piecewise constant functions for a partition (top) and a finer partition (bottom).

Definition 12.2 (The integral). Let f be a function defined on the interval $[a, b]$. If

$$\sup_{\Pi} \underline{I}(\Pi) \text{ and } \inf_{\Pi} \bar{I}(\Pi)$$

exist (where the infimum and supremum are taken over all partitions of $[a, b]$) and are equal, we say that f is *integrable*, and we define the integral of f over $[a, b]$, $\int_a^b f(x) dx$, as the common value of the two.

The construction of the integral is illustrated Figure 12.2, where we see how the piecewise constant approximations become better as the partitions get finer. It turns out that the following holds

Theorem 12.1. *Any continuous function is integrable.*

Proof. Let $\Pi_n = \{x_0, \dots, x_n\}$ be partitions of $[a, b]$ so that the size of the partitions go to 0, the size of a partition being the largest distance between neighbouring points x_{i-1} and x_i . We have that

$$\underline{I}(\Pi_n) \leq \int_a^b f(x) dx \leq \bar{I}(\Pi_n). \quad (12.2)$$

Assume that ϵ is given. Since f is continuous on $[a, b]$, it is also uniformly continuous on $[a, b]$. Due to this, and since the size of the partitions goes to 0, there exists an N so that, for $n \geq N$, $|f(x) - f(y)| \leq \epsilon/(b - a)$ for x, y in the

same sub-interval of Π_n . In particular this means that $M_i - m_i \leq \epsilon/(b-a)$, and we get

$$\bar{I}(\Pi_n) - \underline{I}(\Pi_n) = \sum_{i=1}^n (M_i - m_i)(x_i - x_{i-1}) \leq \frac{\epsilon}{b-a}(x_i - x_{i-1}) = \frac{\epsilon}{b-a}(b-a) = \epsilon.$$

The result now follows from Equation 12.2. $\square$

The integral has a property which is very useful when it comes to numerical computation:

Theorem 12.2. *Let $f: [a, b] \rightarrow \mathbb{R}$ be continuous, let $\Pi_n = \{x_0, \dots, x_n\}$ be a partition of $[a, b]$, and let t_i be a number in $[x_{i-1}, x_i]$ for every $i = 1, \dots, n$. Then the sum*

$$\sum_{i=1}^n f(t_i)(x_i - x_{i-1}) \quad (12.3)$$

will converge to $\int_a^b f(x) dx$ as the “size” of the Π_n goes to 0.

Thus, in general we can derive methods for numerical integration by splitting $[a, b]$ in small sub-intervals, and add together contributions from each sub-interval.

Proof. We have that

$$\underline{I}(\Pi_n) \leq \sum_{i=1}^n f(t_i)(x_i - x_{i-1}) \leq \bar{I}(\Pi_n)$$

The result follows by combining this with Equation 12.2. $\square$

Different choices for the t_i in Theorem 12.2 give rise to various methods for approximating the integral. We could for instance choose t_i as the maximum or the minimum on each sub-interval, but the problem with this is that it can be both hard and time-consuming to find maxima and minima, even for a computer. We could also choose t_i equal to one of the end points x_{i-1} or x_i , but it turns out that better choices exist, as we will see.

12.2 The midpoint method

An even better choice for the t_i turns out to be the midpoint $t_i = (x_i + x_{i-1})/2$ on each sub-interval. With this choice Equation 12.3 gives the approximation

$$I \approx \sum_{i=1}^n f((x_i + x_{i-1})/2)(x_i - x_{i-1}). \quad (12.4)$$

Definition 12.3 (The midpoint method). Let f be a function which is integrable on the interval $[a, b]$, and let $\{x_i\}_{i=0}^n$ be a partition of $[a, b]$. The *midpoint method* approximates the integral of f with

$$\int_a^b f(x) dx \approx I_{\text{mid}}(h) = h \sum_{i=1}^n f(x_{i-1/2}), \quad (12.5)$$

where $x_{i-1/2} = (x_{i-1} + x_i)/2 = a + (i - 1/2)h$.

We can think of this as a constant approximation on each sub-interval, where the constant is the value in the midpoint. The method is illustrated in Figure 12.3 for the cases with one and five sub-intervals.

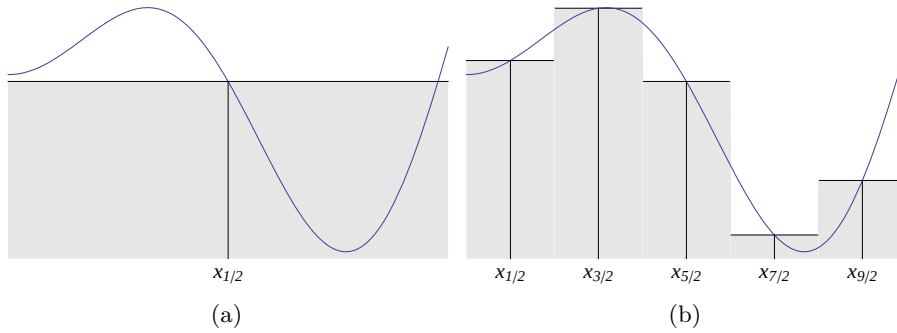


Figure 12.3: The midpoint method with one sub-interval (Figure 12.3a), and five sub-intervals (Figure 12.3b).

An implementation of the midpoint method can be as follows.

```
def midpointn(f, a, b, n):
    h = (b - a)/n
    return h*sum( f(a + h/2 + np.arange(0,n)*h))
```

The midpoint method gives only an approximation, and will produce a small error. As in Chapter 11 we shall estimate how large this error can be. This can be done in two steps. First we analyse the error in the case of only one sub-interval. This is called the *local error*. Then we find an estimate for the general case with many sub-intervals. This is called the *global error*.

12.2.1 The local error

The error in the midpoint method with one sub-interval only is

$$E = \int_a^b f(x) dx - f(m)(b - a), \quad (12.6)$$

where $m = (a + b)/2$. If we use a Taylor expansion of f around the midpoint m we get

$$f(x) = f(m) + (x - m)f'(m) + \frac{(x - m)^2}{2}f''(\xi),$$

where ξ is a number between m and x , which depends on x . Integrating this we get

$$\begin{aligned} \int_a^b f(x) dx &= \int_a^b \left(f(m) + (x - m)f'(m) + \frac{(x - m)^2}{2}f''(\xi) \right) dx \\ &= f(m)(b - a) + \frac{f'(m)}{2}[(x - m)^2]_a^b + \frac{1}{2} \int_a^b (x - m)^2 f''(\xi) dx \\ &= f(m)(b - a) + \frac{1}{2} \int_a^b (x - m)^2 f''(\xi) dx, \end{aligned}$$

where the term in the middle was 0. This leads us to the expression

$$|E| = \left| \int_a^b f(x) dx - f(m)(b - a) \right| = \frac{1}{2} \left| \int_a^b (x - m)^2 f''(\xi) dx \right|.$$

Some more calculations give

$$\begin{aligned} |E| &= \frac{1}{2} \left| \int_a^b (x - m)^2 f''(\xi) dx \right| \leq \frac{1}{2} \int_a^b |(x - m)^2 f''(\xi)| dx \\ &= \frac{1}{2} \int_a^b (x - m)^2 |f''(\xi)| dx \\ &\leq \frac{M_2}{2} \int_a^b (x - m)^2 dx \\ &= \frac{M_2}{2} \frac{1}{3} [(x - m)^3]_a^b \\ &= \frac{M_2}{6} ((b - m)^3 - (a - m)^3) \\ &= \frac{M_2}{24} (b - a)^3, \end{aligned}$$

where $M_2 = \max_{x \in [a, b]} |f''(x)|$. The inequality on the first line holds because, when we move the absolute value inside the integral, the function we integrate becomes non-negative everywhere. This means that areas where the integrand originally was negative, now become positive, so that the second integral is larger than the first.

