Numerical Analysis 2

Complete course

2016

by Omar Lakkis

Copyright and copyleft notice

https://plus.google.com/103353428445025203078

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").

Contents

Copyright and copyleft notice 1
About the cover picture 1
What's this? 13
Disclaimer 13
Recommended reading list 13

Introduction 1
Exact vs. approximate algorithms 1
0.0.1. Example (Euclidean algorithm) 1
0.0.2. Example (Gaussian elimination) 2
0.0.3. Example (approximate method: "real" division) 3

Chapter 1. Gaussian elimination 7
1.1. Triangular matrices 8
1.1.1. Definition of triangular matrices 8
1.1.2. Definition of sets of triangular matrices 8
1.1.3. Proposition 8
1.1.4. Remark (stars and noughts) 8
1.1.5. Example 9
1.2. Solving triangular systems 9
1.2.1. Fore-substitution 9
1.2.2. Back-substitution 10
1.2.3. Example 11
1.2.4. Problem 11
1.2.5. Algorithm (Back-substitution) 11
1.2.6. Implementation of back-substitution 11
1.2.7. Back in block 12
1.2.8. Algorithm (Back-substitution) 13
1.2.9. A recursive implementation of back-substitution 13
1.2.10. Operation count for back-substitution 13
1.3. Gaussian elimination 14
1.3.1. The 2×2 row reduction procedure 14
1.3.2. 2×2 LU factorisation 14
1.3.3. Blockwise algebra 15
1.3.4. Definition of LU-cky matrix 15
1.3.5. Example 16
1.3.6. Theorem (LU factorisation) 16
1.3.7. Algorithm (recursive LU factorisation) 17
1.3.8. A recursive implementation of LU 17

1.3.9. Iterative LU 18
1.3.10. Remark (compact LU) 19
1.3.11. Operation count in LU factorisation 19
1.4. Pivoting and PLU factorisation 20
1.4.1. What about un-LU-cky matrices? 20
1.4.2. Permutation matrices 20
1.4.3. Inversion of lower triangular matrices 21
1.4.4. Problem (permutation to normal-lower-triangle commutator) 21
1.4.5. The Gaussian elimination step 22
1.4.6. Group structure 22
1.4.7. Example (PLU factorisation) 23
1.5. Doolittle's LU algorithm 24
1.5.1. Derivation 24
1.6. Stability and condition numbers 25
1.6.1. Definition of condition number 25
Exercises and problems on linear systems and Gaussian elimination 26

Chapter 2. Matrix analysis 33
2.1. Scalar product spaces 33
2.1.1. Euclidean spaces as models of finite dimensional Hilbert spaces 33
2.1.2. Definition of adjoint operator 34
2.1.3. Proposition (existence and uniqueness of the adjoint operator) 34
2.1.4. Definition of adjoint matrix 34
2.1.5. Remark (confused by adjoint and adjoint?) 34
2.1.6. Remark (transpose and adjoint) 35
2.2. Orthogonality 35
2.2.1. Definition of orthogonal operator 35
2.2.2. Proposition 35
2.2.3. Definition of unitary or orthogonal matrix 35
2.2.4. Definition of orthonormal system 35
2.2.5. Definition of orthonormal basis 35
2.2.6. Proposition (chracterisation of unitary matrices) 35
2.2.7. Proposition (spectral properties of unitary matrices) 36
2.2.8. Example 36
2.2.9. Definition of unitary and orthogonal groups 36
2.2.10. Proposition (group structure) 36
2.2.11. Example (plane rotations) 36
2.2.12. Example (Householder transformation) 36
2.3. Schur factorisation 37
2.3.1. Definition of Householder matrix 37
2.3.2. Proposition (properties of Householder matrices) 37
2.3.3. Lemma (reflectivity of Householder matrix) 37
2.3.4. Exercise 37
2.3.5. Theorem 38
2.4. Self-adjoint operators and matrices 39
2.4.1. Definition of self-adjoint, symmetric and Hermitian operators and matrices 39

2.4.2. Exercise (self-adjoint operators and matrices) 39
2.4.3. Exercise (self-adjoint operators from a vector space) 39
2.4.4. Exercise 39
2.4.5. Exercise 39
2.5. Self-adjoint positive (SPD) matrices 39
2.5.1. Definition of positive definite operator or matrix 39
2.5.2. Exercise 39
2.5.3. Definition of SPD operator or matrix 39
2.5.4. Exercise (algebra of SPD matrices) 40
2.5.5. Remark (SPD($\mathbb{K}^n$) is not an algebra nor a vector space) 40
2.5.6. Exercise (the Grammian is SPD) 40
2.5.7. Definition of normal matrix 40
2.5.8. Exercise 40
2.5.9. Theorem (spectral) 40
Exercises and problems on matrix analysis and symmetry 41
2.3.1. Problem 41

Chapter 3. Orthogonality based factorisations 43
3.1. Cholesky's factorisation 44
3.1.1. Cholesky's factorisation 44
3.1.2. Theorem (Cholesky factorisation) 44
3.1.3. Remark 44
3.1.4. Problem (Uniqueness of Cholesky's factorisation) 44
3.1.5. Derivation of Cholesky's algorithm 45
3.1.6. Problem (SPD matrices and their submatrices) 45
3.1.7. Algorithm (recursive Cholesky) 46
3.1.8. Algorithm (an iterative Cholesky) 46
3.1.9. Remark (Cholesky vs UHU) 46
3.1.10. Remark (real Cholesky and UTU vs UHU) 46
3.1.11. Remark (complex Cholesky vs real Cholesky) 46
3.1.12. Remark (real Cholesky vs LLT) 46
3.2. QR factorisation 47
3.2.1. Theorem 47
3.2.2. Algorithm (recursive QR) 48
3.2.3. Problem (a recursive QR code) 48
3.2.4. Problem (iterative QR code) 48
3.2.5. Remark (QR vs. Schur) 48
Exercises and problems on direct factorisation methods 49

Chapter 4. Least squares polynomial approximation 51
4.1. Linear least-squares approximation 51
4.1.1. How do we define a best fit? 51
4.1.2. Theorem (Hankel's matrix invertibility) 52
4.1.3. Example 53
4.2. General linear least squares 53
4.3. Orthogonal polynomial based least-squares 55
4.3.1. Orthogonal polynomial bases 55

4.3.2. Grammian matrices 56
4.3.3. Algorithm 56
4.3.4. Least square method 58
4.3.5. Example (least square in action) 59
Exercises and problems on least squares 61

Chapter 5. Fixed point 63
5.1. Divide and average 63
5.1.1. Computing the square root 63
5.1.2. Why should the limit in (5.1.2) exist? 63
5.1.3. Error analysis 64
5.1.4. Proposition (convergence of Heron's algorithm) 64
5.2. Fixed-point theory 65
5.2.1. Definition of fixed point 65
5.2.2. Example 65
5.2.3. Remark (graphical interpretation in $\mathbb{R}$) 66
5.2.4. Exercise (contractions are continuous) 66
5.2.5. Theorem (Banach–Caccioppoli's contraction principle) 66
5.2.6. Definition of Lipschitz continuity 67
5.2.7. Exercise 68
5.2.8. Problem 68
5.2.9. Example (applying Banach–Caccioppoli) 68
5.3. Fixed-point theory in metric spaces 70
5.3.1. Definition of metric space 70
5.3.2. Definition of convergent and Cauchy sequences 70
5.3.3. Exercise (convergence characterisation) 70
5.3.4. Exercise (Cauchy criterion) 70
5.3.5. Definition of complete metric space 70
5.3.7. Definition of Lipschitz continuity 70
5.4. Higher convergence rates 70
5.4.1. Heron is faster than the Banach–Caccioppoli 'prediction' 70
5.4.2. A graphical interpretation of Heron 71
5.4.3. Definition of linear convergence, linear error reduction 72
5.4.4. Definition of quadratic convergence, quaratic error reduction 72
5.4.5. Definition of general convergence rate, superlinear convergence 72
5.4.6. Sharp convergence rates 72
5.4.7. Example (convergence rates) 73
5.4.8. Theorem (quadratic Banach–Caccioppoli) 73
5.4.9. Remark 73
5.4.10. What if iterator's derivative is not Lipschitz? 74
5.4.11. Can one have convergence rates lower than 1? 74
5.5. Notes, pointers and challenges 74
5.5.1. Generalisations of Banach–Caccioppoli's contraction principle 74
5.5.2. Other fixed-point results 74
5.5.3. The (sometimes false) power of series 74
Exercises and problems on fixed point 76

Chapter 6. Newton’s method 79
6.1. Derivations of Newton’s method 79
6.1.1. A geometric derivation 80
6.1.2. An analytic derivation 80
6.1.3. Exercise (a fixed point method to compute tan) 80
6.2. Convergence analysis 81
6.2.1. Definition of simple root, multiple root 81
6.2.2. Proposition (simple roots of differentiable functions) 81
6.2.3. Proposition (Lipschitz algebra) 81
6.2.4. Theorem (local convergence of Newton–Raphson’s method) 81
6.3. Variants of Newton–Raphson 82
Exercises and problems on Newton’s method 83

Chapter 7. Optimisation 85

Appendix A. Linear algebra 87
A.1. Algebra 87
A.1.1. Definition of group and Abelian (or commutative) group 87
A.1.2. Exercise 87
A.1.3. Definition of subgroup 87
A.1.4. Exercise 88
A.1.5. Example (the additive integers form a group) 88
A.1.6. Exercise (the multiplicative integers do not form a group) 88
A.1.7. Definition of field 88
A.1.8. Exercise 88
A.1.9. Number fields 88
A.2. Vector spaces 89
A.2.1. Definition of vector space 89
A.2.2. Exercise 89
A.2.3. Definition of vector subspace 89
A.2.4. Exercise 89
A.2.5. Complexification of a $\mathbb{R}$-vector space 89
A.3. The model finite dimensional $\mathbb{K}$-vector space $\mathbb{K}^n$ 90
A.3.1. Definition of matrix 90
A.3.2. Definition of linear combination, spanned space, span, dimension 92
A.3.3. Proposition (characterisation of the span) 92
A.3.4. Definition of basis 93
A.3.5. Theorem (finite dimension) 93
A.3.6. Remark (variants) 93
A.4. Algebras and polynomials 93
A.4.1. Definition 93
A.4.2. Example 93
A.4.3. Definition of polynomial 94
A.4.4. Example (2×2 skew symmetric equidiagonal matrices are the complex numbers) 94
A.4.5. Polynomial identity principle 95
A.4.6. Theorem 95

A.4.7. Theorem (Euclidean algorithm for polynomial of ℝ or ℂ) 95
A.4.8. Theorem (fundamental theorem of algebra) 95
A.4.9. Remark (ℝ is not algebraically closed) 95
A.5. Linear maps 96
A.5.1. Linear maps 96
A.5.2. Theorem 96
A.5.3. Definition of dual space, duality pairing and dual basis 96
A.5.4. Example (coordinate maps) 97
A.5.5. Theorem (primal–dual isomorphism) 97
A.5.6. The bidual space and the natural primal-bidual identification 97
A.5.7. Theorem (matrix representation of linear maps) 97
A.5.8. Example 98
A.5.9. Adjoint operator 99
A.5.10. Invertibility 99
A.5.11. Problem (invertible maps and linearity) 99
A.6. Multlinear forms and determinants 100
A.6.1. Definition of multilinear forms on a space 100
A.6.2. Definition of symmetric and antisymmetric forms 100
A.6.3. Definition of alternating form 100
A.6.4. Exercise 100
A.6.5. Problem (alternating and antisymmetric forms are synonymous) 100
A.6.6. Pull-back and push-forward of k-linear forms by linear maps 101
A.6.7. Determinant of a linear operator on a finite dimensional space 101
A.6.8. Exercise (existence and uniqueness of a linear operator's determinant) 101
A.6.9. Exercise 101
A.6.10. Exercise 101
A.6.11. Theorem (Leibniz formula) 102
A.6.12. Definition of sesquilinear form 102
A.6.13. Remark (real and sesquilinear is same as bilinear) 102
A.7. Spectral theory 102
A.7.1. Complexification of a linear operator 102
A.7.2. Operator eigenstuff 102
A.7.3. Matrix eigenstuff 103
A.7.4. Exercise (eigenspaces are invariant) 103
A.7.5. Eigenvalue multiplicity 103
A.7.6. Exercise (real matrices and their eigenvalues) 104
A.7.7. Spectral or singular values and spectrum 104
A.7.8. Example (eigenvectors of the derivative operator) 104
A.7.9. Theorem (spectral invariance under basis transformation) 104
A.7.10. Corollary (spectral invariance under similarity transformation) 105
A.7.11. Lemma (eigenbasis characterisation of diagonalisable matrices) 105
A.7.12. Lemma (spectral factorisation of a diagonalisable matrix) 105
A.7.13. Remark 105
A.7.14. Remark (where's the factorisation?) 105
A.7.15. Proof of lemma A.7.12 105
A.8. Norms and normed vector spaces 106
A.8.1. Definition of norm 106

A.8.2. Definition of convergent sequence 106
A.8.3. Proposition (uniqueness of the limit) 106
A.8.4. Definition of limit of a sequence 106
A.8.5. Theorem (vector algebra of limits) 107
A.9. Scalar (or inner) products and Hilbert spaces 107
A.9.1. Definition of inner product (space) 107
A.9.2. Remark (sesquilinearity of the scalar product) 107
A.9.3. Hilbertian norms 107
A.9.4. The Euclidean norm 107
A.9.5. Definition of Hilbert space 108
A.9.6. Definition of orthonormal system 108
A.9.7. Proposition 108
A.9.8. Definition of Hilbert (or orthonormal) basis 108
A.9.9. Remark 108
A.9.10. Remark 108
A.9.11. Theorem (Hilbert dimension) 108
A.9.12. Definition 108
A.9.13. Proposition 108
A.9.14. Theorem (Riesz representation) 108
A.9.15. Remark (topological irrelevance in finite dimension) 109
A.9.16. Theorem (sesquilinear forms and linear opertors isomorphism) 109
A.9.17. Adjoint operators 109
Exercises and problems on Linear algebra 109

Appendix B. Analysis 113
B.1. Sequences and convergence in $\mathbb{R}$ 113
B.1.1. Definition of sequence 113
B.1.2. Definition of convergent sequence of real numbers 113
B.1.3. Facts about convergent sequences 114
B.1.4. Remark 114
B.1.5. Theorem (Cauchy criterion) 114
B.2. Normed vector spaces (topological linear algebra) 114
B.2.1. Vector spaces 114
B.2.2. Definition of norm 114
B.2.3. Common Vector Norms 115
B.2.4. Exercise 115
B.2.5. Vector norm inequalities 115
B.3. Matrix analysis 116
B.3.1. Matrix norms 116
B.3.2. Example (maximum row-sum norm) 116
B.3.3. Exercise (max row sum is a norm) 116
B.3.4. Definition of induced bounded-linear operator norm 117
B.3.5. Induced matrix p-norms 117
B.3.6. Exercise (matrix norms) 117
B.3.7. Proposition (matrix p-norm dual inequalities) 117
B.3.8. Definition of Frobenius norm 118
B.3.9. Neumann's series 118

B.3.10. Norms on vector space in general 118
B.3.11. Theorem (equivalence of norms characterises finite dimension) 119
B.3.12. Examples of infinite dimensional normed vector spaces 119
B.3.13. Function-valued functions 119
B.3.14. Balls, neighbourhoods and open sets 120
B.4. Continuity 120
B.4.1. Definition of continuous function 120
B.4.2. Theorem (characterisation of continuity at a point via sequences) 120
B.4.3. Theorem (differentiable implies continuous) 121
B.4.4. Remark (continuous does not imply differentiable) 121
B.4.5. Theorem (linear algebra of continuous functions) 121
B.4.6. Theorem (continuity is invariant under composition) 121
B.4.7. Compactness and the Weierstrass Theorem 121
B.4.8. Theorem (Bolzano–Weierstrass) 121
B.4.9. Theorem (Weierstrass compactness) 121
B.4.10. Theorem (uniform limit) 122
B.4.11. Problem 122
B.4.12. Problem 122
B.5. Completeness and Banach spaces 122
B.6. Lipschitz and Hölder continuity 122
B.6.1. Definition of Lipschitz continuous function 122
B.6.2. Proposition 122
B.6.3. Remark 122
B.6.4. Proposition (differentiability and Lipschitz continuity) 123
B.6.5. Remark 123
B.6.6. Definition of Hölder continuous function 123
B.6.7. Proposition 123
B.6.8. Exercise (Hölder space) 123
B.6.9. Remark (What about 0-Hölder?) 123
B.6.10. Remark (1-Hölder is same as Lipschitz) 123
B.6.11. Proposition (Hölder space hierarchy) 123
B.6.12. Example 124
B.7. The Banach–Caccioppoli Contraction Principle 124
B.7.1. Definition of contraction mapping 124
B.7.2. Remark 124
B.7.3. Remark (continuity of contractions) 124
B.7.4. Theorem (Banach–Caccioppoli contraction principle) 124
B.8. Landau’s big O and small o notation 125
B.8.1. Big O 125
B.8.2. Definition of the rigourous Landau O notation 126
B.8.3. The Formal Landau O Notation 126
B.8.4. Theorem (equivalence of rigorous and formal O notation for analytic functions) 126
B.8.5. Example (smooth function which is not analytic) 127
Exercises and problems on Analysis basics 129

Appendix C. Polynomial interpolation 131

C.1. Polynomials and approximation of functions 131
C.1.1. Theorem (Weierstrass's polynomial approximation) 131
C.1.2. Remark (Weierstrass is not very practical) 131
C.1.3. Taylor's polynomial 131
C.1.4. Definition of Taylor's polynomial 132
C.1.5. Theorem (Taylor's polynomial approximation error) 132
C.1.6. Remark (Taylor's good news) 132
C.1.7. Remark (Taylor's bad news) 132
C.1.8. Example 132
C.2. Interpolation 132
C.2.1. Example (linear Lagrange interpolation) 133
C.2.2. Lagrange polynomials 134
C.2.3. Theorem (interpolant's uniqueness) 134
C.2.4. Solve the Vandermonde linear system (C.2.8) 135
C.2.5. Example 135
C.2.6. Definition of Lagrange basis 135
C.2.7. Remark (interpolation property of Lagrange polynomials) 135
C.2.8. Definition of Lagrange basis 136
C.2.9. Theorem (Lagrange interpolation) 136
C.2.10. Remark 137
C.2.11. Example 137
C.2.12. Theorem (Lagrange interpolation error estimate) 137
C.2.13. Remark (Lagrange vs. Taylor) 138
C.3. Newton's divided differences 138
C.3.1. Newton's basis 138
C.3.2. Lemma (recursive property of Newton's polynomials) 139
C.3.3. Corollary (Newton's divided difference formula) 140
C.3.4. Theorem (divided differences interpolation) 140
C.3.5. Example 141
C.3.6. Example 141
Exercises and problems on polynomial interpolation 142

Appendix D. Coding hints 145
D.1. Generalities 145
D.2. Matlab and Octave 145
D.2.1. Matlab® 145
D.2.2. Octave 146
D.2.3. Object oriented Matlab® and Octave 146
Matlab® class jargon buster 146
D.3. Octave 147
D.3.1. First steps 147
Octave scalars 147
Octave vectors 147
Octave matrices 148
D.3.2. Plotting 148
D.4. Python: NumPy and SciPy 148
D.4.1. References 149

D.4.2. First slithers: interactive mode 149
D.4.3. Programming 150
Sample script 150
Sample function 150
Sample module 150
Using a module interactively 151
D.4.4. Remark (my updates don't work?) 151

Appendix. Bibliography 153

Appendix. Index 157

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

These notes reflect my personal digest of “standard” published sources, as well as lecture notes from previous courses taught at Sussex and elsewhere. Some colleagues (former and present) that have contributed indirectly to the notes are Peter Giesl, Kerstin Hesse, Holger Wendland, David Kay. Other people whose names fade in the fog of twentieth centry will hopefully contact me for adding them to this list.
While this may be “obvious” to experts, since this is a text for beginners, I should emphasise that I hold no pretense at originality whatsoever. I do however take responsibility for all the errors that you may find in the text. If you do find what you think is an error/typo/mistake, I will be grateful if you sent me an email at lakkis.o.maths@gmail.com (even if unsure) about it (subject: “Numerical Analysis 2 notes typo”) so that things can be fixed for future releases of these notes.

Recommended reading list

Atkinson, 1989: Standard undergraduate text. A bit dated, but still excellent in many topics.

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 differential equations.

Stuart and Humphries, 1996: An original outlook on numerical methods for differential equations from a “dynamical systems” point of view.

Other useful texts

J. H. Hubbard and B. B. Hubbard, 2006: A good reference for revising calculus, linear algebra and symbolic computing from a more “mature” perspective.

Omar Lakkis
2nd October 2016
lakkis.o.maths@gmail.com

Introduction

Numerical methods consist in computing (or approximating) using pen and paper, or (rather more frequently 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:         $r_k := \operatorname{mod}_{r_{k-2}} r_{k-1}$
5:         $k \leftarrow k + 1$
6:     end while
7:     return $r_{k-1}$                          ▷ hcf is the last non-zero remainder
8: end procedure=0
```

Here $\operatorname{hcf}(m, n)$ 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 $\operatorname{mod}_s t$ for to 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

$$0 \leq r \leq d - 1. \tag{0.0.2}$$

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

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

Printout of file `Code/hcfEuclid.m`

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

```
%%
%% 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 1. For example, consider the simple 2×2 case, where we are asked to find (x,y) such that

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

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

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} ax+by \\ cx+dc \end{bmatrix}. \tag{0.0.4}$$

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

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

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

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

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

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

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*)

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

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 $ax_n-1$,","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 $ax_n - 1$	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 5 and 6 more about computing the reciprocal. But let us draw some conclusions here:

(i) The geometric series method provides us with a way to successively approximate the reciprocal of a given real number $a \in (0,2)$.

(ii) 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 $1/(1-a)$.

(iii) While the exact value of $1/a$ is never attained, we can push the iteration until a satisfactory residual is reached.

CHAPTER 1

Gaussian elimination

Newton, in notes that he would rather not have seen published, described a process for solving simultaneous equations that later authors applied specifically to linear equations. This method—which Euler did not recommend, which Legendre called “ordinary,” and which Gauss called “common”—is now named after Gauss: “Gaussian” elimination. Gauss’s name became associated with elimination through the adoption, by professional computers, of a specialized notation that Gauss devised for his own least-squares calculations. The notation allowed elimination to be viewed as a sequence of arithmetic operations that were repeatedly optimized for hand computing and eventually were described by matrices.
– Joseph F. Grcar (2011)

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

$$\text{find vector } \boldsymbol{x} \in \mathbb{K}^n \text{ such that } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \text{ for given matrix } \boldsymbol{A} \in \mathbb{K}^{n\times n} \text{ and } \boldsymbol{b} \in \mathbb{K}^n \quad (1.0.1)$$

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 5 and 6. 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.

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

1.1. Triangular matrices

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

- ⋆ *upper triangular* if and only if

$$a_i^j = 0 \text{ for } i > j, \tag{1.1.1}$$

- ⋆ *lower triangular* if and only if

$$a_i^j = 0 \text{ for } i < j, \tag{1.1.2}$$

- ⋆ *triangular* if and only if $\boldsymbol{A}$ is upper triangular *or* lower triangular,
- ⋆ *diagonal* if and only if $\boldsymbol{A}$ is *both* upper *and* lower triangular,
- ⋆ *(diagonally) normalised* if and only if

$$a_i^i = 1 \quad \forall\, i = 1,\dots,n. \tag{1.1.3}$$

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

$$\mathscr{U}(\mathbb{K}^n) := \left\{\boldsymbol{X} \in \mathbb{K}^{n\times n} : \boldsymbol{X} \text{ is upper triangular}\right\}, \tag{1.1.4}$$

$$\mathscr{L}_1(\mathbb{K}^n) := \left\{\boldsymbol{X} \in \mathbb{K}^{n\times n} : \boldsymbol{X} \text{ is normalised lower triangular}\right\}, \tag{1.1.5}$$

and

$$\mathscr{D}(\mathbb{K}^n) := \left\{\boldsymbol{X} \in \mathbb{K}^{n\times n} : \boldsymbol{X} \text{ is diagonal}\right\}. \tag{1.1.6}$$

1.1.3. Proposition.

(a) The sets $\mathscr{U}(\mathbb{K}^n)$ and $\mathscr{D}(\mathbb{K}^n)$ are closed under scaling, matrix addition and matrix multiplication, and are thus algebras of operators.

(b) The set $\mathscr{L}_1(\mathbb{K}^n)$ is closed under matrix multiplication and is a (non-Abelian) group with respect to this operation.

Proof See the problem section. □

1.1.4. 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

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ & 1 & 0 \\ & & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ & 1 & \\ & & 1 \end{bmatrix} = \begin{bmatrix} 1 & * & * \\ & 1 & * \\ & & 1 \end{bmatrix}. \tag{1.1.7}$$

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.

1.1.5. Example. Upper triangular matrices:

$$\begin{bmatrix} 1 & 3 \\ & 2 \end{bmatrix}, \begin{bmatrix} 1 & 3 & 4 \\ & 5 & 0 \\ & & 6 \end{bmatrix}. \tag{1.1.8}$$

Lower triangular matrices:

$$\begin{bmatrix} 1 & \\ 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 & & \\ 2 & 0 & \\ 4 & 3 & 6 \end{bmatrix}. \tag{1.1.9}$$

Diagonal matrices:

$$\begin{bmatrix} 3 & \\ & 4 \end{bmatrix}, \begin{bmatrix} 2 & & \\ & 5 & \\ & & -1 \end{bmatrix} \tag{1.1.10}$$

Normalised lower triangular matrices:

$$\begin{bmatrix} 1 & \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ 6 & 1 & \\ 4 & -1 & 1 \end{bmatrix}. \tag{1.1.11}$$

1.2. 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 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ solve into the solution of two triangular systems, whereby $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$, with $\boldsymbol{L} \in \mathscr{L}(\mathbb{K}^n)$ and $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$. This is useful because the solution could be then split into the solution of two linear systems:

$$\text{find } \boldsymbol{v} : \boldsymbol{L}\boldsymbol{v} = \boldsymbol{b} \text{ then find } \boldsymbol{x} : \boldsymbol{U}\boldsymbol{x} = \boldsymbol{v}. \tag{1.2.1}$$

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

$$\boldsymbol{L}\boldsymbol{x} = \boldsymbol{c} \tag{1.2.2}$$

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

$$\begin{array}{lllllll} l_1^1 x_1 & & & & = & c_1, \\ l_2^1 x_1 & +l_2^2 x_2 & & & = & c_2, \\ \vdots & \cdots & \ddots & & \vdots & \vdots \\ l_{n-1}^1 x_1 & +\cdots & +l_{n-1}^{n-1} x_{n-1} & & = & c_{n-1}, \\ l_n^1 x_1 & +\cdots & +l_n^{n-1} x_{n-1} & +l_n^n x_n & = & c_n. \end{array} \tag{1.2.3}$$

Solving the first equation we have

$$x_1 = \left(l_1^1\right)^{-1} c_1, \tag{1.2.4}$$

then, passing to the second equation, we get

$$x_2 = \left(l_2^2\right)^{-1} \left(c_2 - l_2^1 x_1\right), \tag{1.2.5}$$

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

$$x_k = \left(l_k^k\right)^{-1} \left(c_k - \sum_{j=1}^{k-1} l_k^j x_j \right), \tag{1.2.6}$$

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-subsitution in §1.2.2.

1.2.2. 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 $\boldsymbol{x} = (x_1, \dots, x_n)$, the linear system

$$\boldsymbol{U}\boldsymbol{x} = \boldsymbol{c} \tag{1.2.7}$$

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

$$u_i^i \neq 0 \quad \forall\, i = 1, \dots, n, \tag{1.2.8}$$

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

$$\begin{array}{ccccccc}
u_1^1 x_1 & +u_1^2 x_2 & +\cdots & +u_1^n x_n & = & c_1, \\
 & u_2^2 x_2 & +\cdots & +u_2^n x_n & = & c_2, \\
 & \ddots & \cdots & \vdots & \vdots & \vdots \\
 & & u_{n-1}^{n-1} x_{n-1} & +u_{n-1}^n x_n & = & c_{n-1}, \\
 & & & u_n^n x_n & = & c_n.
\end{array} \tag{1.2.9}$$

Solving the n-th equation, we have that

$$x_n = \left(u_n^n\right)^{-1} c_n. \tag{1.2.10}$$

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

$$x_{n-1} = \left(u_{n-1}^{n-1}\right)^{-1} \left(c_{n-1} - u_{n-1}^n x_n\right) \tag{1.2.11}$$

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 (1.2.9) being

$$u_k^k x_k + u_k^{k+1} x_{k+1} + \cdots + u_k^n x_n = c_k \tag{1.2.12}$$

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

$$\begin{aligned}
x_k &= \left(u_k^k\right)^{-1} \left(c_k - u_k^{k+1} x_{k+1} - \cdots - u_k^n x_n\right) \\
&= \left(u_k^k\right)^{-1} \left(c_k - \sum_{j=k+1}^{n} u_k^j x_j \right) \\
&= \left([\boldsymbol{U}]_k^k\right)^{-1} \left([\boldsymbol{c}]_k - [\boldsymbol{U}]_k^{[k+1\dots n]} [\boldsymbol{x}]_{[k+1\dots n]}\right),
\end{aligned} \tag{1.2.13}$$

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

[2]Whenever unstated a vector is a column-vector.

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

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

1.2.3. Example.

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

$$\boldsymbol{U} := \begin{bmatrix} 1 & & 2 \\ & 1 & -1 \\ & & -3 \end{bmatrix} \text{ and } \boldsymbol{c} := \begin{bmatrix} 3 \\ 0 \\ 6 \end{bmatrix} \tag{1.2.14}$$

.

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

$$x_3 = -\frac{6}{3} = -2, \tag{1.2.15}$$

whence, using the second equation

$$x_2 = x_3 = -2, \tag{1.2.16}$$

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

$$x_1 = 3 - 2x_3 = 3 + 4 = 7. \tag{1.2.17}$$

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

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

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

1.2.6. 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.

Printout of file `backsubstitutionTest.m`

```
%%% -*- mode: octave; -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function backsubstitutionTest(n)
%% function backsubstitutionTest(n)
%%
%% This function is a benchmark Test for the backsubstition()
%% 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
```

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

$$\boldsymbol{U} =: \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ & \boldsymbol{D} \end{bmatrix}, \boldsymbol{x} =: \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix} \text{ and } \boldsymbol{c} =: \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix} \tag{1.2.18}$$

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

$$\boldsymbol{z} = \boldsymbol{D}^{-1}\boldsymbol{g} \text{ and } \boldsymbol{y} = \boldsymbol{A}^{-1}\left(\boldsymbol{f} - \boldsymbol{B}\boldsymbol{z}\right) \tag{1.2.19}$$

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

1.2.8. Algorithm (Back-substitution). Of particular interest in 1.2.7 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 $n = 1$, $\boldsymbol{U} = u \in \mathbb{K}^{1\times 1} = \mathbb{K}$ and $\boldsymbol{c} = c \in \mathbb{K}^1 = \mathbb{K}$ which is simply solved as $x = u^{-1}c$, the matrix $\boldsymbol{U}$ as being decomposed into a $(1, n-1) \times (1, n-1)$ blocks as follows

$$\boldsymbol{U} = \begin{bmatrix} u & \boldsymbol{v}^\intercal \\ \boldsymbol{0} & \boldsymbol{W} \end{bmatrix} \text{ where } u = u_1^1, \boldsymbol{v}^\intercal = \boldsymbol{U}_{[1]}^{[2\dots n]}, \boldsymbol{W} = \boldsymbol{U}_{[2\dots n]}^{[2\dots n]}. \tag{1.2.20}$$

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

$$x_1 = u^{-1}(c_1 - \boldsymbol{v}^\intercal \hat{\boldsymbol{x}}) \tag{1.2.21}$$

Following is a pseudocode for *recursive back-substitution*

```
Require: $n \in \mathbb{N}, \boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n), \boldsymbol{c} \in \mathbb{K}^n$
Ensure: $\boldsymbol{x}$ such that $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{c}$
1: procedure RECURSIVE-BACKSUBSTITUTION($\boldsymbol{U}, \boldsymbol{c}$)
2:     if $n = 1$ then                                ▷ $\boldsymbol{U} = u \in \mathbb{K}$ and $\boldsymbol{c} = c \in \mathbb{K}$
3:         $\boldsymbol{x} \leftarrow \boldsymbol{U}^{-1}\boldsymbol{c}$
4:     else
5:         $[\boldsymbol{x}]_{[2\dots n]} \leftarrow$ RECURSIVE-BACKSUBSTITUTION($\boldsymbol{U}_{[2\dots n]}^{[2\dots n]}, [\boldsymbol{c}]_{[2\dots n]}$)
6:         $x_1 \leftarrow \left(u_1^1\right)^{-1}\left(c_1 - \boldsymbol{U}_{[1]}^{[2\dots n]}[\boldsymbol{x}]_{[2\dots n]}\right)$
7:     end if
8:     return $\boldsymbol{x}$
9: end procedure
```

1.2.9. 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 1.2.2 (after appropriate tweaking) can be used.

