

Numerical Analysis 2


2015

by Omar Lakkis

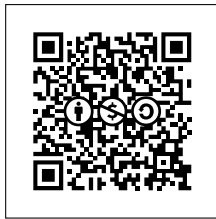


Copyright and copyleft notice

<https://plus.google.com/103353428445025203078>

 Except where otherwise noted, this work is licensed under <http://creativecommons.org/licenses/by-nc-sa/3.0/>

About the cover picture. Photo of Yale Babylonian Collection clay tablet YBC7289 <http://nelc.yale.edu/> illustrating the earliest known calculation of $\sqrt{2}$ using the “divide and average method” (also known as “Heron’s method”).



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported license. <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Copyright (c) 2015 by Omar Lakkis. This work is made available under the terms of the Creative Commons Attribution-Noncommercial-ShareAlike 3.0 license, <http://creativecommons.org/licenses/by-nc-sa/3.0/>.

Contents

Copyright and copyleft notice	1
What's this?	6
Disclaimer	6
Introduction	1
Exact vs. approximate algorithms	1
Chapter 1. Fixed point	7
1.1. Divide and average	7
1.2. Fixed-point theory	9
1.3. But Heron's faster than that!	14
1.4. Notes, pointers and challenges	16
Exercises and problems on Fixed point	17
Chapter 2. Newton's method	21
2.1. Derivations of Newton's method	21
2.2. Convergence analysis	23
2.3. Variants of Newton–Raphson	24
Exercises and problems on Newton's method	25
Chapter 3. Gaussian elimination	29
3.1. Triangular matrices	29
3.2. Solving triangular systems	31
3.3. Gaussian elimination	36
3.4. Pivoting and PLU factorisation	41
Exercises and problems on Linear systems and Gaussian elimination	46
Chapter 4. Matrix analysis	51
4.1. Vector norms	51
4.2. Convergence of matrices	51
Chapter 5. Direct matrix factorisation methods	69
5.1. Reminders	70
5.2. Orthogonality	71
5.3. Schur factorisation	74
5.4. QR factorisation	75
5.5. Cholesky's factorisation	76
Exercises and problems on direct factorisation methods	77
Chapter 6. Basic numerical methods for ODE's	81
6.1. Derivation of the forward Euler method (FE)	81

6.2. Error analysis (of the forward Euler method)	85
6.3. Coding the forward Euler method	93
6.4. The backward (implicit) Euler method	97
6.5. Trapezoidal method	102
6.6. Other higher order methods and variants	108
Exercises and problems on basic numerical methods for ODE's	110
Appendix A. Linear algebra	117
A.1. Algebra	117
A.2. Vector spaces	119
A.3. The model finite dimensional \mathbb{K} -vector space \mathbb{K}^n	120
A.4. Algebras and polynomials	123
A.5. Linear maps	125
A.6. Spectral theory	128
A.7. Multilinear forms	130
A.8. Norms and normed vector spaces	131
A.9. Inner (or scalar) products and Hilbert spaces	132
Exercises and problems on Linear algebra	134
Appendix B. Analysis	135
B.1. Sequences and convergence in \mathbb{R}	135
B.2. Normed vector spaces (topological linear algebra)	136
B.3. Calculus	142
B.4. Continuity	146
B.5. Completeness and Banach spaces	148
B.6. Lipschitz continuity	148
B.7. The Banach–Caccioppoli Contraction Principle	149
B.8. Measurable spaces and measures	151
B.9. Integrals	151
Exercises and problems on Analysis basics	152
Appendix C. A review of ordinary differential equations	153
C.1. ODE's and IVP's	153
C.2. Existence and uniqueness of solutions to IVP's	157
C.3. More examples	160
C.4. Stability	161
Appendix D. Polynomial interpolation	171
D.1. Polynomials and approximation of functions	171
D.2. Interpolation	172
D.3. Newton's divided differences	178
Exercises and problems on polynomial interpolation	182
Appendix E. Least squares polynomial approximation	185
E.1. Linear least-squares approximation	185
E.2. General linear least squares	187
E.3. Orthogonal polynomial based least-squares	189
Exercises and problems on least squares	195

Appendix F. Coding hints	197
F.1. Generalities	197
F.2. Matlab and Octave	197
F.3. Octave	199
F.4. Python: NumPy and SciPy	199
Appendix. Bibliography	203
Appendix. Index	205

What's this?

This is a short introductory (and largely incomplete) introduction to the topic of Numerical Analysis. I use it to teach at Sussex. Hopefully you can find it useful.

Disclaimer

Apart from some isolated parts, these notes are largely digested excerpts of summarising the reading from published sources. I have also used lecture notes from previous courses taught at Sussex taught, in reverse chronological order, by Peter Giesl, Kerstin Hesse, Holger Wendland, David Kay and other people who fade in the fog of twentieth century and will hopefully contact me for adding them to the list.

I therefore have no pretense at originality whatsoever. I do take responsibility for the errors that you may find in the manuscript. If you do find what you think is an error/-typo/mistake, I will be grateful if you sent me an email about it (subject: "Numerical Analysis 2 notes typo") so that things can be fixed for future releases of these notes.

Much of the material in the notes is copyrighted by the authors of the aforementioned sources. The whole booklet is also covered by a Creative Commons attribution-sharealike-noncommercial license, which you should make sure you understand before broadcasting, propagating or otherwise disseminate this material to avoid incurring in legal problems.

Recommended reading list.

Scott, 2011: A good all-rounder (North American advanced undergraduate) which I used as a basis for my courses.

Kelley, 1999: Thorough (second) reading covering linear and nonlinear iterative methods.

Griffiths and Higham, 2010: A solid British undergraduate text covering the essentials of numerical methods for ODE's.

Omar Lakkis

4th March 2015

lakkis.o.maths@gmail.com

Introduction

Numerical methods consist in computing (or approximating) using pen and paper, or more adequately nowadays a computer, the exact solution of an analysis or algebra problem.

Exact vs. approximate algorithms

0.0.1. Example (Euclidean algorithm). An algorithm that is exact is the Euclidean Algorithm (see Lakkis, 2011, §2.5.10 for details) which is given by

```
1: procedure EUCLID( $m, n$ )                                ▷ computes the hcf of  $m$  and  $n$ 
2:    $r_{-1} \leftarrow m, r_0 \leftarrow n, k \leftarrow 1$ 
3:   while  $r_k \neq 0$  do                                    ▷ answer is reached when  $r_k = 0$ 
4:      $r_k := \text{mod}_{r_{k-2}} r_{k-1}$ 
5:      $k \leftarrow k + 1$ 
6:   end while
7:   return  $r_{k-1}$                                           ▷ hcf is the last non-zero remainder
8: end procedure
```

Here $\text{hcf}(m, n)$ is the *highest common factor* (also known as *greatest common divisor*) which is the largest positive integer that divides both m and n with 0 rest. The operation $\text{mod}_s t$ for two integers s and t , $s \neq 0$, is the rest of the Euclidean division by s of t . The Euclidean division is defined by the following result.

THEOREM (Euclidean division). *For each given $n \in \mathbb{Z}$ and $d \in \mathbb{N}$ there exists $q \in \mathbb{Z}$ and $r \in \mathbb{Z}$ such that*

$$n = qd + r \tag{0.0.1}$$

and

$$0 \leq r < d. \tag{0.0.2}$$

The pair (q, r) is unique for each given pair (m, n) .

See Another theorem states that procedure EUCLID terminates and effectively returns the $\text{hcf}(m, n)$. It can be realized in Octave with the following implementation:¹

Printout of file Code/hcfEuclid.m

```
%% -*- mode: octave; -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function rold = hcfEuclid(m,n)
```

¹ Note that both EUCLID and mod are in fact already implemented in Octave, Matlab® and similar environments. The point here (and in the rest of the course) is not to use and apply algorithms but to understand and analyse them.

```

%% function rold = hcfEuclid(m,n)
%%
%% returns the highest common factor (greatest common divisor) of m
%%$ and n
rold = m;
rnew = n;
while(rnew != 0)
    r = rnew;
    [q,rnew] = divEuclid(rold,r);
    rold = r
end

```

and

Printout of file Code/divEuclid.m

```

%% -*- mode: octave -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [q,r] = divEuclid(m,d)
%% function [q,r] = divEuclid(m,d)
%%
%% returns quotient q and rest r of Euclidean division of m by d
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r = m;
q = 0;
while(r>=d)
    r = r-d;
    q = q+1;
end

```

This example has all it takes a numerical algorithm to be complete. We can use as a model for all numerical algorithms. It has a clearly stated *input* consisting of m and n , a well-defined *output* r . By *well-defined output* of an algorithm, we mean that there is a theory that ensures that for each input there is exactly one output (existence and uniqueness). The algorithm is also guaranteed to *terminate* and we can also bound the number of operations by the input. Finally the algorithm is *numerical* because it deals with numbers. This algorithm is *exact*, in the sense that it returns the exact result. We will see later that there are algorithms that are not exact (they are called *approximation algorithms*).

0.0.2. Example (Gaussian elimination). The Gaussian elimination algorithm, taught in linear algebra courses, and which we shall study in Chapter 3. For example, consider the simple 2×2 case, where we are asked to find (x, y) such that

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} e \\ f \end{bmatrix}. \quad (0.0.3)$$

for given $a, b, c, d, e, f \in \mathbb{R}$ and the matrix invertible. Hereafter we use the standard matrix vector multiplication whereby

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}. \quad (0.0.4)$$

Assuming that $a \neq 0$ (else $c \neq 0$ because the matrix is invertible), the Gaussian elimination consists in subtracting c/a times the first row from the second thus obtaining

$$\begin{bmatrix} a & b \\ 0 & d - c/a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} e \\ f - ce/a \end{bmatrix}. \quad (0.0.5)$$

And then solving the resulting system by *back-substitution* to get

$$y = \frac{af - ce}{ad - cb} \text{ and } x = \frac{e - by}{a}. \quad (0.0.6)$$

Assuming we can add, subtract, multiply and divide exactly, this leads to an exact solution in that the computed solution matches the exact solution of the problem. This is why Gaussian elimination and back-substitution are classified as *exact methods*. Note that in practise, while addition, subtraction and multiplication of integers is possible to conduct exactly on a computer, exact division (of integers) becomes cumbersome. Fractions are seldom used as such on a computer and floating-point arithmetic is in fact a more efficient tool for most practical purposes. See Example 0.0.3.

0.0.3. Example (approximate method: “real” division). Real division consists in, given $a \in \mathbb{R}$, to find $x \in \mathbb{R}$ such that $ax = 1$. If $|1 - a| < 1$, i.e., $a \in (0, 2)$ then we know, from basic geometric series, that

$$\frac{1}{a} = \frac{1}{1 - (1 - a)} = \sum_{i=0}^{\infty} (1 - a)^i. \quad (0.0.7)$$

This gives us a way to approximate the reciprocal of a in the form of the *geometric series method* (also known as the *Neumann series method*)

$$\frac{1}{a} \approx \sum_{i=0}^n (1 - a)^i =: x_n, \quad (0.0.8)$$

for some $n \in \mathbb{N}$.

An Octave realisation of this idea is given by

Printout of file Code/neumann1.m

```
%% -*- mode: octave -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [inva,residual] = neumann1(a,N)
b = 1-a;
inva(1) = 1;
residual(1) = inva*a;
if N>0
  for n=1:N
    inva(n+1) = 1 + inva(n)*b;
    residual(n+1) = inva(n+1)*a-1;
  end
end
```

The following allows the testing of neumann1.m

Printout of file Code/neumann1Tester.m

```

%% -*- mode: octave; -*-
%% function [] = neumann1Tester(N)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [] = neumann1Tester(N)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[approximation,residual] = neumann1(.5,N)
error = approximation-2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
timestamp = getTimeStamp();
filename = sprintf("Output/neumann1-%s.csv",timestamp)
file = fopen(filename,"w")
headers = ["step $n$","approximation $x_n$","residual $x_{n-1}$","error←
          ${2-x_n}$"];
out = [1:length(approximation);approximation;residual;error];
fprintf(file,"%s\n",headers);
fprintf(file,"%i,%1.8e,%1.8e,%1.8e\n",out);
fprintf(file,"\n");
fclose(filename)

```

Where we use the following routine to time-stamp the output.

Printout of file Code/getTimeStamp.m

```

%% -*- mode: octave; -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [timestamp] = getTimeStamp()
lt = localtime(time);
lt.year = lt.year-100;
lt.mon = lt.mon+1;
if(lt.year<10)
    ltyear = sprintf("0%i",lt.year);
else
    ltyear = sprintf("%i",lt.year);
end
if(lt.mon<10)
    ltmon = sprintf("0%i",lt.mon);
else
    ltmon = sprintf("%i",lt.mon);
end
if(lt.mday<10)
    ltmday = sprintf("0%i",lt.mday);
else
    ltmday = sprintf("%i",lt.mday);
end
if(lt.hour<10)
    lthour = sprintf("0%i",lt.hour);
else
    lthour = sprintf("%i",lt.hour);
end
if(lt.min<10)
    ltmin = sprintf("0%i",lt.min);
else
    ltmin = sprintf("%i",lt.min);
end
if(lt.sec<10)
    ltsec = sprintf("0%i",lt.sec);
else

```

```

    ltsec = sprintf("%i",lt.sec);
end
timestamp = sprintf("%s%s%s-%s%s%s",ltyear,ltmon,ltmday,lthour,ltmin,↵
    ltsec)
end

```

The resulting output is summarised in the following table:

Column1	Column2	Column3	Column4
step n	approximation x_n	residual $a x_n - 1$	error $2 - x_n$
1	1.00000000e+00	5.00000000e-01	-1.00000000e+00
2	1.50000000e+00	-2.50000000e-01	-5.00000000e-01
3	1.75000000e+00	-1.25000000e-01	-2.50000000e-01
4	1.87500000e+00	-6.25000000e-02	-1.25000000e-01
5	1.93750000e+00	-3.12500000e-02	-6.25000000e-02
6	1.96875000e+00	-1.56250000e-02	-3.12500000e-02
7	1.98437500e+00	-7.81250000e-03	-1.56250000e-02
8	1.99218750e+00	-3.90625000e-03	-7.81250000e-03
9	1.99609375e+00	-1.95312500e-03	-3.90625000e-03
10	1.99804688e+00	-9.76562500e-04	-1.95312500e-03
11	1.99902344e+00	-4.88281250e-04	-9.76562500e-04
12	1.99951172e+00	-2.44140625e-04	-4.88281250e-04
13	1.99975586e+00	-1.22070312e-04	-2.44140625e-04

We will see in Chapters 1 and 2 more about computing the reciprocal. But let us draw some conclusions here:

- The geometric series method provides us with a way to successively approximate the reciprocal of a given real number $a \in (0, 2)$.
- Each iteration, x_n , for $n \in \mathbb{N}_0$, of the geometric series method yields an approximation of 2 that is twice better than x_{n-1} . Further experimentation will show that this improvement factor is in fact equal to $1/(1-a)$.
- While the exact value of $1/a$ is never attained, we can push the iteration until a satisfactory residual is reached.

CHAPTER 1

Fixed point

I believe that we are looking here at the very origins of mathematical reasoning.

— Bill Casselman "Mathematical commentary on YBC 7289"

<http://www.math.ubc.ca/~cass/Euclid/ycb/comments.html>

1.1. Divide and average

A quite old, possibly the oldest, numerical technique, discovered by archaeologists on a clay tablet in Mesopotamia (in current Iraq), is that of computing the square root of a given nonnegative real number y . We start off by revisiting this method, and use it as a model to introduce error analysis.

1.1.1. Computing the square root. This technique goes by the name of *divide and average*, consisting in a successive approximation sequence where an initial guess x_0 is “improved” as follows

$$x_k := \frac{1}{2} \left(x_{k-1} + \frac{y}{x_{k-1}} \right), \quad (1.1.1)$$

for each $k \in \mathbb{N}$. Assuming that the sequence $(x_k)_{k \in \mathbb{N}_0}$ converges, say

$$\hat{x} = \lim_{k \rightarrow \infty} x_k, \quad (1.1.2)$$

then

$$\hat{x} = \frac{1}{2} \left(\hat{x} + \frac{y}{\hat{x}} \right) \quad (1.1.3)$$

which implies

$$\hat{x} = \frac{y}{\hat{x}} \quad (1.1.4)$$

and thus

$$\hat{x}^2 = y, \quad (1.1.5)$$

which means that \hat{x} is a square root of y , as desired.

1.1.2. Why should the limit in (1.1.2) exist? We still have to show that the limit in (1.1.2) does exist. One way of showing this is to look at the sequence of iterates $(x_k)_{k \in \mathbb{N}_0}$ and show that it is a Cauchy sequence (check B.1 for some reminders on sequences of real numbers and convergence, or, more thoroughly, your Analysis lecture notes).

1.1.3. Error analysis. What we develop in this paragraph goes by the name of *error analysis* and is a central idea of Numerical Analysis. The main goal is to understand how the *error* in the approximation of Algorithm (1.1.1), where by error we mean the difference between the exact solution and the approximate solution. Namely, for each $k \in \mathbb{N}_0$ we define the error committed at the k -th step of the iteration (1.1.1) to be

$$e_k := \hat{x} - x_k \text{ where } \hat{x}^2 = y \text{ and } x_k \text{ given by (1.1.1).} \quad (1.1.6)$$

Noting that *both* x_k and x_{k-1} in (1.1.1) may be replaced by \hat{x} , leaving the identity true we have that

$$\begin{aligned} e_k &= \frac{1}{2} \left(\hat{x} + \frac{y}{\hat{x}} - x_{k-1} - \frac{y}{x_{k-1}} \right) \\ \text{(algebra)} \quad &= \frac{1}{2} (\hat{x} - x_{k-1}) - \frac{y}{2} \left(\frac{1}{\hat{x}} - \frac{1}{x_{k-1}} \right) \\ \text{(definition of } e_k, \text{ algebra and } y = \hat{x}^2) \quad &= \frac{e_{k-1}}{2} \left(1 - \frac{\hat{x}^2}{\hat{x} x_{k-1}} \right) \\ \text{(more algebra)} \quad &= \frac{e_{k-1} (x_{k-1} - \hat{x})}{2 x_{k-1}} \\ &= \frac{-(e_{k-1})^2}{2 x_{k-1}}. \end{aligned} \quad (1.1.7)$$

The square appearing in the numerator in the previous term is *very precious*. Recalling that for any $\epsilon \in \mathbb{R}$ we have

$$|\epsilon| < 1 \Rightarrow \epsilon^2 < \epsilon, \quad (1.1.8)$$

and assuming for a moment that

$$1/2 x_{k-1} \leq 1 \quad \forall k \geq 1, \quad (1.1.9)$$

it follows that

$$|e_k| \leq |e_{k-1}|^2. \quad (1.1.10)$$

So if $e_0 < 1$ it follows, by induction, that $e_k < e_0^{2^k}$ and thus by (1.1.7) we have

$$\lim_{k \rightarrow \infty} e_k = 0. \quad (1.1.11)$$

This can be formalised by the following result.

1.1.4. Proposition (convergence of Heron's algorithm). *Let $y \in [1/2, 2]$, $\hat{x} := \sqrt{y}$, and*

$$x_0 \in I := \begin{cases} (1, y) & \text{if } y > 1, \\ \{1\} & \text{if } y = 1, \\ (y, 1) & \text{if } y < 1. \end{cases} \quad (1.1.12)$$

If x_k , for $k \in \mathbb{N}$, is defined by (1.1.1), then for a C_y , that depends only on y , we have

$$|\hat{x} - x_k| \leq C_y |y| |\hat{x} - x_{k-1}|^2 \text{ and } x_k \in I \quad \forall k \in \mathbb{N}. \quad (1.1.13)$$

The sequence $(x_k)_{k \in \mathbb{N}_0}$ converges to $\hat{x} = \sqrt{y}$.

Proof We just need to fill in the gaps in the discussion preceding the statement. Note that since $y \in I$, we have

$$\hat{x} \in I \quad (1.1.14)$$

(if $\hat{x} \notin I$ it can be seen that $\hat{x}^2 \notin I$). Define e_k as in (1.1.6).

Assume first that $y > 1$, and thus $y > \hat{x} > 1$. Let us proceed by induction. Since $x_0 \in I$ by assumption, it follows that $x_0 \geq 1$ and thus by (1.1.7) that

$$|e_1| = \frac{(e_0)^2}{2x_0} \leq \frac{1}{2}(e_0)^2. \quad (1.1.15)$$

Since $x_0 \in (1, y)$ and $y \leq 2$ it follows that $|e_0| < 1$ and thus $|e_1| < |e_0|$, which means that the error is reduced from the first step. Furthermore, since $1 < x_0 < y$, we have

$$x_1 = \frac{1}{2} \left(x_0 + \frac{y}{x_0} \right) > \frac{1}{2}(1 + 1) = 1 \quad (1.1.16)$$

and

$$x_1 < \frac{1}{2}(y + y) = y, \quad (1.1.17)$$

hence $x_1 \in I$. This proves (1.1.13) for $k = 1$, i.e., the base case of the induction. To prove the inductive step, fix $k \geq 1$ and assume

$$x_k \in I. \quad (1.1.18)$$

Then arguing exactly as above with k replacing 0 and $k + 1$ replacing 1, obtain

$$|e_{k+1}| \leq \frac{1}{2}|e_k|^2 < 1 \quad (1.1.19)$$

and $x_{k+1} \in I$, as desired. By induction, the result follows with $C_y = \frac{1}{2}$.

The case $y = 1$ and $y \in [1/2, 1)$ is similar, with a slightly worse and more delicate treatment of C_y . It is left as an exercise. \square

1.2. Fixed-point theory

The analysis developed in §1.1 is a special case of a more systematic approach to the analysis of iterative methods known as *fixed-point* theory. Although we are mainly preoccupied with the real numbers \mathbb{R} (or intervals thereof) here, this theory can be developed without pain for general metric spaces. We refer to §B.7 in Appendix B for more details on the more general case.

1.2.1. Definition of fixed point. Given a function $f : D \rightarrow D$, where D is a given set, we say that x is a fixed point of f if and only if

$$x = f(x). \quad (1.2.1)$$

1.2.2. Example.

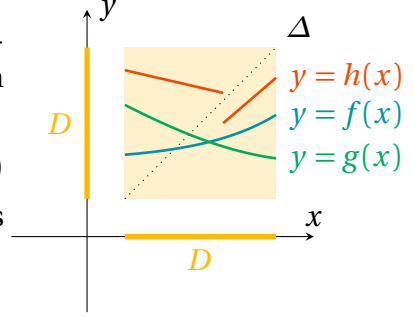
- (a) Let $D := \mathbb{R}$ and $f(x) := x^2$. In this case we can check algebraically that 0 and 1 are both fixed points of x .
- (b) Let $D := \mathbb{R}$ and $f(x) := \cos x$. This is a bit harder (in fact, impossible) to check algebraically, but one can analytically show (using the intermediate value theorem, Rolle's theorem and some simple trigonometric bounds) that f has exactly one fixed point lying between 0 and $\pi/2$ (see Problem ??).
- (c) Taking $f(x) := \exp(x)$ for $x \in D := \mathbb{R}$.

1.2.3. Remark (graphical interpretation in \mathbb{R}). Let $f : D \rightarrow D$ with $D \subseteq \mathbb{R}$.

Then solving the fixed-point problem for f is equivalent to the geometric problem of finding the intersection of the diagonal and the graph of the function:

$$\Delta := \{(x, x) : x \in D\} \text{ and } \{(x, f(x)) : x \in D\}. \quad (1.2.2)$$

In the picture we plot three examples of functions $f, g, h : D \rightarrow D$, where $D = [1/2, 5/2]$.



With reference to the picture:

- ★ The function $f(x) := \exp x/20 + 1$ for $x \in D$ has a fixed point in D as it intersects the diagonal.
- ★ The function $g(x) := 2 - \sin(\pi x/6)$ for $x \in D$ has also a fixed point.
- ★ The function $h(x) := -(3/13)x + 301/130$ for $x < 9/5$ and $(6/7)x - 3/70$ for $x \geq 9/5$ does not have a fixed point.

1.2.4. Exercise (contractions are continuous). Prove that a contraction $g : D \rightarrow \mathbb{R}$ must be continuous.

1.2.5. Theorem (Banach–Caccioppoli's contraction principle). Let D be a closed subset of \mathbb{R} . Let $g : D \rightarrow D$ be a contraction with constant $\kappa \in [0, 1)$ then g has exactly one fixed point $\hat{x} \in D$, i.e.,

$$\hat{x} = g(\hat{x}), \quad (1.2.3)$$

and there is no other point in D satisfying (1.2.3).

Furthermore if $x_0 \in D$ and for each $k \in \mathbb{N}$ we define the iterates

$$x_k := g(x_{k-1}), \quad (1.2.4)$$

then $(x_k)_{k \in \mathbb{N}_0}$ converges and $\lim_{k \rightarrow \infty} x_k = \hat{x}$, the following error reduction between successive iterates holds

$$|x_k - \hat{x}| \leq \kappa |x_{k-1} - \hat{x}| \quad \forall k \in \mathbb{N} \quad (1.2.5)$$

and the following (apriori) error bound

$$|x_k - \hat{x}| \leq |x_0 - \hat{x}| \kappa^k \quad \forall k \in \mathbb{N}, \quad (1.2.6)$$

and the following (aposteriori) error estimate

$$|x_k - \hat{x}| \leq \frac{|x_1 - x_0|}{1 - \kappa} \kappa^k \quad \forall k \in \mathbb{N}. \quad (1.2.7)$$

Let D be a closed subset of \mathbb{R} . Let $g : D \rightarrow D$ be a contraction with constant $\kappa \in [0, 1)$ then g has exactly one fixed point $\hat{x} \in D$, i.e.,

$$\hat{x} = g(\hat{x}), \quad (1.2.8)$$

and there is no other point in D satisfying (1.2.8).

Furthermore if $x_0 \in D$ and for each $k \in \mathbb{N}$ we define the iterates

$$x_k := g(x_{k-1}), \quad (1.2.9)$$

then $(x_k)_{k \in \mathbb{N}_0}$ converges and $\lim_{k \rightarrow \infty} x_k = \hat{x}$, the following error reduction between successive iterates holds

$$|x_k - \hat{x}| \leq \kappa |x_{k-1} - \hat{x}| \quad \forall k \in \mathbb{N} \quad (1.2.10)$$

and the following (apriori) error bound

$$|x_k - \hat{x}| \leq |x_0 - \hat{x}| \kappa^k \quad \forall k \in \mathbb{N}, \quad (1.2.11)$$

and the following (aposteriori) error estimate

$$|x_k - \hat{x}| \leq \frac{|x_1 - x_0|}{1 - \kappa} \kappa^k \quad \forall k \in \mathbb{N}. \quad (1.2.12)$$

Proof Note that for each $k \in \mathbb{N}$ we have

$$|x_{k+1} - x_k| = |g(x_k) - g(x_{k-1})| \leq \kappa |x_k - x_{k-1}|. \quad (1.2.13)$$

It follows, by induction on k , that

$$|x_{k+1} - x_k| \leq \kappa^k |x_1 - x_0|, \quad (1.2.14)$$

whence for any $k, l \geq 0$

$$\begin{aligned} |x_{k+l} - x_k| &= \left| \sum_{i=0}^{l-1} x_{k+i+1} - x_{k+i} \right| \\ &\leq \sum_{i=0}^{l-1} |x_{k+i+1} - x_{k+i}| \leq |x_1 - x_0| \sum_{i=0}^{\infty} \kappa^{k+i} \\ &\leq \frac{|x_1 - x_0|}{1 - \kappa} \kappa^k. \end{aligned} \quad (1.2.15)$$

Since the right hand converges to 0 as $k \rightarrow \infty$, it follows that $(x_k)_{k \in \mathbb{N}_0}$ is a Cauchy sequence. By the Cauchy criterion the sequence converges to a limit $\hat{x} := \lim_{k \rightarrow \infty} x_k$. Recalling that a contraction such as g must be continuous (see 1.2.4), it follows that

$$\hat{x} = \lim_{k \rightarrow \infty} x_k = \lim_{k \rightarrow \infty} g(x_{k-1}) = \lim_{k \rightarrow \infty} g(x_k) = g(\lim_{k \rightarrow \infty} x_k) = g(\hat{x}), \quad (1.2.16)$$

which means that \hat{x} is a fixed point of g .

Suppose now that \check{x} is another fixed point of g , then

$$|\hat{x} - \check{x}| = |g(\hat{x}) - g(\check{x})| \leq \kappa |\hat{x} - \check{x}|. \quad (1.2.17)$$

Whence $(\kappa - 1)|\hat{x} - \check{x}| \geq 0$, which, jointly with $\kappa < 1$, implies $|\hat{x} - \check{x}| = 0$ and thus $\check{x} = \hat{x}$. So the fixed point \hat{x} is indeed unique. \square

1.2.6. Definition of Lipschitz continuity. Contractivity of a function g (i.e., its being a contraction) is a subcase of g 's being a *Lipschitz continuous* one, where this is defined as

$$-L(y - x) \leq g(y) - g(x) \leq L(y - x) \quad \forall x, y \in D, \quad (1.2.18)$$

for some $L \in \mathbb{R}^+$ independent of the x and y ; the two inequalities above can be merged into one using the absolute value

$$|g(y) - g(x)| \leq L|y - x| \quad \forall x, y \in D. \quad (1.2.19)$$

In this case we write $g \in \text{Lip}(D)$.

If g is a Lipschitz function then we its *Lipschitz constant on D* is the smallest L that satisfies (1.2.19), in symbols

$$|g|_{\text{Lip}(D)} := \sup_{D \ni x \neq y \in D} \left| \frac{g(y) - g(x)}{y - x} \right| \quad (1.2.20)$$

1.2.7. Exercise. Prove that if $g \in C^1(D)$ and C is a compact (i.e., closed and bounded in \mathbb{R}) subset of D then $g \in \text{Lip}(C)$ and

$$|g|_{\text{Lip}(C)} = \sup_C |g'| = \max_C |g'|. \quad (1.2.21)$$

(The last equality is just a formulaic way of saying that the least upper bound is achieved by some point $\check{x} \in C$: $|g'(\check{x})| = \sup_C |g'|$.)

Hint. Use the mean value theorem, in the integral form,

$$g(y) - g(x) = \left(\int_0^1 g'(x + \theta(y - x)) d\theta \right) (y - x), \quad (1.2.22)$$

and basic properties of the integral, such as the triangle inequality

$$\left| \int f \right| \leq \int |f| \quad (1.2.23)$$

and monotonicity

$$f \leq h \Rightarrow \int f \leq \int h. \quad (1.2.24)$$

1.2.8. Problem. Let y be a fixed positive number, show that the transformation

$$x \mapsto \frac{1}{2} \left(x + \frac{y}{x} \right) =: g(x) \quad (1.2.25)$$

is a contraction in a closed ball around \sqrt{y} .

Use the Banach–Caccioppoli Contraction Principle to conclude that for suitably chosen x_0 the sequence $(x_n)_n$ of iterates

$$x_k := g(x_{k-1}) \quad (1.2.26)$$

converges, as $k \rightarrow \infty$, to \hat{x} satisfying

$$\hat{x}^2 = y. \quad (1.2.27)$$

1.2.9. Example (applying Banach–Caccioppoli).

PROBLEM. Consider the problem of finding a fixed-point $x \in \mathbb{R}$ for each of the following functions (separately)

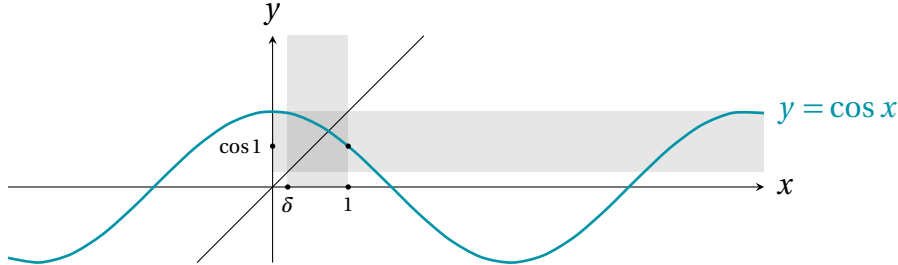
$$g(x) := \cos x \quad (1.2.28)$$

- (a) Verify analytically, but without using the Banach–Caccioppoli contraction principle, that this function has a unique fixed point on the real line.
- (b) Can uniqueness on the whole real line be deduced using the Banach–Caccioppoli contraction principle? Explain why, or why not.

- (c) What rate (order) of convergence do you expect from the iteration $x_n := g(x_{n-1})$, assuming it converges? Explain why.
- (d) Starting from $x_0 = 0$, compute, using only a scientific calculator as many iterations as needed as to ensure 4 digits of accuracy in the residual.

Solution.

(a) Let us start with a plot to gain an intuitive idea of what we want. Basic trigonometry is sufficient to draw the following diagram.



We want to prove the existence of \hat{x} such that $\hat{x} = g(\hat{x})$. Define $h(x) := g(x) - x$. Graphically it appears that h has a unique root in \mathbb{R} . To make this argument precise note first that

$$h(0) = \cos 0 - 0 = 1 > 0 \text{ and } h(\pi/2) = \cos(\pi/2) - \pi/2 = 1 - \pi/2 < 1 - 3/2 = 1/2 < 0. \quad (1.2.29)$$

By the intermediate value theorem there exists \hat{x} between 0 and $\pi/2$ for which $h(\hat{x}) = 0$. This proves existence. To prove uniqueness, suppose that for some $\check{x} \neq \hat{x}$ we have $h(\check{x}) = 0$, then \check{x} must lie between $-\pi/2$ and $\pi/2$, since, using $|\cos x| \leq 1$, we have

$$\begin{aligned} x \leq -\pi/2 &\Rightarrow h(x) \geq -1 + \pi/2 > 0 \\ x \geq \pi/2 &\Rightarrow h(x) \leq 1 - \pi/2 < 0. \end{aligned} \quad (1.2.30)$$

It follows, by Rolle's Theorem, that for some $|\xi| < \pi/2$ we have $0 = h'(\xi) = 1 + \sin \xi$, whence $\sin \xi = -1$, which implies

$$\xi = -\pi/2 + 2k\pi \text{ for some } k \in \mathbb{Z}. \quad (1.2.31)$$

That's $\xi = \dots, -\pi/2, 3\pi/2, \dots$, which is in contradiction with $|\xi| < \pi/2$.

(b) Uniqueness *on the whole real line* cannot be deduced from the Banach–Caccioppoli contraction principle alone because g is not a contraction over \mathbb{R} . (Just about as $|g'(x)| \leq 1$ but *not* $|g'(x)| < 1$.)

(c) Note that if $|x| \leq 1$ then $0 < g(x) \leq 1$. Also for any $x \in \mathbb{R}$ we have $|g(x)| \leq 1$ and hence $0 < g(g(x)) \leq 1$. It follows that the iterates satisfy $0 < x_n \leq 1$ for all $n \geq 2$ for *any choice* of initial value x_0 . It follows that we may apply Banach–Caccioppoli to the function g on an interval $[\delta, 1]$ with a small enough $\delta > 0$. To be sure, take $\delta := \cos 1 \approx 0.54030$. Since $g'(x) = -\sin x$ is negative and decreasing in x on $[0, \pi/2]$, then for $\delta \leq x \leq 1$ we have $0 > g'(\delta) \geq g'(x) \geq g'(1) > -1$. It follows that g is *contractive* on $[\delta, 1]$. Also $g(\delta) \leq 1$ and $g(1) \geq \delta$ which implies that $g([\delta, 1]) \subseteq [\delta, 1]$, which implies that g is a *contraction* on $[\delta, 1]$. We conclude using the Banach–Caccioppoli contraction principle that g must have a unique fixed point in $[\delta, 1]$.

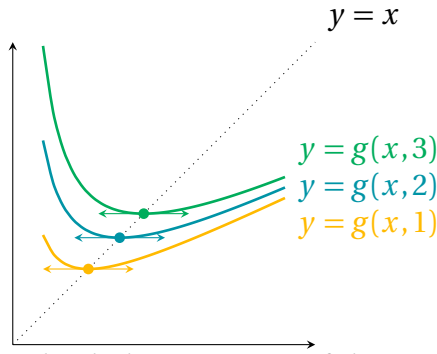
(d) Starting with $x_0 = 0$ it takes 30 punches of cos on the calculator to have the 4-th digit not change anymore.

1.3. But Heron's faster than that!

Banach–Caccioppoli's contraction principle 1.2.5 predicts a *linear convergence rate* for the error, i.e., the error at timestep k is a fraction of the previous error. But in Heron's algorithm exhibits an *quadratic convergence rate* and this is backed by numerical experiments. In fact, a closer inspection of the contraction principle shows that it only gives us an upper bound which is valid for general situations and this bound may be actually better in specific situations, such as Heron. We try to understand this now.

1.3.1. Agraphical interpretation of Heron. It is always instructive to plot something, if possible. Let us look at the graph of Heron's divide-and-average iterative function for finding \sqrt{b} :

$$x \mapsto g(x, b) = \frac{x + b/x}{2}, \quad (1.3.1)$$



(As an example for the plot, we take $b = 2, 3, 1$)
Computing the derivative of g ,

$$g'(x, b) := \partial_x g(x, b) = \frac{1 - b/x^2}{2} \quad (1.3.2)$$

we see that *the slope of the graph at the fixed point is $g'(\sqrt{b}, b) = 0$* . This observation is the key to understanding the higher than expected speed of convergence.

Indeed, the convexity of the graph of $g(\cdot, b)$ means that \sqrt{b} is a minimum of $g(\cdot, b)$. It follows that in a sequence of iterates $x_k = g(x_{k-1}, b)$ for $k \in \mathbb{N}_0$, for *any choice of x_0 , the first iterate x_1 is larger than \sqrt{b} and for any $k \geq 1$ we have $\sqrt{b} \leq x_k \leq x_{k-1}$* .[*] It follows that for each $n \geq 1$ the sequence $(x_k)_{k \geq n}$ has all its terms contained in $I_n := [\sqrt{b}, x_n]$. Looking at the restriction of the function g to I_n , $g(\cdot, b)|_{I_n} : I_n \rightarrow \mathbb{R}$ we see that $g(I_n) \subseteq I_n$ (g maps I_n into itself) and that

[*]: Check!

$$\sup_{x \in I_n} |g'(x, b)| \leq |g'(x_n, b)| = \left| \frac{1 - b/x_n^2}{2} \right| = \left| \frac{x_n^2 - b}{2x_n^2} \right| \rightarrow 0, \text{ as } n \rightarrow \infty. \quad (1.3.3)$$

This means that $g(\cdot, b)$ yields a contraction on each I_n with contraction constant ever smaller and actually converging to zero. By the Banach–Caccioppoli contraction principle (applied on I_n to the sequence $(x_k)_{k \geq n}$), we obtain that

$$|x_k - \sqrt{b}| \leq \kappa_n^{k-n} |x_n - \sqrt{b}|, \text{ where } \kappa_n := \left| \frac{x_n^2 - b}{2x_n^2} \right| \quad \forall k \geq n \geq 1. \quad (1.3.4)$$

In particular, after taking $k = n + 1$, we obtain

$$|x_{n+1} - \sqrt{b}| \leq \left| \frac{x_n^2 - b}{2x_n^2} \right| |x_n - \sqrt{b}| = \frac{x_n + \sqrt{b}}{2x_n^2} |x_n - \sqrt{b}|^2 \quad \forall n \geq 1, \quad (1.3.5)$$

which means that the error (in absolute value) at a given iteration $(k + 1)$ is smaller than a proportional the error (in absolute value) at the previous timestep *squared*.

Furthermore, recalling that $\sqrt{b} \leq x_n$ for $n \geq 1$, it follows that

$$\kappa_n \leq \frac{1}{\sqrt{b}} =: \lambda, \quad (1.3.6)$$

which means that if $b > 1$, and writing $\epsilon_n := |x_n - \sqrt{b}|$, we have an error reduction of the form:

$$\epsilon_{n+1} \leq \lambda \epsilon_n^2 \quad \forall n \geq 1, \quad (1.3.7)$$

for some $\lambda < 1$ *independent of n* . Taking logarithms on both sides we obtain

$$\log \epsilon_{n+1} \leq 2 \log \epsilon_n + \log \lambda \leq 2 \log \epsilon_n \quad \forall n \geq 1. \quad (1.3.8)$$

It follows that for N such that $\epsilon_N < 1$ (existence of such N is guaranteed by Banach–Caccioppoli) and writing $d_n := \lceil \log \epsilon_n \rceil$ (motivated by d_n 's being proportional to the number of significant digits of the approximation of x_n to \sqrt{b}) we have

$$d_{n+1} \geq 2d_n \quad \forall n \geq N, \quad (1.3.9)$$

which means that the number of accurate digits doubles (independent of the basis chosen to represent numbers) at each iteration of Heron's algorithm. Note that a mere application of Banach–Caccioppoli implies

$$d_{n+1} \geq d_n + c, \quad (1.3.10)$$

for some $c = \lceil \log \kappa \rceil$, which is true yet much less precise a bound than (1.3.9).

This explains the really fast convergence of Heron's algorithm and it shows that the reason behind it is the fact that $g'(\sqrt{b}) = 0$ for $g(x) = (b/x + x)/2$. This excellent convergence behaviour is a special case of a more general situation that is outlined in Theorem 1.3.4 and constitutes the basis for Newton's method of Chapter 2.

1.3.2. Definition of linear convergence, linear error reduction. A sequence $(x_k)_{k \in \mathbb{N}_0}$ that converges to a limit \hat{x} is said to *converge linearly*, or to have *linear convergence*, if for some $N \in \mathbb{N}_0$ and $\kappa \in [0, 1)$ the following *linear error reduction* inequalities are satisfied

$$|x_{k+1} - \hat{x}| \leq \kappa |x_k - \hat{x}| \quad \forall k \geq N. \quad (1.3.11)$$

1.3.3. Definition of quadratic convergence, quadratic error reduction. A sequence $(x_k)_{k \in \mathbb{N}_0}$ that converges to a limit \hat{x} is said to *converge quadratically*, or to have *quadratic convergence*, if for some $N \in \mathbb{N}_0$ and some $C > 0$ the following *quadratic error reduction* inequalities are satisfied

$$|x_{k+1} - \hat{x}| \leq C |x_k - \hat{x}|^2 \quad \forall k \geq N. \quad (1.3.12)$$

1.3.4. Theorem (quadratic Banach–Caccioppoli). Suppose $g : D \rightarrow D$ is a contraction in a closed $D \subseteq \mathbb{R}$, with fixed point \hat{x} . Suppose also that $g \in C^1(D, D)$ and $g' \in \text{Lip}(D)$. If $g'(\hat{x}) = 0$ then for some the convergence of iterations is quadratic, i.e., for some $N \in \mathbb{N}$

$$|x_{k+1} - \hat{x}| \leq \frac{\|g'\|_{\text{Lip}(D)}}{2} |x_k - \hat{x}|^2 \quad \forall k \geq N. \quad (1.3.13)$$

Proof Denote $e_k := x_k - \hat{x}$ and $\epsilon_k = |e_k|$. By the mean value theorem we have

$$\epsilon_{k+1} = |x_{k+1} - \hat{x}| = |g(x_k) - g(\hat{x})| = \left| \int_0^1 g'(\hat{x} + \theta e_k) d\theta e_k \right| \quad (1.3.14)$$

Further, $g' \in \text{Lip}(D)$, and say $L := |g'|_{\text{Lip}(D)}$, and $g'(\hat{x})$ imply that

$$|g'(\hat{x} + \theta e_k)| = |g'(\hat{x} + \theta e_k) - g'(\hat{x})| = L\theta|e_k|. \quad (1.3.15)$$

Hence, from (1.3.14), the triangle inequality for integrals and (1.3.15) we obtain, as claimed,

$$\epsilon_{k+1} \leq \frac{L}{2} \epsilon_k^2. \quad (1.3.16)$$

□

1.4. Notes, pointers and challenges

1.4.1. Other fixed-point results. There are many other fixed-point theorems in mathematics. The most famous and celebrated is Brouwer fixed point, as described, e.g., in Karamardian, 1977 or Istratescu, 1981, which states that

a continuous function $f : D \rightarrow D$ from a convex, closed and bounded set D into itself must have a fixed-point.

Brouwer's fixed-point result is very useful in theories such as algebraic topology, differential equations, dynamical systems and game theory. It has also some use in applications. Although Brouwer's initial proof is not constructive, and therefore useless for any practical purpose¹ a constructive proof and an algorithm that finds one of the fixed-points is given by Scarf (1967).

Another useful fixed-point theorem is that of Schauder.

1.4.2. The (false) power of series. Power series expression can be useful in some cases, but their role is quite marginal when it comes to systematic numerical methods. The main problem with power series, or series in general, is that their speed of convergence can be arbitrarily slow.

As an example, a famous problem of mechanics which requires numerical solution is the many body problem, popularised as the “ n -body problem”. While the problem can be solved in “closed form” for $n = 2$, a power series for this problem has been proposed by Sundman (1913) for $n = 3$ and one for $n \geq 2$ by Babadzanjan (1979) and Wang (1991). However, all these “explicit solutions” have limited practical value as the series are known to converge extremely slowly.

A nice account of this problem, its intriguing history, the dangers of over-popularising mathematics, and how numerical analysis cuts to the chase can be found in Diacu (1996).

¹Brouwer was a fervent oppositor of non-constructive proofs, especially proofs by contradiction and was known to reject papers solely on that basis. This led him to have problems with the mathematical establishment of his times, especially Hilbert. Poincaré privately shared many of Brouwer's ideas, but there is no evidence of public manifestations and Brouwer was effectively isolated as a result of his maverick position.

Exercises and problems on Fixed point

Problem 1.1 (Neumann's series for scalars). Let a be a positive real number, find a method to approximate $1/a$ without ever having to divide, with a sequence x_k such that $x_k \rightarrow 1/a$. Explain why your solution works and find as much information as you can about the error.

Hint. Think of a geometric series whose sum is $1/a$.

Problem 1.2 (Heron's divide and average algorithm). Let $y \in [1/2, 2]$. Consider the iteration

$$x_0 \in \mathbb{R} \text{ and } x_{k+1} := \frac{1}{2} \left(x_k + \frac{y}{x_k} \right), \text{ for } k \in \mathbb{N}_0. \quad (\text{P1.2.1})$$

- (a) Show that $x_k^2 \geq y$ for all $k \geq 1$. Deduce $x_k^2 \geq 1/2$ for all $k \in \mathbb{N}_0$.
- (b) Show that

$$|e_{k+1}| \leq \frac{1}{\sqrt{2}} |e_k|^2 \text{ for all } k \in \mathbb{N}_0, \quad (\text{P1.2.2})$$

and hence that the sequence converges if the initial error e_0 is smaller than 1.

- (c) Is the assumption $y \in [1/2, 2]$ necessary?

Problem 1.3 (fixed point iteration). Suppose that $g \in C^1(D; D)$ for a closed interval $D \subseteq \mathbb{R}$, with $\hat{x} \in D$ such that

$$\hat{x} = g(\hat{x}) \text{ and } |g'(\hat{x})| < 1. \quad (\text{P1.3.1})$$

- (a) Prove that there exists a $\delta > 0$ such that $|g'(x)| < 1$ for all $x \in [\hat{x} - \delta, \hat{x} + \delta] \cap D =: D_\delta$.
- (b) Show that δ can be chosen so that $g(D_\delta) \subseteq D_\delta$.
- (c) Deduce that if $x_0 \in D_\delta$ and $x_k = g(x_{k-1})$, for $k \in \mathbb{N}$, then $x_k \in D_\delta$ for all $k \in \mathbb{N}_0$ and $x_k \rightarrow \hat{x}$.

Exercise 1.4 (coding the fixed point method). Given a Lipschitz-continuous function $g : D \rightarrow D$, D closed subset of \mathbb{R} .

- (a) Write a code that will take as input g , x_0 , M and ϵ , and return output x_K for some $K \in \mathbb{N}$ such that

$$x_k := g(x_{k-1}) \text{ for } k = 1, \dots, K, K \leq M \text{ and } |f(x_K)| < \epsilon. \quad (\text{P1.4.1})$$

Here M is the maximum number of allowed iterations (so that any infinite, or simply too long, iteration is broken) and ϵ a given tolerance to be reached.

For test purposes make the code return a whole array of values x_0, x_1, \dots, x_K and the corresponding residuals $x_0 - x_1, x_1 - x_2, \dots, x_{K-1} - x_K$.

Hint. g should be passed as a string, and use the octave/matlab command `inline`

- (b) Write a driver code that will call the code above and will take as inputs the function g (as a string) and an initial guess x_0 .
- (c) Test your code (and use the built-in command `fsolve` to compute the error) on the following benchmark cases
 - (i) $g(x) = \cos x$ and $D = [0, \pi/2]$
 - (ii) $g(x) = x^{1/2}/2$ and $D = [0.1, 0.9]$

Problem 1.5 (polynomial root finder). Let the integer $p > 1$ and consider the iteration

$$x_{k+1} := \frac{1}{p} \left((p-1)x_k + \frac{y}{x_k^{p-1}} \right). \quad (\text{P1.5.1})$$

- (a) If the sequence $(x_k)_{k \in \mathbb{N}_0}$ converges to x , find the relationship between x and y .
- (b) Explore, by writing a small code, the speed of convergence for various starting values x_0 and various values of p .
- (c) Find an error relation between two successive iterates.

Exercise 1.6 (contractions are continuous). Prove that a contraction $g : D \rightarrow \mathbb{R}$ must be continuous.

Problem 1.7 (iterative method computing a scalar's reciprocal). Given $a > 0$, consider the iteration

$$x_{k+1} := 2x_k - ax_k^2. \quad (\text{P1.7.1})$$

- (a) If the sequence $(x_k)_{k \in \mathbb{N}_0}$ converges to x , find the relationship between x and a .
- (b) Explore, by writing a small code, the speed of convergence for various starting values x_0 .
- (c) Find an error relation between two successive iterates.
- (d) Compare this method.

Exercise 1.8. (a) Suppose $g \in C^0(D; D)$ with D a closed subset of \mathbb{R} , define the sequence $(x_k)_{k \in \mathbb{N}_0}$ with

$$x_0 \in D \text{ and } x_k := g(x_{k-1}) \text{ for } k \geq 1, \quad (\text{P1.8.1})$$

and suppose $\lim_{k \rightarrow \infty} x_k = \hat{x}$. Show that \hat{x} is a fixed point of g .

- (b) Suppose that $g \in C^1(D)$ and that it satisfies for some $\lambda > 0$

$$|g(x) - g(y)| < \lambda |x - y| \text{ for all } x, y \in D, \quad (\text{P1.8.2})$$

show that

$$|g'(x)| < \lambda \text{ for all } x \in D. \quad (\text{P1.8.3})$$

Exercise 1.9. Let y be a fixed positive number, show that the transformation

$$x \mapsto \frac{1}{2} \left(x + \frac{y}{x} \right) =: g(x) \quad (\text{P1.9.1})$$

is a contraction in a closed ball around \sqrt{y} .

Use the Banach–Caccioppoli Contraction Principle to conclude that for suitably chosen x_0 the sequence $(x_n)_n$ of iterates

$$x_k := g(x_{k-1}) \quad (\text{P1.9.2})$$

converges, as $k \rightarrow \infty$, to \hat{x} satisfying

$$\hat{x}^2 = y. \quad (\text{P1.9.3})$$

Exercise 1.10. (a) Prove that if $g : D \rightarrow \mathbb{R}$ is differentiable and its derivative g' is a bounded function on D then g is a Lipschitz-continuous function on D .

Hint. Use the mean value theorem, in the integral form,

$$g(y) - g(x) = \left(\int_0^1 g'(x + \theta(y-x)) d\theta \right) (y-x), \quad (\text{P1.10.1})$$

and basic properties of the integral, such as the triangle inequality

$$\left| \int f \right| \leq \int |f| \quad (\text{P1.10.2})$$

and monotonicity

$$f \leq h \Rightarrow \int f \leq \int h. \quad (\text{P1.10.3})$$

- (b) Deduce that if $g \in C^1(D)$ and C is a compact (i.e., closed and bounded) subset of D then $g \in \text{Lip}(C)$ and

$$|g|_{\text{Lip}(C)} = \sup_C |g'| = \max_C |g'|. \quad (\text{P1.10.4})$$

(The last equality is just a formulaic way of saying that the least upper bound is achieved by some point $\check{x} \in C$: $|g'(\check{x})| = \sup_C |g'|$.)

- (c) Find a function f , Lipschitz-continuous on $[-1, 1]$, but not differentiable thereon (i.e., not differentiable at all points of $[-1, 1]$).

Exercise 1.11 (non-convergent fixed-point iteration). Let $y > 0$ and consider the sequence $(x_n)_{n \in \mathbb{N}_0}$ defined by the following fixed-point iteration

$$x_0 > 0 \text{ and } x_n := y/x_{n-1}, \text{ for } n \geq 1. \quad (\text{P1.11.1})$$

- (a) Show that

$$x_n \rightarrow \hat{x} \Rightarrow \hat{x} = \sqrt{y}. \quad (\text{P1.11.2})$$

- (b) Implement the method on a computer and describe its behaviour.
(c) Draw conclusions about convergence (or lack thereof) of $(x_n)_{n \in \mathbb{N}_0}$ and back up your findings with rigorous mathematical arguments.

CHAPTER 2

Newton's method

If thou art bored with this wearisome method of calculation, take pity on me, who had to go through with at least seventy repetitions of it, at a very great loss of time.

— Johannes Kepler

Johannes Kepler, in his study of orbital mechanics, was interested in solving the following equation:

$$\text{find } \hat{E} \geq 0 \text{ such that } \hat{E} - \varepsilon \sin \hat{E} = M, \quad (2.0.3)$$

for a given *anomaly* $M \in [0, 2\pi)$ and *eccentricity* $\varepsilon \in [0, 1]$. Equation (2.0.3) is now known as Kepler's equation. Realising that a closed form solution was difficult (the simplest such form being a complicated power series discovered much later and of not much use anyway) Kepler suggested the following iterative solution (Swerdlow, 2000):

- (1) Guess $E_0 < M_0$ if $M < \pi$ or $E_0 > \pi$ and $\sin E_0 < \sin M$ if $M \geq \pi$.
- (2) For each $k \in \mathbb{N}_0$, while $M - \varepsilon \sin E_k > \text{tol}$, compute

$$E_{k+1} := M - \varepsilon \sin E_k. \quad (2.0.4)$$

We recognise here a fixed point iteration with function $g(x) := M - \varepsilon \sin x$, which is Lipschitz-continuous with constant ε and thus a contraction for $\varepsilon < 1$. If $\varepsilon = 1$ (the case of a circular orbit), g fails to be a contraction and the method may converge very slowly. The method is not very fast anyway, as it is of linear convergence only as shown by Scott, 2011, §2.1. It took Kepler several pages of calculations to deduce a wrong (!) approximation of Mars's orbit, because he made a mistake in inputting the value ε .

Isaac Newton, possibly conscious of Kepler's method and its shortcomings, suggested the following "correction" as a way to speed up (2.0.4):

$$\tilde{E}_{k+1} := \tilde{E}_k + \frac{M - \tilde{E}_k + \varepsilon \sin \tilde{E}_k}{1 - \varepsilon \cos \tilde{E}_k}. \quad (2.0.5)$$

This leads to a quadratic error reduction and is in fact of what is known today at Newton–Raphson's method.

2.1. Derivations of Newton's method

Suppose $f : D \rightarrow \mathbb{R}$ is a given differentiable function. Consider solving the following equation

$$\text{find } \hat{x} \text{ such that } f(\hat{x}) = 0. \quad (2.1.1)$$

We will derive a method that, under suitable conditions allows to find a solution.

2.1.1. A geometric derivation. This is the “classical textbook” derivation which can be found in most elementary books on computational methods. It has the advantage of being graphic and intuitive, but the disadvantage of not being easily generalised it to higher dimensions. The fact that Newton’s method works in arbitrary dimensions (even infinitely many!) is one of the advantages.

2.1.2. An analytic derivation. An analytic derivation is based on the quadratic Banach–Caccioppoli contraction principle 1.3.4. We start by turning equation (2.1.1) into a fixed point problem by adding the identity of both sides

$$\text{find } \hat{x} \text{ such that } \hat{x} = \hat{x} + f(\hat{x}). \quad (2.1.2)$$

This leads to a fixed point problem of the form (1.2.8) with $g(x) = x + f(x)$. For example, if $f(x) = b - \arctan x$,¹ we obtain $g(x) = x + b - \arctan x$ which yields a contraction on any bounded interval (see 2.1.3) and the iteration converges. The rate of convergence from Banach–Caccioppoli contraction principle, is generally only linear. In order to improve this, we note that problem (2.1.1) of finding \hat{x} is equivalent to solving for x the fixed point equation

$$x = x + h(x)f(x) =: g(x) \quad (2.1.3)$$

where $h(x)$ is a function that we are still free to choose, as long as $h(x) \neq 0$. From Theorem 1.3.4, we may derive a quadratic method if $g'(\hat{x}) = 0$. This leads us to choose h such that

$$0 = g'(\hat{x}) = [1 + h'(x)f(x) + h(x)f'(x)]_{x=\hat{x}} = 1 + h(\hat{x})f'(\hat{x}). \quad (2.1.4)$$

Solving for $h(\hat{x})$, and *assuming* $f'(\hat{x}) \neq 0$ we obtain

$$h(\hat{x}) = -(f'(\hat{x}))^{-1} = \frac{1}{f'(\hat{x})}. \quad (2.1.5)$$

Therefore a choice of h satisfying (2.1.5) will work. For example, provided $f'(x) \neq 0$ for all x , we could take

$$h(x) := -f'(x)^{-1} \quad \forall x \in D. \quad (2.1.6)$$

This leads to the *Newton–Raphson iteration*:

$$x_{k+1} := x_k - f'(x_k)^{-1}f(x_k) \text{ for } k \in \mathbb{N}_0. \quad (2.1.7)$$

2.1.3. Exercise (a fixed point method to compute tan). Let $b \in (-\pi/2, \pi/2)$. Show that the sequence $(x_k)_{k \in \mathbb{N}_0}$ produced by the fixed point iteration

$$x_0 := b \text{ and } x_k := x_{k-1} + b - \arctan x_{k-1} \quad \forall k \in \mathbb{N} \quad (2.1.8)$$

converges to $\tan b$. Is the convergence linear? Is it quadratic? Implement this method with a computer code. Play “back-to-basics” by implementing $\arctan x = \int_0^x (1 + \xi^2)^{-1} d\xi$. Note that this yields a method for approximating $\tan b$ without ever resorting to any trigonometric function.

¹The root of $f(x) = \arctan x - b$ is $\tan b$, of course, but we’re playing dumb and pretending we do not know the solution in order to *understand* the method.