The equality on the second line holds since $(x - m)^2$ is never negative, and the inequality on the third line holds since we replace $|f''(\xi)|$ with its maximum on the interval $[a, b]$. On the fourth line we simply compute the integral of $(x - m)^2$. Then we insert the limits. The last line follows from that $(b - m)^3 = -(a - m)^3 = (b - a)^3/8$. We have thus proved the following lemma.

Lemma 12.1. *Let $f: [a, b] \rightarrow \mathbb{R}$ be a continuous function, with continuous first- and second order derivatives. The error in the midpoint method with one sub-interval is then bounded by*

$$\left| \int_a^b f(x) dx - f(m)(b-a) \right| \leq \frac{M_2}{24}(b-a)^3,$$

where $m = (a+b)/2$.

The important thing in Lemma 12.1 is the factor $(b-a)^3$. This implies that, if we halve the width of the intervals, the error in the method will be reduced with a factor of 8.

12.2.2 The global error

Let us now see consider the error resulting from adding together contributions from many sub-intervals. $[a, b]$ is partitioned into n sub-intervals, all with width $h = (b-a)/n$. With

$$I = \int_a^b f(x) dx = \sum_{i=1}^n \int_{x_{i-1}}^{x_i} f(x) dx, \quad I_{\text{mid}}(h) = \sum_{i=1}^n f(x_{i-1/2})h,$$

the total error in the midpoint method is

$$I - I_{\text{mid}}(h) = \sum_{i=1}^n \left(\int_{x_{i-1}}^{x_i} f(x) dx - f(x_{i-1/2})h \right),$$

where the expression inside the parenthesis is the local error on the interval $[x_{i-1}, x_i]$. We therefore have

$$\begin{aligned} |I - I_{\text{mid}}(h)| &= \left| \sum_{i=1}^n \left(\int_{x_{i-1}}^{x_i} f(x) dx - f(x_{i-1/2})h \right) \right| \\ &\leq \sum_{i=1}^n \left| \int_{x_{i-1}}^{x_i} f(x) dx - f(x_{i-1/2})h \right| \\ &\leq \sum_{i=1}^n \frac{h^3}{24} M_2^{(i)} \end{aligned}$$

where $M_2^{(i)} = \max_{x \in [x_{i-1}, x_i]} |f''(x)|$. In the first inequality the triangle inequality was applied, while we in the second inequality have used Lemma 12.1. We can simplify Equation 12.7 further by using the maximum over the entire $[a, b]$, instead of over $[x_{i-1}, x_i]$: For all i we have that

$$M_2^{(i)} = \max_{x \in [x_{i-1}, x_i]} |f''(x)| \leq \max_{x \in [a, b]} |f''(x)| = M_2,$$

so that Equation 12.7 simplifies to

$$|I - I_{\text{mid}}(h)| \leq \sum_{i=1}^n \frac{h^3}{24} M^{(i)} \leq \sum_{i=1}^n \frac{h^3}{24} M_2 = \frac{h^3}{24} n M_2. \quad (12.7)$$

Since $h = (b - a)/n$, we get $hn = b - a$. Inserting this in Equation 12.7, we get the following estimate for the global error.

Theorem 12.3. *Suppose that $f: [a, b] \rightarrow \mathbb{R}$ and its first two derivatives are continuous. Then the error in the midpoint method is bounded by*

$$|I - I_{\text{mid}}(h)| \leq (b - a) \frac{h^2}{24} M_2. \quad (12.8)$$

12.2.3 The step length

We want to approximate the integral, and let us say we want the error in the approximation to be smaller than some $\epsilon > 0$. We can use the error estimate in Equation 12.8 by requiring that

$$(b - a) \frac{h^2}{24} \max_{x \in [a, b]} M_2 \leq \epsilon$$

which easily can be solved for h ,

$$h \leq \sqrt{\frac{24\epsilon}{(b - a)M_2}}.$$

If we can compute or estimate M_2 , the maximal value of the second derivative, we can insert this in the expression above, and find an upper bound for h which guarantees the error in the integral becomes less than ϵ .

12.3 General method for deriving numerical integration methods

There also exist many other methods for numerical integration. The following algorithm for deriving approximations to $I = \int_a^b f(x) dx$ is a direct parallel to that of numerical differentiation.

1. Let $n, k \geq 1$. Split the interval $[a, b]$ into nk equally large sub-intervals

$$a = x_0 < x_1 < x_2 < \cdots < x_{nk-1} < x_{nk} = b.$$

Set $h = (b - a)/n$.

2. For each sub-interval $J_i = [x_{(i-1)k}, x_{ik}]$, find the polynomial $p_{k,i}$ of degree k which interpolates f in the $k + 1$ points $\{x_{(i-1)k}, \dots, x_{ik}\}$.

3. Use $\int_{J_i} p_{k,i}(x) dx$ as an approximation to $\int_{J_i} f(x) dx$, and add these together to obtain

$$I(h) = \sum_{i=1}^n \int_{J_i} p_{k,i}(x) dx.$$

This algorithm is also called the *Newton–Cotes method*. With help of the analysis in Chapter 10 we can estimate the error in this method:

Theorem 12.4. *Let $n, k \geq 1$ and let $f: [a, b] \rightarrow \mathbb{R}$ be a function with $k+1$ continuous derivatives. Then the error in the Newton–Cotes method of order k with step length $h = (b-a)/n$ satisfies*

$$|I - I(h)| \leq \frac{(b-a)M_{k+1}}{(k+1)!} h^{k+1}. \quad (12.9)$$

Proof. Choose a sub-interval J_i , and let $p_{k,i}$ be the polynomial of degree $\leq k$ interpolating f in the points $x_{ik}, \dots, x_{(i+1)k}$ (as describes in the general algorithm of numerical integration). From Theorem 10.1 we know that for $x \in J_i$

$$|f(x) - p_{k,i}(x)| \leq \frac{M_{k+1}}{(k+1)!} (x_{(i+1)k} - x_{ik})^{k+1} = \frac{M_{k+1}}{(k+1)!} h^{k+1}.$$

Therefore

$$\begin{aligned} |I - I(h)| &= \left| \int_a^b f(x) dx - \sum_{i=1}^n \int_{J_i} p_{k,i}(x) dx \right| = \left| \sum_{i=1}^n \int_{J_i} f(x) - p_{k,i}(x) dx \right| \\ &\leq \sum_{i=1}^n \int_{J_i} |f(x) - p_{k,i}(x)| dx \\ &\leq \sum_{i=1}^n \int_{J_i} \frac{M_{k+1}}{(k+1)!} h^{k+1} dx \\ &= \frac{M_{k+1}}{(k+1)!} \sum_{i=1}^n h^{k+2} \\ &= \frac{M_{k+1}}{(k+1)!} n h^{k+2}. \end{aligned}$$

(From the first to the second line we use the triangle inequality, then we inserted the error estimate for the interpolant, and then we integrated the constant $M_{k+1}/(k+1)!h^{k+1}$.) If we now insert $h = (b-a)/n$ we get Equation 12.9. $\square$

From the error estimate Equation 12.9 we see that, to decrease the error, we can increase k , decrease h (i.e., increasing n), or a combination of the two. What combination is the most effective depends on how large the factor M_{k+1} is for different k .