1.2.10. 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., $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$ and $\boldsymbol{c} \in \mathbb{K}^n$. Then

$$C(1) = r + 1 \tag{1.2.22}$$

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 §1.2.8, we get, for $n \geq 2$ that

$$C(n) = C(n-1) + r + (n-1) \tag{1.2.23}$$

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

$$C(n) = (n-1)r + \sum_{i=1}^{n-1} i + C(1) = 1 + nr + \frac{1}{2}n(n-1) = \frac{1}{2}n^2 + \left(r - \frac{1}{2}\right)n + 1. \tag{1.2.24}$$

1.3. Gaussian elimination

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

1.3.1. 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

$$\boldsymbol{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}. \tag{1.3.1}$$

For this to succeed with row operations

$$\text{scale row 1 of } \boldsymbol{A} \text{ by} -ca^{-1} \text{ and add it to row 2 of } \boldsymbol{A} \tag{1.3.2}$$

obtaining thus the matrix

$$\begin{bmatrix} a & b \\ 0 & d - ca^{-1}b \end{bmatrix} =: \boldsymbol{U}. \tag{1.3.3}$$

1.3.2. 2×2 **LU factorisation.** Recalling that

$$\mathbf{e}^1 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{e}^2 := \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ and } \mathbf{e}_i := \mathbf{e}^{i\mathsf{T}}, \tag{1.3.4}$$

we note that for a generic matrix $\boldsymbol{X}$

$$\text{row } i \text{ of } \boldsymbol{X} = \mathbf{e}_i \boldsymbol{X}. \tag{1.3.5}$$

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

$$\begin{bmatrix} \mathbf{e}_1 \boldsymbol{U} \\ \mathbf{e}_2 \boldsymbol{U} \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 \boldsymbol{A} \\ \mathbf{e}_2 \boldsymbol{A} - ca^{-1}\mathbf{e}_1 \boldsymbol{A} \end{bmatrix}. \tag{1.3.6}$$

A bit more matrix algebra yields

$$\boldsymbol{U} = \left(\begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_2\end{bmatrix} - \begin{bmatrix}\mathbf{0}^\mathsf{T}\\ c\,a^{-1}\mathbf{e}_1\end{bmatrix}\right)\boldsymbol{A} = \underbrace{(\mathbf{I}-\boldsymbol{m}\mathbf{e}_1)}_{=:\,\boldsymbol{M}}\boldsymbol{A} \text{ with } \boldsymbol{m} := \begin{bmatrix}0\\ c\end{bmatrix}a^{-1}. \tag{1.3.7}$$

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

$$\boldsymbol{M} = \begin{bmatrix}1 & 0\\ -c\,a^{-1} & 1\end{bmatrix}. \tag{1.3.8}$$

Noting that the inverse of $\boldsymbol{M}$ is also (normalised) lower triangular since[∗] [∗]: Check!

$$\boldsymbol{M}^{-1} = \begin{bmatrix}1 & 0\\ c\,a^{-1} & 1\end{bmatrix} =: \boldsymbol{L}, \tag{1.3.9}$$

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

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}. \tag{1.3.10}$$

1.3.3. Blockwise algebra. Extending the discussion in §1.3.1 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 $\boldsymbol{A},\boldsymbol{M} \in \mathbb{K}^{n\times n}$ are written as

$$\boldsymbol{A} = \begin{bmatrix}a & \boldsymbol{b}^\mathsf{T}\\ \boldsymbol{c} & \boldsymbol{D}\end{bmatrix} \text{ and } \boldsymbol{Z} = \begin{bmatrix}z & \boldsymbol{y}^\mathsf{T}\\ \boldsymbol{x} & \boldsymbol{W}\end{bmatrix} \tag{1.3.11}$$

where

$$a,z \in \mathbb{K},\ \boldsymbol{b},\boldsymbol{c},\boldsymbol{y},\boldsymbol{x} \in \mathbb{K}^{n-1},\ \boldsymbol{D},\boldsymbol{W} \in \mathbb{K}^{(n-1)\times(n-1)}, \tag{1.3.12}$$

then the blockwise product of the two matrices is

$$\boldsymbol{A}\boldsymbol{Z} = \begin{bmatrix}az+\boldsymbol{b}^\mathsf{T}\boldsymbol{x} & a\boldsymbol{y}^\mathsf{T}+\boldsymbol{b}^\mathsf{T}\boldsymbol{W}\\ \boldsymbol{c}z+\boldsymbol{D}\boldsymbol{x} & \boldsymbol{c}\boldsymbol{y}^\mathsf{T}+\boldsymbol{D}\boldsymbol{W}\end{bmatrix}. \tag{1.3.13}$$

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

$$\boldsymbol{A} = \begin{bmatrix}a & \boldsymbol{b}^\mathsf{T}\\ \boldsymbol{c} & \boldsymbol{D}\end{bmatrix} \tag{1.3.14}$$

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

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

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

1.3.5. 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:

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \text{ and } \begin{bmatrix}1 & 2\\ 3 & 0\end{bmatrix} \text{ are LU-cky} \tag{1.3.15}$$

while

$$\begin{bmatrix}1 & 2\\ 3 & 6\end{bmatrix} \text{ and } \begin{bmatrix}0 & 3\\ 2 & 1\end{bmatrix} \text{ are un-LU-cky.} \tag{1.3.16}$$

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

1.3.6. Theorem (LU factorisation). *If a matrix $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ for some $n \in \mathbb{N}$ is LU-cky then $\boldsymbol{A}$ admits a unique LU factorisation, i.e., there exists a unique pair $\boldsymbol{L} \in \mathscr{L}_1(\mathbb{K}^n)$ and $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$ such that*

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}. \tag{1.3.17}$$

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 $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{bmatrix} a & \boldsymbol{b}^\intercal \\ \boldsymbol{c} & \boldsymbol{D}\end{bmatrix}, \text{ where } a \in \mathbb{K},\ \boldsymbol{b}, \boldsymbol{c} \in \mathbb{K}^{n-1}, \boldsymbol{D} \in \mathbb{K}^{(n-1)\times(n-1)}. \tag{1.3.18}$$

By mimicking the argument in §1.3.1 introduce the row-reduction vector and matrix, respectively, as

$$\begin{aligned} \boldsymbol{m} &:= \begin{bmatrix} 0 \\ \boldsymbol{c}a^{-1}\end{bmatrix} \text{ and} \\ \boldsymbol{M} := \mathbf{L}_1(\boldsymbol{m}) := \mathbf{I} - \boldsymbol{m}\mathbf{e}_1 &= \begin{bmatrix} 1 & \mathbf{0}^\intercal \\ -\boldsymbol{c}a^{-1} & \hat{\mathbf{I}}\end{bmatrix} \text{ where } \hat{\mathbf{I}} := \text{ identity in } \mathbb{K}^{(n-1)\times(n-1)}. \end{aligned} \tag{1.3.19}$$

Premultlipying $\boldsymbol{A}$ by $\boldsymbol{M}$ yields

$$\boldsymbol{M}\boldsymbol{A} = \begin{bmatrix} a & \boldsymbol{b}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{A}}\end{bmatrix} \text{ where } \hat{\boldsymbol{A}} := \boldsymbol{D} - \boldsymbol{c}a^{-1}\boldsymbol{b}^\intercal. \tag{1.3.20}$$

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

$$\hat{\boldsymbol{A}} = \hat{\boldsymbol{L}}\hat{\boldsymbol{U}} \text{ for some } \hat{\boldsymbol{L}} \in \mathscr{L}_1(\mathbb{K}^{n-1}),\ \hat{\boldsymbol{U}} \in \mathscr{U}(\mathbb{K}^{n-1}). \tag{1.3.21}$$

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

$$\boldsymbol{M}^{-1} = \mathbf{L}_1(-\boldsymbol{m}) = \mathbf{I} + \boldsymbol{m}\mathbf{e}_1, \tag{1.3.22}$$

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

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ \boldsymbol{c}a^{-1} & \hat{\mathbf{I}}\end{bmatrix}\begin{bmatrix} a & \boldsymbol{b}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{L}}\hat{\boldsymbol{U}}\end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}^\intercal \\ \boldsymbol{c}a^{-1} & \hat{\mathbf{I}}\end{bmatrix}\begin{bmatrix} 1 & \mathbf{0}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{L}}\end{bmatrix}\begin{bmatrix} a & \boldsymbol{b}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{U}}\end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \mathbf{0}^\intercal \\ \boldsymbol{c}a^{-1} & \hat{\boldsymbol{L}}\end{bmatrix}}_{=:\boldsymbol{L}}\underbrace{\begin{bmatrix} a & \boldsymbol{b}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{U}}\end{bmatrix}}_{=:\boldsymbol{U}} \tag{1.3.23}$$

Uniqueness is left as an exercise in Problem 1.12. □

1.3.7. Algorithm (recursive LU factorisation). The inductive nature of the proof of Theorem 1.3.6 provides us with the following recursive algorithm.

```
Require: $n \in \mathbb{N}, \boldsymbol{A} \in \mathbb{K}^{n\times n}$
Ensure: $\boldsymbol{A}$ is LU-cky and $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$ with $\boldsymbol{L} \in \mathscr{L}_1(\mathbb{K}^n)$, $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$
1: procedure RECURSIVE-LU-FACTORISATION($\boldsymbol{A}$)
2:     $\boldsymbol{u}_1 \leftarrow \boldsymbol{a}_1$ and $\boldsymbol{l}^1 \leftarrow \begin{bmatrix} 1 \\ \boldsymbol{A}^{[1]}_{[2\dots n]}\left(a^1_1\right)^{-1}\end{bmatrix}$        ▷ ignore the zeros
3:     if $n > 1$ then
4:         $(\boldsymbol{L}^{[2\dots n]}_{[2\dots n]}, \boldsymbol{U}^{[2\dots n]}_{[2\dots n]}) \leftarrow$ RECURSIVE-LU-FACTORISATION($\boldsymbol{A}^{[2\dots n]}_{[2\dots n]} - \boldsymbol{A}^{[1]}_{[2\dots n]}\left(a^1_1\right)^{-1}\boldsymbol{A}^{[2\dots n]}_{[1]}$)
5:     end if
6:     return $\boldsymbol{L}, \boldsymbol{U}$
7: end procedure
```

Note that $\boldsymbol{A}^{[1]}_{[2\dots n]}\left(a^1_1\right)^{-1}\boldsymbol{A}^{[2\dots n]}_{[1]}$ can be replaced by $\boldsymbol{L}^{[1]}_{[2\dots n]}\boldsymbol{U}^{[2\dots n]}_{[1]}$, where both elements are already computed. This saves on the multiplications.

1.3.8. 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]%% writing zeros for clarity (↩
      inefficient)
  %% and the big uppwer triangular matrix
  U =  [A(1,:);[zeros(n-1,1),Usmall]]%% writing zeros ditto
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
```

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

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

Ensure: $\boldsymbol{A}$ is LU-cky and $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$ with $\boldsymbol{L} \in \mathscr{L}_1(\mathbb{K}^n)$, $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$

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

[4]Careful 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.

```
10:     return $\boldsymbol{L}, \boldsymbol{U}$
11: end procedure
```

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

Require: $n \in \mathbb{N}, \boldsymbol{A} \in \mathbb{K}^{n\times n}, \boldsymbol{c} \in \mathbb{K}^n$
Ensure: $\boldsymbol{A}$ is LU-cky and $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$ with $\boldsymbol{L} \in \mathscr{L}_1(\mathbb{K}^n)$, $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$

```
1: procedure LU-COMPACT($\boldsymbol{A}$)
2:     for $i = 1, \dots, n$ do
3:         if $i < n$ then
4:             $\boldsymbol{A}^{[i]}_{[i+1\dots n]} \leftarrow \boldsymbol{A}^{[i]}_{[i+1\dots n]}\left(a^i_i\right)^{-1}$
5:             $\boldsymbol{A}^{[i+1\dots n]}_{[i+1\dots n]} \leftarrow \boldsymbol{A}^{[i+1\dots n]}_{[i+1\dots n]} - \boldsymbol{A}^{[i]}_{[i+1\dots n]}\boldsymbol{A}^{[i+1\dots n]}_{[i]}$
6:         end if
7:     end for
8:     return $\boldsymbol{L}, \boldsymbol{U}$
9: end procedure
```

1.3.11. 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-factoring a matrix $\boldsymbol{A}$ in $\mathbb{K}^{n\times n}$ then the cost of the scalar inversion (reciprocal) is

$$\operatorname{cost}\left[\left(a^1_1\right)^{-1}\right] = r \tag{1.3.24}$$

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

$$\operatorname{cost}\left[\boldsymbol{A}^{[1]}_{[2\dots n]}\left(a^1_1\right)^{-1}\right] = n-1 \tag{1.3.25}$$

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

$$\operatorname{cost}\left[\boldsymbol{A}^{[2\dots n]}_{[2\dots n]} - \boldsymbol{A}^{[1]}_{[2\dots n]}\left(a^1_1\right)^{-1}\boldsymbol{A}^{[2\dots n]}_{[1]}\right] = 2(n-1)^2 \tag{1.3.26}$$

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

$$\begin{aligned} C(n) &= r+(n-1)+2(n-1)^2+C(n-1) \\ &= r+(n-1)+2(n-1)^2 \\ &\quad + r+(n-2)+2(n-2)^2+C(n-2) \\ &= \cdots \\ &= r(n-1)+\sum_{i=1}^{n-1} i + 2\sum_{i=1}^{n-1} i^2 + C(1) \\ &= r(n-1)+\frac{1}{2}n(n-1)+\frac{2}{3}n^3-n^2+\frac{1}{3}n+0 \\ &= \frac{2}{3}n^3-\frac{1}{2}n^2+\left(r-\frac{1}{6}\right)n-r \\ &\approx \frac{2}{3}n^3+ \text{ lower order terms in } n. \end{aligned} \tag{1.3.27}$$

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 irrelvant in front of the cube which is the leading term as $n \to \infty$.

1.4. Pivoting and PLU factorisation

1.4.1. 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

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{1.4.1}$$

[∗]: Check!

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

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 $\boldsymbol{A}$ and $\boldsymbol{M}$ two generic matrices, with $\boldsymbol{A}$ invertible. An elementary permutation matrix, $\mathbf{P}_{i\leftrightarrow j} \in \mathbb{R}^{n\times n}$, and an elementary (normalised) lower triangular matrix, $\mathbf{L}_i(\boldsymbol{m}) \in \mathbb{K}^{n\times n}$ where $\boldsymbol{m}$ is a column vector such that $\mathbf{e}^{k\mathsf{T}}\boldsymbol{m} = 0$ for all $k \leq i$, are given by

$$\mathbf{P}_{i\leftrightarrow j} := \mathbf{I} - (\mathbf{e}^i - \mathbf{e}^j)(\mathbf{e}^i - \mathbf{e}^j)^\mathsf{T} \text{ and } \mathbf{L}_i(\boldsymbol{m}) := \mathbf{I} - \boldsymbol{m}\mathbf{e}^{i\mathsf{T}}, \tag{1.4.2}$$

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

$$\boldsymbol{P} = \mathbf{P}_{i_1\leftrightarrow j_1} \cdots \mathbf{P}_{i_s\leftrightarrow j_s} \tag{1.4.3}$$

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

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

$$\mathbf{P}_{i\leftrightarrow j}{}^{-1} = \mathbf{P}_{i\leftrightarrow j}. \tag{1.4.4}$$

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

$$\begin{aligned}\left[\mathbf{P}_{i\leftrightarrow j}\boldsymbol{M}\right]_k &= \mathbf{e}^{k\intercal}\mathbf{P}_{i\leftrightarrow j}\boldsymbol{M} = \mathbf{e}^{k\intercal}\left(\mathbf{I}-(\mathbf{e}^i-\mathbf{e}^j)(\mathbf{e}^i-\mathbf{e}^j)^\intercal\right)\boldsymbol{M}\\ &= \left(\mathbf{e}^{k\intercal}-(\mathbf{e}^{k\intercal}\mathbf{e}^i-\mathbf{e}^{k\intercal}\mathbf{e}^j)(\mathbf{e}^i-\mathbf{e}^j)^\intercal\right)\boldsymbol{M} = \left(\mathbf{e}^k-(\delta^i_k-\delta^j_k)(\mathbf{e}^i-\mathbf{e}^j)\right)^\intercal\boldsymbol{M}\\ &= \begin{cases}\mathbf{e}^{k\intercal}\boldsymbol{M}=\boldsymbol{M}_k & \text{if } k\neq i,j,\\ \left(\mathbf{e}^i+\mathbf{e}^j-\mathbf{e}^i\right)^\intercal\boldsymbol{M}=\boldsymbol{M}_j & \text{if } k=i,\\ \left(\mathbf{e}^j-\mathbf{e}^j+\mathbf{e}^i\right)^\intercal\boldsymbol{M}=\boldsymbol{M}_i & \text{if } k=j.\end{cases}\end{aligned} \tag{1.4.5}$$

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

$$\left[\mathbf{P}_{i\leftrightarrow j}\right]_k = \begin{cases}\mathbf{e}^{k\intercal} & \text{if } k\neq i,j,\\ \mathbf{e}^{j\intercal} & \text{if } k=i\\ \mathbf{e}^{i\intercal} & \text{if } k=j.\end{cases} \tag{1.4.6}$$

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

$$\mathbf{P}_{i\leftrightarrow j}\mathbf{P}_{i\leftrightarrow j}=\mathbf{I}, \tag{1.4.7}$$

and thus

$$\mathbf{P}_{i\leftrightarrow j}{}^{-1}=\mathbf{P}_{i\leftrightarrow j}. \tag{1.4.8}$$

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

$$\mathbf{L}_i(\boldsymbol{m})^{-1}=\mathbf{L}_i(-\boldsymbol{m}). \tag{1.4.9}$$

Describe the product of $\mathbf{L}_i(\boldsymbol{m})$ premultiplying a generic matrix $\boldsymbol{M}$.
Denoting $\boldsymbol{m}=\left[m_k\right]_{k=1,\dots,n}$ with $m_k=0$ for $k\leq i$, the k-th row of of the product can be calculated as follows:

$$\begin{aligned}[\mathbf{L}_i(\boldsymbol{m})\boldsymbol{M}]_k &= \mathbf{e}^{k\intercal}\left(\mathbf{I}-\boldsymbol{m}\mathbf{e}^{i\intercal}\right)\boldsymbol{M} = \left(\mathbf{e}^{k\intercal}-\mathbf{e}^{k\intercal}\boldsymbol{m}\mathbf{e}^{i\intercal}\right)\boldsymbol{M}\\ &= \left(\mathbf{e}^{k\intercal}-\mathbf{e}^{k\intercal}\boldsymbol{m}\mathbf{e}^{i\intercal}\right)\boldsymbol{M} = \begin{cases}\boldsymbol{M}_k & \text{if } k\leq i,\\ \boldsymbol{M}_k-m_k\boldsymbol{M}_i & \text{if } k>i.\end{cases}\end{aligned} \tag{1.4.10}$$

In words, premultiplying $\boldsymbol{M}$ by $\mathbf{L}_i(\boldsymbol{m})$ 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 $k>i$.

1.4.4. Problem (permutation to normal-lower-triangle commutator). *Recall the definition of elementary lower triangular matrix*

$$\mathbf{L}_i(\boldsymbol{m}):=\mathbf{I}-\boldsymbol{m}\mathbf{e}_i. \tag{1.4.11}$$

Show that for any $\boldsymbol{m}\in\mathbb{K}^n$, with $m_1=0$, and $\boldsymbol{N}$ of the block-diagonal form

$$\boldsymbol{N}:=\begin{bmatrix}1 & \mathbf{0}^\intercal\\ \mathbf{0} & \hat{\boldsymbol{N}}\end{bmatrix} \textit{ where } \hat{\boldsymbol{N}}\in\mathbb{K}^{(n-1)\times(n-1)} \tag{1.4.12}$$

we have

$$\boldsymbol{N}\mathbf{L}_1(\boldsymbol{m})=\mathbf{L}_1(\boldsymbol{N}\boldsymbol{m})\boldsymbol{N} \textit{ where } [\boldsymbol{N}\boldsymbol{m}]_1=0. \tag{1.4.13}$$

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

$$\mathbf{L}_1(\boldsymbol{m}_1)\mathbf{P}_{1\leftrightarrow r_1}\boldsymbol{A} = \boldsymbol{A}(1) = \begin{bmatrix} a(1) & \boldsymbol{a}(1)^\intercal \\ \mathbf{0} & \boldsymbol{A}'(1) \end{bmatrix}. \tag{1.4.14}$$

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

$$\left[\tilde{\boldsymbol{A}}\right]_k = \begin{cases} [\boldsymbol{A}]_{r_1} & \text{if } k = 1 \\ [\boldsymbol{A}]_k & \text{if } 1 < k \neq r_1 \\ [\boldsymbol{A}]_1 & \text{if } 1 < k = r_1. \end{cases} \tag{1.4.15}$$

Define

$$m_k := \tilde{a}^1_k/\tilde{a}^1_1 = \begin{cases} 0 & \text{for } k = 1 \\ a^1_k/a^1_{r_1} & \text{for } 1 < k \neq r_1 \\ a^1_1/a^1_{r_1} & \text{for } 1 < k = r_1. \end{cases} \tag{1.4.16}$$

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

$$\begin{aligned} b^1_k = \mathbf{e}^{k\intercal}\boldsymbol{B}\mathbf{e}^1 = \mathbf{e}^{k\intercal}\mathbf{L}_1(\boldsymbol{m}_1)\mathbf{P}_{1\leftrightarrow r_1}\boldsymbol{A}\mathbf{e}^k = \left(\left[\tilde{\boldsymbol{A}}\right]_k - m_k\left[\tilde{\boldsymbol{A}}\right]_1\right)\mathbf{e}^1 \\ = \begin{cases} [\boldsymbol{A}]_{r_1}\mathbf{e}^1 = a^1_{r_1} & \text{if } k = 1 \\ \mathbf{e}^{1\intercal}\left(\boldsymbol{a}_k - a^1_k/a^1_{r_1}\boldsymbol{a}_{r_1}\right) = a^1_k - \left(a^1_k/a^1_{r_1}\right)a^1_{r_1} = 0 & \text{if } 1 < k \neq r_1 \\ a^1_k - \left(a^1_1/a^1_{r_1}\right)a^1_{r_1} = 0 & \text{if } 1 < k = r_1. \end{cases} \end{aligned} \tag{1.4.17}$$

This means that $\boldsymbol{B}$ is of the form

$$\boldsymbol{B} = \begin{bmatrix} a(1) & \boldsymbol{a}(1)^\intercal \\ \mathbf{0} & \boldsymbol{A}'(1) \end{bmatrix} \tag{1.4.18}$$

with

$$\left[a(1) \quad \boldsymbol{a}(1)^\intercal\right] = \boldsymbol{a}_{r_1} \text{ and } \boldsymbol{A}'(1) = \left[\mathbf{L}_1(\boldsymbol{m}_1)\tilde{\boldsymbol{A}}\right]^{[2\dots n]}_{[2\dots n]} \tag{1.4.19}$$

1.4.6. Group structure. Denote by $\mathscr{L}_1(\mathbb{K}^n)$ the set of all normalised lower triangular matrices and $\mathscr{P}(n)n$ the set of all permutation matrices. Explain what is meant by "$\mathscr{L}_1(\mathbb{K}^n)$ and $\mathscr{P}(n)n$ are multiplicative groups" and why this is useful to deduce that there are $\boldsymbol{P} \in \mathscr{P}(n)n$, $\boldsymbol{L} \in \mathscr{L}_1(\mathbb{K}^n)$ and $\boldsymbol{U} \in \mathbb{K}^{n\times n}$ upper triangular, such that

$$\boldsymbol{A} = \boldsymbol{PLU}. \tag{1.4.20}$$

Argue by induction on n. The result is trivially true for $n = 1$, with $\boldsymbol{U} = \boldsymbol{A}$ and $\boldsymbol{L} = \boldsymbol{P} = \mathbf{I} = 1$. Suppose it is true in $\mathbb{K}^{n-1}$, using r_1 and $\boldsymbol{m}_1$ found in step 1.4.5 (and keeping the same notation as therein) introduce $\boldsymbol{L}(1) := \mathbf{L}_1(\boldsymbol{m}_1)$ and $\boldsymbol{P}(1) := \mathbf{P}_{1\leftrightarrow r_1}$.
It follows that $\boldsymbol{A}'(1)$ is invertible, so, using the inductive hypothesis, we know that there are $\boldsymbol{P}' \in \mathscr{P}(n)n-1$, $\boldsymbol{L}' \in \mathscr{L}_1(\mathbb{K}^{n-1})$ and $\boldsymbol{U}' \in \mathbb{K}^{(n-1)\times(n-1)}$ upper triangular such that

$$\boldsymbol{A}'(1) = \boldsymbol{P}'\boldsymbol{L}'\boldsymbol{U}'. \tag{1.4.21}$$

Define now

$$\tilde{\boldsymbol{P}} := \begin{bmatrix} 1 & \mathbf{0}^\intercal \\ \mathbf{0} & \boldsymbol{P}' \end{bmatrix}, \tilde{\boldsymbol{L}} := \begin{bmatrix} 1 & \mathbf{0}^\intercal \\ \mathbf{0} & \boldsymbol{L}' \end{bmatrix} \text{ and } \boldsymbol{U} := \begin{bmatrix} a(1) & \boldsymbol{a}(1)^\intercal \\ \mathbf{0} & \boldsymbol{U}' \end{bmatrix}. \tag{1.4.22}$$

Immediately we have

$$\tilde{\boldsymbol{P}} \in \mathscr{P}(n)n,\ \tilde{\boldsymbol{L}} \in \mathscr{L}_1(\mathbb{K}^n) \text{ and } \boldsymbol{U} \text{ upper triangular.} \tag{1.4.23}$$

Furthermore, a bit of algebra implies

$$\tilde{\boldsymbol{P}}\tilde{\boldsymbol{L}}\boldsymbol{U} = \begin{bmatrix} a(1) & \boldsymbol{a}(1)^{\mathsf{T}} \\ \boldsymbol{0} & \boldsymbol{A}'(1) \end{bmatrix} = \boldsymbol{L}(1)\boldsymbol{P}(1)\boldsymbol{A}, \tag{1.4.24}$$

whence

$$\tilde{\boldsymbol{L}}\boldsymbol{U} = \tilde{\boldsymbol{P}}^{-1}\boldsymbol{L}(1)\boldsymbol{P}(1)\boldsymbol{A}, \text{ with } \tilde{\boldsymbol{P}}^{-1} = \begin{bmatrix} 1 & \boldsymbol{0}^{\mathsf{T}} \\ \boldsymbol{0} & \boldsymbol{P}'^{-1} \end{bmatrix} \tag{1.4.25}$$

But 1.4.4 tells us that

$$\tilde{\boldsymbol{P}}^{-1}\boldsymbol{L}(1) = \tilde{\boldsymbol{P}}^{-1}\mathbf{L}_1(\boldsymbol{m}_1) = \mathbf{L}_1(\tilde{\boldsymbol{P}}^{-1}\boldsymbol{m}_1)\tilde{\boldsymbol{P}}^{-1} \tag{1.4.26}$$

and thus

$$\tilde{\boldsymbol{L}}\boldsymbol{U} = \mathbf{L}_1(\tilde{\boldsymbol{P}}^{-1}\boldsymbol{m}_1)\tilde{\boldsymbol{P}}^{-1}\boldsymbol{P}(1)\boldsymbol{A}, \tag{1.4.27}$$

which, after inverting using (1.4.9), is equivalent to

$$\underbrace{\mathbf{L}_1(-\tilde{\boldsymbol{P}}^{-1}\boldsymbol{m}_1)\tilde{\boldsymbol{L}}}_{=:\,\boldsymbol{L}}\boldsymbol{U} = \mathbf{L}_1(\tilde{\boldsymbol{P}}^{-1}\boldsymbol{m}_1)^{-1}\tilde{\boldsymbol{L}}\boldsymbol{U} = \underbrace{\tilde{\boldsymbol{P}}^{-1}\boldsymbol{P}(1)}_{=:\,\boldsymbol{P}^{-1}}\boldsymbol{A} \tag{1.4.28}$$

Thanks to the group properties $\boldsymbol{L} \in \mathscr{L}_1(\mathbb{K}^n)$ and $\boldsymbol{P} \in \mathscr{P}(n)n$ and, as required, they satify

$$\boldsymbol{P}^{-1}\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}, \text{ and hence } \boldsymbol{A} = \boldsymbol{P}\boldsymbol{L}\boldsymbol{U}. \tag{1.4.29}$$

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

$$\begin{bmatrix} 0 & -2 & 3 \\ 2 & -6 & 2 \\ 3 & 0 & -3 \end{bmatrix} \tag{1.4.30}$$

$$\mathbf{P}_{1\leftrightarrow 3}\begin{bmatrix} 0 & -2 & 3 \\ 2 & -6 & 2 \\ 3 & 0 & -3 \end{bmatrix} = \begin{bmatrix} 3 & 0 & -3 \\ 2 & -6 & 2 \\ 0 & -2 & 3 \end{bmatrix} \tag{1.4.31}$$

$$\mathbf{L}_1(\begin{bmatrix} 0 \\ 2/3 \\ 0 \end{bmatrix})\begin{bmatrix} 3 & 0 & -3 \\ 2 & -6 & 2 \\ 0 & -2 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 0 & -3 \\ 2-2/3\times 3 & -6 & 2+2/3\times 3 \\ 0 & -2 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 0 & -3 \\ 0 & -6 & 4 \\ 0 & -2 & 3 \end{bmatrix} \tag{1.4.32}$$

$$\mathbf{L}_1(\begin{bmatrix} 0 \\ 1/3 \end{bmatrix})\begin{bmatrix} -6 & 4 \\ -2 & 3 \end{bmatrix} = \begin{bmatrix} -6 & 4 \\ -2+6/3 & 3-4/3 \end{bmatrix} = \begin{bmatrix} -6 & 4 \\ 0 & 5/3 \end{bmatrix} \tag{1.4.33}$$

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

$$\mathbf{L}_2(\begin{bmatrix} 0 \\ 0 \\ 1/3 \end{bmatrix})\mathbf{L}_1(\begin{bmatrix} 0 \\ 2/3 \\ 0 \end{bmatrix})\mathbf{P}_{1\leftrightarrow 3}\boldsymbol{A} = \begin{bmatrix} 3 & 0 & -3 \\ 0 & -6 & 4 \\ 0 & 0 & 5/3 \end{bmatrix} =: \boldsymbol{U} \tag{1.4.34}$$

or, equivalently,

$$\boldsymbol{A} = \mathbf{P}_{1\leftrightarrow 3}\mathbf{L}_1(\begin{bmatrix} 0 \\ -2/3 \\ 0 \end{bmatrix})\mathbf{L}_2(\begin{bmatrix} 0 \\ 0 \\ -1/3 \end{bmatrix})\boldsymbol{U} = \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{=:\boldsymbol{P}}\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2/3 & 1 & 0 \\ 0 & 1/3 & 1 \end{bmatrix}}_{=:\boldsymbol{L}}\underbrace{\begin{bmatrix} 3 & 0 & -3 \\ 0 & -6 & 4 \\ 0 & 0 & 5/3 \end{bmatrix}}_{=:\boldsymbol{U}}. \tag{1.4.35}$$

1.5. Doolittle's LU algorithm

Doolittle's algorithm produces an LU factorisation of a LU-cky matrix $\boldsymbol{A}$. The difference between it and previous versions of LU factorisation is that it is more efficient from a memory reading, in that access to the data is very

1.5.1. Derivation.

Suppose $\boldsymbol{A}$ has an LU factorisation whereby for a normalised lower triangular matrix $\boldsymbol{L}$ and an upper triangular matrix $\boldsymbol{U}$ we have $\boldsymbol{A} = \boldsymbol{LU}$. Then for any set of indexes i and j we have

$$a_i^j = \boldsymbol{l}_i \boldsymbol{u}^j, \tag{1.5.1}$$

where $\boldsymbol{a}_k$ indicates the k-th row of matrix $\boldsymbol{A}$ and $\boldsymbol{a}^k$ its k-th column. Knowing that $l_1^1 = 1$ and $l_1^i = 0$, for $j = 2, \ldots, n$, it follows that

$$a_1^1 = \boldsymbol{l}_1 \boldsymbol{u}^1 = \sum_{k=1}^{n} l_1^k u_k^1 = u_1^1, \tag{1.5.2}$$

this allows us to compute u_1^1. Similarly for any $j = 2, \ldots, n$ we have

$$a_1^j = \boldsymbol{l}_1 \boldsymbol{u}^j = \sum_{k=1}^{n} l_1^k u_k^j = u_1^j \tag{1.5.3}$$

which allows the computation of the whole of $\boldsymbol{U}$'s first row $\boldsymbol{u}_1$. Note that this could have been summarised as follows: given that $\boldsymbol{l}_1 = \mathbf{e}_1$ and $\boldsymbol{l}_1 = \mathbf{e}_1\boldsymbol{L}$ then

$$\boldsymbol{u}_1 = \mathbf{e}_1 \boldsymbol{U} = \mathbf{e}_1 \boldsymbol{LU} = \mathbf{e}_1 \boldsymbol{A} = \boldsymbol{a}_1. \tag{1.5.4}$$

Now using the fact that only entry l_2^1 is still unknown in row $\boldsymbol{l}_2$ and all of column $\boldsymbol{u}^1$ is now known:

$$a_2^1 = \boldsymbol{l}_2 \boldsymbol{u}^1 = l_2^1 u_1^1 + u_1^2 \tag{1.5.5}$$

which implies the following computation

$$l_2^1 = \left(u_1^1\right)^{-1}\left(a_2^1 - u_1^2\right). \tag{1.5.6}$$

The same argument can be used for $i = 2, \ldots, n$:

$$a_i^1 = \boldsymbol{l}_i \boldsymbol{u}^1 = l_i^1 u_1^1 + u_1^i \tag{1.5.7}$$

which implies the following computation

$$l_i^1 = \left(u_1^1\right)^{-1}\left(a_i^1 - u_1^i\right). \tag{1.5.8}$$

This allows the computation of all of $\boldsymbol{L}$'s first row. Again this could have been written in a more compact form by noting that

1.6. Stability and condition numbers

1.6.1. Definition of condition number. The *condition number* of a given matrix $\boldsymbol{A}$ with respect to a given matrix norm $|\cdot|_\sharp$ is defined as[5]

$$\kappa_\sharp(\boldsymbol{A}) := |\boldsymbol{A}|_\sharp \left|\boldsymbol{A}^{-1}\right|_\sharp. \tag{1.6.1}$$

[5] What is referred to as "condition number" should be in fact the "ill-condition number" because the higher the condition number the more ill-conditioned is the matrix. However, as with a lot of mathematical terminology all are used to the "wrong" term and live with it. And so do we.

Exercises and problems on linear systems and Gaussian elimination

Exercise 1.1 (back-substitution)**.** (a) Solve the following linear system by hand with back substitution:

$$\begin{bmatrix} 1 & 1 & 1 \\ & 2 & 2 \\ & & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ 6 \end{bmatrix}. \tag{P1.1.1}$$

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

$$\begin{bmatrix} 2 & -1 & 3 & 1 \\ & 1 & 2 & -1 \\ & & -2 & 1 \\ & & & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ -3 \\ 1 \\ 9 \end{bmatrix}. \tag{P1.1.2}$$

Exercise 1.2 (Gaussian elimination)**.** Bring the following linear system with Gaussian elimination into upper triangular form:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{A} = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 1 & 2 \\ 2 & 2 & 4 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 1 \\ 8 \\ 4 \end{bmatrix}. \tag{P1.2.1}$$

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

Exercise 1.3 (Gaussian elimination)**.** Consider the linear system of finding $\boldsymbol{x} = \left(x^1, x^2, x^3\right)$ such that

$$\underbrace{\begin{bmatrix} 2 & -1 & 1 \\ 4 & 2 & 1 \\ 6 & -4 & 2 \end{bmatrix}}_{=:\boldsymbol{A}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 4 \\ -2 \end{bmatrix} =: \boldsymbol{b}. \tag{P1.3.1}$$

(a) Using Gaussian elimination, and explaining all your steps, find two lower triangular matrices $\boldsymbol{M}(1)$ and $\boldsymbol{M}(2)$ and an upper triangular $\boldsymbol{U}$ such that

$$\boldsymbol{M}(2)\boldsymbol{M}(1)\boldsymbol{A} = \boldsymbol{U} = \begin{bmatrix} u_1^1 & u_1^2 & u_1^3 \\ 0 & u_2^2 & u_2^3 \\ 0 & 0 & u_3^3 \end{bmatrix}. \tag{P1.3.2}$$

(b) Compute the matrix $\boldsymbol{M} := \boldsymbol{M}(2)\boldsymbol{M}(1)$ and use it, with $\boldsymbol{U}$ and backsubstitution to solve (P1.3.1).

(c) Say $\boldsymbol{b}$ is replaced by $\boldsymbol{c} = (1,4,2)$. Would you have to redo the Gaussian elimination? Describe (without actually computing it) the method to solve this using the results above?

Exercise 1.4 (lower triangular matrices)**.** Write down three different 4×4 elementary lower triangular matrices $\mathbf{L}_i(\boldsymbol{x})$, and explain for each matrix what left-multiplication with this matrix does.
Verify explicitly that

$$\det \mathbf{L}_i(\boldsymbol{x}) = 1 \text{ and } \mathbf{L}_i(\boldsymbol{x})^{-1} = \mathbf{L}_i(-\boldsymbol{x}) \tag{P1.4.1}$$

for each of these three elementary lower triangular matrices.

Exercise 1.5 (lower triangular matrices). Find the elementary lower triangular matrices $\mathbf{L}_1(\boldsymbol{m}^1)$ and $\mathbf{L}_2(\boldsymbol{m}^2)$ which describe the Gaussian elimination to bring the linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{A} = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 1 & 2 \\ 2 & 2 & 4 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 1 \\ 8 \\ 4 \end{bmatrix} \tag{P1.5.1}$$

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

Exercise 1.6 (properties of lower triangular matrices). Prove that an elementary lower triangular $n \times n$ matrix $\mathbf{L}_i(\boldsymbol{x}) = \mathbf{I} - \boldsymbol{x}\mathbf{e}^{i\mathsf{T}}$ where

$$\boldsymbol{x} = \begin{bmatrix} 0 & \dots & 0 & x_{i+1} & \dots & x_n \end{bmatrix}^{\mathsf{T}} \tag{P1.6.1}$$

has the following properties:

(a) $\det \mathbf{L}_i(\boldsymbol{x}) = 1$.

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

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

$$x_k \begin{bmatrix} a_i^1 & a_i^2 & \dots & a_i^n \end{bmatrix} = x_k \boldsymbol{a}_i, \tag{P1.6.2}$$

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

Problem 1.7 (triangular matrix algebra). Consider the following sets of matrices.

$$\mathscr{U}(\mathbb{K}^n) := \left\{\boldsymbol{X} \in \mathbb{K}^{n\times n} : \boldsymbol{X} \text{ is upper triangular}\right\}, \tag{P1.7.1}$$

$$\mathscr{L}_1(\mathbb{K}^n) := \left\{\boldsymbol{X} \in \mathbb{K}^{n\times n} : \boldsymbol{X} \text{ is normalised lower triangular}\right\}, \tag{P1.7.2}$$

and

$$\mathscr{D}(\mathbb{K}^n) := \left\{\boldsymbol{X} \in \mathbb{K}^{n\times n} : \boldsymbol{X} \text{ is diagonal}\right\}. \tag{P1.7.3}$$

Prove the following:

(a) The sets $\mathscr{U}(\mathbb{K}^n)$ and $\mathscr{D}(\mathbb{K}^n)$ are closed under scaling, matrix addition and matrix multiplication, and are thus algebras of operators.

(b) The set $\mathscr{L}_1(\mathbb{K}^n)$ is closed under matrix multiplication and is a (non-Abelian) group with respect to this operation.

Problem 1.8 (invertibility criterion of upper triangular matrices). Prove that if $\boldsymbol{U} \in \mathbb{K}^{n\times n}$, $n \in \mathbb{N}$ is a upper triangular and $u_k^k = 0$ for some $k = 1, \dots, n$ then there exists a nonzero vector $\boldsymbol{v}$ such that $\boldsymbol{U}\boldsymbol{v} = \mathbf{0}$. Deduce that an upper triangular matrix is *invertible* if and only if it has all its diagonal entries are invertible.

Problem 1.9 (recursive definition of upper triangular). Prove the following recursive definition:
A matrix

$$\boldsymbol{U} = \begin{bmatrix} u & \boldsymbol{t}^{\mathsf{T}} \\ \boldsymbol{s} & \boldsymbol{R} \end{bmatrix} \tag{P1.9.1}$$

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

$$\boldsymbol{s} = \begin{cases} \varnothing & \text{for } n = 1 \\ \boldsymbol{0} & \text{for } n \geq 2 \end{cases} \text{ and } \boldsymbol{R} \in \mathscr{U}(\mathbb{K}^{n-1}). \tag{P1.9.2}$$

Problem 1.10 (upper triangular algebra). Denote by $\mathscr{U}(\mathbb{K}^n)$ the set of all upper triangular $n \times n$ matrices on a field $\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$.

(a) Prove that $\mathscr{U}(\mathbb{K}^n)$ is closed under matrix-matrix multiplication, addition and scaling. Deduce that $\mathscr{U}(\mathbb{K}^n)$ is a $\mathbb{K}$-vector space, and thus a $\mathbb{K}$-algebra.
(b) Show, with a counterexample, that $\mathscr{U}(\mathbb{K}^n)$ is *not* a group with respect to multiplication.
(c) Show that if $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$ is invertible, then $\boldsymbol{U}^{-1} \in \mathscr{U}(\mathbb{K}^n)$. That is, $\mathscr{U}(\mathbb{K}^n)$ is closed under matrix inversion. (Note that this is not in conflict with the fact that some elements of $\mathscr{U}(\mathbb{K}^n)$ are not invertible.)
(d) Show that $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$ is invertible if and only if $u_1^1 \neq 0$ and $\boldsymbol{U}_{[2...n]}^{[2...n]}$ is invertible. Deduce that $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$ is invertible if and only if $u_i^i \neq 0$ for all $i = 1, \ldots, n$.

Hint. You may assume the result of Problem 1.9.

Exercise 1.11 (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

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 1 & 2 \\ 2 & 2 & 4 \end{bmatrix}, \tag{P1.11.1}$$

of the form $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$ where $\boldsymbol{L}$ is a *normalised* lower triangular matrix and $\boldsymbol{U}$ an upper triangular matrix.
(b) Prove that there is only one such LU factorisation of $\boldsymbol{A}$.
(c) What happens if we drop the assumption that $\boldsymbol{L}$ be normalised?

Problem 1.12 (LU-cky matrices are invertible). Recall the following definition:
A square matrix $\boldsymbol{A} \in \mathbb{K}^{n \times n}$ written

$$\boldsymbol{A} = \begin{bmatrix} a & \boldsymbol{b}^\intercal \\ \boldsymbol{c} & \boldsymbol{D} \end{bmatrix} \tag{P1.12.1}$$

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

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

Recall also the LU-factorisation theorem:

If a matrix $\boldsymbol{A} \in \mathbb{K}^{n \times n}$ for some $n \in \mathbb{N}$ is LU-cky then $\boldsymbol{A}$ admits a unique LU factorisation, i.e., there exists a unique pair $\boldsymbol{L} \in \mathscr{L}_1(\mathbb{K}^n)$ and $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$ such that

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}. \tag{P1.12.2}$$

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

Problem 1.13 (LU-cky LU factorisation is unique). Let $\boldsymbol{A}$ 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 1.10 and 1.12 are established.

Problem 1.14 (permutation matrix properties). Prove that an elementary permutation matrix

$$\mathbf{P}_{i\leftrightarrow j} := \mathbf{I} - \left(\mathbf{e}^i - \mathbf{e}^j\right)\left(\mathbf{e}^i - \mathbf{e}^j\right)^\intercal \tag{P1.14.1}$$

enjoys the following properties:

(i) ${\mathbf{P}_{i\leftrightarrow j}}^{-1} = \mathbf{P}_{i\leftrightarrow j} = \mathbf{P}_{j\leftrightarrow i} = {\mathbf{P}_{i\leftrightarrow j}}^\intercal$. 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 $\boldsymbol{A}$ by $\mathbf{P}_{i\leftrightarrow j}$ interchanges *rows* i and j. Similarly, postmultiplication interchanges *columns* i and j.

Problem 1.15 (permutation to normal-lower-triangle commutator). Recall the definition of elementary lower triangular matrix

$$\mathbf{L}_i(\boldsymbol{m}) := \mathbf{I} - \boldsymbol{m}\mathbf{e}_i. \tag{P1.15.1}$$

Show that for any $\boldsymbol{m} \in \mathbb{K}^n$, with $m_1 = 0$, and $\boldsymbol{N}$ of the block-diagonal form

$$\boldsymbol{N} := \begin{bmatrix} 1 & \mathbf{0}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{N}} \end{bmatrix} \text{ where } \hat{\boldsymbol{N}} \in \mathbb{K}^{(n-1)\times(n-1)} \tag{P1.15.2}$$

we have

$$\boldsymbol{N}\mathbf{L}_1(\boldsymbol{m}) = \mathbf{L}_1(\boldsymbol{N}\boldsymbol{m})\boldsymbol{N} \text{ where } [\boldsymbol{N}\boldsymbol{m}]_1 = 0. \tag{P1.15.3}$$

Exercise 1.16 (PLU example). Apply the PLU algorithm, i.e., LU algorithm *with row pivoting*, to the following matrix

$$\begin{bmatrix} 0 & -2 & 3 \\ 2 & -6 & 2 \\ 3 & 0 & -3 \end{bmatrix}. \tag{P1.16.1}$$

Make sure you write all your steps and you keep track of all the algebraic operations that you have performed in the form of premultiplication with some matrix.

Problem 1.17 (programming PLU). The purpose of this problem is to produce a code for the LU factorisation with pivoting. Use your favourite computing language to solve it; but some parts are based on Octave/Matlab® notation.

(a) Implement a recursive code, say