2.2. Convergence analysis

2.2.1. Definition of simple root, multiple root. Let $f : D \rightarrow \mathbb{R}$ and $\hat{x} \in D$ a root of f , i.e., $f(\hat{x}) = 0$. Consider the function ϕ defined by

$$\phi(x) := \frac{f(x)}{x - \hat{x}} \text{ for } x \in D \setminus \{\hat{x}\}. \quad (2.2.1)$$

We say that \hat{x} is a *simple root* of f if and only if $\lim_{x \rightarrow \hat{x}} \phi(x)$ (which might be ∞) is non-zero. Otherwise \hat{x} is a *multiple root*.

2.2.2. Proposition (simple roots of differentiable functions).

(i) If $f \in \text{Lip}(D)$ and \hat{x} is a root of f then the function

$$x \mapsto \phi(x) := \frac{f(x)}{x - \hat{x}} \quad (2.2.2)$$

is bounded and it has an upper and lower limit as $x \rightarrow \hat{x}$.

(ii) If f is differentiable then it has simple root at \hat{x} if and only if $f'(\hat{x}) \neq 0$.

Proof Exercise.

Hint. For the second part, if f is differentiable at \hat{x}

$$\lim_{x \rightarrow \hat{x}} \phi(x) = f'(\hat{x}). \quad (2.2.3)$$

□

2.2.3. Proposition (Lipschitz algebra). Suppose $u, v \in \text{Lip}(D)$ then $u + v, uv \in \text{Lip}(D)$ and

$$|u + v|_{\text{Lip}(D)} \leq |u|_{\text{Lip}(D)} + |v|_{\text{Lip}(D)} \quad (2.2.4)$$

$$|uv|_{\text{Lip}(D)} \leq |u|_{\text{Lip}(D)} \|v\|_{\text{L}^\infty(D)} + \|u\|_{\text{L}^\infty(D)} |v|_{\text{Lip}(D)}, \quad (2.2.5)$$

where $\|u\|_{\text{L}^\infty(D)} := \sup_D |u|$. If $u \in \text{Lip}(D)$ and $\inf_D |u| > 0$ then

$$\left| \frac{1}{u} \right|_{\text{Lip}(D)} \leq \frac{|u|_{\text{Lip}(D)}}{\inf_D u^2}. \quad (2.2.6)$$

Proof The proof of these inequalities is an immediate application of definition 1.2.6 and some basic manipulations. For example, to prove the “Lipschitz quotient rule” (2.2.6) it is enough to observe that

$$\left| \frac{1}{u(x)} - \frac{1}{u(y)} \right| = \left| \frac{u(y) - u(x)}{u(x)u(y)} \right| \leq \frac{|u|_{\text{Lip}(D)}}{\inf_D u^2} |x - y|. \quad (2.2.7)$$

□

2.2.4. Theorem (local convergence of Newton–Raphson’s method). Let D be an open subset of \mathbb{R} , $f \in C^2(D)$ with $f'' \in \text{Lip}(D)$ and \hat{x} as a simple root. Then for a small enough $\rho > 0$ Newton–Raphson’s iteration (2.1.7) with initial value $x_0 \in \bar{B}_\rho(\hat{x})$ yields a sequence $(x_k)_{k \in \mathbb{N}_0}$ that converges quadratically to \hat{x} .

Proof It is enough to apply the quadratic Banach–Caccioppoli theorem 1.3.4 to the iteration

$$x_{k+1} = g(x_k) \text{ with } g(x) := x - f'(x)^{-1} f(x). \quad (2.2.8)$$

Let us look at the derivative of g :

$$g'(x) = 1 - f'(x)^{-1} f'(x) + (f'(x))^{-2} f''(x) f(x) = \frac{f''(x) f(x)}{f'(x)^2}. \quad (2.2.9)$$

Since \hat{x} is a single root, it follows that $f'(\hat{x}) \neq 0$ and thus $g'(\hat{x}) = 0$. By continuity it follows that there exists a $\rho > 0$ such that $\|g'\|_{L_\infty(B)} < 1$ and $\inf_B |f'| > 0$ with $B := \overline{B}_\rho(\hat{x})$. The fact that D is open, ρ can be chosen as to have $B \subseteq D$. Since $\hat{x} = g(\hat{x})$ it follows that g is a contraction on B and $g \in C^1(B, B)$. By Lipschitz algebra, outlined in 2.2.3, it also follows that $g' \in \text{Lip}(B)$. Since B is closed, it follows, by the quadratic Banach–Caccioppoli theorem that a sequence $(x_k)_{k \in \mathbb{N}_0}$ with

$$x_0 \in B \text{ and } x_{k+1} := x_k - f'(x_k)^{-1} f(x_k) \quad \forall k \in \mathbb{N}_0 \quad (2.2.10)$$

converges to \hat{x} with

$$|x_{k+1} - \hat{x}| \leq \frac{L}{2} |x_k - \hat{x}|^2. \quad (2.2.11)$$

□

2.3. Variants of Newton–Raphson

Newton’s method, though fast and easily generalisable to higher (even infinite) space dimensions, has many drawbacks that make it impractical in many applications. Most of these drawbacks are not apparent at this stage but we list some nevertheless:

- (1) The method requires the computation of $f'(x_k)$ for each iteration k . While f is usually given, f' might not be available. In all the examples we are using f as some closed form solution of which we easily compute the derivative using the basic rules of calculus, but in practice f may be much too complex to be written as an expression of “elementary” functions.
- (2) Even if $f'(x_k)$ is computable, the method requires the solution for s_k of the linear equation $f'(x_k)s_{k+1} = f(x_k)$ at each iteration k . While this is easy for a scalar problem it can become trickier (and more computationally expensive) for higher dimensions, especially if done repeatedly or if the matrix $f'(x_k)$ is close to being singular.
- (3) Where to start? The local convergence theorem 2.2.4 guarantees convergence, but does not give any practical information about the ρ to be chosen as to guarantee that convergence. In fact, many times, the art of the computational scientist lies in guessing an appropriate initial guess x_0 . Other methods, i.e., fixed point iteration, may be slower than Newton–Raphson, but they guarantee *stability*.

For this (and other unstated reasons) Newton’s method has many variants of which we name a few as follows:

Secant method: the derivative is replaced by a finite-difference approximation.

Chord method: the finite-difference approximation is the same at all steps.

Broyden’s method: an extension of the secant method to higher dimensions.

Steffensen’s method: see Scott, 2011 for a description, works only in dimension 1 and cannot be generalised, hence of little practical interest.

Exercises and problems on Newton's method

Exercise 2.1. (a) Prove that if $g : D \rightarrow \mathbb{R}$ is differentiable and its derivative g' is a bounded function on D then g is a Lipschitz-continuous function on D .

Hint. Use the mean value theorem, in the integral form,

$$g(y) - g(x) = \left(\int_0^1 g'(x + \theta(y-x)) d\theta \right) (y-x), \quad (\text{P2.1.1})$$

and basic properties of the integral, such as the triangle inequality

$$\left| \int f \right| \leq \int |f| \quad (\text{P2.1.2})$$

and monotonicity

$$f \leq h \Rightarrow \int f \leq \int h. \quad (\text{P2.1.3})$$

(b) Deduce that if $g \in C^1(D)$ and C is a compact (i.e., closed and bounded) subset of D then $g \in \text{Lip}(C)$ and

$$\|g\|_{\text{Lip}(C)} = \sup_C |g'| = \max_C |g'|. \quad (\text{P2.1.4})$$

(The last equality is just a formulaic way of saying that the least upper bound is achieved by some point $\tilde{x} \in C$: $|g'(\tilde{x})| = \sup_C |g'|$.)

(c) Find a function f , Lipschitz-continuous on $[-1, 1]$, but not differentiable thereon (i.e., not differentiable at all points of $[-1, 1]$).

Exercise 2.2 (Newton's method computing a scalar's reciprocal). Given a real number $c > 0$, we want to compute $1/c$, the reciprocal of c . This is the unique solution for x of the equation

$$0 = \frac{1}{x} - c =: f(x). \quad (\text{P2.2.1})$$

- (a) Write down Newton's iteration for this problem in the form $x_{n+1} = g(x_n)$.
- (b) How many divisions does your iteration involve? Does this sound familiar?
- (c) Show that this iteration converges for all $x_0 \in [1/2c, 3/2c]$.
- (d) For $c = 3$ and $x_0 = 0.4$ calculate the first 3 iterates x_1, x_2, x_3 and the corresponding residuals.
- (e) Draw conclusions by comparing this method with the *Neumann series* method whereby $1/c$ is approximated by truncating the series

$$\sum_{k=0}^{\infty} (1-c)^k. \quad (\text{P2.2.2})$$

Exercise 2.3 (Newton's method vs. fixed point method). Consider solving the equation

$$f(x) := \exp(x/2) - 25x^2 = 0. \quad (\text{P2.3.1})$$

- (a) Show that f has exactly one root $\hat{x}_1 \in (-\infty, 0)$.
- (b) Show, using the Banach–Caccioppoli Contraction Principle, that the iteration $x_{n+1} = -\exp(x_n/4)/5$ converges to \hat{x}_1 for $x_0 \in (-\infty, 0]$.

Hint. Use any interval $[a, 0]$ with $a < -1/5$.

- (c) Writing or, better, modifying previously written short code, compute some iterations with initial values $x_0 = -30$ and $x_0 = 0$.
- (d) Use the Newton solver to approximate \hat{x}_1 . Compare with the fixed point iteration.

Exercise 2.4 (coding Newton). Given a differentiable function $f : D \rightarrow \mathbb{R}$, $D \subseteq \mathbb{R}$.

- (a) Write a code that will take as input f , f' , x_0 and ϵ , and return output x_K for some $K \in \mathbb{N}$ such that

$$x_k := x_{k-1} - f'(x_{k-1})^{-1} f(x_{k-1}) \text{ for } k = 1, \dots, K, \text{ and } |f(x_K)| < \epsilon. \quad (\text{P2.4.1})$$

For test purposes make the code return a whole array of values x_0, x_1, \dots, x_K and the corresponding residuals $f(x_0), f(x_1), \dots, f(x_K)$.

Hint. f and f' should be passed as strings, and use the Octave (or Matlab®) command `inline`.

- (b) Write a driver code that will call the code above and will take as inputs the functions f and f' (e.g., as strings or function handles, if you prefer) and x_0 . The output should be numerically meaningful: the sequence of iterates, the sequence of residuals, and (in cases where the exact solution is known) the sequence of errors. A proper driver writes the data to a file, so you can use it later for visualisation and typesetting reports.
- (c) Test your code (and use the built-in command `fsolve` to compute the error) on the following benchmark cases
 - (i) $f(x) = \exp(x) - 2$
 - (ii) $f(x) = x^3 + 6x - 8$

Exercise 2.5 (a fixed point method to compute \tan). Let $b \in (-\pi/2, \pi/2)$. Show that the sequence $(x_k)_{k \in \mathbb{N}_0}$ produced by the fixed point iteration

$$x_0 := b \text{ and } x_k := x_{k-1} + b - \arctan x_{k-1} \quad \forall k \in \mathbb{N} \quad (\text{P2.5.1})$$

converges to $\tan b$. Is the convergence linear? Is it quadratic? Implement this method with a computer code. Play “back-to-basics” by implementing $\arctan x = \int_0^x (1 + \xi^2)^{-1} d\xi$. Note that this yields a method for approximating $\tan b$ without ever resorting to any trigonometric function.

Exercise 2.6 (Newton’s method). Consider using Newton’s method to solve for x

$$\tan x = b, \quad (\text{P2.6.1})$$

given $b \in (-\pi/2, \pi/2)$.

- (a) Write down the method and implement a short code.
- (b) Compare this method with the one given by the fixed point iteration.

Problem 2.7 (Steffensen’s method). A variant of Newton’s method for solving $f(x) = 0$ for x , is given by Steffensen’s method whereby the choice for the corrected fixed-point iteration

$$x_{k+1} = g(x_k) \text{ with } g(x) := x + h(x)f(x), \quad (\text{P2.7.1})$$

is given by

$$h(x) = \frac{f(x)}{f(x) - f(x + f(x))}. \quad (\text{P2.7.2})$$

- (a) Assuming that f is differentiable on all its domain, show that if \hat{x} is a simple root of f then $g'(\hat{x}) = 0$.

- (b) Suppose $f \in C^2(D, D)$ and that $f'' \in \text{Lip}(D)$. Show that (P2.7.1) yields a convergent sequence for a sufficiently close initial guess x_0 .
- (c) Can you relax the above conditions on f ?

CHAPTER 3

Gaussian elimination

This chapter assumes some familiarity with basic linear algebra, a skeleton of which is sketched in Appendix A. The solution of linear systems of simultaneous equations with coefficients in a scalar field $\mathbb{K} = \mathbb{R}$ or \mathbb{C} is perhaps the single most important problem in computational mathematics. It might seem strange, at first sight, that such a simple problem as

$$\text{find vector } \mathbf{x} \in \mathbb{K}^n \text{ such that } \mathbf{Ax} = \mathbf{b}, \text{ for given matrix } \mathbf{A} \in \mathbb{K}^{n \times n} \text{ and } \mathbf{b} \in \mathbb{K}^n \quad (3.0.3)$$

needs much study, but this types of problems arise virtually in any quantitative branch of Science, Techonology, Informatics or Econometry, where the number n can easily have 10 or 11 digits and keeps rising as the demand for quantification increases. This huge number of unknowns requires efficient algorithms and constitutes the topic of so-called *Matrix Analysis* which we briefly visit in this course.

In §0.0.2 we have seen the simplest non-trivial case of Gaussian elimination algorithm in 2 dimensions. You will be familiar from elementary courses with the extension of such algorithm, known as Gaussian elimination, to dimension n . In this chapter, we study this procedure in depth and draw some conclusion.

One important aspect of Gaussian elimination and the related algorithms is that it involves no *approximation*, unlike, say iterative methods for nonlinear equations of Chapters 1 and 2. For this, Gaussian elimination is called a *direct method*. Iterative methods's use is not limited to nonlinear problems: these are used, and they lead to very efficient algorithms, for certain classes of linear systems as well. Thus, the linear system solvers are divided into: direct solvers and iterative solvers.¹ In this and the next chapter we will be dealing with direct solvers.

3.1. Triangular matrices

3.1.1. Definition of triangular matrices. A square matrix $\mathbf{A} = [a_i^j]_{i=1, \dots, n}^{j=1, \dots, n} \in \mathbb{K}^{n \times n}$ (where i is the *row-index* and j is *column-index*) is called

★ *upper triangular* if and only if

$$a_i^j = 0 \text{ for } i > j, \quad (3.1.1)$$

★ *lower triangular* if and only if

$$a_i^j = 0 \text{ for } i < j, \quad (3.1.2)$$

★ *triangular* if and only if \mathbf{A} is upper triangular *or* lower triangular,

★ *diagonal* if and only if \mathbf{A} is *both* upper *and* lower triangular,

¹The words solver and method are synonymous and often liberally interchanged in computational mathematics.

★ (diagonally) normalised if and only if

$$a_i^i = 1 \quad \forall i = 1, \dots, n. \quad (3.1.3)$$

3.1.2. Definition of sets of triangular matrices. We will use the following sets of triangular matrices

$$\mathcal{U}(\mathbb{K}^n) := \{X \in \mathbb{K}^{n \times n} : X \text{ is upper triangular}\}, \quad (3.1.4)$$

$$\mathcal{L}(\mathbb{K}^n) := \{X \in \mathbb{K}^{n \times n} : X \text{ is normalised lower triangular}\}, \quad (3.1.5)$$

and

$$\mathcal{D}(\mathbb{K}^n) := \{X \in \mathbb{K}^{n \times n} : X \text{ is diagonal}\}. \quad (3.1.6)$$

3.1.3. Proposition.

- (a) The sets $\mathcal{U}(\mathbb{K}^n)$ and $\mathcal{D}(\mathbb{K}^n)$ are closed under scaling, matrix addition and matrix multiplication, and are thus algebras of operators.
- (b) The set $\mathcal{L}(\mathbb{K}^n)$ is closed under matrix multiplication and is a (non-Abelian) group with respect to this operation.

Proof See the problem section. □

3.1.4. Remark (stars and noughts). Often, 0 entries are omitted when displaying a matrix explicitly. Also a non-specified entry, is usually denoted with a *, for example, we will often encounter expressions of the type

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ & 1 & 0 \\ & & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ & 1 & \\ & & 1 \end{bmatrix} = \begin{bmatrix} 1 & * & * \\ & 1 & * \\ & & 1 \end{bmatrix}. \quad (3.1.7)$$

Of course, the last “=” sign has to be treated with a pinch of salt. Also note that 0 are not always omitted and that $* = 0$ is allowed in such expressions.

3.1.5. Example. Upper triangular matrices:

$$\begin{bmatrix} 1 & 3 \\ & 2 \end{bmatrix}, \begin{bmatrix} 1 & 3 & 4 \\ & 5 & 0 \\ & & 6 \end{bmatrix}. \quad (3.1.8)$$

Lower triangular matrices:

$$\begin{bmatrix} 1 & & \\ 2 & 3 & \\ & & \end{bmatrix}, \begin{bmatrix} 1 & & \\ 2 & 0 & \\ 4 & 3 & 6 \end{bmatrix}. \quad (3.1.9)$$

Diagonal matrices:

$$\begin{bmatrix} 3 & & \\ & 4 & \\ & & \end{bmatrix}, \begin{bmatrix} 2 & & \\ & 5 & \\ & & -1 \end{bmatrix} \quad (3.1.10)$$

Normalised lower triangular matrices:

$$\begin{bmatrix} 1 & & \\ 2 & 1 & \end{bmatrix} \begin{bmatrix} 1 & & \\ 6 & 1 & \\ 4 & -1 & 1 \end{bmatrix}. \quad (3.1.11)$$

3.2. Solving triangular systems

The most basic algorithms for solving a linear system, is substitution, if one is lucky enough to encounter a system that is triangular. The object of Gaussian elimination is to split a general linear system $\mathbf{Ax} = \mathbf{b}$ solve into the solution of two triangular systems, whereby $\mathbf{A} = \mathbf{LU}$, with $\mathbf{L} \in \mathcal{L}(\mathbb{K}^n)$ and $\mathbf{U} \in \mathcal{U}(\mathbb{K}^n)$. This is useful because the solution could be then split into the solution of two linear systems:

$$\text{find } \mathbf{v} : \mathbf{Lv} = \mathbf{b} \text{ then find } \mathbf{x} : \mathbf{Ux} = \mathbf{v}. \quad (3.2.1)$$

3.2.1. Fore-substitution. Forward substitution or *fore-substitution* is a procedure to solve for $\mathbf{x} \in \mathbb{K}^n$ the system

$$\mathbf{Lx} = \mathbf{c} \quad (3.2.2)$$

where $\mathbf{L} \in \mathbb{K}^{n \times n}$ is a given lower triangular matrix and $\mathbf{c} \in \mathbb{K}^n$ a given vector. Explicitly, we want to find, $x_1, \dots, x_n \in \mathbb{K}$, such that

$$\begin{array}{rclcl} l_1^1 x_1 & & & & = c_1, \\ l_2^1 x_1 & + l_2^2 x_2 & & & = c_2, \\ \vdots & \dots & \ddots & & \vdots \\ l_{n-1}^1 x_1 & + \dots & + l_{n-1}^{n-1} x_{n-1} & & = c_{n-1}, \\ l_n^1 x_1 & + \dots & + l_n^{n-1} x_{n-1} & + l_n^n x_n & = c_n. \end{array} \quad (3.2.3)$$

Solving the first equation we have

$$x_1 = (l_1^1)^{-1} c_1, \quad (3.2.4)$$

then, passing to the second equation, we get

$$x_2 = (l_2^2)^{-1} (c_2 - l_2^1 x_1), \quad (3.2.5)$$

and so on and so forth, for each $k = 1, \dots, n$, having computed all x_1, \dots, x_{k-1} we may compute

$$x_k = (l_k^k)^{-1} \left(c_k - \sum_{j=1}^{k-1} l_k^j x_j \right), \quad (3.2.6)$$

where the empty summation (needed to include the case $k = 1$) is defined to be 0. We leave it to the reader to develop an algorithm implementing fore-substitution, by mimicking the work performed next for back-substitution in §3.2.2.

3.2.2. Back-substitution. We here look at back-substitution as the “final step” in the solution of a linear system whose matrix has been LU factorised. Let $n \in \mathbb{N}$ and consider solving for a (column) vector $\mathbf{x} = (x_1, \dots, x_n)$, the linear system

$$\mathbf{U}\mathbf{x} = \mathbf{c} \quad (3.2.7)$$

where $\mathbf{U} = [u_i^j]_{i=1, \dots, n}^{j=1, \dots, n}$ is an *upper triangular matrix* with diagonal entries all invertible, i.e.,

$$u_i^i \neq 0 \quad \forall i = 1, \dots, n, \quad (3.2.8)$$

and $\mathbf{c} = (c_1, \dots, c_n)$ a given (column) vector in \mathbb{K}^n .²

$$\begin{array}{ccccccc} u_1^1 x_1 & + u_1^2 x_2 & + \cdots & + u_1^n x_n & = & c_1, \\ & u_2^2 x_2 & + \cdots & + u_2^n x_n & = & c_2, \\ & & \ddots & & & \vdots \\ & & & u_{n-1}^{n-1} x_{n-1} & + u_{n-1}^n x_n & = c_{n-1}, \\ & & & & u_n^n x_n & = c_n. \end{array} \quad (3.2.9)$$

Solving the n -th equation, we have that

$$x_n = (u_n^n)^{-1} c_n. \quad (3.2.10)$$

Using this in the $(n-1)$ -th equation we get

$$x_{n-1} = (u_{n-1}^{n-1})^{-1} (c_{n-1} - u_{n-1}^n x_n) \quad (3.2.11)$$

which can be computed by *substituting* x_n in the right-hand side. Working our way *backwards* (or *upwards* if you prefer), having computed x_n, \dots, x_{k+1} and the k -th equation in the triangular system (3.2.9) being

$$u_k^k x_k + u_k^{k+1} x_{k+1} + \cdots + u_k^n x_n = c_k \quad (3.2.12)$$

it follows that, for each $k = n-1, \dots, 1$,

$$\begin{aligned} x_k &= (u_k^k)^{-1} (c_k - u_k^{k+1} x_{k+1} - \cdots - u_k^n x_n) \\ &= (u_k^k)^{-1} \left(c_k - \sum_{j=k+1}^n u_k^j x_j \right) \\ &= ([\mathbf{U}]_k^k)^{-1} ([\mathbf{c}]_k - [\mathbf{U}]_k^{k+1:n} [\mathbf{x}]_{k+1:n}), \end{aligned} \quad (3.2.13)$$

where we have respectively used all three notation styles: dots, summation, and matrix.³

Note that it is possible to include the case $k = n$ by defining the empty sum to be 0.

3.2.3. Example.

EXERCISE. Solve for \mathbf{x} the system $\mathbf{U}\mathbf{x} = \mathbf{c}$ where, omitting the 0 entries in the matrix, you are given

$$\mathbf{U} := \begin{bmatrix} 1 & & 2 \\ & 1 & -1 \\ & & -3 \end{bmatrix} \text{ and } \mathbf{c} := \begin{bmatrix} 3 \\ 0 \\ 6 \end{bmatrix} \quad (3.2.14)$$

.

²Whenever unstated a vector is a column-vector.

³We will be extensively using, and liberally mixing, these three notation styles, especially the last two.

Since the system is upper-triangular, working backwards, we have

$$x_3 = -\frac{6}{3} = -2, \quad (3.2.15)$$

whence, using the second equation

$$x_2 = x_3 = -2, \quad (3.2.16)$$

and, using the first equation, x_2 and x_3 we get

$$x_1 = 3 - 2x_3 = 3 + 4 = 7. \quad (3.2.17)$$

Solution is $\mathbf{x} = (7, -2, -2)$, which is easily checked to be correct.

3.2.4. Problem. Prove that if $\mathbf{U} \in \mathbb{K}^{n \times n}$, $n \in \mathbb{N}$ is a upper triangular and $u_k^k = 0$ for some $k = 1, \dots, n$ then there exists a nonzero vector \mathbf{v} such that $\mathbf{U}\mathbf{v} = \mathbf{0}$. Deduce that an upper triangular matrix is invertible if and only if it has all its diagonal entries are invertible.

3.2.5. Algorithm (Back-substitution). We summarise this section in a *pseudocode* form, which means a nearly computer-implementable, but computer-language-independent form.

```

1: procedure BACKSUBSTITUTION( $\mathbf{U}, \mathbf{c}$ )           ▷ computes  $\mathbf{x}$  such that  $\mathbf{U}\mathbf{x} = \mathbf{c}$ 
2:   for  $k = n, \dots, 1$  do
3:      $x_k \leftarrow (u_k^k)^{-1} (c_k - [\mathbf{U}]_k^{k+1:n} [\mathbf{x}]_{k+1:n})$   ▷ for  $k = n$  sum is empty hence 0
4:   end for
5:   return  $\mathbf{x}$ 
6: end procedure

```

3.2.6. Implementation of back-substitution. Here is a possible Octave implementation.

Printout of file backsub.m

```

%%% -*- mode: octave; -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x] = backsub(U,c)
%% function [x] = backsub(U,c)
%%
%% iteration-based implementation of backsubstitution
%%
sizec = size(c);
N = max(sizec);
%%
if (sizec(1)<sizec(2))
    c = c';
end
%%
x = zeros(N,1);
for n=N:-1:1
    x(n)=U(n,n)\(c(n)-U(n,n+1:N)*x(n+1:N));
end
endfunction

```

The following *tester code* can check backsub for bugs.

```

%%% -*- mode: octave; -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function backsubstitutionTest(n)
%% function backsubstitutionTest(n)
%%
%% This function is a benchmark Test for the backsubstitution()
%% function
%%
%% build a "random" non-singular triangular matrix
TriangularMatrix = eye(n)+.5*(rand(n)-.5).*((ones(n,1)*[1:n])>=[1:n]')*←
    ones(1,n));
%% build a random (column) vector
dataVector = rand(n,1);
%%
runTime = time;
solutionVector = backsub(TriangularMatrix,dataVector); %%backsubrec
runTime = time-runTime
%% test
if((dataVector - TriangularMatrix*solutionVector!=0))
    "sigh..."
else
    if(n<16)
        TriangularMatrix
        printf("inverted on\n");
        dataVector
        printf("gives\n");
        solutionVector
    end
    "residual is zero: hurrah!"
end
endfunction

```

3.2.7. Back in block. Back-substitution can be implemented blockwise for a system of the form

$$\mathbf{U} =: \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ & \mathbf{D} \end{bmatrix}, \mathbf{x} =: \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} \text{ and } \mathbf{c} =: \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \quad (3.2.18)$$

with $\mathbf{A} \in \mathbb{K}^{m \times m}$, $\mathbf{D} \in \mathbb{K}^{l \times l}$, $\mathbf{B} \in \mathbb{K}^{m \times l}$, $m + l = n$, and $\mathbf{f}, \mathbf{y} \in \mathbb{K}^m$, $\mathbf{g}, \mathbf{z} \in \mathbb{K}^l$, and \mathbf{A}, \mathbf{D} invertible, so that a block-backsubstitution reads as follows

$$\mathbf{z} = \mathbf{D}^{-1} \mathbf{g} \text{ and } \mathbf{y} = \mathbf{A}^{-1} (\mathbf{f} - \mathbf{B} \mathbf{z}) \quad (3.2.19)$$

where each *matrix inverse* indicates an application of the backsubstitution algorithm if \mathbf{A} and \mathbf{D} are triangular, or some other inversion algorithm, if not.

3.2.8. Algorithm (Back-substitution). Of particular interest in 3.2.7 is the case when m (or l) be 1, because that leads to a recursive version of backsubstitution. In informatics a *recursive algorithm* is one that calls itself. To get a recursive version of back-substitution it is useful to think in blocks: except the *bottom case* where $n = 1$, $\mathbf{U} = u \in \mathbb{K}^{1 \times 1} = \mathbb{K}$ and $\mathbf{c} = c \in \mathbb{K}^1 = \mathbb{K}$ which is simply solved as $x = u^{-1} c$, the matrix \mathbf{U} as being decomposed into a $(1, n-1) \times (1, n-1)$ blocks as follows

$$\mathbf{U} = \begin{bmatrix} u & \mathbf{v}^\top \\ \mathbf{0} & \mathbf{W} \end{bmatrix} \text{ where } u = u_1^1, \mathbf{v}^\top = [\mathbf{U}]_1^{2:n}, \mathbf{W} = [\mathbf{U}]_{2:n}^{2:n}. \quad (3.2.20)$$

Then, assuming that $\hat{\mathbf{x}} := [\mathbf{x}]_{2:n}$ is the solution of $\mathbf{W} \hat{\mathbf{x}} = \hat{\mathbf{c}} := [\mathbf{c}]_{2:n}$, we can compute x_1 by using

$$x_1 = u^{-1}(c_1 - \mathbf{v}^T \hat{\mathbf{x}}) \quad (3.2.21)$$

Following is a pseudocode for *recursive back-substitution*

Require: $n \in \mathbb{N}$, $\mathbf{U} \in \mathcal{U}(\mathbb{K}^n)$, $\mathbf{c} \in \mathbb{K}^n$

Ensure: \mathbf{x} such that $\mathbf{U}\mathbf{x} = \mathbf{c}$

```

1: procedure RECURSIVE-BACKSUBSTITUTION( $\mathbf{U}, \mathbf{c}$ )
2:   if  $n = 1$  then  $\triangleright \mathbf{U} = u \in \mathbb{K}$  and  $\mathbf{c} = c \in \mathbb{K}$ 
3:      $\mathbf{x} \leftarrow \mathbf{U}^{-1}\mathbf{c}$ 
4:   else
5:      $[\mathbf{x}]_{2:n} \leftarrow \text{RECURSIVE-BACKSUBSTITUTION}([\mathbf{U}]_{2:n}^{2:n}, [\mathbf{c}]_{2:n})$ 
6:      $x_1 \leftarrow (u_1^{-1})(c_1 - [\mathbf{U}]_1^{2:n}[\mathbf{x}]_{2:n})$ 
7:   end if
8:   return  $\mathbf{x}$ 
9: end procedure

```

3.2.9. A recursive implementation of back-substitution. While recursive algorithms are not the best to implement (because almost all computer languages are not optimised for recursion and they consume too much memory) they are quite interesting from a theoretical point of view and so is their implementation as a way to investigate them “practically”.

Printout of file backsubrec.m

```

%%% -*- mode: octave; -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x] = backsubrec(U,c)
%% function [x] = backsubrec(U,c)
n = max(size(c));
x = zeros(n,1);
if (n>1)
    x(2:n,1) = backsubrec(U(2:n,2:n),c(2:n));
end
x(1,1) = U(1,1)\(c(1)-U(1,2:n)*x(2:n,1));
endfunction

```

The same tester code as in 3.2.2 (after appropriate tweaking) can be used.

3.2.10. Operation count for back-substitution. Let us use the recursive implementation to count the number of arithmetic operations needed by the back-substitution algorithm (note that both versions, iterative or recursive, amount to the same number of operations). Denote by $C(n)$ the *operation count*, sometimes called *computational cost* or *complexity*, of back-substitution to solve the upper triangular system in \mathbb{K}^n , i.e., $\mathbf{U} \in \mathcal{U}(\mathbb{K}^n)$ and $\mathbf{c} \in \mathbb{K}^n$. Then

$$C(1) = r + 1 \quad (3.2.22)$$

where r (is a constant that) counts the number of operations needed to calculate a reciprocal (that of u) and 1 comes from the multiplication of the reciprocal by c . By the recursive version in §3.2.8, we get, for $n \geq 2$ that

$$C(n) = C(n-1) + r + (n-1) \quad (3.2.23)$$

where $C(n-1)$ comes from the recursive call, $+r$ for the inversion of u_1^1 , $+n-2$ for the product $[\mathbf{U}]_1^{2:n} [\mathbf{x}]_{2:n}$ and $+1$ for the subtraction. Hence the total number of operations is

$$C(n) = (n-1)r + \sum_{i=1}^{n-1} i + C(1) = 1 + nr + \frac{1}{2}n(n-1) = \frac{1}{2}n^2 + \left(r - \frac{1}{2}\right)n + 1. \quad (3.2.24)$$

3.3. Gaussian elimination

Row reduction is a series of linear operations (scale or add) on rows of a matrix where rows cannot be split.

3.3.1. The 2×2 row reduction procedure. Understanding the 2×2 will get us almost all the way, by keeping it as simple as may be. Consider eliminating the lower off-diagonal entry c in the $\mathbb{K}^{2 \times 2}$ matrix

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}. \quad (3.3.1)$$

For this to succeed with row operations

$$\text{scale row 1 of } \mathbf{A} \text{ by } -c a^{-1} \text{ and add it to row 2 of } \mathbf{A} \quad (3.3.2)$$

obtaining thus the matrix

$$\begin{bmatrix} a & b \\ 0 & d - c a^{-1} b \end{bmatrix} =: \mathbf{U}. \quad (3.3.3)$$

3.3.2. 2×2 LU factorisation. Recalling that

$$\mathbf{e}^1 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{e}^2 := \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ and } \mathbf{e}_i := \mathbf{e}^{i\top}, \quad (3.3.4)$$

we note that for a generic matrix \mathbf{X}

$$\text{row } i \text{ of } \mathbf{X} = \mathbf{e}_i \mathbf{X}. \quad (3.3.5)$$

This allows us to algebraically summarise the operation, by comparing the rows of the initial matrix \mathbf{A} and the final matrix \mathbf{U} we have

$$\begin{bmatrix} \mathbf{e}_1 \mathbf{U} \\ \mathbf{e}_2 \mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 \mathbf{A} \\ \mathbf{e}_2 \mathbf{A} - c a^{-1} \mathbf{e}_1 \mathbf{A} \end{bmatrix}. \quad (3.3.6)$$

A bit more matrix algebra yields

$$\mathbf{U} = \left(\begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{0}^\top \\ c a^{-1} \mathbf{e}_1 \end{bmatrix} \right) \mathbf{A} = \underbrace{(\mathbf{I} - \mathbf{m} \mathbf{e}_1)}_{=: \mathbf{M}} \mathbf{A} \text{ with } \mathbf{m} := \begin{bmatrix} 0 \\ c \end{bmatrix} a^{-1}. \quad (3.3.7)$$

Note that we introduced a matrix $\mathbf{M} := \mathbf{I} - \mathbf{m} \mathbf{e}_1 = \mathbf{I} - \mathbf{m} \mathbf{e}^{1\top}$, which inspection reveals to be normalised lower triangular ($\mathcal{L}(\mathbb{K}^n)$) as

$$\mathbf{M} = \begin{bmatrix} 1 & 0 \\ -c a^{-1} & 1 \end{bmatrix}. \quad (3.3.8)$$

[*]: Check!

Noting that the inverse of \mathbf{M} is also (normalised) lower triangular since[*]

$$\mathbf{M}^{-1} = \begin{bmatrix} 1 & 0 \\ c a^{-1} & 1 \end{bmatrix} =: \mathbf{L}, \quad (3.3.9)$$

it follows that we have factored the matrix A into the product of a normalised lower triangular and an upper triangular matrix:

$$A = LU. \quad (3.3.10)$$

3.3.3. Blockwise algebra. Extending the discussion in §3.3.1 to the general, $(1 + n - 1) \times (1 + n - 1)$, will not be hard provided we follow two simple rules:

- (1) mimick the 2×2 case *blockwise matrix algebra*,
- (2) use *recursion*, or *induction*, to put it on a sound footing.

Blockwise algebra is the same as matrix algebra, but for general rings, e.g., if $A, M \in \mathbb{K}^{n \times n}$ are written as

$$A = \begin{bmatrix} a & \mathbf{b}^\top \\ \mathbf{c} & D \end{bmatrix} \text{ and } Z = \begin{bmatrix} z & \mathbf{y}^\top \\ \mathbf{x} & W \end{bmatrix} \quad (3.3.11)$$

where

$$a, z \in \mathbb{K}, \mathbf{b}, \mathbf{c}, \mathbf{y}, \mathbf{x} \in \mathbb{K}^{n-1}, D, W \in \mathbb{K}^{(n-1) \times (n-1)}, \quad (3.3.12)$$

then the blockwise product of the two matrices is

$$AZ = \begin{bmatrix} az + \mathbf{b}^\top \mathbf{x} & a\mathbf{y}^\top + \mathbf{b}^\top W \\ \mathbf{c}z + D\mathbf{x} & \mathbf{c}\mathbf{y}^\top + DW \end{bmatrix}. \quad (3.3.13)$$

3.3.4. Definition of LU-cky matrix. A square matrix $A \in \mathbb{K}^{n \times n}$ written

$$A = \begin{bmatrix} a & \mathbf{b}^\top \\ \mathbf{c} & D \end{bmatrix} \quad (3.3.14)$$

is *LU-cky* if and only if either A is empty (when $n = 0$), or

- (i) the first diagonal entry is invertible, $a \neq 0$
and
- (ii) the matrix $D - \mathbf{c}a^{-1}\mathbf{b}^\top$ is LU-cky.

If a matrix is not LU-cky we call it *un-LU-cky*.

3.3.5. Example. The empty matrix in $\mathbb{K}^{0 \times 0}$ is LU-cky. All nonzero scalars are LU-cky matrices in $\mathbb{K}^{1 \times 1}$. In $\mathbb{K}^{2 \times 2}$ the situation is a bit more interesting (as much as looking at a and calculating the determinant is), for example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix} \text{ are LU-cky} \quad (3.3.15)$$

while

$$\begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 3 \\ 2 & 1 \end{bmatrix} \text{ are un-LU-cky.} \quad (3.3.16)$$

Notice how un-LU-cky matrices come in two flavours: singular and invertible.

3.3.6. Theorem (LU factorisation). *If a matrix $A \in \mathbb{K}^{n \times n}$ for some $n \in \mathbb{N}$ is LU-cky then A admits a unique LU factorisation, i.e., there exists a unique pair $L \in \mathcal{L}(\mathbb{K}^n)$ and $U \in \mathcal{U}(\mathbb{K}^n)$ such that*

$$A = LU. \quad (3.3.17)$$

Proof We proceed by induction on n to give a constructive proof of existence. The base case $n = 1$ is trivial as LU-cky 1×1 matrices (scalars) are the elements of $\mathbb{K} \setminus \{0\}$, whence $a = 1a$ provides an LU factorisation.

For the inductive step, think blockwise of the matrix A :

$$A = \begin{bmatrix} a & \mathbf{b}^\top \\ \mathbf{c} & D \end{bmatrix}, \text{ where } a \in \mathbb{K}, \mathbf{b}, \mathbf{c} \in \mathbb{K}^{n-1}, D \in \mathbb{K}^{(n-1) \times (n-1)}. \quad (3.3.18)$$

By mimicking the argument in §3.3.1 introduce the row-reduction vector and matrix, respectively, as

$$\begin{aligned} \mathbf{m} &:= \begin{bmatrix} 0 \\ \mathbf{c}a^{-1} \end{bmatrix} \text{ and} \\ \mathbf{M} := \mathbf{L}_1(\mathbf{m}) &:= \mathbf{I} - \mathbf{m}\mathbf{e}_1 = \begin{bmatrix} 1 & \mathbf{0}^\top \\ -\mathbf{c}a^{-1} & \hat{\mathbf{I}} \end{bmatrix} \text{ where } \hat{\mathbf{I}} := \text{identity in } \mathbb{K}^{(n-1) \times (n-1)}. \end{aligned} \quad (3.3.19)$$

Premultiplying A by \mathbf{M} yields

$$\mathbf{M}A = \begin{bmatrix} a & \mathbf{b}^\top \\ \mathbf{0} & \hat{A} \end{bmatrix} \text{ where } \hat{A} := D - \mathbf{c}a^{-1}\mathbf{b}^\top. \quad (3.3.20)$$

By the LU-ckiness of A we have that \hat{A} is LU-cky and the inductive hypothesis tells us that

$$\hat{A} = \hat{L}\hat{U} \text{ for some } \hat{L} \in \mathcal{L}(\mathbb{K}^{n-1}), \hat{U} \in \mathcal{U}(\mathbb{K}^{n-1}). \quad (3.3.21)$$

On the other hand, it is an exercise in linear algebra to see that

$$\mathbf{M}^{-1} = \mathbf{L}_1(-\mathbf{m}) = \mathbf{I} + \mathbf{m}\mathbf{e}_1, \quad (3.3.22)$$

call it \mathbf{L}_1 , whence, blockwise we have

$$A = \begin{bmatrix} 1 & 0 \\ \mathbf{c}a^{-1} & \hat{\mathbf{I}} \end{bmatrix} \begin{bmatrix} a & \mathbf{b}^\top \\ \mathbf{0} & \hat{L}\hat{U} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{c}a^{-1} & \hat{\mathbf{I}} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \hat{L} \end{bmatrix} \begin{bmatrix} a & \mathbf{b}^\top \\ \mathbf{0} & \hat{U} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{c}a^{-1} & \hat{L} \end{bmatrix}}_{=: L} \underbrace{\begin{bmatrix} a & \mathbf{b}^\top \\ \mathbf{0} & \hat{U} \end{bmatrix}}_{=: U} \quad (3.3.23)$$

Uniqueness is left as an exercise in Problem 3.11. □

3.3.7. Algorithm (recursive LU factorisation). The inductive nature of the proof of Theorem 3.3.6 provides us with the following recursive algorithm.

Require: $n \in \mathbb{N}, A \in \mathbb{K}^{n \times n}, \mathbf{c} \in \mathbb{K}^n$

Ensure: A is LU-cky and $A = LU$ with $L \in \mathcal{L}(\mathbb{K}^n), U \in \mathcal{U}(\mathbb{K}^n)$

1: **procedure** RECURSIVE-LU-FACTORISATION(A)

2: $\mathbf{u}_1 \leftarrow \mathbf{a}_1$ and $\mathbf{l}^1 \leftarrow \begin{bmatrix} 1 \\ [A]_{2:n}^1 (a_1^1)^{-1} \end{bmatrix}$ ▷ ignore the zeros

3: **if** $n > 1$ **then**

4: $(\hat{L}, \hat{U}) \leftarrow \text{RECURSIVE-LU-FACTORISATION}([A]_{2:n}^{2:n} - [A]_{2:n}^1 (a_1^1)^{-1} [A]_1^{2:n})$

5: **end if**

6: **return** L, U

7: **end procedure**

3.3.8. A recursive implementation of LU.

The following gives a recursive implementation

Printout of file recursiveLU.m

```
%%% -*- mode: octave; -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [L,U] = recursiveLU(A)
%% function LU = recursiveLU(A)

%% get the size of the matrix A
[n,m]=size(A);

%% test for A being square
if n!=m
    error("matrix must be square!\n")
end

%% test for A being LUcky, exit otherwise
if A(1,1)==0
    error("A is an unLUcky matrix, try again\n")
    return;
end

if n<2 %% the bottom case
    "bottom\n"
    L = 1;
    U = A;
    return;
else
    %% build the reduction vector m (no need to worry about zero entries)
    msmall = A(2:n,1)/A(1,1);
    %% prepare the reduced block
    Asmall = A(2:n,2:n) - msmall*A(1,2:n)
    %% recursive call
    [Lsmall,Usmall] = recursiveLU(Asmall);
    %% as is this is a bit wasteful, L doesn't need to be a square
    %% a good idea would be to implement this as a subroutine
    %%
    %% process the output of the recursion
    %% build the big lower triangular matrix
    L = [1,zeros(1,n-1);msmall,Lsmall]
    %% and the big upper triangular matrix
    U = [A(1,:);[zeros(n-1,1),Usmall]]
end
endfunction
```

with the driver code

Printout of file recursiveLUDriver.m

```
%%% -*- mode: octave -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function recursiveLUDriver(n)
%% function recursiveLUDriver(n)
%%
%% This function provides a benchmark test for the
%% function recursiveLU
```

```

%%
%% build a "random" matrix
dataMatrix = eye(n)+.5*(rand(n)-.5);
%%
runTime = time;
[Lsolution,Usolution] = recursiveLU(dataMatrix);
runTime = time-runTime
%% test
if((Lsolution*Usolution - dataMatrix!=0))
    "sigh..."
else
    if (n<9)
        dataMatrix
        printf("has L, U decomposition\n");
        Lsolution
        Usolution
    end
    "LU-A=0"
end
endfunction

```

3.3.9. Iterative LU. LU factorisation is usually implemented iteratively, as opposed to recursively,⁴ because most programming languages are optimised for iterations.

Require: $n \in \mathbb{N}, A \in \mathbb{K}^{n \times n}, c \in \mathbb{K}^n$

Ensure: A is LU-cky and $A = LU$ with $L \in \mathcal{L}(\mathbb{K}^n)$, $U \in \mathcal{U}(\mathbb{K}^n)$

```

1: procedure LU-FACTORISATION( $A$ )
2:    $U \leftarrow O \in \mathbb{K}^{n \times n}$  and  $L \leftarrow I \in \mathbb{K}^{n \times n}$  ▷ initialise the matrices  $U$  and  $L$ 
3:   for  $i = 1, \dots, n$  do
4:      $[U]_i^{i:n} \leftarrow [A]_i^{i:n}$ 
5:     if  $i < n$  then
6:        $[L]_{i+1:n}^i \leftarrow [A]_{i+1:n}^i (a_i^i)^{-1}$ 
7:        $[A]_{i+1:n}^{i+1:n} \leftarrow [A]_{i+1:n}^{i+1:n} - [L]_{i+1:n}^i [U]_i^{i+1:n}$ 
8:     end if
9:   end for
10:  return  $L, U$ 
11: end procedure

```

3.3.10. Remark (compact LU). In fact, a very compact way to implement the iteration is to rewrite L and U directly onto A , thus saving precious memory when systems are very large. Afterall the matrix A , if needed ever after, can be “recreated” by (post)multiplying L with U .

Require: $n \in \mathbb{N}, A \in \mathbb{K}^{n \times n}, c \in \mathbb{K}^n$

Ensure: A is LU-cky and $A = LU$ with $L \in \mathcal{L}(\mathbb{K}^n)$, $U \in \mathcal{U}(\mathbb{K}^n)$

```

1: procedure LU-COMPACT( $A$ )
2:   for  $i = 1, \dots, n$  do
3:     if  $i < n$  then
4:        $[A]_{i+1:n}^i \leftarrow [A]_{i+1:n}^i (a_i^i)^{-1}$ 

```

⁴Careful with “iterative” in this case, it means a different than “iterative” as in fixed-point iterative methods. In the computer science meaning of the word, iterative simply means a for loop, or a while loop, where the same operation is repeated as many times as needed rather than a function calling itself.

```

5:          $[A]_{i+1:n}^{i+1:n} \leftarrow [A]_{i+1:n}^{i+1:n} - [A]_{i+1:n}^i [A]_i^{i+1:n}$ 
6:     end if
7: end for
8: return  $L, U$ 
9: end procedure

```

3.3.11. Operation count in LU factorisation. The operation count (complexity) of LU factorisation is similar whether one uses the iterative or the recursive version. We use here the recursive version. Let $C(n)$ denote the complexity of LU-factorising a matrix A in $\mathbb{K}^{n \times n}$ then the cost of the scalar inversion (reciprocal) is

$$\text{cost}[(a_1^1)^{-1}] = r \quad (3.3.24)$$

and that of multiplying the small $(n-1)$ column

$$\text{cost}[[A]_{2:n}^1 (a_1^1)^{-1}] = n-1 \quad (3.3.25)$$

finally to form the matrix \hat{A} to be sent for recursion we need

$$\text{cost}[[A]_{2:n}^{2:n} - [A]_{2:n}^1 (a_1^1)^{-1} [A]_1^{2:n}] = 2(n-1)^2 \quad (3.3.26)$$

and the cost of applying $\text{RECURSIVE-LU-FACTORISATION}(\hat{A})$ is $C(n-1)$. In total we get, for $n \geq 2$,

$$\begin{aligned}
C(n) &= r + (n-1) + 2(n-1)^2 + C(n-1) \\
&= r + (n-1) + 2(n-1)^2 \\
&\quad + r + (n-2) + 2(n-2)^2 + C(n-2) \\
&= \dots \\
&= r(n-1) + \sum_{i=1}^{n-1} i + 2 \sum_{i=1}^{n-1} i^2 + C(1) \\
&= r(n-1) + \frac{1}{2}n(n-1) + \frac{2}{3}n^3 - n^2 + \frac{1}{3}n + 0 \\
&= \frac{2}{3}n^3 - \frac{1}{2}n^2 + \left(r - \frac{1}{6}\right)n - r \\
&\approx \frac{2}{3}n^3 + \text{lower order terms in } n.
\end{aligned} \quad (3.3.27)$$

The last sentence means that we really care about the highest power in n . The negative square seems like good news (because it lowers the computational cost), but for n large it is irrelevant in front of the cube which is the leading term as $n \rightarrow \infty$.

3.4. Pivoting and PLU factorisation

3.4.1. What about un-LU-cky matrices? Not all matrices are LU-cky, in fact, any singular matrix is un-LU-cky. Worse, there are invertible matrices that are un-LU-cky: think of

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (3.4.1)$$

This matrix is invertible, but it has no LU factorisation. It is a good exercise to show it has no LU factorisation.[*]

[*]: Check!

In this section we look on how to “cure” this problem by generalising the LU algorithm as to include all invertible (but not necessarily LU-cky) matrices. Denote by \mathbf{A} and \mathbf{M} two generic matrices, with \mathbf{A} invertible. An elementary permutation matrix, $\mathbf{P}_{i \leftrightarrow j} \in \mathbb{R}^{nn}$, and an elementary (normalised) lower triangular matrix, $\mathbf{L}_i(\mathbf{m}) \in \mathbb{K}^{nn}$ where \mathbf{m} is a column vector such that $\mathbf{e}^{k\top} \mathbf{m} = 0$ for all $k \leq i$, are given by

$$\mathbf{P}_{i \leftrightarrow j} := \mathbf{I} - (\mathbf{e}^i - \mathbf{e}^j)(\mathbf{e}^i - \mathbf{e}^j)^\top \text{ and } \mathbf{L}_i(\mathbf{m}) := \mathbf{I} - \mathbf{m} \mathbf{e}^{i\top}, \quad (3.4.2)$$

where $(\mathbf{e}^1, \dots, \mathbf{e}^n)$ denotes the canonical basis of \mathbb{K}^n . A general permutation matrix \mathbf{P} to be the product of elementary permutation matrices, i.e.,

$$\mathbf{P} = \mathbf{P}_{i_1 \leftrightarrow j_1} \cdots \mathbf{P}_{i_s \leftrightarrow j_s} \quad (3.4.3)$$

for some given $i_1, \dots, i_s, j_1, \dots, j_s \in 1:n$.

3.4.2. Permutation matrices. Describe the product of $\mathbf{P}_{i \leftrightarrow j}$ postmultiplied by a generic matrix \mathbf{M} and justify your answer algebraically. Show that $\mathbf{P}_{i \leftrightarrow j}$ is an involution, i.e.,

$$\mathbf{P}_{i \leftrightarrow j}^{-1} = \mathbf{P}_{i \leftrightarrow j}. \quad (3.4.4)$$

Inspect the k -th row of the matrix $\mathbf{P}_{i \leftrightarrow j} \mathbf{M}$ (denoting by \mathbf{X}_k the k -th row of a matrix \mathbf{X}) we have:

$$\begin{aligned} [\mathbf{P}_{i \leftrightarrow j} \mathbf{M}]_k &= \mathbf{e}^{k\top} \mathbf{P}_{i \leftrightarrow j} \mathbf{M} = \mathbf{e}^{k\top} (\mathbf{I} - (\mathbf{e}^i - \mathbf{e}^j)(\mathbf{e}^i - \mathbf{e}^j)^\top) \mathbf{M} \\ &= (\mathbf{e}^{k\top} - (\mathbf{e}^{k\top} \mathbf{e}^i - \mathbf{e}^{k\top} \mathbf{e}^j)(\mathbf{e}^i - \mathbf{e}^j)^\top) \mathbf{M} = (\mathbf{e}^k - (\delta_k^i - \delta_k^j)(\mathbf{e}^i - \mathbf{e}^j))^\top \mathbf{M} \\ &= \begin{cases} \mathbf{e}^{k\top} \mathbf{M} = \mathbf{M}_k & \text{if } k \neq i, j, \\ (\mathbf{e}^i + \mathbf{e}^j - \mathbf{e}^i)^\top \mathbf{M} = \mathbf{M}_j & \text{if } k = i, \\ (\mathbf{e}^j - \mathbf{e}^j + \mathbf{e}^i)^\top \mathbf{M} = \mathbf{M}_i & \text{if } k = j. \end{cases} \end{aligned} \quad (3.4.5)$$

In words, premultiplying \mathbf{M} by $\mathbf{P}_{i \leftrightarrow j}$ exchanges \mathbf{M} 's i -th row with its j -th row. The calculation above, with $\mathbf{M} = \mathbf{I}$ shows that

$$[\mathbf{P}_{i \leftrightarrow j}]_k = \begin{cases} \mathbf{e}^{k\top} & \text{if } k \neq i, j, \\ \mathbf{e}^{j\top} & \text{if } k = i \\ \mathbf{e}^{i\top} & \text{if } k = j. \end{cases} \quad (3.4.6)$$

In words, $\mathbf{P}_{i \leftrightarrow j}$ is the identity matrix \mathbf{I} with the i -th and j -th rows interchanged. It follows that

$$\mathbf{P}_{i \leftrightarrow j} \mathbf{P}_{i \leftrightarrow j} = \mathbf{I}, \quad (3.4.7)$$

and thus

$$\mathbf{P}_{i \leftrightarrow j}^{-1} = \mathbf{P}_{i \leftrightarrow j}. \quad (3.4.8)$$

3.4.3. Inversion of lower triangular matrices. Show that for any $\mathbf{m} \in \mathbb{K}^n$ the matrix $\mathbf{L}_i(\mathbf{m})$ is invertible and

$$\mathbf{L}_i(\mathbf{m})^{-1} = \mathbf{L}_i(-\mathbf{m}). \quad (3.4.9)$$

Describe the product of $\mathbf{L}_i(\mathbf{m})$ premultiplying a generic matrix \mathbf{M} .

Denoting $\mathbf{m} = [m_k]_{k=1, \dots, n}$ with $m_k = 0$ for $k \leq i$, the k -th row of the product can be calculated as follows:

$$\begin{aligned} [\mathbf{L}_i(\mathbf{m})\mathbf{M}]_k &= \mathbf{e}^{k\top}(\mathbf{I} - \mathbf{m}\mathbf{e}^{i\top})\mathbf{M} = (\mathbf{e}^{k\top} - \mathbf{e}^{k\top}\mathbf{m}\mathbf{e}^{i\top})\mathbf{M} \\ &= (\mathbf{e}^{k\top} - \mathbf{e}^{k\top}\mathbf{m}\mathbf{e}^{i\top})\mathbf{M} = \begin{cases} \mathbf{M}_k & \text{if } k \leq i, \\ \mathbf{M}_k - m_k\mathbf{M}_i & \text{if } k > i. \end{cases} \end{aligned} \quad (3.4.10)$$

In words, premultiplying \mathbf{M} by $\mathbf{L}_i(\mathbf{m})$ has the effect of leaving all rows up to and including the i -th intact, while subtracting m_k times the i -th row from the k -th row for $k > i$.

3.4.4. Some useful block-matrix algebra. Show that for any $\mathbf{m} \in \mathbb{K}^n$, $r = 2, \dots, n$, and \mathbf{M} of the block-diagonal form

$$\mathbf{M} := \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{M}' \end{bmatrix} \text{ where } \mathbf{M}' \in \mathbb{K}^{(n-1) \times (n-1)} \quad (3.4.11)$$

we have

$$\mathbf{M}\mathbf{L}_1(\mathbf{m}) = \mathbf{L}_1(\mathbf{q})\mathbf{M} \text{ where } \mathbf{q} = \mathbf{M}\mathbf{m}. \quad (3.4.12)$$

Using the block-multiplication of matrices we may write

$$\begin{aligned} \mathbf{M}\mathbf{L}_1(\mathbf{m}) &= \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{M}' \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{m}' & \mathbf{I} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{M}'\mathbf{m}' & \mathbf{M}' \end{bmatrix} \\ &= \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{M}'\mathbf{m}' & \mathbf{I} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{M}' \end{bmatrix} = \mathbf{L}_1(\mathbf{q})\mathbf{M} \text{ and } \mathbf{q} = \mathbf{M}\mathbf{m}. \end{aligned} \quad (3.4.13)$$

3.4.5. The Gaussian elimination step. For an invertible matrix $\mathbf{A} = [a_i^j]_{i=1, \dots, n}^{j=1, \dots, n} \in \mathbb{K}^{n \times n}$, explain how to find and describe $r_1 \in 1 : n$, $\mathbf{m}_1 \in \mathbb{K}^n$, $a(1) \in \mathbb{R}$, $\mathbf{a}(1) \in \mathbb{K}^{n-1}$ and $\mathbf{A}'(1)$ such that

$$\mathbf{L}_1(\mathbf{m}_1)\mathbf{P}_{1 \leftrightarrow r_1}\mathbf{A} = \mathbf{A}(1) = \begin{bmatrix} a(1) & \mathbf{a}(1)^\top \\ \mathbf{0} & \mathbf{A}'(1) \end{bmatrix}. \quad (3.4.14)$$

Since \mathbf{A} is invertible, there must be one element in its first column \mathbf{A}^1 , say the r_1 -th, $a_{r_1}^1$, which is nonzero. Let $\tilde{\mathbf{A}} = \mathbf{P}_{1 \leftrightarrow r_1}\mathbf{A}$, then we have

$$[\tilde{\mathbf{A}}]_k = \begin{cases} [\mathbf{A}]_{r_1} & \text{if } k = 1 \\ [\mathbf{A}]_k & \text{if } 1 < k \neq r_1 \\ [\mathbf{A}]_1 & \text{if } 1 < k = r_1. \end{cases} \quad (3.4.15)$$

Define

$$m_k := \tilde{a}_k^1 / \tilde{a}_1^1 = \begin{cases} 0 & \text{for } k = 1 \\ a_k^1 / a_{r_1}^1 & \text{for } 1 < k \neq r_1 \\ a_1^1 / a_{r_1}^1 & \text{for } 1 < k = r_1. \end{cases} \quad (3.4.16)$$

Then, upon introducing $\mathbf{B} := \mathbf{L}_1(\mathbf{m}_1)\tilde{\mathbf{A}}$ and fixing k , we see that

$$\begin{aligned} b_k^1 &= \mathbf{e}^{k\top} \mathbf{B} \mathbf{e}^1 = \mathbf{e}^{k\top} \mathbf{L}_1(\mathbf{m}_1) \mathbf{P}_{1 \leftrightarrow r_1} \mathbf{A} \mathbf{e}^k = ([\tilde{\mathbf{A}}]_k - m_k [\tilde{\mathbf{A}}]_1) \mathbf{e}^1 \\ &= \begin{cases} [\mathbf{A}]_{r_1} \mathbf{e}^1 = a_{r_1}^1 & \text{if } k = 1 \\ \mathbf{e}^{1\top} (\mathbf{a}_k - a_k^1/a_{r_1}^1 \mathbf{a}_{r_1}) = a_k^1 - (a_k^1/a_{r_1}^1) a_{r_1}^1 = 0 & \text{if } 1 < k \neq r_1 \\ a_k^1 - (a_1^1/a_{r_1}^1) a_{r_1}^1 = 0 & \text{if } 1 < k = r_1. \end{cases} \end{aligned} \quad (3.4.17)$$

This means that \mathbf{B} is of the form

$$\mathbf{B} = \begin{bmatrix} a(1) & \mathbf{a}(1)^\top \\ \mathbf{0} & \mathbf{A}'(1) \end{bmatrix} \quad (3.4.18)$$

with

$$\begin{bmatrix} a(1) & \mathbf{a}(1)^\top \end{bmatrix} = \mathbf{a}_{r_1} \text{ and } \mathbf{A}'(1) = [\mathbf{L}_1(\mathbf{m}_1)\tilde{\mathbf{A}}]_{2:n}^{2:n} \quad (3.4.19)$$

3.4.6. Group structure. Denote by $\mathcal{L}(\mathbb{K}^n)$ the set of all normalised lower triangular matrices and $\mathcal{P}(n)n$ the set of all permutation matrices. Explain what is meant by “ $\mathcal{L}(\mathbb{K}^n)$ and $\mathcal{P}(n)n$ are multiplicative groups” and why this is useful to deduce that there are $\mathbf{P} \in \mathcal{P}(n)n$, $\mathbf{L} \in \mathcal{L}(\mathbb{K}^n)$ and $\mathbf{U} \in \mathbb{K}^{n \times n}$ upper triangular, such that

$$\mathbf{A} = \mathbf{P} \mathbf{L} \mathbf{U}. \quad (3.4.20)$$

Argue by induction on n . The result is trivially true for $n = 1$, with $\mathbf{U} = \mathbf{A}$ and $\mathbf{L} = \mathbf{P} = \mathbf{I} = 1$. Suppose it is true in \mathbb{K}^{n-1} , using r_1 and \mathbf{m}_1 found in step 3.4.5 (and keeping the same notation as therein) introduce $\mathbf{L}(1) := \mathbf{L}_1(\mathbf{m}_1)$ and $\mathbf{P}(1) := \mathbf{P}_{1 \leftrightarrow r_1}$.