In the next sections we will consider the cases $k = 1$, which gives the trapezoidal rule, and the case $k = 2$, which gives Simpson's method.

12.4 The trapezoidal rule

We know that the line which interpolates f in the two points x_{i-1} and x_i is

$$p_{1,i}(x) = f(x_{i-1}) + \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}(x - x_{i-1}).$$

If we apply this in the general algorithm we get *the trapezoidal rule*; as we see from Figure 12.4 the approximation in each sub-interval J_i is a trapezoid. We then get the approximation

$$\int_{x_{i-1}}^{x_i} p_{1,i}(x) dx = \frac{f(x_{i-1}) + f(x_i)}{2} h.$$

Adding the contributions from all sub-intervals we get the approximation

$$I_{\text{trap}}(h) = \sum_{i=1}^n \frac{f(x_{i-1}) + f(x_i)}{2} h.$$

Since x_i is both the end of one interval and the beginning of the next for $i \neq 0, n$, this can be written as

$$I_{\text{trap}}(h) = h \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i) \right). \quad (12.10)$$

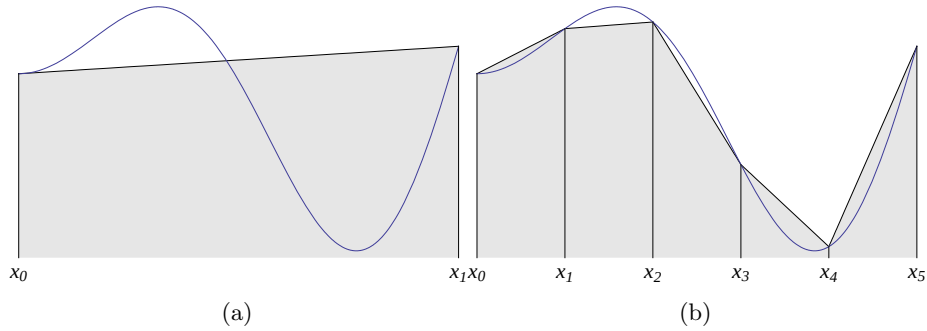


Figure 12.4: The trapezoidal rule with one sub-interval (Figure 12.4a), and 5 sub-intervals (Figure 12.4b).

If we apply Theorem 10.1 with $k = 1$ we get the following error estimate.

Theorem 12.5 (Error estimate for the trapezoidal rule). *Let $f: [a, b] \rightarrow \mathbb{R}$ be a two times continuously differentiable function. The error in the trapezoidal rule with step length h is then bounded by*

$$|I - I_{\text{trap}}(h)| \leq \frac{(b-a)M_2}{2} h^2. \quad (12.11)$$

With some more computation one can show that the trapezoidal rule has an error at most $\frac{(b-a)M_2}{12}h^2$, i.e., six times lower than the right hand side in Equation 12.11 (see Exercise 12.9). The error estimate given by Equation 12.11 is thus an *over-estimate*. The constant factor $\frac{1}{6}$ is not particularly important; the important thing in Equation 12.11 is

- the factor h^2 , which means that if we for instance halve h , the error will be reduced by a factor of $1/4$.
- the factor M_2 , i.e., the error estimate depends on the second derivative.

An implementation of the trapezoidal rule can be as follows.

```
def trapezoidaln(f, a, b, n):
    h = (b - a)/n
    return h*( (f(a) + f(b))/2 + sum( f(a + np.arange(1,n)*h)))
```

12.5 Simpson's method

For $k = 2$ we approximate with a quadratic polynomial on each sub-interval J_i . The method in this case is called *Simpson's method*, and is illustrated in Figure 12.5.

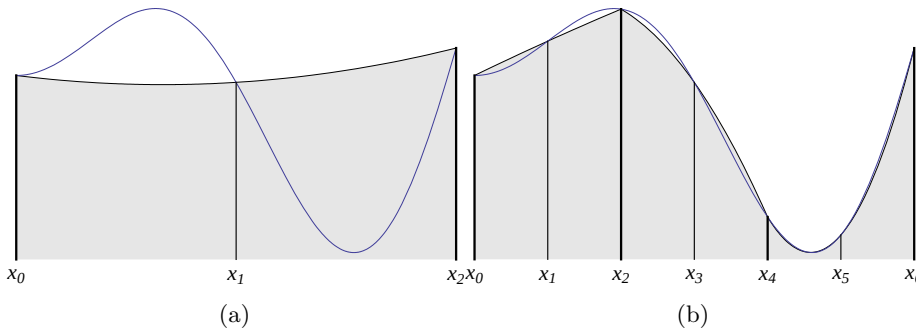


Figure 12.5: Simpson's method with one sub-interval (Figure 12.5a), and three sub-intervals (Figure 12.5b).

In Exercise 12.12 you will be asked to show that the Newton form of the polynomial interpolating f in a , $m = (a + b)/2$, and b is

$$p_2(x) = f(a) + \frac{f(m) - f(a)}{h}(x - a) + \frac{f(a) - 2f(m) + f(b)}{2h^2}(x - a)(x - m),$$

where $h = (b - a)/2$, and that

$$\int_a^b p_2(x) dx = \frac{h}{3} (f(a) + 4f(m) + f(b)).$$

Table 12.1: Truncation errors for three of the methods we have considered, for different numbers of sub-intervals.

n	Midpoint	Trapezoidal	Simpson
2	8.829660e-03	1.760413e-02	1.839786e-05
4	2.195332e-03	4.387233e-03	1.143445e-06
8	5.480824e-04	1.095951e-03	7.136556e-08
16	1.369738e-04	2.739342e-04	4.458792e-09
32	3.424052e-05	6.848020e-05	2.786502e-10
64	8.559946e-06	1.711984e-05	1.741507e-11
128	2.139975e-06	4.279947e-06	1.088685e-12
256	5.349931e-07	1.069986e-06	6.772360e-14
512	1.337482e-07	2.674964e-07	3.774758e-15
1024	3.343705e-08	6.687411e-08	3.330669e-16

This is Simpson's method with one sub-interval. If we add together contributions from all sub-intervals we get

$$I_{\text{Simp}}(h) = \frac{h}{3} \left(f(a) + f(b) + 2 \sum_{i=1}^{n-1} f(x_{2i}) + 4 \sum_{i=1}^n f(x_{2i-1}) \right), \quad (12.12)$$

where now $h = (b - a)/(2n)$. If we insert $k = 2$ in Theorem 10.1, we get the following error estimate for this method.