```
octave:*> [L,U] = recursiveLU(A)
```

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

$$\tilde{\boldsymbol{A}} = \boldsymbol{A} - \boldsymbol{m}\,[\boldsymbol{A}]_1\,. \tag{P1.17.1}$$

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

$$\boldsymbol{A} = \mathbf{L}_1(-\boldsymbol{m})\tilde{\boldsymbol{A}} \text{ where } \tilde{\boldsymbol{A}} = \begin{bmatrix} a_1^1 & [\boldsymbol{A}]_1^{[2\dots n]} \\ \mathbf{0} & \hat{\boldsymbol{A}} \end{bmatrix}. \tag{P1.17.2}$$

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

$$\hat{\boldsymbol{A}} = \hat{\boldsymbol{L}}\hat{\boldsymbol{U}}. \tag{P1.17.3}$$

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) $\boldsymbol{P}_\pi$ characterised by

$$\boldsymbol{P}_\pi \mathbf{e}^j = \mathbf{e}^{\pi(j)}. \tag{P1.17.4}$$

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

$$\boldsymbol{P}^*_\pi \boldsymbol{P}_\pi = \mathbf{I} \qquad (\boldsymbol{P}_\pi \text{ is unitary}) \qquad \boldsymbol{P}^{-1}_\pi = \boldsymbol{P}_{\pi^{-1}} \tag{P1.17.5}$$

and thus

$$\boldsymbol{P}^*_\pi = \boldsymbol{P}_{\pi^{-1}}. \tag{P1.17.6}$$

Deduce that

$$[\boldsymbol{P}_\pi \boldsymbol{v}]_l = v_{\pi^{-1}(l)} \tag{P1.17.7}$$

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

$$\left[\boldsymbol{P}_\rho \boldsymbol{P}_\pi \boldsymbol{v}\right]_l = v_{\pi^{-1}(\rho^{-1}(l))}. \tag{P1.17.8}$$

(Watch the order of composition.)

(d) Base your answer on part (c). Think of a quick way of encoding an index-permutation π by using its *inverse-permutation vector* `pinverse` such that

$$\texttt{pinverse}(i) = \pi^{-1}(i). \tag{P1.17.9}$$

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

```
octave:*> vector(pinverse)
```

Use the following example to test all the objects described above

$$\pi(1) = 3, \pi(2) = 4, \pi(3) = 2, \pi(4) = 1. \tag{P1.17.10}$$

(e) Modify the LU factorisation code in the lecture notes 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

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

```
octave:*> myperm = myperm(myperm)
```

```
octave:*> myperm = myperm(myperm)
```

```
octave:*> ...
```

Once this works, write a function that takes any matrix $\boldsymbol{A}$ as input and returns $\boldsymbol{P} \in \mathscr{P}(n)$, $\boldsymbol{L} \in \mathscr{L}_1(\mathbb{K}^n)$ and $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$ such that

$$\boldsymbol{A} = \boldsymbol{PLU}. \tag{P1.17.11}$$

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

Problem 1.18 (tridiagonal LU). Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a tridiagonal matrix, that is, a matrix of the form

$$\boldsymbol{A} = \begin{bmatrix} a_1 & c_1 & & & \\ b_2 & a_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & b_n & a_n \end{bmatrix}, \tag{P1.18.1}$$

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

$$\begin{gathered} |a_1| > |c_1| > 0, \quad |a_n| \geq |b_n| > 0, \\ |a_i| \geq |b_i| + |c_i|, \qquad b_i, c_i \neq 0, \text{ for } i = 2, \dots, n-1, \end{gathered} \tag{P1.18.2}$$

Show that $\boldsymbol{A}$ is invertible and has an LU factorisation of the form

$$\boldsymbol{A} = \begin{bmatrix} 1 & & & 0 \\ l_2 & 1 & & \\ & \ddots & \ddots & \\ 0 & & l_n & 1 \end{bmatrix} \begin{bmatrix} r_1 & c_1 & & 0 \\ & r_2 & \ddots & \\ & & \ddots & c_{n-1} \\ 0 & & & r_n \end{bmatrix}, \tag{P1.18.3}$$

where the vectors $\boldsymbol{l} = (l_2, \dots, l_n) \in \mathbb{R}^{n-1}$ and $\boldsymbol{r} = (r_1, \dots, r_n) \in \mathbb{R}^n$ can be computed via

$$r_1 = a_1, l_i = b_i / r_{i-1} \text{ and } r_i = a_i - l_i\, c_{i-1} \text{ for } 2 \leq i \leq n. \tag{P1.18.4}$$

Hint. Work by induction on n.

Exercise 1.19 (matrix condition). For the matrix (zero entries omitted)

$$\boldsymbol{A} := \begin{bmatrix} 3/2 & & 1/2 \\ & 3 & \\ 1/2 & & 3/2 \end{bmatrix} \tag{P1.19.1}$$

compute its condition number with respect to the induced matrix p-norms, for $p \in \{1, 2, \infty\}$.

yy

CHAPTER 2

Matrix analysis

Indignum enim est excellentium vitrorum horas servili calculandi labore perire, qui machina adibita vilissimo cuique secure transcribi possert.

It is beneath the dignity of excellent men to waste their time in calculation when any peasant could do the work just as accurately with the aid of a machine.

– Gottfried Leibniz

Welcome my son, welcome to the machine. What did you dream?

– Roger Waters

We assume some basic knowledge on normed and scalar product vector spaces, most of which can be found in §B.2. In this Chapter we look at condition numbers and their usage in numerical analysis. We review some basic facts about inner (or scalar) product spaces (also known as pre-Hilbert spaces). We study symmetric matrices. We will also revisit LU factorisation, by looking how it works for self-adjoint (and real-symmetric) matrices.

2.1. Scalar product spaces

2.1.1. 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*:

$$\mathbb{K}^{n+n} \ni (\boldsymbol{x},\boldsymbol{y}) \mapsto \boldsymbol{y}^*\boldsymbol{x} \in \mathbb{R}, \tag{2.1.1}$$

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

$$\left[\boldsymbol{y}^*\right]^i = \overline{\left[\boldsymbol{y}\right]_i} = \overline{y_i}. \tag{2.1.2}$$

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

$$|\boldsymbol{x}| = \sqrt{\boldsymbol{x}^*\boldsymbol{x}} \in \mathbb{R}_0^+. \tag{2.1.3}$$

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

$$\boldsymbol{x}^*\boldsymbol{y} \leq |\boldsymbol{x}|\,|\boldsymbol{y}| \tag{2.1.4}$$

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

$$\cos\operatorname{angle}\left(\boldsymbol{x},\boldsymbol{y}\right) := \frac{\boldsymbol{y}^*\boldsymbol{x}}{|\boldsymbol{y}|\,|\boldsymbol{x}|} \tag{2.1.5}$$

By the properties of the inner product and Cauchy–Bunyakovsky–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 $\boldsymbol{x}$ and $\boldsymbol{y}$ is 0 if and only if $\boldsymbol{y}^*\boldsymbol{x} = 0$, in which case $\boldsymbol{y}$ and $\boldsymbol{x}$ are said to form an *orthogonal pair.*

2.1.2. 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 $\mathcal{A} \in \operatorname{Lin}(V \to W)$ is a linear operator $\mathcal{B} \in \operatorname{Lin}(W \to V)$ (note the reversed roles of V and W) such that

$$\langle v, \mathcal{B}w\rangle_V = \langle \mathcal{A}v, w\rangle_W \quad \forall\, v \in V, w \in W. \tag{2.1.6}$$

2.1.3. Proposition (existence and uniqueness of the adjoint operator). *If two inner product space V, W are Hilbert spaces, then each operator $\mathcal{A} \in \operatorname{Lin}(V \to W)$ admits a unique adjoint operator, which is denoted by $\mathcal{A}^*$, in* $\operatorname{Lin}(W \to V)$.
Proof The operator $\mathscr{A}$ gives rise to a sesquilinear form as follows:

$$\begin{array}{rccc} b: & V \times W & \to & \mathbb{K} \\ & (v, w) & \mapsto & \langle \mathcal{A}v, w\rangle_W \end{array} . \tag{2.1.7}$$

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

$$\langle v, \mathcal{B}w\rangle_V = b(v, w) \quad \forall\, (v, w) \in V \times W. \tag{2.1.8}$$

Therefore the operator $\mathcal{B}$ satisfies

$$\langle v, \mathcal{B}w\rangle_V = \langle \mathcal{A}v, w\rangle_W \quad \forall\, (v, w) \in V \times W, \tag{2.1.9}$$

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

$$\langle v, \mathcal{C}w\rangle_V = \langle \mathcal{A}v, w\rangle_W \quad \forall\, (v, w) \in V \times W. \tag{2.1.10}$$

It follows that

$$\langle v, (\mathcal{C} - \mathcal{B})\, w\rangle_V = 0 \quad \forall\, (v, w) \in V \times W. \tag{2.1.11}$$

For any $w \in W$, choosing $v := (\mathcal{C} - \mathcal{B})\, w$ in (2.1.11) we obtain

$$\langle (\mathcal{C} - \mathcal{B})\, w, (\mathcal{C} - \mathcal{B})\, w\rangle_V = 0, \tag{2.1.12}$$

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

$$(\mathcal{C} - \mathcal{B})\, w = 0. \tag{2.1.13}$$

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

2.1.4. Definition of adjoint matrix. Similarly to adjoint operator we define the *adjoint of a matrix $\boldsymbol{A}$* to be that unique matrix $\boldsymbol{A}^*$ such that

$$(\boldsymbol{x}^*)(\boldsymbol{A}^*)\boldsymbol{y} = (\boldsymbol{A}\boldsymbol{x})^*\boldsymbol{y} \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{K}^n. \tag{2.1.14}$$

(Over-bracketing intentional to convey meaning.)

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

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

$$\boldsymbol{A}^* = \overline{\boldsymbol{A}^\mathsf{T}}. \tag{2.1.15}$$

2.2. Orthogonality

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

$$\left\langle \mathcal{Q}x, \mathcal{Q}y \right\rangle = \left\langle x, y \right\rangle \quad \forall\, x, y \in V. \tag{2.2.1}$$

2.2.2. Proposition. *An operator* $\mathcal{Q} \in \operatorname{Lin}(V \to V)$ *is orthogonal (or unitary) if and only if*

$$\mathcal{Q}^*\mathcal{Q} = \mathcal{I} = \mathrm{id}_V\,. \tag{2.2.2}$$

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

$$\boldsymbol{Q}^*\boldsymbol{Q} = \mathbf{I}. \tag{2.2.3}$$

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

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

$$\langle v, w\rangle = \delta_v^w \quad \forall\, v, w \in \mathscr{S}. \tag{2.2.4}$$

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

$$v = \sum_{i\in I} \alpha_i v^i, \tag{2.2.5}$$

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

$$\sum_{|\in| alpha_i} \left\| v^i \right\| < \infty. \tag{2.2.6}$$

2.2.6. Proposition (chracterisation of unitary matrices). *For a matrix* $\boldsymbol{Q} \in \mathbb{K}^{n\times n}$*, the following are equivalent*

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

Proof Exercise. □

2.2.7. 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 $\mathbf{Q}$ *has determinant equal* ± 1.

Proof Exercise. □

2.2.8. Example. The matrices

$$\begin{bmatrix} \sqrt{3}/2 & -1/2 \\ 1/2 & \sqrt{3}/2 \end{bmatrix}, \text{ and } \frac{1}{\sqrt{3}}\begin{bmatrix} \mathrm{i} & -1-\mathrm{i} \\ 1+\mathrm{i} & -\mathrm{i} \end{bmatrix} \tag{2.2.7}$$

are unitary. The first one is also orthogonal.
The following matrix is orthogonal

$$\begin{bmatrix} 1/\sqrt{3} & & -\sqrt{2}/\sqrt{3} \\ -1/\sqrt{3} & 1/\sqrt{2} & -1/\sqrt{6} \\ 1/\sqrt{3} & 1/\sqrt{2} & 1/\sqrt{6} \end{bmatrix} \tag{2.2.8}$$

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

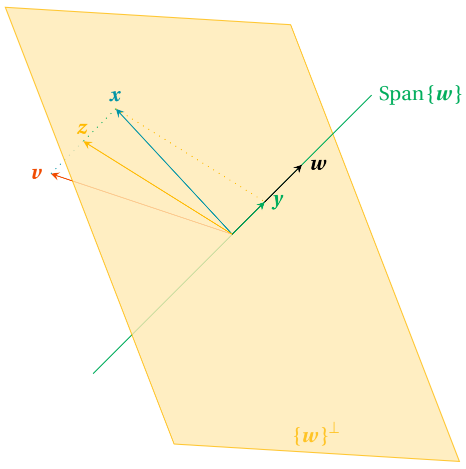
$$\begin{aligned} \mathrm{O}(n) &:= \left\{ \boldsymbol{Q} \in \mathbb{R}^{n\times n} : \boldsymbol{Q}^*\boldsymbol{Q} = \mathbf{I} \right\} = \left\{ \boldsymbol{Q} \in \mathbb{R}^{n\times n} : \boldsymbol{Q}^\intercal\boldsymbol{Q} = \mathbf{I} \right\} \\ \mathrm{U}(n) &:= \left\{ \boldsymbol{Q} \in \mathbb{C}^{n\times n} : \boldsymbol{Q}^*\boldsymbol{Q} = \mathbf{I} \right\} \end{aligned} \tag{2.2.9}$$

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

$$\mathrm{O}(n) = \mathrm{U}(n) \cap \mathbb{R}^{n\times n}. \tag{2.2.10}$$

2.2.11. Example (plane rotations). The rotations in the plane constitute a group of orthogonal transformations.

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

$$\boldsymbol{v} = \boldsymbol{z} - \boldsymbol{y} = \boldsymbol{x} - \boldsymbol{y} - \boldsymbol{y} = \boldsymbol{x} - 2\boldsymbol{y}. \tag{2.2.11}$$

The vector $\boldsymbol{y}$ is given by

$$\boldsymbol{y} = \frac{\boldsymbol{w}^*\boldsymbol{x}}{\sqrt{\boldsymbol{w}^*\boldsymbol{w}}}\frac{\boldsymbol{w}}{\sqrt{\boldsymbol{w}^*\boldsymbol{w}}} = \frac{\boldsymbol{w}\boldsymbol{w}^*}{\boldsymbol{w}^*\boldsymbol{w}}\boldsymbol{x}. \tag{2.2.12}$$

Summarising we can write $\boldsymbol{v}$ as a linear transformation $\boldsymbol{H}$ acting on $\boldsymbol{x}$:

$$\boldsymbol{H}\boldsymbol{x} := \boldsymbol{v} = \boldsymbol{x} - 2\boldsymbol{y} = \boldsymbol{x} - 2\frac{\boldsymbol{w}\boldsymbol{w}^*}{\boldsymbol{w}^*\boldsymbol{w}}\boldsymbol{x} = \left(\mathbf{I} - 2\frac{\boldsymbol{w}\boldsymbol{w}^*}{\boldsymbol{w}^*\boldsymbol{w}}\right)\boldsymbol{x}. \tag{2.2.13}$$

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

$$\mathbf{I} - 2\frac{\boldsymbol{w}\boldsymbol{w}^*}{\boldsymbol{w}^*\boldsymbol{w}} =: \boldsymbol{H}. \tag{2.2.14}$$

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

2.3. Schur factorisation

Schur factorisation is a central result in Linear Algebra. While the result is very important to Mathematics in general, its proof (the one we present here) is in fact as important as the statement for Numerical Analysis.

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

$$\mathbf{H}(\boldsymbol{w}) := \begin{cases} \mathbf{I} & \text{if } \boldsymbol{w} = \mathbf{0} \\ \mathbf{I} - 2\frac{\boldsymbol{w}\boldsymbol{w}^*}{\boldsymbol{w}^*\boldsymbol{w}} & \text{if } \boldsymbol{w} \neq \mathbf{0}. \end{cases} \tag{2.3.1}$$

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

$$\mathbf{H}(\boldsymbol{w}) \in \mathrm{U}(n), \tag{2.3.2}$$

$$\mathbf{H}(\boldsymbol{w}) = \mathbf{H}(\lambda\boldsymbol{w}) \quad \forall\,\lambda \in \mathbb{K}, \tag{2.3.3}$$

$$\mathbf{H}(\boldsymbol{w})^* = \mathbf{H}(\boldsymbol{w}), \tag{2.3.4}$$

$$\mathbf{H}(\boldsymbol{w})^{-1} = \mathbf{H}(\boldsymbol{w}). \tag{2.3.5}$$

In words, Householder matrices are unitary, depend only on the direction of the defining vector, self-adjoint and involutive. (A matrix $\boldsymbol{M}$ is involutive or an involution if and only if $\boldsymbol{M}\boldsymbol{M} = \mathbf{I}$.)
Proof Exercise. □

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

$$\mathbf{H}(\boldsymbol{u} - \boldsymbol{v})\boldsymbol{u} = \boldsymbol{v}. \tag{2.3.6}$$

Proof Exercise. □

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

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

$$\boldsymbol{A} = \boldsymbol{Q}^* \boldsymbol{T} \boldsymbol{Q}. \tag{2.3.7}$$

Proof We proceed by induction on n. The base case, $n = 1$ is trivial with $\boldsymbol{Q} = 1$ and $\boldsymbol{T} = \boldsymbol{A}$. 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 $\boldsymbol{A}$ has at least one eigenvalue $\lambda \in \mathbb{C}$ with eigenvector $\boldsymbol{v}$ which we consider of unit length without loss of generality (replace $\boldsymbol{v}$ by $\boldsymbol{v}/|\boldsymbol{v}|$). Consider now the matrix

$$\boldsymbol{H} := \mathbf{H}(\boldsymbol{v} - \mathbf{e}^1). \tag{2.3.8}$$

By Lemma 2.3.3 we know that

$$\boldsymbol{H}\boldsymbol{v} = \mathbf{e}^1 \text{ and } \boldsymbol{H}\mathbf{e}^1 = \boldsymbol{v} = \boldsymbol{H}^*\mathbf{e}^1. \tag{2.3.9}$$

Therefore, we may write

$$\boldsymbol{A}\boldsymbol{H}^*\mathbf{e}^1 = \boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v} = \lambda\boldsymbol{H}^*\mathbf{e}^1 = \boldsymbol{H}^*\lambda\mathbf{e}^1 \tag{2.3.10}$$

thus

$$\boldsymbol{H}\boldsymbol{A}\boldsymbol{H}^*\mathbf{e}^1 = \lambda\mathbf{e}^1. \tag{2.3.11}$$

In other words, this says that

$$\boldsymbol{H}\boldsymbol{A}\boldsymbol{H}^* = \begin{bmatrix} \lambda & \boldsymbol{b}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{A}} \end{bmatrix}, \tag{2.3.12}$$

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

$$\hat{\boldsymbol{A}} = \hat{\boldsymbol{Q}}^* \hat{\boldsymbol{T}} \hat{\boldsymbol{Q}}. \tag{2.3.13}$$

It follow that

$$\begin{bmatrix} \lambda & \boldsymbol{b}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{A}} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{Q}}^* \end{bmatrix} \begin{bmatrix} \lambda & \boldsymbol{b}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{T}} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{Q}} \end{bmatrix} = \widetilde{\boldsymbol{Q}}^* \boldsymbol{T} \widetilde{\boldsymbol{Q}} \tag{2.3.14}$$

where

$$\widetilde{\boldsymbol{Q}} := \begin{bmatrix} 1 & \mathbf{0}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{Q}} \end{bmatrix} \text{ and } \boldsymbol{T} := \begin{bmatrix} \lambda & \boldsymbol{b}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{T}} \end{bmatrix}. \tag{2.3.15}$$

From this and (2.3.12) it follows that

$$\boldsymbol{A} = \boldsymbol{H}^* \widetilde{\boldsymbol{Q}}^* \boldsymbol{T} \widetilde{\boldsymbol{Q}} \boldsymbol{H}. \tag{2.3.16}$$

The result will now follow provided we take

$$\boldsymbol{Q} := \widetilde{\boldsymbol{Q}} \boldsymbol{H}, \tag{2.3.17}$$

and we check that $\boldsymbol{Q}$ is unitary. But this follows from the group structure of $\mathrm{U}(n)$ and $\boldsymbol{H}, \widetilde{\boldsymbol{Q}} \in \mathrm{U}(n)$. □

2.4. Self-adjoint operators and matrices

2.4.1. Definition of self-adjoint, symmetric and Hermitian operators and matrices. An operator $\mathcal{A} \in \operatorname{Lin}(V \to V)$, V scalar product $\mathbb{K}$-vector space, or a matrix $\boldsymbol{A} \in \mathbb{K}^{n\times n}$, $n \in \mathbb{N}$, are called *self-adjoint* if and only if

$$\mathcal{A}^* = \mathcal{A} \text{ or } \boldsymbol{A}^* = \boldsymbol{A}, \text{ respectively.} \tag{2.4.1}$$

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.*

2.4.2. Exercise (self-adjoint operators and matrices). *If $\mathscr{V}$ is a finite dimensional scalar product $\mathbb{K}$-vector space and operators on $\mathscr{V}$ are identified with matrices in $\mathbb{K}^{n\times n}$ by the means of representation with respect to an orthonormal basis of $\mathscr{V}$, then an operator $\mathcal{A} \in \operatorname{Lin}(\mathscr{V} \to \mathscr{V})$ is self-adjoint if and only if the matrix $\boldsymbol{A}$ that represents $\mathcal{A}$ is self-adjoint.*

2.4.3. Exercise (self-adjoint operators from a vector space). *Show that the set of all self-adjoint operators on a scalar product vector space V is a $\mathbb{R}$-vector space. Is it an algebra? Is it a $\mathbb{C}$-vector space?*

2.4.4. Exercise. *Show that if $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ is self-adjoint then its diagonal entries a^i_i, $i = 1,\dots,n$, are all real.*

2.4.5. Exercise. *Suppose $\boldsymbol{A}$ is self-adjoint, show that its Schur factorisation is equal to a diagonalisation, i.e., that $\boldsymbol{T}$ in (2.3.7) 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.

2.5. Self-adjoint positive (SPD) matrices

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

$$\langle x, \mathcal{A}x\rangle \geq \alpha \langle x, x\rangle \quad \forall\, x \in V, \tag{2.5.1}$$

or, for the matrix case,

$$\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x} \geq \alpha \boldsymbol{x}^*\boldsymbol{x} \quad \forall\, \boldsymbol{x} \in \mathbb{K}^n. \tag{2.5.2}$$

This property, and by extension the constant α, is sometimes referred to as *coercivity.* We say that the operator $\mathcal{A}$, or the matrix $\boldsymbol{A}$, is *positive semidefinite* if and only if (2.5.1), or (2.5.2), respectively, is satisfied with $\alpha \geq 0$.

2.5.2. Exercise. *Show that if $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ is positive definite (not necessarily self-adjoint) then its diagonal entries a^i_i, $i = 1,\dots,n$, are all positive real numbers.*

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

2.5.4. Exercise (algebra of SPD matrices). *Show the following statements.*

(a) $\mathrm{SPD}(\mathbb{K}^n)$ *is closed under addition, i.e.,*

$$\boldsymbol{A}, \boldsymbol{B} \in \mathrm{SPD}(\mathbb{K}^n) \Rightarrow \boldsymbol{A} + \boldsymbol{B} \in \mathrm{SPD}(\mathbb{K}^n). \tag{2.5.3}$$

(b) $\mathrm{SPD}(\mathbb{K}^n)$ *is closed under scaling with positive real numbers, i.e.,*

$$\lambda \in \mathbb{R}^+ \text{ and } \boldsymbol{A} \in \mathrm{SPD}(\mathbb{K}^n) \Rightarrow \lambda \boldsymbol{A} \in \mathrm{SPD}(\mathbb{K}^n). \tag{2.5.4}$$

(c) $\mathrm{SPD}(\mathbb{K}^n)$ *is not closed under scaling.*

(d) $\mathrm{SPD}(\mathbb{K}^n)$ *is closed under matrix inversion, i.e.,*

$$\boldsymbol{A} \in \mathrm{SPD}(\mathbb{K}^n) \Rightarrow \boldsymbol{A}^{-1} \in \mathrm{SPD}(\mathbb{K}^n). \tag{2.5.5}$$

(e) $\mathrm{SPD}(\mathbb{K}^n)$ *is not closed under matrix multiplication for $n > 1$.*

2.5.5. Remark ($\mathrm{SPD}(\mathbb{K}^n)$ **is not an algebra nor a vector space).** There can be no opposite in $\mathrm{SPD}(\mathbb{K}^n)$. Explicitly, if $\boldsymbol{A} \in \mathrm{SPD}(\mathbb{K}^n)$, then $-\boldsymbol{A} \notin \mathrm{SPD}(\mathbb{K}^n)$. So $\mathrm{SPD}(\mathbb{K}^n)$ is not a subspace of $\mathbb{K}^{n\times n}$.
$\mathrm{SPD}(\mathbb{K}^n)$ is also not a group with respect to multiplication, except when $n = 1$.

2.5.6. Exercise (the Grammian is SPD). *Suppose $\boldsymbol{A} = \boldsymbol{B}^*\boldsymbol{B}$ for some invertible matrix $\boldsymbol{B} \in \mathbb{K}^{n\times n}$ then $\boldsymbol{A} \in \mathrm{SPD}(\mathbb{K}^n)$. The matrix $\boldsymbol{A}$ is the Grammian of $\boldsymbol{B}$.*

2.5.7. Definition of normal matrix. A matrix $\boldsymbol{A}$ is called normal if and only if if commutes with its adjoint, i.e., algebraically speaking one has,

$$\boldsymbol{A}^*\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^*. \tag{2.5.6}$$

2.5.8. Exercise. *Normal matrices are an important class of matrices, which includes all self-adjoint and all orthogonal matrices. Prove these two facts.*

2.5.9. Theorem (spectral). *A matrix $\boldsymbol{A} \in \mathbb{K}^{n\times}$ is normal if and only if $\boldsymbol{A}$ admits an orthonormal system of eigenvectors $\boldsymbol{q}^1, \dots, \boldsymbol{q}^n$ corresponding to the eigenvalues $\lambda_1, \dots, \lambda_n$ for which*

$$\boldsymbol{A} = \boldsymbol{Q}^*\boldsymbol{\Lambda}\boldsymbol{Q}. \tag{2.5.7}$$

Exercises and problems on matrix analysis and symmetry

Exercise 2.1 (spectral radius and diagonalisable matrices). Suppose $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ is diagonalisable, i.e., for an invertible matrix $\boldsymbol{V}$ we have

$$\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} =: \Lambda = \operatorname{Diag}\left(\lambda_1^1, \dots, \lambda_n^n\right). \tag{P2.1.1}$$

Show that for any matrix (operator) norm $|\cdot|$, the spectral radius of $\boldsymbol{A}$ is bounded by

$$|\boldsymbol{A}| \leq \operatorname{cond} \boldsymbol{V} \, |\boldsymbol{\Lambda}|, \tag{P2.1.2}$$

where $\operatorname{cond} \boldsymbol{V}$ is the condition of $\boldsymbol{V}$ with resepect to the norm $|\cdot|$. Deduce that if $|\cdot|$ is the Euclidean-induced matrix norm then $\lambda_\sharp$ is the largest eigenvalue of $\boldsymbol{A}$ in absolute value then

$$|\boldsymbol{A}| \leq \operatorname{cond} \boldsymbol{V} \left|\lambda_\sharp\right|. \tag{P2.1.3}$$

Exercise 2.2 (unitary matrix). A matrix $\boldsymbol{Q} \in \mathbb{C}^{n\times n}$ is called *unitary* if and only if

$$\boldsymbol{Q}^*\boldsymbol{Q} = \mathbf{I}. \tag{P2.2.1}$$

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 (often denoted as $\mathrm{U}(n)$).

Exercise 2.3 (orthogonal matrix algebra). Show that if $\boldsymbol{Q}$ is an orthogonal matrix then so is $-\boldsymbol{Q}$.

2.3.1. Problem. *Prove the following:*
For a matrix $\boldsymbol{Q} \in \mathbb{K}^{n\times n}$, the following are equivalent

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

Exercise 2.4 (unitary matrices conserve matrix 2-norm). Let $\boldsymbol{Q} \in \mathbb{C}^{n\times n}$ be a unitary matrix, and let $\boldsymbol{x} \in \mathbb{C}^n$ and $\boldsymbol{A} \in \mathbb{C}^{n\times p}$. Show that

$$|\boldsymbol{Q}\boldsymbol{x}|_2 = |\boldsymbol{x}|_2 \text{ and } |\boldsymbol{Q}\boldsymbol{A}|_2 = |\boldsymbol{A}|_2. \tag{P2.4.1}$$

Exercise 2.5 (basic properties of Householder matrices). Recall the definition of a Householder matrix. Let $\boldsymbol{w} \in \mathbb{K}^n$, define the associate *Householder matrix* to be

$$\mathbf{H}(\boldsymbol{w}) := \begin{cases} \mathbf{I} & \text{if } \boldsymbol{w} = \mathbf{0} \\ \mathbf{I} - 2\frac{\boldsymbol{w}\boldsymbol{w}^*}{\boldsymbol{w}^*\boldsymbol{w}} & \text{if } \boldsymbol{w} \neq \mathbf{0}. \end{cases} \tag{P2.5.1}$$

Let $\boldsymbol{w} \in \mathbb{K}^n$, prove the following properties of Householder matrices

$$\mathbf{H}(\boldsymbol{w}) \in \mathrm{U}(n), \tag{P2.5.2}$$

$$\mathbf{H}(\boldsymbol{w}) = \mathbf{H}(\lambda\boldsymbol{w}) \quad \forall\, \lambda \in \mathbb{K}, \tag{P2.5.3}$$

$$\mathbf{H}(\boldsymbol{w})^* = \mathbf{H}(\boldsymbol{w}), \tag{P2.5.4}$$

$$\mathbf{H}(\boldsymbol{w})^{-1} = \mathbf{H}(\boldsymbol{w}). \tag{P2.5.5}$$

Exercise 2.6 (Householder matrix). Construct a Householder matrix $\boldsymbol{H}$ that maps the vector $\boldsymbol{x} = (4,0,3)$ onto the vector $\boldsymbol{y} = (5,0,0)$, by checking first that $|\boldsymbol{x}| = \left|\boldsymbol{y}\right|$ and then designing a unit vector $\boldsymbol{w}$ such that $\boldsymbol{H} = \mathbf{H}(\boldsymbol{w}) := \mathbf{I} - 2\frac{\boldsymbol{w}\boldsymbol{w}^*}{\boldsymbol{w}^*\boldsymbol{w}}$.

Problem 2.7 (self-adjoint operators and matrices). If $\mathscr{V}$ is a finite dimensional scalar product $\mathbb{K}$-vector space and operators on $\mathscr{V}$ are identified with matrices in $\mathbb{K}^{n\times n}$ by the means of representation with respect to an orthonormal basis of $\mathscr{V}$, then an operator $\mathcal{A} \in \operatorname{Lin}(\mathscr{V} \to \mathscr{V})$ is self-adjoint if and only if the matrix $\boldsymbol{A}$ that represents $\mathcal{A}$ is self-adjoint.

Exercise 2.8 (self-adjoint and algebra). Show that the set of all self-adjoint operators on a scalar product vector space V is a $\mathbb{R}$-vector space. Is it an algebra? Is it a $\mathbb{C}$-vector space?

Exercise 2.9 (a self-adjoint matrix has a real diagonal). Show that if $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ is self-adjoint then its diagonal entries a^i_i, $i = 1,\dots,n$, are all real.

Problem 2.10 (Schur factorisation and SPD). Let $\boldsymbol{A}$ be a self-adjoint (also known as Hermitian) matrix, i.e., $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ and

$$\boldsymbol{A}^* = \boldsymbol{A}. \tag{P2.10.1}$$

(a) Using the Schur factorisation, show that there exists a unitary matrix $\boldsymbol{Q}$ such that $\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{D}$, where $\boldsymbol{D}$ is a diagonal matrix with *real* coefficients.
(b) Argue that the eigenvalues of $\boldsymbol{A}$ are all real.
(c) Use the Schur factorisation to show that $\mathbb{C}^n$ has an orthonormal basis of eigenvectors of $\boldsymbol{A}$.

Exercise 2.11 (a positive matrix has positive diagonal entries). (a) Show that if $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ is positive definite (not necessarily self-adjoint) then its diagonal entries a^i_i, $i = 1,\dots,n$, are all positive real numbers.
(b) What can you say about the principal blocks of a positive matrix $\boldsymbol{A} \in \mathbb{K}^{n\times n}$. A *principal block* is a block of the form $\boldsymbol{A}^{[k\dots l]}_{[k\dots l]}$.

Problem 2.12 (Schur factorisation and postivity). (a) Show that $\boldsymbol{A}$ is positive definite if and only if all eigenvalues are positive.
(b) Show that if $\boldsymbol{A}$ is positive definite, then $\det \boldsymbol{A} > 0$.
(c) Show that $\boldsymbol{A}$ is self-adjoint positive definite if and only if $\boldsymbol{A} = \boldsymbol{B}^*\boldsymbol{B}$ for some invertible matrix $\boldsymbol{B}$.

Problem 2.13 (SPD matrices). 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 $\boldsymbol{A}$ has all eigenvalues real.
(b) Show that an SPD matrix $\boldsymbol{A}$ has all its eigenvalues positive (and thus real).
(c) Using the Schur factorisation theorem show that for some invertible matrix $\boldsymbol{S}$ and diagonal matrix $\boldsymbol{\Lambda}$ we have

$$\boldsymbol{A} = \boldsymbol{S}^{-1}\boldsymbol{\Lambda}\boldsymbol{S}. \tag{P2.13.1}$$

CHAPTER 3

Orthogonality based factorisations

"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]

Addendum: John Francis, a long time resident of Hove, East Sussex in England, was awarded a Doctorate of Sciences by the University of Sussex in July 2015 at the age of 80. He gave an inspiring talk at the award ceremony which can be accessed online. See `http://www.sussex.ac.uk/broadcast/read/31082`

In Chapter 1 we have learned the LU factorisation and some of its variants. We now turn our attention to other factorisation methods, some of which are very close to LU, such as Cholesky, based on orthogonal and unitary matrices, such as the Schur factorisation and the QR factorisation.

3.1. Cholesky's factorisation

3.1.1. Cholesky's factorisation. *Cholesky's algorithm* provides a fast way of performing an LU-type factorisation for a self-adjoint positive (SPD) matrix $\boldsymbol{A}$; i.e.,

$$\boldsymbol{A}^* = \boldsymbol{A} \tag{3.1.1}$$

and for some $\alpha > 0$

$$\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x} \geq \alpha \boldsymbol{x}^*\boldsymbol{x} \; \forall \, \boldsymbol{x} \in \mathbb{K}^n. \tag{3.1.2}$$

Symmetry (or more generally, self-adjoint nature) of $\boldsymbol{A}$, is exploited in that the factorisation is sought to in the form

$$\boldsymbol{A} = \boldsymbol{U}^*\boldsymbol{U} \text{ for one matrix } \boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n). \tag{3.1.3}$$

As we know from §2.5.6, a necessary condition for such a factorisation is that $\boldsymbol{A}$ is symmetric and positive. It turns out that this is also sufficient.

3.1.2. Theorem (Cholesky factorisation). *Given a matrix $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ that is self-adjoint and positive (semidefinite) there exists a matrix $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$ that satisfies (3.1.3).*
Proof It is reasonable to expect such a factorisation because of the existence of an LDU factorisation of $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{V}$, where $\boldsymbol{L}$ and $\boldsymbol{V}^*$ are in $\mathscr{L}_1(\mathbb{K}^n)$ and $\boldsymbol{D}$ diagonal. Uniqueness of the LDU factorisation and the fact that $\boldsymbol{A}^* = \boldsymbol{A}$ imply then that $\boldsymbol{L} = \boldsymbol{V}^*$. The fact that $\boldsymbol{A}$ is positive implies that $\boldsymbol{D}$ is positive too and thus $\boldsymbol{D} \in \text{SPD}(\mathbb{K}^n)$. Because $\boldsymbol{D}$ is diagonal, its entries are all positive and it can be easily factored as

$$\boldsymbol{D} = \boldsymbol{E}^*\boldsymbol{E}, \text{ where } \boldsymbol{E} = \operatorname{Diag}\left(\sqrt{d_1^1}, \dots, \sqrt{d_n^n}\right). \tag{3.1.4}$$

Writing everything in terms of $\boldsymbol{V}$ and $\boldsymbol{E}$ we obtain

$$\boldsymbol{A} = \boldsymbol{V}^*\boldsymbol{E}^*\boldsymbol{E}\boldsymbol{V} = \boldsymbol{U}^*\boldsymbol{U} \text{ where } \boldsymbol{U} = \boldsymbol{E}\boldsymbol{V}. \tag{3.1.5}$$

Thus (although the choice of $\boldsymbol{E}$ is not unique) we have established the result. □

3.1.3. Remark. The matrix $\boldsymbol{U}$ is invertible if and only if $\boldsymbol{U}^*\boldsymbol{U}$ is SPD($\mathbb{K}^n$).

3.1.4. Problem (Uniqueness of Cholesky's factorisation). *Show that Cholesky's factorisation is not unique in general.*
Hint. Think of the equation $|u|^2 = a$ for $u \in \mathbb{K}$ when $a \in \mathbb{R}^+$.
Hint. Suppose $\boldsymbol{A} = \boldsymbol{U}^*\boldsymbol{U}$ with $\boldsymbol{U} \in \mathscr{U}(\mathbb{K}^n)$, find another matrix $\boldsymbol{V} \in \mathscr{U}(\mathbb{K}^n)$ of the form $\boldsymbol{Q}\boldsymbol{U}$ with $\boldsymbol{Q}$ diagonal and such that $\boldsymbol{Q}^*\boldsymbol{Q} = \mathbf{I}$.

3.1.5. Derivation of Cholesky's algorithm. We now describe a recursive way to get a factorisation of the form (3.1.3). Suppose that the given matrix $\boldsymbol{A}$ is self-adjoint ($\boldsymbol{A}^* = \boldsymbol{A}$) hence writable in the $(1, n-1)\times(1, n-1)$ block-form

$$\boldsymbol{A} = \begin{bmatrix} a & \boldsymbol{b}^* \\ \boldsymbol{b} & \boldsymbol{C} \end{bmatrix}. \tag{3.1.6}$$

To make it simple, we assume first that $\boldsymbol{A}$ is invertible. We try to find a matrix $\boldsymbol{U} \in \mathscr{U}(\mathbb{C}^n)$ such that (3.1.3). In block-form this reads as

$$\begin{bmatrix} a & \boldsymbol{b}^* \\ \boldsymbol{b} & \boldsymbol{C} \end{bmatrix} = \begin{bmatrix} u^* & \boldsymbol{0}^* \\ \hat{\boldsymbol{u}} & \hat{\boldsymbol{U}}^* \end{bmatrix}\begin{bmatrix} u & \hat{\boldsymbol{u}}^* \\ 0 & \hat{\boldsymbol{U}} \end{bmatrix} = \begin{bmatrix} u^* u & u^*\hat{\boldsymbol{u}}^* \\ \hat{\boldsymbol{u}} u & \hat{\boldsymbol{u}}\hat{\boldsymbol{u}}^* + \hat{\boldsymbol{U}}^*\hat{\boldsymbol{U}} \end{bmatrix} \tag{3.1.7}$$

and leads, by equating corresponding blocks, to the following equations for $\boldsymbol{U}$:

$$a = u^* u = |u|^2, \quad \boldsymbol{b} = \hat{\boldsymbol{u}}\, u, \text{ and} \tag{3.1.8}$$

$$\boldsymbol{C} = \hat{\boldsymbol{u}}\hat{\boldsymbol{u}}^* + \hat{\boldsymbol{U}}^*\hat{\boldsymbol{U}}. \tag{3.1.9}$$

The first two are solvable since $a > 0$ as follows

$$u := \sqrt{a}\, \mathrm{e}^{\mathrm{i}\theta} \text{ for any } \theta \in [0, 2\pi) \qquad \left(\text{pick the unique real positive one, i.e., } \sqrt{a}\right), \tag{3.1.10}$$

and

$$\hat{\boldsymbol{u}} := \boldsymbol{b}\, u^{-1}. \tag{3.1.11}$$

The last equation in (3.1.9)consists in finding $\hat{\boldsymbol{U}} \in \mathscr{U}(\mathbb{K}^n)$ such that

$$\hat{\boldsymbol{U}}^*\hat{\boldsymbol{U}} = \boldsymbol{C} - \hat{\boldsymbol{u}}\hat{\boldsymbol{u}}^* =: \hat{\boldsymbol{A}}. \tag{3.1.12}$$

In order to find $\hat{\boldsymbol{U}}$ it would be sufficient to show that $\hat{\boldsymbol{A}}$ is SPD and apply recursion (or the inductive hypothesis in a properly constructed proof by induction on n). For this consider an arbitrary "short" vector $\hat{\boldsymbol{x}} \in \mathbb{K}^{n-1}$ and look at

$$\hat{\boldsymbol{x}}^*\hat{\boldsymbol{A}}\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}^*\boldsymbol{C}\hat{\boldsymbol{x}} - \hat{\boldsymbol{x}}^*\hat{\boldsymbol{u}}\hat{\boldsymbol{u}}^*\hat{\boldsymbol{x}}. \tag{3.1.13}$$

But

$$\hat{\boldsymbol{x}}^*\hat{\boldsymbol{u}}\hat{\boldsymbol{u}}^*\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}^*\boldsymbol{b}\, u^{-1} u^{-1}\boldsymbol{b}^*\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}^*\boldsymbol{b}\, a^{-1}\boldsymbol{b}^*\hat{\boldsymbol{x}}. \tag{3.1.14}$$

It follows (see 3.1.6) that the matrix $\hat{\boldsymbol{A}}$ is in SPD($\mathbb{K}^{n-1}$).

3.1.6. Problem (SPD matrices and their submatrices). *Consider* $\boldsymbol{A} \in$ SPD($\mathbb{K}^n$) *with coercivity* $\alpha > 0$*:*

$$\alpha \boldsymbol{x}^*\boldsymbol{x} \le \boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\ \forall\, \boldsymbol{x} \in \mathbb{K}^n. \tag{3.1.15}$$

(a) Writing in the $(1, n-1)\times(1, n-1)$*-block form*

$$\boldsymbol{A} = \begin{bmatrix} a & \boldsymbol{b}^* \\ \boldsymbol{b} & \boldsymbol{C} \end{bmatrix} \tag{3.1.16}$$

show that $a \in \mathbb{R}^+$*,* $a > \alpha$ *and the matrix* $\boldsymbol{C} - \boldsymbol{b}\, a^{-1}\boldsymbol{b}^*$ *is in* SPD($\mathbb{K}^{n-1}$) *by proving*

$$\hat{\boldsymbol{x}}^*\boldsymbol{C}\hat{\boldsymbol{x}} \ge \alpha\hat{\boldsymbol{x}}^*\hat{\boldsymbol{x}} + \hat{\boldsymbol{x}}^*\boldsymbol{b}\, a^{-1}\boldsymbol{b}^*\hat{\boldsymbol{x}}\ \forall\, \hat{\boldsymbol{x}} \in \mathbb{K}^{n-1}. \tag{3.1.17}$$

(b) State where in the derivation of Cholesky's algorithm (or the recursive proof of Choleskey's factorisation theorem) this fact is used.

(c) Suppose the matrix $\boldsymbol{A} \in$ SPD($\mathbb{K}^n$) *is written* $(l, m)\times(l, m)$*-blocks*

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{D} & \boldsymbol{B}^* \\ \boldsymbol{B} & \boldsymbol{C} \end{bmatrix}; \tag{3.1.18}$$

and show that the matrix $\boldsymbol{C} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{B}^*$ *is* SPD($\mathbb{K}^m$).

3.1.7. Algorithm (recursive Cholesky).