It follows that $\mathbf{A}'(1)$ is invertible, so, using the inductive hypothesis, we know that there are $\mathbf{P}' \in \mathcal{P}(n)n-1$, $\mathbf{L}' \in \mathcal{L}(\mathbb{K}^{n-1})$ and $\mathbf{U}' \in \mathbb{K}^{(n-1) \times (n-1)}$ upper triangular such that

$$\mathbf{A}'(1) = \mathbf{P}' \mathbf{L}' \mathbf{U}'. \quad (3.4.21)$$

Define now

$$\tilde{\mathbf{P}} := \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{P}' \end{bmatrix}, \tilde{\mathbf{L}} := \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{L}' \end{bmatrix} \text{ and } \mathbf{U} := \begin{bmatrix} a(1) & \mathbf{a}(1)^\top \\ \mathbf{0} & \mathbf{U}' \end{bmatrix}. \quad (3.4.22)$$

Immediately we have

$$\tilde{\mathbf{P}} \in \mathcal{P}(n)n, \tilde{\mathbf{L}} \in \mathcal{L}(\mathbb{K}^n) \text{ and } \mathbf{U} \text{ upper triangular.} \quad (3.4.23)$$

Furthermore, a bit of algebra implies

$$\tilde{\mathbf{P}} \tilde{\mathbf{L}} \mathbf{U} = \begin{bmatrix} a(1) & \mathbf{a}(1)^\top \\ \mathbf{0} & \mathbf{A}'(1) \end{bmatrix} = \mathbf{L}(1) \mathbf{P}(1) \mathbf{A}, \quad (3.4.24)$$

whence

$$\tilde{\mathbf{L}} \mathbf{U} = \tilde{\mathbf{P}}^{-1} \mathbf{L}(1) \mathbf{P}(1) \mathbf{A}, \text{ with } \tilde{\mathbf{P}}^{-1} = \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{P}'^{-1} \end{bmatrix} \quad (3.4.25)$$

But 3.4.4 tells us that

$$\tilde{\mathbf{P}}^{-1} \mathbf{L}(1) = \tilde{\mathbf{P}}^{-1} \mathbf{L}_1(\mathbf{m}_1) = \mathbf{L}_1(\tilde{\mathbf{P}}^{-1} \mathbf{m}_1) \tilde{\mathbf{P}}^{-1} \quad (3.4.26)$$

and thus

$$\tilde{\mathbf{L}} \mathbf{U} = \mathbf{L}_1(\tilde{\mathbf{P}}^{-1} \mathbf{m}_1) \tilde{\mathbf{P}}^{-1} \mathbf{P}(1) \mathbf{A}, \quad (3.4.27)$$

which, after inverting using (??), is equivalent to

$$\underbrace{\mathbf{L}_1(-\tilde{\mathbf{P}}^{-1} \mathbf{m}_1) \tilde{\mathbf{L}} \mathbf{U}}_{=: \mathbf{L}} = \mathbf{L}_1(\tilde{\mathbf{P}}^{-1} \mathbf{m}_1)^{-1} \tilde{\mathbf{L}} \mathbf{U} = \underbrace{\tilde{\mathbf{P}}^{-1} \mathbf{P}(1) \mathbf{A}}_{=: \mathbf{P}^{-1}} \quad (3.4.28)$$

Thanks to the group properties $\mathbf{L} \in \mathcal{L}(\mathbb{K}^n)$ and $\mathbf{P} \in \mathcal{P}(n)$ and, as required, they satisfy

$$\mathbf{P}^{-1}\mathbf{A} = \mathbf{L}\mathbf{U}, \text{ and hence } \mathbf{A} = \mathbf{P}\mathbf{L}\mathbf{U}. \quad (3.4.29)$$

3.4.7. Example (PLU factorisation). Apply the LU algorithm *with row pivoting* to the following matrix

$$\begin{bmatrix} 0 & -2 & 3 \\ 2 & -6 & 2 \\ 3 & 0 & -3 \end{bmatrix} \quad (3.4.30)$$

$$\mathbf{P}_{1 \leftrightarrow 3} \begin{bmatrix} 0 & -2 & 3 \\ 2 & -6 & 2 \\ 3 & 0 & -3 \end{bmatrix} = \begin{bmatrix} 3 & 0 & -3 \\ 2 & -6 & 2 \\ 0 & -2 & 3 \end{bmatrix} \quad (3.4.31)$$

$$\mathbf{L}_1 \left(\begin{bmatrix} 0 \\ 2/3 \\ 0 \end{bmatrix} \right) \begin{bmatrix} 3 & 0 & -3 \\ 2 & -6 & 2 \\ 0 & -2 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 0 & -3 \\ 2 - 2/3 \times 3 & -6 & 2 + 2/3 \times 3 \\ 0 & -2 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 0 & -3 \\ 0 & -6 & 4 \\ 0 & -2 & 3 \end{bmatrix} \quad (3.4.32)$$

$$\mathbf{L}_1 \left(\begin{bmatrix} 0 \\ 1/3 \end{bmatrix} \right) \begin{bmatrix} -6 & 4 \\ -2 & 3 \end{bmatrix} = \begin{bmatrix} -6 & 4 \\ -2 + 6/3 & 3 - 4/3 \end{bmatrix} = \begin{bmatrix} -6 & 4 \\ 0 & 5/3 \end{bmatrix} \quad (3.4.33)$$

Hence, denoting the starting matrix \mathbf{A} , we have

$$\mathbf{L}_2 \left(\begin{bmatrix} 0 \\ 0 \\ 1/3 \end{bmatrix} \right) \mathbf{L}_1 \left(\begin{bmatrix} 0 \\ 2/3 \\ 0 \end{bmatrix} \right) \mathbf{P}_{1 \leftrightarrow 3} \mathbf{A} = \begin{bmatrix} 3 & 0 & -3 \\ 0 & -6 & 4 \\ 0 & 0 & 5/3 \end{bmatrix} =: \mathbf{U} \quad (3.4.34)$$

or, equivalently,

$$\mathbf{A} = \mathbf{P}_{1 \leftrightarrow 3} \mathbf{L}_1 \left(\begin{bmatrix} 0 \\ -2/3 \\ 0 \end{bmatrix} \right) \mathbf{L}_2 \left(\begin{bmatrix} 0 \\ 0 \\ -1/3 \end{bmatrix} \right) \mathbf{U} = \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{=: \mathbf{P}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2/3 & 1 & 0 \\ 0 & 1/3 & 1 \end{bmatrix}}_{=: \mathbf{L}} \underbrace{\begin{bmatrix} 3 & 0 & -3 \\ 0 & -6 & 4 \\ 0 & 0 & 5/3 \end{bmatrix}}_{=: \mathbf{U}}. \quad (3.4.35)$$

Exercises and problems on Linear systems and Gaussian elimination

Exercise 3.1 (back-substitution). (a) Solve the following linear system by hand with back substitution:

$$\begin{bmatrix} 1 & 1 & 1 \\ & 2 & 2 \\ & & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ 6 \end{bmatrix}. \quad (\text{P3.1.1})$$

(b) Solve the following linear system by hand with the back substitution algorithm:

$$\begin{bmatrix} 2 & -1 & 3 & 1 \\ & 1 & 2 & -1 \\ & & -2 & 1 \\ & & & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ -3 \\ 1 \\ 9 \end{bmatrix}. \quad (\text{P3.1.2})$$

Exercise 3.2 (Gaussian elimination). Bring the following linear system with Gaussian elimination into upper triangular form:

$$\mathbf{Ax} = \mathbf{b}, \quad \mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 1 & 2 \\ 2 & 2 & 4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 8 \\ 4 \end{bmatrix}. \quad (\text{P3.2.1})$$

Then use back substitution to solve the new system in upper triangular form.

Exercise 3.3 (lower triangular matrices). Write down three different 4×4 elementary lower triangular matrices $\mathbf{L}_i(\mathbf{x})$, and explain for each matrix what left-multiplication with this matrix does.

Verify explicitly that

$$\det \mathbf{L}_i(\mathbf{x}) = 1 \text{ and } \mathbf{L}_i(\mathbf{x})^{-1} = \mathbf{L}_i(-\mathbf{x}) \quad (\text{P3.3.1})$$

for each of these three elementary lower triangular matrices.

Exercise 3.4 (lower triangular matrices). Find the elementary lower triangular matrices $\mathbf{L}_1(\mathbf{m}^1)$ and $\mathbf{L}_2(\mathbf{m}^2)$ which describe the Gaussian elimination to bring the linear system

$$\mathbf{Ax} = \mathbf{b}, \quad \mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 1 & 2 \\ 2 & 2 & 4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 8 \\ 4 \end{bmatrix} \quad (\text{P3.4.1})$$

into the form $\mathbf{Ux} = \mathbf{b}'$ with an upper triangular 3×3 matrix \mathbf{U} and a new right-hand side $\mathbf{b}' \in \mathbb{R}^3$. Explain why you chose those \mathbf{m}_j in the $\mathbf{L}_j(\mathbf{m}^j)$ and verify the result (in particular, do the Gaussian elimination to compute \mathbf{U} and \mathbf{b}').

Exercise 3.5 (properties of lower triangular matrices). Prove that an elementary lower triangular $n \times n$ matrix $\mathbf{L}_i(\mathbf{x}) = \mathbf{I} - \mathbf{x}\mathbf{e}^{i\top}$ where

$$\mathbf{x} = [0 \quad \dots \quad 0 \quad x_{i+1} \quad \dots \quad x_n]^\top \quad (\text{P3.5.1})$$

has the following properties:

- (a) $\det \mathbf{L}_i(\mathbf{x}) = 1$.
- (b) $\mathbf{L}_i(\mathbf{x})^{-1} = \mathbf{L}_i(-\mathbf{x})$.

- (c) Premultiplying (i.e., multiplying from the left) a matrix $\mathbf{A} = [a_i^j]_{i=1, \dots, n}^{j=1, \dots, n} \in \mathbb{K}^{n \times n}$, $\mathbb{K} = \mathbb{R}$ or \mathbb{C} , by $\mathbf{L}_i(\mathbf{x})$ leaves the first i rows unchanged and, starting from row $k = i + 1$, it subtracts \mathbf{A} 's i -th row scaled by x_k ,

$$x_k [a_i^1 \quad a_i^2 \quad \dots \quad a_i^n] = x_k \mathbf{a}_i, \quad (\text{P3.5.2})$$

from \mathbf{A} 's k -th row, \mathbf{a}_k .

Problem 3.6 (triangular matrix algebra). Consider the following sets of matrices.

$$\mathcal{U}(\mathbb{K}^n) := \{\mathbf{X} \in \mathbb{K}^{n \times n} : \mathbf{X} \text{ is upper triangular}\}, \quad (\text{P3.6.1})$$

$$\mathcal{L}(\mathbb{K}^n) := \{\mathbf{X} \in \mathbb{K}^{n \times n} : \mathbf{X} \text{ is normalised lower triangular}\}, \quad (\text{P3.6.2})$$

and

$$\mathcal{D}(\mathbb{K}^n) := \{\mathbf{X} \in \mathbb{K}^{n \times n} : \mathbf{X} \text{ is diagonal}\}. \quad (\text{P3.6.3})$$

Prove the following:

- (a) The sets $\mathcal{U}(\mathbb{K}^n)$ and $\mathcal{D}(\mathbb{K}^n)$ are closed under scaling, matrix addition and matrix multiplication, and are thus algebras of operators.
- (b) The set $\mathcal{L}(\mathbb{K}^n)$ is closed under matrix multiplication and is a (non-Abelian) group with respect to this operation.

Problem 3.7 (invertibility criterion of upper triangular matrices). Prove that if $\mathbf{U} \in \mathbb{K}^{n \times n}$, $n \in \mathbb{N}$ is an upper triangular and $u_k^k = 0$ for some $k = 1, \dots, n$ then there exists a nonzero vector \mathbf{v} such that $\mathbf{U} \mathbf{v} = \mathbf{0}$. Deduce that an upper triangular matrix is *invertible* if and only if it has all its diagonal entries are invertible.

Problem 3.8 (recursive definition of upper triangular). Prove the following recursive definition:

A matrix

$$\mathbf{U} = \begin{bmatrix} u & \mathbf{t}^\top \\ \mathbf{s} & \mathbf{R} \end{bmatrix} \quad (\text{P3.8.1})$$

is in $\mathcal{U}(\mathbb{K}^n)$ if and only if either \mathbf{U} is empty (when $n = 0$), or

$$\mathbf{s} = \begin{cases} \emptyset & \text{for } n = 1 \\ \mathbf{0} & \text{for } n \geq 2 \end{cases} \text{ and } \mathbf{R} \in \mathcal{U}(\mathbb{K}^{n-1}). \quad (\text{P3.8.2})$$

Problem 3.9 (upper triangular algebra). Denote by $\mathcal{U}(\mathbb{K}^n)$ the set of all upper triangular $n \times n$ matrices on a field $\mathbb{K} = \mathbb{R}$ or \mathbb{C} .

- (a) Prove that $\mathcal{U}(\mathbb{K}^n)$ is closed under matrix-matrix multiplication, addition and scaling. Deduce that $\mathcal{U}(\mathbb{K}^n)$ is a \mathbb{K} -vector space, and thus a \mathbb{K} -algebra.
- (b) Show, with a counterexample, that $\mathcal{U}(\mathbb{K}^n)$ is *not* a group with respect to multiplication.
- (c) Show that if $\mathbf{U} \in \mathcal{U}(\mathbb{K}^n)$ is invertible, then $\mathbf{U}^{-1} \in \mathcal{U}(\mathbb{K}^n)$. That is, $\mathcal{U}(\mathbb{K}^n)$ is closed under matrix inversion.
- (d) Show that $\mathbf{U} \in \mathcal{U}(\mathbb{K}^n)$ is invertible if and only if $u_1^1 \neq 0$ and $[\mathbf{U}]_{2:n}^{2:n}$ is invertible. Deduce that $\mathbf{U} \in \mathcal{U}(\mathbb{K}^n)$ is invertible if and only if $u_i^i \neq 0$ for all $i = 1, \dots, n$.

Hint. You may assume the result of Problem 3.8.

Exercise 3.10 (LU factorisation). (a) Recall that we call a matrix *diagonally normalised*, or just *normalised*, if all its diagonal entries equal 1. Find an LU factorisation of

$$A = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 1 & 2 \\ 2 & 2 & 4 \end{bmatrix}, \quad (\text{P3.10.1})$$

of the form $A = LU$ where L is a *normalised* lower triangular matrix and U an upper triangular matrix.

- (b) Prove that there is only one such LU factorisation of A .
(c) What happens if we drop the assumption that L be normalised?

Problem 3.11 (LU-cky matrices are invertible). Recall the following definition:
A square matrix $A \in \mathbb{K}^{n \times n}$ written

$$A = \begin{bmatrix} a & \mathbf{b}^\top \\ \mathbf{c} & D \end{bmatrix} \quad (\text{P3.11.1})$$

is *LU-cky* if and only if either A is empty (when $n = 0$), or

- (i) the first diagonal entry is invertible, $a \neq 0$
and
(ii) the matrix $D - \mathbf{c}a^{-1}\mathbf{b}^\top$ is LU-cky.

Recall also the LU-factorisation theorem:

If a matrix $A \in \mathbb{K}^{n \times n}$ for some $n \in \mathbb{N}$ is LU-cky then A admits a unique LU factorisation, i.e., there exists a unique pair $L \in \mathcal{L}(\mathbb{K}^n)$ and $U \in \mathcal{U}(\mathbb{K}^n)$ such that

$$A = LU. \quad (\text{P3.11.2})$$

- (a) Show that if A is LU-cky then U is invertible.
(b) Deduce that a LU-cky matrix A must be invertible.

Problem 3.12 (LU-cky LU factorisation is unique). Let A be a LU-cky matrix. Prove that if it has an LU factorisation, then it has at most one.

Hint. You may assume the results of 3.9 and 3.11 are established.

Problem 3.13 (permutation matrix properties). Prove that an elementary permutation matrix

$$\mathbf{P}_{i \leftrightarrow j} := \mathbf{I} - (\mathbf{e}^i - \mathbf{e}^j)(\mathbf{e}^i - \mathbf{e}^j)^\top \quad (\text{P3.13.1})$$

enjoys the following properties:

- (i) $\mathbf{P}_{i \leftrightarrow j}^{-1} = \mathbf{P}_{i \leftrightarrow j} = \mathbf{P}_{j \leftrightarrow i} = \mathbf{P}_{i \leftrightarrow j}^\top$. In particular, $\mathbf{P}_{i \leftrightarrow j}$ is an orthogonal matrix.
(ii) $\det \mathbf{P}_{i \leftrightarrow j} = -1$ for any $i \neq j$, and, for any i $\mathbf{P}_{i \leftrightarrow i} = \mathbf{I}$.
(iii) Premultiplying a matrix A by $\mathbf{P}_{i \leftrightarrow j}$ interchanges *rows* i and j . Similarly, post-multiplication interchanges *columns* i and j .

Problem 3.14 (programming PLU). The purpose of this problem is to produce a code for the LU factorisation with pivoting.

- (a) Start by implementing a recursive code, say

$$[L, U] = \text{recursiveLU}(A) \quad (\text{P3.14.1})$$

for a *LUcky* matrix \mathbf{A} . The matrix \mathbf{A} is *LUcky* if $a_1^1 \neq 0$ and the matrix $\hat{\mathbf{A}}$ is *LUcky*; the matrix $\hat{\mathbf{A}}$ being the block $[\tilde{\mathbf{A}}]_{2:n}^{2:n}$, with $\tilde{\mathbf{A}}$ obtained by row-reducing the first column of \mathbf{A}

$$\tilde{\mathbf{A}} = \mathbf{A} - \mathbf{m} [\mathbf{A}]_1. \quad (\text{P3.14.2})$$

The recursion should be based on the fact that if \mathbf{A} is *LUcky* and passed to the function then $a_1^1 \neq 0$ and taking $\hat{\mathbf{m}} = [\mathbf{A}]_{2:n}^1 / a_1^1$, $\mathbf{m} = (0, \hat{\mathbf{m}})$ we get

$$\mathbf{A} = \mathbf{L}_1(-\mathbf{m})\tilde{\mathbf{A}} \text{ where } \tilde{\mathbf{A}} = \begin{bmatrix} a_1^1 & [\mathbf{A}]_1^{2:n} \\ \mathbf{0} & \hat{\mathbf{A}} \end{bmatrix}. \quad (\text{P3.14.3})$$

The matrix $\hat{\mathbf{A}} \in \mathbb{R}^{(n-1) \times (n-1)}$ can now be passed to the function, which should return $\hat{\mathbf{L}}$, normalised lower triangular, and $\hat{\mathbf{U}}$, upper triangular, such that

$$\hat{\mathbf{A}} = \hat{\mathbf{L}}\hat{\mathbf{U}}. \quad (\text{P3.14.4})$$

Do not forget to implement the bottom case when $n = 1$.

- (b) Show with a counterexample in $\mathbb{R}^{2 \times 2}$, that *not all invertible matrices are LUcky*. It is hence necessary to modify the LU factorisation algorithm to make it more robust.
- (c) Let π be an *index-permutation* on $\{1, \dots, n\}$ (i.e., a bijection from $\{1, \dots, n\}$ into itself), we associate to π the *natural basis permutation* linear map (or matrix) \mathbf{P}_π characterised by

$$\mathbf{P}_\pi \mathbf{e}^j = \mathbf{e}^{\pi(j)}. \quad (\text{P3.14.5})$$

Show that if τ is an index-transposition such that $\tau(i) = j$, $\tau(j) = i$, then the matrix $\mathbf{P}_\tau = \mathbf{P}_{i \leftrightarrow j}$, the elementary permutation matrix. Then show that for any index-permutation π

$$\mathbf{P}_\pi^* \mathbf{P}_\pi = \mathbf{I} \quad (\mathbf{P}_\pi \text{ is unitary}) \quad \mathbf{P}_\pi^{-1} = \mathbf{P}_{\pi^{-1}} \quad (\text{P3.14.6})$$

and thus

$$\mathbf{P}_\pi^* = \mathbf{P}_{\pi^{-1}}. \quad (\text{P3.14.7})$$

Deduce that

$$[\mathbf{P}_\pi \mathbf{v}]_l = v_{\pi^{-1}(l)} \quad (\text{P3.14.8})$$

where π^{-1} is the inverse of π . Show that if ρ and π are index-permutations then

$$[\mathbf{P}_\rho \mathbf{P}_\pi \mathbf{v}]_l = v_{\pi^{-1}(\rho^{-1}(l))}. \quad (\text{P3.14.9})$$

(Watch the order of composition.)

- (d) Based on the previous point think of a quick way of encoding an index-permutation π by using its *inverse-permutation vector* pinverse such that

$$\text{pinverse}(i) = \pi^{-1}(i). \quad (\text{P3.14.10})$$

In particular if \mathbf{v} is encoded as vector then $\mathbf{P}_\pi \mathbf{v}$ can be simply calculated as the result of the operation

$$\text{vector}(\text{pinverse}). \quad (\text{P3.14.11})$$

Use the following example to test all the objects described above

$$\pi(1) = 3, \pi(2) = 4, \pi(3) = 2, \pi(4) = 1. \quad (\text{P3.14.12})$$

- (e) Modify the LU factorisation code as to allow for the elementary permutation $\mathbf{P}_{1 \leftrightarrow r}$ to be included. The function should now return a vector `pinverse`, and the matrices L, U . You will need to “invert” the vector `pinverse` in the sense that you have to find the inverse-permutation vector. In view of this it is good to experiment with (and understand) the following Octave, or Matlab[®], lines

```
> myperm([4,3,1,2])=(1:4)
> myperm = myperm(myperm)
> myperm = myperm(myperm)
> ...
```

(P3.14.13)

Once this works, write a function that takes any matrix A as input and returns $\mathbf{P} \in \mathcal{P}(n)$, $\mathbf{L} \in \mathcal{L}(\mathbb{K}^n)$ and $\mathbf{U} \in \mathcal{U}(\mathbb{K}^n)$ such that

$$\mathbf{A} = \mathbf{P}\mathbf{L}\mathbf{U}. \quad (\text{P3.14.14})$$

Test your code with up to five benchmark cases for each of $n = 2, 4, 8, 16, 32$.

CHAPTER 5

Direct matrix factorisation methods

"Dear Colleagues,

For many years, I have been interested in meeting J G F Francis, one of the co-inventors of the QR algorithm for computing eigenvalues of general matrices. Through a lead provided by the late Erin Brent and with the aid of Google, I finally made contact with him.

John Francis was born in 1934 in London and currently lives in Hove, near Brighton. His residence is about a quarter mile from the sea; he is a widower. In 1954, he worked at the National Research Development Corp (NRDC) and attended some lectures given by Christopher Strachey. In 1955–56 he was a student at Cambridge but did not complete a degree. He then went back to NRDC as an assistant to Strachey where he got involved in flutter computations and this led to his work on QR.

After leaving NRDC in 1961, he worked at the Ferranti Corp and then at

the University of Sussex. Subsequently, he had positions with various industrial organizations and consultancies. He is now retired. His interests were quite general and included Artificial Intelligence, computer languages, systems engineering. He has not returned to numerical computation.

He was surprised to learn there are many references to his work and that the QR method is considered one of the ten most important algorithms of the 20th century. He was unaware of such developments as TeX and Math Lab. Currently he is working on a degree at the Open University.

John Francis did remarkable work and we are all in his debt. Along with the conjugate gradient method, it provided us with one of the basic tools of numerical analysis."

– Gene Golub (1932–2007)

[NA-digest, Sun, 19 Aug 2007 13:54:47 -0700 (PDT) Subject: John Francis, Co-Inventor of QR]

In Chapter 3 we have learned the LU factorisation and some of its variants. We now turn our attention to other factorisations, such as the Schur factorisation and the QR factorisation. We will also revisit LU factorisation for self-adjoint (and real-symmetric) matrices.

5.1. Reminders

5.1.1. Euclidean spaces as models of finite dimensional Hilbert spaces. We refer to Appendix A for most of the basic requirements on vector spaces, linear maps and spectral theory. Unless otherwise stated, the spaces \mathbb{K}^n are considered with the *Euclidean inner product*:

$$\mathbb{K}^{n+n} \ni (\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}^* \mathbf{y} \in \mathbb{R}, \quad (5.1.1)$$

where row-vector \mathbf{x}^* is the *adjoint* of the (column) vector \mathbf{x} , whose entries are the corresponding complex conjugate, that is, in symbols

$$[\mathbf{x}^*]^i = \overline{[\mathbf{x}]_i}. \quad (5.1.2)$$

The corresponding *Euclidean length* (also known as *length*) of a vector \mathbf{x} is defined as

$$|\mathbf{x}| = \sqrt{\mathbf{x}^* \mathbf{x}}. \quad (5.1.3)$$

The Cauchy–Bunyakovskii–Schwarz inequality states that for any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{K}^n$ we have

$$\mathbf{x}^* \mathbf{y} \leq |\mathbf{x}| |\mathbf{y}| \quad (5.1.4)$$

where equality is realised if and only if $\mathbf{y} = \alpha \mathbf{x}$ for some $\alpha \in \mathbb{K}$. The *angle-cosine* of two nonzero vectors is, by definition, given as

$$\cos \text{angle}(\mathbf{x}, \mathbf{y}) := \sqrt{\frac{\mathbf{x}^* \mathbf{y} \mathbf{y}^* \mathbf{x}}{\mathbf{x}^* \mathbf{x} \mathbf{y}^* \mathbf{y}}} \quad (5.1.5)$$

By the properties of the inner product and Cauchy–Bunyakovskii–Schwarz inequality we see that the angle-cosine is a real number in $[0, 1]$ and the square root makes usual sense. Also the angle-cosine of \mathbf{x} and \mathbf{y} is 0 if and only if $\mathbf{y}^* \mathbf{x} = 0$, in which case \mathbf{y} and \mathbf{x} are said to form an *orthogonal pair*.

5.1.2. Definition of adjoint operator. Let V, W be two inner product (also known as pre-Hilbert) vector spaces with inner products $\langle \cdot, \cdot \rangle_V$ and $\langle \cdot, \cdot \rangle_W$ respectively. The adjoint of an operator $\mathcal{A} \in \text{Lin}(V \rightarrow W)$ is a linear operator $\mathcal{B} \in \text{Lin}(W \rightarrow V)$ (note the reversed roles of V and W) such that

$$\langle v, \mathcal{B} w \rangle_V = \langle \mathcal{A} v, w \rangle_W \quad \forall v \in V, w \in W. \quad (5.1.6)$$

5.1.3. Proposition (existence and uniqueness of the adjoint operator). *If two inner product space V, W are Hilbert spaces, then each operator $\mathcal{A} \in \text{Lin}(V \rightarrow W)$ admits a unique adjoint operator, which is denoted by \mathcal{A}^* , in $\text{Lin}(W \rightarrow V)$.*

Proof The operator \mathcal{A} gives rise to a sesquilinear form as follows:

$$\begin{aligned} b : V \times W &\rightarrow \mathbb{K} \\ (v, w) &\mapsto \langle \mathcal{A} v, w \rangle_W \end{aligned} \quad (5.1.7)$$

By Theorem (A.9.14) the sesquilinear form b induces a linear operator, call it $\mathcal{B} \in \text{Lin}(W \rightarrow V)$ such that

$$\langle v, \mathcal{B} w \rangle_V = b(v, w) \quad \forall (v, w) \in V \times W. \quad (5.1.8)$$

Therefore the operator \mathcal{B} satisfies

$$\langle v, \mathcal{B} w \rangle_V = \langle \mathcal{A} v, w \rangle_W \quad \forall (v, w) \in V \times W, \quad (5.1.9)$$

which proves existence. To prove uniqueness, suppose that for an operator $\mathcal{C} \in \text{Lin}(W \rightarrow V)$ such that

$$\langle v, \mathcal{C} w \rangle_V = \langle \mathcal{A} v, w \rangle_W \quad \forall (v, w) \in V \times W. \quad (5.1.10)$$

It follows that

$$\langle v, (\mathcal{C} - \mathcal{B}) w \rangle_V = 0 \quad \forall (v, w) \in V \times W. \quad (5.1.11)$$

For any $w \in W$, choosing $v := (\mathcal{C} - \mathcal{B}) w$ in (5.1.11) we obtain

$$\langle (\mathcal{C} - \mathcal{B}) w, (\mathcal{C} - \mathcal{B}) w \rangle_V = 0, \quad (5.1.12)$$

which, by definiteness of $\langle \cdot, \cdot \rangle_V$, implies

$$(\mathcal{C} - \mathcal{B}) w = 0. \quad (5.1.13)$$

Since w is arbitrary in W , this means that the operator $\mathcal{C} - \mathcal{B}$ is null, or, equivalently, that $\mathcal{C} = \mathcal{B}$. \square

5.1.4. Definition of adjoint matrix. Similarly to adjoint operator we define the *adjoint of a matrix* A to be that unique matrix A^* such that

$$(x^*)(A^*)y = (Ax)^*y \quad \forall x, y \in \mathbb{K}^n. \quad (5.1.14)$$

(Overbracketing intentional to convey meaning.)

5.1.5. Remark (confused by adjoint and adjoint?) Don't worry. The definition is so designed that the matrix B representing the adjoint \mathcal{A}^* of an operator \mathcal{A} represented by A (in a given basis) is given by $B = A^*$.

5.1.6. Remark (transpose and adjoint). The transpose (of an operator and that) of a matrix is related to the adjoint:

$$A^* = \overline{A^T}. \quad (5.1.15)$$

5.1.7. Definition of self-adjoint, symmetrix and Hermitian operators and matrices. An operator $\mathcal{A} \in \text{Lin}(V \rightarrow V)$, V Hilbert \mathbb{K} -vector space, or a matrix $A \in \mathbb{K}^{n \times n}$, $n \in \mathbb{N}$, are called *self-adjoint* if and only if

$$\mathcal{A}^* = \mathcal{A} \text{ or } A^* = A, \text{ respectively.} \quad (5.1.16)$$

If the space $\mathbb{K} = \mathbb{R}$ a self-adjoint operator or matrix is called *symmetric*. If the space $\mathbb{K} = \mathbb{C}$ then we say that the operator or matrix is *Hermitian*. We will mostly employ *self-adjoint*.

5.2. Orthogonality

5.2.1. Definition of orthogonal operator. Let V be a complete inner-product (also known as Hilbert) vector space. An operator $\mathcal{Q} \in \text{Lin}(V \rightarrow V)$ is called *orthogonal* or *unitary* if and only if

$$\langle \mathcal{Q} x, \mathcal{Q} y \rangle = \langle x, y \rangle \quad \forall x, y \in V. \quad (5.2.1)$$

5.2.2. Proposition. An operator $\mathcal{Q} \in \text{Lin}(V \rightarrow V)$ is orthogonal (or unitary) if and only if

$$\mathcal{Q}^* \mathcal{Q} = \mathcal{I} = \text{id}_V. \quad (5.2.2)$$

5.2.3. Definition of unitary or orthogonal matrix. A matrix $Q \in \mathbb{K}^{n \times n}$ is called *unitary*, if and only if

$$Q^* Q = I. \quad (5.2.3)$$

If Q is unitary and $Q \in \mathbb{R}^{n \times n}$ we say that Q is *orthogonal*.

5.2.4. Definition of orthonormal system. Let V be an inner-product space and $\mathcal{S} \subseteq V$, we say that \mathcal{S} is an *orthonormal system* if and only if

$$\langle v, w \rangle = \delta_v^w \quad \forall v, w \in \mathcal{S}. \quad (5.2.4)$$

5.2.5. Definition of orthonormal basis. A orthonormal basis is a subset \mathcal{V} of V such that \mathcal{V} is an orthonormal system, and for any $v \in V$ there exists a (finite or infinite) sequence $(v^i)_{i \in I}$ in \mathcal{V} and a corresponding $(\alpha_i)_{i \in I}$ such that

$$v = \sum_{i \in I} \alpha_i v^i, \quad (5.2.5)$$

where, when I is infinite, the convergence is absolute, i.e., the series

$$\sum_{i \in I} \|\alpha_i v^i\| < \infty. \quad (5.2.6)$$

5.2.6. Proposition. Let $Q \in \mathbb{K}^{n \times n}$ The following are equivalent

- (i) Q is unitary,
- (ii) $x^* Q y = (Q x)^* y$ for all $x, y \in \mathbb{K}^n$,
- (iii) the columns of Q form an orthonormal basis of \mathbb{K}^n ,
- (iv) Q is invertible and $Q^{-1} = Q^*$.

Proof Exercise. □

5.2.7. Proposition (spectral properties of unitary matrices).

- (a) A unitary operator \mathcal{Q} has all its eigenvalues of absolute value 1.
- (b) A unitary matrix has all eigenvalues and determinant equal 1 in absolute value.
- (c) An orthogonal matrix Q has determinant equal ± 1 .

Proof Exercise. □

5.2.8. Example. The matrices

$$\begin{bmatrix} \sqrt{3}/2 & -1/2 \\ 1/2 & \sqrt{3}/2 \end{bmatrix}, \text{ and } \frac{1}{\sqrt{3}} \begin{bmatrix} i & -1-i \\ 1+i & -i \end{bmatrix} \quad (5.2.7)$$

are unitary. The first one is also orthogonal.

$$\begin{bmatrix} 1/\sqrt{3} & -\sqrt{2}/\sqrt{3} \\ -1/\sqrt{3} & 1/\sqrt{2} & -1/\sqrt{6} \\ 1/\sqrt{3} & 1/\sqrt{2} & 1/\sqrt{6} \end{bmatrix} \quad (5.2.8)$$

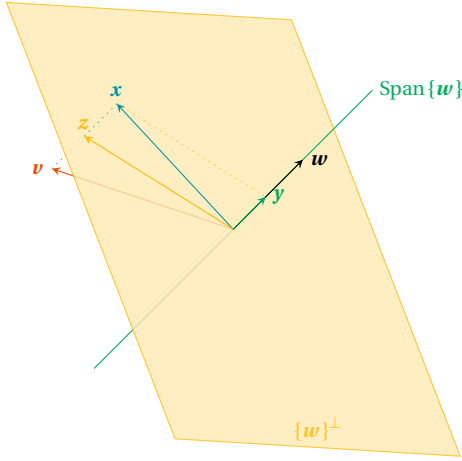
5.2.9. Definition of unitary and orthogonal groups. The orthogonal and unitary group are give as.

$$\begin{aligned} O(n) &:= \{Q \in \mathbb{R}^{n \times n} : Q^*Q = I\} = \{Q \in \mathbb{R}^{n \times n} : Q^T Q = I\} \\ U(n) &:= \{Q \in \mathbb{C}^{n \times n} : Q^*Q = I\} \end{aligned} \quad (5.2.9)$$

5.2.10. Proposition (group structure). *The orthogonal and the unitary groups are that (groups). Also*

$$O(n) = U(n) \cap \mathbb{R}^{n \times n}. \quad (5.2.10)$$

5.2.11. Example (Householder transformation). *A Householder transformation generalises a reflection in \mathbb{R}^2 or \mathbb{R}^3 . Let us start with some geometry in \mathbb{R}^3 .*



Consider a vector $w \in \mathbb{R}^3 \setminus \{0\}$ and its orthogonal plane $\pi = \{w\}^\perp$. We would like to reflect a given vector x about π . For this we project orthogonally x along the line $\text{Span}\{w\}$ as to obtain a vector y which is parallel to w and gives the vectorial distance of x to the plane π . In order to reflect, we now subtract y from x twice: once to touch the plane π (as the vector z) and another time to go to the other side, ending in a vector v . In summary, the algebra is

$$v = z - y = x - y - y = x - 2y. \quad (5.2.11)$$

The vector y is given by

$$y = \frac{w^*x}{\sqrt{w^*w}} \frac{w}{\sqrt{w^*w}} = \frac{ww^*}{w^*w} x. \quad (5.2.12)$$

Summarising we can write v as a linear transformation H acting on x :

$$Hx := v = x - 2y = x - 2 \frac{ww^*}{w^*w} x = \left(I - 2 \frac{ww^*}{w^*w} \right) x. \quad (5.2.13)$$

Since x is a generic element of \mathbb{R}^3 this leads to the reflection in $\pi = \{w\}^\perp$ to be given by

$$I - 2 \frac{ww^*}{w^*w} =: H. \quad (5.2.14)$$

Reviewing the construction, we see that the same argument applies to any vector space \mathbb{R}^n and even \mathbb{C}^n .

5.2.12. Definition of Householder matrix. Let $w \in \mathbb{K}^n$, define the associate *Householder matrix* to be

$$H(w) := \begin{cases} I & \text{if } w = 0 \\ I - 2 \frac{ww^*}{w^*w} & \text{if } w \neq 0. \end{cases} \quad (5.2.15)$$

5.2.13. Proposition (properties of Householder matrices). Let $\mathbf{w} \in \mathbb{K}^n$ then

$$\mathbf{H}(\mathbf{w}) \in \mathbb{K}^{n \times n} \quad (5.2.16)$$

$$\mathbf{H}(\mathbf{w}) = \mathbf{H}(\lambda \mathbf{w}) \quad \forall \lambda \in \mathbb{K}, \quad (5.2.17)$$

$$\mathbf{H}(\mathbf{w})^* = \mathbf{H}(\mathbf{w}), \quad (5.2.18)$$

$$\mathbf{H}(\mathbf{w})^{-1} = \mathbf{H}(\mathbf{w}). \quad (5.2.19)$$

In words, Householder matrices are unitary, self-adjoint and involutive. (A matrix \mathbf{M} is involutive or an involution if and only if $\mathbf{M}\mathbf{M} = \mathbf{I}$.)

Proof Exercise. □

5.2.14. Lemma (reflectivity of Householder matrix). For each pair $\mathbf{u}, \mathbf{v} \in \mathbb{K}^n$ we have

$$\mathbf{H}(\mathbf{u} - \mathbf{v})\mathbf{u} = \mathbf{v}. \quad (5.2.20)$$

Proof Exercise. □

5.2.15. Exercise. Compute explicitly a Householder matrix $\mathbf{H} \in \mathbb{R}^{2 \times 2}$ that transforms the vector $\mathbf{u} := (3/2, 2)$ into the vector $\mathbf{v} := (5/4, 0)$, i.e., $\mathbf{H}\mathbf{u} = \mathbf{v}$. (Check first that such a matrix is possible, by looking at the vectors's lengths.)

5.3. Schur factorisation

5.3.1. Theorem. For any matrix $\mathbf{A} \in \mathbb{K}^{n \times n}$ there exists a triangular matrix $\mathbf{T} \in \mathbb{C}^{n \times n}$ and a unitary matrix $\mathbf{Q} \in \mathbb{C}^{n \times n}$ such that

$$\mathbf{A} = \mathbf{Q}^* \mathbf{T} \mathbf{Q}. \quad (5.3.1)$$

Proof We proceed by induction on n . The base case, $n = 1$ is trivial with $\mathbf{Q} = 1$ and $\mathbf{T} = \mathbf{A}$. We have to show the inductive step: assuming the result holds true for $n - 1$ we will show it holds true for n .

By the fundamental theorem of algebra \mathbf{A} has at least one eigenvalue $\lambda \in \mathbb{C}$ with eigenvector \mathbf{v} which we consider of unit length without loss of generality (replace \mathbf{v} by $\mathbf{v}/|\mathbf{v}|$). Consider now the matrix

$$\mathbf{H} := \mathbf{H}(\mathbf{v} - \mathbf{e}^1). \quad (5.3.2)$$

By Lemma 5.2.14 we know that

$$\mathbf{H}\mathbf{v} = \mathbf{e}^1 \text{ and } \mathbf{H}\mathbf{e}^1 = \mathbf{v} = \mathbf{H}^* \mathbf{e}^1. \quad (5.3.3)$$

Therefore, we may write

$$\mathbf{A}\mathbf{H}^* \mathbf{e}^1 = \mathbf{A}\mathbf{v} = \lambda \mathbf{v} = \lambda \mathbf{H}^* \mathbf{e}^1 = \mathbf{H}^* \lambda \mathbf{e}^1 \quad (5.3.4)$$

thus

$$\mathbf{H}\mathbf{A}\mathbf{H}^* \mathbf{e}^1 = \lambda \mathbf{e}^1. \quad (5.3.5)$$

In other words, this says that

$$\mathbf{H}\mathbf{A}\mathbf{H}^* = \begin{bmatrix} \lambda & \mathbf{b}^\top \\ \mathbf{0} & \hat{\mathbf{A}} \end{bmatrix}, \quad (5.3.6)$$

for some $\mathbf{b} \in \mathbb{C}^{n-1}$ and $\hat{\mathbf{A}} \in \mathbb{C}^{(n-1) \times (n-1)}$. By the inductive hypothesis, we know that there exists $\hat{\mathbf{Q}} \in \mathbf{U}(n-1)$ (the unitary group on \mathbb{C}^{n-1}) and $\hat{\mathbf{Q}} \in \mathcal{U}(\mathbb{K}^{n-1})$ (upper triangular matrices in $\mathbb{C}^{(n-1) \times (n-1)}$) such that

$$\hat{\mathbf{A}} = \hat{\mathbf{Q}}^* \hat{\mathbf{T}} \hat{\mathbf{Q}}. \quad (5.3.7)$$

It follows that

$$\begin{bmatrix} \lambda & \mathbf{b}^\top \\ \mathbf{0} & \hat{\mathbf{A}} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \hat{\mathbf{Q}}^* \end{bmatrix} \begin{bmatrix} \lambda & \mathbf{b}^\top \\ \mathbf{0} & \hat{\mathbf{T}} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \hat{\mathbf{Q}} \end{bmatrix} = \tilde{\mathbf{Q}}^* \mathbf{T} \tilde{\mathbf{Q}} \quad (5.3.8)$$

where

$$\tilde{\mathbf{Q}} := \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \hat{\mathbf{Q}} \end{bmatrix} \text{ and } \mathbf{T} := \begin{bmatrix} \lambda & \mathbf{b}^\top \\ \mathbf{0} & \hat{\mathbf{T}} \end{bmatrix}. \quad (5.3.9)$$

From this and (5.3.6) it follows that

$$\mathbf{A} = \mathbf{H}^* \tilde{\mathbf{Q}}^* \mathbf{T} \tilde{\mathbf{Q}} \mathbf{H}. \quad (5.3.10)$$

The result will now follow provided we take

$$\mathbf{Q} := \tilde{\mathbf{Q}} \mathbf{H}, \quad (5.3.11)$$

and we check that \mathbf{Q} is unitary. But this follows from the group structure of $\mathbf{U}(n)$ and $\mathbf{H}, \tilde{\mathbf{Q}} \in \mathbf{U}(n)$. \square

5.3.2. Exercise. Suppose \mathbf{A} is self-adjoint, show that its Schur factorisation is equal to a diagonalisation, i.e., that \mathbf{T} in (5.3.1) is in fact diagonal.

Hint. A diagonal matrix is one that is simultaneously upper and lower triangular and the adjoint of an upper or lower triangular matrix is, respectively, lower or upper triangular.

5.4. QR factorisation

Let $\mathbf{A} \in \mathbb{K}^{n \times n}$. The Schur factorisation described in §5.3, while useful for developing matrix theory, is not very practical because it requires the solution of n eigenproblems. A partial remedy for that is to find a QR factorisation, which we now explain.

5.4.1. Theorem. Any matrix $\mathbf{A} \in \mathbb{K}^{n \times n}$ admits a QR factorisation, whereby for some $\mathbf{Q} \in \mathbf{U}(n)$ and $\mathbf{R} \in \mathcal{U}(\mathbb{K}^n)$ such that

$$\mathbf{A} = \mathbf{Q} \mathbf{R}. \quad (5.4.1)$$

If $\mathbb{K} = \mathbb{R}$ then \mathbf{Q} can be chosen in $\mathbf{O}(n)$.

Proof Proceed by induction on n . The base case, $n = 1$ is trivial with $\mathbf{Q} = 1$ and $\mathbf{R} = \mathbf{A}$. To prove the inductive step assume the result holds true for matrices in $\mathbb{K}^{(n-1) \times (n-1)}$ and consider the block-form for

$$\mathbf{A} = \begin{bmatrix} a & \mathbf{b}^\top \\ \mathbf{c} & \mathbf{D} \end{bmatrix} \text{ and let } l := \left\| \begin{bmatrix} a \\ \mathbf{c} \end{bmatrix} \right\| = \sqrt{\bar{a}a + \mathbf{c}^* \mathbf{c}}. \quad (5.4.2)$$

The vector $l\mathbf{e}^1$ has also length l , therefore we can form the Householder transformation:

$$\mathbf{H}(\mathbf{a}^1 - l\mathbf{e}^1) =: \mathbf{H}. \quad (5.4.3)$$

Now since $\mathbf{H}\mathbf{a}^1 = l\mathbf{e}^1$ we obtain

$$\mathbf{H}\mathbf{A} = \begin{bmatrix} l & \hat{\mathbf{b}}^\top \\ \mathbf{0} & \hat{\mathbf{A}} \end{bmatrix}, \quad (5.4.4)$$

where $\hat{A} \in \mathbb{K}^{(n-1) \times (n-1)}$ and $\hat{\mathbf{b}} \in \mathbb{K}^{n-1}$ can be explicitly computed by

$$\begin{bmatrix} \hat{\mathbf{b}}^\top \\ \hat{A} \end{bmatrix} := \mathbf{H}[\mathbf{A}]^{2:n} \text{ and } [\mathbf{A}]^{2:n} = \begin{bmatrix} a_1^2 & \dots & a_1^n \\ \vdots & \ddots & \vdots \\ a_n^2 & \dots & a_n^n \end{bmatrix}. \quad (5.4.5)$$

By the inductive hypothesis, we know that there are $\hat{\mathbf{Q}} \in \mathbf{U}(n-1)$ and $\hat{\mathbf{R}} \in \mathcal{U}(\mathbb{K}^{n-1})$ such that

$$\hat{A} = \hat{\mathbf{Q}}^* \hat{\mathbf{R}}. \quad (5.4.6)$$

It follows that

$$\mathbf{H}\mathbf{A} = \begin{bmatrix} l & \hat{\mathbf{b}}^\top \\ \mathbf{0} & \hat{A} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \hat{\mathbf{Q}}^* \end{bmatrix}}_{=: \tilde{\mathbf{Q}}} \underbrace{\begin{bmatrix} l & \hat{\mathbf{b}}^\top \\ \mathbf{0} & \hat{\mathbf{R}} \end{bmatrix}}_{=: \mathbf{R}} = \tilde{\mathbf{Q}}\mathbf{R}. \quad (5.4.7)$$

From the properties of Householder matrices, such as \mathbf{H} , it follows that

$$\mathbf{A} = \mathbf{H}\tilde{\mathbf{Q}}\mathbf{R} = \mathbf{Q}\mathbf{R}, \quad (5.4.8)$$

for $\mathbf{Q} := \mathbf{H}\tilde{\mathbf{Q}}$, which is unitary.

If $\mathbb{K} = \mathbb{R}$ then \mathbf{Q} and \mathbf{R} can be found in $\mathbb{R}^{n \times n}$. This can be proved inductively, based on the fact that both \mathbf{a}^1 and $l\mathbf{e}^1$ are in \mathbb{R}^n (thus making $\mathbf{H} \in \mathbb{R}^{n \times n}$ hence in $\mathbf{O}(n)$). \square

5.4.2. Remark (QR's \mathbf{Q} is not Schur's \mathbf{Q}). Note that in general the \mathbf{Q} in the factorisation is *not the same* as (nor the adjoint of) the one obtained in Schur's factorisation. Indeed, $\mathbf{T}\mathbf{Q}$ is generally not in $\mathcal{U}(\mathbb{K}^n)$.

5.5. Cholesky's factorisation

5.5.1. Definition of positive definite operator or matrix. An operator $\mathcal{A} \in \text{Lin}(V \rightarrow V)$, V Hilbert \mathbb{K} -vector space, or a matrix $\mathbf{A} \in \mathbb{K}^{n \times n}$, $n \in \mathbb{N}$ are called *positive definite* (or *positively definite*, or just *positive*) if there exists $c > 0$ for which

$$\langle x, \mathcal{A}x \rangle \geq c \langle x, x \rangle \quad \forall x \in V, \quad (5.5.1)$$

or, for the matrix case,

$$\mathbf{x}^* \mathbf{A} \mathbf{x} \geq c \mathbf{x}^* \mathbf{x}. \quad (5.5.2)$$

5.5.2. Definition of SPD operator or matrix. An operator $\mathcal{A} \in \text{Lin}(V \rightarrow V)$, V Hilbert \mathbb{K} -vector space, or a matrix $\mathbf{A} \in \mathbb{K}^{n \times n}$, $n \in \mathbb{N}$ are called *SPD* if they are *self-adjoint* and *positively definite*. Sometimes we employ the notation $\text{SPD}(\mathbb{K}^n)$ to indicate the set of all SPD matrices in $\mathbb{K}^{n \times n}$.

5.5.3. Remark ($\text{SPD}(\mathbb{K}^n)$ is not an algebra nor a vector space). There can be no opposite in $\text{SPD}(\mathbb{K}^n)$. Cholesky's algorithm, described in (Scott, 2011, §4.2), provides a fast way of performing and LU factorisation for a self-adjoint matrix.

Exercises and problems on direct factorisation methods

Exercise 5.1 (unitary matrix). A matrix $Q \in \mathbb{C}^{n \times n}$ is called *unitary* if and only if

$$Q^* Q = I. \quad (\text{P5.1.1})$$

Show that an unitary matrix is invertible and that its inverse is also unitary. Deduce from this result to show that the unitary matrices in $\mathbb{C}^{n \times n}$ with the matrix multiplication form a (multiplicative) group.

Exercise 5.2 (Householder matrix). Construct a Householder matrix H that maps the vector $\mathbf{x} = (4, 0, 3)$ onto the vector $\mathbf{y} = (5, 0, 0)$, by checking first that $|\mathbf{x}| = |\mathbf{y}|$ and then designing a unit vector \mathbf{w} such that $H = H(\mathbf{w}) := I - 2 \frac{\mathbf{w} \mathbf{w}^*}{\mathbf{w}^* \mathbf{w}}$.

Exercise 5.3 (Schur factorisation). Let A be a Hermitian matrix, i.e., $A \in \mathbb{C}^{n \times n}$ and

$$A^* = A. \quad (\text{P5.3.1})$$

- By using the Schur factorisation, show that there exists a unitary matrix S such that $S^* A S = D$, where D is a diagonal matrix with *real* coefficients.
- Use the Schur factorisation to show that \mathbb{C}^n has an orthonormal basis of eigenvectors of A .
- Show that A is positive definite if and only if all eigenvalues are positive.
- Show that if A is positive definite, then $\det A > 0$.
- Show that A is positive definite if and only if $A = Q^* Q$ for some invertible matrix Q .

Problem 5.4 (tridiagonal LU). Let $A \in \mathbb{R}^{n \times n}$ be a tridiagonal matrix, that is, a matrix of the form

$$A = \begin{bmatrix} a_1 & c_1 & & & \\ b_2 & a_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & b_n & a_n \end{bmatrix}, \quad (\text{P5.4.1})$$

for given $a_1, \dots, a_n, b_2, \dots, b_n, c_1, \dots, c_{n-1}$ and all the unmarked entries equal zero. Assume that

$$\begin{aligned} |a_1| > |c_1| > 0, \quad |a_n| \geq |b_n| > 0, \\ |a_i| \geq |b_i| + |c_i|, \quad b_i, c_i \neq 0, \text{ for } i = 2, \dots, n-1, \end{aligned} \quad (\text{P5.4.2})$$

Show that A is invertible and has an LU factorisation of the form

$$A = \begin{bmatrix} 1 & & & 0 \\ l_2 & 1 & & \\ & \ddots & \ddots & \\ 0 & & l_n & 1 \end{bmatrix} \begin{bmatrix} r_1 & c_1 & & 0 \\ & r_2 & \ddots & \\ & & \ddots & c_{n-1} \\ 0 & & & r_n \end{bmatrix}, \quad (\text{P5.4.3})$$

where the vectors $\mathbf{l} = (l_2, \dots, l_n) \in \mathbb{R}^{n-1}$ and $\mathbf{r} = (r_1, \dots, r_n) \in \mathbb{R}^n$ can be computed via

$$r_1 = a_1, l_i = b_i / r_{i-1} \text{ and } r_i = a_i - l_i c_{i-1} \text{ for } 2 \leq i \leq n. \quad (\text{P5.4.4})$$

Hint. Work by induction on n .

Exercise 5.5 (unitary matrices conserve matrix 2-norm). Let $\mathbf{Q} \in \mathbb{C}^{n \times n}$ be a unitary matrix, and let $\mathbf{x} \in \mathbb{C}^n$ and $\mathbf{A} \in \mathbb{C}^{n \times p}$. Show that

$$|\mathbf{Q}\mathbf{x}|_2 = |\mathbf{x}|_2 \text{ and } |\mathbf{Q}\mathbf{A}|_2 = |\mathbf{A}|_2. \quad (\text{P5.5.1})$$

Exercise 5.6 (Hermitian matrices). A matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ is Hermitian (also known as self-adjoint) if and only if

$$\mathbf{A}^* = \mathbf{A}. \quad (\text{P5.6.1})$$

A matrix \mathbf{A} is positive definite if and only if

$$\mathbf{x}^* \mathbf{A} \mathbf{x} \geq c \mathbf{x}^* \mathbf{x} = c |\mathbf{x}|^2 \quad \forall \mathbf{x} \in \mathbb{K}^n. \quad (\text{P5.6.2})$$

for some $c > 0$ independent of \mathbf{x} .

A matrix that is both self-adjoint and positive definite is called SPD.

- (a) Using the definition of a self-adjoint matrix, show that the self-adjoint matrix \mathbf{A} has all eigenvalues real.
- (b) Show that an SPD matrix \mathbf{A} has all its eigenvalues positive (and thus real).

Exercise 5.7 (QR factorisation). Compute the QR factorisation of the following matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 3 \\ 2 & -6 & 3 \\ -2 & 3 & -3 \end{bmatrix} \quad (\text{P5.7.1})$$

Show all details of your calculation and verify that what you found is indeed a QR factorisation of \mathbf{A} .

Exercise 5.8 (QR facotrisation). Compute the QR factorization of the following matrix with pen, paper and pocket-calculator

$$\mathbf{A} = \begin{bmatrix} 2 & 5 & 3 \\ 4 & 4 & -3 \\ -4 & 2 & 3 \end{bmatrix}. \quad (\text{P5.8.1})$$

Show all details of your calculations.

Exercise 5.9 (least squares's and uniqueness). Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, with $m > n$, and assume that \mathbf{A} has full rank, that is, $\text{rank } \mathbf{A} = n$. Let $\mathbf{b} \in \mathbb{R}^m$. Consider the following quadratic $f: \mathbb{R}^n \rightarrow \mathbb{R}$ defined by

$$f(\mathbf{x}) = |\mathbf{A}\mathbf{x} - \mathbf{b}|^2, \quad \mathbf{x} \in \mathbb{R}^n, \quad (\text{P5.9.1})$$

where $|\cdot|$ denotes the usual Euclidean (ℓ^2) norm in \mathbb{R}^n . Show that there exists a unique vector $\hat{\mathbf{x}} \in \mathbb{R}^n$ that minimizes the functional f and that this vector $\hat{\mathbf{x}}$ is the unique solution of the *normal equations*

$$\mathbf{A}^T \mathbf{A} \hat{\mathbf{x}} = \mathbf{A}^T \mathbf{b}. \quad (\text{P5.9.2})$$

Exercise 5.10 (lease squares and SVD). Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m \geq n$ be given. Assume that $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^*$ with orthogonal matrices $\mathbf{U} \in \mathbb{R}^{m \times m}$, $\mathbf{V} \in \mathbb{R}^{n \times n}$, and a diagonal matrix

$\Sigma \in \mathbb{R}^{m \times n}$ that has only non-negative diagonal entries σ_i , $i = 1, 2, \dots, n$, ordered in decreasing order, that is,

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_n \\ 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{bmatrix} \quad \text{where } \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0. \quad (\text{P5.10.1})$$

Use the decomposition $A = U \Sigma V^*$ to show that there exists a minimiser $\tilde{x} \in \mathbb{R}^n$ to the cost function $x \mapsto \|Ax - b\|_2$, i.e.,

$$\|A\tilde{x} - b\|_2 = \min_{x \in \mathbb{R}^n} \|Ax - b\|_2, \quad (\text{P5.10.2})$$

and find a formula for this solution. Show that the solution \hat{x} is uniquely determined if and only if $\text{rank } A = n$.