Theorem 12.6 (Error estimate for Simpson's method). *Let $f: [a, b] \rightarrow \mathbb{R}$ be a four times continuously differentiable function. The error in Simpson's method with step length h is then bounded by*

$$|I - I_{\text{Simp}}(h)| \leq \frac{(b - a)M_4}{6} h^4. \quad (12.13)$$

Simpson's method is much used. Since it is based on quadratic polynomials, it must be exact for such polynomials. It is in fact exact for all cubic polynomials also (see Exercise 12.10). There are sharper error estimates for Simpson's method than this (see Exercise 12.11, which goes a bit outside the curriculum).

An implementation of Simpson's method can be as follows

```
def simpsonn(f, a, b, n):
    h = (b - a)/(2*n)
    return h*( f(a) + f(b) + 2*sum( f(a+2*h*np.arange(1,n)) ) \
              + 4*sum( f(a+h+2*h*np.arange(0,n)) ) )/3
```

In Table 12.1 we have listed errors in the approximations to $\int_0^1 \cos x \, dx = \sin 1 \approx 0.8414709848$ obtained by the three methods we have considered. The interval $[0, 1]$ has been split into 2^k sub-intervals of equal length, for $k = 1, 2, \dots, 10$,

Each time h is halved we see that

- For the midpoint method the error is reduced about a factor of 4. This confirms what we derived in Theorem 12.3.
- For the trapezoidal rule the error is also reduced about a factor of 4, but the error is actually twice as big when compared to the midpoint method (even if we now use two function values in each sub-interval to estimate the integral!). This can also be seen from the error estimate in Equation 12.11, since $1/12$ is twice as big as $1/24$ in Equation 12.8.
- For Simpson's method, the error is reduced by a factor of about 16.

Exercises

Exercise 12.1. In this exercise we will use the definition of the integral on the function $f(x) = e^x$ on the interval $[0, 1]$.

- (a) Write and run a program that computes $\underline{I}$ and $\bar{I}$ for a uniform partition of 10 sub-intervals (let the method take the number of sub-intervals as argument). What is the error from the exact value $I = e - 1 = 1.718281828$ of the integral?

Hint

Instead of a `for`-loop here you can use the `sum`-command when you add together the contribution from the sub-intervals.

Solution

On each sub-interval f assumes its smallest value in the left end point, its largest value in the right end point. Here we can use the `sum`-command the following way to find $\underline{I}(\Pi)$ and $\bar{I}(\Pi)$.

```
def int_vals(n):
    l = (1/n)*np.sum(np.exp( np.arange(0,n)/n ))
    u = (1/n)*np.sum(np.exp( np.arange(1,n+1)/n ))
    l_err = np.abs(l-(np.exp(1)-1))
    u_err = np.abs(u-(np.exp(1)-1))
    return l,u,l_err,u_err
```

Running the code you get

```
l,u,l_err,u_err=int_vals(10)
```

- (b) Write a program which finds how many sub-intervals that are needed to compute $\underline{I}$ with an error less than 3×10^{-3} .

! Solution

We run the following code

```
n = 1
l,u,l_err,u_err=int_vals(n)
while l_err > 3*10**(-3):
    n += 1
    l,u,l_err,u_err = int_vals(n)
n
287
```

Exercise 12.2. Find an approximation to

$$\int_0^{\pi/2} \frac{\sin x}{1+x^2} dx = 0.526978557614 \dots$$

using the midpoint method with 6 sub-intervals.

! Solution

The interval $[0, \pi/2]$ is here split into the intervals

$$[0, \pi/12], [\pi/12, 2\pi/12], [2\pi/12, 3\pi/12], [3\pi/12, 4\pi/12], \\ [4\pi/12, 5\pi/12], [5\pi/12, 6\pi/12].$$

The midpoints on these intervals are $\pi/24, 3\pi/24, 5\pi/24, 7\pi/24, 9\pi/24, 11\pi/24$. With $h = \pi/12$ the midpoint method gives

$$\begin{aligned} I_{\text{mid}} &= \frac{\pi}{12} (f(\pi/24) + f(3\pi/24) + f(5\pi/24) + f(7\pi/24) + f(9\pi/24) + f(11\pi/24)) \\ &= \frac{\pi}{12} \left(\frac{\sin(\pi/24)}{1 + (\pi/24)^2} + \frac{\sin(3\pi/24)}{1 + (3\pi/24)^2} + \frac{\sin(5\pi/24)}{1 + (5\pi/24)^2} + \frac{\sin(7\pi/24)}{1 + (7\pi/24)^2} \right. \\ &\quad \left. + \frac{\sin(9\pi/24)}{1 + (9\pi/24)^2} + \frac{\sin(11\pi/24)}{1 + (11\pi/24)^2} \right) \\ &\approx 0.530624. \end{aligned}$$

Exercise 12.3.

(a) Test the midpoint method with 10 sub-intervals on the integral

$$\int_0^1 e^x dx = e - 1.$$

! Solution

The following code solves the exercise.

```
midpointn(np.exp,0,1,10)
1.7175660864611277
```

(b) Find a value h which guarantees that the absolute error is less than 10^{-10} .

! Solution

From Theorem 12.3 we know that the error is bounded by $(b - a) \frac{h^2}{24} \max_{x \in [a,b]} |f''(x)|$. Since $f''(x) = e^x$, $\max_{x \in [0,1]} |f''(x)| = e$, so that this is $\frac{eh^2}{24}$. We therefore must have that $\frac{eh^2}{24} < 10^{-10}$, so that $h < \sqrt{\frac{24}{e}} 10^{-5} \approx 2.97 \times 10^{-5}$.

Exercise 12.4. Repeat the previous exercise, but instead with the integral

$$\int_2^6 \ln x \, dx = \ln(11664) - 4.$$

! Solution

Approximation with 10 sub-intervals: 5.36648 (with 6 digits). $h \leq 4.89 \times 10^{-5}$.

Exercise 12.5. Repeat Exercise 12.2 and Exercise 12.3 for the trapezoidal rule.

! Solution

Code which solves Exercise 12.2 are as follows:

```
trapezoidaln(lambda x : np.sin(x)/(1+x**2), 0, np.pi/2, 6)
```

0.5197246575929705

Code which solves Exercise 12.3:

```
trapezoidaln(np.exp,0,1,10)
```

1.7197134913893146

The error is bounded by

$$(b - a) \frac{h^2}{12} \max_{x \in [0,1]} |f''(x)| = \frac{h^2}{12} e.$$

We have that $\frac{h^2}{12} e \leq 10^{-10}$ when $h \leq \sqrt{12/e} \times 10^{-5} \approx 2.1 \times 10^{-5}$.

Exercise 12.6. How many function evaluations are needed to compute the integral

$$\int_0^1 \frac{dx}{1+2x}$$

with error less than 10^{-10} for

(a) the trapezoidal rule,

! Solution

Since the error in the trapezoidal rule is bounded by

$$(b-a) \frac{h^2}{12} \max_{x \in [a,b]} |f''(x)|,$$

and since $f''(x) = \frac{8}{(1+2x)^3}$ (which has maximum absolute value for $x = 0$), we need to choose h so that

$$(b-a) \frac{h^2}{12} \max_{x \in [a,b]} |f''(x)| = 8 \frac{h^2}{12} \leq 10^{-10},$$

which gives that $h \leq 10^{-5} \sqrt{3/2}$. The number of function evaluations is then $\approx 1/h = 10^5 / \sqrt{2/3} \approx 81649.66$, which means that at least 81650 evaluations are needed.