```
Require: $A \in \mathrm{SPD}(\mathbb{K}^n)$
Ensure: $U \in \mathscr{U}(\mathbb{K}^n)$ such that $U^*U = A$
1: procedure RECURSIVE-CHOLESKY($A$)
2:     $u_1 \leftarrow \left(a_1^1\right)^{-1/2} a_1$        ▷ this includes $u_1^1 \leftarrow \sqrt{a_1^1}$
3:     if $n > 1$ then
4:         $U_{[2\dots n]}^{[2\dots n]} \leftarrow$ RECURSIVE-CHOLESKY($A_{[2\dots n]}^{[2\dots n]} - A_{[2\dots n]}^{[1]} {a_1^1}^{-1} A_{[1]}^{[2\dots n]}$)        ▷ in practice
   operate only on the upper half of $U$ (and its diagonal)
5:     end if
6:     return $U$
7: end procedure
```

3.1.8. Algorithm (an iterative Cholesky). As usual, iterative (for loop) implementations are more efficient for most computing languages, so we give a possible one here

```
Require: SPD matrix $A$
Ensure: upper triangular matrix $U$ such that $U^*U = A$
  for $j = 1, \dots, n$ do
      $u_j^j = \sqrt{a_j^j - \sum_{1\le k\le j-1} \left(u_k^j\right)^2}$        ▷ $\sum_\varnothing = 0$ by definition
      for $i = j+1, \dots, n$ do
          $u_j^i = (a_i^j - \sum_{k=1}^{j-1} u_k^i u_k^j)/u_j^j$
      end for
  end for
```

3.1.9. Remark (Cholesky vs UHU). Some people call Cholesky's factorisation a *UHU factorisation* because some authors use the notation U^H to indicate the adjoint (the H stands for "Hermite") resulting in $A = U^H U$ whence "UHU". We prefer sticking (pun unintended) to H as a variable an not overload it. We also attribute the algorithm to its inventor.

3.1.10. Remark (real Cholesky and UTU vs UHU). If the matrix A belongs to $\mathrm{SPD}(\mathbb{R}^n)$, then Cholesky's algorithm produces a real matrix $U \in \mathscr{U}(\mathbb{R}^n)$. You should convince yourself of this by going through the algorithm and checking that $\mathbb{R}^{n\times n}$ is closed under Cholesky's algorithm.
Since this means that $U^* = U^\intercal$. In this case we often use the latter notation to emphasise the "reality". Note also, as a warning, that some authors use UTU to indicate a unitary-triangular-unitary (Schur-like) factorisation, so it's definitely better to use "Cholesky".

3.1.11. Remark (complex Cholesky vs real Cholesky). What we have presented is known as *complex Cholesky*, to distinguish it from *real Cholesky*. From a mathematical view-point the distinction is a bit silly, as real Cholesky is a subcase of complex Cholesky. From a computational point of view the distinction makes more sense.

3.1.12. Remark (real Cholesky vs LLT). In many textbooks Cholesky computes, instead of U, the lower triangular matrix U^* which is called L in that case and the factorisation looks like $A = LL^* = LL^\intercal$, which prompts the name *LLT factorisation*. We eschew this approach lest the reader think that L is normalised (in fact it is not and we don't even have a notation for the set of non-normalised lower triangular matrices, nor wish to introduce one). Also the "$*$" or "$\intercal$" in the middle is more esthetically pleasing as it produces a more symmetric (pun intended) expression.

3.2. QR factorisation

Let $\boldsymbol{A} \in \mathbb{K}^{n\times n}$. The Schur factorisation described in §2.3, 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.

3.2.1. Theorem. *Any matrix $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ admits a QR factorisation, whereby for some $\boldsymbol{Q} \in \mathrm{U}(n) \cap \mathbb{K}^{n\times n}$ and $\boldsymbol{R} \in \mathscr{U}(\mathbb{K}^n)$ such that*

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}. \tag{3.2.1}$$

Proof Proceed by induction on n. The base case, $n = 1$ is trivial with $\boldsymbol{Q} = 1$ and $\boldsymbol{R} = \boldsymbol{A}$. 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

$$\boldsymbol{A} =: \begin{bmatrix} a & \boldsymbol{b}^\intercal \\ \boldsymbol{c} & \boldsymbol{D} \end{bmatrix} \text{ and let } l := \left\| \begin{bmatrix} a \\ \boldsymbol{c} \end{bmatrix} \right\| = \sqrt{\overline{a}\, a + \boldsymbol{c}^* \boldsymbol{c}}. \tag{3.2.2}$$

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

$$\mathrm{H}(\boldsymbol{a}^1 - l\mathbf{e}^1) =: \boldsymbol{H}. \tag{3.2.3}$$

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

$$\boldsymbol{H}\boldsymbol{A} = \begin{bmatrix} l & \hat{\boldsymbol{b}}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{A}} \end{bmatrix}, \tag{3.2.4}$$

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

$$\begin{bmatrix} \hat{\boldsymbol{b}}^\intercal \\ \hat{\boldsymbol{A}} \end{bmatrix} := \boldsymbol{H}\,[\boldsymbol{A}]^{[2\dots n]} \text{ and } [\boldsymbol{A}]^{[2\dots n]} = \begin{bmatrix} a^n_1 & \dots & a^n_2 \\ \vdots & \ddots & \vdots \\ a^n_1 & \dots & a^n_2 \end{bmatrix}. \tag{3.2.5}$$

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

$$\hat{\boldsymbol{A}} = \hat{\boldsymbol{Q}}^* \hat{\boldsymbol{R}}. \tag{3.2.6}$$

It follows that

$$\boldsymbol{H}\boldsymbol{A} = \begin{bmatrix} l & \hat{\boldsymbol{b}}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{A}} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \mathbf{0}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{Q}}^* \end{bmatrix}}_{=:\, \widetilde{\boldsymbol{Q}}} \underbrace{\begin{bmatrix} l & \hat{\boldsymbol{b}}^\intercal \\ \mathbf{0} & \hat{\boldsymbol{R}} \end{bmatrix}}_{=:\, \boldsymbol{R}} = \widetilde{\boldsymbol{Q}}\boldsymbol{R}. \tag{3.2.7}$$

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

$$\boldsymbol{A} = \boldsymbol{H}\widetilde{\boldsymbol{Q}}\boldsymbol{R} = \boldsymbol{Q}\boldsymbol{R}, \tag{3.2.8}$$

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

3.2.2. Algorithm (recursive QR). In practice it is better to store the normalised vectors $\boldsymbol{w}^i$, $i = 1,\dots,n-1$ instead of the matrix $\boldsymbol{Q}$ as this saves on the computations and the memory. Indeed, the matrix $\boldsymbol{W}$ whose columns are the $\boldsymbol{w}^i$ is an upper triangular one while $\boldsymbol{Q}$ may be full. When the matrix $\boldsymbol{Q}$ is needed, for example to multiply another matrix $\boldsymbol{B}$, then it is enough to *back-multiply* $\boldsymbol{B}$ as follows:

$$\mathbf{Q}(\boldsymbol{W})\boldsymbol{B} \text{ where } \mathbf{Q}(\boldsymbol{W}) := \mathbf{H}(\boldsymbol{w}^1)\cdots\mathbf{H}(\boldsymbol{w}^{n-1}) = \left(\mathbf{I} - 2\frac{\boldsymbol{w}^1\boldsymbol{w}^{1*}}{\boldsymbol{w}^{1*}\boldsymbol{w}^1}\right)\cdots\left(\mathbf{I} - 2\frac{\boldsymbol{w}^{n-1}\boldsymbol{w}^{n-1*}}{\boldsymbol{w}^{n-1*}\boldsymbol{w}^{n-1}}\right). \tag{3.2.9}$$

This operation is more economic than the product $\boldsymbol{Q}\boldsymbol{B}$, which involves n^3 operations, while the product $\mathbf{H}(\boldsymbol{w}^k)\boldsymbol{B}$ in fact involves only the lowest $n-k$ rows of $\boldsymbol{B}$ and can be obtained by multiplying the (short) row vector $\hat{\boldsymbol{w}}^{k*}$ with the block $[\boldsymbol{B}]_{[k\dots n]}$, hence $2k^2n$ operations, which by summing over $k = 1,\dots,n-1$ we obtain a reduction of the operations by 3. We can compress storage even more (at the cost of increasing operations a bit) by taking the $\boldsymbol{w}^k$'s such that $w^k_k = 1$ and having to compute the denominators $\boldsymbol{w}^{k*}\boldsymbol{w}^k$.

Require: $\boldsymbol{A} \in \mathbb{K}^{n\times n}$
Ensure: $\boldsymbol{R} \in \mathscr{U}(\mathbb{K}^n)$ such that $\mathbf{Q}(\boldsymbol{W})\boldsymbol{R} = \boldsymbol{A}$