Exercise 5.11 (Cholesky factorisation). Compute the Cholesky factorisation of the following matrix by hand:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 8 \\ 3 & 8 & 14 \end{bmatrix}. \quad (\text{P5.11.1})$$

Do not verify beforehand that A is positive definite (which it is). Use the Cholesky factorisation which you computed to argue that A must be positive definite.

Basic numerical methods for ODE's

We will introduce some basic numerical methods in this chapter. The main object of study is not so much the method per se, as a means of finding approximations to ODE's, but to understand why, how, and to which extent, a certain method works or fails. To do this, we employ a mixture of theory and practice. The theory consists in the so-called *error analysis*, while the practical part consists in implementing the methods on a computer and try to appreciate the results on test cases, this is called *computational benchmarking*.

In fact, while there is a definite answer to what fails, it is not so clear what it means for a method to work. As we shall see, some methods which seem quite well designed to approximate end up failing and the main reason for that is that the numerical error (after neglecting round-off errors) can be split into 2 main parts: *truncation error* (also known as *consistency error*) due to the method's accuracy and the *accumulation error* (also known as *stability error*) due to the method's stability under perturbations. Whereas the truncation error is usually clear from the method's derivation, it is not so clear whether a method is stable or not from first sight (at least for the inexperienced). The starting point for our discussion is *Euler's Methods*: the *forward Euler (FE)* (also known as *explicit Euler*) method and the *backward Euler (BE)* (also known as *implicit Euler*) method.

6.1. Derivation of the forward Euler method (FE)

We are after a numerical approximation of the solution $Y : I \rightarrow \mathbb{K}^d$ of the basic IVP (C.1.6)–(C.1.7). Often we refer (a bit improperly) to the numerical approximation as *numerical solution*. There are many ways of deriving the forward Euler method. We shall give three of them here.

6.1.1. Definition of time partition, timestep, step size. Consider a time partition, i.e., a finite sequence of consecutive adjacent time intervals delimited by points

$$x_0 < x_1 < \cdots < x_N, \text{ for some } N \in \mathbb{N}. \quad (6.1.1)$$

Without loss of generality, we will consider

$$x_0 = 0, \quad (6.1.2)$$

unless otherwise stated.¹

By *n-th timestep* we refer to the interval (x_{n-1}, x_n) , and we define the *step size* of the *n-th timestep* as

$$h_n := x_n - x_{n-1}. \quad (6.1.3)$$

¹We note that any IVP can be “forced” to satisfy (6.1.2) by teaking the field f appropriately. Namely replace $f(x, y)$ by $f(x + x_0, y)$.

The *maximum step size* is defined as

$$h := \max_{n=1, \dots, N} h_n. \quad (6.1.4)$$

We say that a time partition $(x_n)_{n \in \{0, \dots, N\}}$ (or its timestep) is *uniform* if all the timesteps have the same size, i.e.,

$$h_n = h, \forall n = 1, \dots, N. \quad (6.1.5)$$

When we consider uniform timestep, we sometime indicate N by $N(h)$ and bear in mind the relationships $x_n = nh$ and $T = hN(h)$.

6.1.2. The naive “derivation”. The most natural way of producing a numerical solution of (C.1.6)–(C.1.7), is to think that the derivative Y' of the exact solution is the limit of the difference quotient, i.e.,

$$Y'(x) = \lim_{h \rightarrow 0} \frac{Y(x+h) - Y(x)}{h}. \quad (6.1.6)$$

This gives us a way of approximating the derivative at time x_{n-1} with

$$Y'(x_{n-1}) \approx \frac{Y(x_n) - Y(x_{n-1})}{h_n}, \quad (6.1.7)$$

which is reasonable if h_n is small enough. Replacing $Y'(x_n)$ in (C.1.6) by this approximation we get

$$\frac{Y(x_n) - Y(x_{n-1})}{h_n} \approx f(x_{n-1}, Y(x_{n-1})). \quad (6.1.8)$$

A manipulation leads to

$$Y(x_n) \approx Y(x_{n-1}) + h_n f(x_{n-1}, Y(x_{n-1})). \quad (6.1.9)$$

6.1.3. Definition of the (forward) Euler method. Noting that $Y(x_n)$ lies on one side while $Y(x_{n-1})$ on the other side of the above relation, prompts us to consider the recursive sequence given by the recursion

$$\mathbf{y}^n := \mathbf{y}^{n-1} + h_n \mathbf{f}(x_{n-1}, \mathbf{y}^{n-1}), \quad (6.1.10)$$

along side the seed

$$\mathbf{y}^0 := \mathbf{y}_0, \quad (6.1.11)$$

in the hope that this will give us an acceptable approximation of Y at the partition points x_0, x_1, \dots, x_N . This recursive definition is known as the *(forward) Euler method* (or scheme), which Euler employed to show that the solution of (C.1.6)–(C.1.7) exists. A drawback with this derivation is that it is hard to generalise in order to obtain “better” approximations. One way to get free of this problem is to think in terms of Taylor’s formula.

6.1.4. A Taylor's formula approach. Suppose the solution Y of (C.1.6)–(C.1.7) is twice differentiable and that the second derivative is “well-behaved”. By Taylor's Sum Theorem (of order 2, see B.3.8 in Appendix ??) we have

$$Y(x_n) = Y(x_{n-1}) + h_n Y'(x_{n-1}) + \mathcal{T}_n, \quad (6.1.12)$$

where \mathcal{T}_n is the remainder term.

If $Y = Y$ is real valued, then $\mathcal{T}_n = Y''(\theta_n)h_n^2/2$ for an appropriate $\theta_n \in (x_{n-1}, x_n)$. If Y is *not* real valued, i.e., $Y \in \mathbb{R}^d$ with $d \geq 2$ this is not necessarily true and one has to use the remainder in integral form, as in App. ?? (B.3.29),

$$\mathcal{T}_n = h_n^2 \int_0^1 Y''(x_n + s h_n)(1-s) ds, \quad (6.1.13)$$

which yields the inequality

$$|\mathcal{T}_n| \leq \frac{h_n^2}{2} \|Y''\|_{L_\infty(x_{n-1}, x_n)} \quad (6.1.14)$$

The scheme given by (6.1.10) and (6.1.11) can be now obtained by “ignoring” the term \mathcal{T}_n at each timestep and using $Y' = f(Y)$; i.e., take the approximation

$$Y(x_n) \approx Y(x_{n-1}) + h_n f(x_{n-1}, Y(x_{n-1})). \quad (6.1.15)$$

There are two advantages of this approach over the one in §6.1.2.

1. This method can be generalised easily. For example, one could consider a Taylor expansion of order 3 instead of (6.1.12), i.e.,

$$Y(x_n) = Y(x_{n-1}) + h_n Y'(x_{n-1}) + \frac{h_n^2}{2} Y''(x_{n-1}) + \tilde{\mathcal{T}}_n. \quad (6.1.16)$$

In the scalar case, we have

$$\tilde{\mathcal{T}}_n = \frac{h_n^3}{6} Y'''(\theta_n) \quad (6.1.17)$$

for an appropriate $\theta_n \in (x_{n-1}, x_n)$. In the vector case, using (B.3.34) and a bound on the integral we have

$$\tilde{\mathcal{T}}_n \leq \frac{h_n^3}{6} \|Y'''\|_{L_\infty(x_{n-1}, x_n)}. \quad (6.1.18)$$

Suppose (to make it a bit simpler) that f does not depend on the time, then by differentiating the ODE relation for Y we obtain

$$Y''(x) = \frac{d}{dx} Y'(x) = \frac{d}{dx} f(Y(x)) = D f(Y(x)) Y'(x) = D f(Y(x)) f(x). \quad (6.1.19)$$

Thus we get the following as an inspiration for an approximate scheme

$$Y(x_n) \approx Y(x_{n-1}) + h_n f(Y(x_{n-1})) + \frac{h_n^2}{6} D f(Y(x_{n-1})) f(x_{n-1}). \quad (6.1.20)$$

This recursive scheme, though more complicated than (6.1.10), should be more accurate, because the part that we are “truncating and throwing” away is $\tilde{\mathcal{T}}_n$ which has one more factor h_n in it than \mathcal{T}_n . So if derivatives of Y all the way to order 3 are bounded above by some constant, then h_n is very small then $\tilde{\mathcal{T}}_n = O(h_n^3)$ is much smaller than $\mathcal{T}_n = O(h_n^2)$ (see §??). This type of reasoning leads to the so-called *Taylor formula methods* of which Runge-Kutta methods constitute an important subfamily. We shall study the Runge-Kutta methods later in this course.

2. The second advantage of the Taylor formula approach, with respect to the naive approach, is that it quantifies the part that we sacrifice in our approximation. Namely, we have a clear description of \mathcal{T}_n . This will prove crucial in the error analysis of the numerical scheme.

6.1.5. A quadrature rule approach. As noted in (C.2.14), the IVP can be written in an integral formulation as

$$Y(x_n) = Y(x_{n-1}) + \int_{x_{n-1}}^{x_n} f(x, Y(x)) dx. \quad (6.1.21)$$

We can now exploit this formulation by trying to replace the integral by a more “computable” object. In fact, any textbook on basic Numerical Analysis Atkinson, 1989; Quarteroni, Sacco and Saleri, 2007, e.g., usually has a whole section on how to approximate integrals and related error analysis. These formulas that approximate integrals are known as *quadrature rules*. The simplest such rule is the left-point Riemann-sum rule whereby

$$\int_a^b g(x) dx \approx g(a)(b-a). \quad (6.1.22)$$

The expression on the right-hand side defines the quadrature rule. It is not very hard to derive an error bound for this quadrature rule. Indeed, an integration by parts yields

$$\begin{aligned} \int_a^b g(x) dx &= - \int_a^b g(x) \frac{d}{dx}(b-x) dx = -[g(x)(b-x)]_{x=a}^{x=b} + \int_a^b g'(x)(b-x) dx \\ &= g(a)(b-a) + \int_a^b g'(x)(b-x) dx. \end{aligned} \quad (6.1.23)$$

In the real-valued case where $g(x) \in \mathbb{R}$, by the integral-form of the mean value theorem B.3.7, assuming g' is continuous, we infer that

$$\int_a^b g'(x)(b-x) dx = g'(\theta) \int_a^b (b-x) dx = g'(\theta) \frac{(b-a)^2}{2}. \quad (6.1.24)$$

Using this fact with $g(x) = Y'(x)$, $a = x_{n-1}$ and $b = x_n$ we get

$$\int_{x_{n-1}}^{x_n} f(x, Y(x)) dx = f(x_{n-1}, Y(x_{n-1}))h_n + Y''(\theta_n) \frac{h_n^2}{2}, \quad (6.1.25)$$

for some $\theta_n \in (x_{n-1}, x_n)$. In the vector valued case we can write

$$\left| \int_a^b g'(x)(b-x) dx \right| \leq \frac{(b-a)^2}{2} \|g'\|_{L_\infty(a,b)}, \quad (6.1.26)$$

which leads, for $g = Y'$, to

$$\left| \int_{x_{n-1}}^{x_n} Y''(x)(x_n - x) dx \right| \leq \frac{h_n^2}{2} \|Y''\|_{L_\infty(x_{n-1}, x_n)}. \quad (6.1.27)$$

By replacing the left-hand side of (6.1.25) by its right-hand side in (6.1.21) we get

$$\mathbf{Y}(x_n) = \mathbf{Y}(x_{n-1}) + \mathbf{f}(x_{n-1}, \mathbf{Y}(x_{n-1}))h_n + \mathcal{T}_n, \quad (6.1.28)$$

where

$$\mathcal{T}_n = \int_{x_{n-1}}^{x_n} \mathbf{Y}''(x)(x_n - x) dx \quad (6.1.29)$$

in which we recognise (6.1.12). The discussion that follows, hence is the same. At this level of simplicity, the Taylor's formula and the quadrature approaches are pretty much two sides of the same coin. This analogy can be found in many different schemes and it is useful to switch from one to another as it suits the purpose. We will see however, that the quadrature approach can be generalised to obtain *linear multistep methods*, which are hard (if not outright impossible) to view as Taylor's methods.

6.2. Error analysis (of the forward Euler method)

The term *error analysis* means a rigorous mathematical theory that provides a bound for the error of a numerical method, in terms of the given data and the approximation parameters. In our case, the exact mathematical object is the solution \mathbf{Y} of the basic IVP (C.1.6)–(C.1.7) and the approximate solution $(\mathbf{y}^n)_{n \in \{0, \dots, N\}}$ defined by the Forward Euler recursive scheme (6.1.10)–(6.1.11).

The *error* usually means the difference (or the distance) between the exact solution and its approximation. Because the exact solution here is a function, while the approximate solution is a sequence, we need to reconcile the two. This can be done in two ways: (a) extend the sequence $(\mathbf{y}^n)_{n \in \{0, \dots, N\}}$ to a function on the time interval $[0, T]$, or (b) restrict the function \mathbf{Y} to the sequence of its values at $(x_n)_{n \in \{0, \dots, N\}}$, i.e., $(\mathbf{Y}(x_n))_{n \in \{0, \dots, N\}}$. While idea (a) can be used successfully, we will choose (b) for the entire course.

6.2.1. Definition of error. We call *error (sequence)* of the method the \mathbb{K}^d -valued sequence given by

$$\mathbf{e}^n := \mathbf{Y}(x_n) - \mathbf{y}^n, \text{ for } n = 0, \dots, N. \quad (6.2.1)$$

The (*absolute pointwise*) *error* is defined as the sequence of non-negative real numbers

$$\varepsilon^n := |\mathbf{e}^n|,$$

where $|\cdot|$ is, unless otherwise stated, the Euclidean norm on \mathbb{K}^d . The *maximum (or uniform) error* is defined as

$$\varepsilon := \max_{n=1, \dots, N} |\mathbf{e}^n|. \quad (6.2.2)$$

6.2.2. Definition of error bound. The main goal in this section consists in getting an *upper bound* on $\varepsilon^n = \mathbf{e}^n$, in terms of the IVP data, i.e., \mathbf{f} and \mathbf{y}_0 , and the *discretization parameter*, i.e., h , as defined in 6.1.1. Synthetically, this means that we look for a function of the meshsize, $h \rightarrow \mathcal{E}_{\mathbf{f}, \mathbf{y}_0, T}(h)$, that satisfies

$$\max_n \varepsilon^n = \varepsilon \leq \mathcal{E}_{\mathbf{f}, \mathbf{y}_0, T}(h) \quad \forall h \in (0, h_0), \quad (6.2.3)$$

for a certain $h_0 > 0$.

If for each \mathbf{f} in a fixed class of functions which, unless otherwise stated, is taken to be

$$\mathcal{F}(\mathbb{R} \times \mathbb{K}^d) := \mathcal{A}(\mathbb{R} \times \mathbb{K}^d; \mathbb{K}^d) \cap \text{Lip}(\mathbb{K}^d; \mathbb{K}^d) \quad (6.2.4)$$

of functions that are analytic (i.e., they have infinitely many derivatives and their Taylor series about any point has a positive convergence radius) in both variables and globally Lipschitz in the second variable.

A function $\mathcal{E}_{\mathbf{f}, \mathbf{y}_0, T}$ satisfying (6.2.3), is called an *error bound* for the particular choice of \mathbf{f} and \mathbf{y}_0 .

We say that an *error bound* is available for a given method on a certain class of functions $\mathcal{G}(\Omega)$ (which is $\mathcal{F}(\mathbb{R} \times \mathbb{K}^d)$ if unspecified), if for each $\mathbf{f} \in \mathcal{G}$ and each $\mathbf{y}_0 \in \Omega$.

Usually, when there is no risk of confusion, the subscripts $\mathbf{f}, \mathbf{y}_0, T$ are dropped and we write

$$\mathcal{E}(h) := \mathcal{E}_{\mathbf{f}, \mathbf{y}_0}(h). \quad (6.2.5)$$

6.2.3. Definition of convergent method. We say that a numerical method to solve the IVP (C.1.6)– (C.1.7) is *convergent* if we have

$$\max_{n=0, \dots, N} |Y(x_n) - \mathbf{y}^n| \leq \mathcal{E}(h), \quad (6.2.6)$$

for all $h < h_0$ and $\mathbf{f} \in \mathcal{F}$, $\mathbf{y}_0 \in \mathbb{K}^d$, $T > 0$, and

$$\mathcal{E}(h) \rightarrow 0, \text{ as } h \rightarrow 0. \quad (6.2.7)$$

6.2.4. Definition of convergence order. In most cases, the error bound is of the form

$$\mathcal{E}(h) = C[\mathbf{f}, \mathbf{y}_0, T] h^q, \quad (6.2.8)$$

for some integer $q \geq 1$ and a number $C[\mathbf{f}, \mathbf{y}_0, T] \geq 0$ which is constant with respect to h . Let

$$p := \max \{p \in \mathbb{N} : q \text{ satisfies (6.2.8) for all } \mathbf{f} \in \mathcal{F}\}, \quad (6.2.9)$$

we say that the method has *convergence order* p .

6.2.5. A simple error analysis of the forward Euler method.

Simplifying assumptions. In order to get a clearer analysis, we simplify the situation by assuming that:

- (i) The unknown Y is \mathbb{K} -valued.
- (ii) The stepsize is uniform, i.e., $h_n = h$, for all $n = 1, \dots, N$.

Recall also the blanket assumption on f , whereby the function f belongs to the standard assumption space \mathcal{F} , i.e., f is analytic and Lipschitz continuous with respect to its second argument, i.e., there exists $\Lambda \geq 0$ such that

$$|f(x, y) - f(x, z)| \leq \Lambda(|y - z|). \quad (6.2.10)$$

Bounding the truncation error. Under these assumptions, we know that the exact solution Y can be expanded using Taylor's formula about x_{n-1} as follows

$$Y(x_n) = Y(x_{n-1}) + Y'(x_{n-1})h + Y''(\eta_n)\frac{h^2}{2}, \quad (6.2.11)$$

for some $\eta_n \in (x_{n-1}, x_n)$. Thus, since Y solves the ODE (??), we may rewrite the above as

$$Y(x_n) = Y(x_{n-1}) + hf(x_{n-1}, Y(x_{n-1})) + \tau_n, \quad (6.2.12)$$

where we have introduced the *truncation error* as

$$\tau_n := Y''(\eta_n)\frac{h^2}{2}. \quad (6.2.13)$$

The right-hand side can be bounded by noting first that

$$\begin{aligned} Y''(\eta_n) &= \left[\frac{d}{dx} Y'(x) \right]_{x=\eta_n} = \left[\frac{d}{dx} f(x, Y(x)) \right]_{x=\eta_n} \\ &= \partial_1 f(\eta_n, Y(\eta_n)) + \partial_2 f(\eta_n, Y(\eta_n))Y'(\eta_n). \\ &= \partial_1 f(\eta_n, Y(\eta_n)) + \partial_2 f(\eta_n, Y(\eta_n))f(\eta_n, Y(\eta_n)). \end{aligned} \quad (6.2.14)$$

Because f is analytic, it follows that $\partial_1 f$ and $\partial_2 f$ and thus $\partial_1 f + \partial_2 f f$ is also an analytic function, so it is also continuous. But also Y is continuous, thus the composed function

$$x \mapsto \partial_1 f(x, Y(x)) + \partial_2 f(x, Y(x))f(x, Y(x)) \quad (6.2.15)$$

is continuous. It follows, in view of Weierstrass Compactness Theorem (see App. ??§ B.4.9), that there exists $C_2 = C_2[f] > 0$ such that

$$|[\partial_1 f + \partial_2 f f](x, Y(x))| \leq C_2, \quad \forall x \in [0, T]. \quad (6.2.16)$$

Given that $\eta_n \in (x_{n-1}, x_n) \subseteq [0, T]$, we obtain the following bound on the truncation error

$$|\tau_n| \leq \frac{C_2}{2} h_n^2, \quad \forall n \in 1 : N. \quad (6.2.17)$$

Accounting for the . On the discrete side, the scheme (6.1.10), implies that

$$y^n = y^{n-1} + h_n f(x_{n-1}, y^{n-1}), \quad (6.2.18)$$

and subtraction leads to

$$e_n (= Y(x_n) - y^n) = e_{n-1} + h_n (f(x_{n-1}, Y(x_{n-1})) - f(x_{n-1}, y^{n-1})) + \tau_n, \quad (6.2.19)$$

which implies that

$$|e_n| \leq |e_{n-1}| + \Lambda h_n |e_{n-1}| + |\tau_n| = (1 + \Lambda h_n) |e_{n-1}| + |\tau_n|. \quad (6.2.20)$$

We have obtained a recursive inequality for $\varepsilon_n = |e_n|$ as $n = 1, \dots, N$, which can be solved as follows:

$$\begin{aligned} \varepsilon_n &\leq (1 + \Lambda h_n) \varepsilon_{n-1} + \tau_n \\ &\leq (1 + \Lambda h_n)(1 + \Lambda h_{n-1}) \varepsilon_{n-2} + (1 + \Lambda h_n) \tau_{n-1} + \tau_n \\ &\leq (1 + \Lambda h_n)(1 + \Lambda h_{n-1})(1 + \Lambda h_{n-2}) \varepsilon_{n-3} \\ &\quad + (1 + \Lambda h_n)(1 + \Lambda h_{n-1}) \tau_{n-2} + (1 + \Lambda h_n) \tau_{n-1} + \tau_n \\ &\dots \\ &\leq \underbrace{(1 + \Lambda h_n)^n}_{=0} \varepsilon_0 + \sum_{m=1}^n \prod_{k=m+1}^n (1 + \Lambda h_k) \tau_m, \end{aligned} \quad (6.2.21)$$

where

$$\begin{aligned} \prod_{k=m+1}^n (1 + \Lambda h_k) &:= (1 + \Lambda h_{m+1}) \prod_{k=m+2}^n (1 + \Lambda h_k) \\ &= (1 + \Lambda h_{m+1}) \cdots (1 + \Lambda h_n), \text{ for } m < n, \end{aligned} \quad (6.2.22)$$

and with the usual convention that the empty product (when $m = n$) is 1, i.e.,

$$\prod_{k=n+1}^n (1 + \Lambda h_k) := 1. \quad (6.2.23)$$

Using the basic relation

$$1 + \xi \leq \exp \xi \quad \forall \xi \in \mathbb{R}, \quad (6.2.24)$$

we may bound

$$1 + \Lambda h_k \leq \exp(\Lambda h_k) \quad \forall k = 1, \dots, n, \quad (6.2.25)$$

to obtain

$$\prod_{k=m+1}^n (1 + \Lambda h_k) \leq \prod_{k=m+1}^n \exp(\Lambda h_k) = \exp(\Lambda(x_n - x_m)), \quad (6.2.26)$$

for all $m = 1, \dots, n$, and thus

$$\varepsilon_n \leq \sum_{m=1}^n \exp(\Lambda(x_n - x_m)) \tau_m. \quad (6.2.27)$$

Closing the estimate. To close this estimate, let us exploit our knowledge on τ_m for $m = 1, \dots, n$, given in (6.2.17). It follows that

$$\varepsilon_n \leq \frac{C_2}{2} \sum_{m=1}^n \exp(\Lambda(x_n - x_m)) h_n^2 \leq \frac{C_2 h}{2} \sum_{m=1}^n \exp(\Lambda(x_n - x_m)) h_n. \quad (6.2.28)$$

To simplify, we observe, that the summation on the right-hand side is the lower Riemann sum approximating from below the integral

$$\int_0^{x_n} \exp(\Lambda(x_n - x)) dx, \quad (6.2.29)$$

which is readily computed as

$$\frac{1}{\Lambda} [\exp(\Lambda(x_n - x))]_{x=x_n}^{x=0} = \frac{\exp(\Lambda x_n) - 1}{\Lambda}. \quad (6.2.30)$$

Hence, the bound simplifies to

$$\varepsilon_n \leq C_1[x_n]h, \quad (6.2.31)$$

where

$$C_1[x_n] = C_1[f, x_n] := \frac{C_2[f](\exp(\Lambda x_n) - 1)}{2\Lambda}. \quad (6.2.32)$$

CONCLUSION (convergence of the forward Euler method). *The Forward Euler method (6.1.10) is convergent, in the sense that for $h \rightarrow 0$, we have that the error $\varepsilon_n \rightarrow 0$, for all n .*

Furthermore the method is (at least) first order, in the sense that $\varepsilon_n = O(h)$.

6.2.6. Exercise (FE is exactly of first order). *Using the model problem $\dot{y} = \lambda y$, $y \in \mathbb{R}$, show that the Forward Euler method is exactly of order 1, in the sense that there exists a constant $c > 0$, such that*

$$\varepsilon_N \geq c h. \quad (6.2.33)$$

Hint. Compute an explicit formula for the exact solution and one for the discrete solution and hence for the error.

6.2.7. Exercise (vector case is not too hard). *Adapt the analysis in §6.2.5 to the vector case where Y takes values in \mathbb{K}^d , by using the vector-valued Taylor's formula (B.3.29) App. ?? to*

$$\text{replace } Y''(\eta_n) \frac{h_n^2}{2} \text{ with } h_n^2 \int_0^1 Y''(x_{n-1} + \theta h_n)(1 - \theta) d\theta. \quad (6.2.34)$$

6.2.8. The magnification factor and stability of the forward Euler method. From §6.2.5 it looks like the forward Euler method has acceptable convergence properties. Indeed from the error bound (6.2.31), we infer that for small enough h , the error $\max_n \varepsilon_n$ can be made smaller than a given *tolerance* ρ by taking $h = \rho / C_2$. The problem with this argument is that the constant C_2 , after a closer inspection turns out to be quite large. Indeed, the exponential part of C_2 makes this constant huge and this forces us to take a very small h . To fix ideas, suppose $\Lambda = 3$, $T = 5$, which are perfectly reasonable choices in practice, compute C_2 and obtain a condition on h for which the error $\varepsilon_n \leq 0.01$ for all n .

So the analysis given in §6.2.5, though interesting from the theoretical view-point, turns out to be disappointing in practical cases because the error bound that is obtained seems to be too pessimistic. Can we maybe improve this?

To answer this question, we need to understand, from the analysis, what is making the constant C_2 this big. After some inspection, we realise that the crucial step was in using the Lipschitz continuity of the function f which led to the appearance of the term $(1 + h\Lambda) > 1$, which was then raised to a successive power at each timestep. This factor amplifies the error accumulated from previous time-steps at each time-step and ended up appearing in the exponential. So, if there is any hope to improve the situation we should at least be able to control this term better, say, by replacing it with something *smaller* than 1. Let us try and see what we can do for the simplest possible nontrivial situation, i.e., the linear model problem

$$\dot{y} = \lambda y, \text{ and } y(0) = y_0, \quad (6.2.35)$$

for some $\lambda \in \mathbb{C}$. As we know this problem has for solution the function $Y(x) = \exp(\lambda x)y_0$ which converges to 0 as $t \rightarrow \infty$, when $\operatorname{re} \lambda < 0$. Let us see how the error of the FE method gets amplified in passing from x_{n-1} to x_n . Working like in §6.2.5, albeit without absolute values, we have

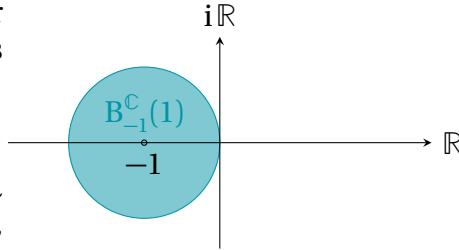
$$\begin{aligned} e_n &= Y(x_{n-1}) + h\lambda Y(x_{n-1}) + \tau_n - y^{n-1} - h\lambda y^{n-1} \\ &= (1 + h\lambda)e_{n-1} + \tau_n. \end{aligned} \quad (6.2.36)$$

Thus $1 + h\lambda$ quantifies the amplification of the accumulated error and it is thus called the *magnification factor*.

For the error *not to be amplified*, the magnification factor must be kept under or equal to 1 in absolute value. This boils down to requiring

$$|1 + h\lambda| \leq 1, \quad (6.2.37)$$

which means that the complex number $h\lambda$ lies in the disc of radius 1 and center -1 , denoted by $B_{-1}^{\mathbb{C}}(1)$.



6.2.9. An alternative interpretation error analysis. We now give an interpretation of the convergence result obtained at the end of §6.2.5. The reason for doing this second, apparently more complicated, analysis is that it has a more general reach which can be applied to methods which are more sophisticated than forward Euler and turns out to be useful in the long run. If you are interested only in skimming these notes, you may skip this paragraph on first reading.

The discrete auxilliary solution. We start by introducing a “splitting” of our error. Indeed, neglecting round-off errors caused by machine (im)precision, the error e_n at time x_n is the sum of two errors:

- (1) the error e_{n-1} , accumulated all the way to time x_{n+1} and amplified (or damped) by a certain factor, and
- (2) the error committed by replacing the solution of the ODE by that of the FE scheme (6.1.10).

In symbols we may write

$$e_n = Y(x_n) - \tilde{y}^n + \tilde{y}^n - y^n, \quad (6.2.38)$$

where $\tilde{\mathbf{y}}^n$ is defined as the solution of one step of scheme (6.1.10) with \mathbf{y}^{n-1} replaced by $\mathbf{Y}(x_{n-1})$, i.e.,

$$\tilde{\mathbf{y}}^n := \mathbf{Y}(x_{n-1}) + h_n \mathbf{f}(x_{n-1}, \mathbf{Y}(x_{n-1})). \quad (6.2.39)$$

Truncation. The truncation error is now expressed as

$$\tau_n = |\mathbf{Y}(x_n) - \tilde{\mathbf{y}}^n|, \quad (6.2.40)$$

which, following 6.2.5, is seen to satisfy

$$\tau_n \leq h_n^2 \int_0^1 |\mathbf{Y}''(x_{n-1} + \theta h_n)| (1 - \theta) d\theta \leq \frac{C_2}{2} h_n^2, \quad (6.2.41)$$

where C_2 may depend on \mathbf{f} and \mathbf{Y} , but not on h .

Accumulation. As for the accumulation error we have

$$\mathbf{a}_n := \tilde{\mathbf{y}}^n - \mathbf{y}^n = \mathbf{Y}(x_{n-1}) - \mathbf{y}^{n-1} + h_n (\mathbf{f}(x_{n-1}, \mathbf{Y}(x_{n-1})) - \mathbf{f}(x_{n-1}, \mathbf{y}^{n-1})), \quad (6.2.42)$$

which implies, in view of the Lipschitz condition on \mathbf{f} , that

$$\alpha_n := |\mathbf{a}_n| \leq (1 + \Lambda h_n) \varepsilon_{n-1}. \quad (6.2.43)$$

A discrete Gronwall argument. We now have

$$\varepsilon_n \leq |\tau_n| + (1 + \Lambda h_n) \varepsilon_{n-1}, \quad (6.2.44)$$

which is the same “one-step bound” we had in the previous analysis (6.2.20). So we may close the estimate using the same argument as before. However, we will take advantage of this situation to introduce a (more general) variant of that argument.

LEMMA (discrete Gronwall’s inequality). *Suppose $(\phi_n)_{n \in \mathcal{N}}$, $(\eta_n)_{n \in \mathcal{N}}$, $(\chi_n)_{n \in \mathcal{N}}$ are sequences of real numbers with $\mathcal{N} = 1 : N$ for some $N \in \mathbb{N}$ such that*

$$\phi_n, \eta_n, \chi_n \geq 0 \text{ and } \chi_n < 1, \forall n \in \mathcal{N} \quad (6.2.45)$$

satisfying the recursive inequality

$$\phi_n \leq \eta_n + \sum_{k=1}^n \chi_k \phi_k, \forall n \in \mathcal{N}. \quad (6.2.46)$$

for each $\delta > 0$ there exists $\bar{\chi}_\delta \in (0, 1)$ such that if $\chi_n \leq \bar{\chi}_\delta$, and $X_0 = 0$ and $X_n = \sum_{k=1}^n \chi_k$, for all $n \in \mathcal{N}$, then (P6.5.3) implies

$$\phi_n \leq \eta_n + \sum_{k=1}^n \chi_k \eta_k \exp((1 + \delta)(X_n - X_{k-1})). \quad (6.2.47)$$

Proof See Exercise 6.2.12. □

To finish note that (6.2.44) implies

$$\varepsilon_k - \varepsilon_{k-1} \leq |\tau_k| + \Lambda h_k \varepsilon_{k-1} \quad \forall k \in 1 : n \quad (6.2.48)$$

which, upon summation on k and some manipulations recalling $\varepsilon_0 = 0$, yields

$$\varepsilon_n \leq \sum_{k=1}^n |\tau_k| + \sum_{k=1}^{n-1} \Lambda h_{k+1} \varepsilon_k. \quad (6.2.49)$$

Setting $\phi_n := \varepsilon_n$, $\eta_n := \sum_{k=1}^n |\tau_k|$, $\chi_k := \Lambda h_{k+1}$, for $1 \leq k < n$ and $\chi_n := 0$, in Gronwall's Lemma, we conclude that for any $\delta_0 > 0$ fixed, there exists h_0 such that, for all $h < h_0$ we have

$$\varepsilon_n \leq \sum_{k=1}^n |\tau_k| + \sum_{k=1}^{n-1} \Lambda h_{k+1} \left(\sum_{j=1}^k |\tau_j| \right) e^{(1+\delta_0)\Lambda \sum_{j=k}^{n-1} h_{j+1}}. \quad (6.2.50)$$

To simplify this, note that

$$\sum_{j=1}^k |\tau_j| \leq \frac{C_2}{2} \sum_{j=1}^k h_j^2 \leq \frac{C_2}{2} \max_{1 \leq j \leq k} h_j \sum_{j=1}^k h_j \leq \frac{C_2}{2} h x_k, \quad (6.2.51)$$

because $h_k = x_k - x_{k-1}$, $h = \max_{1 \leq k \leq N} h_n$, and $x_0 = 0$. It follows that

$$\varepsilon_n \leq \frac{C_2}{2} \left(x_n + \Lambda \sum_{k=1}^{n-1} h_{k+1} x_k e^{\Lambda_0(x_n - x_k)} \right) h, \quad (6.2.52)$$

where we wrote $\Lambda_0 := (1 + \delta_0)\Lambda$. Finally noting that the summation in brackets is the lower Riemann sum of the integral

$$\int_0^{x_n} x \exp(\Lambda_0(x_n - x)) dx = \frac{e^{\Lambda_0 x_n} - 1 - \Lambda_0 x_n}{\Lambda_0^2} \quad (\text{by parts}). \quad (6.2.53)$$

Further manipulations involving the Taylor's series of \exp and $\lambda < \Lambda_0$ conclude our last step, which is the estimate

$$\varepsilon_n \leq \frac{C_2(e^{\Lambda_0 x_n} - 1)}{2\Lambda_0} h \quad \forall h \in (0, h_0), \quad (6.2.54)$$

where $\Lambda_0 > \Lambda$ and h_0 depends on how big is that gap.

6.2.10. Remark (So what? Your constant is exponential.) Estimate (6.2.31) brings good and bad news. The good news, as you know, is that it implies that, for a given \mathbf{f} , T and \mathbf{y}_0 , the approximate solution converges to the exact solution as $h \rightarrow 0$.

$$|\mathbf{y}_h^n - \mathbf{Y}(x_n)| \leq C h \quad \forall n = 1, \dots, N(h). \quad (6.2.55)$$

The bad news is that the constant C obtained in our proof which is in an *exponentially increasing relation* on Λ and T is *pessimistic*, in that it is for the worse case scenario, e.g., $\dot{y} = \lambda y$ with $\mathbb{R}\lambda > 0$.

Indeed, if $\mathbb{R}\lambda \leq 0$, it can be shown, with a more careful analysis and using the fact that f is dissipative in that case, that this constant in fact is very small (and actually decreases as x_n increases!).

6.2.11. Remark (a finer analysis using Jacobians). Note that finer assumptions on \mathbf{f} , can lead to finer estimates. For example, using the fact that \mathbf{f} is differentiable and the Mean Value Theorem (App. ??§ B.3.7), one may replace $1 + \Lambda h_n$ by $|\mathbf{I} + h_n \mathbf{J}|$, with \mathbf{J} being the *averaged Jacobian matrix* given by

$$\mathbf{J} = \int_0^1 \partial_2 \mathbf{f}(x_{n-1}, (1-\theta)\tilde{\mathbf{y}}^{n-1} + \theta \mathbf{y}^{n-1}) d\theta. \quad (6.2.56)$$

We leave it to the reader, as an exercise to work out the details of such a finer analysis by following the traces of this one.

6.2.12. Exercise. Suppose $(\phi_n)_{n \in \mathcal{N}}, (\eta_n)_{n \in \mathcal{N}}, (\chi_n)_{n \in \mathcal{N}}$ are sequences of real numbers with $\mathcal{N} = 1 : N$ for some $N \in \mathbb{N}$ such that

$$\phi_n, \eta_n, \chi_n \geq 0 \text{ and } \chi_n < 1, \forall n \in \mathcal{N} \quad (6.2.57)$$

satisfying the recursive inequality

$$\phi_n \leq \eta_n + \sum_{k=1}^n \chi_k \phi_k, \forall n \in \mathcal{N}. \quad (6.2.58)$$

(a) Show that

$$\phi_n \leq \eta_n + \sum_{k=1}^n \left(\chi_k \eta_k \prod_{i=k}^n \frac{1}{1 - \chi_i} \right). \quad (6.2.59)$$

Hint. Let $R_0 = 0$ and $R_n = \sum_{k=1}^n \chi_k \phi_k$, for $n \geq 1$, and rewrite the recursion (P6.5.2) as

$$R_n - R_{n-1} - \chi_n R_n \leq \chi_n \eta_n, \forall n \in \mathcal{N}. \quad (6.2.60)$$

Multiply both sides of each of these inequalities by the summing factor $\prod_{k=1}^{n-1} (1 - \chi_k)$, then sum and manipulate as to obtain an upper bound on R_n and replace in (P6.5.2).

(b) Show that for each $\delta > 0$ there exists $\bar{\chi}_\delta \in (0, 1)$ such that if $\chi_n \leq \bar{\chi}_\delta$, and $X_0 = 0$ and $X_n = \sum_{k=1}^n \chi_k$, for all $n \in \mathcal{N}$, then (P6.5.3) implies

$$\phi_n \leq \eta_n + \sum_{k=1}^n \chi_k \eta_k \exp((1 + \delta)(X_n - X_{k-1})). \quad (6.2.61)$$

This inequality looks very similar to the conclusion in the continuous analogue of Gronwall's Lemma, except for the factor $(1 + \delta)$ which seems necessary in this case.

Hint. Start by showing that the inequality $1 - x \geq \exp(-(1 + \delta)x)$ is valid for $x \in (0, \log(1 + \delta)/(1 + \delta))$. You may find it useful to study the sign of the function $(0, 1) \ni x \mapsto g(x) = 1 - x - \exp(-(1 + \delta)x) \in \mathbb{R}$.

6.3. Coding the forward Euler method

6.3.1. An abstract look at FE. Consider the problem

$$\dot{y} = \lambda y, \quad y(0) = y_0. \quad (6.3.1)$$

The FE method described in §6.1.3 with uniform time step h gives

$$y^n = y^{n-1} + h\lambda y^{n-1} = (1 + h\lambda)y^{n-1} \quad \forall n = 1, \dots, N, \quad (6.3.2)$$

and hence (cf. Exercise 6.2)

$$y^n = (1 + h\lambda)^n y^0 \quad \forall n = 1, \dots, N. \quad (6.3.3)$$

However, in general y^n cannot be written explicitly as a simple function of y^0 . Indeed, consider applying the FE method to the problem

$$y' = \sin y \text{ and } y(0) = \pi/4. \quad (6.3.4)$$

The FE method can be written as an abstract explicit single-step scheme

$$\mathbf{y}^n = \mathbf{F}_1[\mathbf{f}](h_n, x_{n-1}, \mathbf{y}^{n-1}), \quad \forall n = 1, \dots, N, \quad (6.3.5)$$

where the *stepping* function \mathbf{F}_1 is defined by

$$\mathbf{F}_1[\mathbf{f}](h, x, \mathbf{y}) := \mathbf{y} + h \mathbf{f}(x, \mathbf{y}). \quad (6.3.6)$$

This is how we implement the method, where the evaluation of F_1 is embedded in each step.

6.3.2. Sample Code for a uniform timestep FE method.

Printout of file NDE/Code/FEstep.m

```
function ynew = FEstep(funk,h,x,y)

ynew = y+h*fval(funk,x,y);
```

Printout of file NDE/Code/uniFE.m

```
function [ally,t] = uniFE(funk,T,N,y_0)
% function ally = uniFE(funk,T,N,y_0)
%
% implements a uniform forward Euler (FE) method approximating y:
% dy/dt = funk(t,y)
%
% INPUT: T: final time, N: number of timesteps, y_0 is initial value of
% size [d,1], funk: handle to function for the ODE's field (RHS)
%
% OUTPUT: ally: solution sequence of N columns of size [d,1], size(ally)
%         = [N,d], t: time partition

h=T/N; %time-step (uniform)

t=[0:h:T]; %time partition

%initial value: BEWARE vectors have first index 1 instead of 0
ally(:,1) = y_0;%

%iterate the Forward Euler recursion
for i=1:N
    ally(:,i+1) = FEstep(funk,h,t(i),ally(:,i));
end
```

6.3.3. Benchmarking the uniform FE code. “Benchmarking” means testing the code against cases where the exact solution is known in order to check that everything (theory and code) works. It is important to compute the so called *experimental order of convergence*.

Printout of file NDE/Code/FEbenchmark.m

```
%%A small driver script for ODE methods (default is uniFE)
close all;% graphics
clear all;% variables

% set the paramters to defaults
interaction = 0;%1, where 0=noninteractive/1=interactive script
graphics = 1;%1, where 0=graphics off/1=graphics on
levels = 9;
level = 1;% first refinement level
method = 'uniFE';% uniFE=uniform step forward Euler
T = 1;% default final time
```

```

        field = 'funkidentity';
        exactfunk = 'funksemigroup';
        initial_y = 1;

lastlevel = level+levels;
level = level-1;
ctr=1;
EOC=[];
maxerr=[];
max_error_old=0;
while(level<lastlevel)
    level = level+1;
    N = 2^level;
    %% compute the numerical solution
    [discrete_y,t_discrete] = feval(method,field,T,N,initial_y);

    if ((graphics ~= 0) & (interaction == 0))
        %% plot results
        figure(1);
        hold on;
        t_exact = t_discrete;
        exact_y = feval(exactfunk,t_exact,initial_y); %exact solution
        error=colnorm(exact_y-discrete_y,2);% colnorm is an in-house
            % sub

        style=strcat(choosecolor(level),'-');
        plot(t_exact,log(error),style,'LineWidth',2);
        printpoint=length(t_exact);
        text(t_exact(printpoint),log(error(printpoint)),sprintf(['h=1/' ...
            '2^{%i}'],level),'color',choosecolor(level));

    end
    max_error = max(error);
    maxerr = [maxerr,max_error];
    if((max_error_old ~=0)&&(max_error ~=0))
        EOC = [EOC,log(max_error_old/max_error)/log(2)];
    end
    max_error_old = max_error;
end
title('errors for various levels');
EOC
hold off;
figure(2);
plot(log(maxerr),'s-','LineWidth',2);
for level=1:length(EOC)
    labelme=sprintf('EOC=%5.4f',EOC(level));
    text(level+0.5,log(maxerr(level)),labelme);
end
title('|log h|->log e');
annotation('textbox',[1,1,.4,.1]);

```

With the following subroutine functions.

Printout of file NDE/Code/funkidentity.m

```

function y = funkidentity(t,y)
%% a minimalist file
%% implementing the identity function

y = y;

```

Printout of file NDE/Code/funksemigroup.m

```
function y = funksemigroup(t,y0)
%% a small file implementing the simplest semigroup (i.e., the
%% exponential which corresponds to $\dot{y} = y$

y = exp(t).*y0;
```

The subrouting that computes the Euclidean norm of each column is

Printout of file NDE/Code/colnorm.m

```
function norms = colnorm(A,p)
% calculates the p vector norm of each column in matrix A and
% returns a row vector of length size(A)[1]
s = size(A);
N = s(2);
for n=1:N
    norms(n) = norm(A(:,n),p);
end
```

6.3.4. The experimental order of convergence. Let us discuss the concept of EOC appearing in the code 6.3.3. One of the most important aspects of numerical analysis is the derivation of order of convergence for various methods. So when developing code based on a numerical analysis, it is important to reconcile the crunched numbers with the theory by recovering the order of convergence from an “experimental” point of view. Where the word “experimental” refers to “computer experiments” or “simulations”.

In the case of methods for ODE’s the order is defined in §6.2.4 where we assume that

$$\varepsilon(h) \left(:= \max_{n=1, \dots, N(h)} \varepsilon_n \right) \leq \mathcal{E}(h) = C[\mathbf{f}, T]h^p, \quad (6.3.7)$$

on a uniform timestep partition of $[0, T]$ of size h and $N(h) + 1$ points. Assuming that the *bound is sharp*, in the sense that

$$(C - \delta(h))\mathcal{E}(h) \leq \varepsilon, \quad (6.3.8)$$

for some function $h \mapsto \delta(h)$ such that $\delta(h) \rightarrow 0$ as $h \rightarrow 0$, it is possible to approximate the value of p quite well from benchmark computations, i.e., numerical experiments where the error can be explicitly computed.

Indeed, by evaluating the error at two different (uniform) timestep sizes, say $h_1 > h_2$, then, denoting the errors by $\varepsilon(h_1)$ and $\varepsilon(h_2)$ we have

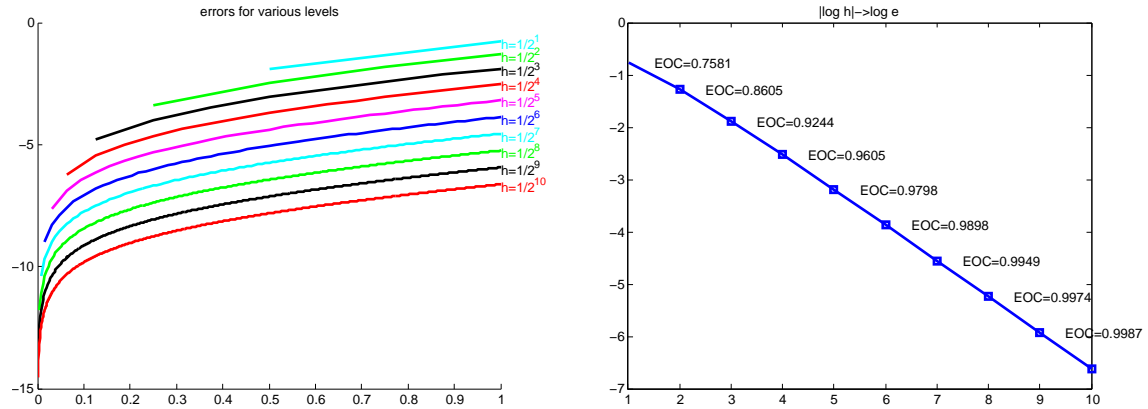
$$\varepsilon(h_i) \approx C h_i^p, \quad i = 1, 2. \quad (6.3.9)$$

Hence

$$\frac{\varepsilon(h_1)}{\varepsilon(h_2)} \approx \left(\frac{h_1}{h_2} \right)^p, \quad (6.3.10)$$

which implies that

$$\log \frac{\varepsilon(h_1)}{\varepsilon(h_2)} \approx p \log \frac{h_1}{h_2} \text{ and } p \approx \frac{\log \varepsilon(h_1) - \log \varepsilon(h_2)}{\log h_1 - \log h_2} =: \text{EOC}(h_1, h_1/h_2). \quad (6.3.11)$$



(a) Error, as a function of time for the forward Euler method applied to problem $\dot{y} = y$ and $y(0) = 1$. At each “refinement level”, i.e., $\log h_i$ we plot the error as a function of time. Because error decreases very fast as the level increases, it is visually clearer to plot the *logarithm* of the error, rather than the error itself.

(b) Experimental order of convergence (EOC) study for the forward Euler method applied to problem $\dot{y} = y$ and $y(0) = 1$. We plot the maximal error versus the timestep on a *loglog scale*. The slope of each segment represents thus the EOC in that portion of the computation.

FIGURE 1. Computer experiments for benchmarking forward Euler, using `FEbenchmark.m`.

In computational experiments we often take $h_1/h_2 = 2$, so we shorten the expression $\text{EOC}(h_1, 2)$ to $\text{EOC}(h_1)$.

So as $h \rightarrow 0$ we should have $\text{EOC}(h) \rightarrow p$, and we say that a computation *enter the asymptotic regime* when $\text{EOC}(h)$ becomes close to p , by an order or 0.05.

Note that $\text{EOC}(h, p)$ is nothing but the slope of the graph of $\log \varepsilon(h)$ as a function of $\log h$. That is why plotting the so-called *loglog graph* is a standard way of visualising the convergence rates and EOC's.

6.3.5. Example (basic benchmark problem for FE). As a first benchmark we run `FEbenchmark.m` in Matlab[®], using `funkidentity.m` which means that we are numerically computing an approximation of $Y(x) = \exp x$. The results are shown and commented in Figure 1.

6.4. The backward (implicit) Euler method

6.4.1. Why not forward (explicit) Euler. As we have seen, it is very easy to program and calculate the solution of the forward Euler method. However, a drawback of the FE method is its poor stability properties shown by the small region of absolute stability introduced in §6.2.8.

To see why this is a problem consider the following Matlab[®] functions

Printout of file `NDE/Code/FEunstable.m`

```

function ally = FEunstable
%%function ally = FE(funk,N,T,y_0)
%
%%implements a funky FE method to find the vector y of approximations
%%to the solution of the IVP dy/dt = funk(t,y) at times
%%[0: T/N :T]={0,T/N,2*T/N,...,(N-1)*T/N,T}
%%
%%the timestep is chosen, not too small as to exhibit unsable behaviour

close all;

funk=@funkminusidentity;
T=17;
y_0=1;

figure(1)
hold on;
time=[0:0.01:T];
plot(time,exp(-time),'k','LineWidth',2);

Legs{1} = 'exact solution';
count=2;

for N=[8:2:14];

    h=T/N; %time-step (uniform)

    %discrete time levels
    t=h*[0:N];

    %initial value: note vectors have first index 1
    y=y_0;
    ally(:,1)=y;

    %iterate the Forward Euler recursion
    for i=1:N
        y=FEstep(funk,h,t(i),y);
        ally(:,i+1)=y;
    end

    plot(t,ally, strcat('-',choosecolor(N/2)), 'LineWidth',1);
    Legs{count} = sprintf('h=%5.5f',h);
    count=count+1;
end

legend(Legs);

```

Printout of file NDE/Code/funkminusidentity.m

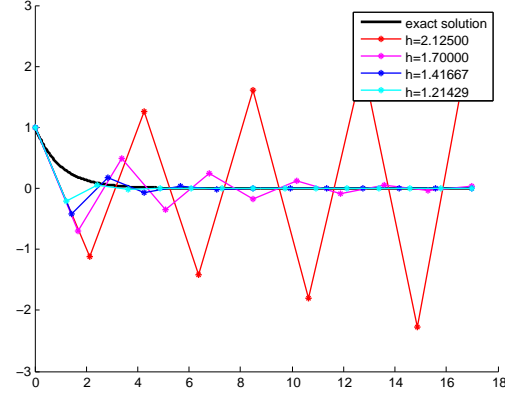
```

function y = funkminusidentity(t,y)
%% a minimalist file
%% implementing the sign change function

y = -y;

```

We plot the output of the code here. We observe that there is a problem if the timestep is not small enough. While the discrete solution is not expected to be close to the exact one for large timesteps, another problem is apparent. The oscillations for large timestep grow uncontrollably and this leads to the crashing of the code. A remedy to this problem is to consider a variant of the FE scheme, which we do in this section.



6.4.2. Backward Euler method's derivation. Based on the same approximation of the Riemann rectangle quadrature rule, as in §6.1.5, but switchin the right-end point, i.e., x_n , we “replace” the integral in

$$Y(x_n) = Y(x_{n-1}) + \int_{x_{n-1}}^{x_n} f(x, Y(x)) dx, \quad (6.4.1)$$

to get

$$Y(x_n) = Y(x_{n-1}) + f(x_n, Y(x_n))h_n. \quad (6.4.2)$$

This leads to the following timestepping scheme, called the *backward Euler (BE)* (also known as the *implicit Euler*) *method*,

$$\mathbf{y}^n = \mathbf{y}^{n-1} + h_n \mathbf{f}(x_n, \mathbf{y}^n), \text{ for } n \in 1 : N, \quad (6.4.3)$$

coupled, like the forward Euler method, with the initial condition

$$\mathbf{y}^0 = \mathbf{y}(0). \quad (6.4.4)$$

6.4.3. Remark (implicit versus explicit). The BE method is called implicit, because the solution at timestep n , is not a straightforward “plugging in \mathbf{y}^{n-1} ” into \mathbf{F}_1 as was the case with FE (cf. (6.3.5)). In fact, a more abstract form of the scheme (6.4.3) is given by

$$\mathbf{F}_0[\mathbf{f}](h_n, x_n, \mathbf{y}^n) = \mathbf{y}^{n-1}, \quad (6.4.5)$$

where the implicit part of the *timestepping* function is

$$\mathbf{F}_0[\mathbf{f}](h, x, \mathbf{y}) := \mathbf{y} - h \mathbf{f}(x, \mathbf{y}). \quad (6.4.6)$$

This means that at the n -th timestep where we need to calculate \mathbf{y}^n (with n , x_n , h_n , \mathbf{y}^{n-1} and \mathbf{f} all fixed) we need to *solve for $\mathbf{y} \in \mathbb{K}^d$ the equation*

$$\mathbf{G}(\mathbf{y}) = \mathbf{b}, \quad (6.4.7)$$

where $\mathbf{b} = \mathbf{y}^{n-1}$ and $\mathbf{G} = \mathbf{F}_0[\mathbf{f}](h_n, x_n, \cdot)$.

Note that in general there is no straightforward method of solving equations like (6.4.7) and it all depends on the nature of the function $\mathbf{G} : \mathbb{K}^d \rightarrow \mathbb{K}^d$. In our case $\mathbf{G} = \mathbf{F}_0[\mathbf{f}](h_n, x_n, \cdot)$ which means that the solvability of the equation depends on $\mathbf{f}(x_n, \cdot)$ and on h_n .

We will address this issue more thoroughly later, when we study the Trapezoidal Method. For now, we will assume the following.

6.4.4. HYPOTHESIS (well-posedness of the BE step). Given \mathbf{f} , x_n , h_n and \mathbf{y}^{n-1} it is possible to solve equation (6.4.3) uniquely for \mathbf{y}^n .

6.4.5. Exercise (BE for linear problems). Suppose \mathbf{f} is linear in its second argument, i.e.,

$$\mathbf{f}(x, \mathbf{y}) := \mathbf{A}(x)\mathbf{y} \quad (6.4.8)$$

where $\mathbf{A}(x) \in \mathbb{K}^{d \times d}$ is a matrix whose coefficients depend on x , and $\mathbf{A} \in C^\infty([0, \infty); \mathbb{K}^{d \times d})$.

(a) Write an expression, in terms of \mathbf{y}^{n-1} , x_n and h_n and $\mathbf{A}(x_n)$ for the n -th value of \mathbf{y}^n of the backward Euler scheme for the IVP

$$\dot{\mathbf{y}} = \mathbf{A}(x)\mathbf{y} \quad (= \mathbf{f}(x, \mathbf{y})) \quad \text{and} \quad \mathbf{y}(0) = \mathbf{y}_0. \quad (6.4.9)$$

(b) Is \mathbf{y}^n always well-defined? If not, find a condition relating $\mathbf{A}(x_n)$ and h_n which allows you to ensure that \mathbf{y}^n is well defined.

Hint. Use Neumann's series Lemma B.2.10.

6.4.6. Stability analysis. The BE beats FE in stability properties. To understand this, let us calculate the BE method's magnification factor, as we did in §6.2.8 for FE. Consider the BE solution sequence $(y^n)_{n=0, \dots, N}$ of the scalar model linear problem

$$\dot{y} = \lambda y \quad \text{and} \quad y(0) = y_0. \quad (6.4.10)$$

That is, for $n \geq 1$, the n -th term of the sequence is defined by

$$y^n = y^{n-1} + h\lambda y^n, \quad (6.4.11)$$

which implies

$$y^n = \frac{1}{1 - h\lambda} y^{n-1}, \quad (6.4.12)$$

when $h \neq 1/\lambda$. (We are considering $h_n = h$ for simplicity.) It follows that

$$|y^n| \leq m(h) |y^{n-1}|, \quad \text{where} \quad m(h) := \frac{1}{|1 - h\lambda|}. \quad (6.4.13)$$

Note that an error analysis leads to the inequalities

$$m(h)(\varepsilon_{n-1} - |\tau_n|) \leq \varepsilon_n \leq m(h)(\varepsilon_{n-1} + |\tau_n|). \quad (6.4.14)$$

[*]: Check!

This follows[*] from the definition of truncation error $\tau_n := Y(x_n) - \tilde{y}^n$, as in §6.2.9, but with the *auxilliary solution* now satisfying

$$\tilde{y}^n = Y(x_{n-1}) + h_n \lambda \tilde{y}^n. \quad (6.4.15)$$

Thus the error from previous timesteps is magnified if $m(h) > 1$, reduced if $m(h) = 1$, and reduced if $m(h) < 1$.

6.4.7. Definition of absolute stable method at $z \in \mathbb{C}$. The previous discussion and the similar one for FE at 6.2.8, prompt the following definition. A numerical method producing is called *absolutely stable at a point* $z \in \mathbb{C}$ if and only if the *infinite* solution sequence $(y^n)_{n \in \mathbb{N}_0}$ of the scheme with uniform timestep h —approximating the exact solution $t \mapsto Y(t) = \exp(zt/h)y_0$ of the IVP (6.4.10) with $\lambda = z/h$ —is bounded with respect to n , i.e., there exists a constant $C = C[\lambda]$ such that

$$|y^n| \leq C \quad \forall n \geq 0. \quad (6.4.16)$$

6.4.8. Definition of stability function. Given a numerical method of order p , we say that $R : D \subseteq \mathbb{C} \rightarrow \mathbb{C}$ is the stability function for this method if for each given $z \in D$, $h > 0$, $\lambda \in \mathbb{C}$ such that $h\lambda = z$ and $(y^n)_{n \in \mathbb{N}_0}$ as in §6.4.7 we may write

$$y^n = R(h\lambda)y^{n-1} + O(h^{p+1}). \quad (6.4.17)$$

This means that up to the approximation order, $R(h\lambda)$ dictates the scheme's behaviour for linear problems. The *magnification factor* is defined as

$$m_\lambda(h) = |R(h\lambda)|. \quad (6.4.18)$$

6.4.9. Definition of region of absolute stability. The *region of absolute stability* of a numerical method is the set of all points $z \in \mathbb{C}$ such that the method is absolutely stable at z .