(b) The midpoint method,

! Solution

Since the error in the midpoint method is bounded by

$$(b-a) \frac{h^2}{24} \max_{x \in [a,b]} |f''(x)|,$$

we get as in (a) that

$$(b-a) \frac{h^2}{24} \max_{x \in [a,b]} |f''(x)| = 2 \frac{h^2}{6} \leq 10^{-10},$$

which gives that $h \leq 10^{-5} \sqrt{3}$. The number of function evaluations is thus about $1/h = 10^5 / \sqrt{3} \approx 57735.03$. In other words, at least 57736 evaluations are needed.

(c) and Simpson's method?

! Solution

Since the error in Simpson's method is bounded by

$$(b-a) \frac{h^4}{180} \max_{x \in [a,b]} |f^{(iv)}(x)|,$$

and since $f^{(iv)}(x) = \frac{384}{(1+2x)^5}$ (which has maximum absolute value for $x = 0$), we need to choose h so that

$$(b-a) \frac{h^4}{180} \max_{x \in [a,b]} |f^{(iv)}(x)| = 384 \frac{h^4}{180} \leq 10^{-10},$$

which means that h is bounded by $h \leq (180 \cdot 10^{-10}/384)^{1/4}$. The number of function evaluations is then $\approx 1/h = (180 \cdot 10^{-10}/384)^{-1/4} \approx 382.17$, which means that at least 383 evaluations are needed.

Exercise 12.7. Repeat Exercise 12.2 and Exercise 12.3 with Simpson's method.

! Solution

$$h \leq 1.8 \times 10^{-2}$$

Code which solves Exercise 12.2:

```
simpsonn(lambda x : np.sin(x)/(1+x**2), 0, np.pi/2, 6)
```

0.5269911292783586

Code which solves Exercise 12.3:

```
simpsonn(np.exp, 0, 1, 10)
```

1.7182818881038566

Exercise 12.8. We want to write a method for numerical integration on the form

$$\int_a^b f(x) dx \approx w_1 f(a) + w_2 f((a+b)/2) + w_3 f(b),$$

where the w_i are to be decided. Show that, if the method is exact for the three polynomials $f(x) = x^i$ for $i = 0, 1, 2$, then the method coincides with Simpson's method.

! Solution

Since $\int_a^b f(x) dx = w_1 f(a) + w_2 f((a+b)/2) + w_3 f(b)$ for $f(x) = x^i$ for $i = 0, 1, 2$, then this also holds for the polynomial p_2 which interpolates f

in a , $(a+b)/2$, and b . We therefore get

$$\int_a^b p_2(x) dx = w_1 p_2(a) + w_2 p_2((a+b)/2) + w_3 p_2(b) = w_1 f(a) + w_2 f((a+b)/2) + w_3 f(b).$$

We have also shown that the first equals $\frac{b-a}{6}(f(a) + 4f((a+b)/2) + f(b))$, so that

$$w_1 f(a) + w_2 f((a+b)/2) + w_3 f(b) = \frac{b-a}{6}(f(a) + 4f((a+b)/2) + f(b)).$$

By setting $f(a) = 1$, $f((a+b)/2) = f(b) = 0$ we see that $w_1 = \frac{b-a}{6}$, and similarly for w_2 , w_3 . It follows that the method coincides with Simpson's method.

Exercise 12.9. Let $m = \frac{a+b}{2}$ be the midpoint of the interval $[a, b]$, and consider the following application of integration by parts:

$$\begin{aligned} - \int_a^b (x-m)f'(x) dx &= -[(x-m)f(x)]_a^b + \int_a^b f(x) dx \\ &= -\frac{b-a}{2}f(b) - \frac{b-a}{2}f(a) + \int_a^b f(x) dx \\ &= \int_a^b f(x) dx - (b-a)\frac{f(a)+f(b)}{2}. \end{aligned}$$

This shows that the local error in the trapezoidal rule can be written as

$$L = - \int_a^b (x-m)f'(x) dx. \quad (12.14)$$

(a) Apply integration by parts to Equation 12.14 to show that

$$L = \int_a^b \left(C + \frac{1}{2}(x-m)^2 \right) f''(x) dx. \quad (12.15)$$

where C is a constant (we will get back to how we can choose C).

! Solution

Integration by parts on Equation 12.14 with constant C in the anti-

derivative of $x - m$ gives

$$\begin{aligned} L &= - \left[\left(C + \frac{1}{2}(x - m)^2 \right) f'(x) \right]_a^b + \int_a^b \left(C + \frac{1}{2}(x - m)^2 \right) f''(x) dx \\ &= \int_a^b \left(C + \frac{1}{2}(x - m)^2 \right) f''(x) dx. \end{aligned}$$

which shows Equation 12.15.

- (b) Show that, with $C = -(b - a)^2/8$, we have that $C + \frac{1}{2}(x - m)^2 \leq 0$ for $x \in [a, b]$. Show also that, with this choice of C , it follows from Equation 12.15 that $|L| \leq \frac{M_2(b-a)^3}{12}$.

! Solution

For $x \in [a, b]$ we have that $\frac{1}{2}(x - m)^2 \leq \frac{1}{8}(b - a)^2 = -C$, so that $-C - \frac{1}{2}(x - m)^2 \geq 0$ for such x . Taking absolute values in Equation 12.15 with this choice of C gives

$$\begin{aligned} |L| &= \left| \int_a^b \left(\frac{1}{2}(x - m)^2 - (b - a)^2/8 \right) f''(x) dx \right| \\ &\leq \int_a^b \left| \frac{1}{2}(x - m)^2 - (b - a)^2/8 \right| |f''(x)| dx \\ &\leq \int_a^b \left((b - a)^2/8 - \frac{1}{2}(x - m)^2 \right) M_2 dx \\ &= M_2 \left[x(b - a)^2/8 - \frac{1}{6}(x - m)^3 \right]_a^b \\ &= M_2 \left((b - a)^3/8 - \frac{1}{48}(b - a)^3 - \frac{1}{48}(b - a)^3 \right) \\ &= M_2 \left(\frac{1}{8} - \frac{1}{24} \right) (b - a)^3 = \frac{M_2(b - a)^3}{12}. \end{aligned}$$

- (c) Use (b) to show that the global error in the trapezoidal rule is $\leq (b - a) \frac{h^2}{12} M_2$.

! Solution

As for the midpoint method we get the global error by summing the local

errors (replace $b - a$ with h , and nh with $b - a$):

$$\sum_{k=1}^n \frac{M_2 h^3}{12} = n \frac{M_2 h^3}{12} = (b - a) \frac{h^2}{12} M_2.$$

This is what we needed to show.

To simplify notation in the next exercises we define $m = a + h$, and $b = a + 2h$. We also define

$$q(x) = (x - a)(x - m)(x - b),$$

and $p_2(x)$ as the quadratic polynomial interpolating f in a , m , and b .

Exercise 12.10. In this exercise we will look at different ways of proving that Simpson's method is exact for all cubic polynomials.

- (a) Verify that Simpson's method is exact for $f(x) = x^i$ for $i = 0, 1, 2, 3$.