```
1: procedure RECURSIVE-QR($\boldsymbol{A}$)
2:     $\boldsymbol{w}^1 \leftarrow \boldsymbol{a}^1 - |\boldsymbol{a}^1|\mathbf{e}^1$        ▷ note that $w^1_1 = 0 \iff a^1_1 > 0$ and $a^j_1 = 0 \quad \forall j = 2,\dots,n$
3:     $\boldsymbol{w}^1 \leftarrow \boldsymbol{w}^1 / w^1_1$        ▷ excluding the trivial case when $w^1_1 = 0$
4:     $\boldsymbol{r}_1 \leftarrow [\mathbf{H}(\boldsymbol{w}^1)]_1 \boldsymbol{A}$        ▷ only first row of $\mathbf{H}(\boldsymbol{w}^1)$ is neededed here
5:     if $n > 1$ then
6:         $(\boldsymbol{W}^{[2\dots n]}_{[2\dots n]}, \boldsymbol{R}^{[2\dots n]}_{[2\dots n]}) \leftarrow$RECURSIVE-QR($[\mathbf{H}(\boldsymbol{w}^1)]_{[2\dots n]}[\boldsymbol{A}]^{[2\dots n]}$)        ▷ in practice one may use the upper triangle with diagonal for $\boldsymbol{R}$ and the lower triangle without diagonal for $\boldsymbol{W}$
7:     end if
8:     return $(\boldsymbol{W}, \boldsymbol{R})$
9: end procedure
```

3.2.3. Problem (a recursive QR code). *Write a code that provides a recursive implementation of QR based on Householder matrices. Note that is is not necessary to explicitly compute* $\boldsymbol{Q}$*, it is enough to have a function (or a method) that can evaluate*

$$\mathbf{H}(\boldsymbol{w}^1)\dots\mathbf{H}(\boldsymbol{w}^{n-1})\boldsymbol{X} \tag{3.2.10}$$

for a given matrix $\boldsymbol{X}$ *and lower triangular matrix* $\boldsymbol{W}$ *storing the Householder vectors.*

3.2.4. Problem (iterative QR code). *Derive and implement an iterative version of the QR algorithm, using Householder matrices.*

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

Exercises and problems on direct factorisation methods

Exercise 3.1 (Cholesky factorisation)**.** Compute (if any) a real Cholesky factorisation $\boldsymbol{U}^\intercal\boldsymbol{U}$, with $\boldsymbol{U}\in\mathscr{U}(\mathbb{R}^n)$ of the following symmetric matrix:

$$\boldsymbol{A}=\begin{bmatrix}1&2&3\\2&5&8\\3&8&14\end{bmatrix}. \tag{P3.1.1}$$

$\boldsymbol{A}$ is positive definite; use the fact that it has a real Cholesky factorisation to prove this.

Problem 3.2. Consider $\boldsymbol{A}\in\mathrm{SPD}(\mathbb{K}^n)$ with coercivity $\alpha>0$:

$$\alpha\boldsymbol{x}^*\boldsymbol{x}\le\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\;\forall\,\boldsymbol{x}\in\mathbb{K}^n. \tag{P3.2.1}$$

(a) Writing in the $(1,n-1)\times(1,n-1)$-block form

$$\boldsymbol{A}=\begin{bmatrix}a&\boldsymbol{b}^*\\\boldsymbol{b}&\boldsymbol{C}\end{bmatrix} \tag{P3.2.2}$$

show that $a\in\mathbb{R}^+$, $a>\alpha$ and the matrix $\boldsymbol{C}-\boldsymbol{b}a^{-1}\boldsymbol{b}^*$ is in $\mathrm{SPD}(\mathbb{K}^{n-1})$ by proving

$$\hat{\boldsymbol{x}}^*\boldsymbol{C}\hat{\boldsymbol{x}}\ge\alpha\hat{\boldsymbol{x}}^*\hat{\boldsymbol{x}}+\hat{\boldsymbol{x}}^*\boldsymbol{b}a^{-1}\boldsymbol{b}^*\hat{\boldsymbol{x}}\;\forall\,\hat{\boldsymbol{x}}\in\mathbb{K}^{n-1}. \tag{P3.2.3}$$

(b) State where in the derivation of Cholesky's algorithm (or the recursive proof of Choleskey's factorisation theorem) this fact is used.

(c) Suppose the matrix $\boldsymbol{A}\in\mathrm{SPD}(\mathbb{K}^n)$ is written $(l,m)\times(l,m)$-blocks

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{D}&\boldsymbol{B}^*\\\boldsymbol{B}&\boldsymbol{C}\end{bmatrix}; \tag{P3.2.4}$$

and show that the matrix $\boldsymbol{C}-\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{B}^*$ is $\mathrm{SPD}(\mathbb{K}^m)$.

Exercise 3.3 (QR factorisation)**.** Compute the QR factorisation of the following matrix:

$$\boldsymbol{A}=\begin{bmatrix}1&0&3\\2&-6&3\\-2&3&-3\end{bmatrix} \tag{P3.3.1}$$

by computing the matrix $\boldsymbol{R}$ and two Householder matrices $\boldsymbol{H}(1)$ and $\boldsymbol{H}(2)$ such that:

$$\boldsymbol{A}=\boldsymbol{H}(1)\boldsymbol{H}(2)\boldsymbol{R}. \tag{P3.3.2}$$

There is *no need to compute* explicitly the $\boldsymbol{H}(i)$, nor their product, as long as you state the vectors $\boldsymbol{w}^i$ such that $\boldsymbol{H}(i)=\mathbf{H}(\boldsymbol{w}^i)$.
Show all details of your calculation and verify that what you found is indeed a QR factorisation of $\boldsymbol{A}$.

Exercise 3.4 (QR factorisation)**.** Compute the QR factorisation of the following matrix with pen, paper and pocket-calculator

$$\boldsymbol{A}=\begin{bmatrix}2&5&3\\4&4&-3\\-4&2&3\end{bmatrix}. \tag{P3.4.1}$$

Show all details of your calculations.

Problem 3.5 (a recursive QR code)**.** Write a code that provides a recursive implementation of QR based on Householder matrices. Note that is is not necessary to explicitly compute $\boldsymbol{Q}$, it is enough to have a function (or a method) that can evaluate

$$\mathbf{H}(\boldsymbol{w}^1)\ldots\mathbf{H}(\boldsymbol{w}^{n-1})\boldsymbol{X} \tag{P3.5.1}$$

for a given matrix $\boldsymbol{X}$ and lower triangular matrix $\boldsymbol{W}$ storing the Householder vectors.

Problem 3.6 (iterative QR code)**.** Derive and implement an iterative version of the QR algorithm, using Householder matrices.

Exercise 3.7. Compute the *Q-less QR factorisation* of the matrix

$$\boldsymbol{A} = \begin{bmatrix} 2/3 & 1 & 2 & 1 \\ 4/3 & 2 & 2 & -3 \\ 0 & 2 & 1 & 2 \\ -4/3 & -2 & -3 & -1 \end{bmatrix} \tag{P3.7.1}$$

by computing the upper triangular matrix $\boldsymbol{R} \in \mathscr{U}(\mathbb{K}^n)$ and a *normalised lower triangular matrix* $\boldsymbol{W} \in \mathscr{L}_1(\mathbb{K}^n)$ such that

$$\boldsymbol{A} = \mathbf{Q}(\boldsymbol{W})\boldsymbol{R}, \text{ where } \mathbf{Q}(\boldsymbol{W}) = \left(\mathbf{I} - 2\frac{\boldsymbol{w}^1\boldsymbol{w}^{1*}}{\boldsymbol{w}^{1*}\boldsymbol{w}^1}\right)\cdots\left(\mathbf{I} - 2\frac{\boldsymbol{w}^3\boldsymbol{w}^{3*}}{\boldsymbol{w}^{3*}\boldsymbol{w}^3}\right). \tag{P3.7.2}$$

Show all details of your calculation, but do not write the matrix $\boldsymbol{Q}$.

CHAPTER 4

Least squares polynomial approximation

Approximation Theory is the branch of Analysis and Computational Mathematics that, given a function f and a set of functions $\mathbb{V}$, considers the problem of finding an element v that is as "close" as possible to f.
Polynomial interpolation, as discussed in Chapter C, seems very attractive as an approximation procedure where $\mathbb{V} = \mathbb{P}^N$ for some $N \in \mathbb{N}_0$. Afterall, if the aim is to approximate a function f, of which we know the values $f(x_j)$ at the nodes x_j, $j = 1, \dots, N$, the interpolant polynomial p does the best job possible at the interpolation nodes as it coincides with f thereon. The problem is that when x is *away from interpolation nodes* the interpolation polynomial's value $p(x)$ usually does a pretty bad job (compared to other methods) at approximating the interpolated function's value $f(x)$. A well known example where approximation via interpolation may go wrong is Runge's example (see Exercise C.5).
There are many alternative approximation procedures to interpolation: piecewise approximation, adaptive interpolation, rational approximation, trigonometric (Fourier) series, wavelets, nonlinear approximation, or least-squares polynomial approximation, to cite a few (a nice in-depth exposition can be found in DeVore and Lorentz, 1993). In this chapter we focus on *least-squares polynomial approximation*, (also known as *polynomial fitting*).

4.1. Linear least-squares approximation

In many scientific experiments errors, although minor, occur e.g. measurements are never one hundred percent accurate. By taking many studies and many measurements it is hoped that the small amounts of *noise* can be averaged out. In this subsection we will assume that a set of data is following a trend, we will try to find a best fit function to the data. This is how scientists can check their assumptions e.g. a scientist believes cancer cells grow at a quadratic rate with respect to time, they may take many measurements over time and then try to find a *best fit* quadratic polynomial to the data. The goal is thus now not to *interpolate exactly* the data, but, more generally, to *approximate* it.

4.1.1. How do we define a best fit? We want to fit a polynomial of degree N to a set of $M \geq N+1$ data points, $x_1, x_2, \dots, x_M$ with at least $N+1$ of these points distinct (why do we make this assumption?). Let $f_1, f_2, \dots, f_M$ be approximate measurements of a function f at the set of data points. We wish to find a polynomial

$$p_N(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_N x^N, \tag{4.1.1}$$

such that

$$\sum_{i=1}^{M} \left(p_N(x_i) - f_i\right)^2, \tag{4.1.2}$$

is minimized over all polynomials of degree N. This will define our *least squares best fit.*
Clearly this minimisation problem is a question of how to find the coefficients of the polynomial. Define for $\boldsymbol{a} = (a_0, \dots, a_N) \in \mathbb{R}^{N+1}$

$$F(\boldsymbol{a}) := \sum_{i=1}^{M} \left(\sum_{j=0}^{N} a_j (x_i)^j - f_i \right)^2 = \sum_{i=1}^{M} \left(a_0 + a_1 x_i + a_2 (x_i)^2 + \cdots + a_N (x_i)^N - f_i \right)^2. \quad (4.1.3)$$

Hence F is just a quadratic function on $\mathbb{R}^{N+1}$ which we want to minimise over $\boldsymbol{a} \in \mathbb{R}^{N+1}$: i.e., find a vector $\hat{\boldsymbol{a}} \in \mathbb{R}^{N+1}$ such that

$$F(\hat{\boldsymbol{a}}) = \inf_{\boldsymbol{a} \in \mathbb{R}^{N+1}} F(\boldsymbol{a}) = \min_{\boldsymbol{a} \in \mathbb{R}^{N+1}} F(\boldsymbol{a}). \quad (4.1.4)$$

(Recall: and inf is a min if it is attained. We are looking for a point $\hat{\boldsymbol{a}}$ on which this occurs.)
A necessary condition for F to be minimised at $\hat{\boldsymbol{a}}$ is that its gradient thereat, $\nabla F(\hat{\boldsymbol{a}})$, be $\mathbf{0}$. This is equivalent to say that all the first partial derivatives of F all be zero. The chain rule yields:

$$\frac{\partial F}{\partial a_n}(\boldsymbol{a}) = 2 \sum_{i=1}^{M} x_i^n \left(a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_N x_i^N - f_i \right) = 0 \quad \text{for each } n = 0, \dots, N. \quad (4.1.5)$$

Hence, for $j = 0, 1, 2, \dots, N$ we have the system of $N + 1$ linear equations

$$\sum_{i=1}^{M} x_i^j \left((a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_N x_i^N) - f_i \right) = 0. \quad (4.1.6)$$

These equations are known as the *normal equations.*
We may also write this system in the form

$$\begin{bmatrix} s_0 & s_1 & s_2 & \cdots & s_N \\ s_1 & s_2 & s_3 & \cdots & s_{N+1} \\ s_2 & & \cdots & \cdots & s_{N+2} \\ \vdots & & & \ddots & \\ s_N & & \cdots & & s_{2N} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \cdot \\ \cdot \\ a_N \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \cdot \\ \cdot \\ b_N \end{bmatrix}, \quad (4.1.7)$$

where $s_j = \sum_{i=1}^{M} x_i^j$ for $j = 0, \dots, 2N$ and $b_j = \sum_{i=1}^{M} x_i^j f_i$ for $j = 0, 1, 2, \dots, N$. This symmetric matrix, that is defined by only $2N + 1$ values, $s_0, s_1, \dots, s_{2N}$, is known as the *Hankel matrix.* Note, the cross diagonals are constant.
We call the procedure just outlined a *linear least squares* fiyt because the approximating function p_N is a linear combination of the monomials $1, x, x^2, \dots, x^N$.

4.1.2. Theorem (Hankel's matrix invertibility). *If at least $N + 1 \le M$ of the $\{x_i\}$, $i = 1, 2, \dots, M$, are distinct, then the associated Hankel matrix*

$$\boldsymbol{H} := \begin{bmatrix} s_0 & s_1 & s_2 & \cdots & s_N \\ s_1 & s_2 & s_3 & \cdots & s_{N+1} \\ s_2 & & \cdots & \cdots & s_{N+2} \\ \vdots & & & \ddots & \\ s_N & & \cdots & & s_{2N} \end{bmatrix} \quad (4.1.8)$$

is invertible.
Proof A matrix $\boldsymbol{H}$ is called positive definite, if

$$\boldsymbol{a}^\intercal \boldsymbol{H}\boldsymbol{a} > 0, \quad \forall\, \boldsymbol{a} \neq \boldsymbol{0}. \tag{4.1.9}$$

If $\boldsymbol{H}$ is positive definite, then obviously H is non-singular. Indeed, if $\boldsymbol{H}$ is positive definite then it follows that $H\boldsymbol{a} \neq 0$ for all $\boldsymbol{a} \neq \boldsymbol{0}$.
Consider the matrix

$$\boldsymbol{E} = \begin{bmatrix} 1 & x_1 & {x_1}^2 & \cdots & {x_1}^N \\ 1 & x_2 & {x_2}^2 & \cdots & {x_2}^N \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & x_M & {x_M}^2 & \cdots & {x_M}^N \end{bmatrix}. \tag{4.1.10}$$

$\boldsymbol{E}$ is a $M \times (N+1)$ matrix and is such that

$$\boldsymbol{H} = \boldsymbol{E}^\intercal \boldsymbol{E}. \tag{4.1.11}$$

Using $\boldsymbol{E}$ we have

$$\boldsymbol{a}^\intercal \boldsymbol{H}\boldsymbol{a} = \boldsymbol{a}^\intercal \boldsymbol{E}^\intercal \boldsymbol{E}\boldsymbol{a} = (\boldsymbol{E}\boldsymbol{a})^\intercal (\boldsymbol{E}\boldsymbol{a}) \geq 0 \quad \forall\, \boldsymbol{a} \in \mathbb{R}^{N+1}. \tag{4.1.12}$$

Using the definition of $\boldsymbol{E}$ we have

$$\boldsymbol{E}\boldsymbol{a} = \boldsymbol{0} \Rightarrow a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_N x_i^N = 0, \text{ for } i = 1, 2, \ldots, M \geq N+1. \tag{4.1.13}$$

Since at least $N+1$ of the x_i are distinct, this implies that the N-th degree polynomial

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_N x^N, \tag{4.1.14}$$

has at least $N+1$ distinct roots. This is only possible if the polynomial is the zero polynomial, i.e. $a_0 = \ldots = a_N = 0$. Thus, $\boldsymbol{a}^\intercal \boldsymbol{H}\boldsymbol{a} = 0$ if and only if $\boldsymbol{a} = \boldsymbol{0}$ and $\boldsymbol{H}$ is non-singular. □

4.1.3. Example. Fit a cubic least squares polynomial to the data $(x_k, \cos x_k)$, where $x_k = k\pi/8$ for $k = 0, 1, 2, \ldots, 8$.
In Octave (or Matlab®) we may write the following (note that $c(k+1) = s_k$ and $N = 3$) – recall that the function `sum` sums up the entries of a vector.
In Figure **??** we show the cosine function (blue) together with the cubic least squares polynomial, which approximates the function very well in $[0, \pi]$.

4.2. General linear least squares

For general linear least squares the fitting function does not have to be a polynomial. All we require is that the function is of the form

$$\sum_{j=0}^{N} a_j \phi_j(x), \tag{4.2.1}$$

where the basis functions $\{\phi_j\}$ are given.
Note choosing $\phi_j(x) = x^j$, gives the polynomials of the last subsection.
Other types of basis functions include

$$\phi_j(x) = \sin(j\pi x). \tag{4.2.2}$$

Printout of file Code/leastSquares1.m

```
%% -*- mode: octave; -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [] = leastSquares1()
%% function [] = leastSquares1()
x = [0:pi/8:pi];
for k = 0:6
  c(k+1) = sum(x.^k);
end
for k = 0:3
  b(k+1) = sum(x.^k.*cos(x));
end
for i = 1:4
  for j = 1:4
    A(i,j) = c(i+j-1);
  end
end
a = A\b';


% polynomial evaluation points.
y = [0:0.01*2*pi:2*pi];

% evaluate the polynomial at the data points
p = zeros(1,101);
for i = 0:3
  p = p + a(i+1)*y.^i;
end

hold on
plot(y,cos(y))
plot(y,p,'k')
plot(x,cos(x),' s')
```

FIGURE 1. A cubic interpolation code

This type of basis functions would be used to model data that is suspected of having a periodic nature, e.g. heart pumping.

$$\phi_j(x) = \exp(\beta_j x). \tag{4.2.3}$$

This type of basis functions would be used to model data that is suspected of having a exponential growth or decay, e.g. nuclear radiation.
Using a general basis, the least squares approximation is given by, find $a_0, a_1, \ldots, a_N$ such that

$$g(a_0, a_1, \ldots, a_N) = \sum_{i=1}^{M}\left[\sum_{k=0}^{N} a_k\phi_k(x_i) - f_i\right]^2, \tag{4.2.4}$$

is minimised.
Again taking partial derivatives with respect to the a_j's and setting them $= 0$, we obtain $\frac{\partial g}{\partial a_j} = 2\sum_{i=1}^{M}\phi_j(x_i)[\sum_{k=0}^{N} a_k\phi_k(x_i) - f_i] = 0$. Hence, we obtain the linear system of

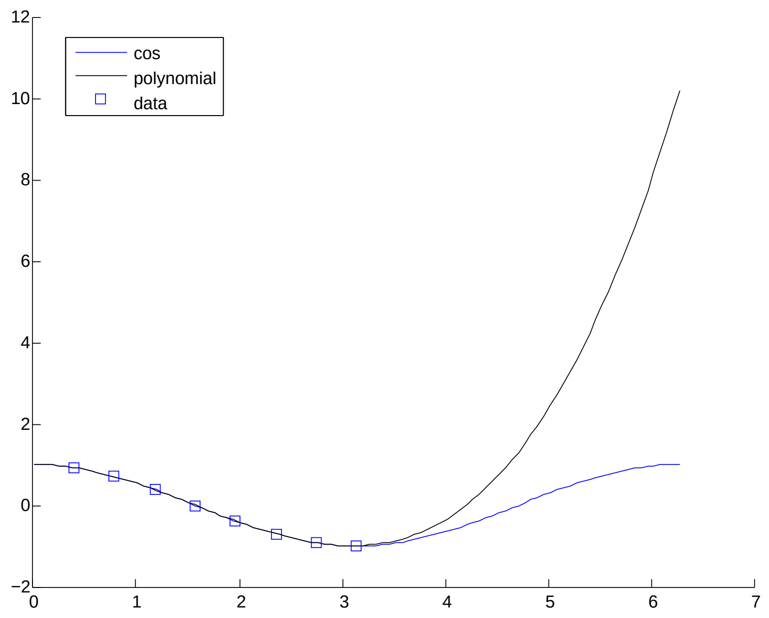


FIGURE 2. The figure shows $\cos x$ (blue) together with the cubic least squares polynomial, which approximates the function very well in $[0,\pi]$.

equations

$$\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & \cdots & s_{0,N} \\ s_{1,0} & s_{1,1} & s_{1,2} & \cdots & s_{1,N} \\ s_{2,0} & & \cdots & \cdots & s_{2,N} \\ \vdots & & & \ddots & \\ s_{N,0} & & \cdots & & s_{N,N} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \cdot \\ \cdot \\ a_N \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \cdot \\ \cdot \\ c_N \end{bmatrix}, \tag{4.2.5}$$

where

$$s_{j,k} = \sum_{i=1}^{M} \phi_j(x_i)\phi_k(x_i) \text{ and } c_j = \sum_{i=1}^{M} \phi_j(x_i) f_i. \tag{4.2.6}$$

This general linear least squares matrix (which is symmetric) is known as a ***G***rammian matrix. We have to require certain conditions to hold on both the ϕ_i's and the x_i's for the matrix to be non-singular. In the next subsection we construct a special basis ϕ_j such that the matrix is even diagonal.

4.3. Orthogonal polynomial based least-squares

Returning to using polynomial least squares, we see from the last subsection that any set of basis functions can be used. We will use a basis that spans all polynomials of degree less than or equal to n. The idea is now to choose the basis such that the Grammian matrix is diagonal.

4.3.1. Orthogonal polynomial bases. We will build up basis polynomials, $q_0, q_1, \ldots, q_N$ of degree $0, 1, 2, \ldots, N$, respectively. Then we will say that the polynomials q_i are *with* *respect to the points* $x_1, \ldots, x_M$ if and only if

$$\sum_{i=1}^{M} q_k(x_i) q_j(x_i) = 0 \text{ if } j \neq k \tag{4.3.1}$$

which is equivalent to

$$\begin{bmatrix} q_k(x_1) & q_k(x_2) & \cdots & q_k(x_M) \end{bmatrix} \begin{bmatrix} q_j(x_1) \\ q_j(x_2) \\ \vdots \\ q_j(x_M) \end{bmatrix} = 0 \text{ if } j \neq k. \tag{4.3.2}$$

The latter reminds us of orthogonality of vectors, whence the name.

4.3.2. Grammian matrices. If we can construct such a set of polynomials, then the Grammian matrix using this basis, reduces to a diagonal matrix and the related least squares system of equations reduces to: find a_k, $k = 0, 1, 2, \ldots, N$ such that

$$\sum_{i=1}^{M} [q_k(x_i)]^2 a_k = \sum_{i=1}^{M} q_k(x_i) f_i, \text{ i.e. } a_k = \frac{\sum_{i=1}^{M} q_k(x_i) f_i}{\sum_{i=1}^{M} [q_k(x_i)]^2}. \tag{4.3.3}$$

Using these a_k, this gives the least squares polynomial

$$q(x) = \sum_{k=0}^{N} a_k q_k(x). \tag{4.3.4}$$

So using orthogonal polynomials makes it a trivial matter to find the least squares polynomial, if we have the orthogonal basis. Hence we must construct the polynomials and this is where the work is now taken.

4.3.3. Algorithm. Given nodes $x_1, x_2, \ldots, x_M$ the following method is used to construct the polynomials q_j. We summarise the method in Method 4.3.4.

(1) $q_0(x) = 1$

(2) $q_1(x) = x - \alpha_1$. We require that

$$0 = \sum_{i=1}^{M} q_0(x_i) q_1(x_i) = \sum_{i=1}^{M} x_i - M\alpha_1, \tag{4.3.5}$$

so that

$$\alpha_1 = \frac{1}{M} \sum_{i=1}^{M} x_i. \tag{4.3.6}$$

Note that

$$\sum_{i=1}^{M} q_1(x_i) = 0. \tag{4.3.7}$$

(3) $q_2(x) = xq_1(x) - \alpha_2 q_1(x) - \beta_1$. We require that

$$\begin{aligned} \sum_{i=1}^{M}[x_i q_1(x_i) - \alpha_2 q_1(x_i) - \beta_1] &= 0, \\ \sum_{i=1}^{M}[x_i q_1(x_i) - \alpha_2 q_1(x_i) - \beta_1] q_1(x_i) &= 0. \end{aligned} \tag{4.3.8}$$

From (4.3.7), these equations become

$$\begin{aligned} \sum_{i=1}^{M} x_i q_1(x_i) - M\beta_1 &= 0, \\ \sum_{i=1}^{M} x_i [q_1(x_i)]^2 - \alpha_2 \sum_{i=1}^{M}[q_1(x_i)]^2 &= 0. \end{aligned} \tag{4.3.9}$$

Hence

$$\begin{aligned} \beta_1 &= \frac{1}{M}\sum_{i=1}^{M} x_i q_1(x_i) \\ \text{and } \alpha_2 &= \frac{1}{\sum_{i=1}^{M}[q_1(x_i)]^2}\sum_{i=1}^{M} x_i q_1(x_i)^2. \end{aligned} \tag{4.3.10}$$

Introducing $\gamma_j = \sum_{i=1}^{M}[q_j(x_i)]^2$, we obtain

$$\begin{aligned} \alpha_2 = & \frac{1}{\gamma_1}\sum_{i=1}^{M} x_i [q_1(x_i)]^2 \\ \beta_1 = & \frac{\gamma_1}{\gamma_0} \end{aligned} \tag{4.3.11}$$

since $\gamma_0 = M$ and, by definition of $q_1(x)$,

$$\gamma_1 = \sum_{i=1}^{M}[q_1(x_i)]^2 = \sum_{i=1}^{M}(x_i - \alpha_1)q_1(x_i) = \sum_{i=1}^{M} x_i q_1(x_i) - \alpha_1 \underbrace{\sum_{i=1}^{M} q_1(x_i)}_{=0}. \tag{4.3.12}$$

(4) For $j \geq 1$, assuming we have defined the orthogonal set $q_0, q_1, \ldots, q_j$, we define the three term recurrence relationship (similar to case 3. where $j = 1$):

$$q_{j+1}(x) \;:=\; xq_j(x) - \alpha_{j+1} q_j(x) - \beta_j q_{j-1}(x). \tag{4.3.13}$$

For $k < j - 1$, we have

$$\sum_{i=1}^{M} q_{j+1}(x_i) q_k(x_i) = \sum_{i=1}^{M} x_i q_j(x_i) q_k(x_i) - \alpha_{j+1}\sum_{i=1}^{M} q_j(x_i) q_k(x_i) \tag{4.3.14}$$

$$-\beta_j \sum_{i=1}^{M} q_{j-1}(x_i) q_k(x_i). \tag{4.3.15}$$

Clearly the last two terms of this sum are zero. The first term involves the product of q_j with a polynomial, xq_k, of degree strictly less than j. Hence, this polynomial is a sum of all the orthogonal polynomials $q_0, q_1, \ldots, q_{j-1}$ and so the sum

$$\sum_{i=1}^{M} x_i q_j(x_i) q_k(x_i) = 0. \tag{4.3.16}$$

We are left with finding the values α_{j+1} and β_j such that

$$\begin{aligned} \sum_{i=1}^{M} q_{j+1}(x_i) q_j(x_i) = &\quad 0 \\ \text{and } \sum_{i=1}^{M} q_{j+1}(x_i) q_{j-1}(x_i) = &\quad 0. \end{aligned} \tag{4.3.17}$$

Replacing $q_{j+1}(x_i)$ by using the recurrence relationship (4.3.13) we obtain

$$\begin{aligned} \alpha_{j+1} = &\quad \frac{1}{\gamma_j} \sum_{i=1}^{M} x_i [q_j(x_i)]^2, \\ \text{and } \beta_j = &\quad \frac{1}{\gamma_{j-1}} \sum_{i=1}^{M} x_i q_j(x_i) q_{j-1}(x_i), \end{aligned} \tag{4.3.18}$$

where, as defined above, $\gamma_j = \sum_{i=1}^{M} [q_j(x_i)]^2$.

Finally, using the recurrence relationship we have a formula for $x_i q_{j-1}(x_i)$, so for $j \geq 2$

$$\begin{aligned} \beta_j = &\quad \frac{1}{\gamma_{j-1}} \sum_{i=1}^{M} x_i q_j(x_i) q_{j-1}(x_i) \\ = &\quad \frac{1}{\gamma_{j-1}} \sum_{i=1}^{M} q_j(x_i)[q_j(x_i) + \alpha_j q_{j-1}(x_i) + \beta_{j-1} q_{j-2}(x_i)] \\ \beta_j = &\quad \frac{\gamma_j}{\gamma_{j-1}}. \end{aligned} \tag{4.3.19}$$

The advantage of using this method is now not so obvious, since many calculation will have to be performed. But unlike using the full Hankel matrix we don't have to solve a large system of equations. Finally, if we want to use the next degree of data fitting we need only build upon the basis functions we already have.

4.3.4. Least square method. A numerical algorithm for finding orthogonal polynomials $q_j(x)$ and use them to find the linear least square fit $q(x)$ over polynomials of degree at most N with data points x_i, f_i, $i = 1, \ldots, M$ looks like this:

(1) Set

$$\begin{aligned} q_0(x) &= 1 \\ \gamma_0 &= M \\ q_1(x) &= x - \frac{1}{\gamma_0}\sum_{i=1}^{M} x_i \\ \gamma_1 &= \sum_{i=1}^{M} [q_1(x_i)]^2 \end{aligned} \tag{4.3.20}$$

(2) Compute for $j = 1$ to $N - 1$

$$\begin{aligned} \alpha_{j+1} &= \frac{1}{\gamma_j}\sum_{i=1}^{M} x_i [q_j(x_i)]^2 \\ \beta_j &= \frac{\gamma_j}{\gamma_{j-1}} \\ q_{j+1}(x) &= x q_j(x) - \alpha_{j+1} q_j(x) - \beta_j q_{j-1}(x) \\ \gamma_{j+1} &= \sum_{i=1}^{M} [q_{j+1}(x_i)]^2 \end{aligned} \tag{4.3.21}$$

(3) Compute the coefficients $a_0, a_1, \ldots, a_N$ through

$$a_k = \frac{1}{\gamma_k}\sum_{i=1}^{M} q_k(x_i) f_i \tag{4.3.22}$$

to obtain the least squares approximation

$$q(x) = a_0 q_0(x) + a_1 q_1(x) + \cdots + a_N q_N(x). \tag{4.3.23}$$

4.3.5. Example (least square in action). We are given the data:

$$x_1 = 0, x_2 = 1, x_3 = 2, \tag{4.3.24}$$

$$f_1 = 1, f_2 = 3, f_3 = 2. \tag{4.3.25}$$

Construct the first three orthogonal polynomials and the linear least squares fit over polynomials of degree at most one.
This means $M = 3$ and $N = 1$.

(1) We calculate

$$\begin{aligned} q_0(x) &= 1 \\ \gamma_0 &= 3 \\ \sum_{i=1}^{3} x_i &= 3 \\ q_1(x) &= x - 1 \\ \gamma_1 &= \sum_{i=1}^{3} [q_1(x_i)]^2 = 2. \end{aligned} \tag{4.3.26}$$

(2) Do nothing (see later why).

(3) We now need only find the coefficients a_0 and a_1 given by:

$$\begin{aligned} a_0 = &\ \frac{1}{\gamma_0}[q_0(0)\cdot 1 + q_0(1)\cdot 3 + q_0(2)\cdot 2] = 2 \\ a_1 = &\ \frac{1}{\gamma_1}[q_1(0)\cdot 1 + q_1(1)\cdot 3 + q_1(2)\cdot 2] = \frac{1}{2} \end{aligned} \tag{4.3.27}$$

This gives the least square approximation,

$$Q(x) = 2 + \frac{1}{2}(x-1) = \frac{1}{2}(x+3). \tag{4.3.28}$$

If we choose $N = 2$, then we obtain the interpolation polynomial. We build on the previous calculation and go one step further:

(1) As before.

(2) Instead of doing nothing

$$\begin{aligned} \sum_{i=1}^{3} x_i[q_1(x_i)]^2 = &\ 2, \\ \alpha_2 = &\ \frac{2}{\gamma_1} = 1 \\ \beta_1 = &\ \frac{\gamma_1}{\gamma_0} = \frac{2}{3} \\ q_2(x) = &\ x q_1(x) - \alpha_2 q_1(x) - \beta_1 q_0(x) \\ = &\ x^2 - x - (x-1) - \frac{2}{3} \\ = &\ x^2 - 2x + \frac{1}{3} \\ \gamma_2 = &\ \frac{1}{9} + \frac{4}{9} + \frac{1}{9} = \frac{2}{3} \end{aligned} \tag{4.3.29}$$

(3) Hence,

$$a_2 = \frac{1}{\gamma_2}[q_2(0)\cdot 1 + q_2(1)\cdot 3 + q_2(2)\cdot 2] = \frac{3}{2}\left[\frac{1}{3} - 2 + \frac{2}{3}\right] = -\frac{3}{2}. \tag{4.3.30}$$

This gives the interpolation polynomial

$$Q(x) = 2 + \frac{1}{2}(x-1) - \frac{3}{2}\left(x^2 - 2x + \frac{1}{3}\right) = -\frac{3}{2}x^2 + \frac{7}{2}x + 1. \tag{4.3.31}$$

We go back to the linear least squares fit over polynomial of degree at most 1. Here, the result was $Q(x) = \frac{1}{2}(x+3)$. If we calculate this polynomial using the linear least squares from §4.1, then we have to solve the system

$$\left(\begin{array}{cc|c} 3 & 3 & 6 \\ 3 & 5 & 7 \end{array}\right) \longrightarrow \left(\begin{array}{cc|c} 3 & 3 & 6 \\ 0 & 2 & 1 \end{array}\right) \tag{4.3.32}$$

which has the solution $a_1 = \frac{1}{2}$ and $a_0 = \frac{3}{2}$. This seems to be a different solution than above because the coefficients are different, but note that these are coefficients of a different basis of polynomials. The least square polynomial is given by

$$a_0 \cdot 1 + a_1 \cdot x = \frac{3}{2} + \frac{1}{2}x \tag{4.3.33}$$

and is indeed the same polynomial as above.

Exercises and problems on least squares

Exercise 4.1. Calculate the constant and linear least square fit to the data

$$f(-1)=0, \qquad f(0)=0, \qquad f(1)=1, \qquad f(2)=0. \tag{P4.1.1}$$

Evaluate the error term $\sum_{i=1}^{M}(p_N(x_i)-f_i)^2$ for both fits.

Exercise 4.2. Calculate the linear and quadratic least square fit to the data

$$f(0)=0, \qquad f(0)=1, \qquad f(1)=1, \qquad f(2)=0, \qquad f(-1)=1 \tag{P4.2.1}$$

Plot the least square fits and the data points in one figure.

Exercise 4.3 (coding the least squares)**.** (1) Write a MATLAB program that takes in the vector x of data points and the vector f of data values, calculates the approximation $g(x)=\sum_{j=1}^{N} a_j\varphi_j(x)$ using the basis functions $\varphi_j(x)=\sin(j\pi x)$, $j=1,2,\ldots,N$ and plots the approximating function from x_0 to x_{end}. It should start like this:

```
function [p,a,y] = least(x,f,x0,xend,N)