6.4.10. Proposition (characterisation of RAS via stability function). *If a method possesses a stability function R , then the region of absolute stability \mathcal{S} of this method is given by*

$$\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\} \quad (6.4.19)$$

6.4.11. Exercise (stability function for Euler's method). *Find the stability functions (and their domain), R_{FE} and R_{BE} for the FE and BE methods, respectively. Show, that the FE's region of absolute stability \mathcal{S}_{FE} coincides with the one derived in 6.2.8.*

6.4.12. Problem (error analysis for BE). *Denote by Y the solution of the IVP*

$$\dot{y} = f(x, y) \text{ on } [0, T], \text{ and } y(0) = y_0, \quad (6.4.20)$$

with $f \in \mathcal{F}$ (analytic and globally Lipschitz). Denote by $(y^n)_{n=0, \dots, N}$ the solution of the corresponding backward Euler scheme, with respect to a (possibly nonuniform) time interval partition

$$x_0 := 0 < x_1 < \dots < x_N := T \text{ and } h_n := x_n - x_{n-1} \quad \forall n = 1, \dots, N. \quad (6.4.21)$$

(a) *Define the truncation error of the scheme as*

$$\tau_n := Y(x_n) - \tilde{y}^n, \quad (6.4.22)$$

where \tilde{y}^n is an auxiliary discrete solution given by

$$\tilde{y}^n := Y(x_{n-1}) + h_n f(x_n, Y(x_n)) \quad (6.4.23)$$

and show, using Taylor's expansion or otherwise, that for any Y of class C^2 and bounded, there exists a constant $C = C[f, T, y_0]$ such that

$$|\tau_n| \leq C h^2. \quad (6.4.24)$$

(b) *Argue that since $f \in \mathcal{F}$ then the solution Y is defined on $[0, T]$ and has bounded derivatives therein, proving thus that the bound above is not only formal.*

(c) *Calculate a relation between the error $e_n := Y(x_n) - y^n$ in terms of f , x_n , h_n , e_{n-1} and τ_n .*

(d) *Find an as tight as possible bound on $\varepsilon_n := |e_n|$, in terms of $\Lambda := \text{Lip } f$, x_n , h_n , ε_{n-1} and C .*

- (e) Use a recursion argument, or the Gronwall Lemma, to find an estimate on ϵ_n that ensures convergence as $h \rightarrow 0$.

6.5. Trapezoidal method

6.5.1. Why not backward Euler. We have seen that backward Euler is a much more stable scheme than forward Euler. This seems to be the ideal numerical scheme for numerical approximation of ODE's, except it suffers from two major drawbacks. And namely

- (a) BE is only first order (just like FE), maybe one can find higher order methods, which would be nice because the higher the order the faster the code (assuming overhead costs do not increase dramatically).
- (b) BE is *dissipative*, this means that applying it to equations that have some conservation property, for example, the solution of

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ -u \end{bmatrix} \text{ and } \begin{bmatrix} u(0) \\ v(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (6.5.1)$$

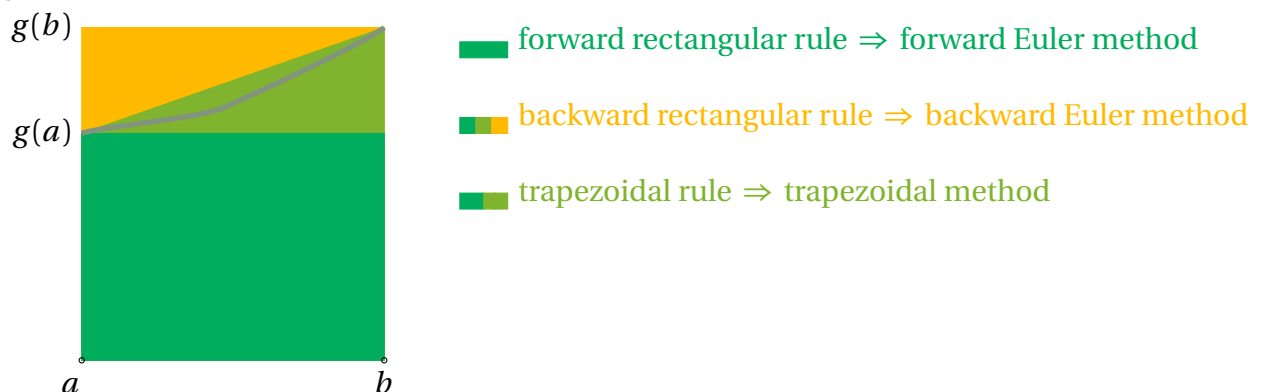
which is readily found to be $U(x) = \cos x$ and $V(x) = \sin x$, satisfies $|U(x), V(x)| = 1$ for all x . In many cases, it is important for the scheme to preserve (or to approximate very well) these properties of the exact solution in the discrete solution. BE does not and is called “dissipative” for this reason.

A numerical method that will improve both these aspects of BE is known to be the Trapezoidal method.

6.5.2. Derivation. To improve the order of BE, we return to the derivation via the integral formulation. The one-step exact solution is given by

$$Y(x_n) = Y(x_{n+1}) + \int_{x_{n+1}}^{x_n} f(x, Y(x)) dx. \quad (6.5.2)$$

Instead of approximating the integral with a rectangular rule (which has, as we have seen in the past sections) order of approximation 2, let us try something “finer”. For example, we can use the trapezoid approximation, as depicted in the following diagram.



Algebraically, the *trapezoidal rule* to approximate the integral of a function g over the interval $[a, b]$ is defined as

$$T_a^b[\mathbf{g}] := \frac{b-a}{2}(\mathbf{g}(a) + \mathbf{g}(b)) \approx \int_a^b \mathbf{g}(x) dx. \quad (6.5.3)$$

This is equivalent to defining

$$T_a^b[\mathbf{g}] = \int_a^b l(x) dx, \text{ where } l(x) := g(b)\frac{x-a}{b-a} + g(a)x - b. \quad (6.5.4)$$

Using this rule to replace the integral in (6.5.2), we obtain

$$\mathbf{Y}(x_n) \approx \mathbf{Y}(x_{n+1}) + \frac{h_n}{2}(\mathbf{f}(x_{n+1}, \mathbf{Y}(x_{n+1})) + \mathbf{f}(x_n, \mathbf{Y}(x_n))). \quad (6.5.5)$$

Replacing $\mathbf{Y}(x_n)$ by \mathbf{y}^n we obtain the scheme

$$\mathbf{y}^0 := \mathbf{y}_0 \text{ and } \mathbf{y}^n := \mathbf{y}^{n+1} + \frac{h_n}{2}(\mathbf{f}(x_{n+1}, \mathbf{y}^{n+1}) + \mathbf{f}(x_n, \mathbf{y}^n)) \text{ for } n = 1, \dots, N. \quad (6.5.6)$$

6.5.3. Exercise (analysis of the trapezoidal method). Consider the numerical approximation, via the trapezoidal method, of the IVP

$$\dot{y} = f(x, y) \text{ on } [0, T] \text{ and } y(0) = y_0, \quad (6.5.7)$$

with $y_0, y \in \mathbb{R}$ and $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ analytic and globally Lipschitz.

Let $0 = x_0 < x_1 < \dots < x_N = T$ be a time partition, with uniform timestep $h := x_n - x_{n-1}$ for all $n \in 1 : N$. The trapezoidal method is given by

$$y_0 = y_0 \text{ and } y^n = y^{n-1} + \frac{h}{2}(f(x_n, y^n) + f(x_{n-1}, y^{n-1})), \text{ for } n \in 1 : N. \quad (6.5.8)$$

(a) Is this an explicit or implicit scheme? Explain in a sentence what is involved in an implementation of this method. State a good feature of the method. State a possible difficulty that one may face in implementing this method, regarding the timestep h , when $f(x, y)$ is not linear in y .

(b) Explain why the exact solution of (P6.10.1), Y , and all its derivatives are bounded on $[0, T]$.

(c) The n -th timestep local truncation error (LTE) is

$$T_n(h) = Y(x_n) - Y(x_{n-1}) - \frac{h}{2}(f(x_{n-1}, Y(x_{n-1})) + f(x_n, Y(x_n))). \quad (6.5.9)$$

Show that there exists $B > 0$ and $p \in \mathbb{N}$, such that

$$|T_n(h)| \leq B h^{p+1}. \quad (6.5.10)$$

Identify the value of p and relate p to the order of the method.

(d) Suppose that $\mu \leq \partial_y f \leq \lambda$, where $\mu, \lambda \in \mathbb{R}$, $\mu \leq \lambda < 0$. Show that for $h_0 = -2/\mu$ we have

$$|Y(x_n) - y^n| \leq \frac{2+h\lambda}{2-h\lambda} |Y(x_{n-1}) - y^{n-1}| + \frac{2}{2-h\lambda} |T_n(h)|, \forall n \in 1 : N, h \in (0, h_0). \quad (6.5.11)$$

(e) Deduce that there exists a constant $C > 0$ such that

$$|Y(x_n) - y^n| \leq C h^p, \forall n \in 1 : N, h \in (0, h_0). \quad (6.5.12)$$

6.5.4. Theorem (convergence of the trapezoidal method). Suppose f is a globally Lipschitz smooth function and let Y be the solution of IVP (C.1.6)–(C.1.7). Then there exists $C = C_{\text{TM}}[f, T, y_0]$ such that

$$|Y(x_n) - y^n| \leq C h^2, \quad (6.5.13)$$

where $(y_n)_{n=0, \dots, N}$ is the solution of the Trapezoidal Method (6.5.6) and $h := \max_{n=1, \dots, N} h_n$.

6.5.5. Solving nonlinear problems (Newton). Note that the relation defining y^n in terms of y^{n+1} is *implicit* in y^n and must be solved at each time step. Namely we have the problem of finding $y^n \in \mathbb{K}^d$ such that

$$\left[\mathbf{I} - \frac{h_n}{2} f(x_{n+1}, \cdot) \right] (y^{n+1}) = y^{n+1} + \frac{h_n}{2} f(x_{n+1}, y^{n+1}) \quad (6.5.14)$$

at each timestep.

If f is linear in y^n , say $f(x_n, y) = A(x_n)y$ where $A(x_n)$ is a matrix whose entries depend possibly on x_n but not on y , this problem involves the inversion of the matrix $\mathbf{I} - A(x_n)h_n/2$.

Because the matrix \mathbf{I} is invertible, its perturbation $\mathbf{I} - Ah_n/2$ has a solution for h_n small enough, by applying the following result (see the appendix for more details).

6.5.6. Lemma (Neumann's series). Given a matrix $M \in \mathbb{K}^{d \times d}$, such that $|M| < 1$ —where $|\cdot|$ is any matrix (vector-induced) norm—then the inverse of $\mathbf{I} - M$ exists and

$$[\mathbf{I} - M]^{-1} = \sum_{k=0}^{\infty} M^k. \quad (6.5.15)$$

If f is nonlinear, as is the case in general, then we cannot invert the problem directly and we have to use a nonlinear method. The methods of choice for such problems are the iterative nonlinear solver given by Newton–Raphson iteration and the fixed-point iteration.

6.5.7. Solving nonlinear problems (fixed-point iteration). The problem with Newton's method is that it involves the derivative of the function f to be evaluated, possibly at each step. This can be quite costly numerically, and what is gained in terms of order is lost in overhead for side computation.

An other iterative method is based on the *fixed point iteration* which is the object of the following exercise.

6.5.8. Problem (a fixed-point iterative solver for the Trapezoidal Method). The n -th step of the trapezoidal scheme the scalar ODE

$$\dot{y}(x) = f(x, y) \quad (6.5.16)$$

is:

$$\begin{aligned} &\text{given } y^0, \dots, y^{n-1} \in \mathbb{R} \\ &\text{find } y^n \in \mathbb{R} \text{ such that} \\ &y^n = y^{n-1} + \frac{h}{2} (f(x_{n-1}, y^{n-1}) + f(x_n, y^n)). \end{aligned} \quad (6.5.17)$$

Assuming that y^{n-1} has been computed, consider, as a way to approximate the y^n in (6.6.10), the following iteratively defined sequence:

$$\begin{aligned} p^0 &:= y^{n-1} + hf(x_{n-1}, y^{n-1}) \\ p^i &:= y^{n-1} + \frac{h}{2} (f(x_{n-1}, y^{n-1}) + f(x_n, p^{i-1})), \text{ for } i \in \mathbb{N}. \end{aligned} \quad (6.5.18)$$

(a) Supposing that the function f satisfies the Lipschitz condition

$$|f(x, y) - f(x, z)| \leq \Lambda |z - y|, \quad (6.5.19)$$

and looking at (6.6.11) as a fixed point iteration, show that $h < 2/\Lambda$ is a sufficient condition for the sequence (p^i) to converge.

Assume condition (6.5.19) holds in the rest of this problem.

(b) Show that $\lim_{i \rightarrow \infty} p^i = y^n$, where y^n is the solution of (6.6.10).

(c) Show that $|y^n - p^i| = O(h^{2+i})$ for $i \in \mathbb{N}_0$.

Hint. Use the fact that $|y^n - \tilde{Y}(x_n)| = O(h^3)$ and $|Y^0 - \tilde{Y}(x_n)| = O(h^2)$ where \tilde{Y} is the exact solution of the ODE (6.6.9) with initial condition $\tilde{Y}(x_{n-1}) = y^{n-1}$.

6.5.9. Trapezoidal method code. We discuss now the actual implementation of the trapezoidal method. This is in fact a modification of the Euler's methods.

We start with the heart of the matter, i.e., the trapezoidal step, implemented as an Octave function `TMstep.m` (which is easily adapted to become a Matlab[®] function if one wishes) and lists as follows:

Printout of file NDE/Code/TMstep.m

```
function [ynew, iteration, residual] = TMstep(funk, h, x, y)
%% [ynew, iteration, residual] = TMstep(funk, h, x, y)
%%
%% simple fixed-point iterative solver for the trapezoidal method
%% approximating Y(x+h) satisfying Y'=funk(.,Y) and Y(x)=y
%% provides one step of Trapezoidal method

maxit=10;
tolerance=h^3;
ynew = y;
residual = 1;
iteration = 0;
while((iteration<=maxit)&&(residual>tolerance))
    iteration = iteration+1;
    ynew = y+h/2*(feval(funk, x+h, ynew)+feval(funk, x, y));
    residual = norm(-ynew+y+h/2*(feval(funk, x+h, ynew)+feval(funk, x, y)), 2) <-
    ;
    % no the above is *not* zero... this is code, not math :-)
end
```

A 1-dimensional use of `TMstep.m` is illustrated by the following driver code

Printout of file NDE/Code/TrapScript.m

```
%%A small driver script for ODE methods (default is uniFE)
close all;% graphics
clear all;% variables
```

```

% set the paramters to defaults
interaction = 0;%1, where 0=noninteractive/1=interactive script
graphics = 1;%1, where 0=graphics off/1=graphics on
levels = 9;
level = 1;% first refinement level
method = 'Trap';% FE=forward Euler, Trap=trapezoidal (iterative)
T = 1;% default final time
field = 'funkharmonic';
initial_y = 1;
% and conditionally prompt the user for them
if (interaction != 0)
    % prompt the user for some paramters
    T = input('enter the final time T: \n');%
    levels = input('enter the number of refinement levels: ');%asks user←
    for value
        level = input('enter the first refinement level: ');
        field = input('enter the function-name for the vector field f: ');
    initial_y = input('enter the initial value vector: ');
    method = input('enter the numerical methods name: ');
end

lastlevel = level+levels;
level = level-1;
ctr=0;
while(level<lastlevel)
    level = level+1;
    N = 2^level;
    %% compute the numerical solution
    [discrete_y,t_discrete] = feval('uni',method,field,T,N,initial_y);

    if ((graphics != 0) & (interaction == 0))
        %% plot results
        figure(1);
        hold on;
        t_exact = t_discrete;
        exact_y = initial_y*exp(t_exact); %exact solution for f_1 (because we←
        %cheating, thi is not always
        %possible!)

        error=abs(exact_y-discrete_y);
        plot(t_exact,error,'-*;error;');
    end
    max_error = max(error);
    if(ctr > 0)
        maxerr(ctr) = max_error;
        EOC(ctr) = log(max_error_old/max_error)/log(2);
    end
    max_error_old = max_error;
    ctr=ctr+1;
end
title('errors for various levels');
EOC
hold off;
figure(2);
plot(log(maxerr),'r-*;|log h|->log error;');

```

A vector-valued uses the same function TMstep.m and is realised by TrapScript2d.m which lists as follows:

```

%%A small driver script for ODE methods (default is uniFE)
close all;% graphics
clear all;% variables

% set the paramters to defaults
interaction = 0;%1, where 0=noninteractive/1=interactive script
graphics = 1;%1, where 0=graphics off/1=graphics on
levels = 4;
level = 6;% first refinement level
method = 'Trap';% FE=forward Euler, Trap=trapezoidal (iterative)
T = 4*pi;% default final time
field = 'funkharmonic';
initial_y = [1;0];
% and conditionally prompt the user for them
if (interaction != 0)
    % prompt the user for some paramters
    T = input('enter the final time T: \n');%
    levels = input('enter the number of refinement levels: ');%asks user←
        for value
    level = input('enter the first refinement level: ');
    field = input('enter the function-name for the vector field f: ');
    initial_y = input('enter the initial value vector: ');
    method = input('enter the numerical methods name: ');
end

lastlevel = level+levels;
level = level-1;
ctr=0;
while(level<lastlevel)
    level = level+1;
    N = 2^level;
    %% compute the numerical solution
    [discrete_y,t_discrete] = feval('uni',method,field,T,N,initial_y);

    if ((graphics != 0) & (interaction == 0))
        %% plot results
        figure(1);
        hold on;
        t_exact = t_discrete;

        %exact solution for simple linear problem
        cost = cos(t_exact);
        sint = sin(t_exact);
        exact_y = [initial_y(1)*cost-initial_y(2)*sint;...
                    initial_y(1)*sint+initial_y(2)*cost];

        error=colnorm(exact_y-discrete_y,2);
        plot(discrete_y(1,:),discrete_y(2,:));
        %plot(t_exact,error,'-*;error;');
    end
    max_error = max(error);
    if(ctr > 0)
        maxerr(ctr) = max_error;
        EOC(ctr) = log(max_error_old/max_error)/log(2);
    end
    max_error_old = max_error;
end

```

```

ctr=ctr+1;
end
title('errors for various levels');
EOC
hold off;
figure(2);
plot(log(maxerr),'r-*;|log h|->log error;');

```

6.6. Other higher order methods and variants

6.6.1. Taylor's expansion methods. Earlier, in §6.1.4, we have seen that Taylor's expansion of order 2 about x_{n-1} leads to the FE method. Similarly we could see that a Taylor's expansion of order 2 about x_n leads to the BE method.

Let us see what happens if we push the Taylor expansion one order higher. For example, for a scalar-valued, sufficiently smooth function Y we have

$$Y(x_n) = Y(x_{n-1}) + Y'(x_{n-1})h_n + \frac{1}{2}Y''(x_{n-1})h_n^2 + \tau_n, \quad (6.6.1)$$

where

$$\tau_n = O(h_n^3). \quad (6.6.2)$$

If Y is the solution of an IVP with ode $\dot{y} = f(x, y)$, then we may write the expansion as follows

$$Y(x_n) = Y(x_{n-1}) + h_n f(x_{n-1}, Y(x_{n-1})) + \frac{h_n}{2} \frac{d}{dx} [f(x, Y(x))]_{x=x_{n-1}} + \tau_n. \quad (6.6.3)$$

The chain rule then implies

$$Y(x_n) = Y(x_{n-1}) + [h_n f + h_n^2 \partial_1 f + h_n^2 \partial_2 f f](x_{n-1}, Y(x_{n-1})) + \tau_n. \quad (6.6.4)$$

This suggest therefore the following numerical scheme

$$y^n = y^{n-1} + [h_n f + h_n^2 \partial_1 f + h_n^2 \partial_2 f f](x_{n-1}, y^{n-1}), \quad (6.6.5)$$

which has a local truncation error in $O(h_n^3)$, and, provided stability works out well for it, it is expected to yield a global error approximation error $O(h^2)$. This is also an explicit scheme, which makes it easy to compute. The trouble with the above scheme is that the derivatives of f may not be easy (or may be too costly computationally) to compute.

An idea to cure this problem is to replace those derivatives by finite differences. Indeed, we may consider using the expansion of Y' about x_{n-1} that gives

$$\frac{h_n^2}{2} Y''(x_{n-1}) = \frac{h_n}{2} (Y'(x_n) - Y'(x_{n-1}) + O(h_n^2)), \quad (6.6.6)$$

to replace the third term in (6.6.1) which yields

$$Y(x_n) = Y(x_{n-1}) + \frac{h_n}{2} f(x_{n-1}, Y(x_{n-1})) + \frac{h_n}{2} f(x_n, Y(x_n)) + O(h_n^3). \quad (6.6.7)$$

Perhaps unsurprisingly, we end up with the scheme

$$y^n = y^{n-1} + \frac{h_n}{2} f(x_{n-1}, y^{n-1}) + \frac{h_n}{2} f(x_n, y^n), \quad (6.6.8)$$

which we recognise as the Trapezoidal Method studied earlier. And our argument here is an alternative local truncation analysis proving (again) that the truncation error is of order 3. Thus providing a method which (assuming stability) has global order 2.

6.6.2. An explicited trapezoidal method. The main problem with the Trapezoidal Method is that it is implicit. A nose-first approach to the problem is to use Newton's method to solve the nonlinear timestep. But Newton[*] leads to the evaluation of derivatives of f , which is what we were trying to avoid in the first place. As an alternative to Newton's Method, we have seen a fixed-point method in 6.5.7 can lead to an equally good solution. [*]: Check!

Both Newton-type methods and fixed-point methods are iterative, we also see that one does not need to compute too many iterations. As soon as the iteration's error becomes comparable with the truncation error it is a good idea to stop the computation. Any further computations amount to a waste of computational resources and time, as one tries to be more precise than an already committed (truncation) error. For example, we have seen that the fixed-point iteration for the Trapezoidal Method can be interrupted at the second iteration. Similarly it is possible to consider an even simpler method. Where the first iteration is not performed using a FE step rather than one loop of the Trapezoidal Method's iteration. The first iteration is called a *predictor*, while the second iteration is a *corrector*, hence the name predictor–corrector for such an explicited Trapezoidal Method.

6.6.3. Problem (a trapezoidal-based predictor–corrector method). *The n -th step of the trapezoidal scheme the scalar ODE*

$$\dot{y}(x) = f(x, y) \quad (6.6.9)$$

is:

$$\begin{aligned} &\text{given } y^0, \dots, y^{n-1} \in \mathbb{R} \\ &\text{find } y^n \in \mathbb{R} \text{ such that} \\ &y^n = y^{n-1} + \frac{h}{2} (f(x_{n-1}, y^{n-1}) + f(x_n, y^n)). \end{aligned} \quad (6.6.10)$$

Assuming that y^{n-1} has been computed, consider, as a way to approximate the y^n in (6.6.10), the following iteratively defined sequence:

$$\begin{aligned} p^0 &:= y^{n-1} + h f(x_{n-1}, y^{n-1}) \\ p^i &:= y^{n-1} + \frac{h}{2} (f(x_{n-1}, y^{n-1}) + f(x_n, p^{i-1})), \text{ for } i \in \mathbb{N}. \end{aligned} \quad (6.6.11)$$

Following up on Problem 6.5.8 consider answering the following question. Deduce that one can achieve the same order of convergence with (6.6.11) as with (6.6.10) by stopping the iteration after one step, i.e., by using the solution \hat{y}^n of the scheme

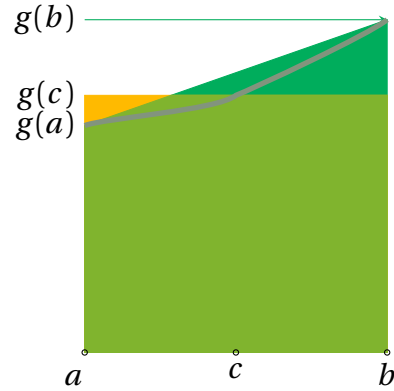
$$\begin{aligned} p^0 &:= y^{n-1} + h f(x_{n-1}, y^{n-1}) \\ \hat{y}^n &:= y^{n-1} + \frac{h}{2} (f(x_{n-1}, y^{n-1}) + f(x_n, p^0)). \end{aligned} \quad (6.6.12)$$

6.6.4. Midpoint Scheme. The Trapezoidal Method was derived, by a third order quadrature rule (the Trapezoidal Rule) approximating $\int_{x_{n-1}}^{x_n} Y'$ in order to obtain a second order method. Another third order quadrature is given by the Midpoint Rule:

$$\int_a^b g(x) dx = (b-a)g\left(\frac{a+b}{2}\right) + O((b-a)^3), \quad (6.6.13)$$

for functions g that are sufficiently smooth. The point $(a+b)/2$ is the midpoint between a and b , whence the name of the quadrature rule.

It seems counterintuitive, at first glance, that an approximation with a rectangular rule, can be as good in order as a trapezoidal approximation. However, noting that the point lies in the middle, means that as the interval width $b-a$ becomes small, the graph of g “resembles more and more” the straight line segment joining $(a, g(a))$ to $(b, g(b))$. Then the point $(c, g(c))$ where $c = (a+b)/2$ approaches the midpoint of the trapezoid’s oblique side. Therefore the part that is “added” matches closer and closer the part that is “subtracted” to the rectangle’s area to obtain the trapezoid’s.



Exercises and problems on basic numerical methods for ODE's

Exercise 6.1 (the forward Euler (FE) method). (a) Apply the forward Euler method to the IVP

$$\dot{y} = \frac{2y}{1+t}, \quad y(0) = 1 \quad (P6.1.1)$$

with $y^0 = 1$ and $T = 1$ to obtain approximations y_h^N to $y(1)$ for uniform time-steps $h = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$. You may help yourself with a pocket calculator and a table.

Hint. The exact solution is $Y(t) = (1+t)^2$.

- (b) Implement the forward Euler method applied to (P6.1.1) with $y^0 = 1$ and $T = 1$. Check your code by using it to find approximations y_h^N to $Y(1)$ for the uniform time-steps given in part (a) and comparing your answers.
- (c) Obtain approximations to $Y(1)$ for $h = 1/2, 1/4, 1/8, 1/16, 1/32, 1/64$. Plot a graph of $\log|y^{N(h)} - Y(1)|$ against $\log h$. Assuming that $|y^{N(h)} - Y(1)| = Ch^p$, use the graph to give an approximation of C and p .

Exercise 6.2 (the Euler methods for the model linear problem). Consider the Euler methods (both FE and BE) applied to approximate the solution of IVP

$$y' = \lambda y \text{ and } y(0) = y_0. \quad (P6.2.1)$$

- (a) Show that the forward Euler method applied to this problem yields the solution

$$y^n = (1 + h\lambda)^n y_0 \quad \forall n = 0, \dots, N(h). \quad (P6.2.2)$$

- (b) Show that the backward Euler method applied to this problem yields the solution

$$y^n = \frac{1}{(1 - h\lambda)^n} y_0 \quad \forall n = 0, \dots, N(h). \quad (P6.2.3)$$

(c) Suppose that $T > 0$ is fixed and $h > 0$, $N(h) \in \mathbb{N}$ variable satisfying $N(h)h = T$, denote by $(y_h^n)_{n=0,\dots,N(h)}$ the solution sequence of FE with initial value y_h^0 satisfying

$$y_h^0 \rightarrow y_0 \quad \text{as } h \rightarrow 0. \quad (\text{P6.2.4})$$

Show directly—without the use of the lecture's convergence results—that

$$y_h^{N(h)} \rightarrow Y(T) \quad \text{as } h \rightarrow 0. \quad (\text{P6.2.5})$$

Repeat the question for BE instead of FE.

Hint. Use the standard limit

$$\left(1 + \frac{x}{n}\right)^n \rightarrow \exp x \quad \text{as } n \rightarrow \infty. \quad (\text{P6.2.6})$$

Problem 6.3 (long time stability for Euler methods). Consider the numerical solutions to problem (P6.2.1) given by the forward and backward Euler methods, (P6.2.2) and (P6.2.3) respectively. Show that for $\lambda < 0$,

- (a) For $y_0 \neq 0$, $Y(x) \rightarrow 0$ monotonically as $x \rightarrow \infty$.
- (b) For the backward Euler method with $y^0 \neq 0$, $y^n \rightarrow 0$ monotonically as $n \rightarrow \infty$ for any $h > 0$.
- (c) For the forward Euler method with $y^0 \neq 0$,
 - ★ $|y^n| \rightarrow \infty$ as $n \rightarrow \infty$ if $h > 2/|\lambda|$;
 - ★ $|y^n| \rightarrow 0$ as $n \rightarrow \infty$ with oscillatory convergence if $h \in (1/|\lambda|, 2/|\lambda|)$;
 - ★ $|y^n| \rightarrow 0$ monotonically as $n \rightarrow \infty$ if $h < 1/|\lambda|$.

Exercise 6.4 (FE method for systems). Consider the numerical approximation of the \mathbb{R}^2 -valued IVP

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} (y_2 - 1)/y_1 + (1 - x) \\ (y_2 - 1)/y_1 + (2 + x) \end{bmatrix} \text{ for } x \in [0, 1) \text{ and } \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}. \quad (\text{P6.4.1})$$

(a) Apply the forward Euler method to obtain approximations $y_1^{N(h)}, y_2^{N(h)}$ to $Y_1(1), Y_2(1)$ for uniform time steps $h = 1$ and $1/2$. (The exact solution is given by $Y_1(x) = 2 + x$ and $Y_2(x) = (1 + x)^2$ for $x \geq 0$.)

Hint. Let $\mathbf{y} = (y_1, y_2)$. For each h , take $\mathbf{y}^0 = (2, 1)$ and apply the forward Euler method in vector form to find $\mathbf{y}^1, \dots, \mathbf{y}^{N(h)}$.

- (b) For each h and $0 \leq n \leq N(h)$, compare \mathbf{y}^n with the exact solution $\mathbf{Y}(nh)$ by finding $|\mathbf{Y}(nh) - \mathbf{y}^n|$.
- (c) Implement a code that computes the forward Euler approximation solution to (P6.4.1). Compare the output $\mathbf{y}^{N(h)}$ for the uniform timesteps $h = 1$ and $1/2$ and comparing your answers with those found in (a).
- (d) For $h = 1/128$ and $T = 1$, plot the error $|\mathbf{y}(nh) - \mathbf{y}^n|$ against $x_n = nh$ for $0 \leq n \leq N(h)$.

Problem 6.5 (discrete Gronwall Lemma). Suppose $(\phi_n)_{n \in \mathcal{N}}, (\eta_n)_{n \in \mathcal{N}}, (\chi_n)_{n \in \mathcal{N}}$ are sequences of real numbers with $\mathcal{N} = 1 : N$ for some $N \in \mathbb{N}$ such that

$$\phi_n, \eta_n, \chi_n \geq 0 \text{ and } \chi_n < 1, \forall n \in \mathcal{N} \quad (\text{P6.5.1})$$

satisfying the recursive inequality

$$\phi_n \leq \eta_n + \sum_{k=1}^n \chi_k \phi_k, \forall n \in \mathcal{N}. \quad (\text{P6.5.2})$$

(a) Show that

$$\phi_n \leq \eta_n + \sum_{k=1}^n \left(\chi_k \eta_k \prod_{i=k}^n \frac{1}{1 - \chi_i} \right). \quad (\text{P6.5.3})$$

Hint. Let $R_0 = 0$ and $R_n = \sum_{k=1}^n \chi_k \phi_k$, for $n \geq 1$, and rewrite the recursion (P6.5.2) as

$$R_n - R_{n-1} - \chi_n R_n \leq \chi_n \eta_n, \forall n \in \mathcal{N}. \quad (\text{P6.5.4})$$

Multiply both sides of each of these inequalities by the *summing factor* $\prod_{k=1}^{n-1} (1 - \chi_k)$, then sum and manipulate as to obtain an upper bound on R_n and replace in (P6.5.2).

(b) Show that for each $\delta > 0$ there exists $\bar{\chi}_\delta \in (0, 1)$ such that if $\chi_n \leq \bar{\chi}_\delta$, and $X_0 = 0$ and $X_n = \sum_{k=1}^n \chi_k$, for all $n \in \mathcal{N}$, then (P6.5.3) implies

$$\phi_n \leq \eta_n + \sum_{k=1}^n \chi_k \eta_k \exp((1 + \delta)(X_n - X_{k-1})). \quad (\text{P6.5.5})$$

This inequality looks very similar to the conclusion in the continuous analogue of Gronwall's Lemma, except for the factor $(1 + \delta)$ which seems necessary in this case.

Hint. Start by showing that the inequality $1 - x \geq \exp(-(1 + \delta)x)$ is valid for $x \in (0, \log(1 + \delta)/(1 + \delta))$. You may find it useful to study the sign of the function $(0, 1) \ni x \mapsto g(x) = 1 - x - \exp(-(1 + \delta)x) \in \mathbb{R}$.

Exercise 6.6 (Euler's forward-backward "reflection"). The aim of this exercise is to illustrate the reason behind the fact that forward and backward Euler methods are said each to be the *reflected* method of the other.

- (a) Show that applying one step of the backward Euler method with time step h followed by one step of the forward Euler method with time step $-h$ always results in the initial value \mathbf{y}^0 being recovered exactly.
- (b) Show that the result in a need not hold in general for one FE step with time step h followed by one BE step with negative timestep $-h$.
- (c) Suppose $\text{Dom } \mathbf{f}(x_1, \cdot) = D$ is a simply connected domain (e.g., all of \mathbb{K}^d), and using the fact that a map $\mathbf{g} : D \rightarrow \mathbb{R}^d$ is invertible when $D \ni \mathbf{g}(\mathbf{y})$ is bijective for all $\mathbf{y} \in D$, find one a sufficient conditions on \mathbf{f} and h for which a FE step followed by BE step with negative timestep is the identity.

Hint. Use the Neumann series Lemma to ensure that $\partial_{\mathbf{y}}[\mathbf{y} + h \mathbf{f}(x_1, \mathbf{y})]$ is an invertible linear map.

- (d) (i) Apply the backward Euler method to the IVP

$$\dot{y} = -\frac{y}{1+x} \text{ on } [0, 1] \text{ and } y_0 = 1, \quad (\text{P6.6.1})$$

with $y^0 = y_0$ and $h = 1/2$ to obtain approximations y^1 and y^2 of $Y(1/2)$ and $Y(1)$, respectively.

(ii) Check the value y^2 found in (i) by using it as a new initial value $\tilde{y}^0 = y^2$ and applying the forward Euler method with negative uniform time step $h = -1/2$ to obtain an approximations \tilde{y}^1 and \tilde{y}^0 to $Y(1/2)$ and $Y(0) = y_0$.

Exercise 6.7 (BE for linear problems). Suppose \mathbf{f} is linear in its second argument, i.e.,

$$\mathbf{f}(x, \mathbf{y}) := \mathbf{A}(x)\mathbf{y} \quad (\text{P6.7.1})$$

where $\mathbf{A}(x) \in \mathbb{K}^{d \times d}$ is a matrix whose coefficients depend on x , and $\mathbf{A} \in C^\infty([0, \infty); \mathbb{K}^{d \times d})$.

(a) Write an expression, in terms of \mathbf{y}^{n-1} , x_n and h_n and $\mathbf{A}(x_n)$ for the n -th value of \mathbf{y}^n of the backward Euler scheme for the IVP

$$\dot{\mathbf{y}} = \mathbf{A}(x)\mathbf{y} (= \mathbf{f}(x, \mathbf{y})) \text{ and } \mathbf{y}(0) = \mathbf{y}_0. \quad (\text{P6.7.2})$$

(b) Is \mathbf{y}^n always well-defined? If not, find a condition relating $\mathbf{A}(x_n)$ and h_n which allows you to ensure that \mathbf{y}^n is well defined.

Hint. Use Neumann's series Lemma B.2.10.

Problem 6.8 (fixed point iteration for BE on nonlinear ODE's). Consider the backward Euler method:

$$\mathbf{y}^n = \mathbf{y}^{n-1} + h_n \mathbf{f}(x_n, \mathbf{y}^n) \text{ for } n = 1, \dots, N. \quad (\text{P6.8.1})$$

For a fixed n , assuming \mathbf{y}^{n-1} has been computed, then \mathbf{y}^n is the solution of the (possibly nonlinear) system (P6.8.1) and it can be computed (or approximated) using a fixed-point iteration.

For example, for a given initial guess \mathbf{y}_0^n , we define a sequence $(\mathbf{y}_k^n)_{k \in \mathbb{N}_0}$ via

$$\mathbf{y}_{k+1}^n := \mathbf{y}^{n-1} + h_n \mathbf{f}(x_{n+1}, \mathbf{y}_k^n), \text{ for } k \in \mathbb{N}_0. \quad (\text{P6.8.2})$$

If this sequence converges, say, \mathbf{y}_* , and $\mathbf{f}(x_{n+1}, \cdot)$ is a continuous map, then it follows that

$$\mathbf{y}_*^n = \mathbf{y}^{n-1} + h_n \mathbf{f}(x_n, \mathbf{y}_*^n) \quad (\text{P6.8.3})$$

and hence that $\mathbf{y}^n = \mathbf{y}_*^n$.

Denote by $D := \text{Dom } \mathbf{f}$ and consider the map $\mathbf{g} : D \rightarrow \mathbb{K}^d$ such that

$$\mathbf{g}(\mathbf{y}) := \mathbf{y}^{n-1} + h_n \mathbf{f}(x_n, \mathbf{y}) \text{ for } \mathbf{y} \in \text{Dom } \mathbf{f}. \quad (\text{P6.8.4})$$

Show that if the map \mathbf{f} is globally Lipschitz with respect to its second argument with Lipschitz constant Λ and h is taken less than $1/\Lambda$ then the sequence of iterates \mathbf{y}_k^n converges to the solution \mathbf{y}^n as $k \rightarrow \infty$.

Hint. Using the Banach–Caccioppoli Contraction Lemma show that map \mathbf{g} is has a fixed point under the above condition.

Exercise 6.9 (the backward Euler method). (1) Apply the backward Euler method to the IVP (P6.1.1) in exercise 6.1 with $y^0 = 1$ and $T = 1$ to obtain approximations $y^{N(h)}$ to $y(1)$ for uniform time steps $h = \frac{1}{2}, \frac{1}{4}$.

(2) Use Matlab[®] to implement the backward Euler method applied to (P6.1.1) with $y^0 = 1$ and $T = 1$.

(i) Write and implement a Matlab[®] function in `BEspecial.m` which finds $y^1, \dots, y^{N(h)}$ directly. This will not need to call any “function_f.m”.

- (ii) Write a Matlab[®] function in BEFPI.m which uses the a fixed point iteration to find $y^1, \dots, y^{N(h)}$. For each $n = 0, \dots, N(h) - 1$, take the initial guess y_0^{n+1} of y^{n+1} to be the numerical solution at the previous discrete time level y^n :

$$y_0^{n+1} = y^n. \quad (\text{P6.9.1})$$

Your file BEFPI.m should accept as inputs function_f (a file handle to the ODE field f), N, T, y_0 and

- ★ the maximum number of permitted iterations max_iteration_count,
- ★ the tolerance for the fixed point iteration iteration_error_tolerance.

Your code should perform fixed point iterations until

$$\left| \frac{y_k^{n+1} - y_{k-1}^{n+1}}{y_{k-1}^{n+1}} \right| \leq \text{iteration_error_tolerance} \text{ or } k = \text{max_iteration_count}, \quad (\text{P6.9.2})$$

and then take $y^{n+1} := y_k^{n+1}$.

In each case, check your code by using it (with max_iteration_count = 10000 and iteration_error_tolerance = 1e-10) to obtain approximations $y^{N(h)}$ to $y(1)$ for uniform time steps $h = \frac{1}{2}, \frac{1}{4}$ and comparing your answers with those found in part (1).

Problem 6.10 (trapezoidal method). Consider the numerical approximation, via the trapezoidal method, of the IVP

$$\dot{y} = f(x, y) \text{ on } [0, T] \text{ and } y(0) = y_0, \quad (\text{P6.10.1})$$

with $y_0, y \in \mathbb{R}$ and $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ analytic and globally Lipschitz.

Let $0 = x_0 < x_1 < \dots < x_N = T$ be a time partition, with uniform timestep $h := x_n - x_{n-1}$ for all $n \in 1 : N$. The trapezoidal method is given by

$$y_0 = y_0 \text{ and } y^n = y^{n-1} + \frac{h}{2} (f(x_n, y^n) + f(x_{n-1}, y^{n-1})), \text{ for } n \in 1 : N. \quad (\text{P6.10.2})$$

(a) Is this an explicit or implicit scheme? Explain in a sentence what is involved in an implementation of this method. State a good feature of the method. State a possible difficulty that one may face in implementing this method, regarding the timestep h , when $f(x, y)$ is not linear in y .

(b) Explain why the exact solution of (P6.10.1), Y , and all its derivatives are bounded on $[0, T]$.

(c) The n -th timestep *local truncation error* (LTE) is

$$T_n(h) = Y(x_n) - Y(x_{n-1}) - \frac{h}{2} (f(x_{n-1}, Y(x_{n-1})) + f(x_n, Y(x_n))). \quad (\text{P6.10.3})$$

Show that there exists $B > 0$ and $p \in \mathbb{N}$, such that

$$|T_n(h)| \leq B h^{p+1}. \quad (\text{P6.10.4})$$

Identify the value of p and relate p to the order of the method.

(d) Suppose that $\mu \leq \partial_y f \leq \lambda$, where $\mu, \lambda \in \mathbb{R}$, $\mu \leq \lambda < 0$. Show that for $h_0 = -2/\mu$ we have

$$\left| Y(x_n) - y^n \right| \leq \frac{2 + h\lambda}{2 - h\lambda} \left| Y(x_{n-1}) - y^{n-1} \right| + \frac{2}{2 - h\lambda} |T_n(h)|, \quad \forall n \in 1 : N, h \in (0, h_0). \quad (\text{P6.10.5})$$

(e) Deduce that there exists a constant $C > 0$ such that

$$\left| Y(x_n) - y^n \right| \leq C h^p, \forall n \in 1:N, h \in (0, h_0). \quad (\text{P6.10.6})$$

APPENDIX A

Linear algebra

This section summarises the linear algebra concepts used in these notes. It is not meant to replace any textbook on the subject. Linear Algebra is most likely, at a par with Calculus, the most written about topic in Mathematics, with a corresponding zoo of textbooks. The “decent textbooks” section of the zoo is fortunately much smaller and in that section has been living **Halmos:74:book:Finite-dimensional** for over half-century which is my favourite as it emphasises the “coordinate-free” approach.

A.1. Algebra

A.1.1. Definition of group and Abelian (or commutative) group. A set G and an operation \star form a group (G, \star) , or simply G when \star is understood from the context, if and only if

$$x \star (y \star z) = (x \star y) \star z \quad \forall x, y, z \in G, \quad (\text{A.1.1})$$

for some $e \in G$, called a *neutral element*,

$$x \star e = e \star x = x \quad \forall x \in G, \quad (\text{A.1.2})$$

and for each $x \in G$ there exists x' such that

$$x \star x' = x' \star x = e, \quad (\text{A.1.3})$$

and x' is usually indicated as x^{-1} . We say that a group (G, \star) is *Abelian* (or *commutative*) if

$$x \star y = y \star x \quad \forall x, y \in G. \quad (\text{A.1.4})$$

A.1.2. Exercise. Let (G, \star) be a group with a neutral element e .

- (a) a group has a unique neutral element e by assuming there is another one, say f and showing that $f = e$,
- (b) each element x has a unique inverse x' by assuming there is another one, say x'' and showing that $x' = x''$.

A.1.3. Definition of subgroup. Given a group (G, \star) , a subset $H \subseteq G$ is called a subgroup of G , if (H, \star) forms a group.

A.1.4. Exercise. Let X be any set, and let X^X be the set of all maps $\phi : X \rightarrow X$. Consider the composition operation on X^X denoted by \circ and defined as

$$\phi \circ \psi(x) := \phi(\psi(x)) \text{ for } x \in X, \quad (\text{A.1.5})$$

for each $\phi, \psi \in X^X$.

- (a) Show that \circ is associative on X^X and that it has a neutral element.
- (b) Show with a counterexample that (X^X, \circ) does not form a group in general.

- (c) Consider the subset \mathcal{S} of X^X formed by the bijective maps and show that (\mathcal{S}, \circ) is a group.
- (d) Show with a counterexample that (\mathcal{S}, \circ) is generally not Abelian.
- (e) Can you explain why X^X is a good notation?
- (f) The group \mathcal{S} is sometimes denoted $X!$, can you explain why this a good notation?

A.1.5. Example (the additive integers form a group). The set of all integers

$$\mathbb{Z} = \{0, 1, -1, 2, -2, \dots\} \quad (\text{A.1.6})$$

forms an Abelian group with respect to the usual addition $+$.

A.1.6. Exercise (the multiplicative integers do not form a group). Show that neither (\mathbb{Z}, \times) nor $(\mathbb{Z} \setminus \{0\}, \times)$ form a group, where \times indicates the usual multiplication.

A.1.7. Definition of field. A *field* is a set K with two operations, say $+$ and \times , such that

- (i) $(K, +)$ is an Abelian group with 0 denoting its neutral element
- (ii) $(K \setminus \{0\}, \times)$ is also an Abelian group,
- (iii)

$$x \times (y + z) = x \times y + x \times z. \quad (\text{A.1.7})$$

Usually the operation \times is omitted and the neutral element for (K, \times) is denoted 1.

A.1.8. Exercise. Let $(K, +, \times)$ be a field. Show that

$$0 \times x = 0 \quad \forall x \in K. \quad (\text{A.1.8})$$

A.1.9. Number fields. The primary examples of fields are \mathbb{Q} , \mathbb{R} and \mathbb{C} , with the usual summation $+$ and multiplication \times (omitted when using variables). Also the set of rational numbers \mathbb{Q} is a field with these operations. The fields \mathbb{Q} and \mathbb{R} are ordered with \leq ; \mathbb{C} cannot be ordered in a meaningful way. The fields \mathbb{R} and \mathbb{C} are complete metric spaces (i.e., a sequence is Cauchy if and only if it converges) with respect to the metric define by the absolute value (or modulus) of the difference, i.e.,

$$d(x, y) := |x - y|. \quad (\text{A.1.9})$$

The field of rationals is not complete; and this can be a source of headache. So all mathematically sound theories of matrices occur over one of the fields \mathbb{R} or \mathbb{C} , although in practice, as we shall see, all “real numbers” are “realised” on a computer as floating-point numbers. Thus the “real numbers” on a computer are actually a finite subset of \mathbb{Q} ... yet, developping mathematical theory of matrices based on \mathbb{Q} is cumbersome and \mathbb{R} or \mathbb{C} are usually used, with \mathbb{Q} being just a handy way to approximate \mathbb{R} .

We recall that \mathbb{C} is defined as being \mathbb{R}^2 with the notation $1 := 1_{\mathbb{C}} := (1_{\mathbb{R}}, 0)$ and $i := (0, 1_{\mathbb{R}})$ allowing to write each $z \in \mathbb{C}$ as $z = x + i y$ for some unique pair $(x, y) \in \mathbb{R}$. If $z = x + i y$ we write $x = \text{re}(z) = \text{re } z$ and $y = \text{im}(z) = \text{im } z$. The *conjugate* \bar{z} or z^* of $z = x + i y$ is $\bar{z} = z^* := x - i y$. We have $z \in \mathbb{R}$ if and only if $z = \bar{z}$.

In the rest of these notes, we will denote by \mathbb{K} any of \mathbb{R} or \mathbb{C} .

A.2. Vector spaces

Throughout these notes $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ despite the fact that many definitions and results apply to more general fields.

A.2.1. Definition of vector space. A set V is called a \mathbb{K} -vector space if and only if V is an (additive) Abelian group and a *scalar-vector multiplication* is defined

$$\begin{aligned} \langle \cdot, \cdot \rangle_{\mathbb{K}, V} : \mathbb{K} \times V &\rightarrow V \\ (\lambda, v) &\mapsto \langle \lambda, v \rangle_{\mathbb{K}, V} = \lambda v, \end{aligned} \quad (\text{A.2.1})$$

(where the brackets are omitted when the meaning is clear) and satisfies the properties

$$\lambda(v + w) = \lambda v + \lambda w, (\lambda + \mu)v = \lambda v + \mu v, \quad (\text{A.2.2})$$

for all $\lambda, \mu \in \mathbb{K}$ and $v, w \in V$. The $0_{\mathbb{K}}$ and 0_V are both denoted by 0 as this does not create any confusion. The unit of \mathbb{K} is 1 . Also addition and its inverse (*opposite*) in \mathbb{K} and that in V are denoted by the same symbols $+$ and $-$, respectively.

Although not a must, it is often useful to use the scalar-vector commutativity convention whereby

$$v\lambda := \lambda v. \quad (\text{A.2.3})$$

A.2.2. Exercise. Let V be a \mathbb{K} -vector space. Show that

- (a) For any $v \in V$, $1v = v$.
- (b) For any $v \in V$, $0v = 0$.
- (c) For any $v \in V$, $(-1)v = -v$.

A.2.3. Definition of vector subspace. Given a \mathbb{K} -vector space V a *vector subspace* U of V is a subset $U \subseteq V$ which is closed under the vector space operations $+$ and $\langle \cdot, \cdot \rangle_{\mathbb{K}, V}$.

A.2.4. Exercise. Show that each \mathbb{C} -vector space is a \mathbb{R} -vector space.

A.2.5. Complexification of a \mathbb{R} -vector space.

PROBLEM. Let V be a \mathbb{R} -vector space, show that there exists a \mathbb{C} -vector space W such that $V \subseteq W$ and V is a \mathbb{R} -subspace, albeit not a \mathbb{C} -subspace there, of W .

Solution. Let W be the direct sum $V \oplus iV$ whereby

$$V \oplus iV := V \times V = \{z = (x, y) : x, y \in V\} \quad (\text{A.2.4})$$

with the vector addition of $z_1, z_2 \in W$ given by

$$z_1 + z_2 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ y_1 + y_2 \end{bmatrix} \quad (\text{A.2.5})$$

and the following complex scalar-vector multiplication of $\lambda = \alpha + i\beta$ with $z = (x, y)$

$$(\alpha + i\beta)z = \begin{bmatrix} \alpha x - \beta y \\ \alpha y + \beta x \end{bmatrix}. \quad (\text{A.2.6})$$

It is worth making sure that the axioms of \mathbb{C} -vector space are satisfied for W . A notation sometimes used to indicate W is $\mathbb{C}V$.

A.3. The model finite dimensional \mathbb{K} -vector space \mathbb{K}^n

Most of these notes (and most of most linear algebra books around) are about the finite dimensional spaces \mathbb{K}^n with $\mathbb{K} = \mathbb{R}$ or \mathbb{C} . By definition the set \mathbb{K}^n is the n -th cartesian power of \mathbb{K}

$$\mathbb{K}^n := \left\{ \mathbf{v} = [v_i]_{i=1,\dots,n} : v_i \in \mathbb{K} \quad \forall i = 1, \dots, n \right\} \quad (\text{A.3.1})$$

where

$$\mathbf{v} = [v_i]_{i=1,\dots,n} = (v_1, \dots, v_n). \quad (\text{A.3.2})$$

When using *matrix notation* the vectors of \mathbb{K}^n are written as columns, leaving us with the 4 alternative notations:

$$\mathbf{v} = (v_1, \dots, v_n) = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = [v_i]_{i=1,\dots,n}. \quad (\text{A.3.3})$$

Note that unlike many other texts, in this notation the round braces (and) do not distinguish between a column and a row, whereas the square braces [and] do. Unless otherwise stated a *vector of \mathbb{K}^n* is a *column vector*.

A.3.1. Definition of matrix. A *matrix* (plural *matrices*¹) is defined as an “row vector of column vectors”. That is given m vectors $\mathbf{a}^1, \dots, \mathbf{a}^m \in \mathbb{K}^n$, with that order, we may form a corresponding as an ordered array (or table) of vectors:

$$\mathbf{A} := [\mathbf{a}^1 \quad \dots \quad \mathbf{a}^m] = [\mathbf{a}^1 \quad \dots \quad \mathbf{a}^m]. \quad (\text{A.3.4})$$

Unless otherwise stated, we use the convention that a matrix is denoted in the bold uppercase letter corresponding to the bold lowercase letter denoting its columns. Often, it is useful for one to explicitly look at the column vectors forming the matrix as

$$\mathbf{a}^j = a_i^j \text{ for } i = 1, \dots, n, j = 1, \dots, m, \quad (\text{A.3.5})$$

and in fact many textbooks merely define (somewhat misleadingly) a matrix as a “table” (or “2-dimensional array”) of scalar “entries”.

Matrix notation is a special way to denote the matrix \mathbf{A} as

$$\mathbf{A} = \begin{bmatrix} a_1^1 & \dots & a_1^m \\ \vdots & \ddots & \vdots \\ a_n^1 & \dots & a_n^m \end{bmatrix} = [a_i^j]_{i=1,\dots,n}^{j=1,\dots,m}. \quad (\text{A.3.6})$$

Note that in most texts the entry “row-index-down-col-index-up” (RIDCIU) a_i^j is written “row-index-first-col-index-last” (RIFCIL) $a_{i,j}$ or a_{ij} .² We try to stick to RIDCIU notation, but, due to cultural inertia, RIFCIL may sneak into our text or, even more so, in the problems and exercises.

¹“matrixes” is an acceptable spelling in English, but for some strange reason mathematicians dislike it and stick to the inconsistent “matrices”.

²The RIDCIU notation, which originates in differential geometry and theoretical physics, is more precise than RIFCIL: when dropping one of the indexes it is still clear whether we are talking about a row or a column entry. It has also other advantages such as space-economy and variance-check that hopefully need no preaching about as they should become apparent to the experienced reader. For those who know differential geometry, strictly speaking the usual notation is RIUCII (a_j^i), the row is known as “covariant” and the column as “contravariant”.

As with vectors we can “extract” the i -th column of a matrix \mathbf{A}

$$[\mathbf{A}]^i = \begin{bmatrix} a_1^i \\ \vdots \\ a_n^i \end{bmatrix} = \mathbf{a}^i. \quad (\text{A.3.7})$$

and its i -th row

$$[\mathbf{A}]_i = [a_i^1 \quad \dots \quad a_i^m] =: \mathbf{a}_i. \quad (\text{A.3.8})$$

This means that the boldface lowercase version of the (uppercase version of a) letter is used with a superindex to indicate a row and with a subindex to indicate a column. The square brackets are used when special operations are needed.

A notable exception to the above rules is the *identity matrix* \mathbf{I} whose entries are traditionally denoted by *Kronecker's delta notation* whereby

$$\delta_i^j := \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \quad (\text{A.3.9})$$

for $i, j = 1, \dots, n$ and whose i -th column and row are denoted by

$$\mathbf{e}^i := \begin{bmatrix} \delta_1^i \\ \vdots \\ \delta_n^i \end{bmatrix} \quad \mathbf{e}_i := [\delta_i^1 \quad \dots \quad \delta_i^n] \quad (\text{A.3.10})$$

Matrix-matrix multiplication of a matrix $\mathbf{A} \in \mathbb{K}^{n \times m}$ and $\mathbf{B} \in \mathbb{K}^{m \times l}$ is defined as follows

$$[\mathbf{AB}]_i^j = \sum_{k=1}^m [\mathbf{A}]_i^k [\mathbf{B}]_k^j. \quad (\text{A.3.11})$$

In particular, if a row \mathbf{a} and column \mathbf{a} have the same number of entries, say n , then

$$\mathbf{a}\mathbf{a} = \sum_{i=1}^n a^i a_i. \quad (\text{A.3.12})$$

A very useful operation on matrices is the so-called *transposition* which is defined for a column

$$\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \text{ as the row } \mathbf{a}^\mathsf{T} = [a^1 \quad \dots \quad a^n] \text{ with } a^i = a_i. \quad (\text{A.3.13})$$

And for a $\mathbb{K}^{n \times m}$ matrix

$$\mathbf{A} = [\mathbf{a}^1 \quad \dots \quad \mathbf{a}^m] \text{ as the } \mathbb{K}^{m \times n} \text{ matrix } \mathbf{A}^\mathsf{T} = \begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_m \end{bmatrix} \text{ where } \mathbf{a}_i := (\mathbf{a}^i)^\mathsf{T}. \quad (\text{A.3.14})$$

A related operation (which is the same if $\mathbb{K} = \mathbb{R}$) is the *adjoint*

$$\mathbf{A}^* = \begin{bmatrix} \mathbf{a}^{1*} \\ \vdots \\ \mathbf{a}^{m*} \end{bmatrix} \text{ where } \mathbf{a}^{i*} := [\bar{a}^1 \quad \dots \quad \bar{a}^n] \text{ with } \bar{a}^i = \overline{a_i} \quad (\text{A.3.15})$$

and $\bar{z} := z^* := x - iy$ for a complex number $z = x + iy$.

A.3.2. Definition of linear combination, spanned space, span, dimension. Given a finite sequence of vectors $(v^i)_{i=1,\dots,n}$, a *linear combination* thereof, is an expression of the type

$$\sum_{i=1}^n \alpha_i v^i := \alpha_1 v^1 + \dots + \alpha_n v^n. \quad (\text{A.3.16})$$

In matrix notation this can also be written as simply as

$$v^T \alpha, \text{ where } v^T = [v^1 \ \dots \ v^n] \text{ and } \alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}. \quad (\text{A.3.17})$$

Given a set of vectors $\mathcal{V} \subseteq V$, not necessarily finite, we define the *spanned space* (also known as *span*) of \mathcal{V} , $\text{Span } \mathcal{V}$ to be the set of all possible (finite!) linear combinations³ of V . In symbols this may be written as

$$\text{Span } \mathcal{V} := \left\{ \sum_{i=1}^n \alpha_i v^i : n \in \mathbb{N}, \alpha \in \mathbb{K}^n, (v^i)_{i=1,\dots,n} \text{ in } \mathcal{V} \right\}. \quad (\text{A.3.18})$$

A.3.3. Proposition (characterisation of the span). *Given a subset \mathcal{V} of a \mathbb{K} -vector space V . The span of \mathcal{V} is the smallest possible subspace of V that contains \mathcal{V} .*

A.3.4. Definition of basis. A subset \mathcal{V} of a \mathbb{K} -vector space V is called a *basis* if and only if

$$\mathcal{V} \text{ is linearly independent and } \text{Span } \mathcal{V} = V. \quad (\text{A.3.19})$$

We say that V is a *finite dimensional vector space* if it has on basis that is finite. In this case, we think of a basis of V as being an ordered set, more precisely we think of it as being a row of elements of V

$$v^T = (v^j)_{j=1,\dots,n} = [v^1 \ \dots \ v^n] = [v^j]^{j=1,\dots,n} \quad (\text{A.3.20})$$

in four interchangeable notations.