! Solution

It is enough to verify that Simpson's method is exact on each sub-interval. The integrals become

$$\begin{aligned} \int_{a-h}^{a+h} x^3 dx &= \frac{1}{4}((a+h)^4 - (a-h)^4) = 2a^3h + 2ah^3 \\ \int_{a-h}^{a+h} x^2 dx &= \frac{1}{3}((a+h)^3 - (a-h)^3) = \frac{1}{3}(6a^2h + 2h^3) = \frac{h}{3}(6a^2 + 2h^2) \\ \int_{a-h}^{a+h} x dx &= \frac{1}{2}((a+h)^2 - (a-h)^2) = \frac{1}{2}4ah = 2ah \\ \int_{a-h}^{a+h} dx &= 2h. \end{aligned}$$

we also get that the method becomes

$$\begin{aligned} \frac{h}{3}((a-h)^3 + 4a^3 + (a+h)^3) &= \frac{h}{3}(6a^3 + 6ah^2) = 2a^3h + 2ah^3 \\ \frac{h}{3}((a-h)^2 + 4a^2 + (a+h)^2) &= \frac{h}{3}(6a^2 + 2h^2) \\ \frac{h}{3}((a-h) + 4a + (a+h)) &= 2ah \\ \frac{h}{3}(1 + 4 + 1) &= 2h. \end{aligned}$$

This shows that Simpson's method is exact for the given polynomials.

- (b) Use (a) to show that Simpson's method is exact for any cubic polynomial.

! Solution

For $f(x) = bx^3 + cx^2 + dx + e$ the integral becomes

$$\int_{a-h}^{a+h} f(x)dx = b \int_{a-h}^{a+h} x^3 dx + c \int_{a-h}^{a+h} x^2 dx + d \int_{a-h}^{a+h} x dx + e \int_{a-h}^{a+h} 1 dx.$$

The method now becomes

$$\begin{aligned} & \frac{h}{3}(f(a-h) + 4f(a) + f(a+h)) \\ &= b \frac{h}{3}((a-h)^3 + 4a^3 + (a+h)^3) + c \frac{h}{3}((a-h)^2 + 4a^2 + (a+h)^2) \\ &+ d \frac{h}{3}((a-h) + 4a + (a+h)) + e \frac{h}{3}(1 + 4 + 1) \end{aligned}$$

Equality follows now from that we have equality for x^3 , x^2 , x , and 1.

- (c) Could you come to the same conclusion as in (b) by considering the error estimate given by Equation 12.13?

! Solution

Since $f^{(4)}(x) = 0$ for all x for any cubic polynomial, it follows directly from the error estimate that the method is exact for such functions.

- (d) Explain that q is anti-symmetric about m , and that, if f is a cubic polynomial, then $f - p_2$ is a multiple of q . Why does exactness of Simpson's method for cubic polynomial follow from this?

! Solution

We have that

$$\begin{aligned} q(m-y) &= (h-y)(-y)(-h-y) \\ q(m+y) &= (h+y)y(-h+y) = -(h-y)(-y)(-h-y), \end{aligned}$$

so that $q(m-y) = -q(m+y)$. It follows that q is anti-symmetric about m , so that $\int_a^b q(x) dx = 0$. Suppose now that f is a cubic polynomial. The Newton forms are

$$\begin{aligned} p_2(x) &= c_0 + c_1(x-a) + c_2(x-a)(x-m) \\ f(x) &= c_0 + c_1(x-a) + c_2(x-a)(x-m) + c_3(x-a)(x-m)(x-b) \end{aligned}$$

Since $\int_a^b (x-a)(x-m)(x-b) dx = \int_a^b q(x) dx = 0$ it follows that

$\int_a^b f(x) dx = \int_a^b p_2(x)$, so that Simpson's method is exact for all cubic polynomials.

Exercise 12.11 (A sharper error estimate for Simpson's method). In this exercise we will prove a sharper error estimate than Equation 12.13 for Simpson's method. We let

$$E = \int_a^b f(x) dx - \int_a^b p_2(x) dx$$

be the error in Simpson's method with one sub-interval.

(a) Show that $\int_a^b xq(x) dx = -\frac{1}{120}(b-a)^5$, and that $(xq(x))^{(4)} = 4!$.

! Solution

Integration by parts three times gives

$$\begin{aligned} \int_a^b xq(x) dx &= \int_a^b x(x-a)(x-m)(x-b) dx = -\frac{1}{2} \int_a^b (x-a)^2(x-m)(x-b) dx \\ &= \frac{1}{6} \int_a^b (x-a)^3(x-b) dx \\ &= -\frac{1}{24} \int_a^b (x-a)^4 dx \\ &= -\frac{1}{120}(b-a)^5, \end{aligned}$$

which proves the first part. That $(xq(x))^{(4)} = 4!$ follows by using the rule for differentiating a product four times.

(b) Consider the continuous function

$$\phi(x) = f(x) - p_2(x) - kq(x) - \frac{E}{\int_a^b xq(x) dx} xq(x),$$

Show that

$$\phi^{(4)}(x) = f^{(4)}(x) + 120 \frac{4!E}{(b-a)^5}, \quad (12.16)$$

and that $\int_a^b \phi(x) dx = 0$. Finally, explain that $\phi(a) = \phi(m) = \phi(b) = 0$.

! Solution

We have that $p_2^{(4)}(x) = q^{(4)}(x) = 0$. Insertion of $(xq(x))^{(4)} = 4!$, and $\int_a^b xq(x) dx = -\frac{1}{120}(b-a)^5$ from (a), proves the first part. For the second

part, we know from before that $\int_a^b q(x) dx = 0$. Integration of ϕ from a to b therefore gives

$$\int_a^b f(x) dx - \int_a^b p_2(x) dx - E = E - E = 0.$$

The last part follows from that f and p_2 agree on a , m , and b , and from that $q(a) = q(m) = q(b) = 0$.

(c) Show that there exists a k so that $\int_a^m \phi(x) dx = 0$.

! Solution

$\int_a^m \phi(x) dx = 0$ is the same as

$$0 = \int_a^m f(x) dx - \int_a^m p_2(x) dx - k \int_a^m q(x) dx - \frac{E}{\int_a^b xq(x) dx} \int_a^m xq(x) dx,$$

which can be solved for k by dividing with $\int_a^m q(x) dx \neq 0$ (which is $\neq 0$ since q does not have a sign change on (a, m)).

For the last part of the exercise we let k be defined as in (c).

(d) Show that ϕ has at least five different zeros in $[a, b]$.

! Solution

Since $\int_a^b \phi(x) dx = \int_a^m \phi(x) dx = 0$, we must have that $\int_m^b \phi(x) dx = 0$. Since ϕ is continuous, there is a $u \in (a, m)$ so that $\phi(u) = 0$, and a $v \in (m, b)$ so that $\phi(v) = 0$. ϕ therefore has at least five different zeros in $[a, b]$, since we also proved that $\phi(a) = \phi(m) = \phi(b) = 0$.

(e) Rolle's theorem says that, for a continuous, differentiable function defined on $[a, b]$ so that $\phi(a) = \phi(b) = 0$, there exists a point $c \in (a, b)$ so that $\phi'(c) = 0$. Use Rolle's theorem to show that there exists a $\xi \in (a, b)$ so that $\phi^{(4)}(\xi) = 0$.