% Calculates the least square approximation
% g(x)=sum_{j=1}^N a_j phi_j(x) of the data g(x_i)=f_i
% where phi_j(x)=sin(j pi x)
% It also plots g(x) from  x0 to xend
```

Hint: Use the commands in Example 3.5.17 as a start.

(2) Use the MATLAB program in (a) to approximate the following data points using the basis $\varphi_j(x) = \sin(j\pi x)$, $j = 1,2,\dots,N$, for $N = 1$, $N = 2$, $N = 3$, $N = 4$ and $N = 5$.

$$f(0.2) = 0.87, f(0.5) = 1.2, f(0.8) = 0, f(1.2) = 0,$$
$$f(1.7) = -1.2, f(2.3) = 1.15, f(2.9) = -0.1,$$
$$f(3.4) = -0.8, f(4) = 0, f(5.5) = -1.25$$

Plot the corresponding approximations together with the data points in the interval $[0,6]$. From which N on does the approximation not become visibly better?

Exercise 4.4 (least squares and orthogonal polynomials). (1) Construct orthogonal polynomials for the data points $x_1 = -1$, $x_2 = 0$, $x_3 = 2$, $x_4 = 3$.

(2) Use the orthogonal polynomials to find the quadratic least square fit to the data $f_1 = f_2 = f_3 = 1$, $f_4 = -1$.

(3) Use the orthogonal polynomials to find the quadratic least square fit to the data $f_1 = f_2 = f_3 = 1$, $f_4 = 0$.

Exercise 4.5 (least squares's and uniqueness). Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, with $m > n$, and assume that $\boldsymbol{A}$ has full rank, that is, $\operatorname{rank}\boldsymbol{A} = n$. Let $\boldsymbol{b} \in \mathbb{R}^m$. Consider the following quadratic $f : \mathbb{R}^n \to \mathbb{R}$ defined by

$$f(\boldsymbol{x}) = |\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}|^2, \qquad \boldsymbol{x} \in \mathbb{R}^n, \tag{P4.5.1}$$

where $|\cdot|$ denotes the usual Euclidean (ℓ^2) norm in $\mathbb{R}^n$. Show that there exists a unique vector $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ that minimizes the functional f and that this vector $\hat{\boldsymbol{x}}$ is the unique solution of the *normal equations*

$$\boldsymbol{A}^\intercal \boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{A}^\intercal \boldsymbol{b}. \tag{P4.5.2}$$

Exercise 4.6 (lease squares and SVD). Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $m \geq n$ be given. Assume that $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ with orthogonal matrices $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$, and a diagonal matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ that has only non-negative diagonal entries σ_i, $i = 1,2,\dots,n$, ordered in decreasing order, that is,

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_n \\ 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{bmatrix} \quad \text{where} \quad \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0. \tag{P4.6.1}$$

Use the decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ to show that there exists a minimiser $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ to the cost function $\boldsymbol{x} \mapsto |\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}|_2$, i.e.,

$$|\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}|_2 = \min_{\boldsymbol{x}\in\mathbb{R}^n} |\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}|_2, \tag{P4.6.2}$$

and find a formula for this solution. Show that the solution $\widehat{\boldsymbol{x}}$ is uniquely determined if and only if $\operatorname{rank} A = n$.

CHAPTER 5

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/ybc/comments.html`

5.1. 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.

5.1.1. 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

$$x_k := \frac{1}{2}\left(x_{k-1} + \frac{y}{x_{k-1}}\right), \tag{5.1.1}$$

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

$$\hat{x} = \lim_{k\to\infty} x_k, \tag{5.1.2}$$

then

$$\hat{x} = \frac{1}{2}\left(\hat{x} + \frac{y}{\hat{x}}\right) \tag{5.1.3}$$

which implies

$$\hat{x} = \frac{y}{\hat{x}} \tag{5.1.4}$$

and thus

$$\hat{x}^2 = y, \tag{5.1.5}$$

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

5.1.2. Why should the limit in (5.1.2) exist? We still have to show that the limit in (5.1.2) 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 is it a Cauchy sequence (check B.1 for some reminders on sequences of real numbers and convergence, or, more thoroughly, your Analysis lecture notes).

5.1.3. 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 (5.1.1), 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-the step of the iteration (5.1.1) to be

$$e_k := \hat{x} - x_k \text{ where } \hat{x}^2 = y \text{ and } x_k \text{ given by (5.1.1).} \tag{5.1.6}$$

Noting that *both x_k and x_{k-1} in (5.1.1) may be replaced by $\hat{x}$*, leaving the identity true we have that

$$\begin{aligned} e_k &= \frac{1}{2}\left(\hat{x} + \frac{y}{\hat{x}} - x_{k-1} - \frac{y}{x_{k-1}}\right) \\ \text{(algebra)} \quad &= \frac{1}{2}(\hat{x} - x_{k-1}) + \frac{y}{2}\left(\frac{1}{\hat{x}} - \frac{1}{x_{k-1}}\right) \\ \text{(definition of } e_k\text{, algebra and } y = \hat{x}^2) \quad &= \frac{e_{k-1}}{2}\left(1 - \frac{\hat{x}^2}{\hat{x}\, x_{k-1}}\right) \\ \text{(more algebra)} \quad &= \frac{e_{k-1}(x_{k-1} - \hat{x})}{2x_{k-1}} \\ &= \frac{-(e_{k-1})^2}{2x_{k-1}}. \end{aligned} \tag{5.1.7}$$

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

$$|\epsilon| < 1 \Rightarrow \epsilon^2 < \epsilon, \tag{5.1.8}$$

and assuming for a moment that

$$1/2x_{k-1} \leq 1 \quad \forall\, k \geq 1, \tag{5.1.9}$$

it follows that

$$|e_k| \leq |e_{k-1}|^2. \tag{5.1.10}$$

So if $e_0 < 1$ it follows, by induction, that $e_k < e_0^{2k}$ and thus by (5.1.7) we have

$$\lim_{k \to \infty} e_k = 0. \tag{5.1.11}$$

This can be formalised by the following result.

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

$$x_0 \in I := \begin{cases} (1, y) & \text{if } y > 1, \\ \{1\} & \text{if } y = 1, \\ (y, 1) & \text{if } y < 1. \end{cases} \tag{5.1.12}$$

If x_k, for $k \in \mathbb{N}$, is defined by (5.1.1), then for a C_y, that depends only on y, we have

$$|\hat{x} - x_k| \leq C_[y]|\hat{x} - x_{k-1}|^2 \text{ and } x_k \in I \quad \forall\, k \in \mathbb{N}. \tag{5.1.13}$$

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

$$\hat{x} \in I \tag{5.1.14}$$

(if $\hat{x} \notin I$ it can be seen that $\hat{x}^2 \notin I$). Define e_k as in (5.1.6).

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 (5.1.7) that

$$|e_1| = \frac{(e_0)^2}{2x_0} \leq \frac{1}{2}(e_0)^2. \tag{5.1.15}$$

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

$$x_1 = \frac{1}{2}\left(x_0 + \frac{y}{x_0}\right) > \frac{1}{2}(1+1) = 1 \tag{5.1.16}$$

and

$$x_1 < \frac{1}{2}(y+y) = y, \tag{5.1.17}$$

hence $x_1 \in I$. This proves (5.1.13) for $k = 1$, i.e., the base case of the induction. To prove the inductive step, fix $k \geq 1$ and assume

$$x_k \in I. \tag{5.1.18}$$

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

$$|e_{k+1}| \leq \frac{1}{2}|e_k|^2 < 1 \tag{5.1.19}$$

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. □

5.2. Fixed-point theory

The analysis developed in §5.1 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.

5.2.1. Definition of fixed point.

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

$$x = f(x). \tag{5.2.1}$$

5.2.2. Example.

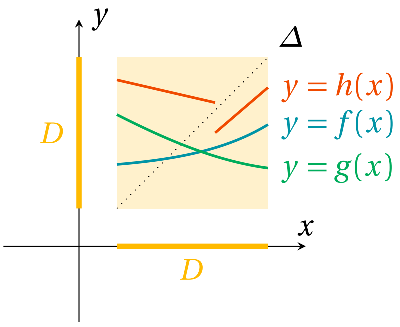
(a) Let $D := \mathbb{R}$ and $f(x) := x^2$. In this case we can check algebraically that 0 and 1 are both fixed points of x.
(b) Let $D := \mathbb{R}$ and $f(x) := \cos x$. This is a bit harder (in fact, impossible) to check algebraically, but one can analytically show (using the intermediate value 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 $x \in D := \mathbb{R}$.

5.2.3. Remark (graphical interpretation in $\mathbb{R}$). Let $f : D \to D$ 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:

$$\Delta := \{(x,x) : x \in D\} \text{ and } \{(x, f(x)) : x \in D\}. \tag{5.2.2}$$

In the picture we plot three examples of functions $f, g, h : D \to D$, 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.

5.2.4. Exercise (contractions are continuous). *Prove that a contraction $g : D \to \mathbb{R}$ must be continuous, i.e., that for any $\epsilon > 0$ we can find at least one $\delta > 0$ such that*

$$x, y \in D \text{ and } \left|x - y\right| \leq \delta \Rightarrow \left|g(x) - g(y)\right| \leq \epsilon. \tag{5.2.3}$$

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

$$\hat{x} = g(\hat{x}), \tag{5.2.4}$$

and there is no other point in D satisfying (5.2.9).
Furthermore if $x_0 \in D$ and for each $k \in \mathbb{N}$ we define the iterates

$$x_k := g(x_{k-1}), \tag{5.2.5}$$

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

$$|x_k - \hat{x}| \leq \kappa |x_{k-1} - \hat{x}| \quad \forall\, k \in \mathbb{N}. \tag{5.2.6}$$

We also have the following apriori and aposteriori error bounds, respectively:

$$|x_k - \hat{x}| \leq |x_0 - \hat{x}| \kappa^k \quad \forall\, k \in \mathbb{N}, \tag{5.2.7}$$

$$|x_k - \hat{x}| \leq \frac{|x_1 - x_0|}{1-\kappa} \kappa^k \quad \forall\, k \in \mathbb{N}. \tag{5.2.8}$$

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

$$\hat{x} = g(\hat{x}), \tag{5.2.9}$$

and there is no other point in D satisfying (5.2.9).
Furthermore if $x_0 \in D$ and for each $k \in \mathbb{N}$ we define the iterates

$$x_k := g(x_{k-1}), \tag{5.2.10}$$

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

$$|x_k - \hat{x}| \le \kappa |x_{k-1} - \hat{x}| \quad \forall\, k \in \mathbb{N} \tag{5.2.11}$$

and the following (apriori) error bound

$$|x_k - \hat{x}| \le |x_0 - \hat{x}|\kappa^k \quad \forall\, k \in \mathbb{N}, \tag{5.2.12}$$

and the following (aposteriori) error estimate

$$|x_k - \hat{x}| \le \frac{|x_1 - x_0|}{1-\kappa}\kappa^k \quad \forall\, k \in \mathbb{N}. \tag{5.2.13}$$

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

$$|x_{k+1} - x_k| = \left|g(x_k) - g(x_{k-1})\right| \le \kappa |x_k - x_{k-1}|. \tag{5.2.14}$$

It follows, by induction on k, that

$$|x_{k+1} - x_k| \le \kappa^k |x_1 - x_0|, \tag{5.2.15}$$

whence for any $k, l \ge 0$

$$\begin{aligned} |x_{k+l} - x_k| &= \left|\sum_{i=0}^{l-1} x_{k+i+1} - x_{k+i}\right| \\ &\le \sum_{i=0}^{l-1} |x_{k+i+1} - x_{k+i}| \le |x_1 - x_0| \sum_{i=0}^{\infty} \kappa^{k+i} \\ &\le \frac{|x_1 - x_0|}{1-\kappa}\kappa^k. \end{aligned} \tag{5.2.16}$$

Since the right hand converges to 0 as $k \to \infty$, it follows that $(x_k)_{k\in\mathbb{N}_0}$ is a Cauchy sequence. By the Cauchy criterion the sequence converges to a limit $\hat{x} := \lim_{k\to\infty} x_k$. Recalling that a contraction such as g must be continuous (see 5.2.4), it follows that

$$\hat{x} = \lim_{k\to\infty} x_k = \lim_{k\to\infty} g(x_{k-1}) = \lim_{k\to\infty} g(x_k) = g(\lim_{k\to\infty} x_k) = g(\hat{x}), \tag{5.2.17}$$

which means that $\hat{x}$ is a fixed point of g.
Suppose now that $\check{x}$ is another fixed point of g, then

$$|\hat{x} - \check{x}| = \left|g(\hat{x}) - g(\check{x})\right| \le \kappa |\hat{x} - \check{x}|. \tag{5.2.18}$$

Whence $(\kappa - 1)|\hat{x} - \check{x}| \ge 0$, 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. □

5.2.6. 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

$$-L\left(y - x\right) \le g(y) - g(x) \le L\left(y - x\right) \quad \forall\, x, y \in D, \tag{5.2.19}$$

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

$$\left|g(y) - g(x)\right| \le L\left|y - x\right| \quad \forall\, x, y \in D. \tag{5.2.20}$$

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

If g is a Lipschitz function then we its *Lipschitz constant on D* is the smallest L that satisfies (5.2.20), in symbols

$$\left|g\right|_{\operatorname{Lip}(D)} := \sup_{D \ni x \neq y \in D} \left|\frac{g(y) - g(x)}{y - x}\right| \tag{5.2.21}$$

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

$$\left|g\right|_{\operatorname{Lip}(C)} = \sup_C \left|g'\right| = \max_C \left|g'\right|. \tag{5.2.22}$$

(The last equality is just a formulaic way of saying that the least upper bound is achieved by some point $\check{x} \in C$: $\left|g'(\check{x})\right| = \sup_C \left|g'\right|$.)
Hint. Use the mean value theorem, in the integral form,

$$g(y) - g(x) = \left(\int_0^1 g'(x + \theta(y - x))\,\mathrm{d}\theta\right)(y - x), \tag{5.2.23}$$

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

$$\left|\int f\right| \leq \int |f| \tag{5.2.24}$$

and monotonicity

$$f \leq h \Rightarrow \int f \leq \int h. \tag{5.2.25}$$

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

$$x \longmapsto \frac{1}{2}\left(x + \frac{y}{x}\right) =: g(x) \tag{5.2.26}$$

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

$$x_k := g(x_{k-1}) \tag{5.2.27}$$

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

$$\hat{x}^2 = y. \tag{5.2.28}$$

5.2.9. Example (applying Banach–Caccioppoli).

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

$$g(x) := \cos x \tag{5.2.29}$$

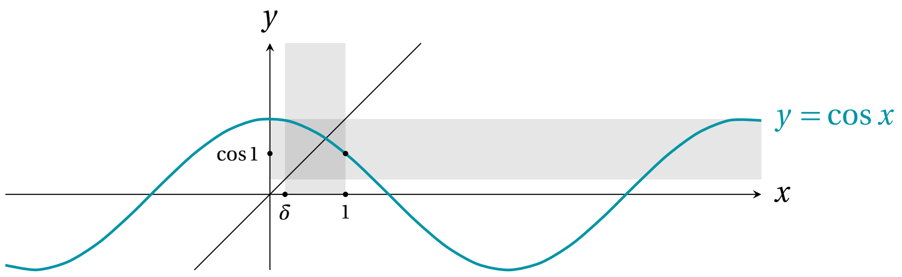
(a) Verify analytically, but without using the Banach–Caccioppoli contraction principle, that this function has a unique fixed point on the real line.
(b) Can uniqueness on the whole real line *be deduced using the Banach–Caccioppoli contraction principle? Explain why, or why not.*

(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

$$h(0) = \cos 0 - 0 = 1 > 0 \text{ and } h(\pi/2) = \cos(\pi/2) - \pi/2 = 1 - \pi/2 < 1 - 3/2 = 1/2 < 0. \tag{5.2.30}$$

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

$$\begin{aligned} x \leq -\pi/2 &\Rightarrow h(x) \geq -1 + \pi/2 > 0 \\ x \geq \pi/2 &\Rightarrow h(x) \leq 1 - \pi/2 < 0. \end{aligned} \tag{5.2.31}$$

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

$$\xi = -\pi/2 + 2k\pi \text{ for some } k \in \mathbb{Z}. \tag{5.2.32}$$

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 $0 > g'(\delta) \geq g'(\delta) \geq g'(1) > -1$. 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.

5.3. Fixed-point theory in metric spaces

The theory developed in §5.2 can be extended to general metric spaces, almost verbatim, following minor modification.

5.3.1. Definition of metric space. A *metric space* is a pair $(\mathscr{X}, d)$, where $\mathscr{X}$ is a set (often referred to as metric space as well) and $d : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is a function satisfying,

$$d(x,y) \geq 0 \quad \forall\, x,y \in \mathscr{X} \qquad (\text{positivity}), \tag{5.3.1}$$

$$d(x,y) = 0 \Leftrightarrow x = y \qquad (\text{non-degeneracy}), \tag{5.3.2}$$

$$d(x,y) = d(y,x) \quad \forall\, x,y \in \mathscr{X} \qquad (\text{symmetry}) \tag{5.3.3}$$

$$d(x,y) \leq d(x,z) + d(z,y) \quad \forall\, x,y,z \in \mathscr{X} \qquad (\text{triangle inequality}). \tag{5.3.4}$$

5.3.2. Definition of convergent and Cauchy sequences. Let $(\mathscr{X}, d)$ be a metric space, $x \in \mathscr{X}$ and $(x_n)_{n\in\mathbb{N}}$ be an $\mathscr{X}$-valued sequence. We say that $(x_n)_{n\in\mathbb{N}}$ converges to x if and only if

$$\forall\, \epsilon > 0 : \exists\, N \in \mathbb{N} : n \geq N \Rightarrow d(x, x_n) < \epsilon. \tag{5.3.5}$$

$(x_n)_{n\in}$ is called a *Cauchy sequence* if and only if

$$\forall\, \epsilon > 0 : \exists\, N \in \mathbb{N} \tag{5.3.6}$$

5.3.3. Exercise (convergence characterisation). *Prove that a sequence $(x_n)_{n\in\mathbb{N}}$ in a metric space $(\mathscr{X}, d)$ converges to $x \in \mathscr{X}$ if and only if the sequence of errors $(d(x, x_n))_{n\in\mathbb{N}}$ converges in $\mathbb{R}$.*

5.3.4. Exercise (Cauchy criterion). *Prove that if a sequence $(x_n)_{n\in\mathbb{N}}$ in a metric space $(\mathscr{X}, d)$ converges then it is a Cauchy sequence.*
Show, with a countrerxample, that the converse is not always true.

5.3.5. Definition of complete metric space. A metric space $(\mathscr{X}, d)$ is called *complete* if and only if each Cauchy sequence therein is also convergent.

5.3.7. Definition of Lipschitz continuity. A function $f : \mathscr{X} \to \mathscr{Y}$, where $(\mathscr{X}, d)$ and $(\mathscr{Y}, e)$ are metric spaces is called Lipschitz continuous if and only if

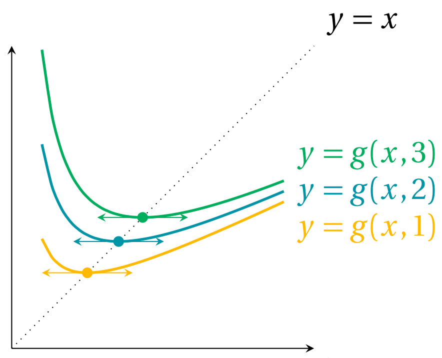
$$\sup_{\mathscr{X} \ni x \neq z \in \mathscr{X}} \frac{e(f(x), f(z))}{d(x,z)} < \infty. \tag{5.3.7}$$

5.4. Higher convergence rates

5.4.1. Heron is faster than the Banach–Caccioppoli 'prediction'. The Banach–Caccioppoli contraction principle 5.2.5 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.

5.4.2. A graphical interpretation of Heron. It is always instructive to plot something, if possible. Let us look at the graph of Heron's divide-and-average iterator for finding $\sqrt{b}$:

$$x \mapsto g(x,b) = \frac{x + {}^{b}/_{x}}{2}, \tag{5.4.1}$$



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

$$g'(x,b) := \partial_x g(x,b) = \frac{1 - {}^{b}/_{x^2}}{2} \tag{5.4.2}$$

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* $\sqrt{b} \leq x_k \leq x_{k-1}$.[∗] [∗]: Check! It follows that for each $n \geq 1$ the sequence $(x_k)_{k \in k \geq n}$ has all its terms contained in $I_n := \left[\sqrt{b}, x_n\right]$. Looking at the restriction of the function g to I_n, $g(\cdot,b)\big|_{I_n} : I_n \to \mathbb{R}$ we see that $g(I_n) \subseteq I_n$ (g maps I_n into itself) and that

$$\sup_{x \in I_n} \left|g'(x,b)\right| \leq \left|g'(x_n,b)\right| = \left|\frac{1 - {}^{b}/_{x_n{}^2}}{2}\right| = \left|\frac{x_n^2 - b}{2x_n^2}\right| \to 0, \text{ as } n \to \infty. \tag{5.4.3}$$

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 $(x_k)_{k \in k \geq n}$), we obtain that

$$\left|x_k - \sqrt{b}\right| \leq \kappa_n^{k-n} \left|x_n - \sqrt{b}\right|, \text{ where } \kappa_n := \left|\frac{x_n^2 - b}{2x_n^2}\right| \quad \forall\, k \geq n \geq 1. \tag{5.4.4}$$

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

$$\left|x_{n+1} - \sqrt{b}\right| \leq \left|\frac{x_n^2 - b}{2x_n^2}\right| \left|x_n - \sqrt{b}\right| = \frac{x_n + \sqrt{b}}{2x_n^2} \left|x_n - \sqrt{b}\right|^2 \quad \forall\, n \geq 1, \tag{5.4.5}$$

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

$$\kappa_n \leq \frac{1}{\sqrt{b}} =: \lambda, \tag{5.4.6}$$

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

$$\epsilon_{n+1} \leq \lambda \epsilon_n^2 \quad \forall\, n \geq 1, \tag{5.4.7}$$

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

$$\log \epsilon_{n+1} \leq 2 \log \epsilon_n + \log \lambda \leq 2 \log \epsilon_n \quad \forall\, n \geq 1. \tag{5.4.8}$$

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

$$d_{n+1} \geq 2 d_n \quad \forall\, n \geq N, \tag{5.4.9}$$

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

$$d_{n+1} \geq d_n + c, \tag{5.4.10}$$

for some $c = \left|\log \kappa\right|$, which is true yet much less precise a bound than (5.4.9).
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 beheviour is a special case of a more general situation that is outlined in Theorem 5.4.8 and constitutes the basis for Newton's method of Chapter 6.

5.4.3. 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

$$|x_{k+1} - \hat{x}| \leq \kappa |x_k - \hat{x}| \quad \forall\, k \geq N. \tag{5.4.11}$$

5.4.4. Definition of quadratic convergence, quaratic 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

$$|x_{k+1} - \hat{x}| \leq C |x_k - \hat{x}|^2 \quad \forall\, k \geq N. \tag{5.4.12}$$

5.4.5. Definition of general convergence rate, superlinear convergence. A sequence $(x_k)_{k \in \mathbb{N}_0}$ that converges to a limit $\hat{x}$ is said to *converge with rate p* (at least), for a given $p \in \mathbb{R}$, $p > 1$, if and only if for some $N \in \mathbb{N}_0$ and some $C > 0$ the following *rate p error reduction* inequalities are satisfied

$$|x_{k+1} - \hat{x}| \leq C |x_k - \hat{x}|^p \quad \forall\, k \geq N. \tag{5.4.13}$$

If a sequence has a convergence rate higher than 1 we say that it *converges superlinearly*, or that it exhibits *superlinear convergence*.

5.4.6. Sharp convergence rates. If $1 \leq p < q$ and given $(x_k)_{k \in \mathbb{N}_0}$ with convergence rate q then the given sequence has also convergence rate p.[$*$] This raises the question whether there is a highest p^* for which the given sequence has rate p^* but not rate p for any $p > p^*$. The answer is not always. However it is possible to show[$*$] that for any convergent sequence there is a p^* such that:

[$*$]: Check!

[$*$]: Check!

(i) for any $p \in (1, p^*)$ there exist C and N, which may depend on p, such that

$$|e_k| \leq C |e_{k-1}|^p \quad \forall\, k \geq N; \tag{5.4.14}$$

(ii) for any $p > p^*$ there exist C and N, which may depend on p, such that

$$|e_k| \geq C|e_{k-1}|^p \quad \forall\, k \geq N. \tag{5.4.15}$$

It may happen (but not always) that for $p = p^*$ there exits $C_\flat$ and $C_\sharp$ and N such that

$$C_\flat|e_{k-1}|^p \leq |e_k| \leq C_\sharp|e_{k-1}|^p. \tag{5.4.16}$$

In this case we call p^* the sharp (polynomial) convergence rate of $(x_k)_{k\in\mathbb{N}_0}$.

5.4.7. Example (convergence rates). Suppose $\alpha \in (0,1]$ and consider the iteration

$$x_k = g(x_{k-1}) \text{ for } k \in \mathbb{N} \text{ and } x_0 \text{ given in } D \tag{5.4.17}$$

where $D := [0,2]$ and

$$g(x) = 1 + \frac{1}{3}(x-1)|x-1|^\alpha. \tag{5.4.18}$$

(a) Discuss the convergence of $(x_k)_{k\in\mathbb{N}_0}$ with respect to the choice of α.[$*$] [$*$]: Check!

(b) Find out the convergence rate. (You may be able to infer something experimentally first, by using the log-log EOC, and then use it as a basis for a rigorous proof.)[$*$] [$*$]: Check!

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

$$|x_{k+1} - \hat{x}| \leq \frac{|g'|_{\mathrm{Lip}(D)}}{2}|x_k - \hat{x}|^2 \quad \forall\, k \geq N. \tag{5.4.19}$$

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

$$\epsilon_{k+1} = |x_{k+1} - \hat{x}| = \left|g(x_k) - g(\hat{x})\right| = \left|\int_0^1 g'(\hat{x} + \theta\, e_k)\,\mathrm{d}\theta\; e_k\right| \tag{5.4.20}$$

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

$$\left|g'(\hat{x} + \theta\, e_k)\right| = \left|g'(\hat{x} + \theta\, e_k) - g'(\hat{x})\right| = L\theta|e_k|. \tag{5.4.21}$$

Hence, from (5.4.20), the triangle inequality for integrals and (5.4.21) we obtain, as claimed,

$$\epsilon_{k+1} \leq \frac{L}{2}\epsilon_k^2. \tag{5.4.22}$$

□

5.4.9. Remark. Note that the quadratic Banach–Caccioppoli theorem 5.4.8 requires g', the *derivative* of iterator g, to be Lipschitz-continuous. If $g \in \mathrm{C}^2(D)$ and D is bounded this is autmatically ensured. If D is unbounded, theorem 5.4.8 can be extended to require g' only locally Lipschitz, which again would be guaranteed by $g \in \mathrm{C}^2(D)$.

If however all we know about g' is that it is in $\mathrm{C}^{0,\alpha}$ for some $\alpha \in (0,1)$, then it is possible to reprove the theorem for to obtain a convergence rate of $1 + \alpha$.

5.4.10. What if iterator's derivative is not Lipschitz? An interesting question is what happens if g' falls short of being Lipschitz. For example, it could be only α-Hölder continuous for some positive $\alpha < 1$, meaning

$$\left|g(x)-g(y)\right| \leq \left|x-y\right|^{\alpha} \text{ for all } x, y \in D. \tag{5.4.23}$$

In this case the best one can get is a convergence rate of order $1+\alpha$, which would be less thatn 2. Here's an example.
The function

$$D := [-3/8, 3/8] \ni x \mapsto |x|^{1/3} x =: g(x) \in D \tag{5.4.24}$$

has a fixed point at 0, convergence rate is better than linear, but not quadratic. Show that this is the case and calculate the rate.
Hint. Follow the proof of quadratic Banach–Caccioppoli and use Hölder continuity.

5.4.11. Can one have convergence rates lower than 1? Yes, but not for contractions since Banach–Caccioppoli ensures a rate that is 1, or higher. If the rate is lower than 1 the iterator is not a contraction, but it can still have a fixed point.

5.5. Notes, pointers and challenges

5.5.1. Generalisations of Banach–Caccioppoli's contraction principle. The Banach–Caccioppoli contraction principle can be generalised to include weaker concepts of contraction. A nice review of such methods as well as additional new results are provided by Suzuki and Alamri (2015).

5.5.2. 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 Istrăţescu, 1981, which states that

> *a continuous function $f : D \to D$ 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, an therefore useless for any practical purpose[1] 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.

5.5.3. The (sometimes 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 two bodies ($n = -2$), a power series for this problem

[1]Brouwer 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.

has been proposed by Sundman (1913) for three-bodies ($n = 3$) and one for many-bodies ($n \geq 2$) by Babadzanjanz (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).

Exercises and problems on fixed point

Problem 5.1 (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 $x_k \to 1/a$. 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 5.2 (Heron's divide and average algorithm). Let $y \in [1/2, 2]$. Consider the iteration

$$x_0 \in \mathbb{R} \text{ and } x_{k+1} := \frac{1}{2}\left(x_k + \frac{y}{x_k}\right), \text{ for } k \in \mathbb{N}_0. \tag{P5.2.1}$$

(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

$$|e_{k+1}| \leq \frac{1}{\sqrt{2}}|e_k|^2 \text{ for all } k \in \mathbb{N}_0, \tag{P5.2.2}$$

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

$$\hat{x} = g(\hat{x}) \text{ and } \left|g'(\hat{x})\right| < 1. \tag{P5.3.1}$$

(a) Prove that there exists a $\delta > 0$ such that $\left|g'(x)\right| < 1$ 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 $x_k \to \hat{x}$.

Exercise 5.4 (coding the fixed point method). Given a Lipschitz-continuous function $g : D \to D$, 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

$$x_k := g(x_{k-1}) \text{ for } k = 1, \ldots, K,\ K \leq M \text{ and } \left|f(x_K)\right| < \epsilon. \tag{P5.4.1}$$

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 $x_0, x_1, \ldots, x_K$ and the corresponding residuals $x_0 - x_1, x_1 - x_2, \ldots, x_{K-1} - x_K$.

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

$$x_{k+1} := \frac{1}{p}\left((p-1)x_k + \frac{y}{x_k^{p-1}}\right). \tag{P5.5.1}$$

(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 covergence for various starting values x_0 and various values of p.
(c) Find an error relation between two successive iterates.

Exercise 5.6 (contractions are continuous)**.** Prove that a contraction $g : D \to \mathbb{R}$ must be continuous, i.e., that for any $\epsilon > 0$ we can find at least one $\delta > 0$ such that

$$x, y \in D \text{ and } \left|x-y\right| \leq \delta \Rightarrow \left|g(x)-g(y)\right| \leq \epsilon. \tag{P5.6.1}$$

Problem 5.7 (iterative method computing a scalar's reciprocal)**.** Given $a > 0$, consider the iteration

$$x_{k+1} := 2x_k - a x_k^2. \tag{P5.7.1}$$

(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 covergence for various starting values x_0.
(c) Find an error relation between two successive iterates.
(d) Compare this method.

Exercise 5.8. (a) Suppose $g \in \mathrm{C}^0(D;D)$ with D a closed subset of $\mathbb{R}$, define the sequence $(x_k)_{k\in\mathbb{N}_0}$ with

$$x_0 \in D \text{ and } x_k := g(x_{k-1}) \text{ for } k \geq 1, \tag{P5.8.1}$$

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

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

$$\left|g(x)-g(y)\right| < \lambda\left|x-y\right| \text{ for all } x, y \in D, \tag{P5.8.2}$$

show that

$$\left|g'(x)\right| < \lambda \text{ for all } x \in D. \tag{P5.8.3}$$

Exercise 5.9. Let y be a fixed positive number, show that the transformation

$$x \mapsto \frac{1}{2}\left(x + \frac{y}{x}\right) =: g(x) \tag{P5.9.1}$$

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

$$x_k := g(x_{k-1}) \tag{P5.9.2}$$

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

$$\hat{x}^2 = y. \tag{P5.9.3}$$

Exercise 5.10. (a) Prove that if $g : D \to \mathbb{R}$ 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,

$$g(y)-g(x)=\left(\int_0^1 g'(x+\theta(y-x))\,\mathrm{d}\theta\right)(y-x), \tag{P5.10.1}$$

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

$$\left|\int f\right| \le \int |f| \tag{P5.10.2}$$

and monotonicity

$$f \le h \Rightarrow \int f \le \int h. \tag{P5.10.3}$$

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

$$|g|_{\mathrm{Lip}(C)} = \sup_C |g'| = \max_C |g'|. \tag{P5.10.4}$$

(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 5.11 (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

$$x_0 > 0 \text{ and } x_n := y/x_{n-1}, \text{ for } n \ge 1. \tag{P5.11.1}$$

(a) Show that

$$x_n \to \hat{x} \Rightarrow \hat{x} = \sqrt{y}. \tag{P5.11.2}$$

(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 6

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:

$$\text{find } \hat{E} \geq 0 \text{ such that } \hat{E} - \varepsilon \sin \hat{E} = M, \tag{6.0.1}$$

for a given *anomaly* $M \in [0, 2\pi)$ and *eccentricity* $\varepsilon \in [0, 1]$. Equation (6.0.1) 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

$$E_{k+1} := M - \varepsilon \sin E_k. \tag{6.0.2}$$

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 inputing the value ε.
Isaac Newton, possibly conscious of Kepler's method and its shortcomings, suggested the following "correction" as a way to speed up (6.0.2):

$$\tilde{E}_{k+1} := \tilde{E}_k + \frac{M - \tilde{E}_k + \varepsilon \sin \tilde{E}_k}{1 - \varepsilon \cos \tilde{E}_k}. \tag{6.0.3}$$

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

6.1. Derivations of Newton's method

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

$$\text{find } \hat{x} \text{ such that} f(\hat{x}) = 0. \tag{6.1.1}$$

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

6.1.1. 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 arbirary dimensions (even infinitely many!) is one of the advantages.

6.1.2. An analytic derivation. An analytic derivation is based on the quadratic Banach–Caccioppoli contraction principle 5.4.8. We start by turning equation (6.1.1) into a fixed point problem by adding the identity of both sides

$$\text{find } \hat{x} \text{ such that } \hat{x} = \hat{x} + f(\hat{x}). \tag{6.1.2}$$

This leads to a fixed point problem of the form (5.2.9) 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 6.1.3) 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 (6.1.1) of finding $\hat{x}$ is equivalent to solving for x the fixed point equation

$$x = x + h(x)f(x) =: g(x) \tag{6.1.3}$$

where $h(x)$ is a function that we are still free to choose, as long as $h(x) \neq 0$. From Theorem 5.4.8, we may derive a quadratic method if $g'(\hat{x}) = 0$. This leads us to choose h such that

$$0 = g'(\hat{x}) = \left[1 + h'(x)f(x) + h(x)f'(x)\right]_{x=\hat{x}} = 1 + h(\hat{x})f'(\hat{x}). \tag{6.1.4}$$

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

$$h(\hat{x}) = -\left(f'(\hat{x})\right)^{-1} = \frac{1}{f'(\hat{x})}. \tag{6.1.5}$$

Therefore a choice of h satisfying (6.1.5) will work. For example, provided $f'(x) \neq 0$ for all x, we could take

$$h(x) := -f'(x)^{-1} \quad \forall\, x \in D. \tag{6.1.6}$$

This leads to the *Newton–Raphson iteration*:

$$x_{k+1} := x_k - f'(x_k)^{-1} f(x_k) \text{ for } k \in \mathbb{N}_0. \tag{6.1.7}$$

6.1.3. 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*

$$x_0 := b \text{ and } x_k := x_{k-1} + b - \arctan x_{k-1} \quad \forall\, k \in \mathbb{N} \tag{6.1.8}$$

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

[1]The 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.

6.2. Convergence analysis

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

$$\phi(x) := \frac{f(x)}{x - \hat{x}} \text{ for } x \in D \smallsetminus \{\hat{x}\}. \tag{6.2.1}$$

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

6.2.2. Proposition (simple roots of differentiable functions).
(i) If $f \in \mathrm{Lip}(D)$ and $\hat{x}$ is a root of f then the function

$$x \mapsto \phi(x) := \frac{f(x)}{x - \hat{x}} \tag{6.2.2}$$

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

$$\lim_{x\to\hat{x}} \phi(x) = f'(\hat{x}). \tag{6.2.3}$$

$\square$

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

$$|u + v|_{\mathrm{Lip}(D)} \le |u|_{\mathrm{Lip}(D)} + |v|_{\mathrm{Lip}(D)} \tag{6.2.4}$$
$$|uv|_{\mathrm{Lip}(D)} \le |u|_{\mathrm{Lip}(D)} \|v\|_{\mathrm{L}_\infty(D)} + \|u\|_{\mathrm{L}_\infty(D)} |v|_{\mathrm{Lip}(D)}, \tag{6.2.5}$$

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

$$\left|\frac{1}{u}\right|_{\mathrm{Lip}(D)} \le \frac{|u|_{\mathrm{Lip}(D)}}{\inf_D u^2}. \tag{6.2.6}$$

Proof The proof of these inequalities is an immediate application of definition 5.2.6 and some basic manipulations. For example, to prove the “Lipschitz quotient rule” (6.2.6) it is enought to observe that

$$\left|\frac{1}{u(x)} - \frac{1}{u(y)}\right| = \left|\frac{u(x) - u(y)}{u(x)u(y)}\right| \le \frac{|u|_{\mathrm{Lip}(D)}}{\inf_D u^2} \left|x - y\right|. \tag{6.2.7}$$

$\square$

6.2.4. Theorem (local convergence of Newton–Raphson’s method). *Let D be an open subset of $\mathbb{R}$, $f \in \mathrm{C}^2(D)$ with $f'' \in \mathrm{Lip}(D)$ and $\hat{x}$ as a simple root. Then for a small enough $\rho > 0$ Newton–Raphson’s iteration (6.1.7) with initial value $x_0 \in \overline{\mathrm{B}}_\rho(\hat{x})$ 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 5.4.8 to the iteration

$$x_{k+1} = g(x_k) \text{ with } g(x) := x - f'(x)^{-1} f(x). \tag{6.2.8}$$

Let us look at the derivative of g:

$$g'(x) = 1 - f'(x)^{-1} f'(x) + \left(f'(x)\right)^{-2} f''(x) f(x) = \frac{f''(x) f(x)}{f'(x)^2}. \qquad (6.2.9)$$

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 $\left\|g'\right\|_{\mathrm{L}_\infty(B)} < 1$ and $\inf_B \left|f'\right| > 0$ with $B := \overline{\mathrm{B}}_\rho(\hat{x})$. 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 $g \in \mathrm{C}^1(B, B)$. By Lipschitz algebra, outlined in 6.2.3, it also follows that $g' \in \mathrm{Lip}(B)$. Since B is closed, it follows, by the quadratic Banach–Caccioppoli theorem that a sequence $(x_k)_{k \in \mathbb{N}_0}$ with

$$x_0 \in B \text{ and } x_{k+1} := x_k - f'(x_k)^{-1} f(x_k) \quad \forall\, k \in \mathbb{N}_0 \qquad (6.2.10)$$

converges to $\hat{x}$ with

$$|x_{k+1} - \hat{x}| \leq \frac{L}{2} |x_k - \hat{x}|^2. \qquad (6.2.11)$$

□

6.3. Variants of Newton–Raphson

Newton's method, though fast and easily generlisable 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 6.2.4 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 dimenstions.
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 6.1 (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

$$0 = \frac{1}{x} - c =: f(x). \tag{P6.1.1}$$

(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

$$\sum_{k=0}^{\infty} (1-c)^k. \tag{P6.1.2}$$

Exercise 6.2 (Newton's method vs. fixed point method). Consider solving the equation

$$f(x) := \exp(x/2) - 25x^2 = 0. \tag{P6.2.1}$$

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

(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

$$x_k := x_{k-1} - f'(x_{k-1})^{-1} f(x_{k-1}) \text{ for } k = 1, \dots, K, \text{ and } \left|f(x_K)\right| < \epsilon. \tag{P6.3.1}$$

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 6.4 (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

$$x_0 := b \text{ and } x_k := x_{k-1} + b - \arctan x_{k-1} \quad \forall\, k \in \mathbb{N} \tag{P6.4.1}$$

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

Exercise 6.5 (Newton's method)**.** Consider using Newton's method to solve for x

$$\tan x = b, \tag{P6.5.1}$$

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 6.6 (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

$$x_{k+1} = g(x_k) \text{ with } g(x) := x + h(x)f(x), \tag{P6.6.1}$$

is given by

$$h(x) = \frac{f(x)}{f(x) - f\left(x + f(x)\right)}. \tag{P6.6.2}$$

(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 $f \in \mathrm{C}^2(D, D)$ and that $f'' \in \mathrm{Lip}(D)$. Show that (P6.6.1) yields a convergent sequence for a sufficiently close initial guess x_0.
(c) Can you relax the above conditions on f?

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. Therein you may find

Strang (2009): which benefits from a video support (Strang and MIT-OCW, 2016),
Treil (2015): a modern treatment which is freely available online,
and
Halmos (1974): a classic, in my favourites as it emphasises the "coordinate-free" approach (it requires some reading skills).

All these texts favour thinking and reasoning over the (alas too often found) training and repetitive drills.

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

$$x\star(y\star z)=(x\star y)\star z \quad \forall\, x,y,z\in G, \tag{A.1.1}$$

for some $e\in G$, called a *neutral element*,

$$x\star e=e\star x=x \quad \forall\, x\in G, \tag{A.1.2}$$

and for each $x\in G$ there exists x' such that

$$x\star x'=x'\star x=e, \tag{A.1.3}$$

and x' is usually indicated as x^{-1}. We say that a group $(G,\star)$ is *Abelian* (or *commutative*) if

$$x\star y=y\star x \quad \forall\, x,y\in G. \tag{A.1.4}$$

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 $\phi : X \to X$. Consider the composition operation on X^X denoted by $\circ$ and defined as*

$$\phi \circ \psi(x) := \phi(\psi(x)) \text{ for } x \in X, \tag{A.1.5}$$

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 $\mathscr{S}$ of X^X formed by the bijective maps and show that $(\mathscr{S}, \circ)$ is a group.
(d) Show with a counterexample that $(\mathscr{S}, \circ)$ is generally not Abelian.
(e) Can you explain why X^X is a good notation?
(f) The group $\mathscr{S}$ 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

$$\mathbb{Z} = \{0, 1, -1, 2, -2, \ldots\} \tag{A.1.6}$$

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)

$$x \times (y + z) = x \times y + x \times z. \tag{A.1.7}$$

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*

$$0 \times x = 0 \quad \forall\, x \in K. \tag{A.1.8}$$

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.,

$$d(x, y) := \left| x - y \right|. \tag{A.1.9}$$

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 $\mathrm{i} := (0, 1_{\mathbb{R}})$ allowing to write each $z \in \mathbb{C}$ as $z = x + \mathrm{i}\, y$ for some unique pair $(x, y) \in \mathbb{R}$. If $z = x + \mathrm{i}\, y$ we write $x = \mathrm{re}(z) = \mathrm{re}\, z$ and $y = \mathrm{im}(z) = \mathrm{im}\, z$. The *conjugate* $\overline{z}$ or z^* of $z = x + \mathrm{i}\, y$ is $\overline{z} = z^* := x - \mathrm{i}\, y$. We have $z \in \mathbb{R}$ if and only if $z = \overline{z}$.
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 $\mathscr{V}$ is called a $\mathbb{K}$-vector space if and only if $\mathscr{V}$ is an (additive) Abelian group and a *scalar–vector multiplication* is defined

$$\begin{array}{rccl} \langle\cdot\rangle \cdot_{\mathbb{K},\mathscr{V}} : & \mathbb{K} \times \mathscr{V} & \to & \mathscr{V} \\ & (\lambda, v) & \mapsto & \langle\lambda\rangle\, v_{\mathbb{K},\mathscr{V}} = \lambda v\ , \end{array} \tag{A.2.1}$$

(where the brackets are omitted when the meaning is clear) and satisfies the properties

$$\lambda(v + w) = \lambda v + \lambda w, (\lambda + \mu)v = \lambda v + \lambda w, \tag{A.2.2}$$

for all $\lambda, \mu \in \mathbb{K}$ and $v, w \in \mathscr{V}$. The $0_{\mathbb{K}}$ and $0_{\mathscr{V}}$ 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 $\mathscr{V}$ are denoted by the same symbols + and −, respectively.
Although not a must, it is often useful to use the scalar-vector commutativity convention whereby

$$v\lambda := \lambda v. \tag{A.2.3}$$

A.2.2. Exercise. *Let $\mathscr{V}$ be a $\mathbb{K}$-vector space. Show that*

(a) For any $v \in \mathscr{V}$, $1v = v$.
(b) For any $v \in \mathscr{V}$, $0v = 0$.
(c) For any $v \in \mathscr{V}$, $(-1)v = -v$.

A.2.3. Definition of vector subspace. Given a $\mathbb{K}$-vector space $\mathscr{V}$ a *vector subspace* U of $\mathscr{V}$ is a subset $U \in \mathscr{V}$ which is closed under the vector space operations + and $\langle\cdot\rangle \cdot_{\mathbb{K},\mathscr{V}}$.

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 $\mathscr{V}$ be a $\mathbb{R}$-vector space, show that there exists a $\mathbb{C}$-vector space $\mathscr{W}$ such that $\mathscr{V} \subseteq \mathscr{W}$ and $\mathscr{V}$ is a $\mathbb{R}$-subspace, albeit not a $\mathbb{C}$-subspace there, of $\mathscr{W}$.*

Solution. Let $\mathscr{W}$ be the direct sum $\mathscr{V} \oplus \mathrm{i}\,\mathscr{V}$ whereby

$$\mathscr{V} \oplus \mathrm{i}\,\mathscr{V} := \mathscr{V} \times \mathscr{V} = \left\{ \boldsymbol{z} = \left(x, y\right) :\ x, y \in \mathscr{V} \right\} \tag{A.2.4}$$

with the vector addition of $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathscr{W}$ given by

$$\boldsymbol{z}_1 + \boldsymbol{z}_2 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ y_1 + y_2 \end{bmatrix} \tag{A.2.5}$$

and the following complex scalar-vector multiplication of $\lambda = \alpha + \mathrm{i}\beta$ with $\boldsymbol{z} = (x, y)$

$$(\alpha + \mathrm{i}\beta)\boldsymbol{z} = \begin{bmatrix} \alpha x - \beta y \\ \alpha y + \beta x \end{bmatrix}. \tag{A.2.6}$$

It is worth making sure that the axioms of $\mathbb{C}$-vector space are satisfied for $\mathscr{W}$. A notation somtimes used to indicate $\mathscr{W}$ is $\mathbb{C}\mathscr{V}$. Note that if $\mathscr{V}$ is a $\mathbb{C}$-vector space then $\mathbb{C}\mathscr{V}$ still makes sense, except it is trivial in that $\mathbb{C}\mathscr{V} = \mathscr{V}$; we use this fact to make statements shorter.

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 dimesnsional 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 $\mathbb{K}$

$$\mathbb{K}^n := \left\{ \boldsymbol{v} = \left[v_i \right]_{i=1,\dots,n} : v_i \in \mathbb{K} \quad \forall\, i = 1,\dots,n \right\} \tag{A.3.1}$$

where

$$\boldsymbol{v} = \left[v_i \right]_{i=1,\dots,n} = (v_1,\dots,v_n). \tag{A.3.2}$$

When using *matrix notation* the vectors of $\mathbb{K}^n$ are written as columns, leaving us with the 4 alternative notations:

$$\boldsymbol{v} = (v_1,\dots,v_n) = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = \left[v_i \right]_{i=1,\dots,n}. \tag{A.3.3}$$