A.3.5. Theorem (finite dimension). *Let V be a finite dimensional vector space, then any two bases have the same number of elements. We call this number the dimension of the vector space.*

Proof Omitted. □

A.3.6. Remark (variants). The concept of basis given in A.3.4 is also known as *algebraic basis* or *Hamel basis*. There are other definitions of basis, such as orthonormal bases, Schauder bases, etc. Since these are more sophisticated notions, we always prepend them with an adjective while reserving the plain “basis” for an algebraic (or Hamel) basis.

³Linear combinations are finite by definition. Infinite linear combinations are useful, and we always prepend the adjective in that case.

A.4. Algebras and polynomials

A.4.1. Definition. A \mathbb{K} -vector space M is called an \mathbb{K} -*algebra* if on top of being a vector space a multiplication $\star : M \times M \rightarrow M$ is defined such that $(M, +, \star)$ is a ring with unit \mathcal{I} and

$$\lambda(\mathcal{A} \star \mathcal{B}) = (\lambda \mathcal{A}) \star \mathcal{B} \quad \forall \lambda \in \mathbb{K}, \mathcal{A}, \mathcal{B} \in M \quad (\text{A.4.1})$$

$$(\mathcal{A} + \mathcal{B}) \star \mathcal{C} = \mathcal{A} \star \mathcal{C} + \mathcal{B} \star \mathcal{C} \quad \forall \mathcal{A}, \mathcal{B}, \mathcal{C} \in M \quad (\text{A.4.2})$$

$$\mathcal{A} \star (\mathcal{B} + \mathcal{C}) = \mathcal{A} \star \mathcal{B} + \mathcal{A} \star \mathcal{C} \quad \forall \mathcal{A}, \mathcal{B}, \mathcal{C} \in M \quad (\text{A.4.3})$$

In usual algebras the operation sign \star is omitted and the field of scalars \mathbb{K} understood by the context and we talk simply about an “algebra M ”. An element $\mathcal{A} \in M$ is *invertible* (or *nonsingular*) if there exists another element $\mathcal{B} \in M$ such that

$$\mathcal{A} \mathcal{B} = \mathcal{B} \mathcal{A} = \mathcal{I} \quad (M\text{'s unit}). \quad (\text{A.4.4})$$

Such a \mathcal{B} , if it exists, is denoted by \mathcal{A}^{-1} . When working with an algebra M with unit \mathcal{I} , the elements of the form $\lambda \mathcal{I}$ with $\lambda \in \mathbb{K}$ are identified with λ and thus \mathcal{I} is often denoted simply 1.

A.4.2. Example. The prime example of an algebra is that of linear operators on a vector space V , $\text{Lin}(V \rightarrow V)$ that will be defined later in ???. Using the *standard isomorphism* the operator algebra $\text{Lin}(\mathbb{K}^n \rightarrow \mathbb{K}^n)$ is seen to be isomorphic to the *matrix algebra* $\mathbb{K}^{n \times n}$ discussed in A.3.1.

A.4.3. Definition of polynomial. A (*univariate*) *polynomial with coefficients* in \mathbb{K} is a composition of \mathbb{K} -algebra operations of the type

$$p(X) := \sum_{i \in S} p_i X^i \quad (\text{A.4.5})$$

for some finite set $S \subseteq \mathbb{N}_0$, $[p_i]_{i \in \mathbb{N}_0}$ for all $i \in S$, and where X^i symbolizes $X \star \dots \star X$ i -times. If $S = \{0\}$ or $p_i = 0$ for all $i > 0$, $i \in S$, p is called a *constant polynomial* and if moreover $p_0 = 0$ (or $S = \emptyset$) then p is called the *polynomial-zero*.

The *degree* of the polynomial p is $\deg p$ is defined as

$$\deg p := \max k \in \mathbb{N}_0 : p_k \neq 0. \quad (\text{A.4.6})$$

(By definition $\max \emptyset = -\infty$ and thus $\deg 0 = -\infty$.) The polynomial p can be *evaluated* on any \mathbb{K} -algebra M (not only \mathbb{K}) to produce a *polynomial map*, also denoted $p : M \rightarrow M$.

A.4.4. Example (2×2 skew symmetric equidiagonal matrices are the complex numbers). Consider the subset C of $\mathbb{R}^{2 \times 2}$ matrices A that are skew symmetric and equidiagonal, i.e., of the form

$$A = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \text{ for some } a, b \in \mathbb{R}. \quad (\text{A.4.7})$$

Check that C is a \mathbb{R} -algebra with the usual scaling, sum and multiplication of matrices, making sure you identify the zero element and the identity in C . Surprisingly the matrix-matrix multiplication, which is usually not commutative, is commutative in C : check this as well. Furthermore, any nonzero element in C is invertible. Find the inverse of the matrix A in (A.4.7).

Consider the polynomial $p(X) := X^2 + 1$. This is a polynomial of degree 2. It is well known that p has no root in \mathbb{R} . Now show that the matrix

$$J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \in C \quad (\text{A.4.8})$$

satisfies $p(J) = \mathbf{0}$ (the zero matrix) and is hence a root of the polynomial p in C . In fact, it is possible to show that the \mathbb{R} -algebra C is in fact isomorphic to the \mathbb{R} -algebra \mathbb{C} .

A.4.5. Polynomial identity principle. If p is polynomial-zero, then p is constantly zero-valued on \mathbb{R} , i.e., $p(x) = 0$ for all $x \in \mathbb{R}$. The converse, which is not obvious, is also true *if \mathbb{K} is the field of real or complex numbers.*⁴

THEOREM (identity principle of polynomials). *A polynomial p with coefficients in $\mathbb{K} = \mathbb{R}$ or \mathbb{C} evaluates constantly to zero on \mathbb{R} (i.e., $p(x) = 0$ for all $x \in \mathbb{R}$) if and only if p is the polynomial-zero, i.e., all of its coefficients are zero.*

The consequences of this theorem are important in analysis when approximating functions by polynomials, such as Taylor's polynomials and here are some of them:

- (1) Two polynomials are equal (i.e., they have all coefficients equal) if and only if they coincide on \mathbb{R} .
- (2) A polynomial of degree $n \in \mathbb{N}_0$ can have no more than n roots. (Reminder: the polynomial-zero has degree $-\infty$ which is not a number and thus not in \mathbb{N}_0 and is excluded from this statement.)
- (3) The set of all polynomials of degree at most n on \mathbb{K} is a \mathbb{K} -vector space of dimension $n + 1$.

A.4.6. Theorem. *The set of all univariate polynomials on a field \mathbb{K} , denoted $\mathbb{P}_{\mathbb{K}}$, where X is the indeterminate, is an (infinite dimensional) \mathbb{K} -algebra with the following operations*

$$\lambda p(X) = \sum_{i=0}^{\deg p} \lambda p_i(X), \quad (\text{A.4.9})$$

$$p(X) + q(X) = \sum_{i=0}^{\deg p \vee \deg q} (p_i + q_i) X^i, \quad (\text{A.4.10})$$

$$p(X)q(X) = \sum_{i=0}^{\deg p + \deg q} \left(\sum_{j=0}^i p_j q_{i-j} \right) X^i. \quad (\text{A.4.11})$$

When $\mathbb{K} = \mathbb{R}$ we write \mathbb{P} to indicate $\mathbb{P}_{\mathbb{R}}$.

A.4.7. Theorem (Euclidean algorithm for polynomial of \mathbb{R} or \mathbb{C}). *Given two polynomials $p, d \in \mathbb{P}_{\mathbb{K}}$, there exists a unique pair of polynomials $(p, r) \in \mathbb{P}_{\mathbb{K}} \times \mathbb{P}_{\mathbb{K}}$ such that*

$$p(X) = q(X)d(X) + r(X) \text{ and } \deg r < \deg d. \quad (\text{A.4.12})$$

(Note that uniqueness is not guaranteed in general, if \mathbb{K} is not \mathbb{R} or \mathbb{C} .)

⁴Finite fields which play a crucial role in cryptanalysis, for example, have no identity principle: the polynomial $X^q - X$, different from the polynomial-zero, evaluates to 0 on its splitting field.

A.4.8. Theorem (fundamental theorem of algebra). A polynomial p over \mathbb{C} has at least one root, i.e., there is at least one $z \in \mathbb{C}$ such that $p(z) = 0$.

A.4.9. Remark (\mathbb{R} is not algebraically closed). The fundamental theorem of algebra is sometimes stated by saying that

the field \mathbb{C} of complex numbers is algebraically closed

where by *algebraically closed field* we mean one where any polynomial has at least one root. The field \mathbb{R} is not algebraically closed: the polynomial $p(X) = X^2 + 1$ has no roots in \mathbb{R} .

A.5. Linear maps

Let V, W be two \mathbb{K} -vector spaces.

A.5.1. Linear maps. A *linear map* from V to W is a mapping $\mathcal{A} : V \rightarrow W$ such that

$$\mathcal{A}(\alpha x + \beta y) = \alpha \mathcal{A}x + \beta \mathcal{A}y \quad \forall x, y \in V. \quad (\text{A.5.1})$$

If \mathcal{A} is a linear map, it is customary to write $\mathcal{A}v$ for $\mathcal{A}(v)$. It is often useful to consider the set of all linear maps from V to W which we denote as $\text{Lin}(V \rightarrow W)$. Sum and scaling of W -valued maps (linear or not) can be defined as follows

$$[\alpha \mathcal{A} + \beta \mathcal{B}]v := \alpha \mathcal{A}v + \beta \mathcal{B}v \text{ for } v \in V. \quad (\text{A.5.2})$$

These operations turn the space $\text{Lin}(V \rightarrow W)$ into a \mathbb{K} -vector space. [*] A *linear operator* on a \mathbb{K} -vector space is a linear map with target space equal to the source space $\text{Lin}(V \rightarrow V)$. A *linear form* on V is a linear map from V to \mathbb{K} , i.e., when the target space is the field of scalars. [*]: Check!

A.5.2. Theorem. A linear map $\mathcal{A} \in \text{Lin}(V \rightarrow W)$ is completely determined by its values on a basis of V .

Proof We give the proof in the finite dimensional case only. Let $\mathbf{v}^\mathbf{T} = (v^i)_{i=1, \dots, n}$ be a basis of V and assume we know that $\mathcal{A}v^i = y^i$ for each $i = 1, \dots, n$ and a given subset $\{y_i : i = 1, \dots, n\}$ of W , then, if x is a generic element in V , we have $x = \sum_{i=1}^n x_i v_i$ for some $[x_i]_{i=1, \dots, n}$ in \mathbb{K}^n and thus, by linearity of \mathcal{A} , we obtain

$$\mathcal{A}x = \sum_{i=1}^n x_i y_i. \quad (\text{A.5.3})$$

This means that all the values of \mathcal{A} can be recovered by the knowledge of its effect on the basis $\mathbf{v}^\mathbf{T}$. □

A.5.3. Duality. The linear space of all linear forms on V , $\text{Lin}(V \rightarrow \mathbb{K})$, is called the (algebraic) *dual space* of V and is denoted as V^* . Given a finite dimensional vector space V and a basis of it, $\mathbf{v}^\mathbf{T} = (v^i)_{i=1, \dots, n}$, we may define its associated *dual basis* as the ordered subset $\mathbf{v}^{\mathbf{T}*}$ of V^* , with elements $v_i^{\mathbf{v}^\mathbf{T}}$ by

$$v_i^{\mathbf{v}^\mathbf{T}}(v^j) = \delta_i^j. \quad (\text{A.5.4})$$

When the basis $\mathbf{v}^\mathbf{T}$ is known we omit the superscript and write simply v_i for $v_i^{\mathbf{v}^\mathbf{T}}$.

A.5.4. Example. If $V = \mathbb{K}^n$, then the dual basis of the natural basis is the set of *co-ordinate maps*:

$$e_i(\mathbf{x}) = x_i \text{ for each } \mathbf{x} = [x_j]_{j=1,\dots,n} \in \mathbb{K}^n \text{ and } i = 1, \dots, n. \quad (\text{A.5.5})$$

Hence e_i is simply the linear form that returns the i -th component, or coordinate, of a vector $\mathbf{x} \in \mathbb{K}^n$. It is immediate that

$$e_i(\mathbf{x}) = \mathbf{e}_i \mathbf{x}, \text{ where } e_i^j = \delta_i^j. \quad (\text{A.5.6})$$

A.5.5. Theorem (duality). *The dual V^* of a finite dimensional \mathbb{K} -vector space V is itself a finite vector space and*

$$\dim V^* = \dim V. \quad (\text{A.5.7})$$

Moreover given a basis $\mathbf{v}^\mathbf{r} = (v^i)_{i=1,\dots,n}$ of V , its dual basis $\mathbf{v}^{\mathbf{r}*} = (v^i)_{i=1,\dots,n}$, as defined in §A.5.3 is actually a basis of V^* (which justifies the terminology).

Proof All we need to show is that $\mathbf{v}^{\mathbf{r}*}$ is a basis of V^* . □

A.5.6. Theorem (matrix representation of linear maps). *If V and W are finite dimensional and respective bases thereof $\mathbf{v}^\mathbf{r} = (v^i)_{i=1,\dots,m}$ and $\mathbf{w}^\mathbf{r} = (w^i)_{i=1,\dots,n}$, then there is a one-to-one correspondence ι between $\text{Lin}(V \rightarrow W)$ and $\mathbb{K}^{n \times m}$, such that for each \mathcal{A} and $\mathbf{A} := \iota \mathcal{A}$ we have*

$$\mathbf{A} = [a_i^j]_{i=1,\dots,n}^{j=1,\dots,m} \text{ and } a_i^j = w^i(\mathcal{A} v_j) \quad (\text{A.5.8})$$

for $i = 1, \dots, n, j = 1, \dots, m$.

where $\{w^i : i = 1, \dots, n\} =: \mathbf{w}^{\mathbf{r}*}$ is the dual basis of $\mathbf{w}^\mathbf{r}$ as defined in §A.5.3.

A.5.7. Example. Consider the space \mathbb{P}^n of polynomials of degree n with real coefficients. A basis for \mathbb{P}^n is given by $\phi_i, i = 0, \dots, n$, where $\phi_i(x) = x^i$. Hence

$$\dim \mathbb{P}^n = n + 1 \quad (\text{A.5.9})$$

and a polynomial $p \in \mathbb{P}^n$ can be written as

$$p = \sum_{i=1}^n p_i \phi_i \quad (\text{i.e., } p(x) = p_0 + p_1 x + \dots + p_n x^n) \quad (\text{A.5.10})$$

for a unique $\mathbf{p} = (p_i)_{i=0,\dots,n} \in \mathbb{R}^{n+1}$. The dual basis of $\{\phi_i : i = 0, \dots, n\}$ is given by the “ i -th coefficient extractor”

$$\phi_i(p) = p_u i. \quad (\text{A.5.11})$$

Examples of linear forms on \mathbb{P}^n are

(a) Evaluation at a given point $x_0 \in \mathbb{R}$

$$\delta_{x_0} : \mathbb{P}^n \ni p \mapsto \delta_{x_0} p := p(x_0) \in \mathbb{R}. \quad (\text{A.5.12})$$

This operator is known also as the *Dirac mass*, or *Dirac delta*. The row-vector $\mathbf{d}_{x_0}^\mathbf{r}$ representing δ_{x_0} , say for $n = 4$, is thus given by

$$\mathbf{d}_{x_0}^\mathbf{r} = [1 \quad x_0 \quad x_0^2 \quad x_0^3 \quad x_0^4]. \quad (\text{A.5.13})$$

(b) n -th derivative

$$D^n : \mathbb{P}^n \ni p \mapsto p^{(n)} \in \mathbb{R} \quad (\text{A.5.14})$$

The row-vector representing D^4 in the monomial basis of \mathbb{P}^4 is

$$[0 \ 0 \ 0 \ 0 \ 1] \quad (\text{A.5.15})$$

(c) Definite integral over $[0, 1]$

$$I : \mathbb{P}^n \ni p \mapsto \int_0^1 p(x) dx. \quad (\text{A.5.16})$$

The row-vector representing I , say for $n = 3$, is

$$[1 \ 1/2 \ 1/3 \ 1/4] \quad (\text{A.5.17})$$

Consider now the first derivative map

$$\begin{aligned} D : \mathbb{P}^n &\rightarrow \mathbb{P}^{n-1} \\ p &\mapsto Dp = p' \end{aligned} \quad (\text{A.5.18})$$

$$\text{where } Dp(x) := p'(x) := \frac{dp(x)}{dx} \text{ for all } x \in \mathbb{R}.$$

From basic calculus we know that D is linear. [*] Let us find the matrix \mathbf{D} representing D with respect to the monomial basis $\{\phi_i : i = 0, \dots, n\}$. Writing $\mathbf{D} = [d_i^j]_{i=1, \dots, n}^{j=1, \dots, n-1}$ we have, for $i, j = 0, \dots, n$:

$$d_i^j = \phi^i(D\phi_j) = \phi^i(j\phi_{j-1}) = j\delta_i^{j-1}. \quad (\text{A.5.19})$$

That is, in extended form, for $n = 5$ say

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}. \quad (\text{A.5.20})$$

A.5.8. Adjoint operator. We will see later adjoint operators in a Hilbertian setting. In general, when no Hilbert structure is available the adjoint \mathcal{A}^* of $\mathcal{A} \in \text{Lin}(V \rightarrow W)$ is defined as the unique

$$\begin{aligned} \mathcal{A}^* : W^* &\rightarrow V^* \\ \psi &\mapsto \phi \end{aligned} \quad \text{where } \langle \phi | v \rangle_V = \langle \psi | \mathcal{A}v \rangle_W \quad \forall v \in V. \quad (\text{A.5.21})$$

We then note that this is indeed a linear operator and it is unique.

A.5.9. Invertibility. A (not necessarily linear) map $\mathcal{A} : V \rightarrow W$ is called⁵

★ *left invertible* or *postinvertible* if and only if there exists a map $\mathcal{B} : W \rightarrow V$ such that

$$\mathcal{B}\mathcal{A} = \text{id}_V, \quad (\text{A.5.22})$$

the identity on V . Such a \mathcal{B} is called a *left inverse* or *postinverse*.

⁵The prefixes “pre” and “post” refer to composition rules, while “left” and “right” refer to typographical rules. These are the opposite. Think of $g \circ f$: notation means that f is applied first and then g , but we write g first. The mnemoning is that “the (compositional) postinverse (typographically) premultiplies and the preinverse postmultiplies”.

- ★ *right invertible* or *preinvertible* if and only if there exists a map $\mathcal{B} : W \rightarrow V$ such that

$$\mathcal{A} \mathcal{B} = \text{id}_W. \quad (\text{A.5.23})$$

- ★ A map is invertible if and only if it is simultaneously postinvertible and preinvertible.

Recall the basic facts that the inverse, when it exists, is unique and coincides with the postinverse and preinverse. If one makes the assumption that \mathcal{A} is linear, then its inverse is linear (see A.5.10).

A.5.10. Problem (invertible maps and linearity). Consider a map $\mathcal{A} : V \rightarrow W$.

- Show that \mathcal{A} is postinvertible if and only if it is injective.
- Preinvertible if and only if it is surjective.
- Deduce that \mathcal{A} is invertible if and only if it is bijective.
- The map \mathcal{A} is invertible if and only if its postinverse exists and is unique.
- The map \mathcal{A} is invertible if and only if its preinverse exists and is unique.
- Show that the postinverse \mathcal{B} of a postinvertible linear map \mathcal{A} is linear.

A.6. Spectral theory

In this section we review the basics about eigenvalues, eigenvectors, eigenspaces, eigenetc. This is also known as “spectral theory”.

A.6.1. Complexification of a linear operator. Let V be a \mathbb{R} -vector space and $W := V \oplus iV$ its complexification, as introduced in §A.2.5. Suppose \mathcal{A} is a linear operator on V . Then \mathcal{A} can be extended in a unique way to an operator $\mathcal{A}_{\mathbb{C}}$ on W by defining, for $\mathbf{z} = (x, y) \in W$

$$\mathcal{A}_{\mathbb{C}} \begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} \mathcal{A}x \\ \mathcal{A}y \end{bmatrix}. \quad (\text{A.6.1})$$

This is so straightforward (in fact, it is a natural generalisation of the product of a real number by a complex one) that the subindex \mathbb{C} is usually dropped.

A.6.2. Eigenstuff. Given an operator $\mathcal{A} \in \text{Lin}(V \rightarrow V)$, we say that $\lambda \in \mathbb{C}$ is an *eigenvalue* of \mathcal{A} if there exists a $v \in V$ such that $v \neq 0$ and

$$\mathcal{A}v = \lambda v. \quad (\text{A.6.2})$$

Any vector v satisfying the *eigenequation* (A.6.2) is called an *eigenvector* of \mathcal{A} associated with λ . A pair (λ, v) satisfying (A.6.2) is called an *eigenpair*. Similarly, given a square matrix $A \in \mathbb{K}^{n \times n}$ and thinking of it as being an operator on \mathbb{K}^n , we use the same terminology for $\lambda \in \mathbb{C}$ and $\mathbf{v} \in \mathbb{C}^n$ in that we say that λ and \mathbf{v} are an eigenvalue and an eigenvector, respectively, or that (λ, \mathbf{v}) is an eigenpair of the matrix A if and only if

$$\mathbf{v} \neq 0 \text{ and } A\mathbf{v} = \lambda \mathbf{v}. \quad (\text{A.6.3})$$

The set of all eigenvalues is called the *point spectrum* of the operator \mathcal{A} .

A.6.3. Spectral values and spectrum. The concept of spectral value (also known as singular value) generalises that of eigenvalue. A *spectral value* of $\mathcal{A} \in \text{Lin}(V \rightarrow V)$ is any λ for which $\mathcal{A} - \lambda$ is singular (not invertible). An eigenvalue is necessarily a spectral value. If V is finite dimensional then a spectral value must be an eigenvalue, but if $\dim V = \infty$, then there are examples of operators with spectral values that are not eigenvalues. The set of all spectral values is called *spectrum*.

A.6.4. Example (eigenvectors of the derivative operator). Consider the operator

$$\begin{aligned} D: C^\infty(\mathbb{R}) &\rightarrow C^\infty(\mathbb{R}) \\ \phi &\mapsto D\phi := \phi' \end{aligned} \quad (\text{A.6.4})$$

where

$$\phi'(x) := \frac{d}{dx} \phi(x). \quad (\text{A.6.5})$$

It is then a matter of simple calculus to check that any $\lambda \in \mathbb{R}$ is an eigenvalue for the operator D , with associated eigenvector the function $x \mapsto \exp(\lambda x)$. This means that the whole real line is contained in the spectrum of the operator D .

A.6.5. Theorem (spectral invariance under basis transformation). *If V is finite dimensional, with a basis $\mathbf{v}^\mathbf{T}$, $\mathcal{A} \in \text{Lin}(V \rightarrow V)$. Then λ is an eigenvalue of \mathcal{A} if and only if λ is an eigenvalue of \mathbf{A} , the matrix representing \mathcal{A} with respect to $\mathbf{v}^\mathbf{T}$.*

Proof If λ is an eigenvalue of \mathcal{A} , then $\lambda x = \mathcal{A}x$ for some $x \in V$. For \mathbf{x} such that $x = \mathbf{v}^\mathbf{T} \mathbf{x}$ we have, for any $i = 1, \dots, n$,

$$v^i(\mathcal{A}x) = v^i\left(\mathcal{A} \sum_{j=1}^n v_j x^j\right) = \sum_{j=1}^n v^i(\mathcal{A} v_j) x^j = \sum_{j=1}^n a_i^j x^j = [\mathbf{A}\mathbf{x}]_i. \quad (\text{A.6.6})$$

On the other hand, for the same $i = 1, \dots, n$,

$$v^i(\mathcal{A}x) = v^i(\lambda x) = [\lambda \mathbf{x}]_i. \quad (\text{A.6.7})$$

Since i is arbitrary it follows that

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}. \quad (\text{A.6.8})$$

□

A.6.6. Corollary (spectral invariance under similarity transformation). *Let $\mathbf{S} \in \mathbb{K}^{n \times n}$ be an invertible matrix and $\mathbf{A} \in \mathbb{K}^{n \times n}$ a generic matrix. Then the spectrum of \mathbf{A} and that of $\mathbf{S}^{-1} \mathbf{A} \mathbf{S}$ coincide.*

Proof The short proof consists in thinking of \mathbf{A} and $\mathbf{B} := \mathbf{S}^{-1} \mathbf{A} \mathbf{S}$ to represent the same linear operator \mathcal{A} in two different bases, the canonical one and

$$\mathbf{S} := (\mathbf{s}^j)_{j=1, \dots, n} = [\mathbf{s}^1 \quad \dots \quad \mathbf{s}^n]. \quad (\text{A.6.9})$$

We then use the fact that a $\lambda \in \mathbb{C}$ is an eigenvalue of \mathcal{A} if and only if it is an eigenvalue of \mathbf{A} (and \mathbf{B}). A longer proof consists in messing around with characteristic polynomials and determinants: we leave it as an exercise for determinant lovers. □

A.7. Multilinear forms

A useful generalisation of linear forms is that of multilinear forms. Two important applications of multilinear forms are *inner products* and *determinants*. If you are already trained in linear algebra, advanced calculus and a bit of differential geometry you should consult Abraham, Marsden and Ratiu, 1988 which is an excellent advanced treatment of abstract multilinear algebra.

A.7.1. Definition of multilinear forms on a space. A *multilinear form* (also known as *k-linear*), on a \mathbb{K} -vector space V is a function $a : V \times \cdots \times V \rightarrow \mathbb{K}$ that is linear in each of its arguments separately, i.e., such that

$$a(x_1, \dots, \lambda x_i + \mu y_i, \dots, x_k) = \lambda a(x_1, \dots, x_i, \dots, x_k) + \mu a(x_1, \dots, y_i, \dots, x_k) \\ \forall x_1, \dots, x_k, y_i \in V, \lambda, \mu \in \mathbb{K} \text{ and } i = 1, \dots, k. \quad (\text{A.7.1})$$

Two important special cases of *k-linear* are *linear forms*, when $k = 1$, and *bilinear forms*, when $k = 2$. Also, an important case is that of *d-linear forms* when the dimension of the space V is d .

A.7.2. Definition of symmetric and antisymmetric forms. A *k-linear form* a on V is called *symmetric* if and only if

$$a(x_1, \dots, x_i, \dots, x_j, \dots, x_k) = a(x_1, \dots, x_j, \dots, x_i, \dots, x_k) \quad (\text{A.7.2})$$

or *antisymmetric* if and only if

$$a(x_1, \dots, x_i, \dots, x_j, \dots, x_k) = -a(x_1, \dots, x_j, \dots, x_i, \dots, x_k). \quad (\text{A.7.3})$$

A.7.3. Definition of alternating form. A *k-linear form* a on V is called *alternating* if and only if for any linearly dependent system of k vectors, x_1, \dots, x_k , we have $a(x_1, \dots, x_k) = 0$.

A.7.4. Problem (alternating and antisymmetric forms are synonymous). Show that a *k-linear form* a on a vector space V is alternating if and only if it is antisymmetric.

A.7.5. Determinant of a linear operator on a finite dimensional space. Following Abraham, Marsden and Ratiu (1988, Ch. 7) it is possible to define the determinant of a linear operator \mathcal{A} on a finite dimensional vector space V with alternating *d-forms* ($d = \dim V$) in an “absolute” way, i.e., without using the concept of norm, inner product or matrix representation of \mathcal{A} . This means that the determinant is an intrinsic property of the linear operator, like eigenvalues, that does not depend on which matrix representation is used to describe the operator.

The *pull-back* by \mathcal{A} of a given *k-linear form* α is the *k-linear form* β defined by

$$\beta(x_1, \dots, x_k) = \alpha(\mathcal{A}x_1, \dots, \mathcal{A}x_k) \quad \forall x_1, \dots, x_k \in V; \quad (\text{A.7.4})$$

we denote the unique such β with $\mathcal{A} \triangleleft \alpha$.

A.7.6. Definition of sesquilinear form. When $\mathbb{K} = \mathbb{C}$ a closely related concept to bilinear form is that of *sesquilinear form*, which is a \mathbb{K} -valued function of two variables, say $a : V \times V \rightarrow \mathbb{K}$, such that

$$a(x, \lambda y + \mu z) = \lambda a(x, y) + \mu a(x, z) \quad (\text{A.7.5})$$

$$a(\lambda x + \mu y, z) = \bar{\lambda} a(x, z) + \bar{\mu} a(y, z) \quad (\text{A.7.6})$$

for any $\lambda, \mu \in \mathbb{K}$ and $x, y, z \in V$.

A.7.7. Remark (real and sesquilinear is same as bilinear). If $\mathbb{K} = \mathbb{R}$ then a is bilinear if and only if a is sesquilinear. So the terminology sesquilinear is really useful only when $\mathbb{K} = \mathbb{C}$.

A.8. Norms and normed vector spaces

Informally a norm on a vector space is the realisation of the concept of length on a vector space. In fact, a norm turns out to be more general than the Euclidean length, which is but an example of norm. Normed vector spaces play a central role in Analysis, the analysis of differential equations and numerical analysis.⁶ Their importance derives from the fact that once equipped with a norm, we are able to talk about the convergence of sequences and the continuity of functions. This opens the doors of differential and (partially) integral calculus.

A.8.1. Definition of norm. A *norm* on a vector space V is a function

$$\begin{aligned} \|\cdot\| : V &\rightarrow \mathbb{R} \\ x &\mapsto |x|, \end{aligned} \quad (\text{A.8.1})$$

such that

(1) $\|\cdot\|$ is positive definite, i.e.,

$$\|x\| \geq 0 \text{ and } \|x\| = 0 \Leftrightarrow x = 0, \quad (\text{A.8.2})$$

(2) $\|\cdot\|$ is homogeneous (of degree 1), i.e.,

$$\|\lambda x\| = |\lambda| \|x\| \quad \forall \lambda \in \mathbb{K}, x \in V, \quad (\text{A.8.3})$$

(3) $\|\cdot\|$ satisfies the triangle inequality, i.e.,

$$\|x + y\| \leq \|x\| + \|y\|. \quad (\text{A.8.4})$$

A.8.2. Definition of convergent sequence. Let $(V, \|\cdot\|_V)$ be a normed \mathbb{K} -vector space. A sequence $(x_n)_{n \in \mathbb{N}}$ is *convergent* if and only if for a given $x \in V$

$$\lim_{n \rightarrow \infty} \|x_n - x\|_V = 0. \quad (\text{A.8.5})$$

Equivalently, $(x_n)_{n \in \mathbb{N}}$ converges to x if and only if

$$\forall \varepsilon > 0 : \exists N \in \mathbb{N} : n \geq N \Rightarrow \|x_n - x\| < \varepsilon. \quad (\text{A.8.6})$$

A.8.3. Proposition (uniqueness of the limit). Suppose a sequence $(x_n)_n$ converges to a point x and to a point y , then $x = y$.

⁶In fact, this section belongs more to an Analysis synopsis rather than a Linear Algebra one, but I put it here for completeness.

A.8.4. Definition of limit of a sequence. The unique point to which a sequence converges is called the *limit* of the sequence and is denoted by

$$\lim_{n \rightarrow \infty} x_n. \quad (\text{A.8.7})$$

Thanks to uniqueness we can use the “=”, “≤”, “≥” and other operators applying in the vector space when manipulating limits.

A.8.5. Theorem (vector algebra of limits). *If $(V, \|\cdot\|_V)$ is a normed \mathbb{K} -vector space, and given $(a_n)_n, (b_n)_n$ convergent in \mathbb{K} , and $(x_n)_n, (y_n)_n$ convergent in V , then $(a_n x_n + b_n y_n)_{n \in \mathbb{N}}$ converges and*

$$\lim_{n \rightarrow \infty} [a_n x_n + b_n y_n] = \lim_{n \rightarrow \infty} a_n \lim_{n \rightarrow \infty} x_n + \lim_{n \rightarrow \infty} b_n \lim_{n \rightarrow \infty} y_n. \quad (\text{A.8.8})$$

A.9. Inner (or scalar) products and Hilbert spaces

Euclidean geometry enjoys more special properties than just covering the concept of “length”: a central role is played by angles. In particular orthogonality.

A.9.1. Definition of inner product (space). A vector space V is called an *inner product space* if and only if V is endowed with an *inner product*, i.e., a map

$$\begin{aligned} \langle \cdot, \cdot \rangle : V \times V &\rightarrow \mathbb{K} \\ (v, w) &\mapsto \langle v, w \rangle \end{aligned} \quad (\text{A.9.1})$$

which satisfies

(i) linearity in the second argument:

$$\langle v, \lambda w + \mu x \rangle = \lambda \langle v, w \rangle + \mu \langle v, x \rangle; \quad (\text{A.9.2})$$

(ii) Hermitianity:

$$\langle v, w \rangle = \overline{\langle w, v \rangle} \quad \forall v, w \in V; \quad (\text{A.9.3})$$

(iii) positivity:

$$\langle v, v \rangle \geq 0 \quad \forall v \in V; \quad (\text{A.9.4})$$

(iv) definiteness:

$$\langle v, v \rangle = 0 \Rightarrow v = 0. \quad (\text{A.9.5})$$

A.9.2. Remark. Note that i and ii imply that an inner-product is sesquilinear:

$$\langle \lambda v + \mu w, x \rangle = \overline{\langle x, \lambda v + \mu w \rangle} = \overline{\lambda \langle x, v \rangle + \mu \langle x, w \rangle} = \overline{\lambda} \overline{\langle x, v \rangle} + \overline{\mu} \overline{\langle x, w \rangle} = \overline{\lambda} \langle v, x \rangle + \overline{\mu} \langle w, x \rangle \quad (\text{A.9.6})$$

A.9.3. Hilbertian norms. On an inner product space $(V, \langle \cdot, \cdot \rangle)$ a norm is given by the function $\|\cdot\|$ defined

$$\|x\| \geq 0 \text{ and } \|x\|^2 = \langle x, x \rangle. \quad (\text{A.9.7})$$

Whenever a norm is given as the square root of an inner-product we call it a *Hilbertian norm*.

A.9.4. The Euclidean norm. A special case of a Hilbertian norm, is that of Euclidean norm. When working in \mathbb{R}^n or \mathbb{C}^n with the canonical inner product

$$(\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}^* \mathbf{y}, \quad (\text{A.9.8})$$

the corresponding norm is called the *Euclidean norm* and is denoted by

$$|\mathbf{x}|_2 \text{ or } |\mathbf{x}|. \quad (\text{A.9.9})$$

A.9.5. Definition of Hilbert space. An inner product space $(V, \langle \cdot, \cdot \rangle)$ is said to be a *Hilbert space* if and only if V is complete with respect to the distance

$$d(v, w) := \sqrt{\langle v - w, v - w \rangle}. \quad (\text{A.9.10})$$

A.9.6. Definition of orthonormal system. Consider an inner product \mathbb{K} -vector space V . A subset $\mathcal{O} \subseteq V$ forms an orthonormal system in if and only if

$$\langle v, w \rangle = \delta_v^w = \begin{cases} 1 & \text{if } v = w, \\ 0 & \text{if } v \neq w. \end{cases} \quad (\text{A.9.11})$$

A.9.7. Proposition. If V is a finite dimensional inner product \mathbb{K} -vector space then any orthonormal system $\mathcal{O} \subseteq V$ has at most $\dim V$ elements.

A.9.8. Definition of Hilbert (or orthonormal) basis. Let V be a Hilbert space, we say that an orthonormal system \mathcal{V} forms a Hilbert basis (also known as orthonormal basis) of V if and only if each element $x \in V$ can be represented as an absolutely convergent series

$$x = \sum_{v \in \mathcal{V}_x} b^v v \quad (\text{A.9.12})$$

for some sequence $(b_v)_{v \in \mathcal{V}_x}$ where $\mathcal{V}_x \in \mathcal{V}$ and \mathcal{V}_x is countable.

A.9.9. Remark. Noting that any finite sum is an absolutely convergent series the definition above makes sense also when V is finite dimensional.

A.9.10. Remark. Given that the series converges absolutely, order doesn't matter.

A.9.11. Theorem (Hilbert dimension). Let V be a Hilbert space, then any two orthonormal bases are equipotent. I.e., if V is finite dimensional, any orthonormal basis has $\dim V$ elements, if V is infinite dimensional then all orthonormal bases have the same cardinality. We declare this cardinal number to be the Hilbert dimension of the Hilbert space.

Proof Omitted. □

A.9.12. Definition. A Hilbert space V is called *separable Hilbert space* if and only if V has a countable Hilbert dimension (i.e., either finite or numberable).

A.9.13. Proposition. A finite dimensional inner product \mathbb{K} -vector space is a Hilbert space. Its Hilbert dimension coincides with the usual algebraic dimension.

Proof Omitted. □

A.9.14. Theorem (Riesz representation). *Let V be a separable Hilbert space, then ϕ is a (continuous) linear form on V if and only if there exists an $v \in V$ such that*

$$\phi(x) = \langle x, v \rangle. \quad (\text{A.9.13})$$

This creates a one to one correspondence between V and its (topological) dual V' .

A.9.15. Remark (topological irrelevance in finite dimension). The words “continuous” and “topological” are superfluous in the case of *finite dimensional spaces*, because linear maps are continuous therein for any reasonable topology. For this we bracket them out in this Appendix. In the infinite dimensional case, the situation is much trickier and the role of topology becomes essential. This forms the main motivation behind *Functional Analysis* which is beyond the scope of this Appendix.

A.9.16. Theorem (sesquilinear forms and linear operators isomorphism). *Let V be a vector space with inner product $\langle \cdot, \cdot \rangle$.*

(a) *For each (continuous) linear operator $\mathcal{A} \in \text{Lin}(V \rightarrow V)$ the map a defined by*

$$\begin{aligned} a : V \times V &\rightarrow \mathbb{K} \\ (x, y) &\mapsto \langle x, \mathcal{A}y \rangle \end{aligned} \quad , \quad (\text{A.9.14})$$

is a sesquilinear form on V .

(b) *For each (continuous) sesquilinear form a on V , there exists a linear operator $\mathcal{A} : V \rightarrow V$ such that*

$$a(x, y) = \langle x, \mathcal{A}y \rangle. \quad (\text{A.9.15})$$

Thus the space of (continuous) bilinear forms on V and that of (continuous) linear operators on V are isomorphic.

Proof

(a) Suppose $\mathcal{A} \in \text{Lin}(V \rightarrow V)$ is given and a defined by

□

A.9.17. Adjoint operators. Given an operator $\mathcal{A} : V \rightarrow V$ on a Hilbert space $(V, \langle \cdot, \cdot \rangle)$ we define the (Hilbert–Riesz) *adjoint operator* of \mathcal{A} as the operator \mathcal{B} such that

$$\langle \mathcal{B}v, w \rangle = \langle v, \mathcal{A}w \rangle \quad \forall v, w \in V. \quad (\text{A.9.16})$$

By Theorem A.9.16 we know that there exists a unique linear operator \mathcal{B} satisfying (A.9.16), so this definition is fully justified.

Exercises and problems on Linear algebra

Problem A.1. Consider a map $\mathcal{A} : V \rightarrow W$.

- Show that \mathcal{A} is postinvertible if and only if it is injective.
- Preinvertible if and only if it is surjective.
- Deduce that \mathcal{A} is invertible if and only if it is bijective.
- The map \mathcal{A} is invertible if and only if its postinverse exists and is unique.
- The map \mathcal{A} is invertible if and only if its preinverse exists and is unique.
- Show that the postinverse \mathcal{B} of a postinvertible *linear map* \mathcal{A} is linear.

Problem A.2. Show that a k -linear form a on a vector space V is alternating if and only if it is antisymmetric.

APPENDIX B

Analysis

Warning: This is not a complete treatment of Analysis or Calculus, but just a collection of useful facts, used as "reminders" for some other course. In particular, very few proofs and problems are provided, so the pedagogic value of this chapter (or appendix) alone is limited. A good reference for a proper treatment the material contained herein is Fleming, 1977.

B.1. Sequences and convergence in \mathbb{R}

B.1.1. Definition of sequence. A *sequence* in a set X is a function whose domain is \mathbb{N} , or \mathbb{N}_0 , or a subset thereof and whose codomain is X . In the context of sequence the variable is called *index* and the value *term*. It is customary to denote the terms of sequences with the subscript (or superscript) notation rather than the bracket. In symbols, we say that $(x_n)_{n \in \mathbb{N}}$ is a sequence in X if $x_n \in X$ for each $n \in \mathbb{N}$.

B.1.2. Definition of convergent sequence of real numbers. A sequence $(x_n)_{n \in \mathbb{N}}$ in \mathbb{R} is said to *converge to* a number $x \in \mathbb{R}$ if and only if

$$\forall \epsilon > 0: \exists N \in \mathbb{N}: n \geq N \Rightarrow |x - x_n| < \epsilon. \quad (\text{B.1.1})$$

A sequence of real numbers may fail to converge, but if it does there is only one number it converges to: this is called *uniqueness of the limit*. This point is denoted $\lim_{n \rightarrow \infty} x_n$. Indeed, supposing a given sequence $(x_n)_{n \in \mathbb{N}}$ converges to both \hat{x} and x . Then by the triangle inequality we have that, *for all* $n \in \mathbb{N}$,

$$|\hat{x} - x| \leq |\hat{x} - x_n| + |x_n - \bar{n}|. \quad (\text{B.1.2})$$

Therefore for any given ϵ there exists \hat{N} and \bar{N} such that

$$n \geq \hat{N} \vee \bar{N} \Rightarrow |\hat{x} - x_n| \vee |x - x_n| \leq \frac{\epsilon}{2}. \quad (\text{B.1.3})$$

It follows from (B.1.2) and by choosing appropriately n that

$$|\hat{x} - x| \leq \epsilon. \quad (\text{B.1.4})$$

(Notice how the last inequality has no n in it!) Because $\epsilon > 0$ is arbitrary, it follows that $|\hat{x} - x|$ is smaller than any positive number, and, since it is an absolute value, hence is 0. Which means that \hat{x} and x are one and the same. Hence, when it does exist, $\lim_{n \rightarrow \infty} x_n$ is unique and the equal sign can safely be used to write:

$$\lim_{n \rightarrow \infty} x_n = x. \quad (\text{B.1.5})$$

B.1.3. Facts about convergent sequences. A convergent sequence is necessarily bounded. But not viceversa: a bounded sequence may not converge.

A convergent sequence $(x_n)_{n \in \mathbb{N}}$ has a unique *cluster point*, namely its limit, $\lim_{n \rightarrow \infty} x_n$.

B.1.4. Remark. A sequence $(x_n)_{n \in \mathbb{N}}$ that fails to converge is said to *diverge*. A sequence can diverge in many different ways:

- (a) $(x_n)_{n \in \mathbb{N}}$ may never settle around any value, e.g.,
- (b) $(x_n)_{n \in \mathbb{N}}$ may be unbounded (while a convergent sequence is necessarily bounded)

B.1.5. Theorem (Cauchy criterion). A sequence $(x_n)_{n \in \mathbb{N}}$ in \mathbb{R} is convergent if and only if it satisfies the Cauchy criterion

$$\forall \epsilon > 0 : \exists N \in \mathbb{N} : n \geq N, k \in \mathbb{N}_0 \Rightarrow |x_n - x_{n+k}| < \epsilon. \quad (\text{B.1.6})$$

Proof This result is the object of basic analysis or calculus courses and is omitted here. □

B.2. Normed vector spaces (topological linear algebra)

B.2.1. Vector spaces. Familiarity with *vector spaces* (also known as *linear spaces*) over \mathbb{K} is assumed. Examples of vector spaces include \mathbb{K}^d , the set \mathbb{P}^k of polynomials of degree less or equal to $k \in \mathbb{N}_0$ with coefficients in \mathbb{R} (or \mathbb{C}), the set $C^0(\Omega)$ of continuous functions on an open set Ω , the set $C^1(\Omega)$ of continuously differentiable functions on Ω , and the set of analytic functions on Ω . A good reference for the theory of finite-dimensional vector space is Halmos' classic on the subject **Halmos:book:vector:1974**. Linear algebra and topology in infinite dimensional vector spaces goes by the name of *functional analysis* and studies topics like *Hilbert spaces*, *Banach spaces*, *operator theory*, *duality*, *weak convergence*. This subject has libraries full of texts, an excellent example of which, including also measure theory, is Lieb and Loss, 2001.

B.2.2. Definition of norm. The concept of absolute value (or modulus) is central to the technical understanding of limits in \mathbb{R} (or \mathbb{C}). This can be generalised to all vector spaces by introducing the concept of norm.¹ Given a vector space X over \mathbb{K} , a *norm* on X is a function

$$\|\cdot\| : X \rightarrow \mathbb{R} \quad (\text{B.2.1})$$

which satisfies

$$\|x\| \geq 0 \text{ and } \|x\| = 0 \Leftrightarrow x = 0 \quad (\text{positivity}), \quad (\text{B.2.2})$$

$$\|\lambda x\| = |\lambda| \|x\| \quad (\text{homogeneity}), \quad (\text{B.2.3})$$

$$\text{and } \|x + y\| \leq \|x\| + \|y\| \quad (\text{triangle inequality}), \quad (\text{B.2.4})$$

for all $x, y \in X$ and $\lambda \in \mathbb{K}$.

¹Even more general tools for convergence are those of *distance*, *metric* and *topology*. While metrics and distances these may be used from time to time in this course we will not discuss them here by referring to specialised text such as **Friedman:70:book:Foundations**

B.2.3. Common Vector Norms. The most used norm in \mathbb{R}^m is the Euclidean norm

$$|\mathbf{x}| = |\mathbf{x}|_2 := (x_1^2 + x_2^2 + \cdots + x_m^2)^{1/2}. \quad (\text{B.2.5})$$

This norm is the most natural, as $|\mathbf{x} - \mathbf{y}|$ is the usual physical distance between \mathbf{x} and \mathbf{y} ; and because of this it takes the name of *Euclidean norm*. What distinguishes this norm (among all norms on \mathbb{R}^n) is that it can be expressed in terms of the usual *inner (scalar or dot) product*:

$$|\mathbf{x}|^2 = \mathbf{x} \cdot \mathbf{x} \quad (\text{B.2.6})$$

where

$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + \cdots + x_m y_m. \quad (\text{B.2.7})$$

Arguably, the most useful property of this norm, is the Cauchy–Bunyakovski–Schwarz inequality²

$$|\mathbf{x} \cdot \mathbf{y}| \leq |\mathbf{x}| |\mathbf{y}|. \quad (\text{B.2.8})$$

The Euclidean norm can be generalized, to the L_p norm, for any fixed $p \in [1, \infty]$. This norm is usually denoted $|\cdot|_p$, is defined by

$$|\mathbf{x}|_p = (|x_1|^p + |x_2|^p + \cdots + |x_m|^p)^{1/p}. \quad (\text{B.2.9})$$

- ★ Choosing $p = 1$ gives the so-called the *Manhattan taxi-cab norm*. Travelling by taxi in a rectangular grid city (such as downtown Manhattan), the distance to your destination is the number of blocks North plus the number of blocks East, in absolute value.
- ★ Choosing $p = 2$ gives the Euclidean norm. In the example above, the taxi would take us along two sides of a rectangle, while the Euclidean norm is the length of the diagonal.
- ★ Choosing $p = \infty$ gives called the maximum or L^∞ norm. If we take $p \rightarrow \infty$ in the definition we obtain

$$|\mathbf{u}|_\infty = \max_{1 \leq i \leq m} |u_i|. \quad (\text{B.2.10})$$

For $p, q \in [1, \infty]$ with $1/p + 1/q = 1$, we have the *Hölder inequality*

$$|\mathbf{x} \cdot \mathbf{y}| \leq \sum_{i=1}^d |x_i y_i| \leq |\mathbf{x}|_p |\mathbf{y}|_q \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{K}^d. \quad (\text{B.2.11})$$

It follows that the Euclidean norm satisfies the *Cauchy–Bunyakovsky–Schwarz inequality*³

$$|\mathbf{x} \cdot \mathbf{y}| \leq \sum_{i=1}^d |x_i y_i| \leq |\mathbf{x}|_2 |\mathbf{y}|_2 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{K}^d. \quad (\text{B.2.12})$$

²This inequality is more commonly known as Cauchy–Schwarz. The Frenchman Augustin Cauchy was apparently the first one to employ it in 1821, in finite dimensional spaces. The Russian Viktor Bunyakovski, whose supervisor was Cauchy, extended it to cover also functions in 1859. The German mathematician Hermann Schwarz proved and used the inequality only 25 years after Bunyakovski.

³ This inequality is more commonly referred to as Cauchy–Schwarz’s, especially in the Western European and American literature. Augustin Cauchy, the famous French 19th century mathematician, proved the inequality for sums/series of finite/infinite sequences. The Russian mathematician Viktor Y. Bunyakosky, himself a student of Augustin Cauchy, proved the inequality for functions and integrals. The Prussian/German mathematician Hermann Schwarz rediscovered Bunyakovsky’s proof some 30 years later.

B.2.4. Matrix norms. The notion of size of, or distance between, matrixes will also be required, and there are norms for that too. Notice that in this case the geometric idea of norm as being the length of a vector no longer holds; however most properties of vector norms are retained and nothing prevents us from using them to measure the size, and the distance between, matrixes.

Suppose that $\mathbf{A} = (A_{ij})$ and $\mathbf{B} = (B_{ij})$ are matrixes in $\mathbb{R}^{m \times m}$, a *matrix norm* is a function that associates to each matrix a real number and that satisfies the following properties:

- (1) Positivity: $|\mathbf{A}| \geq 0$, and $|\mathbf{A}| = 0 \Rightarrow \mathbf{A} = \mathbf{O}$, the zero matrix.
- (2) Homogeneity: $|\mu \mathbf{A}| = |\mu| |\mathbf{A}|, \forall \mu \in \mathbb{R}$.
- (3) Triangle inequality: $|\mathbf{A} + \mathbf{B}| \leq |\mathbf{A}| + |\mathbf{B}|$.
- (4) Matrix norm inequality: $|\mathbf{AB}| \leq |\mathbf{A}| |\mathbf{B}|$.

The first three properties are the same as those defining a norm in \mathbb{R}^m while the fourth property is typical of matrix norms only, since the matrix product of two vectors in \mathbb{R}^m is not defined.

In spite of having many different matrix norms to choose from, we will consider only the ‘maximum row sum’ norm

$$|\mathbf{A}|_{\max} = \max_{1 \leq i \leq m} \sum_{j=1}^m |A_{ij}|.$$

B.2.5. Exercise. Check that the maximum row sum norm satisfies the properties for a matrix norm.

So, to compute the norm of a matrix, we compute the sum of the moduli of the elements in each row, then take the largest value among the sums.

The maximum row sum norm is “compatible” with the L^∞ norm of \mathbb{R}^m , in that it the following property is satisfied

$$|\mathbf{Ax}|_\infty \leq |\mathbf{A}|_{\max} |\mathbf{x}|_\infty \quad (\text{B.2.13})$$

for all $\mathbf{A} \in \mathbb{R}^{m \times m}$ and $\mathbf{x} \in \mathbb{R}^m$.

Due to this compatibility property, the same symbol $|\cdot|_\infty$ is used for both the vector and the matrix norm.

B.2.6. Definition of operator norm. Given two normed vector spaces $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ and a linear transformation $T : X \rightarrow Y$, we say that T is bounded if and only if

$$\sup_{x \in X \setminus \{0\}} \frac{\|Tx\|_Y}{\|x\|_X} < \infty. \quad (\text{B.2.14})$$

If this happens we define the *operator norm of T induced by the norms of X and Y* as the left-hand side above

$$\|T\|_{X \rightarrow Y} := \sup_{x \in X \setminus \{0\}} \frac{\|Tx\|_Y}{\|x\|_X}. \quad (\text{B.2.15})$$

If $X = Y$ and $\|\cdot\|_X = \|\cdot\|_Y$ then we write $\|T\|_X$ instead of $\|T\|_{X \rightarrow X}$ and when X and Y are clear from the context we omit the subscript and simply write $\|T\|$.

B.2.7. Induced matrix p -norms. There is a general procedure to define matrix norms in $\mathbb{K}^{d \times k}$ based on Definition B.2.6. Indeed, a matrix $A \in \mathbb{K}^{d \times k}$ can be identified with the linear map $T_A: \mathbb{K}^k \rightarrow \mathbb{K}^d$ defined by

$$T_A(\mathbf{y}) := A\mathbf{y} \in \mathbb{K}^d \quad \forall \mathbf{y} \in \mathbb{K}^k. \quad (\text{B.2.16})$$

Since this identification (i.e., a one-to-one surjective map) T is so common, it is customary to write A for both the matrix and the transformation T_A .

Given two norms, one on \mathbb{K}^d and one on \mathbb{K}^k , the *induced matrix norm* of A is the operator norm of A induced by these two normed vector spaces. If the two norms chosen on \mathbb{K}^d and \mathbb{K}^k happen to be both vector p -norms with the same index $p \in [0, \infty]$, we denote the corresponding matrix norm by $|A|_p$ too. Namely, for a matrix $A \in \mathbb{K}^{d \times k}$, we define

$$|A|_p := \sup_{\mathbf{y} \in \mathbb{K}^k \setminus \{0\}} \frac{|A\mathbf{y}|_p}{|\mathbf{y}|_p}, \quad \forall p \in [1, \infty]. \quad (\text{B.2.17})$$

B.2.8. Exercise (matrix norms). Check that this definition makes $|\cdot|_p$ a matrix norm, in the sense of §B.2.4, on $\mathbb{K}^{d \times k}$.

B.2.9. Proposition (matrix p -norm dual inequalities). Let $p, q \in [1, \infty]$ such that $1/p + 1/q = 1$. Given $A \in \mathbb{K}^{d \times k}$, and denoting by A_i the i -th row (vector) of A for $i \in 1:d$, we have that

$$|A|_p \leq \left| (|A_i|_q)_{i=1, \dots, d} \right|_p, \quad (\text{B.2.18})$$

where the norms $|\cdot|_q$ and $|\cdot|_p$ appearing on the right hand side are taken, respectively, on \mathbb{R}^k and \mathbb{R}^d .

Further, for $p = 1$, or ∞ the equality is satisfied (but for $p \in (1, \infty)$ the equality may fail). Namely,

$$\begin{aligned} |A|_1 &= \sum_{i=1}^d \max_{j=1, \dots, k} |A_{ij}| \\ |A|_\infty &= \max_{i=1, \dots, d} \sum_{j=1}^k |A_{ij}| \end{aligned} \quad (\text{B.2.19})$$

A proof of this result is obtained by applying Hölder inequality (B.2.11). A useful result using matrix norms is the following.

B.2.10. Lemma (Neumann's series). Given a matrix $M \in \mathbb{K}^{d \times d}$, such that $|M| < 1$ —where $|\cdot|$ is any matrix (vector-induced) norm—then the inverse of $I - M$ exists and

$$[I - M]^{-1} = \sum_{k=0}^{\infty} M^k. \quad (\text{B.2.20})$$

B.2.11. Norms on vector space in general. We have seen norms on \mathbb{R}^d and \mathbb{C}^d , but various norms can be introduced on many other vector space, even when it is not finite dimensional. An example of vector spaces that are not finite dimensional is

$$C^0(D) := \{f : D \rightarrow \mathbb{K} : f \text{ is continuous on } D\}, \quad (\text{B.2.21})$$

where $D \subseteq \mathbb{R}^d$ and E is some vector space (say \mathbb{R} or \mathbb{R}^d).

To understand what we mean by finite dimensional, we need to define a set of generators (or spanning elements). Given a set of elements $S \subseteq X$, where X is a vector space on \mathbb{K} , we define

$$\text{Span } S := \left\{ \sum_{s \in V} \lambda_s s : V \text{ finite subset of } S \text{ and } (\lambda_s)_{s \in V} \in \mathbb{K}^S \right\}, \quad (\text{B.2.22})$$

in words $\text{Span } S$ is the set of all possible *finite linear combinations* of elements in S , with coefficients in \mathbb{K} . A vector space is called *finite dimensional*, if there exists a finite set G such that $X = \text{Span } G$; such a set is called a set of generators (or a spanning set). If the elements of a spanning set (finite or not) are linearly independent, then the spanning set is called a (*Hamel*) *basis* for the vector space X . It is possible to show, constructively, that each finite dimensional vector space has a basis. (One can prove, though non-constructively by using the Axiom of Choice, that this is true for infinite dimensional spaces too.) In particular, a finite dimensional vector space must have a finite basis. It is then possible to prove that all bases must have the same number of elements, and this is declared to be the *dimension* of the space. See Halmos **Halmos:book:vector:1974** for the details.

Two norms $\|\cdot\|$ and $\|\cdot\|'$ on a vector space X are said to be *equivalent* if there exist constants C_1 and C_2 such that

$$C_1 \|x\| \leq \|x\|' \leq C_2 \|x\| \quad \forall x \in X. \quad (\text{B.2.23})$$

B.2.12. Theorem (equivalence of norms characterises finite dimension). *A vector space X is finite dimensional if and only if any two norms on X are equivalent.*

A useful consequence of this theorem is that for a finite dimensional space X , to prove convergence of a sequence $(x_n)_{n \in \mathbb{N}}$ in X , continuity or Lipschitz continuity of a function $f : X \rightarrow X$ or openness/closedness of a set $B \subseteq X$ we can use any norm we wish. All these properties are independent of which norm we choose, i.e., it is true in all norms on X if and only if it is true for one norm in X .

The same is not true if X is infinite dimensional as we see next.