! Solution

You can argue as follows

1. If you apply Rolle's theorem on the four intervals (a, u) , (u, m) , (m, v) , and (v, b) , we get four points $a_1 < a_2 < a_3 < a_4$, where all $\phi'(a_i) = 0$.

2. If you apply Rolles teorem again, but now on the three intrvals (a_1, a_2) , (a_2, a_3) , and (a_3, a_4) , we get three points $b_1 < b_2 < b_3$, where all $\phi''(b_i) = 0$.
3. If you apply Rolles teorem a third time, now on the two intervals (b_1, b_2) , (b_2, b_3) , we get two points $c_1 < c_2$ so that $\phi'''(c_1) = \phi'''(c_2) = 0$.
4. If you apply Rolles teorem a fourth and final time we get a $\xi \in (c_1, c_2)$ so that $\phi^{(4)}(\xi) = 0$,

- (f) Explain that, with ξ as in (e), we get that $E = -f^{(4)}(\xi)(b-a)^5/2880$. It follows that the error for one interval is $\leq M_4(b-a)^5/2880$. Since we have contribution from n such, we get the upper bound

$$nM_4(2h)^5/2880 = (b-a)M_4(2h)^4/2880 = M_4(b-a)h^4/180$$

for the error. This is considerably better than the global error estimate given in Equation 12.13.

! Solution

Evaluation of Equation 12.16 in ξ gives

$$0 = f^{(4)}(\xi) + 120 \frac{4!E}{(b-a)^5},$$

so that $E = -f^{(4)}(\xi)(b-a)^5/2880$. An upper bound for the error on one interval is therefore $M_4(b-a)^5/2880$.

Exercise 12.12.

- (a) Show that the Newton form of the polynomial interpolating f in a , $m = (a+b)/2$, and b is

$$p_2(x) = f(a) + \frac{f(m) - f(a)}{h}(x-a) + \frac{f(a) - 2f(m) + f(b)}{2h^2}(x-a)(x-m),$$

where $h = (b-a)/2$.

! Solution

The Newton form is obtained by solving the equations

$$f(a) = c_0$$

$$f(m) = c_0 + c_1h$$

$$f(b) = c_0 + 2c_1h + 2c_2h^2.$$

We get $c_0 = f(a)$, $c_1 = \frac{f(m)-f(a)}{h}$, $c_2 = \frac{f(a)-2f(m)+f(b)}{2h^2}$. The interpolating

polynomial is thus

$$p_2(x) = f(a) + \frac{f(m) - f(a)}{h}(x - a) + \frac{f(a) - 2f(m) + f(b)}{2h^2}(x - a)(x - m).$$

(b) Show that $\int_a^b p_2(x) dx = \frac{h}{3} (f(a) + 4f(m) + f(b))$.

 Hint

Compute $\int_a^b (x - a)(x - m) dx$ first, see Exercise 10.12.

 Solution

Here it can be of help to first compute $\int_a^b (x - a)(x - m) dx$ by means of integration by parts:

$$\int_a^b (x - a)(x - m) dx = \left[\frac{1}{2}(x - a)^2(x - m) \right]_a^b - \frac{1}{2} \int_a^b (x - a)^2 dx = 2h^3 - \frac{4}{3}h^3 = \frac{2}{3}h^3.$$

Integrating $p_2(x)$ gives

$$\begin{aligned} & \left[f(a)(x - a) + \frac{f(m) - f(a)}{2h}(x - a)^2 \right]_a^b + \frac{f(a) - 2f(m) + f(b)}{2h^2} \frac{2}{3}h^3 \\ &= 2f(a)h + 2(f(m) - f(a))h + \frac{f(a) - 2f(m) + f(b)}{h} \frac{1}{3}h \\ &= h \left(2f(m) + \frac{1}{3}(f(a) - 2f(m) + f(b)) \right) \\ &= \frac{h}{3} (6f(m) + f(a) - 2f(m) + f(b)) \\ &= \frac{h}{3} (f(a) + 4f(m) + f(b)). \end{aligned}$$

Exercise 12.13. On the interval $[x_{i-1}, x_i]$ the trapezoidal rule uses only the function values $f(x_{i-1})$ and $f(x_i)$. On the next interval the function values $f(x_i)$ and $f(x_{i+1})$ are used. All function values, except the first and last, therefore occur twice in the sum to the right in Equation 12.10. Function evaluations can be costly - they can for example stem from sensors which are expensive to use. Rewrite Equation 12.10 in such a way that we repeat repeating function evaluations.

! Solution

We can rewrite this as follows.

$$I_{\text{trap}}(h) = h \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i) \right).$$

Exercise 12.14. Suppose we know $n + 1$ function values $\{f(a + kh)\}_{k=0}^n$. Show that

$$\int_a^{a+nh} p_n(x) dx = hn \left(f(a) + \begin{pmatrix} \frac{n}{2} & \frac{n^2}{3} & \dots & \frac{n^n}{n+1} \end{pmatrix} \begin{pmatrix} h^1 p_n^{(1)}(a)/1! \\ h^2 p_n^{(2)}(a)/2! \\ \vdots \\ h^n p_n^{(n)}(a)/n! \end{pmatrix} \right)$$

Also show that, for $n = 2$ and $n = 3$, we get in particular

$$\begin{aligned} \int_a^{a+2h} p_2(x) dx &= \frac{h}{3} (f(a) + 4f(a+h) + f(a+2h)) \\ \int_a^{a+3h} p_3(x) dx &= \frac{3h}{8} (f(a) + 3f(a+h) + 3f(a+2h) + f(a+3h)). \end{aligned}$$

💡 Hint

Theorem 11.1 gives us an expression for $\begin{pmatrix} h^1 p_n^{(1)}(a)/1! \\ h^2 p_n^{(2)}(a)/2! \\ \vdots \\ h^n p_n^{(n)}(a)/n! \end{pmatrix}$.

! Solution

We have that

$$\begin{aligned} \int_a^{a+nh} p_n(x) dx &= \int_a^{a+nh} \sum_{k=0}^n \frac{p_n^{(k)}(a)}{k!} (x-a)^k dx = \left[\sum_{k=0}^n \frac{p_n^{(k)}(a)}{(k+1)!} (x-a)^{k+1} \right]_a^{a+nh} \\ &= \sum_{k=0}^n \frac{p_n^{(k)}(a)}{(k+1)!} (nh)^{k+1} \\ &= hn \left(f(a) + \begin{pmatrix} \frac{n}{2} & \frac{n^2}{3} & \dots & \frac{n^n}{n+1} \end{pmatrix} \begin{pmatrix} h^1 p_n^{(1)}(a)/1! \\ h^2 p_n^{(2)}(a)/2! \\ \vdots \\ h^n p_n^{(n)}(a)/n! \end{pmatrix} \right) \end{aligned}$$

which is the expression we wanted. For $n = 2$ Theorem 11.1 gives

$$\begin{pmatrix} h^1 p_2^{(1)}(a)/1! \\ h^2 p_2^{(2)}(a)/2! \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 4 \end{pmatrix}^{-1} \begin{pmatrix} f(a+h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix} = \begin{pmatrix} 2 & -1/2 \\ -1 & 1/2 \end{pmatrix} \begin{pmatrix} f(a+h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix}$$