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.*
We will also use *row vectors* in a specialised meaning, the notation of which will be introduced in A.3.1.

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 $\boldsymbol{a}^1,\dots,\boldsymbol{a}^m \in \mathbb{K}^n$, with that order, we may form a corresponding as an ordered array (or table) of vectors:

$$\boldsymbol{A} := \begin{bmatrix} \boldsymbol{a}^1 & \dots & \boldsymbol{a}^m \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}^1 & \dots & \boldsymbol{a}^m \end{bmatrix}. \tag{A.3.4}$$

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 explicity look at the column vectors forming the matrix as

$$\boldsymbol{a}^j = a_i^j \text{ for } i = 1,\dots,n,\, j = 1,\dots,m, \tag{A.3.5}$$

and in fact many textbooks merely define (somewhat misleadingly) a matrix as a "table" (or "2-dimensional array") of scalar "entries".

[1]"matrixes" is an acceptable spelling in English, but for some strange reason mathematicians dislike it and stick to the inconsistent "matrices".

Matrix notation is a special way to denote the matrix $\boldsymbol{A}$ as

$$\boldsymbol{A} = \begin{bmatrix} a_1^1 & \dots & a_n^1 \\ \vdots & \ddots & \vdots \\ a_1^m & \dots & a_n^m \end{bmatrix} = \begin{bmatrix} a_i^j \end{bmatrix}_{i=1,\dots,n}^{j=1,\dots,m}. \tag{A.3.6}$$

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] In this text, we do our best 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. Note that in Mathematical Physics, (most of) Classical Mechanics, Relativity Theory, and Differential Geometry the common convention is RIUCID.
As with vectors we can "extract" the j-th column of a matrix $\boldsymbol{A}$

$$[\boldsymbol{A}]^j = \begin{bmatrix} a_1^j \\ \vdots \\ a_n^j \end{bmatrix} = \boldsymbol{a}^j. \tag{A.3.7}$$

and its i-th row

$$[\boldsymbol{A}]_i = \begin{bmatrix} a_i^1 & \dots & a_i^m \end{bmatrix} =: \boldsymbol{a}_i. \tag{A.3.8}$$

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

$$\delta_i^j := \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \tag{A.3.9}$$

for $i, j = 1, \dots, n$ and whose i-th column and row are denoted by

$$\mathbf{e}^i := \begin{bmatrix} \delta_1^i \\ \vdots \\ \delta_n^i \end{bmatrix} \qquad \mathbf{e}_i := \begin{bmatrix} \delta_i^1 & \dots & \delta_i^n \end{bmatrix} \tag{A.3.10}$$

Matrix-matrix multiplication of a matrix $\boldsymbol{A} \in \mathbb{K}^{n\times m}$ and $\boldsymbol{B} \in \mathbb{K}^{m\times l}$ is defined as follows

$$[\boldsymbol{AB}]_i^j = \sum_{k=1}^{m} [\boldsymbol{A}]_i^k [\boldsymbol{B}]_k^j. \tag{A.3.11}$$

[2]The 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". In this basic treatment we switch the order for convenience: we never take the square (or any power for that matter) of vectors, so $\boldsymbol{a}^2$ should not confuse students into thinking that we are squaring the (unindexed) vector $\boldsymbol{a}$.

In particular, if a row $\boldsymbol{\alpha}$ and column $\boldsymbol{a}$ have the same number of entries, say n, then

$$\boldsymbol{\alpha a} = \sum_{i=1}^{n} \alpha^i a_i. \tag{A.3.12}$$

A very useful operation on matrices is the so-called *transposition* which is defined for a column

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \text{ as the row } \boldsymbol{a}^\mathsf{T} = \begin{bmatrix} a^1 & \dots & a^n \end{bmatrix} \text{ with } a^i = a_i. \tag{A.3.13}$$

And for a $\mathbb{K}^{n\times m}$ matrix

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}^1 & \dots & \boldsymbol{a}^m \end{bmatrix} \text{ as the } \mathbb{K}^{m\times n} \text{ matrix } \boldsymbol{A}^\mathsf{T} = \begin{bmatrix} \boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_m \end{bmatrix} \text{ where } \boldsymbol{a}_i := \left(\boldsymbol{a}^i\right)^\mathsf{T}. \tag{A.3.14}$$

A related operation (which is the same if $\mathbb{K} = \mathbb{R}$) is the *adjoint*

$$\boldsymbol{A}^* = \begin{bmatrix} \boldsymbol{a}^{1*} \\ \vdots \\ \boldsymbol{a}^{m*} \end{bmatrix} \text{ where } \boldsymbol{a}^{i*} := \begin{bmatrix} \overline{a}^1 & \dots & \overline{a}^n \end{bmatrix} \text{ with } \overline{a}^i = \overline{a_i} \tag{A.3.15}$$

and $\overline{z} := z^* := x - \mathrm{i}\, y$ for a complex number $z = x + \mathrm{i}\, y$.
The adjoint of a vector is often denoted by the *dot product notation*

$$\boldsymbol{a}^* = \boldsymbol{a}^* \tag{A.3.16}$$

A.3.2. Definition of linear combination, spanned space, span, dimension. Given a finite sequence of vectors $\left(\boldsymbol{v}^i\right)_{i\in[1\dots n]}$, a *linear combination* thereof, is an expression of the type

$$\sum_{i=1}^{n} \alpha_i v^i := \alpha_1 v^1 + \cdots + \alpha_n v^n. \tag{A.3.17}$$

In matrix notation this can also be written as simply as

$$\boldsymbol{v}^\mathsf{T}\boldsymbol{\alpha}, \text{ where } \boldsymbol{v}^\mathsf{T} = \begin{bmatrix} v^1 & \dots & v^n \end{bmatrix} \text{ and } \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}. \tag{A.3.18}$$

Given a set of vectors $S \subseteq \mathcal{V}$, not necessarily finite, we define the *spanned space* (also known as *span*) of S, $\operatorname{Span} S$ to be the set of all possible (finite!) linear combinations[3] of $\mathcal{V}$. In symbols this may be written as

$$\operatorname{Span} S := \left\{ \sum_{i=1}^{n} \alpha_i v^i : \ n \in \mathbb{N}, \boldsymbol{\alpha} \in \mathbb{K}^n, \left[v^i\right]^{i=1,\dots,n} \text{ in } S^n \right\}. \tag{A.3.19}$$

A.3.3. Proposition (characterisation of the span). *Given a subset S of a $\mathbb{K}$-vector space $\mathcal{V}$. The span of S is the smallest possible subspace of $\mathcal{V}$ that contains S.*

[3]Linear combinations are finite by definition. Infinite linear combinations are useful, and we always prepend the adjective in that case.

A.3.4. Definition of basis. An subset V of a $\mathbb{K}$-vector space $\mathscr{V}$ is called a *basis* if and only if

$$V \text{ is linearly independent and } \operatorname{Span} V = \mathscr{V}. \tag{A.3.20}$$

We say that $\mathscr{V}$ is a *finite dimensional vector space* if it has on basis that is finite. In this case, we think of a basis of $\mathscr{V}$ as being an ordered set[4]. Precisely we define a *finite basis* as being a row of elements of $\mathscr{V}$

$$V = \left[v^j\right]^{j=1,\dots,n} = \left[v^1 \quad \dots \quad v^n\right]. \tag{A.3.21}$$

A.3.5. Theorem (finite dimension). *Let $\mathscr{V}$ 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.

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 $\star : M \times M \to M$ is defined such that $(M,+,\star)$ is a ring with unit $\mathscr{I}$ and

$$\lambda(\mathscr{A} \star \mathscr{B}) = (\lambda\mathscr{A}) \star \mathscr{B} \quad \forall \lambda \in \mathbb{K}, \mathscr{A}, \mathscr{B} \in M \tag{A.4.1}$$

$$(\mathscr{A} + \mathscr{B}) \star \mathscr{C} = \mathscr{A} \star \mathscr{C} + \mathscr{B} \star \mathscr{C} \quad \forall \mathscr{A}, \mathscr{B}, \mathscr{C} \in M \tag{A.4.2}$$

$$\mathscr{A} \star (\mathscr{B} + \mathscr{C}) = \mathscr{A} \star \mathscr{B} + \mathscr{A} \star \mathscr{C} \quad \forall \mathscr{A}, \mathscr{B}, \mathscr{C} \in M \tag{A.4.3}$$

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 $\mathscr{A} \in M$ is *invertible* (or *nonsingular*) if there exists another element $\mathscr{B} \in M$ such that

$$\mathscr{A}\mathscr{B} = \mathscr{B}\mathscr{A} = \mathscr{I} \qquad (M\text{'s unit}). \tag{A.4.4}$$

Such a $\mathscr{B}$, if it exists, is denoted by $\mathscr{A}^{-1}$. When working with an algebra M with unit $\mathscr{I}$, the elements of the form $\lambda\mathscr{I}$ with $\lambda \in \mathbb{K}$ are identified with λ and thus $\mathscr{I}$ is often denoted simply 1.

A.4.2. Example. The prime example of an algebra is that of linear operators on a vector space $\mathscr{V}$, $\operatorname{Lin}(\mathscr{V} \to \mathscr{V})$ that will be defined later in **??**. Using the *standard isomorphism* the operator algebra $\operatorname{Lin}(\mathbb{K}^n \to \mathbb{K}^n)$ is seen to be isomorphic to the *matrix algebra* $\mathbb{K}^{n\times n}$ discussed in A.3.1.

[4]Strictly speaking a set does not need to be ordered to fulfil the two properties of a basis, but thinking of it as an orderd set organises things when it comesto coordinate vectors and matrix representation of operators.

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

$$p(X) := \sum_{i \in S} p_i X^i \tag{A.4.5}$$

for some finite set $S \subseteq \mathbb{N}_0$, $\left[p_i\right]_{i=\in,\dots,\mathbb{K}}$ for all $i \in S$, and where X^i symbolizes $X \star \cdots \star X$ 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 $S = \varnothing$) then p is called the *polynomial-zero*.
The *degree* of the polynomial p is $\deg p$ is defined as

$$\deg p := \max k \in \mathbb{N}_0 : p_k \neq 0. \tag{A.4.6}$$

(By definition $\max\varnothing = -\infty$ 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 $p : M \to M$.

A.4.4. Example (2×2 **skew symmetric equidiagonal matrices are the complex numbers).** Consider the subset C of $\mathbb{R}^{2\times2}$ matrices $\boldsymbol{A}$ that are skew symmetric and equidiagonal, i.e., of the form

$$\boldsymbol{A} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \text{ for some } a, b \in \mathbb{R}. \tag{A.4.7}$$

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 $\boldsymbol{A}$ 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

$$\boldsymbol{J} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \in C \tag{A.4.8}$$

satifies $p(\boldsymbol{J}) = \mathbf{O}$ (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 $\mathbb{C}$. This means that we have just provided a method to build $\mathbb{C}$.
This one of many ways of constructing the field of complex numbers, $\mathbb{C}$, starting from that of the real numbers, $\mathbb{R}$, is very natural. Its advantage over other methods is that it has a very nice geometric interpretation and allows for immediate definitions of modulus, argument, angle, products, and power series (sin, cos, exp) functions. For example, the modulus of the "complex number" $\boldsymbol{A}$ is $\det\boldsymbol{A}$. If you then take an $\boldsymbol{A}$ that has modulus (or determinant) 1 and note that $\boldsymbol{A}^*\boldsymbol{A} = \mathbf{I}$, it follows that $\boldsymbol{A}$ with $\det\boldsymbol{A}$ in C is a rotation matrix. Thus one can build a one-to-one correspondence between angles (rotations) and all matrices in C that are orthogonal or, equivalently, with determinant equaling 1.

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.*[5]

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*

$$\lambda p(X) = \sum_{i=0}^{\deg p} \lambda p_i(X), \tag{A.4.9}$$

$$p(X) + q(X) = \sum_{i=0}^{\deg p \vee \deg q} \left(p_i + q_i\right) X^i, \tag{A.4.10}$$

$$p(X) q(X) = \sum_{i=0}^{\deg p + \deg q} \left(\sum_{j=0}^{i} p_j q_{i-j} \right) X^i. \tag{A.4.11}$$

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*

$$p(X) = q(X)d(X) + r(X) \text{ and } \deg r < \deg d. \tag{A.4.12}$$

(Note that uniqueness is not guaranteed in general, if $\mathbb{K}$ is not $\mathbb{R}$ or $\mathbb{C}$.)

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 $\mathbb{R}$.

[5]Finite 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.5. Linear maps

Let $\mathscr{V},\mathscr{W}$ be two $\mathbb{K}$-vector spaces.

A.5.1. Linear maps. A *linear map* from $\mathscr{V}$ to $\mathscr{W}$ is a mapping $\mathscr{A}:\mathscr{V}\to\mathscr{W}$ such that

$$\mathscr{A}\left(\alpha x+\beta y\right)=\alpha\mathscr{A}x+\beta\mathscr{A}y \quad \forall\, x,y\in\mathscr{V}. \tag{A.5.1}$$

If $\mathscr{A}$ is a linear map, it is customary to write $\mathscr{A}v$ for $\mathscr{A}(v)$. It is often useful to consider the set of all linear maps from $\mathscr{V}$ to $\mathscr{W}$ which we denote as $\operatorname{Lin}(\mathscr{V}\to\mathscr{W})$. Sum and scaling of $\mathscr{W}$-valued maps (linear or not) can be defined as follows

$$\left[\alpha\mathscr{A}+\beta\mathscr{B}\right]v:=\alpha\mathscr{A}v+\beta\mathscr{B}v \text{ for } v\in\mathscr{V}. \tag{A.5.2}$$

[$*$]: Check! These operations turn the space $\operatorname{Lin}(\mathscr{V}\to\mathscr{W})$ 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 $\operatorname{Lin}(\mathscr{V}\to\mathscr{V})$. A *linear form* on $\mathscr{V}$ is a linear map from $\mathscr{V}$ to $\mathbb{K}$, i.e., when the target space is the field of scalars.

A.5.2. Theorem. *A linear map $\mathscr{A}\in\operatorname{Lin}(\mathscr{V}\to\mathscr{W})$ is completely determined by its values on a basis of $\mathscr{V}$.*

Proof We give the proof in the finite dimensional case only. Let $V=\left[v^i\right]^{i=1,\dots,n}$ be a basis of $\mathscr{V}$ and assume we know that $\mathscr{A}v^i=y^i$ for each $i=1,\dots,n$ and a given subset $\left\{y_i:\ i=1,\dots,n\right\}$ of $\mathscr{W}$, then, if x is a generic element in $\mathscr{V}$, we have $x=\sum_{i=1}^n x_i v_i$ for some $\left[x_i\right]_{i=1,\dots,n}$ in $\mathbb{K}^n$ and thus, by linearity of $\mathscr{A}$, we obtain

$$\mathscr{A}x=\sum_{i=1}^n x_i y_i. \tag{A.5.3}$$

This means that all the values of $\mathscr{A}$ can be recovered by the knowledge of its effect on the basis V. □

A.5.3. Definition of dual space, duality pairing and dual basis. The linear space of all linear forms on $\mathscr{V}$, $\operatorname{Lin}(\mathscr{V}\to\mathbb{K})$, is called the (algebraic) *dual space* of $\mathscr{V}$ and is denoted as $\mathscr{V}^*$.

It is often handy to denote the action of an element ϕ of $\mathscr{V}^*$ on an element v of $\mathscr{V}$ with the *duality bracket notation*

$$\langle\phi\,|\,v\rangle_{\mathscr{V}}:=\phi(v). \tag{A.5.4}$$

You should convince yourself of the useful fact that the duality bracket is linear in each of its two arguments separately, i.e.,

$$\langle\alpha\phi+\beta\psi\,|\,v\rangle_{\mathscr{V}}=\alpha\langle\phi\,|\,v\rangle_{\mathscr{V}}+\beta\langle\psi\,|\,v\rangle_{\mathscr{V}} \tag{A.5.5}$$

and

$$\langle\phi\,|\,\alpha v+\beta w\rangle_{\mathscr{V}}=\alpha\langle\phi\,|\,v\rangle_{\mathscr{V}}+\beta\langle\phi\,|\,w\rangle_{\mathscr{V}} \tag{A.5.6}$$

$$\forall\,\phi,\psi\in\mathscr{V}^*,v,w\in\mathscr{V},\alpha,\beta\in\mathbb{K}. \tag{A.5.7}$$

In other words $\langle\cdot\,|\,\cdot\rangle_{\mathscr{V}}$ is a bilinear form.

Given a finite dimensional vector space $\mathscr{V}$ and a basis of it, $V = \left[v^i\right]^{i=1,\dots,n}$, we may define its associated *dual basis* as the ordered subset V^* of $\mathscr{V}^*$, with elements v_i^V, for all $i = 1,\dots,n$ by

$$\langle v_i^V \,|\, v^j \rangle_{\mathscr{V}} = \delta_i^j \quad \forall\, j = 1,\dots,n. \tag{A.5.8}$$

When the basis V is understood from the context we omit the superscript and write simply v_i for v_i^V.

A.5.4. Example (coordinate maps). If $\mathscr{V} = \mathbb{K}^n$, then the dual basis of the natural basis is the set of *coordinate maps*:

$$e_i(\boldsymbol{x}) = x_i \text{ for each } \boldsymbol{x} = \left[x_j\right]_{j=1,\dots,n} \in \mathbb{K}^n \text{ and } i = 1,\dots,n. \tag{A.5.9}$$

Hence e_i is simply the linear form that returns the i-th component, or coordinate, of a vector $\boldsymbol{x} \in \mathbb{K}^n$. It is immediate that

$$\langle e_i \,|\, \boldsymbol{x} \rangle_{\mathbb{K}^n} = \mathbf{e}_i \boldsymbol{x}, \text{ where } e_i^j = \delta_i^j. \tag{A.5.10}$$

A.5.5. Theorem (primal–dual isomorphism). *The dual $\mathscr{V}^*$ of a finite dimensional $\mathbb{K}$-vector space $\mathscr{V}$ is itself a finite vector space and*

$$\dim \mathscr{V}^* = \dim \mathscr{V}. \tag{A.5.11}$$

Moreover given a basis $V = \left[v^i\right]^{i=1,\dots,n}$ of $\mathscr{V}$, its dual basis $V^* = \left[v^i\right]^{i=1,\dots,n}$, as defined in §A.5.3 is actually a basis of $\mathscr{V}^*$ (which justifies the terminology).
Proof All we need to show is that V^* is a basis of $\mathscr{V}^*$. □

A.5.6. The bidual space and the natural primal-bidual identification. Let $\mathscr{V}$ be a vector space and $\mathscr{V}^*$ its dual. Since $\mathscr{V}^*$ is also a vector space, it has also a dual space, which we call the *bidual space* of $\mathscr{V}$ and denote by $\mathscr{V}^{**}$. By the prima–dual isomorphism theorem A.5.5 we know that $\mathscr{V}^{**} \leftrightarrows \mathscr{V}^*$ and $\mathscr{V}^* \leftrightarrows \mathscr{V}$ hence $\mathscr{V}^{**} \leftrightarrows \mathscr{V}$.
If $\mathscr{V}$ is finite dimensional then, $\mathscr{V}^{**}$ is naturally identified with $\mathscr{V}$ as follows: each $v \in \mathscr{V}$ gives rise to a linear functional v^{**} on $\mathscr{V}'$ given by

$$\langle v^{**} \,|\, \phi \rangle_{\mathscr{V}^*} = \langle \phi \,|\, v \rangle_{\mathscr{V}} \quad \forall\, \phi \in \mathscr{V}^*. \tag{A.5.12}$$

A.5.7. Theorem (matrix representation of linear maps). *If $\mathscr{V}$ and $\mathscr{W}$ are finite dimensional and respective bases thereof $V = \left[v^i\right]^{i=1,\dots,m}$ and $W = \left[w^i\right]^{i=1,\dots,n}$, then there is a one-to-one correspondence ι between* $\operatorname{Lin}(\mathscr{V} \to \mathscr{W})$ *and* $\mathbb{K}^{n\times m}$, *such that for each $\mathscr{A}$ and $\boldsymbol{A} := \iota\mathscr{A}$ we have*

$$\begin{gathered} \boldsymbol{A} = \left[a_i^j\right]_{i=1,\dots,n}^{j=1,\dots,m} \text{ and} \\ a_i^j = w^i(\mathscr{A} v_j) \\ \text{for } i = 1,\dots,n,\ j = 1,\dots,m. \end{gathered} \tag{A.5.13}$$

where $\left\{w^i : i = 1,\dots,n\right\} =: W^*$ is the dual basis of W as defined in §A.5.3.

A.5.8. Example. Consider the space $\mathbb{P}^n$ of polynomials of degree n with real coefficients. A basis for $\mathbb{P}^n$ is given by ϕ_i, $i=0,\dots,n$, where $\phi_i(x)=x^i$. Hence

$$\dim \mathbb{P}^n = n+1 \tag{A.5.14}$$

and a polynomial $p \in \mathbb{P}^n$ can be written as

$$p = \sum_{i=1}^{n} p_i \phi_i \qquad \big(\text{i.e., } p(x) = p_0 + p_1 x + \dots + p_n x^n\big) \tag{A.5.15}$$

for a unique $\boldsymbol{p} = \big(p_i\big)_{i\in[i\dots0]n} \in \mathbb{R}^{n+1}$. The dual basis of $\big\{\phi_i :\ i=0,\dots,n\big\}$ is given by the "i-th coefficient extractor"

$$\phi_i(p) = p_u i. \tag{A.5.16}$$

Examples of linear forms on $\mathbb{P}^n$ are

(a) Evaluation at a given point $x_0 \in \mathbb{R}$

$$\delta_{x_0} : \mathbb{P}^n \ni p \mapsto \langle \delta_{x_0} \,|\, p \rangle := p(x_0) \in \mathbb{R}. \tag{A.5.17}$$

This operator is known also as the *Dirac mass*, or *Dirac delta*. The row-vector $\boldsymbol{d}_{x_0}^{\mathsf{T}}$ representing δ_{x_0}, say for $n=4$, is thus given by

$$\boldsymbol{d}_{x_0}^{\mathsf{T}} = \begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & x_0^4 \end{bmatrix}. \tag{A.5.18}$$

(b) n-th derivative

$$\mathrm{D}^n : \mathbb{P}^n \ni p \mapsto p^{(n)} \in \mathbb{R} \tag{A.5.19}$$

The row-vector representing D^4 in the monomial basis of $\mathbb{P}^4$ is

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5.20}$$

(c) Definite integral over $[0,1]$

$$I : \mathbb{P}^n \ni p \mapsto \int_0^1 p(x)\,\mathrm{d}x. \tag{A.5.21}$$

The row-vector representing I, say for $n=3$, is

$$\begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \end{bmatrix} \tag{A.5.22}$$

Consider now the first derivative map

$$\begin{array}{rccl} \mathrm{D}: & \mathbb{P}^n & \longrightarrow & \mathbb{P}^{n-1} \\ & p & \longmapsto & \mathrm{D}p = p' \end{array}$$
$$\text{where } \mathrm{D}p(x) := p'(x) := \frac{\mathrm{d}p(x)}{\mathrm{d}x} \text{ for all } x \in \mathbb{R}. \tag{A.5.23}$$

[$*$]: Check!

From basic calculus we know that D is linear.[$*$] Let us find the matrix $\boldsymbol{D}$ representing D with respect to the monomial basis $\big\{\phi_i :\ i=0,\dots,n\big\}$. Writing $\boldsymbol{D} = \big[d_i^j\big]_{i=1,\dots,n}^{j=1,\dots,n-1}$ we have, for $i,j=0,\dots,n$:

$$d_i^j = \phi^i(\mathrm{D}\phi_j) = \phi^i(j\phi_{j-1}) = j\delta_i^{j-1}. \tag{A.5.24}$$

That is, in extended form, for $n=5$ say

$$\boldsymbol{D}=\begin{bmatrix}0&1&0&0&0&0\\0&0&2&0&0&0\\0&0&0&3&0&0\\0&0&0&0&4&0\\0&0&0&0&0&5\end{bmatrix}. \tag{A.5.25}$$

A.5.9. 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 $\mathcal{A}\in\operatorname{Lin}(\mathscr{V}\to\mathscr{W})$ is defined as the unique

$$\begin{array}{rccc}\mathcal{A}^*: & \mathscr{W}^* & \to & \mathscr{V}^* \\ & \psi & \mapsto & \phi\end{array} \quad \text{where } \langle\phi\,|\,v\rangle_{\mathscr{V}}=\langle\psi\,|\,\mathcal{A}v\rangle_{\mathscr{W}} \quad \forall\, v\in\mathscr{V}. \tag{A.5.26}$$

We then note that this is indeed a linear operator and it is unique.

A.5.10. Invertibility. A (not necessarily linear) map $\mathcal{A}:\mathscr{V}\to\mathscr{W}$ is called[6]

- ⋆ *left invertible* or *postinvertible* if and only if there exists a map $\mathcal{B}:\mathscr{W}\to\mathscr{V}$ such that

$$\mathcal{B}\mathcal{A}=\mathrm{id}_{\mathscr{V}}, \tag{A.5.27}$$

  the identity on $\mathscr{V}$. Such a $\mathcal{B}$ is called a *left inverse* or *postinverse.*
- ⋆ *right invertible* or *preinvertible* if and only if there exists a map $\mathcal{B}:\mathscr{W}\to\mathscr{V}$ such that

$$\mathcal{A}\mathcal{B}=\mathrm{id}_{\mathscr{W}}. \tag{A.5.28}$$

- ⋆ 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.5.11).

A.5.11. Problem (invertible maps and linearity). *Consider a map $\mathcal{A}:V\to W$.*

(a) Show that is postinvertible if and only if it is injective.
(b) Preinvertible if and only if it is surjective.
(c) Deduce that $\mathcal{A}$ is invertible if and only if is is bijective.
(d) The map $\mathcal{A}$ is invertible if and only if its postinverse exists and is unique.
(e) The map $\mathcal{A}$ is invertible if and only if its preinverse exists and is unique.
(f) Show that the postinverse $\mathcal{B}$ of a postinvertible linear map $\mathcal{A}$ is linear.

[6]The 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".

A.6. Multlinear forms and determinants

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 may consult Abraham, Marsden and Ratiu (1988) who propose an advanced treatment of abstract multilinear algebra.

A.6.1. Definition of multilinear forms on a space. A *multilinear form* (also known as *k-linear*), on a $\mathbb{K}$-vector space $\mathscr{V}$ is a function $a : \mathscr{V} \times \cdots \times \mathscr{V} \to \mathbb{K}$ that is linear in each of its arguments separately, i.e., such that

$$\begin{aligned} a(x_1,\dots,\lambda x_i + \mu y_i,\dots,x_k) = \lambda a(x_1,\dots,x_i,\dots,x_k) + \mu a(x_1,\dots,y_i,\dots,x_k) \\ \forall\, x_1,\dots,x_k, y_i, \in \mathscr{V} \lambda,\mu \in \mathbb{K} \text{ and } i = 1,\dots,k. \end{aligned} \tag{A.6.1}$$

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 $\mathscr{V}$ is d.

A.6.2. Definition of symmetric and antisymmetric forms. A k-linear form a on $\mathscr{V}$ is called *symmetric* if and only if

$$a(x_1,\dots,x_i,\dots,x_j,\dots,x_k) = a(x_1,\dots,x_j,\dots,x_i,\dots,x_k) \tag{A.6.2}$$

or *antisymmetric* if and only if

$$a(x_1,\dots,x_i,\dots,x_j,\dots,x_k) = -a(x_1,\dots,x_j,\dots,x_i,\dots,x_k). \tag{A.6.3}$$

A.6.3. Definition of alternating form. A k-linear form a on a $\mathbb{K}$-vector space $\mathscr{V}$ is called *alternating k-form* if and only if for any linearly dependent system of k vectors, $x_1,\dots,x_k$, we have $a(x_1,\dots,x_k) = 0$. Alternating forms are known as *antisymmetric forms, skew symmetric forms* or *exterior forms.* The set of all k-linear alternating forms is a $\mathbb{K}$-vector space denoted as $\wedge^k(\mathscr{V})$. It follows from the definition that for $k = 1$, $\wedge^1(\mathscr{V}) = \mathscr{V}^*$ while an exercise shows that for $k = n$, $\wedge^n(\mathscr{V}) \leftrightarrows \mathbb{K}$. For $k = 0$, it is convenient to define $\wedge^0(\mathscr{V}) := \mathbb{K}$.

A.6.4. Exercise. *Show that if $\mathscr{V}$ is a vector space on $\mathbb{K}$, with $n = \dim \mathscr{V}$ then the set $\wedge^n(\mathscr{V})$ has dimension 1 and is thus isomorphic to $\mathbb{K}$.*
Hint. Suppose $a, b \in \wedge^n(\mathscr{V})$, $b \neq 0$, then there exists a scalar $a_b \in \mathbb{K}$ such that

$$a(x_1,\dots,x_n) = a_b\, b(x_1,\dots,x_n) \quad \forall\,(x_1,\dots,x_n) \in \mathscr{V}^n. \tag{A.6.4}$$

A.6.5. 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.6.6. Pull-back and push-forward of k-linear forms by linear maps. Suppose $\mathcal{A} \in \operatorname{Lin}(\mathscr{V} \to \mathscr{W})$. Given a k-linear form b, the *pull-back* of b by $\mathcal{A}$ (also known as $\mathcal{A}$-pull back of b), is defined as the k-linear a on $\mathscr{V}$ by

$$a(x_1, \dots, x_k) = b(\mathcal{A}x_1, \dots, \mathcal{A}x_k) \quad \forall\, x_1, \dots, x_k \in \mathscr{V}; \tag{A.6.5}$$

we denote the unique such b with $\mathcal{A} \triangleleft b$. The concept of pull-back generalises (in one particular direction) that of map composition in that when $k = 1$, we have $\mathcal{A} \triangleleft b = b \circ A$.

If $\mathcal{A}$ is invertible, then we can also define its *push-forward* of a k-linear form $a \in \wedge^k(\mathscr{V})$ as the k-linear form b given by

$$b(y_1, \dots, y_k) = a\left(\mathcal{A}^{-1} y_1, \dots, \mathcal{A}^{-1} y_k\right), \tag{A.6.6}$$

and we denote such b with $\mathcal{A} \triangleright a$

A.6.7. 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 $\mathscr{V}$ with altenating d-forms ($d = \dim \mathscr{V}$) 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 nor on the concept of length or norm. Of course, any "new" definition should coincide with the "old" definition, which is the formulaic one given by the recursive formula on a matrix. This goal will be achieved by the Leibniz formula theorem.

A.6.8. Exercise (existence and uniqueness of a linear operator's determinant). *Consider a linear operator $\mathcal{A} \in \operatorname{Lin}(\mathscr{V} \to \mathscr{V})$.*

(a) Show that for each $b \in \wedge^n(\mathscr{V})$ there is a constant $\alpha_b \in \mathbb{K}$ such that

$$\mathcal{A} \triangleleft b(x_1, \dots, x_n) = \alpha_b\, b(x_1, \dots, x_n) \tag{A.6.7}$$

(b) Show that $\alpha_b = \alpha_c$ for any $b, c \in \wedge^n(\mathscr{V})$.

Thanks to § A.6.8, given a linear operator $\mathcal{A} \in \operatorname{Lin}(\mathscr{V} \to \mathscr{V})$, with $n := \dim \mathscr{V}$, we may define its determinant as the unique scalar α such that

$$\mathcal{A} \triangleleft b = \alpha b \quad \forall\, b \in \wedge^n(\mathscr{V}). \tag{A.6.8}$$

We write $\det \mathcal{A} := \alpha$.

A.6.9. Exercise. *Let $\mathcal{A} \in \operatorname{Lin}(\mathscr{V} \to \mathscr{V})$. Then $\det \mathcal{A} \neq 0$ if and only if $\mathcal{A}$ is invertible.*

A.6.10. Exercise. *Let $b \in \wedge^n(\mathscr{V})$, where $n := \dim \mathscr{V}$ for a finite dimensional vector space. Suppose $V = \left[v^1, \dots, v^n\right]$ is a basis of $\mathscr{V}$, then*

$$\det \mathcal{A} = \frac{b\left(\mathcal{A}v^1, \dots, \mathcal{A}v^n\right)}{b(v^1, \dots, v^n)} \tag{A.6.9}$$

A.6.11. Theorem (Leibniz formula). *Suppose $\mathscr{V}$ is a finite dimensional vector space, $n := \dim \mathscr{V}$ and V a basis of $\mathscr{V}$. Let $\mathcal{A} \in \operatorname{Lin}(\mathscr{V} \to \mathscr{V})$ and $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ the matrix associated with $\mathcal{A}$ and $V = \left[v^i\right]^{i=1,\dots,n}$. Then the determinant can be theoretically calculated*[7] *as*

$$\det \mathcal{A} = \sum_{\sigma\in \operatorname{Bij}[1\dots n]} (-1)$$

$\sigma \prod_{j=1}^{n} a^{j}_{\sigma j}$ (A.6.10) *where* $\operatorname{Bij}[1\dots n]$ *is the symmetric group of* $[1\dots n]$*, i.e., the group of all permutations of n elements, i.e., the group of all bijections on* $1n$ *and* σ *is the parity of a permutation* $\sigma \in \operatorname{Bij}[1\dots n]$.

A.6.12. 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 $a : \mathscr{V} \times \mathscr{V} \to \mathbb{K}$, such that

$$a(x, \lambda y + \mu z) = \lambda a(x, y) + \mu a(x, z) \tag{A.6.11}$$

$$a(\lambda x + \mu y, z) = \overline{\lambda}\, a(x, y) + \overline{\mu}\, a(x, z) \tag{A.6.12}$$

for any $\lambda, \mu \in \mathbb{K}$ and $x, y, z \in \mathbb{K}$.

A.6.13. 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.7. Spectral theory

In this section we review the basics about eigenvalues, eigenvectors, eigenspaces, eigenetc. This is also known as "spectral theory".

A.7.1. Complexification of a linear operator. Let $\mathscr{V}$ be a $\mathbb{R}$-vector space and $\mathscr{W} := \mathbb{C}\mathscr{V}$ its *complexification* (or *complexified space*), as introduced in §A.2.5. Suppose $\mathcal{A}$ is a linear operator on $\mathscr{V}$. Then $\mathcal{A}$ can be extended in a unique way to an operator $\mathcal{A}_{\mathbb{C}}$, called the *complexified operator* (or $\mathcal{A}$'s *complexification*) on $\mathscr{W}$ by defining, for $\boldsymbol{z} = (x, y) \in \mathscr{W}$

$$\mathcal{A}_{\mathbb{C}} \begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} \mathcal{A}x \\ \mathcal{A}y \end{bmatrix}. \tag{A.7.1}$$

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.7.2. Operator eigenstuff. Let $\mathscr{V}$ be a $\mathbb{K}$-vector space and $\mathbb{C}\mathscr{V}$ its complexified space. Given an operator $\mathcal{A} \in \operatorname{Lin}(\mathscr{V} \to \mathscr{V})$, we say that $\lambda \in \mathbb{C}$ is an *eigenvalue* of $\mathcal{A}$ if and only if there exists a $v \in \mathbb{C}\mathscr{V}$ such that $v \neq 0$ and

$$\mathcal{A}v = \lambda v. \tag{A.7.2}$$

[7] Leibniz's formula, while interesting in theory, has little practical use. In fact, for matrices larger than 3×3, determinants do not have an important practical use. The are, though, very important to develop analysis.

Any vector v satisfying the *eigenequation* (A.7.2) is called an *eigenvector* of $\mathcal{A}$ associated with λ. A pair (λ, v) satisfying (A.7.2) is called an *eigenpair.* The set of all eigenvectors $\boldsymbol{v}$ for a given λ and operator $\mathcal{A}$ forms a vector subspace of $\mathscr{V}$ called the *eigenspace* associated with $(\mathcal{A}, \lambda)$ and denoted $\operatorname{Eis}_\lambda^{\mathcal{A}}$, i.e.,

$$\operatorname{Eis}_\lambda^{\mathcal{A}} = \operatorname{Ker}(\mathcal{A} - \lambda). \tag{A.7.3}$$

Note that although trivial the definition of eigenspace works also when λ is not an eigenvalue of $\mathcal{A}$ and in fact turns out to be useful in some cases. To be sure, the following holds

$$\lambda \in \operatorname{Spec}(\mathcal{A}) \Leftrightarrow \operatorname{Eis}_\lambda^{\mathcal{A}} \neq \{0\}. \tag{A.7.4}$$

The set of all eigenvalues is called the *point spectrum* of the operator $\mathcal{A}$.

A.7.3. Matrix eigenstuff. Similarly, given a square matrix $\boldsymbol{A} \in \mathbb{K}^{n \times n}$ and thinking of $\boldsymbol{A}$ as being an operator on $\mathbb{K}^n$ (with the canonical basis), we use the same terminology for $\lambda \in \mathbb{C}$ and $\boldsymbol{v} \in \mathbb{C}^n$ in that we say that λ and $\boldsymbol{v}$ are an *eigenvalue* and an *eigenvector,* respectively, or that $(\lambda, \boldsymbol{v})$ is an *eigenpair of the matrix* $\boldsymbol{A}$ if and only if

$$\boldsymbol{v} \neq \mathbf{0} \text{ and } \boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}. \tag{A.7.5}$$

For reasons that become apparent later, whenever possible it is best to work with operators rather than matrices when dealing with eigenstuff.

A.7.4. Exercise (eigenspaces are invariant). *Let $\phi : X \to X$ be an operator on X, and $S \subseteq X$ we call S an invariant (sub)set for ϕ if and only if its image is a subset of S*

$$\phi(S) \subseteq S. \tag{A.7.6}$$

Show that if $\mathcal{A}$ is a linear operator on a vector space $\mathscr{V}$ then any one $\operatorname{Eis}_\lambda^{\mathcal{A}}$ *of its eigenspaces is an invariant space for $\mathscr{V}$.*

A.7.5. Eigenvalue multiplicity. The *geometric multiplicity* of an eigenvalue λ of $\boldsymbol{A}$ is defined as

$$\dim \operatorname{Ker}(\boldsymbol{A} - \lambda \mathbf{I}), \tag{A.7.7}$$

and its *algebraic multiplicity* is its multiplicity as a root of $\boldsymbol{A}$'s characterisitic polynomial, i.e., the highest $\alpha \in \mathbb{N}$ such that

$$\det(\boldsymbol{A} - X\mathbf{I}) = (X - \lambda)^\alpha q(X) \tag{A.7.8}$$

for a polynomial $q(X)$. The geometric multiplicity is between 1 and the algebraic multiplicity. If geometric multiplicity is strictly smaller than the algebraic one, the corresponding eigenvalue is called *defective.* A matrix is called *defective* if and only if one of its eigenvalues is defective. A matrix that is not defective and has all eigenvalues in $\mathbb{C}$ is called *nondefective* or *complex diagonalisable* or $\mathbb{C}$*-diagonalisable* (which is hard to prounce but less ugly than nondefective, which is why we stick to it). The reason for the name "diagonalisable" will be apparent in Lemma A.7.12. A $\mathbb{C}$-diagonalisable $\mathbb{K}^{n\times n}$ matrix that has all its eigenvalues in $\mathbb{K}$ is called $\mathbb{K}$*-diagonalisable* or simply *diagonalisable.*

A.7.6. Exercise (real matrices and their eigenvalues). *Let the real matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ have eigenvalue λ in $\mathbb{C}$; show the following.*

(a) If $\lambda \in \mathbb{R}$ then it has an associated real eigenvector $\boldsymbol{v}$, i.e.,

$$\boldsymbol{v} \in \mathbb{R}^n \text{ and } \boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}. \tag{A.7.9}$$

(b) If λ is not a real number then $\operatorname{Eis}_\lambda^A \cap \mathbb{R}^n = \{0\}$, i.e., all nonzero eigenvectors associated to λ and $\boldsymbol{A}$ are not in $\mathbb{R}^n$.

(c) Find an example for $n = 2$ where the eigenvalues are not real.

A.7.7. Spectral or singular values and spectrum. The concept of spectral value (also known as singular value) generalises that of eigenvalue. A *spectral value* of $\mathcal{A} \in \operatorname{Lin}(\mathscr{V} \to \mathscr{V})$ is any λ for which $\mathcal{A}-\lambda$ is singular (not invertible). An eigenvalue is necessarily a spectral value. If $\mathscr{V}$ is finite dimensional then a spectral value must be an eigenvalue, but if $\dim \mathscr{V} = \infty$, then there are examples of operators with spectral values that are not eigenvalues. The set of all spectral values of $\mathcal{A}$ is called *spectrum* of $\mathcal{A}$; we denote this set by $\operatorname{Spec}\mathcal{A}$ [8] The *spectral radius* of $\mathcal{A}$ is defined as

$$\sup\{|\lambda| : \lambda \in \mathbb{C} \text{ spectral value of } \mathcal{A}\}. \tag{A.7.10}$$

A.7.8. Example (eigenvectors of the derivative operator). Consider the operator

$$\begin{aligned} \mathrm{D}\colon \quad \mathrm{C}^\infty(\mathbb{R}) &\to \mathrm{C}^\infty(\mathbb{R}) \\ \phi &\mapsto \mathrm{D}\phi := \phi' \end{aligned} \tag{A.7.11}$$

where

$$\phi'(x) := \frac{\mathrm{d}}{\mathrm{d}x}\phi(x). \tag{A.7.12}$$

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 a the operator D.

A.7.9. Theorem (spectral invariance under basis transformation). *If $\mathscr{V}$ is finite dimensional, with a basis V, $\mathcal{A} \in \operatorname{Lin}(\mathscr{V} \to \mathscr{V})$. Then λ is an eigenvalue of $\mathscr{A}$ if and only if λ is an eigenvalue of $\boldsymbol{A}$, the matrix representing $\mathcal{A}$ with respect to V.*

Proof If λ is an eigenvalue of $\mathscr{A}$, then $\lambda x = \mathscr{A}x$ for some $x \in \mathscr{V}$. For $\boldsymbol{x}$ such that $x = \boldsymbol{v}^\intercal \boldsymbol{x}$ we have, for any $i = 1, \dots, n$,

$$v^i(\mathscr{A}x) = v^i(\mathscr{A}\sum_{j=1}^n v_j x^j) = \sum_{j=1}^n v^i(\mathscr{A}v_j)x^j = \sum_{j=1}^n a_i^j x^j = [\boldsymbol{A}\boldsymbol{x}]_i. \tag{A.7.13}$$

On the other hand, for the same $i = 1, \dots, n$,

$$v^i(\mathscr{A}x) = v^i(\lambda x) = [\lambda \boldsymbol{x}]_i. \tag{A.7.14}$$

Since i is arbitrary it follows that

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}. \tag{A.7.15}$$

□

[8]Many authors denote the spectrum of $\mathcal{A}$ by $\sigma(\mathcal{A})$, but that may lead to confusion as σ is used in many other contexts.

A.7.10. Corollary (spectral invariance under similarity transformation). *Let $\boldsymbol{V} \in \mathbb{K}^{n\times n}$ be an invertible matrix and $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ a generic matrix. Then the spectrum of $\boldsymbol{A}$ and that of $\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V}$ coincide.*
Proof The short proof consists in thinking of $\boldsymbol{A}$ and $\boldsymbol{B} := \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V}$ to represent the same linear operator $\mathcal{A}$ in two different bases, the canonical one for $\boldsymbol{A}$ and

$$\boldsymbol{V} := \left[\boldsymbol{v}^j\right]^{j=1,\dots,n} = \left[\boldsymbol{v}^1 \quad \dots \quad \boldsymbol{v}^n\right], \tag{A.7.16}$$

for $\boldsymbol{B}$. Using then the fact that a $\lambda \in \mathbb{C}$ an eigenvalue of $\mathcal{A}$ if and only if it is an eigenvalue of $\boldsymbol{A}$, and $\boldsymbol{B}$, it follows that the spectra of $\boldsymbol{A}$ and $\boldsymbol{B}$ are the same.[9] □

A.7.11. Lemma (eigenbasis characterisation of diagonalisable matrices). *A matrix $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ is $\mathbb{K}$-diagonalisable if and only if all its eigenvalues are in $\mathbb{K}$ and $\mathbb{K}^n$ has a basis formed of eigenvectors of $\boldsymbol{A}$.*

A.7.12. Lemma (spectral factorisation of a diagonalisable matrix). *Let $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ be not defective and have the eigenvalues $\lambda_i \in \mathbb{C}$, for $i = 1,\dots,n$, with a correspondingly ordered set of eigenvectors $\left[\boldsymbol{v}^1 \quad \dots \quad \boldsymbol{v}^n\right] =: \boldsymbol{V} \in \mathbb{C}^{n\times n}$ then*

$$\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda}, \text{ where } \boldsymbol{\Lambda} = \operatorname{Diag}(\lambda_1,\dots,\lambda_n) := \begin{bmatrix} \lambda_1 & \mathbf{0}^\mathsf{T} & 0 \\ \mathbf{0} & \ddots & \mathbf{0} \\ 0 & \mathbf{0}^\mathsf{T} & \lambda_n \end{bmatrix}. \tag{A.7.17}$$

A.7.13. Remark. It is customary, but not necessary, to consider the eigenvalues ordered in such a way that

$$\operatorname{re}(\lambda_{i-1}) = \operatorname{re}(\lambda_i) \text{ and } \operatorname{im}(\lambda_{i-1}) \leq \operatorname{im}(\lambda_i) \tag{A.7.18}$$

for all $i = 2,\dots,n$.

A.7.14. Remark (where's the factorisation?) Note that $\boldsymbol{V}$ postmultiplies on the left member and premultiplies on the right member of (A.7.17). The fact that it switches sides it the point of the lemma because the immediate and most important corollaries are the following factorisations

$$\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} = \boldsymbol{\Lambda} \text{ and } \boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}. \tag{A.7.19}$$

A.7.15. Proof of lemma A.7.12. By definition of eigenpair we know that for each $i = 1,\dots,n$

$$\boldsymbol{A}\boldsymbol{v}^i = \lambda_i \boldsymbol{v}^i. \tag{A.7.20}$$

By writing this in a matrix form and doing some matrix manipulations we get

$$\begin{aligned} \boldsymbol{A}\boldsymbol{V} &= \left[\lambda_1\boldsymbol{v}^1 \quad \dots \quad \lambda_n\boldsymbol{v}^n\right] = \left[\lambda_1\boldsymbol{V}\mathbf{e}^1 \quad \dots \quad \lambda_n\boldsymbol{V}\mathbf{e}^n\right] \\ &= \left[\boldsymbol{V}\lambda_1\mathbf{e}^1 \quad \dots \quad \boldsymbol{V}\lambda_n\mathbf{e}^n\right] = \sum_{i=1}^{n} \boldsymbol{V}\lambda_i\mathbf{e}^i\mathbf{e}_i = \boldsymbol{V}\boldsymbol{\Lambda}, \end{aligned} \tag{A.7.21}$$

[9] A longer proof consists in messing around with characteristic polynomials and determinants: we leave is as an exercise for determinant lovers.

upon noting that a handy expression using tensor products for $\boldsymbol{\Lambda}$ is

$$\boldsymbol{\Lambda} = \sum_{i=1}^{n} \lambda_i \mathbf{e}^i \mathbf{e}_i. \tag{A.7.22}$$

□

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 Euclidan 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.[10] 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 $\mathscr{V}$ is a function

$$\begin{array}{rccc} \|\cdot\| : & \mathscr{V} & \to & \mathbb{R} \\ & x & \mapsto & |x| \,, \end{array} \tag{A.8.1}$$

such that

(1) $\|\cdot\|$ is positive definite, i.e.,

$$\|x\| \geq 0 \text{ and } \|x\| = 0 \Leftrightarrow x = 0, \tag{A.8.2}$$

(2) $\|\cdot\|$ is homogeneous (of degree 1), i.e.,

$$\|\lambda x\| = |\lambda| \, \|x\| \quad \forall\, \lambda \in \mathbb{K}, x \in \mathscr{V}, \tag{A.8.3}$$

(3) $\|\cdot\|$ satisfies the triangle inequality, i.e.,

$$\left\|x + y\right\| \leq \|x\| + \left\|y\right\|. \tag{A.8.4}$$

A.8.2. Definition of convergent sequence. Let $(\mathscr{V}, \|\cdot\|_{\mathscr{V}})$ be a normed $\mathbb{K}$-vector space. A sequence $(x_n)_{n\in\mathbb{N}}$ is *convergent* if and only if for a given $x \in \mathscr{V}$

$$\lim_{n\to\infty} \|x_n - x\|_{\mathscr{V}} = 0. \tag{A.8.5}$$

Equivalently, $(x_n)_{n\in\mathbb{N}}$ converges to x if and only if

$$\forall\, \varepsilon > 0 : \exists\, N \in \mathbb{N} : n \geq N \;\Rightarrow\; \|x_n - x\| < \varepsilon. \tag{A.8.6}$$

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 $x = y$.*

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

$$\lim_{n\to\infty} x_n. \tag{A.8.7}$$

Thanks to uniqueness we can use the "=", "≤", "≥" and other operators applying in the vector space when manipulating limits.

[10] In fact, this section belongs more to an Analysis synopsis rather than a Linear Algebra one, but I put it here for completeness.

A.8.5. Theorem (vector algebra of limits). *If $(\mathscr{V}, \|\cdot\|_{\mathscr{V}})$ 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 $\mathscr{V}$, then $(a_n x_n + b_n y_n)_{n\in\mathbb{N}}$ converges and*

$$\lim_{n\to\infty} \left[a_n x_n + b_n y_n\right] = \lim_{n\to\infty} a_n \lim_{n\to\infty} x_n + \lim_{n\to\infty} b_n \lim_{n\to\infty} y_n. \tag{A.8.8}$$

A.9. Scalar (or inner) products and Hilbert spaces