B.2.13. Examples of infinite dimensional normed vector spaces. Unlike finite dimensional spaces, one sequence may converge in one norm and fail to do so in some other norm, meaning that *norms may not be equivalent on infinite dimensional vector spaces*. As a example, take the space

$$C^0(\overline{\Omega}) := \{f : \Omega \rightarrow \mathbb{K} : f \text{ is continuous at each } x \in \Omega \text{ and } f \text{ is bounded on } \Omega\}. \quad (\text{B.2.24})$$

We can equip X with either $\|\cdot\|_{L_\infty(\Omega)}$ or $\|\cdot\|_{L_2(\Omega)}$, where

$$\|f\|_{L_\infty(\Omega)} := \sup_{x \in \Omega} |f(x)| \quad (\text{B.2.25})$$

$$\|f\|_{L_2(\Omega)} := \sqrt{\int_{\Omega} |f(x)|^2 dx}. \quad (\text{B.2.26})$$

It is a good exercise to check that both define a norm on $C^0(\overline{\Omega})$. It is also possible to show, by using Hölder's inequality for function,

$$\int_{\Omega} |f(x)g(x)| dx = \sup_{\Omega} |f| \int_{\Omega} |g(x)| dx, \quad (\text{B.2.27})$$

that every sequence of functions in X which converges with respect to the $\|\cdot\|_{L_\infty(\Omega)}$ converges with respect to $\|\cdot\|_{L_2(\Omega)}$, but the converse is not true. Take $\Omega = (0, 1)$ and $f_n(x) = n^{1/3} \mathbb{1}_{(0, 1/n)}(x)$, with $\mathbb{1}_D$ denoting the characteristic function of D , for instance.

B.2.14. Function-valued functions. Ordinary differential equations (ODEs) typically involve functions (or fields) of the form

$$\begin{aligned} \mathbf{f}: [0, T] \times D &\rightarrow D \\ (t, \mathbf{y}) &\mapsto \mathbf{f}(t, \mathbf{y}) \end{aligned}, \quad (\text{B.2.28})$$

with $D \subseteq \mathbb{R}^d$ an open domain. Often, we denote by $\mathbf{f}(t)$ the function \mathbf{f} frozen at time t given by

$$\begin{aligned} \mathbf{f}(t): D &\rightarrow D \\ \mathbf{y} &\mapsto \mathbf{f}(t, \mathbf{y}) \end{aligned} \quad (\text{B.2.29})$$

If we then set t free, we could view \mathbf{f} not as a function mapping $[0, T] \times D$ into D but as a function mapping $[0, T]$ into $\text{Map}(D, D)$, the set of all mappings going from D into itself, also known as *operators* on D and denoted by D^D . Many times, we may require some more “regularity” on each single $\mathbf{f}(t)$, as $t \in [0, T]$. For example, we may ask that $\mathbf{f}(t) \in \text{Lip}(D; D)$, i.e., the set of all operators that satisfy a (global) Lipschitz condition on D , or that $\mathbf{f}(t) \in \text{Lip}_{\text{loc}}(D, D)$ which means it is merely locally Lipschitz.

B.2.15. Balls, neighbourhoods and open sets. Let $(X, \|\cdot\|)$ be a normed vector space. For $x \in X$ and $r \in \mathbb{R}^+$, the *open ball*, and respectively the *closed ball, centred at x and radius r* are the sets

$$\begin{aligned} B_x(\|\cdot\|)r &:= \{y \in X : \|x - y\| < r\}, \\ \overline{B}_r^{\|\cdot\|}(x) &:= \{y \in X : \|x - y\| \leq r\}. \end{aligned} \quad (\text{B.2.30})$$

When the norm $\|\cdot\|$ is obvious from the context, the superscript $\|\cdot\|$ is replaced by X , or d when $X = \mathbb{R}^d$, or dropped altogether from the notation. It is a useful exercise to convince oneself that

$$\overline{B_r(x)} = \overline{B}_r(x). \quad (\text{B.2.31})$$

Given a point $x \in X$ and a set $N \subseteq X$, N is said to be a *neighbourhood* of x if there exists $r \in \mathbb{R}^+$ such that $B_r(x) \subseteq N$.

An *open set* in X is defined to be one which contains a neighbourhood of each of its points.

So $G \subseteq X$ is an open set if and only if for each $x \in G$ there exists $r(x) \in \mathbb{R}^+$ such that $B_{r(x)}(x) \subseteq G$.

B.3. Calculus

We review some elements of calculus. There is no pretense to be exhaustive, this is just a list of useful results, some given without proof. These results can be found in standard textbooks in calculus, a classical example is Fleming, 1977.

B.3.1. Differentiable function and its derivative. Let $\mathbf{g} : D \rightarrow \mathbb{R}^k$ be a given function (also known as field) on an open $D \subseteq \mathbb{R}^l$, for some $k \in \mathbb{N}$. We say \mathbf{g} is *Fréchet differentiable* at a point \mathbf{x} if and only there exists a linear map (or matrix if you prefer) $\mathbf{J} : \mathbb{R}^l \rightarrow \mathbb{R}^k$ such that

$$\mathbf{g}(\mathbf{x} + \mathbf{v}) - (\mathbf{g}(\mathbf{x}) + \mathbf{J}\mathbf{v}) = o(|\mathbf{v}|), \quad (\text{B.3.1})$$

i.e.,

$$\lim_{|\mathbf{v}| \rightarrow 0} \frac{\mathbf{g}(\mathbf{x} + \mathbf{v}) - (\mathbf{g}(\mathbf{x}) + \mathbf{J}\mathbf{v})}{|\mathbf{v}|} = \mathbf{0}. \quad (\text{B.3.2})$$

Such a map \mathbf{J} , when it exists, is unique and is denoted as

$$D\mathbf{g}(\mathbf{x}) \text{ or } \mathbf{g}'(\mathbf{x}). \quad (\text{B.3.3})$$

The *directional derivative* of a function \mathbf{g} at a point \mathbf{x} in the direction \mathbf{y} is defined as

$$\lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} (\mathbf{g}(\mathbf{x} + \epsilon\mathbf{y}) - \mathbf{g}(\mathbf{x})), \quad (\text{B.3.4})$$

and is denoted as $\partial_{\mathbf{y}}\mathbf{g}(\mathbf{x})$. If \mathbf{g} is Fréchet-differentiable at \mathbf{x} then for any $\mathbf{y} \in \mathbb{R}^l$ its directional derivative at \mathbf{x} in the direction \mathbf{y} exists and

$$\partial_{\mathbf{y}}\mathbf{g}(\mathbf{x}) = D\mathbf{g}(\mathbf{x})\mathbf{y}. \quad (\text{B.3.5})$$

If the Fréchet derivative exists we always try to use the second notation and reserve the first one for those (rare) occurrences where the directional derivative exists while the Fréchet derivative does not (or is not known to) exist.

If \mathbf{y} happens to be the i -th element \mathbf{e}^i of \mathbb{R}^l 's natural basis the directional derivative of \mathbf{g} at $\mathbf{x} = [x_i]_{i=1,\dots,l} = (x_1, \dots, x_l)$ in \mathbf{y} is called the i -th *partial derivative* and denoted as

$$\frac{\partial \mathbf{g}(x_1, \dots, x_l)}{\partial x_i} \text{ or, more economically, as } \partial_i \mathbf{g}(\mathbf{x}). \quad (\text{B.3.6})$$

The latter notation allows to consistently drop the (\mathbf{x}) without ambiguity, whereas the former, more traditional, may lead to confusion). For this reason we use the second, unless in very specific situations where the traditional notation is superior. If \mathbf{g} is Fréchet differentiable then

$$D\mathbf{g}(\mathbf{x})\mathbf{e}^i = \partial_i \mathbf{g}(\mathbf{x}) = \frac{\partial \mathbf{g}(x_1, \dots, x_l)}{\partial x_i}. \quad (\text{B.3.7})$$

When $l = k = 1$ then the (only) directional derivative's existence implies the Fréchet's derivative. The same occurs when $l = 1$ and $k \geq 1$. But when $l > 1$, there are examples where all directional derivatives may exist at a point, and yet no Fréchet derivative. These are pathological cases that seldom arise in practice.

B.3.2. Fréchet versus Gâteaux. A concept of derivative weaker than Fréchet's is the Gâteaux derivative. By “weaker” we mean that whenever Fréchet derivative exists then Gâteaux derivative does too and the two coincide. A function $\mathbf{g} : D \rightarrow \mathbb{R}^k$ on open $D \subseteq \mathbb{R}^l$ is *Gâteaux differentiable* at a point $\mathbf{x} \in D$ if there exists a linear map $\mathbf{G} : \mathbb{R}^l \rightarrow \mathbb{R}^k$ such that

$$\lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} (\mathbf{g}(\mathbf{x} + \epsilon \mathbf{y}) - \mathbf{g}(\mathbf{x})) = \mathbf{G} \mathbf{y} \quad \forall \mathbf{y} \in \mathbb{R}^l. \quad (\text{B.3.8})$$

Included in the Gâteaux differentiability of \mathbf{g} is the fact that \mathbf{g} has directional derivatives in all directions at \mathbf{x} . In fact, because of (B.3.5) it follows that the Fréchet derivative, provided it exists, coincides always with the Gâteaux derivative. That is why it is redundant to talk about the “Fréchet derivative” and “derivative” alone suffices. You should convince yourself that Fréchet implies Gâteaux.

However, the converse is not true in general: sometimes the Gâteaux derivative exists yet not the Fréchet derivative, as seen by the following standard textbook-example (Andrews and Hopper, 2011, Appendix A):

$$\mathbf{g}(x, y) := \begin{cases} 0 & \text{if } x = 0 \text{ and } y = 0 \\ \frac{x^4 y}{x^6 + y^3} & \text{otherwise.} \end{cases} \quad (\text{B.3.9})$$

In finite dimensional vector spaces these examples are rare (and somewhat artificial) called “pathologies”. In infinite dimensional spaces there are natural examples of Gâteaux differentiable that are nowhere Fréchet differentiable, while this goes beyond the scope of these notes, the interested reader may refer to Lindenstrauss and Preiss, 2003.

B.3.3. Exercise (Gâteaux is weaker than Fréchet).

- (a) Let $\mathbf{f} : D \rightarrow \mathbb{R}^k$ be a function that is Fréchet differentiable at a given point $\mathbf{x} \in D$. Show that \mathbf{f} is continuous at \mathbf{x} .
(b) Show that the function

$$\mathbf{g}(x, y) := \begin{cases} 0 & \text{if } x = 0 \text{ and } y = 0 \\ \frac{x^4 y}{x^6 + y^3} & \text{otherwise.} \end{cases} \quad (\text{B.3.10})$$

has a Gâteaux derivative at $(0, 0)$. Show that \mathbf{g} is discontinuous at $(0, 0)$. Deduce it has no Fréchet derivative thereon.

B.3.4. The Mean Value Theorem in higher dimensions. Fix $\mathbf{x}, \mathbf{y} \in D$ such that the segment with endpoints \mathbf{x} and \mathbf{y} is entirely contained in D . Consider the function $\boldsymbol{\phi} : [a, b] \rightarrow \mathbb{R}^k$ defined by

$$\boldsymbol{\phi}(t) = \mathbf{g}(\mathbf{x} + t(\mathbf{y} - \mathbf{x})). \quad (\text{B.3.11})$$

By the chain rule we have

$$\boldsymbol{\phi}'(t) = \mathbf{D} \mathbf{g}(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))(\mathbf{y} - \mathbf{x}) \quad (\text{B.3.12})$$

where $\mathbf{D} \mathbf{g}$ is the Jacobian (which is a matrix in $\mathbb{R}^{k \times l}$) of \mathbf{g} . Applying the Fundamental Theorem of Calculus to the function $\boldsymbol{\phi}$ we have

$$\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{x}) = \boldsymbol{\phi}(1) - \boldsymbol{\phi}(0) = \int_0^1 \boldsymbol{\phi}'(s) \, ds = \left(\int_0^1 \mathbf{D} \mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x})) \, ds \right) (\mathbf{y} - \mathbf{x}). \quad (\text{B.3.13})$$

This simple relation can be used in many useful ways.

To understand it better, let us focus on the first factor on the right-hand side. First note that the Jacobian matrix can be explicitly written as

$$D\mathbf{g}(\mathbf{z}) = \begin{bmatrix} \partial_1 g_1(\mathbf{z}) & \dots & \partial_j g_1(\mathbf{z}) & \dots & \partial_l g_1(\mathbf{z}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \partial_1 g_i(\mathbf{z}) & \dots & \partial_j g_i(\mathbf{z}) & \dots & \partial_l g_i(\mathbf{z}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \partial_1 g_k(\mathbf{z}) & \dots & \partial_j g_k(\mathbf{z}) & \dots & \partial_l g_k(\mathbf{z}) \end{bmatrix} = [\partial_j g_i(\mathbf{z})]_{i \in 1:k, j \in 1:l} \quad (\text{B.3.14})$$

where $\partial_j g_i(\mathbf{z}) := \partial g_i(\mathbf{z}) / \partial z_j$. The integral appearing in (B.3.13) is thus understood component-wise; i.e.,

$$\int_0^1 D\mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x})) ds = \left[\int_0^1 \partial_j g_i(\mathbf{x} + s(\mathbf{y} - \mathbf{x})) ds \right]_{i \in 1:k, j \in 1:l}. \quad (\text{B.3.15})$$

Taking norms on both sides of (B.3.13) and using the *triangle inequality for integrals*, i.e.,

$$\left| \int_A \mathbf{f}(\mathbf{x}) d\mathbf{x} \right| \leq \int_A |\mathbf{f}(\mathbf{x})| d\mathbf{x}, \text{ for } \mathbf{f} \text{ integrable on } A \in \mathbb{R}^k, \quad (\text{B.3.16})$$

we obtain that

$$|\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{x})| \leq \int_0^1 |D\mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x}))| ds |\mathbf{y} - \mathbf{x}|, \quad (\text{B.3.17})$$

where $|\cdot|$ denotes two given norms on \mathbb{R}^l , \mathbb{R}^k , as well as the induced matrix norm on $\mathbb{R}^{k \times l}$. Since most results are independent of the choice of these norms we do not specify them unless expressly needed. Using the monotonicity of the integral, i.e.,

$$f(\mathbf{x}) \leq g(\mathbf{x}), \forall \mathbf{x} \in A \Rightarrow \int_A f(\mathbf{x}) d\mathbf{x} \leq \int_A g(\mathbf{x}) d\mathbf{x}. \quad (\text{B.3.18})$$

PROPOSITION (Mean Value Theorem). *Suppose $\mathbf{g} : D \rightarrow \mathbb{R}^k$ is a differentiable function and that $\mathbf{x}, \mathbf{y} \in D$ are such that the segment S joining \mathbf{x} to \mathbf{y} lies entirely in D then*

$$|\mathbf{g}(\mathbf{y}) - \mathbf{g}(\mathbf{x})| \leq \sup_S |D\mathbf{g}| |\mathbf{y} - \mathbf{x}|. \quad (\text{B.3.19})$$

B.3.5. Remark (why an inequality?) Inequality (B.3.19) may seem a waste when one applies it to the $\mathbb{R} \rightarrow \mathbb{R}$ case. This is the price we pay for generality. Indeed the case with equality does not hold in general. The problem being that if the target space, \mathbb{R}^k has enough room then one can beat the MVT as stated in basic calculus courses. For example, consider the following function

$$\begin{aligned} \mathbf{g} : [0, 2\pi] &\rightarrow \mathbb{R}^3 \\ t &\mapsto (\cos(t), \sin(t), t) \end{aligned} \quad ; \quad (\text{B.3.20})$$

it is a good exercise for you to check that *there is no $\xi \in (0, 1)$ such that*

$$\mathbf{g}'(\xi) = \frac{\mathbf{g}(2\pi) - \mathbf{g}(0)}{2\pi}. \quad (\text{B.3.21})$$

In fact, the vector $\mathbf{g}'(\xi)$ is not even parallel to the right-hand side, let alone being equal. If $k = 1$, however, for any $l \in \mathbb{N}$ we can give a stronger version of the Mean Value Theorem, as follows.

B.3.6. Proposition (Real-valued Mean Value). Suppose $g : D(\subseteq \mathbb{R}^l) \rightarrow \mathbb{R}$, is of class $C^1(D)$ and $\mathbf{x}, \mathbf{y} \in D$ such that their spanned segment, S , lies entirely in D , then there exists a ξ on this segment such that

$$g(\mathbf{x}) - g(\mathbf{y}) = Dg(\xi)(\mathbf{x} - \mathbf{y}) = \nabla g(\xi) \cdot (\mathbf{x} - \mathbf{y}). \quad (\text{B.3.22})$$

The proof, which is left as an exercise, can be conducted by using (B.3.13) and the following.

B.3.7. Theorem (Mean Value in Integrals). Let $f, g : X \rightarrow \mathbb{R}$ be two integrable functions on $X \subseteq \mathbb{R}^l$, such that $f \in C^0(D)$ and $g \geq 0$. Then there exists a point $\mathbf{x}_0 \in D$ such that

$$f(\mathbf{x}_0) \int_D g(\mathbf{x}) d\mathbf{x} = \int_D f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x}. \quad (\text{B.3.23})$$

B.3.8. Taylor's expansion in arbitrary dimension. Suppose that the field $\mathbf{g} : D \rightarrow \mathbb{R}^k$ is $n + 1$ times differentiable, then, by (B.3.13) we have

$$\mathbf{g}(\mathbf{y}) = \mathbf{g}(\mathbf{x}) + \int_0^1 D\mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x})) ds (\mathbf{y} - \mathbf{x}). \quad (\text{B.3.24})$$

Applying the integration by parts formula

$$\int_0^1 u(s) v'(s) ds = u(1)v(1) - u(0)v(1) - \int_0^1 u'(s) v(s) ds \quad (\text{B.3.25})$$

with the following choices

$$u(s) = D\mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x})) \text{ and } v(s) = (1 - s), \quad (\text{B.3.26})$$

we see that

$$\begin{aligned} \int_0^1 D\mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x})) ds &= - \int_0^1 u(s) v'(s) ds \\ &= - [D\mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x}))]_0^1 + \int_0^1 \frac{d}{ds} D\mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x})) ds \\ &= D\mathbf{g}(\mathbf{x}) \cdot + \left\langle \int_0^1 D^2 \mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x})) (1 - s) ds; \mathbf{y} - \mathbf{x}, \cdot \right\rangle, \end{aligned} \quad (\text{B.3.27})$$

where $\langle D^2 \cdot(\mathbf{z}); \cdot, \cdot \rangle$ is the “second derivative at \mathbf{z} operator” with the following meaning

$$\langle D^2 \mathbf{f}(\mathbf{z}); \mathbf{w}, \mathbf{v} \rangle := \sum_{i,j=1}^l (\partial_{ij} f_1(\mathbf{z}) w_i v_j, \dots, \partial_{ij} f_k(\mathbf{z}) w_i v_j) \quad (\text{B.3.28})$$

for a field $\mathbf{f} : D(\subseteq \mathbb{R}^l) \rightarrow \mathbb{R}^k$ and $\mathbf{w}, \mathbf{v} \in \mathbb{R}^l$. Thus

$$\mathbf{g}(\mathbf{y}) = \mathbf{g}(\mathbf{x}) + D\mathbf{g}(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \left\langle \int_0^1 D^2 \mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x})) (1 - s) ds; \mathbf{y} - \mathbf{x}, \mathbf{y} - \mathbf{x} \right\rangle. \quad (\text{B.3.29})$$

Supposing $n > 2$ and applying the integration by parts again we have

$$\begin{aligned} \left\langle \int_0^1 D^2 \mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x}))(1-s) ds; \cdot, \cdot \right\rangle &= \langle D^2 \mathbf{g}(\mathbf{x}); \cdot, \cdot \rangle \\ &+ \left\langle \int_0^1 D^3 \mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x})) \frac{(1-s)^2}{2} ds; \mathbf{y} - \mathbf{x}, \cdot, \cdot \right\rangle \end{aligned} \quad (\text{B.3.30})$$

and thus (B.3.29) implies

$$\begin{aligned} \mathbf{g}(\mathbf{y}) &= \mathbf{g}(\mathbf{x}) + D \mathbf{g}(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \langle D^2 \mathbf{g}(\mathbf{x}); \mathbf{y} - \mathbf{x}, \mathbf{y} - \mathbf{x} \rangle \\ &+ \frac{1}{2} \left\langle \int_0^1 D^3 \mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x}))(1-s)^2 ds; (\mathbf{y} - \mathbf{x})^3 \right\rangle \end{aligned} \quad (\text{B.3.31})$$

where

$$; (\mathbf{y} - \mathbf{x})^3 \rangle \text{ is short for } ; \mathbf{y} - \mathbf{x}, \mathbf{y} - \mathbf{x}, \mathbf{y} - \mathbf{x} \rangle \quad (\text{B.3.32})$$

and the application of $D^3 \mathbf{g}(\mathbf{z})$ to it means

$$\langle D^3 \mathbf{g}(\mathbf{z}) \langle \mathbf{w} \rangle^3 \rangle = \sum_{i,j,h=1}^l \partial_{ijh} \mathbf{g}(\mathbf{z}) w_i w_j w_h. \quad (\text{B.3.33})$$

Iterating this argument as many times as n allows it we obtain

$$\begin{aligned} \mathbf{g}(\mathbf{y}) &= \mathbf{g}(\mathbf{x}) + D \mathbf{g}(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \sum_{r=2}^n \frac{1}{r!} \langle D^r \mathbf{g}(\mathbf{x}); (\mathbf{y} - \mathbf{x})^r \rangle \\ &+ \frac{1}{n!} \left\langle \int_0^1 D^{n+1} \mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x}))(1-s)^n ds; (\mathbf{y} - \mathbf{x})^{n+1} \right\rangle \\ &= \sum_{r=0}^n \frac{1}{r!} \langle D^r \mathbf{g}(\mathbf{x}); (\mathbf{y} - \mathbf{x})^r \rangle + \frac{1}{n!} \left\langle \int_0^1 D^{n+1} \mathbf{g}(\mathbf{x} + s(\mathbf{y} - \mathbf{x}))(1-s)^n ds; (\mathbf{y} - \mathbf{x})^{n+1} \right\rangle, \end{aligned} \quad (\text{B.3.34})$$

by convening that

$$\begin{aligned} D^0 &= \text{id}, \quad D^1 = D, \quad \langle \mathbf{a}; \mathbf{w}^0 \rangle = \mathbf{a} \in \mathbb{R}^l, \quad \langle \mathbf{A}; \mathbf{w}^1 \rangle = \mathbf{A} \mathbf{w} \text{ for } \mathbf{A} \in \mathbb{R}^{l \times l}, \\ \text{and } \langle D^r \mathbf{g}(\mathbf{z}); \mathbf{w}^r \rangle &= \sum_{1 \leq i_1, \dots, i_r \leq l} \partial_{i_1, \dots, i_r} \mathbf{g}(\mathbf{z}) w_{i_1} \cdots w_{i_r}. \end{aligned} \quad (\text{B.3.35})$$

Identity (B.3.34) goes by the name of *Taylor's Sum Formula of order $n + 1$ with remainder in integral form*. Note that this formula generalises the one-dimensional formula which you are familiar with from calculus. It is interesting that in order to use the formula in higher dimensions the remainder is better written in an integral form rather than in a differential form taught in basic level courses.

B.4. Continuity

Many times we use the fact that a function is continuous to conclude something useful about it. We gather here the most widely used aspects of continuity.

B.4.1. Definition of continuous function. A function $\mathbf{f} : S \subseteq \mathbb{K}^k \rightarrow \mathbb{K}^d$ where S is any subset of \mathbb{K}^k (not necessarily open!) is said to be *continuous at a point* $\mathbf{x} \in S$ if and only if for each $\varepsilon > 0$ there exists a $\delta = \delta_{\mathbf{x}, \varepsilon} > 0$ such that

$$\mathbf{y} \in S \text{ and } |\mathbf{y} - \mathbf{x}| < \delta \Rightarrow |\mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{x})| < \varepsilon. \quad (\text{B.4.1})$$

Note that, if \mathbf{x} is fixed, then the δ will generally change if ε changes. This is sometimes emphasised by writing δ_ε instead of δ . Also, if \mathbf{f} is continuous at two different points, say $\mathbf{x} \neq \mathbf{z}$ in S , then the δ will change with the point under scrutiny, even if the ε is the same, i.e., denoting δ with $\delta_{\varepsilon, \mathbf{x}}$ in (B.4.1), $\delta_{\varepsilon, \mathbf{x}}$ may be different from $\delta_{\varepsilon, \mathbf{z}}$.

If \mathbf{f} is continuous at all points of $R \subseteq S$, we say that \mathbf{f} is *continuous on (the set) R* . If for each ε it is possible to choose δ_ε independently of $\mathbf{x} \in R$, we say that \mathbf{f} is *uniformly continuous on R* .

B.4.2. Theorem (characterisation of continuity at a point via sequences). A function $\mathbf{f} : S \rightarrow \mathbb{K}^d$ is continuous at $\mathbf{x} \in S$ if and only if for each sequence $(x_k)_{k \in \mathbb{N}}$

$$(x_k \rightarrow \mathbf{x}) \Rightarrow (\mathbf{f}(x_k) \rightarrow \mathbf{f}(\mathbf{x})). \quad (\text{B.4.2})$$

Proof This is a standard proof. Note that to prove that sequential continuity implies continuity you need to use (maybe subconsciously) the Axiom of Choice. \square

B.4.3. Theorem (differentiable implies continuous). If a function $\mathbf{f} : \Omega \subseteq \mathbb{K}^k \rightarrow \mathbb{K}^d$ is differentiable at a point $\mathbf{x} \in \Omega$, then \mathbf{f} is continuous at that point.

Thus, if \mathbf{f} is differentiable (at all the points) in Ω , then \mathbf{f} is continuous (at each point) on Ω .

Proof This is a standard result in calculus. \square

B.4.4. Remark (continuous does not imply differentiable). Note that the converse of Theorem B.4.3 is generally false: there are continuous functions, and some widely used ones, that are not differentiable, for example the absolute value function (or more generally the norm) is not differentiable at 0.

B.4.5. Theorem (linear algebra of continuous functions). If $\mathbf{f}, \mathbf{g} : S \rightarrow \mathbb{K}^d$ are continuous and $\lambda, \mu \in \mathbb{K}$, then the linear combination function $\lambda \mathbf{f} + \mu \mathbf{g}$ is continuous. In fancy words, the set of \mathbb{K}^d -valued continuous functions on S forms a \mathbb{K} -linear space (or \mathbb{K} -vector space).

B.4.6. Theorem (continuity is invariant under composition). Suppose \mathbf{f} and \mathbf{g} are composable functions (i.e., $\text{Cod } \mathbf{f} \subseteq \text{Dom } \mathbf{g}$). If \mathbf{f} is continuous on a set S and \mathbf{g} is continuous on $\text{Dom } \mathbf{g}$, then $S \ni \mathbf{x} \mapsto \mathbf{g}(\mathbf{f}(\mathbf{x}))$ is a continuous function on S .

B.4.7. Compactness and the Weierstrass Theorem. An important result (and arguably the most important one) about continuous functions is the Weierstrass compactness theorem. This result is nicely (i.e., compactly) presented using the concept of compact set.

DEFINITION (compact set). A set $X \subseteq \mathbb{K}^d$ is called a closed set, if and only if for each sequence $(\mathbf{x}_k)_{k \in \mathbb{N}}$ and $\mathbf{x} \in \mathbb{K}^d$

$$((\mathbf{x}_k \in X \ \forall k \in \mathbb{N}) \text{ and } \mathbf{x}_k \rightarrow \mathbf{x} \text{ in } \mathbb{K}^d) \Rightarrow \mathbf{x} \in X. \quad (\text{B.4.3})$$

A set $X \in \mathbb{K}^d$ is called bounded set, if and only if there exists a number $r > 0$ such that

$$\mathbf{x} \in X \Rightarrow |\mathbf{x}| < r. \quad (\text{B.4.4})$$

A set $K \in \mathbb{K}^d$ is called compact set if and only if K is closed and bounded.

B.4.8. Theorem (Bolzano–Weierstrass). A set $K \subseteq \mathbb{K}^d$ is compact if and only if each sequence $(\mathbf{x}_k)_{k \in \mathbb{N}}$ such that $\{\mathbf{x}_k\}_{k \in \mathbb{N}} \subseteq K$ has a subsequence $(\mathbf{x}_{k_j})_{j \in \mathbb{N}}$ ($j \mapsto k_j$ strictly increasing) such that

$$\lim_{j \rightarrow \infty} \mathbf{x}_{k_j} \text{ exists and belongs to } K. \quad (\text{B.4.5})$$

B.4.9. Theorem (Weierstrass compactness). If a function $\mathbf{f} : S \subseteq \mathbb{K}^k \rightarrow \mathbb{K}^d$ is continuous, then for each compact set $K \subseteq S$ the image $\mathbf{f}(K)$ is a compact set. In particular, \mathbf{f} is bounded on K , i.e.,

$$\exists C > 0 : \forall x \in K : |\mathbf{f}(\mathbf{x})| < C, \quad (\text{B.4.6})$$

and there exists a point $\mathbf{x}_* \in K$ such that

$$\mathbf{f}(\mathbf{x}_*) = \sup_{\mathbf{x} \in K} |\mathbf{f}(\mathbf{x})|. \quad (\text{B.4.7})$$

B.5. Completeness and Banach spaces

Continuity is a very general concept that can be conducted on *topological spaces*. But in general understanding it on normed vector spaces is enough for many purposes.

B.6. Lipschitz continuity

In this section we consider D to be a closed subset of a Banach space. (possibly $D = \mathbb{R}$).

B.6.1. Definition of Lipschitz continuous function. Let $f : D \rightarrow \mathbb{R}$, we say that f is *Lipschitz-continuous* on D if and only if

$$\sup_{\substack{x, y \in D \\ x \neq y}} \frac{|f(x) - f(y)|}{|x - y|} =: L \in \mathbb{R}_0^+. \quad (\text{B.6.1})$$

B.6.2. Proposition. If f is Lipschitz continuous, then f is (sequentially) continuous, i.e.,

$$x_n \rightarrow \hat{x} \Rightarrow f(x_n) \rightarrow f(\hat{x}). \quad (\text{B.6.2})$$

B.6.3. Remark. The square root function

$$\begin{aligned} \sqrt{\cdot} : \mathbb{R}_0^+ &\rightarrow \mathbb{R}_0^+ \\ x &\mapsto \sqrt{x} \end{aligned} \quad (\text{B.6.3})$$

is continuous yet not Lipschitz-continuous. Indeed, from basic analysis we know that $\sqrt{\cdot}$ is continuous. To see that it is not Lipschitz continuous consider the difference quotient around 0

$$\frac{|\sqrt{x} - \sqrt{0}|}{|x - 0|} = \frac{1}{\sqrt{x}} \rightarrow \infty, \text{ as } x \rightarrow 0 \in \mathbb{R}_0^+. \quad (\text{B.6.4})$$

Hence

$$\sup_{\substack{x, y \in \mathbb{R}_0^+ \\ x \neq y}} \frac{|\sqrt{x} - \sqrt{y}|}{|x - y|} \quad (\text{B.6.5})$$

B.6.4. Proposition (differentiability and Lipschitz continuity). Suppose a function $f : D \rightarrow \mathbb{R}$ is differentiable on D . Then f is Lipschitz if and only if its derivative is bounded, i.e.,

$$\|f'\|_{L_\infty(D)} := \sup_{x \in D} |f'(x)| < \infty. \quad (\text{B.6.6})$$

Furthermore the Lipschitz constant of f coincides with $\|f'\|_{L_\infty(D)}$.

Proof

□

B.6.5. Remark. Note that proposition B.6.4 does not say that a Lipschitz function must be differentiable. In fact, there are nondifferentiable functions that are Lipschitz continuous, e.g., the absolute value defined on $[-1, 1]$.⁴

B.7. The Banach–Caccioppoli Contraction Principle

Consider a vector space X equipped with the norm $\|\cdot\|$. We say that $(X, \|\cdot\|)$ is a *Banach space* if every Cauchy sequence therein is convergent (i.e., has a limit) in X .⁵

B.7.1. Definition of contraction mapping. Let X be a Banach space and $Y \subseteq X$ where Y is closed. A mapping $\mathcal{T} : Y \rightarrow Y$ is called a *contraction* on Y if there exists $\kappa \in \mathbb{R}$ such that

$$0 \leq \kappa < 1 \text{ and } \forall x, y \in Y : \|\mathcal{T}x - \mathcal{T}y\| \leq \kappa \|x - y\|. \quad (\text{B.7.1})$$

B.7.2. Remark. A contraction mapping on Y reduces the distance between points of Y , whence the name.

⁴Note however that proposition B.6.4 is close to being sharp, in the sense that that one can do away with the differentiability assumption by using weak derivatives and differentiability everywhere. This important result goes by the name of Rademacher's theorem.

⁵The theory described here works also in *complete metric spaces*, which are more general structures than Banach spaces, but we will not need them here.

B.7.3. Remark (continuity of contractions). Note that a contraction $\mathcal{T} : Y \rightarrow Y$, $Y \subseteq X$ and $(X, \|\cdot\|)$ Banach space is uniformly continuous. Indeed, for $\varepsilon > 0$, choosing $\delta = \varepsilon/k$ and $x_0, x \in Y$ such that $\|x - x_0\| < \delta$ implies

$$\|\mathcal{T}x - \mathcal{T}x_0\| \leq \kappa \|x - x_0\| < \varepsilon. \quad (\text{B.7.2})$$

Continuity of \mathcal{T} is crucial for the next result's validity.⁶

B.7.4. Theorem (Banach–Caccioppoli contraction principle). *Let X be a Banach space and Y a closed subset (but not necessarily a subspace) of X . If $\mathcal{T} : Y \rightarrow Y$ is a contraction mapping, then \mathcal{T} has a unique fixed point, namely there exists exactly one $x_* \in Y$ such that*

$$x_* = \mathcal{T}x_*. \quad (\text{B.7.3})$$

Furthermore, x_ can be approximated using the fixed-point iteration starting with any $x_0 \in Y$ and building the sequence $(x_k)_{k \in \mathbb{N}_0}$*

$$x_k := \mathcal{T}x_{k-1}, \text{ for } k \in \mathbb{N}. \quad (\text{B.7.4})$$

Then this sequence converges and

$$\lim_{k \rightarrow \infty} x_k = x_*. \quad (\text{B.7.5})$$

Proof Consider a sequence $(x_k)_{k \in \mathbb{N}_0}$, such as the one constructed like in the statement. Then this sequence is a Cauchy sequence, indeed, for any $k, j \in \mathbb{N}_0$ we have

$$\begin{aligned} \|x_k - x_{k+j}\| &= \|x_k - x_{k+1} + x_{k+1} - \dots - x_{k+j-1} + x_{k+j-1} - x_{k+j}\| \\ &\leq \sum_{i=1}^j \|x_{k+i-1} - x_{k+i}\|. \end{aligned} \quad (\text{B.7.6})$$

Furthermore for each $l \geq 1$ we have

$$\|x_{l-1} - x_l\| = \|\mathcal{T}x_{l-2} - \mathcal{T}x_{l-1}\| \leq \kappa \|x_{l-2} - x_{l-1}\| \leq \dots \leq \kappa^{l-1} \|x_0 - x_1\|, \quad (\text{B.7.7})$$

and thus, defining $d_0 := \|x_0 - x_1\|$,

$$\|x_k - x_{k+j}\| \leq d_0 \sum_{i=1}^j \kappa^{k+i-1} = d_0 \kappa^k \sum_{i=0}^{j-1} \kappa^i \leq \frac{d_0}{1-\kappa} \kappa^k. \quad (\text{B.7.8})$$

But $\kappa^k \rightarrow 0$ as $k \rightarrow \infty$ because $0 \leq \kappa < 1$. It follows that for each $\varepsilon > 0$ there exists $k = k_\varepsilon \in \mathbb{N}$ such that

$$\|x_k - x_{k+j}\| < \varepsilon \quad \forall j > 0. \quad (\text{B.7.9})$$

That is $(x_k)_{k \in \mathbb{N}_0}$ is a Cauchy sequence. But X is a Banach space, which implies that $(x_k)_{k \in \mathbb{N}_0}$ converges, i.e., there exists $x_* \in X$ such that $x_k \rightarrow x_*$ as $k \rightarrow \infty$. Recalling that $x_k \in Y$ for all k and that Y is a closed subset of X , it follows that $x_* \in Y$. Furthermore, we have

$$x_* = \lim_{k \rightarrow \infty} x_k = \lim_{k \rightarrow \infty} \mathcal{T}x_{k-1} = \mathcal{T} \lim_{k \rightarrow \infty} x_{k-1} = \mathcal{T}x_*, \quad (\text{B.7.10})$$

⁶In fact, a contraction is Lipschitz continuous with constant less than 1, which is much stronger than uniform continuous, but this fact is not needed for the proof of the fixed-point theorem. Fixed-point theorems are valid under weaker assumptions are available, an example is Brower's fixed-point and Schauder's extension to infinite dimensions. These important topology results are not needed in this course though.

where the second last step is due to the continuity of \mathcal{T} . Thus we have proved that \mathcal{T} has a fixed point x_* .

To close, we need to ascertain the uniqueness of x_* . Suppose another point $x_{**} \in Y$ is also a fixed point, it follows that

$$0 \leq \|x_* - x_{**}\| = \|\mathcal{T}x_* - \mathcal{T}x_{**}\| \leq \kappa \|x_* - x_{**}\|. \quad (\text{B.7.11})$$

It follows that $(1 - \kappa)\|x_* - x_{**}\| \leq 0$, but $1 - \kappa > 0$, so this means that $\|x_* - x_{**}\| = 0$ and thus $x_* = x_{**}$. This proves uniqueness. \square

B.8. Measurable spaces and measures

B.8.1. Definition of measure space, σ -algebra. A space X is called a *measurable space* if it has a family of subsets \mathcal{F} such that

- (i) \mathcal{F} contains X ,
- (ii) \mathcal{F} is closed under complementation,
- (iii) \mathcal{F} is closed under countable (but possibly infinite) unions.

Such a family \mathcal{F} of subsets is called a sigma-algebra (or sigma-field) often written σ -algebra.⁷

B.8.3. Definition of generated σ -algebra. Let X be a set and \mathcal{S} a collection of its subsets. We define the σ -algebra generated by \mathcal{S} to be the smallest σ -algebra containing \mathcal{S} .

B.8.4. Definition of measurable function. Let $(X, \mathcal{F}), (Y, \mathcal{G})$ be two measure spaces, a function $f : D \rightarrow X$ is measurable if and only if

$$B \in \mathcal{G} \Rightarrow f^{-1}(B) \in \mathcal{F} \quad (\text{B.8.1})$$

B.9. Integrals

B.9.1. Lemma (du Bois-Reymond). Let (X, \mathcal{F}, μ) be a measure space, with $\mathcal{F} = \sigma(\mathcal{S})$ and $f : X \rightarrow \mathbb{R}$ a μ -measurable function. If

$$\int_A f \, d\mu = 0 \quad \forall A \in \mathcal{S} \quad (\text{B.9.1})$$

then the function f is μ -almost identically 0, i.e., $f \equiv 0$ μ -a.e.

B.9.2. Applications of du Bois-Reymond. In spite of its simplicity (or perhaps because of it) du Bois-Reymond's lemma is one of the most useful results in analysis. Usually we use it to compare two measurable functions, say g and h , of whom we know that $\int_A g \, d\mu = \int_A h \, d\mu$ for A in a family \mathcal{S} of subsets of X that generates μ 's σ -algebra. A consequence of the linearity of $\int_A \cdot \, d\mu$ (for any fixed A) and du Bois-Reymond's lemma is that $g = h$ μ -a.e..

⁷Sigma is shorthand for (set) "summation" an old term for "union", was introduced. A more meaningful name for a sigma-field (sigma-algebra) would be set-field (set-algebra).

B.9.3. Example (how to choose \mathcal{S}). One very useful thing about du Bois-Reymond's lemma is that one doesn't have to check the assumption (B.9.1) for all subsets of the σ -algebra \mathcal{F} but just for a subclass of them. A typical situation is where $X = \mathbb{R}^d$ and \mathcal{F} is the set of Lebesgue measurable sets while \mathcal{S} is that of rectangles. For example, using Fubini's theorem, the fundamental theorem of calculus in one dimension one can prove the divergence formula. Indeed for any rectangle $R = \prod_{i=1}^d (a_i, b_i)$ we have

$$\int_R f(\mathbf{x}) d\mathbf{x} = \int_{a_1}^{b_1} \int_{R'} f(x_1, \mathbf{x}') d\mathbf{x}' dx_1 = \int_{a_1}^{b_1} \quad (\text{B.9.2})$$

Exercises and problems on Analysis basics

Problem B.1 (Fréchet differentiability implies continuity). Let $\mathbf{f} : D \rightarrow \mathbb{R}^k$ be a function that is Fréchet differentiable at a given point $\mathbf{x} \in D$. Show that \mathbf{f} is continuous at \mathbf{x} .

Exercise B.2. Show that the function

$$g(x, y) := \begin{cases} 0 & \text{if } x = 0 \text{ and } y = 0 \\ \frac{x^4 y}{x^6 + y^3} & \text{otherwise.} \end{cases} \quad (\text{PB.2.1})$$

has a Gâteaux derivative at $(0, 0)$. Show that \mathbf{g} is discontinuous at $(0, 0)$. Deduce it has no Fréchet derivative thereon.

Problem B.3 (derivative of the inverse matrix function). Denote by $\text{GL}(\mathbb{R}^n)$ the set of all invertible matrices in $\mathbb{R}^{n \times n}$ and by inv the inverse matrix function

$$\begin{aligned} \text{inv} : \text{GL}(\mathbb{R}^n) &\rightarrow \text{GL}(\mathbb{R}^n) \\ \mathbf{X} &\mapsto \mathbf{X}^{-1} \end{aligned} \quad (\text{PB.3.1})$$

Show that for any matrix $\mathbf{Z} \in \mathbb{R}^{n \times n}$ sufficiently small ϵ the Neumann series

$$\mathbf{I} - \epsilon \mathbf{Z} + \epsilon^2 \mathbf{Z}^2 - \dots = \sum_{k=0}^{\infty} (-\epsilon \mathbf{Z})^k \quad (\text{PB.3.2})$$

converges absolutely, the matrix $\mathbf{I} + \epsilon \mathbf{Z}$ is invertible, and

$$(\mathbf{I} + \epsilon \mathbf{Z})^{-1} = \sum_{k=0}^{\infty} (-\epsilon \mathbf{Z})^k. \quad (\text{PB.3.3})$$

Use this result to show the existence and compute the directional derivative of inv at a point $\mathbf{X} \in \text{GL}(\mathbb{R}^n)$ in the direction $\mathbf{Y} \in \mathbb{R}^{n \times n}$. Check that your result agrees with what you know from basic calculus when $n = 1$.

A review of ordinary differential equations

C.1. ODE's and IVP's

To understand the numerical aspects of ODE's, one must first understand ODE's for their own sake. Although we assume that you have some familiarity with the concept of ODE's, before setting off to their computational approximation and numerical analysis we will review some of their most important analytic properties.

C.1.1. Setting. The notation introduced now will be used throughout the course, unless otherwise stated. Let $d \in \mathbb{N}$, we consider a function or field

$$D \ni (x, \mathbf{y}) \mapsto \mathbf{f}(x, \mathbf{y}) \in \mathbb{R}^d \quad (\text{C.1.1})$$

where the *domain* $D \subseteq \mathbb{R} \times \mathbb{R}^d$ is a non-empty open set (cf. B.2.15). Given such a function \mathbf{f} and a point $(x_0, \mathbf{y}_0) \in D$, a *solution* of the *ordinary differential equation* (ODE) defined by \mathbf{f} with initial value \mathbf{y}_0 is a function

$$\mathbf{Y} : I \rightarrow \mathbb{R}^d \quad (\text{C.1.2})$$

where is to find an open interval $I \subseteq \mathbb{R}$, such that

$$\forall x \in I : (x, \mathbf{Y}(x)) \in D, \quad (\text{C.1.3})$$

$$\mathbf{Y} \text{ is differentiable and } \frac{d\mathbf{Y}(x)}{dx} = \mathbf{f}(x, \mathbf{Y}(x)) \quad (\text{C.1.4})$$

$$\text{and } \mathbf{Y}(x_0) = \mathbf{y}_0. \quad (\text{C.1.5})$$

Such a function \mathbf{Y} is called *exact solution* of the *initial value problem* (IVP)

$$\dot{\mathbf{y}} = \mathbf{f}(x, \mathbf{y}), \quad (\text{C.1.6})$$

$$\mathbf{y}(x_0) = \mathbf{y}_0. \quad (\text{C.1.7})$$

Equation (C.1.6) is a vector-valued *ordinary differential equation of the first order*, (C.1.7) is known as *initial condition* and \mathbf{y}_0 is the *initial value*. The same ODE gives rise to different IVP's by choosing a different initial value.

C.1.2. Remarks, terminology, notation.

- (a) Owing to the importance of time in many physical models described by ODE's, the first variable of \mathbf{f} is called *time variable* or *time*. We will use very often this terminology. The interval I is known also as the *time interval* and x_0 is the *initial time*.
- (b) The function \mathbf{f} is called the *defining (or generating) function* for the IVP (C.1.6)–(C.1.7); this terminology is used especially in the context of *dynamical systems*. In many instances, for a fixed \mathbf{y} , the function $\mathbf{f}(\cdot, \mathbf{y})$ is a constant. In that case we think of \mathbf{f} as being simply $\mathbf{f} : D' \rightarrow \mathbb{R}^d$, with $D := \mathbb{R} \times D'$. In this case, the ODE is said to be *autonomous*.

- (c) The condition (C.1.3) simply insures that what we are writing makes sense. It implies that the line segment $I \times \{\mathbf{y}_0\}$ lies in D .
- (d) The shorthand for the derivatives

$$\frac{d\mathbf{Y}(x)}{dx} = d_x \mathbf{Y}(x) = \mathbf{Y}'(x) \quad (\text{C.1.8})$$

will be used interchangeably throughout the course.

Note however that the “dot” notation, $\dot{\mathbf{y}}$, will be confined only to the definition of the differential equation, such as (C.1.6), and the letter \mathbf{y} itself will be treated just as a symbol. This allows us to use the same letter in two different ODE’s without worrying that it may mean the solution.

- (e) When $d > 1$, many people prefer the term *field*, instead of function, to indicate \mathbf{f} .
- (f) The function \mathbf{Y} introduced above is usually simply called *solution* of the IVP. We call it *exact solution* to highlight the contrast with the numerical solution that will be introduced later on.¹
- (g) Notice that while we use lower case letters to indicate the *unknown* in (C.1.6) and, the actual solution is indicated in upper case letters. In particular, the letter \mathbf{y} will be used only to indicate the unknown, without ever considering it as a particular function. Throughout this course, unless otherwise stated, we use upper case letters, such as \mathbf{Y} , Y , \mathbf{Z} , etc., to indicate the *exact solutions to ODE’s*, whereas lower case letter such as \mathbf{u} , \mathbf{z} , z , etc., possibly indexed, indicate vectors (if boldfaced) or numbers (if not boldfaced).
- (h) We use the boldface characters to indicate vectors when $d > 2$. If $d = 1$ we use instead normal-boldness characters.
- (i) By *equilibrium point* or *steady state* we mean any point $\mathbf{u} \in \mathbb{R}^d$ such that the problem with initial condition $\mathbf{y}(x_0) = \mathbf{u}$ is solved by the constant solution $\mathbf{Y} \equiv \mathbf{u}$.

C.1.3. First order scalar linear ODE’s. Consider the IVP

$$\dot{y} = p(x)y + g(x) \quad (\text{C.1.9})$$

$$y(x_0) = y_0 \in \mathbb{R} \quad (\text{C.1.10})$$

where $p, g : (a, b) \rightarrow \mathbb{R}$ are continuous functions, for some fixed $a < b \in \mathbb{R}$. Thus $D = (a, b) \times \mathbb{R}$.

This is one of the simplest ODE’s, one of the rare cases in which the solution can be sought explicitly. An explicit representation for the solution of this IVP can be derived by multiplying both sides of (C.1.9) with the *integrating factor*

$$\mu(x) = \exp\left(-\int_{x_0}^x p(t) dt\right) \quad (\text{C.1.11})$$

and then integrate with respect to x as to obtain

$$Y(x) = y_0 \exp \int_{x_0}^x p(t) dt + \int_{x_0}^x g(s) \exp \int_s^x p(t) dt ds. \quad (\text{C.1.12})$$

¹Many authors use the words *real solution* (even when they look for a complex or vector valued solution!) or *true solution* (even though they do not think of their numerical solution to be any more false than the true one!).

C.1.4. Remark (explicit vs. numerical solution and benchmark solutions). Whenever an ODE can have its solutions written explicitly in terms of the data by using “elementary” functions such as the exponential, we say that an “explicit solution” is available. Provided we know how to compute the elementary functions involved and the integrals then we are able to compute the values of the function Y .

There is no need for a numerical method for ODE's in this case; it seems foolish to spend effort in approximating a solution that we already know. However, any numerical method that works on general ODE's (for example nonlinear ones) should work on the simplest cases. Therefore, despite having already solved the problem, the simplest ODE's constitute a good *testbed* or *benchmark problems*. By *benchmark problem*, we mean a problem of which we know the solution and that can be used as a test of a novel numerical method by comparing the numerical result with the exact solution. In this sense, the first order scalar linear ODE's constitute a very important class of benchmark problems.

Let us now consider some particular cases, depending on various choices of p and g in (C.1.9), of scalar linear equations.

C.1.5. Example (constant coefficients and homogeneous). A simple (yet interesting) special case of (C.1.9) is when $p(x) \equiv \lambda \in \mathbb{R}$ and $g(x) \equiv 0$. In this case, the representation (C.1.12) takes the simplified form

$$Y(x) = y_0 e^{\lambda(x-x_0)}. \quad (\text{C.1.13})$$

It is easy to see that 0 is the only equilibrium point for this equation.

If $\lambda < 0$ then the equilibrium point 0 is *stable* for this equation, this meaning that the exact solution Y starting from any initial value $y_0 \in \mathbb{R}$ satisfies

$$\lim_{x \rightarrow \infty} Y(x) = 0. \quad (\text{C.1.14})$$

We say that the ODE with $\lambda < 0$ is *stable*.

On the other hand, if $\lambda > 0$ this ODE is *unstable*, in that the fixed point 0, corresponding, to the initial value $y_0 = 0$ is unstable.

To understand what stability means, let us compare Y to Y_ε , the solution of the *perturbed IVP* consisting of (C.1.9) with initial condition

$$y(x_0) = y_0 + \varepsilon, \quad (\text{C.1.15})$$

for $\varepsilon > 0$. Indeed, by homogeneity and linearity, the “error” $Y_\varepsilon - Y$ satisfies the same ODE, i.e.,

$$Y'_\varepsilon(x) - Y'(x) = \lambda(Y_\varepsilon(x) - Y(x)) \quad (\text{C.1.16})$$

and, by the representation formula (C.1.13), we have

$$Y_\varepsilon(x) - Y(x) = \varepsilon e^{\lambda(x-x_0)}. \quad (\text{C.1.17})$$

On one hand this is telling us that as $\varepsilon \rightarrow 0$ then $Y_\varepsilon \rightarrow Y$ uniformly on any bounded interval. On the other hand this same formula tells us that the convergence $Y_\varepsilon \rightarrow Y$ deteriorates rapidly as time increases (so there is no uniform convergence on unbounded intervals). This is bad news. To understand why this is could be damaging numerically, assume $\varepsilon = 10^{-8}$, which could be our machine precision in an actual computations, and $\lambda = 20$, which is easily the case in some applications. Say $x_0 = 0$,

then at times $x = 1, 2, 3$ the error is, respectively, $4.851, 2.352 \times 10^9, 1.1420 \times 10^{18}$. In figure 1(a) and 1(b) we plot examples of the stable and unstable situation, respectively.

This example seems somewhat artificial as the error increases very fast, but also a non-zero solution increases very fast, so much as to make the *relative error*

$$\frac{Y_\epsilon(x) - Y(x)}{Y(x)} = \frac{\epsilon}{y_0} \quad (\text{C.1.18})$$

constant (and small, if ϵ is small with respect to y_0). We will see next that in some situations even the relative error cannot be bounded.

C.1.6. Example (and things can get nastier). In Example C.1.5, although the absolute error gets “out of hand”, the relative error is controllable in terms of the problem’s data. But this needs not be the case for all problems. For example, consider the IVP

$$\begin{aligned} \dot{y} &= 100y - 101 \exp(-x) \\ y(0) &= 1 + \epsilon =: y_\epsilon. \end{aligned} \quad (\text{C.1.19})$$

The solution of this IVP, in view of the representation formula (C.1.12), is given by

$$Y_\epsilon = y_\epsilon \exp(100x) - 101 \int_0^x \exp(-s) \exp(100(x-s)) \, ds = \epsilon \exp(100x) + \exp(-x), \quad (\text{C.1.20})$$

for any $\epsilon \in \mathbb{R}$. In this case, even the relative error,

$$\frac{Y_\epsilon(x) - Y(x)}{Y(x)} = \epsilon \exp(101x), \quad (\text{C.1.21})$$

cannot be bounded by a constant depending only on the problem’s data (i.e., one that involves only the initial value and ϵ).

Notice that $\lim_{\epsilon \rightarrow 0} Y_\epsilon = Y$, uniformly on each bounded interval $[0, T]$, (but not on all of $[0, +\infty)$), but in practice this convergence is not very useful. For example, for $\epsilon = 10^{-6}$, $x = 1$ then $Y_\epsilon(1) = 10^{-6} + 10^{-6} 10^{50} \approx 10^{44}$, which is a too big a number for computers to handle efficiently and without too much round-off error. We say that *the IVP is relatively unstable, or ill conditioned*. In Figure 1(c) we plot the solution of the problem and some perturbations.

C.1.7. Remark (conclusion on stability). It is important to understand stability of an ODE, or that of an IVP. When we introduce numerical methods, we will see that the method itself comes with its own stability properties. Some numerical methods may be unstable, even if they are applied to stable problems. Therefore, *stable problems will constitute a test-bed for the numerical methods we will derive in this course*.

On the other hand, unstable problems cannot be improved by a particular numerical method. Therefore, using a particular numerical method for unstable problems is a very delicate matter as errors may violently accumulate and make the numerical results useless: even methods that are good for stable problems may fail on unstable problems.

C.1.8. Example (Multiple scale (“multiscale”) problems). Let $\alpha \gg 1$ (e.g., $\alpha = 20$) and let Y be the solution of the IVP

$$\begin{aligned} \dot{y} &= \alpha y + \alpha \sin x + \cos x, \\ y(x_0) &= y_0 \end{aligned} \quad (\text{C.1.22})$$

Again, this is a scalar linear problem and its exact solution is readily seen to be

$$Y(x) = y_0 e^{-\alpha x} + \sin x. \quad (\text{C.1.23})$$

In this equation, α being quite big, the $\sin x$ part of the solution is a slow time-scale component, which oscillates slowly in x , whereas $y_0 e^{-\alpha x}$ is the fast time-scale component, which drops to zero very fast as x increases, as shown in Figure 1(d). We have thus *two very disparate time-scales* which need to be solved accurately and this is a source of difficulties for a numerical scheme.

Since $Y' \approx \alpha Y$ for small times, small changes in y_0 can lead to big changes in Y' , the slope. In numerical algorithms Y' is approximated and this can lead to a poor approximation. This problem is known as *stiff problem*. Stiffness in a problem, such as this one, comes often from the interplay of two or more disparate time scales.

C.1.9. Definition of stiff problems. An IVP is *stiff* if the solution of the problem is slowly changing, but perturbations of the problem have very rapidly changing solutions.

C.1.10. Example (stiff bump). Another interesting type of stiff problems is provided by the IVP

$$\begin{aligned} \dot{y} &= -2\alpha(x-1)y, \\ y(x_0) &= \varepsilon. \end{aligned} \quad (\text{C.1.24})$$

The exact solution is

$$Y_\varepsilon(x) = \varepsilon e^{\alpha - \alpha(x-1)^2}. \quad (\text{C.1.25})$$

In the figure we plot the solution for various values of ε , including $\varepsilon = 0$; we see that the very steady 0 solution is surrounded by rapidly changing ones, as soon as $\varepsilon \neq 0$. The IVP is thus seen to be a stiff one.

C.2. Existence and uniqueness of solutions to IVP's

In this section we use the notation introduced in C.1.1. The following is a fundamental result about ODE's which guarantees that an IVP has a solution under some assumptions on the defining function.

C.2.1. Theorem (local existence of a solution). Suppose that $f : D \rightarrow \mathbb{R}^d$ is continuous and satisfies the following Lipschitz condition in the second variable about the initial conditions (x_0, y_0) :

For some $\rho_0, K_0 > 0$:

$$\forall (x, y_1), (x, y_2) \in B_{\rho_0}((x_0, y_0)) \cap D : |f(x, y_1) - f(x, y_2)| \leq K_0 |y_1 - y_2|. \quad (\text{C.2.1})$$

Then, there exists a (local) solution to the IVP (C.1.6)–(C.1.7). (Meaning that there exist a $\delta > 0$ and a $Y : (x_0 - \delta, x_0 + \delta) \rightarrow \mathbb{R}^d$ which solves the IVP.)

C.2.2. Remark (Lipschitz condition). The condition (C.2.1), whereby the variation of the function f can be controlled linearly by the variation of its argument, is called a *local Lipschitz condition* on f . Because we need this condition only in the variable y , we say that the function f is *Lipschitz continuous at the point* (x_0, y_0) , *with respect to its second argument or variable*.

If the function f is Lipschitz continuous at each point $(x, y) \in D$, then we say that f is *locally Lipschitz continuous*.

Notice that if f is locally Lipschitz continuous, the ρ, K in (C.2.1) depend on the point (x, y) ; in particular there is not necessarily an upper bound on K or a lower bound on ρ . If there are such bounds then we say that the function f is *globally Lipschitz continuous*.

Concisely, f is globally Lipschitz continuous if and only if

$$\exists K > 0 : \forall (x, y_1), (x, y_2) \in D : |f(x, y_1) - f(x, y_2)| \leq K |y_1 - y_2|. \quad (\text{C.2.2})$$

You should appreciate the differences between condition (C.2.2) and (C.2.1).

As you read through this chapter, you may find it useful to refer to Appendix ??§??.

C.2.3. Example (local solution). Consider the problem of finding a solution to the IVP

$$\dot{y} = -y^2 \text{ and } y(x_0) = y_0. \quad (\text{C.2.3})$$

To apply C.2.1, let us check the local Lipschitz condition (C.2.1) with $f(x, y) := -y^2$. Fix $\rho \in \mathbb{R}^+$ and $K = 2(|y_0| + \rho)$, then for all $(x, y_1), (x, y_2) \in B_\rho(x_0, y_0)$ we have

$$\begin{aligned} |y_2^2 - y_1^2| &= |(y_2 + y_1)(y_2 - y_1)| = |y_2 + y_1| |y_2 - y_1| \\ &\leq 2 \max\{|y_1|, |y_2|\} |y_2 - y_1| \\ &\leq K |y_2 - y_1|. \end{aligned} \quad (\text{C.2.4})$$

By Theorem C.2.1, we know that there exists (at least) a small interval $I := B_\delta^1(x_0)$, and a function $Y : I \rightarrow \mathbb{R}$ such that

$$Y' = -Y^2 \text{ and } Y(x_0) = y_0. \quad (\text{C.2.5})$$

Notice, however, that in this example f is not globally Lipschitz continuous in its second argument (prove this as an exercise, by showing that (C.2.2) fails). This fact will give us the chance to come back to this example in a little while.