Inserting this is the expression above we get

$$\begin{aligned} & 2h \left(f(a) + \begin{pmatrix} 1 & 4/3 \end{pmatrix} \begin{pmatrix} 2 & -1/2 \\ -1 & 1/2 \end{pmatrix} \begin{pmatrix} f(a+h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix} \right) \\ &= 2h \left(f(a) + \begin{pmatrix} 2/3 & 1/6 \end{pmatrix} \begin{pmatrix} f(a+h) - f(a) \\ f(a+2h) - f(a) \end{pmatrix} \right) \\ &= 2h \left(\left(1 - \frac{2}{3} - \frac{1}{6}\right) f(a) + \frac{2}{3} f(a+h) + \frac{1}{6} f(a+2h) \right) \\ &= \frac{h}{3} (f(a) + 4f(a+h) + f(a+2h)). \end{aligned}$$

For $n = 3$ Theorem 11.1 gives

$$\begin{aligned} \begin{pmatrix} h^1 p_3^{(1)}(a)/1! \\ h^2 p_3^{(2)}(a)/2! \\ h^3 p_3^{(3)}(a)/3! \end{pmatrix} &= \begin{pmatrix} 1 & 1 & 1 \\ 2 & 4 & 8 \\ 3 & 9 & 27 \end{pmatrix}^{-1} \begin{pmatrix} f(a+h) - f(a) \\ f(a+2h) - f(a) \\ f(a+3h) - f(a) \end{pmatrix} \\ &= \begin{pmatrix} 3 & -3/2 & 1/3 \\ -5/2 & 2 & -1/2 \\ 1/2 & -1/2 & 1/6 \end{pmatrix} \begin{pmatrix} f(a+h) - f(a) \\ f(a+2h) - f(a) \\ f(a+3h) - f(a) \end{pmatrix} \end{aligned}$$

Inserting this is the expression we get

$$\begin{aligned} & 3h \left(f(a) + \begin{pmatrix} 3/2 & 3 & 27/4 \end{pmatrix} \begin{pmatrix} 3 & -3/2 & 1/3 \\ -5/2 & 2 & -1/2 \\ 1/2 & -1/2 & 1/6 \end{pmatrix} \begin{pmatrix} f(a+h) - f(a) \\ f(a+2h) - f(a) \\ f(a+3h) - f(a) \end{pmatrix} \right) \\ &= 3h \left(f(a) + \begin{pmatrix} 3/8 & 3/8 & 1/8 \end{pmatrix} \begin{pmatrix} f(a+h) - f(a) \\ f(a+2h) - f(a) \\ f(a+3h) - f(a) \end{pmatrix} \right) \\ &= 3h \left(\left(1 - \frac{3}{8} - \frac{3}{8} - \frac{1}{8}\right) f(a) + \frac{3}{8} f(a+h) + \frac{3}{8} f(a+2h) + \frac{1}{8} f(a+3h) \right) \\ &= \frac{3h}{8} (f(a) + 3f(a+h) + 3f(a+2h) + f(a+3h)). \end{aligned}$$

Index

- Augmented matrix, 55
- Basis, 83, 86
 - standard, 84
- Block matrix
 - definition, 43
 - determinant, 123
 - in proof of the spectral theorem, 163
 - write a subspace as a null space, 99
- Change of coordinate matrix, 92
- Characteristic
 - equation, 125
 - polynomial, 125
- Coefficient matrix, 35
- Column space, 86
- Column vector, 27
- Component-wise, 7
- Condition number, 166
- Cramer's rule, 123
- Determinant, 106
- DFT matrix, 33
- Diagonal matrix, 29
- Diagonalisable, 133
- Dimension, 88
- Echelon form, 54
 - reduced, 55
- Eigen
 - space, 140
 - value, 125
 - vector, 125
- Elementary
 - matrix, 57
 - row operation, 56
- Equilibrium, 131
- Error
 - absolute, 165
 - relative, 165
- Factorisation
 - Schur, 163
- Farkas lemma, 68
- Fourier
 - basis, 84
- Gaussian elimination, 53
- Hilbert matrix, 32
- Horner's rule, 195
- Identity matrix, 29
- Index of nilpotence, 77
- Inverse, 70
 - left, 69
 - right, 69
- Invertible, 70
- Invertible matrix theorem, 60, 70, 84, 111
- Isomorphism, 90
- Leading one, 53
- Length
 - complex, 13
 - real, 12
- Linear
 - combination, 81
 - dependence relation, 81
 - independence, 81
 - transformation, 39
- Linear differential equation
 - Higher order, 152
- Linear system of differential equations

- First order, 145
- Loglog plot, 208
- Magnification factor, 113
- Matrix, 27
 - idempotent, 142
 - nilpotent, 77
- Minor, 106
- Null space, 86
- Ones matrix, 29
- Operation count
 - definition of determinant, 107
 - Gaussian elimination, 61
 - Horner's rule, 195
 - Matrix multiplication, 50
 - Triangular systems, 181
- Orthogonal, 12
- Orthogonally diagonalisable, 162
- Outer product, 37
- Parallelepiped, 114
- Pivot
 - column, 55
 - element, 55
 - row, 55
 - variable, 55
- Projection, 16
- Projection matrices, 40
- Proof by contradiction
 - All bases have the same number of vectors, 88
 - Eigenvectors for distinct eigenvalues are linearly independent, 131
 - Farkas lemma, 68
 - Reduced echelon form is unique, 89
- Proof by contraposition
 - Condition for linear independence, 82
 - Farkas lemma, 68
- python
 - np.abs, 9
 - np.arange, 5
 - np.block, 11, 42, 190
 - np.conj, 15
 - np.cos, 9
 - np.diag, 29
 - np.exp, 10
 - np.eye, 29
 - np.fft.fft, 34, 190
 - np.linalg.norm, 13
 - np.linalg.cond, 168
 - np.linalg.det, 107
 - np.linalg.eig, 128
 - np.linalg.inv, 71
 - np.linalg.matrix_power, 77
 - np.linalg.solve, 78
 - np.linspace, 5
 - np.log, 10
 - np.max, 6
 - np.maximum, 10
 - np.min, 6
 - np.minimum, 10
 - np.ones, 29
 - np.prod, 6
 - np.real, 190
 - np.reshape, 32
 - np.shape, 28
 - np.sin, 9
 - np.sqrt, 9
 - np.sum, 6
 - np.vdot, 15
 - np.zeros, 29
 - plt.loglog, 208
 - plt.plot, 42
 - plt.semilogx, 171
 - plt.semilogy, 172
 - scp.hilbert, 32
 - sym.Matrix, 55
- Rank, 88
- Rotation matrix, 40
- Row space, 86
- Row vector, 27
- Scalar product
 - complex, 14
 - real, 12
- Schwarz' inequality, 18

- Span, 81
- Square wave, 189
- Subspace, 85

- The spectral theorem, 164
- Transpose, 28
- Triangle inequality, 18
 - reverse, 23
- Triangle wave, 203
- Triangular
 - lower, 108
 - upper, 108

- Vandermonde matrix
 - definition, 33
 - determinant, 121
 - generators, 33
 - in interpolation, 177
 - in numerical differentiation, 212
 - in plotting polynomials, 42
 - inverse, 73
- Vector product, 116
- Vectorisation, 9
- Volume, 113

- Zero matrix, 29