C.2.4. Theorem (uniqueness). Suppose f satisfies (C.2.1), and that

$$Y : I \rightarrow \mathbb{R}^d \text{ and } Z : J \rightarrow \mathbb{R}^d \quad (\text{C.2.6})$$

are both solutions of (C.1.6)–(C.1.7). Then Y and Z coincide on their common domain; i.e.,

$$Y(x) = Z(x), \forall x \in I \cap J. \quad (\text{C.2.7})$$

C.2.5. Definition of maximal solution. A function $Y : I \rightarrow \mathbb{R}^d$ is called *maximal solution* of the IVP (C.1.6)–(C.1.7) if and only if for any solution $Z : J \rightarrow \mathbb{R}^d$ of the IVP we have

$$J \subseteq I \text{ and } \forall x \in J : Y(x) = Z(x). \quad (\text{C.2.8})$$

Using Theorems C.2.1, C.2.4 and Zorn's Lemma, it is possible to show that if f satisfies the local Lipschitz condition then there exists a unique maximal solution.

C.2.6. Example (on the maximal solution's domain). Intuitively, the maximal solution is the biggest one that can be found. The maximal solution's domain (interval) is not necessarily as big as possible. To see this, let us have a closer look at the solutions of IVP (C.2.3).

This is a fortunate case where we can compute the solution explicitly, by using the separation of variables (we know that the solution is unique thanks to Theorem C.2.4). If $y_0 = 0$, then the solution is $Y \equiv 0$. If $y_0 \neq 0$ and $x_0 = 0$ (for simplicity), then separation of variables leads to the solution

$$Y(x) = \frac{y_0}{1 + x y_0}. \quad (\text{C.2.9})$$

Notice that the function Y has domain $\mathbb{R} \setminus \{-1/y_0\}$. Define $x_* := -1/y_0$. If $y_0 < 0$, then $x_* > 0$ and

$$\lim_{x \nearrow x_*} Y(x) = -\infty. \quad (\text{C.2.10})$$

We say that x_* is a *blow-up time*.

Although the values $Y(x)$, as defined above, exist for $x > x_*$, the fact that Y cannot be extended continuously (let alone in a differentiable way) across x_* makes the part of the domain above x_* to be refuted as the domain of the solution to the ODE.

Thus, by uniqueness, the maximal solution is given by (C.2.9) has domain $(-\infty, -1/y_0)$, which is not the biggest possible interval that can be chosen as domain for a function with graph D .

C.2.7. Theorem (global existence). *If the function f is globally Lipschitz continuous, then the maximal solution exists for all possible times.*

C.2.8. Remarks.

- (a) The proof of existence in theorem C.2.1 relies on a *fixed point argument*. The idea is to introduce the map

$$T : C^0(a, b) \rightarrow C^0(a, b) \\ T Y(x) := y_0 + \int_{x_0}^x f(\xi, Y(\xi)) d\xi, \text{ for } x \in (a, b). \quad (\text{C.2.11})$$

We must have $a < x_0 < b$ for this definition to make sense. The rest of the proof consists in showing that it is possible to find $a, b \in \mathbb{R}$ such that T is a contraction, i.e.,

$$\exists \gamma \in (0, 1) : \forall Y_1, Y_2 \in C^0(a, b) \|T Y_2 - T Y_1\|_{L^\infty(a, b)} \leq \gamma \|Y_2 - Y_1\|_{L^\infty(a, b)}, \quad (\text{C.2.12})$$

for a $\gamma \in (0, 1)$ that is independent of Y_1, Y_2 .

By the Contraction Mapping Theorem (Banach–Cacciopoli), it follows then that T admits a fixed point in $C^0(a, b)$. The last step is to show that this fixed point is in fact a differentiable function and that it satisfies the original ODE.

(b) The solution Y of (C.1.6) is $C^1(a, b)$ and it is characterised by the relation

$$Y = TY, \quad (\text{C.2.13})$$

where T is the contraction mapping defined in (C.2.11). Explicitly

$$Y(x) = y_0 + \int_{x_0}^x f(\xi, Y(\xi)) d\xi. \quad (\text{C.2.14})$$

This relation is called the *integral formulation* of the IVP (C.1.6)–(C.1.7).

C.3. More examples

C.3.1. Example. (non-uniqueness) Consider the IVP

$$\dot{y} = 2\sqrt{|y|} \text{ and } y(0) = 0. \quad (\text{C.3.1})$$

Note that the function $f(x, y) = \sqrt{|y|}$ is not Lipschitz at $(x, 0)$ (for any $x \in \mathbb{R}$). Therefore the uniqueness Theorem C.2.4 cannot be invoked. In fact, there is no uniqueness. Two different solutions are given by

$$Y_1(x) = 0, \quad (\text{C.3.2})$$

$$Y_2(x) = x^2. \quad (\text{C.3.3})$$

Can you find other ones? (There are infinitely many solutions to this IVP.)

C.3.2. Example (damped linear oscillator: a stiff IVP). Consider the second order scalar IVP

$$\ddot{z} + 2d\dot{z} + z = 0, \quad (\text{C.3.4})$$

$$z(0) = z_0, \dot{z}(0) = w_0,$$

where $d \in [1, \infty)$ is a parameter representing the amount of *damping*. The solution has the form

$$Z(x) = Ae^{\lambda_1 x} + Be^{\lambda_2 x} \quad (\text{C.3.5})$$

where A, B depend on the initial conditions and the λ_i 's are the solutions of the characteristic equation

$$\lambda^2 + 2d\lambda + 1 = 0. \quad (\text{C.3.6})$$

The roots are given by

$$\lambda_{1,2} = -d \pm \sqrt{d^2 - 1}. \quad (\text{C.3.7})$$

A more instructive way of getting to the same solution is to write the ODE as a system, with $y_1 = z$ and $y_2 = z'$ so that

$$y_1' = z' = y_2 \quad (\text{C.3.8})$$

$$y_2' = z'' = -2dz' - z = -2dy_2 - y_1 \quad (\text{C.3.9})$$

which we rewrite as a first order linear vector ODE in \mathbb{R}^2

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ -1 & 2d \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \quad (\text{C.3.10})$$

To find the solution we must diagonalise this matrix. The eigenvalues are the solutions of the characteristic equation (C.3.6). The eigenvectors $\mathbf{e}^1, \mathbf{e}^2$ can be found accordingly. Let us denote by $\mathbf{P} := [\mathbf{e}^1, \mathbf{e}^2]$. Taking $\mathbf{u} = \mathbf{P}\mathbf{y}$, the system (C.3.8) can be written in the diagonal form

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}' = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad (\text{C.3.11})$$

Since $d \geq 1$ both eigenvalues are real and negative. This implies that the origin $\mathbf{0}$ is a stable point. However, inspecting the case where $d \gg 1$, and after some calculations, we see that

$$\lambda_1 = -d - \sqrt{d^2 - 1} \approx -2d \quad (\text{C.3.12})$$

$$\lambda_2 = -d + \sqrt{d^2 - 1} \approx \frac{-1}{2d} \quad (\text{C.3.13})$$

(Show that in fact $\lambda_2 < -1/2d$.) For high values of d it turns out that the two components of the general solution, $x \mapsto \exp(\lambda_1 x), \exp(\lambda_2 x)$, have very different time-scales; i.e., this is a very stiff problem, despite it being stable.

C.4. Stability

The various flavours of Lipschitz conditions play an important role for existence and uniqueness. Do they have some other relevance?

Stability (like stiffness) is such a general concept that it is better to not define it rigorously. Despite (or due to) its wide use throughout analysis, applied mathematics, mechanics and computational sciences, any attempt to frame stability in a rigorous definition makes it hard to use somewhere else. For example, there are more than 45 different, and non-equivalent, definitions of stability in ODE theory only.

To add on this confusion, from each discipline to another the term stability has a related, but slightly different meaning. That is why the word “stability” comes always accompanied by a specifier, such as asymptotic, Lyapunov, absolute, conditional, etc. From the analysis view-point, *stability of a problem (or class of problems)* is a measure of the sensitivity of the solution to the problem's data. From the numerical analysis view-point *stability of a method* measures the reliability of its results when it is applied to problems that are deemed stable analytically. Thus the two concepts are related and understanding the analytic aspects is propedeutic to a study of numerical stability. While our chief concern in the course will be numerical stability, in this section we look at stability from the analytical view-point.

C.4.1. A perturbation approach to stability. To focus on the stability of an ODE, let us consider a perturbed version of IVP (C.1.6), (C.1.7):

$$\dot{\mathbf{y}} = \mathbf{f}(x, \mathbf{y}) + \boldsymbol{\delta}(x), \quad (\text{C.4.1})$$

$$\mathbf{y}(0) = \mathbf{y}_0 + \boldsymbol{\epsilon} \quad (\text{C.4.2})$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^n$ and $\boldsymbol{\delta} : I \rightarrow \mathbb{R}^n$ are such that $|\boldsymbol{\epsilon}|, \|\boldsymbol{\delta}\|_\infty \leq \epsilon$. We think of ϵ to be small ($0 < \epsilon \ll 1$) and both $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}$ could be consequences of machine precision arithmetic or measurement errors in the data. (Notice that we have taken, without loss of generality, $x_0 = 0$. Indeed, any equation with $x_0 \neq 0$ can be reduced to an equivalent one with $x_0 = 0$ by the change of variable $x \mapsto x - x_0$.)

Assume that both (C.1.6)–(C.1.7) and (C.4.1)–(C.4.2) possess global solutions and denote them by Y and Z respectively.

C.5. Question. Can we control—i.e., bound by data and ϵ the norm of—the *perturbation error*

$$e(x) := Z(x) - Y(x)? \quad (\text{C.5.1})$$

We will answer this question by distinguishing different cases. As a warm-up exercise, you should tackle the following.

C.5.1. Exercise (stability analysis of the scalar linear IVP). Let Y be the solution of the scalar first order linear IVP

$$\dot{y} = f(t, y) := a(t)y, \quad y(0) = y_0, \quad (\text{C.5.2})$$

where $a \in C^0(\mathbb{R})$ is a continuous function satisfying

$$\lambda := \|a\|_\infty = \sup_{t \in \mathbb{R}} |a(t)| < \infty. \quad (\text{C.5.3})$$

Suppose that $b \in C^0(\mathbb{R})$ and $\epsilon \in \mathbb{R}^+$, $\delta \in \mathbb{R}$ are such that

$$\|b\|_\infty, |\delta| \leq \epsilon. \quad (\text{C.5.4})$$

Consider the solution \tilde{Y} of the perturbed IVP

$$\dot{y} = \tilde{f}(t, y) := a(t)y + b(t), \quad y(0) = y_0 + \delta. \quad (\text{C.5.5})$$

(a) By showing that both f and \tilde{f} are globally Lipschitz with Lipschitz constant λ , use the results from lectures to deduce that both (C.5.2) and (C.5.5) possess unique global solutions.

(b) Show that the perturbation error $e(t) := \tilde{Y}(t) - Y(t)$ satisfies

$$e'(t) = a(t)e(t) + b(t), \quad e(0) = \delta. \quad (\text{C.5.6})$$

(c) Using an integrating factor or otherwise, show that

$$e(t) = \delta \exp(A(t)) + \int_0^t b(r) \exp(A(t) - A(r)) dr, \quad (\text{C.5.7})$$

where

$$A(t) := \int_0^t a(s) ds. \quad (\text{C.5.8})$$

(d) Suppose that there exists $\mu \leq \lambda$ such that

$$-\lambda \leq a(t) \leq \mu \quad \forall t \in \mathbb{R}. \quad (\text{C.5.9})$$

(i) Show that if $\mu < 0$, then

$$|e(t)| \leq \frac{1 + (|\mu| - 1) \exp(\mu t)}{|\mu|} \epsilon \leq \begin{cases} \epsilon / |\mu|, & \text{for } \mu \in (-1, 0) \\ \epsilon, & \text{for } \mu \leq -1 \end{cases}, \quad (\text{C.5.10})$$

and hence that as $\epsilon \rightarrow 0$, the perturbation error converges to zero uniformly on $[0, \infty)$.

(ii) Show that if $\mu > 0$, then

$$|e(t)| \leq \frac{(|\mu| + 1) \exp(\mu t) - 1}{|\mu|} \epsilon. \quad (\text{C.5.11})$$

Can it be concluded that as $\epsilon \rightarrow 0$

- * $e(t)$ converges to zero as for each fixed t ?
- * e converges to zero uniformly on any bounded time interval $[0, T]$ ($T < \infty$)?
- * e converges to zero uniformly on $[0, \infty)$?

(e) Suppose $a \equiv \nu$, $b \equiv 0$ and $\delta = \epsilon > 0$. Show that

$$e(t) = \epsilon \exp(\nu t). \quad (\text{C.5.12})$$

What can you say about pointwise convergence, uniform convergence on $[0, T]$ ($T < \infty$) and uniform convergence on $[0, \infty)$ of e for (i) $\nu > 0$ and (ii) $\nu \leq 0$?

C.5.2. Stability for linear ODE's. For the scalar case, see Exercise C.5.1. The vector case is similar, except one has to use matrixes (and their exponentials) instead of scalars. Consider, in the ODE's (C.1.6) and (C.4.1), with the defining function

$$\mathbf{f}(x, \mathbf{y}) = \mathbf{A}\mathbf{y}, \quad (\text{C.5.13})$$

where $\mathbf{A} \in \mathbb{R}^{d \times d}$. For simplicity we assume that \mathbf{A} is constant with respect to $x \in \mathbb{R}$ (constant coefficients) and that its eigenvalues, $\lambda_1, \dots, \lambda_d$, such that $\text{re } \lambda_1 \leq \dots \leq \text{re } \lambda_d$, are all non-defective. To say that the eigenvalues are non-defective means that there exists a matrix $\mathbf{P} \in \mathbb{C}^{d \times d}$ such that

$$\mathbf{A} = \mathbf{P}^{-1} \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_d) \mathbf{P}. \quad (\text{C.5.14})$$

Using linearity we find that the error \mathbf{e} satisfies

$$\mathbf{e}' = \mathbf{A}\mathbf{e} + \boldsymbol{\delta} \text{ and } \mathbf{e}(0) = \boldsymbol{\epsilon} \quad (\text{C.5.15})$$

From the theory of matrix exponential ?? and Duhamel's principle we can express the solution of (C.5.13) as

$$\mathbf{e}(x) = \exp(x\mathbf{A})\boldsymbol{\epsilon} + \int_0^x \exp((x-\xi)\mathbf{A})\boldsymbol{\delta} \, d\xi. \quad (\text{C.5.16})$$

To bound the function \mathbf{e} we notice first that, by properties of the exponential, the product inequality and the Cauchy–Buyankovski–Schwarz inequality for the diagonal matrix with the norm induced by the Euclidean norm, we have

$$\begin{aligned} |\exp(x\mathbf{A})| &\leq |\mathbf{P}^{-1}| |\mathbf{P}| \left(\sum_{i=1}^d |\exp x \lambda_i|^2 \right)^{1/2} \\ &\leq C_1 \exp(x \text{re } \lambda_d), \end{aligned} \quad (\text{C.5.17})$$

where $C_1 = d |\mathbf{P}^{-1}| |\mathbf{P}|$.

We can now distinguish three cases.

Case 1. $\operatorname{re} \lambda_d < 0$; that is, all the eigenvalues have negative real part. Taking norms in (C.5.16) and using the triangle inequality and other basic manipulations, we obtain

$$|\mathbf{e}(x)| \leq C_1 \frac{1 + (|\operatorname{re} \lambda_d| + 1) \exp(x \operatorname{re} \lambda_d)}{|\operatorname{re} \lambda_d|} \varepsilon \quad (\text{C.5.18})$$

for $x \in \mathbb{R}_0^+$. A small exercise shows that

$$C(\lambda_d) := \sup_{x \in [0, \infty)} \frac{(1 + (|\operatorname{re} \lambda_d| + 1)) \exp(x \operatorname{re} \lambda_d)}{|\operatorname{re} \lambda_d|} \leq \begin{cases} 1, & \text{for } \operatorname{re} \lambda_d < -1 \\ 1/|\operatorname{re} \lambda_d|, & \text{for } -1 \leq \operatorname{re} \lambda_d < 0. \end{cases} \quad (\text{C.5.19})$$

Thus (C.5.18)

$$\|\mathbf{e}\|_{L_\infty(\mathbb{R}_0^+)} \left(:= \sup_{x \in [0, \infty)} |\mathbf{e}(x)| \right) \leq C_1 C(\lambda_d) \varepsilon. \quad (\text{C.5.20})$$

It follows that *if* $\operatorname{re} \lambda_d < 0$ *then the perturbation is not amplified in time*. The ODE is called *stable* in this case.

Case 2. $\operatorname{re} \lambda_d > 0$; that is, at least one eigenvalue has positive real part. For simplicity, assume the eigenvalue λ_d is real, i.e., $\lambda_d \in (0, \infty)$. Pick a corresponding eigenvector $\mathbf{v} \neq 0$, with $|\mathbf{v}| = 1$. By the spectral properties of the matrix exponential ?? (??), $\exp(x\mathbf{A})\mathbf{v} = \exp(x\lambda_d)\mathbf{v}$. If we choose the perturbations $\epsilon := \epsilon \mathbf{v}$ and $\delta := \delta \mathbf{v}$, with $\epsilon, \delta \in [0, \varepsilon]$ then (C.5.16) implies

$$\begin{aligned} |\mathbf{e}(x)| &= \left| \epsilon \left(\exp(x\lambda_d) + \delta \int_0^x \exp((x-\xi)\lambda_d) d\xi \right) \mathbf{v} \right| \\ &= \epsilon \exp(x\lambda_d) + \delta \frac{\exp x\lambda_d - 1}{\lambda_d}. \end{aligned} \quad (\text{C.5.21})$$

It follows that the perturbation gets amplified exponentially with respect to time. Even if $\delta = 0$, *an initial perturbation intensity* $\epsilon > 0$ *causes an exponential build up of the error in time*. We say in this case that the ODE is *unstable*.

Case 3. $\operatorname{re} \lambda_d = 0$; this is the boundary case between stability and instability. For $\mathbf{v} \neq \mathbf{0}$, corresponding eigenvector of norm 1, and a similar choice of perturbations as in Case 2.

$$|\mathbf{e}(x)| = \epsilon + x\delta. \quad (\text{C.5.22})$$

The initial perturbation does not get amplified in time, but the perturbation in the ODE does pile up linearly in time. This is a *mildly unstable* ODE.

A similar analysis can be conducted for the case of non-constant coefficients, i.e., when the matrix \mathbf{A} is a function of \mathbf{x} . We omit this analysis as it requires further technical properties of the exponential for matrices.

C.5.3. Stability for $\mathbf{f} \in C^1(D)$. Suppose \mathbf{f} , $\partial_x \mathbf{f}$, $\partial_y \mathbf{f}$ are $C^1(D)$ functions. Here we use the notation

$$\partial_x \mathbf{f}(x, \mathbf{y}) = \frac{\partial \mathbf{f}}{\partial x}(x, \mathbf{y}) \in \mathbb{R}^d \quad \text{and} \quad \partial_y \mathbf{f}(x, \mathbf{y}) = D_y \mathbf{f}(x, \mathbf{y}) \in \mathbb{R}^{d \times d}, \quad (\text{C.5.23})$$

respectively, for the partial derivative of \mathbf{f} with respect to its first argument and the (partial) Jacobian matrix of \mathbf{f} with respect to its second argument.

Differentiating \mathbf{e} and using the ODE's (C.1.6) and (C.4.1) we obtain the error equation

$$\mathbf{e}'(x) = \mathbf{Z}'(x) - \mathbf{Y}'(x) = \boldsymbol{\delta}(x) + \mathbf{f}(x, \mathbf{Z}(x)) - \mathbf{f}(x, \mathbf{Y}(x)) \quad (\text{C.5.24})$$

By the mean value theorem in integral form we know that, for all $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^d$,

$$\begin{aligned} \mathbf{f}(x, \mathbf{y}_1) - \mathbf{f}(x, \mathbf{y}_2) &= \int_0^1 \partial_{\mathbf{y}} \mathbf{f}(x, s\mathbf{y}_1 + (1-s)\mathbf{y}_2)(\mathbf{y}_1 - \mathbf{y}_2) \, ds \\ &= \left(\int_0^1 \partial_{\mathbf{y}} \mathbf{f}(x, s\mathbf{y}_1 + (1-s)\mathbf{y}_2) \, ds \right) (\mathbf{y}_1 - \mathbf{y}_2) \end{aligned} \quad (\text{C.5.25})$$

It follows from (C.5.24) and (C.5.25) that \mathbf{e} satisfies the differential equation

$$\begin{aligned} \mathbf{e}'(x) &= \boldsymbol{\delta}(x) + \mathbf{A}(x)\mathbf{e}(x), \\ \mathbf{A}(x) &:= \int_0^1 \partial_{\mathbf{y}} \mathbf{f}(x, \mathbf{Y}(s) + s\mathbf{e}(s)) \, ds. \end{aligned} \quad (\text{C.5.26})$$

By continuity of $\partial_{\mathbf{y}} \mathbf{f}$ we assume that \mathbf{e} stays small enough as to allow the approximation

$$\mathbf{A}(x) \approx \partial_{\mathbf{y}} \mathbf{f}(x, \mathbf{Y}(x)), \quad (\text{C.5.27})$$

up to higher order terms involving \mathbf{e} . This heuristic leads to studying the growth of the solution IVP

$$\mathbf{e}' = (\partial_{\mathbf{y}} \mathbf{f}) \mathbf{e} + \boldsymbol{\delta} \text{ and } \mathbf{e}(0) = \boldsymbol{\epsilon}. \quad (\text{C.5.28})$$

From the linear case, discussed in C.5.2 when the coefficients are constant, the behaviour of $|\mathbf{e}|$ is dictated by the real part of the eigenvalues of $\partial_{\mathbf{y}} \mathbf{f}(x, \mathbf{Y}(x))$ where \mathbf{Y} is the exact solution. Call these eigenvalues, which may vary with x , $\lambda_1(x), \dots, \lambda_d(x) \in \mathbb{C}$. If $\max_{i \in 1:d} \sup_{x \in \mathbb{R}^+} \operatorname{re} \lambda_i(x) < 0$ then the error stays bounded by a constant factor of $\boldsymbol{\epsilon}$ as $x \rightarrow \infty$.

But if $\operatorname{re}(\lambda_i(x)) \geq 0$ for some i and for x in some interval $(a, b) \subseteq (0, \infty)$, then the error may grow exponentially with time and the ODE is considered *unstable*.

C.5.4. Example. Consider the IVP (C.1.6)–(C.1.7) with

$$f(x, y) = -2\alpha(x-1)y. \quad (\text{C.5.29})$$

In this case we have

$$\partial_{\mathbf{y}} f(x, y) = -2\alpha(x-1) \begin{cases} > 0 & \text{for } x \in (0, 1) \\ \leq 0 & \text{for } x \geq 1. \end{cases} \quad (\text{C.5.30})$$

Although the interval on which $\partial_{\mathbf{y}} f$ is positive is only $(0, 1)$ the effects of an initial perturbation will be felt for long time. This is an early sign that the numerical approximation of this problem is not straightforward: a very good job must be done in the initial transient phase as not to compromise the overall accuracy.

C.5.5. Case where f is globally Lipschitz. Sometime f is not $C^1(D)$ and it is merely Lipschitz. Suppose all we know about f is that it satisfies the global Lipschitz condition (C.2.2). What kind of analysis can we conduct here?

Since $Y'(x) = f(x, Y(x))$ and $Z'(x) = f(x, Z(x)) + \delta(x)$

Subtraction yields

$$e'(x) = Z'(x) - Y'(x) = f(x, Z(x)) - f(x, Y(x)) + \delta(x) \quad (\text{C.5.31})$$

Integrating on $[0, x]$

$$\begin{aligned} e(x) &= e(0) + \int_0^x e'(t) dt \\ &= \epsilon + \int_0^x f(t, Z(t)) - f(t, Y(t)) dt + \int_0^x \delta(t) dt \end{aligned} \quad (\text{C.5.32})$$

Note that here we should resist the temptation of applying the mean value theorem, because the differentiability assumptions necessary for its application are not fulfilled. We have to proceed directly and try to use the Lipschitz condition instead. Take norms in (C.5.32) and apply the triangle inequality for sums and integrals, $|\int_I g(x) dx| = \int_I |g(x)| dx$, as to obtain

$$e(x) := |e(x)| \leq |\epsilon| + \int_0^x |\delta(t)| dt + \int_0^x |f(t, Z(t)) - f(t, Y(t))| dt \quad (\text{C.5.33})$$

Observe that by the Lipschitz condition (C.2.2) we have

$$\int_0^x |f(t, Z(t)) - f(t, Y(t))| dt \leq K |Z(t) - Y(t)| \quad (\text{C.5.34})$$

It follows that

$$e(x) \leq |\epsilon| + \int_0^x |\delta| dt + K \int_0^x e(t) dt \quad (\text{C.5.35})$$

This inequality, which is typical of ODE's, calls for the application of the Gronwall Lemma.

C.5.6. Lemma (Gronwall's Lemma in integral form). Let $e, \chi, \varphi \in C^0[0, T]$ with $\chi, \varphi \geq 0$. If

$$e(x) \leq \varphi(x) + \int_0^x \chi(t) e(t) dt, \quad \forall x \in (0, T], \quad (\text{C.5.36})$$

then

$$e(x) \leq \varphi(x) + \int_0^x \chi(t) \varphi(t) \exp\left(\int_t^x \chi(s) ds\right) dt, \quad \forall x \in (0, T]. \quad (\text{C.5.37})$$

Proof Let $R(x) := \int_0^x \chi(t) e(t) dt$ and $X(x) = \int_0^x \chi(t) dt$. Then

$$R'(x) = \chi(x) e(x) \leq \chi(x) (\varphi(x) + R(x)), \quad (\text{C.5.38})$$

Therefore

$$R'(x) - \chi(x) R(x) \leq \chi(x) \varphi(x). \quad (\text{C.5.39})$$

Applying the integrating factor method we see that

$$\frac{d}{dx} [\exp(-X(x))R(x)] = \exp(-X(x))(R'(x) - \chi(x)R(x)) \leq \chi(x)\varphi(x)\exp(-X(x)), \quad (\text{C.5.40})$$

for all $x \in [0, T]$. Integrating on $[0, x] \subseteq [0, T]$ we obtain

$$(\exp(-X(x))R(x)) \leq \exp(-X(0))R(0) + \int_0^x \chi(t)\varphi(t)\exp(-X(t))dt. \quad (\text{C.5.41})$$

Since $R(0) = 0$ we obtain

$$R(x) \leq \int_0^x \chi(t)\varphi(t)\exp(X(x) - X(t))dt \quad (\text{C.5.42})$$

Finally, inequality (C.5.36) implies

$$e(x) \leq \varphi(x) + \int_0^x \chi(t)\varphi(t)\exp\left(\int_t^x \chi(s)ds\right)dt, \quad (\text{C.5.43})$$

for all $x \in (0, T]$, as we wanted to prove. \square

Inequality (C.5.35) can be now made to play the role of (C.5.36) by taking with $\varphi(x) = |\epsilon| + \int_0^x |\delta(t)|dt$ and $\chi \equiv K$. The Gronwall Lemma implies then that

$$e(x) \leq |\epsilon| + \int_0^x |\delta(t)|dt + K \int_0^x \left(|\epsilon| + \int_0^t |\delta(s)|ds \right) \exp(K(x-t))dt. \quad (\text{C.5.44})$$

To finish, observe that

$$\int_{t_1}^{t_2} |\delta(s)|ds \leq \epsilon |t_2 - t_1| \leq \epsilon x, \quad \forall t_1, t_2 \in [0, x] \quad (\text{C.5.45})$$

Thus

$$\begin{aligned} e(x) &\leq (1+x) \left(1 + K \int_0^x \exp(K(x-t))dt \right) \epsilon \\ &\leq (1+x) \exp(Kx) \epsilon, \end{aligned} \quad (\text{C.5.46})$$

for all $x \in [0, T]$.

C.5.7. Conclusions. If the perturbation intensity ϵ tends to 0, then the perturbation error $e(x)$ tends to 0 too, for each fixed $x > 0$. But the constant depends exponentially on x , which means that the speed of convergence may deteriorate (become slower) very fast as x increases. In particular, the convergence of the error to zero is uniform on any bounded interval $[0, T]$ but it is not necessarily so on $[0, \infty)$. Lipschitz continuity alone is not enough to give any finer bounds on the error.

C.5.8. Exercise (Gronwall's Lemma in differential form). *There is a more "direct" way to prove the stability bound (C.5.46) than the one presented in the lectures, by using the Gronwall Lemma in its differential form variant, when the time perturbation vanishes, $\delta = 0$ and only the initial perturbation ϵ is there.*

To obtain the bound, apply $e(x) \cdot$ (the scalar product) on both members of

$$e'(x) = f(x, Z(x)) - f(x, Y(x)), \quad \forall x \in [0, T], \quad (\text{C.5.47})$$

use the Cauchy–Bunyakovskii–Schwarz inequality and the Lipschitz continuity of \mathbf{f} in its second argument to obtain the Gronwall hypothesis bound:

$$\varphi'(x) \leq 2K \varphi(x) \quad \forall x \in [0, T], \quad (\text{C.5.48})$$

where $\varphi = |\mathbf{e}|^2 = \mathbf{e} \cdot \mathbf{e}$.

(a) Prove, using the integrating factor technique, that

$$\varphi(x) \leq e^{2Kx} \varepsilon \quad \forall x \in [0, T] \quad (\text{C.5.49})$$

and conclude that

$$|\mathbf{e}(x)| \leq e^{Kx} \varepsilon \quad \forall x \in [0, T]. \quad (\text{C.5.50})$$

(b) Consider now the case where $\delta \neq \mathbf{0}$, retrace the proof, by using the Young inequality to control the $\mathbf{e} \cdot \delta$ term

$$x y \leq \alpha x^2 + \frac{1}{4\alpha} y^2, \quad (\text{C.5.51})$$

valid for any fixed parameter $\alpha > 0$.

C.5.9. Exercise (stability for dissipative maps). A map $\mathbf{f} : \Omega \rightarrow \Omega$ ($\Omega \subseteq \mathbb{R}^d$, $d \in \mathbb{N}$) is called (globally) monotone dissipative on Ω if there exists a $c > 0$ (called the dissipation coefficient)

$$(\mathbf{f}(\mathbf{z}) - \mathbf{f}(\mathbf{y})) \cdot \mathbf{z} - \mathbf{y} \leq -c |\mathbf{z} - \mathbf{y}|^2 \quad \forall \mathbf{z}, \mathbf{y} \in \Omega. \quad (\text{C.5.52})$$

(a) Show that $\mathbb{R} \ni x \mapsto -(x^3 + x)$ is monotone dissipative according to this definition.

(b) Show that if $f : \mathbb{R} \rightarrow \mathbb{R}$ is differentiable and $f'(x) < -c$, then f is monotone dissipative.

(c) Find a condition on the parameter $a \in \mathbb{R}$ such that the linear map

$$\begin{aligned} \mathbf{f} : \quad \mathbb{R}^2 &\rightarrow \mathbb{R}^2 \\ (u, v) &\mapsto \begin{bmatrix} a & -1 \\ 1 & a \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned} \quad (\text{C.5.53})$$

is monotone dissipative on \mathbb{R}^2 .

(d) Suppose \mathbf{f} is a dissipative map on Ω , let $\mathbf{y}_0 \in \Omega$ and suppose that \mathbf{Y} and \mathbf{Y}_ε solve IVP's for the same ODE

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}) \quad (\text{C.5.54})$$

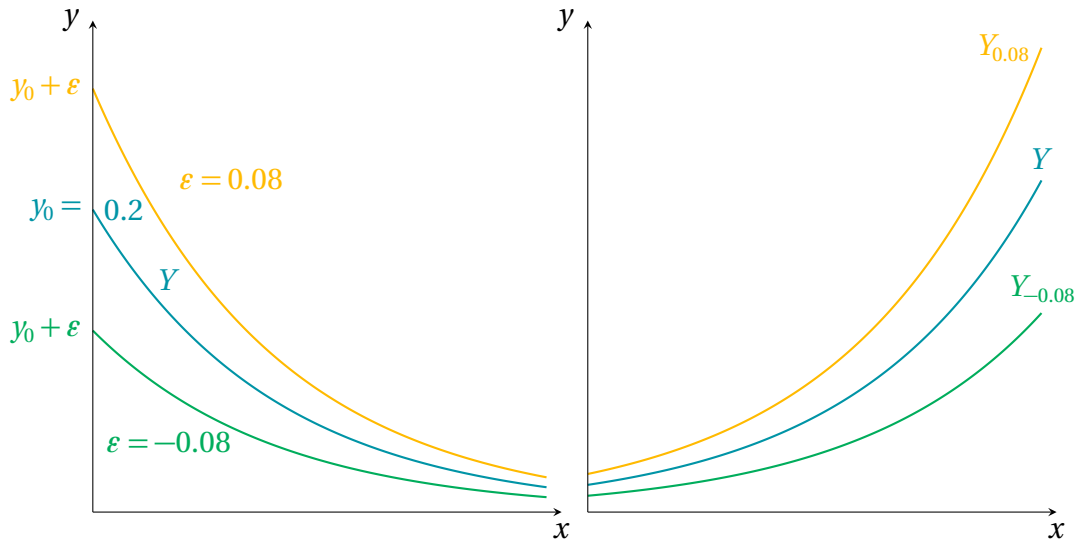
but with the two respective different initial conditions

$$\mathbf{Y}(0) = \mathbf{y}_0 \text{ and } \mathbf{Y}_\varepsilon(0) = \mathbf{y}_0 + \boldsymbol{\varepsilon}. \quad (\text{C.5.55})$$

Using the dissipative nature of \mathbf{f} and an integrating factor technique, show that

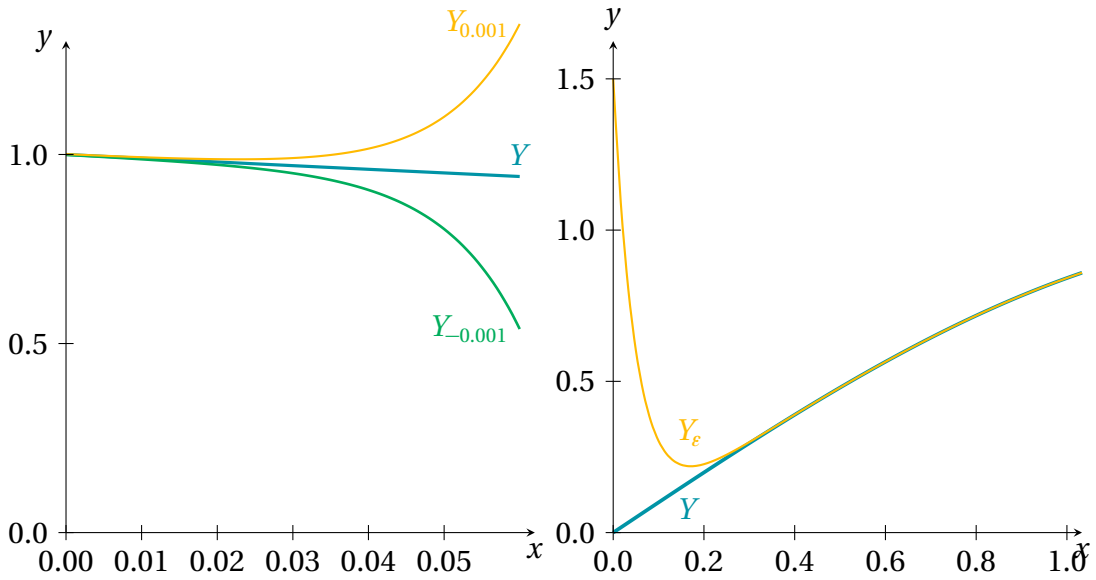
$$|\mathbf{Y}_\varepsilon(t) - \mathbf{Y}(t)| \leq \exp(-ct) |\boldsymbol{\varepsilon}| \quad \forall t > 0. \quad (\text{C.5.56})$$

(e) Explain why “dissipative” comes in the name of such maps, in relation to what happens to the perturbation $\boldsymbol{\varepsilon}$ as time t grows.



(a) IVP's with the scalar linear ODE $\dot{y} = \lambda y$, with $\lambda < 0$. The difference between the solution $Y(x)$ and the perturbed solution $Y_{\varepsilon}(x)$ narrows as x grows.

(b) Same ODE but with $\lambda > 0$. The 0 solution is unstable and the gap between $Y(x)$ and its perturbation $Y_{\varepsilon}(x)$ widens as $x \rightarrow \infty$.



(c) Solutions of the relatively unstable IVP from Example C.1.6: even for very small perturbations, $\varepsilon = \pm 0.001$, the relative error $(Y_{\varepsilon} - Y)/Y$ cannot be controlled in terms of problem data (i.e., initial value and other parameters) for all times.

(d) Solutions of the scalar multiscale problem in Example C.1.8: a rapid change at the beginning of the time interval makes place to a slowly oscillating solution. There are two independent time-scales, that a good method should pick up by wasting as less as possible computational effort.

FIGURE 1. Various linear scalar problems provide examples of stable and unstable situations. Multiscale phenomena are detected even in linear problems.

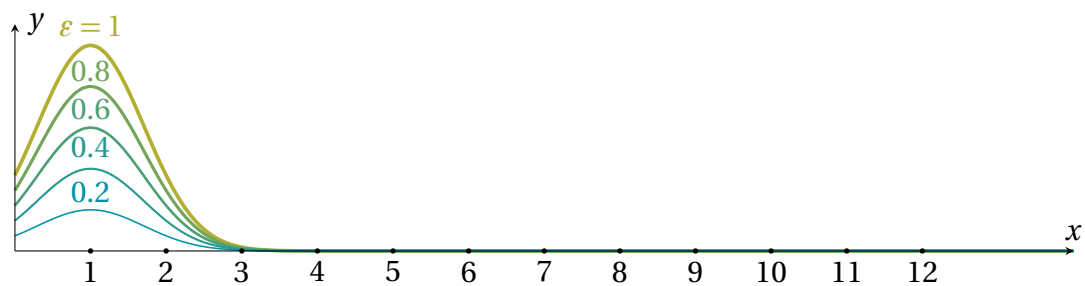


FIGURE 2. Solutions of the stiff IVP's of Example C.1.10 where the stiffness is exhibited in the time interval $[0,3]$ as the initial perturbation ranges in $\varepsilon = 0.2, \dots, 1.0$.

APPENDIX F

Coding hints

Computer coding (also known as hacking)¹ lies at the heart of a central one in scientific computing and an essential part of numerical analysis.

In this appendix (or chapter) we gather some ragtag collection of coding hints.

F.1. Generalities

In theory, there is no big difference between the various programming languages, as the most important capability of the scientific computing researcher or user is to understand the algorithms and the mathematics and the science lying behind these algorithms.

However, in practice (and programming *is the practice* of computer science), the choice of the programming language is a problem. In scientific computing, especially numerical analysis, the choice of programming language is a problem for beginners and experienced alike: beginners should not be bothered with the technicalities of a computing language (e.g., functional vs. object oriented) and must choose a language that is “easy” to code in. Free-choice is not always a good idea.

Numerical analysis students are lucky in that a language (mostly Matlab[®]) is imposed on them by their teacher, and they have to learn it and use it.

F.2. Matlab and Octave

F.2.1. Matlab[®]. If you take a first course in numerical analysis, almost surely, you will be introduced to Matlab[®]. I personally am not a big fan of Matlab^{®2}, but there are three good reasons for considering Matlab[®]:

- (i) Matlab[®] is available to you, most likely via a site-license to your school or employer,

¹I like to use “hacker” and “to hack” in their original English meaning which mean, respectively, someone who likes developing good and useful computer code. The term has been distorted by journalists, which have attributed a negative connotation and use the term “hacker” to indicate a “cracker”, i.e., a person who likes “to crack” into IT systems.

²Matlab[®] was born out of the best intentions of Cleve Moler to write a user-friendly interface to linpack and eispack. For this Moler deserves the highest credit for his contribution to the diffusion and teaching of scientific computing. Unfortunately Matlab[®] is also the example of an academic project born with the best intentions, and which could have benefitted immensely from becoming open-source and free, that was subsequently transformed, with the help of Jack Little and Steve Bangert into a commercial venture aimed less at facilitating scientific computing and more at milking academic institutions for a product that has equally valid (if not better in many aspects) free and open-source software competitors, such as Octave and Scilab; which I personally recommend as a cheap way to get into scientific computing rather than spending money for Matlab[®]. On the other hand Matlab[®], which has some very nice features that make learning scientific computing easy, is available at many higher-education institutions and, given it is already there, it would be stupid not to use it.

- (ii) this is your first experience in numerical analysis or scientific computing and you have never programmed in any computer language,
- (iii) your teacher (or boss) *requires* it.

If all the above are fulfilled you should go with Matlab® without second thoughts. (Usually iii implies i and ii and is, of course, a sole reason for using Matlab®.) As a proselytising believer in software freedom I do not subscribe to iii, so I would relax my statement as follows if i and ii are fulfilled, go with Matlab®, but keep in mind that the world offers you much more than this. Think of it this way: you are learning to drive, you (or your parents, or the taxpayers) are paying a driving instructor to teach you driving, and your instructor provides you with a car. You accept it, you learn to drive and then you move on.

Bottom line: if you are aiming at a scientific computing or numerical analysis career, follow Paul Halmos advice concerning Matlab®: learn Matlab®, absorb it, and forget it. Then use your knowledge to move to something more powerful and efficient (e.g., Python or Java and compiled languages such as C++, C, or even Fortran which remains one of the most efficient languages on many architectures when it comes to pure computing).

An added advantage of Matlab® is the plethora of interesting academic code written in it.

F.2.2. Octave. Octave is thought by many as the poor person's Matlab®. In fact, generally speaking the Octave user is richer (at least a hundred pounds or so) because Octave is (legally) free of charge. It is in fact covered by a Gnu Public License which means that download and usage are completely free. Where one must ever pay, it is for the print version of manual Eaton, Bateman and Hauberg, 2011. Octave's syntax is very similar to Matlab®, in fact current versions of Octave treat Matlab® syntax as a subset (with some initial tweaking that disables some of Octave's "smart" features: see `-braindead` option). Note however that this goes one way: Matlab® may choke on an Octave file.

F.2.3. Object oriented Matlab® and Octave. Although initially created for a flat, functional, Fortran-like type programming, Matlab® has evolved as to offer an object oriented style, for those who prefer it. Octave, following its maintainers's philosophy, shadows this.

Matlab® class jargon buster. MATLAB classes use the following words to describe different parts of a class definition and related concepts.

class definition: Description of what is common to every instance of a class.

property: Data storage for class instances.

method: Special functions that implement operations that are usually performed only on instances of the class

event: Messages that are defined by classes and broadcast by class instances when some specific action occurs

attribute: Values that modify the behavior of properties, methods, events, and classes

listener: Objects that respond to a specific event by executing a callback function when the event notice is broadcast

object: Instances of classes, which contain actual data values stored in the objects' properties

subclass: Classes that are derived from other classes and that inherit the methods, properties, and events from those classes (subclasses facilitate the reuse of code defined in the superclass from which they are derived).

superclass: Classes that are used as a basis for the creation of more specifically defined classes (i.e., subclasses). Packages Directories that define a scope for class and function naming

F.3. Octave

First steps.

F.4. Python: NumPy and SciPy

Python is one of the most successful computing languages at par with Java. It can be used for a multitude of purposes and one of them is scientific computation. Python's power derives from the fact that:

- ★ it is free software (GPL),
- ★ it is widely used by the scientific computing community,
- ★ it is widely used by the general computing community,
- ★ it is very well documented,
- ★ it can be extended by the user,
- ★ it is natively object oriented (unlike Matlab® or Octave where object-orientation is an afterthought).

The one difficulty that newbies find with Python is its rather rigid rules in writing code. What seems initially an unnecessary over-regulation pays in the longer run as it encourages a somewhat uniform coding style across programmers and developers which makes other people's code readable.

F.4.1. References. The literature on Python is huge, including that specialised for numerical analysis and scientific computation. Here are some of the sources that I have found useful:

Langtangen, 2011: A detailed introduction to Python as a Scientific Computation tool. Pros: the style is patient and meticulous, the exercises and problems are excellent. The con is that it is rather lengthy (it would take a busy person months to go through the whole thing) and difficult to use as a manual. In nutshell, this is a perfect textbook for teaching, but not for the impatient.

python.org, 2015: The prime reference for anyone (already computer literate) that would like to take a first stroll through Python. I use it to translate some of my octave/matlab code into python for teaching (myself and the students) Python. Most of these notes are based on this tutorial.

F.4.2. First slithers: interactive mode. Python, just as Matlab® and Octave, can be run interactively or as a code. There are many interactive shells or interfaces. If you, like myself, are a Unix shell-animal that lives primarily in a Linux or Mac OS X environment you will find it convenient to launch Python from the shell (here bash) by simply issuing


```
bash» python
```

You should now be able to see the *Python prompt* which looks like this

```
>>>
```

Two important interactive commands are

```
>>> exit()
```

which terminates the session and brings you back the initial shell, and

```
>>> help()
```

which allows you to access quick information about keywords, functions, concepts and more. `help()` launches an interactive session; if you just want to look-up one command try

```
>>> help(<string>)
```

where `<string>` stands for a Python keyword, command, function or concept. If this is the first time you opened help I bet you *read* the welcome lines (those that pop up after you hit enter). Didn't you? To exit the help session you need to type

```
help> quit()
```

F.4.3. Programming. A Python code can be a script or a module.

Sample script. One of the simplest python codes that does something more than greeting you prints some of the first Fibonacci numbers.

Printout of file `fibonacci.py`

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
a, b = 0, 1
while b < 10:
    print b,
    a, b = b, a+b
```

Sample function. Python can be used as a *functional language*, as illustrated by the following Python script.

Printout of file `fibonacciFun.py`

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
def fibo(a,b,n):
    """compute and print Fibonacci sequence with seeds a,b up to n"""
    while b < n:
        print b,
        a, b = b, a+b

fibo(0,1,10)
```

Sample module. Often one wants to define a function in one file and call that function from one (or, better, more) files, or from an interactive mode. A file that defines functions to be used by other files (or sessions) is called a module. Here is a module:

Printout of file fibonacciMod.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
def fibo(a,b,n):
    """compute and print the Fibonacci sequence with seeds a,b up to n"""
    fiboseq = []
    while b < n:
        fiboseq.append(b)
        print b,
        a, b = b, a+b
    return fiboseq

def fibona(a,b,n):
    """compute and return the Fibonacci sequence with seeds a,b, up to n"""
    fibosequence = []
    while b < n:
        fibosequence.append(b)
        a, b = b, a + b
    return fibosequence
```

Using a module interactively. There are (at least) two alternative ways of using a module interactively.

(a) *Full import* In this case the whole module is imported by issuing

```
>>> import fibonacciMod
```

When this is done the single functions in the module must be called as methods on that module as an object, i.e., to use fibo we issue

```
>>> fibonacciMod.fibo(1,2,3)
```

(b) *Selective import* Here the user imports only the functions that are needed by selecting them as follows

```
>>> from fibonacciMod import fibo, fibona
```

F4.4. Remark (my updates don't work?) A frustrating experience for the python beginner is when a bug sneaks in a module and, after the bug gets fixed, python shell keeps on giving the same error as if the module was never changed!

That's because inspite of the file being changed, the changes must be picked up by the interactive session. But since the file has been already imported, it cannot be unimported (that doesn't exists in Python), but it can be reloaded. For example suppose there was an error in the module fibonacciMod.py and that was fixed. First the module needs to be reloaded by issuing

```
>>> fibonacciMod = reload(fibonacciMod)
```

and then, if some of fibonacciMod.py's functions had been imported selectively, say fibona, they need to be reimported as normally importing

```
>>> from fibonacciMod import fibona
```

Bibliography

- [1] R. Abraham, J. E. Marsden and T. Ratiu. *Manifolds, tensor analysis, and applications*. Second. Vol. 75. Applied Mathematical Sciences. Springer-Verlag, New York, 1988, pp. x+654. ISBN: 0-387-96790-7. DOI: 10.1007/978-1-4612-1029-0. URL: <http://dx.doi.org/10.1007/978-1-4612-1029-0> (cit. on p. 130).
- [2] Ben Andrews and Christopher Hopper. *The ricci flow in Riemannian geometry a complete proof of the differentiable 1/4-pinching sphere theorem*. English. Berlin; Heidelberg; New York: Springer, 2011. ISBN: 9783642162862 364216286X. URL: <http://rave.ohiolink.edu/ebooks/ebc/9783642162862> (visited on 27/01/2015) (cit. on p. 143).
- [3] Kendall E. Atkinson. *An introduction to numerical analysis*. Second. New York: John Wiley & Sons Inc., 1989, pp. xvi+693. ISBN: 0-471-62489-6 (cit. on p. 84).
- [4] L. K. Babadzanjan. "Existence of the continuations in the N -body problem". en. In: *Celestial mechanics* 20.1 (July 1979), pp. 43–57. ISSN: 0008-8714, 1572-9478. DOI: 10.1007/BF01236607. URL: <http://link.springer.com.ezproxy.sussex.ac.uk/article/10.1007/BF01236607> (visited on 21/01/2015) (cit. on p. 16).
- [5] Florin Diacu. "The solution of the n -body problem". In: *The Mathematical Intelligencer* 18.3 (1996), pp. 66–70. ISSN: 0343-6993. DOI: 10.1007/BF03024313. URL: <http://www.ams.org/mathscinet-getitem?mr=1412994> (visited on 21/01/2015) (cit. on p. 16).
- [6] John W. Eaton, David Bateman and Søren Hauberg. *GNU Octave*. Edition 3 for Octave version 3.6.1. Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301–1307, USA, Feb. 2011 (cit. on p. 198).
- [7] Wendell Fleming. *Functions of several variables*. Second. Undergraduate Texts in Mathematics. New York: Springer-Verlag, 1977, pp. xi+411 (cit. on pp. 135, 142).
- [8] David F. Griffiths and Desmond J. Higham. *Numerical methods for ordinary differential equations*. Springer Undergraduate Mathematics Series. Initial value problems. Springer-Verlag London, Ltd., London, 2010, pp. xiv+271. ISBN: 978-0-85729-147-9. DOI: 10.1007/978-0-85729-148-6. URL: <http://dx.doi.org/10.1007/978-0-85729-148-6> (cit. on p. 6).
- [9] Vasile I. Istratescu. *Fixed point theory*. Vol. 7. Mathematics and its Applications. An introduction, With a preface by Michiel Hazewinkel. D. Reidel Publishing Co., Dordrecht-Boston, Mass., 1981. ISBN: 90-277-1224-7. URL: <http://www.ams.org/mathscinet-getitem?mr=620639> (visited on 21/01/2015) (cit. on p. 16).

- [10] Stephan Karamardian, ed. *Computing Fixed Points with Applications*. Proceedings of Conference in Clemson, South Carolina 1, 1976. New York [u.a.]: Academic Press, 1977. ISBN: 0-12-398050-X (cit. on p. 16).
- [11] Tim C. T. Kelley. *Iterative methods for optimization*. Vol. 18. Frontiers in Applied Mathematics. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1999, pp. xvi+180. ISBN: 0-89871-433-8 (cit. on p. 6).
- [12] Omar Lakkis. *Introduction to Pure Mathematics*. Online lecture notes. published freely online under Creative Commons license. University of Sussex, Dec. 2011. URL: https://dl.dropboxusercontent.com/u/15751353/omar_lakkis-mathematics/Notes/General/Introduction_to_Pure_Mathematics--Lakkis--2011.pdf (cit. on p. 1).
- [13] Hans Petter Langtangen. *A Primer on Scientific Programming with Python*. Texts in Computational Science and Engineering. Springer, 2011. ISBN: 9783642183652. URL: <http://books.google.co.uk/books?id=Hi1KVomG148C> (cit. on p. 199).
- [14] Elliott H. Lieb and Michael Loss. *Analysis*. Second. Vol. 14. Graduate Studies in Mathematics. Providence, RI: American Mathematical Society, 2001, pp. xxii+346. ISBN: 0-8218-2783-9 (cit. on p. 136).
- [15] Joram Lindenstrauss and David Preiss. "On Fréchet differentiability of Lipschitz maps between Banach spaces". In: *Annals of Mathematics* 157.1 (2003), pp. 257–288. URL: <http://arxiv.org/abs/math/0402160> (visited on 27/01/2015) (cit. on p. 143).
- [16] python.org. *The Python Tutorial Python 2.7.9 documentation*. URL: <https://docs.python.org/2/tutorial/index.html> (visited on 02/01/2015) (cit. on p. 199).
- [17] Alfio Quarteroni, Riccardo Sacco and Fausto Saleri. *Numerical mathematics*. Second. Vol. 37. Texts in Applied Mathematics. Berlin: Springer-Verlag, 2007, pp. xviii+655. ISBN: 978-3-540-34658-6; 3-540-34658-9 (cit. on p. 84).
- [18] Herbert Scarf. "The approximation of fixed points of a continuous mapping". In: *SIAM Journal on Applied Mathematics* 15.5 (Sept. 1967), pp. 1328–1343. ISSN: 0036-1399. URL: <http://www.jstor.org/stable/2099173> (visited on 21/01/2015) (cit. on p. 16).
- [19] L. Ridgway Scott. *Numerical analysis*. <http://people.cs.uchicago.edu/~ridg/newna/nalrs.pdf>. Princeton, NJ: Princeton University Press, 2011, pp. xvi+325. ISBN: 978-0-691-14686-7. URL: <http://www.worldcat.org/oclc/679940621> (cit. on pp. 6, 21, 24, 76).
- [20] Karl F. Sundman. "Mémoire sur le problème des trois corps". In: *Acta Mathematica* 36.1 (1913), pp. 105–179. ISSN: 0001-5962. DOI: 10.1007/BF02422379. URL: <http://www.ams.org/mathscinet-getitem?mr=1555085> (visited on 21/01/2015) (cit. on p. 16).
- [21] Noel M. Swerdlow. "Kepler's Iterative Solution to Kepler's Equation". In: *Journal for the History of Astronomy* 31 (Nov. 2000), p. 339 (cit. on p. 21).
- [22] Qiu Dong Wang. "The global solution of the n -body problem". In: *Celestial Mechanics & Dynamical Astronomy. An International Journal of Space Dynamics* 50.1 (1991), pp. 73–88. ISSN: 0923-2958. DOI: 10.1007/BF00048987. URL: <http://link.springer.com/article/10.1007/BF00048987> (visited on 21/01/2015) (cit. on p. 16).

Index

- k -linear, 64
- apriori, 10, 11
- Abelian, 51
- Abelian group, 51
- adjoint, 55
- adjoint operator, 68
- algebra, 57
- algebraic basis, 56
- algebraically closed field, 59
- alternating, 64
- antisymmetric, 64
- aposteriori, 10, 11
- approximation, 29
- approximation algorithms, 2
- attribute, 132
- back-substitution, 3
- backwards, 32
- basis, 56
- bilinear forms, 64
- blockwise matrix algebra, 37
- class definition, 132
- column vector, 54
- column-index, 29
- commutative, 51
- commutative group, 51
- complexity, 35
- computational cost, 35
- constant polynomial, 57
- constructive proof, 38
- contraction, 12, 13, 18
- contractive, 13
- converge linearly, 15
- converge quadratically, 15
- convergence
 - quadratic, 15
- convergent, 65
- determinants, 64
- diagonal, 29
- diagonally normalised matrix, 30
- Dirac delta, 60
- Dirac mass, 60
- direct method, 29
- direct solvers, 29
- empty summation, 31
- entries, 54
- entry, 54
- error, 8
- error analysis, 8
- error bound, 10, 11
- error estimate, 10, 11
- error reduction, 10, 11
 - linear, 15
 - quadratic, 15
- Euclidean algorithm, 1
- Euclidean norm, 67
- event, 132
- exact, 2
- exact method, 3
- exact solution, 1
- field, 52
- finite dimensional vector space, 56
- fixed-point, 9
- fore-substitution, 31
- Functional Analysis, 68
- functional language, 134
- Gaussian elimination, 2
- geometric series, 3
- geometric series method, 3, 5
- greatest common divisor, 1
- group, 51
- Hamel basis, 56
- highest common factor, 1
- Hilbert dimension, 67
- Hilbert space, 67
- Hilbertian norm, 66
- induction, 37
- inner product, 66
- inner product space, 66
- inner products, 64

- input, 2
- interactive mode, 133
- invertible, 33, 47
- iterates, 10
- iteration, 5
- iterative solvers, 29

- Kronecker's delta notation, 55

- left inverse, 61
- left invertible, 61
- limit, 66
- limit of a sequence, 66
- linear combination, 56
- linear convergence, 15
- linear convergence rate, 14
- linear error reduction, 15
- linear form, 59
- linear forms, 64
- linear map, 59
- linear operator, 59
- Lipschitz continuous, 11
- listener, 132
- lower triangular, 29
- LU factorisation, 38, 48
- LU-cky, 37, 48

- matrix, 54
- matrix inverse, 34
- matrix notation, 54
- matrix-matrix multiplication, 55
- method, 29, 132
- multilinear form, 64
- multiple root, 23

- Neumann series method, 3
- Newton–Raphson iteration, 22
- norm, 65
- normalised, 30
- numerical, 2

- object, 133
- operation count, 35
- output, 2

- point spectrum, 62
- polynomial, 57
- polynomial-zero, 57
- postinverse, 61
- postinvertible, 61
- preinvertible, 62
- property, 132
- pseudocode, 33
- pull-back, 64

- quadratic convergence, 15
- quadratic convergence rate, 14
- quadratic error reduction, 15

- recursion, 37
- recursive, 34
- recursive algorithm, 34
- recursive back-substitution, 35
- RIDCIU, 54
- RIFCIL, 54
- right invertible, 62
- row-index, 29
- row-index-down-col-index-up, 54
- row-index-first-col-index-last, 54

- scalar–vector multiplication, 53
- separable Hilbert space, 67
- sesquilinear form, 65
- simple root, 23
- singular value, 63
- solver, 29
- span, 56
- spanned space, 56
- spectral value, 63
- spectrum, 63
- stability, 24
- subclass, 133
- substituting, 32
- superclass, 133
- symmetric, 64

- table, 54
- terminate, 2
- tester code, 33
- triangular, 29

- un-LU-cky, 37
- upper triangular, 29
- upper triangular matrix, 32
- upwards, 32

- vector of \mathbb{K}^n , 54
- vector space, 53

- well-defined output, 2