```
Warning:  This section is not complete.
```

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 $\mathscr{V}$ is called an *inner product space* if and only if $\mathscr{V}$ is endowed with an *inner product*, i.e., a map

$$\begin{aligned} \langle\cdot,\cdot\rangle : \quad \mathscr{V}\times\mathscr{V} &\to \mathbb{K} \\ (v,w) &\mapsto \langle v,w\rangle \end{aligned} \tag{A.9.1}$$

which satisfies

(i) linearity in the second argument:

$$\left\langle v, \lambda w + \mu x\right\rangle = \lambda \langle v,w\rangle + \mu \langle w,v\rangle; \tag{A.9.2}$$

(ii) Hermitianity:

$$\langle v,w\rangle = \overline{\langle w,v\rangle} \quad \forall\, v,w\in\mathscr{V}; \tag{A.9.3}$$

(iii) positivity:

$$\langle v,v\rangle \geq 0 \quad \forall\, v\in\mathscr{V}; \tag{A.9.4}$$

(iv) definiteness:

$$\langle v,v\rangle = 0 \Rightarrow v = 0. \tag{A.9.5}$$

A.9.2. Remark (sesquilinearity of the scalar product). Note that i and ii imply that an inner-product is sesquilinear:

$$\left\langle \lambda v + \mu w, x\right\rangle = \overline{\left\langle x, \lambda v + \mu w\right\rangle} = \overline{\lambda}\,\overline{\langle x,v\rangle} + \overline{\mu}\,\overline{\langle x,w\rangle} = \overline{\lambda}\,\langle v,x\rangle + \overline{\mu}\langle w,x\rangle \tag{A.9.6}$$

A.9.3. Hilbertian norms. On an inner product space $(\mathscr{V}, \langle\cdot,\cdot\rangle)$ a norm is given by the function $\|\cdot\|$ defined

$$\|x\| \geq 0 \text{ and } \|x\|^2 = \langle x,x\rangle. \tag{A.9.7}$$

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

$$(\boldsymbol{x},\boldsymbol{y}) \mapsto \boldsymbol{x}^*\boldsymbol{y}, \tag{A.9.8}$$

the corresponding norm is called the *Euclidean norm* and is denoted by

$$|\boldsymbol{x}|_2 \text{ or } |\boldsymbol{x}|. \tag{A.9.9}$$

A.9.5. Definition of Hilbert space. An inner product space $(\mathscr{V},\langle\cdot,\cdot\rangle)$ is said to be a *Hilbert space* if and only if $\mathscr{V}$ is complete with respect to the distance

$$d(v,w) := \sqrt{\langle v-w, v-w\rangle}. \tag{A.9.10}$$

A.9.6. Definition of orthonormal system. Consider and inner product $\mathbb{K}$-vector space $\mathscr{V}$. A subset $\mathscr{O} \subseteq \mathscr{V}$ forms an orthonormal system in if and only if

$$\langle v,w\rangle = \delta_v^w = \begin{cases} 1 & \text{if } v = w, \\ 0 & \text{if } v \neq w. \end{cases} \tag{A.9.11}$$

A.9.7. Proposition. *If $\mathscr{V}$ is a finite dimensional inner product $\mathbb{K}$-vector space then any orthonormal system $\mathscr{O} \subseteq \mathscr{V}$ has at most* $\dim \mathscr{V}$ *elements.*

A.9.8. Definition of Hilbert (or orthonormal) basis. Let $\mathscr{V}$ be a Hilbert space, we say that an orthonormal system $\mathscr{V}$ forms a Hilbert basis (also known as orthonormal basis) of $\mathscr{V}$ if and only if each element $x \in \mathscr{V}$ can be represented as an absolutely convergent series

$$x = \sum_{v\in\mathscr{V}_x} b^v v \tag{A.9.12}$$

for some sequence $(b_v)_{v\in\mathscr{V}_x}$ where $\mathscr{V}_x \in \mathscr{V}$ and $\mathscr{V}_x$ is countable.

A.9.9. Remark. Noting that any finite sum is an absolutely convergent series the definition above makes sense also when $\mathscr{V}$ is finite dimensional.

A.9.10. Remark. Given that the series converges absolutely, order doesn't matter.

A.9.11. Theorem (Hilbert dimension). *Let $\mathscr{V}$ be a Hilbert space, then any two orthonormal bases are equipotent. I.e., if $\mathscr{V}$ is finite dimensional, any orthonormal basis has* $\dim \mathscr{V}$ *elements, if $\mathscr{V}$ 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. $\square$

A.9.12. Definition. A Hilbert space $\mathscr{V}$ is called *separable Hilbert space* if and only if $\mathscr{V}$ 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. $\square$

A.9.14. Theorem (Riesz representation). *Let $\mathscr{V}$ be a separable Hilbert space, then ϕ is a (continuous) linear form on $\mathscr{V}$ if and only if there exists an $v \in \mathscr{V}$ such that*

$$\phi(x) = \langle x, v\rangle. \tag{A.9.13}$$

This creates a one to one correspondence between $\mathscr{V}$ and its (topological) dual $\mathscr{V}'$.

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 inifinite 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 opertors isomorphism). *Let $\mathscr{V}$ be a vector space with inner product $\langle\cdot,\cdot\rangle$.*

(a) For each (continuous) linear operator $\mathcal{A}\in\operatorname{Lin}(\mathscr{V}\to\mathscr{V})$ the map a defined by

$$\begin{aligned} a:\ \mathscr{V}\times\mathscr{V} &\to \mathbb{K}\\ (x,y) &\mapsto \left\langle x,\mathscr{A}y\right\rangle \end{aligned}\quad, \tag{A.9.14}$$

is a sesquilinear form on $\mathscr{V}$.

(b) For each (continuous) sesquilinear form a on $\mathscr{V}$, there exists a linear operator $\mathscr{A}:\mathscr{V}\to\mathscr{V}$ such that

$$a(x,y)=\left\langle x,\mathscr{A}y\right\rangle. \tag{A.9.15}$$

Thus the space of (continuous) bilinear forms on $\mathscr{V}$ and that of (continuous) linear operators on $\mathscr{V}$ are isomorphic.

Proof

(a) Suppose $\mathcal{A}\in\operatorname{Lin}(\mathscr{V}\to\mathscr{V})$ is given and a defined by

$\square$

A.9.17. Adjoint operators. Given an operator $\mathscr{A}:\mathscr{V}\to\mathscr{V}$ on a Hilbert space $(\mathscr{V},\langle\cdot,\cdot\rangle)$ we define the (Hilbert–Riesz) *adjoint operator* of $\mathscr{A}$ as the operator $\mathscr{B}$ such that

$$\langle\mathscr{B}v,w\rangle=\langle v,\mathscr{A}w\rangle\quad\forall\, v,w\in\mathscr{V}. \tag{A.9.16}$$

By Theorem A.9.16 we know that there exists a unique linear operator $\mathscr{B}$ satisfying (A.9.16), so this definition is fully justified.

Exercises and problems on Linear algebra

Problem A.1. Consider a map $\mathcal{A}:V\to W$.

(a) Show that is postinvertible if and only if it is injective.
(b) Preinvertible if and only if it is surjective.
(c) Deduce that $\mathcal{A}$ is invertible if and only if is is bijective.
(d) The map $\mathcal{A}$ is invertible if and only if its postinverse exists and is unique.
(e) The map $\mathcal{A}$ is invertible if and only if its preinverse exists and is unique.
(f) 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.

Exercise A.3 (real matrices and their eigenvalues)**.** Let the real matrix $\boldsymbol{A}\in\mathbb{R}^{n\times n}$ have eigenvalue λ in $\mathbb{C}$; show the following.

(a) If $\lambda\in\mathbb{R}$ then it has an associated *real* eigenvector $\boldsymbol{v}$, i.e.,

$$\boldsymbol{v}\in\mathbb{R}^n \text{ and } \boldsymbol{A}\boldsymbol{v}=\lambda\boldsymbol{v}. \tag{PA.3.1}$$

(b) If λ is *not a real number* then $\operatorname{Eis}_\lambda^{\boldsymbol{A}}\cap\mathbb{R}^n=\{0\}$, i.e., all nonzero eigenvectors associated to λ and $\boldsymbol{A}$ are not in $\mathbb{R}^n$.

(c) Find an example for $n = 2$ where the eigenvalues are not real.

Problem A.4 (diagonalisable matrix). Suppose a matrix is diagonalisable, i.e., for a given $\boldsymbol{V} \in \mathrm{GL}(\mathbb{R}^n)$ and $\boldsymbol{\Lambda} = \operatorname{Diag}(\lambda_1, \dots, \lambda_n)$, we have

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}, \tag{PA.4.1}$$

then the columns of $\boldsymbol{V}$ form a system of eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{\Lambda}$ has all the eigenvalues of $\boldsymbol{A}$ on the diagonal.

Exercise A.5. For each of the following choices

$$\text{(a)}\quad \boldsymbol{A} = \frac{1}{3}\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \quad \text{(b)}\quad \boldsymbol{A} = \frac{1}{3}\begin{bmatrix} 2 & 2 \\ -2 & 7 \end{bmatrix} \qquad \text{(c)}\quad \boldsymbol{A} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

do the following:

- ⋆ compute the eigenvalues of $\boldsymbol{A}$,
- ⋆ for each eigenvalue λ an basis for the eigenspace $\operatorname{Eis}_\lambda^{\boldsymbol{A}}$,
- ⋆ if the matrix is real, decide whether the $\operatorname{Eis}_\lambda^{\boldsymbol{A}} \cap \mathbb{R}^n$ is different than $\{0\}$,
- ⋆ decide whether the matrix provides $\mathbb{K}^n$ (fir the appropriate $\mathbb{K}$ and n) with an $\boldsymbol{A}$-eigenbasis.

Problem A.6 (diagonalisable matrix). Show that a matrix $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ is diagonalisable if and only if it is not defective, i.e., each of its eigenvalues has its algebraic multiplicity equaling its geometric multiplicity.

Exercise A.7 (eigenvalues of powers). Suppose $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ has eigenvalue $\lambda \in \mathbb{C}$ show the following.

(a) For any $n \in \mathbb{N}_0$, λ^n is an eigenvalue of $\boldsymbol{A}^n := \boldsymbol{A} \cdots \boldsymbol{A}$ (n times).
(b) If $\boldsymbol{A}$ is invertible then the above formula extends to $n \in \mathbb{Z}$. (Start with $n = -1$.)
(c) What can you say about the eigenspaces of $\boldsymbol{A}$ and $\boldsymbol{A}^n$, for $n \in \mathbb{N}_0$ ($\mathbb{Z}$ if $\boldsymbol{A}$ invertible).
(d) How about a polynomial $p(x) \in \mathbb{K}[X]$? Does the expression $p(\boldsymbol{A})$ make any sense?

Problem A.8 (eigenvalues of analytic functions). Suppose the power series $\sum_{n=0}^{\infty} a_n z^n$ has radius of convergence $R > 0$ and thus defines an analytic function $\phi(z) = \sum_{n=0}^{\infty} a_n z^n$. Let $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ ($\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$) have spectral radius $\rho < R$; show the following.

(a) The series of $\sum_{n=0}^{\infty} a_n \boldsymbol{A}^n$ converges in $\mathbb{K}^{n\times n}$, hence its sum defines the matrix $\phi(\boldsymbol{A})$.
(b) If $(\lambda, \boldsymbol{v}) \in \mathbb{K} \times \mathbb{K}^d$ is an eigenpair of $\boldsymbol{A}$ then $(\phi(\lambda), \boldsymbol{v})$ is an eigenpair of $\phi(\boldsymbol{A})$.

Problem A.9 (range and null spaces of symmetric semidefinite matrices). Let $\boldsymbol{A}$ be a symmetric positive semidefinite matrix, i.e.,

$$\boldsymbol{A}^* = \boldsymbol{A} \text{ and } \boldsymbol{v}^*\boldsymbol{A}\boldsymbol{v} \geq 0 \quad \forall\, \boldsymbol{v} \in \mathbb{K}^n. \tag{PA.9.1}$$

Show the following

(a) The range and the null space of $\boldsymbol{A}$ are orthogonal complements; in symbols

$$\operatorname{Ran} \boldsymbol{A} = (\operatorname{Ker} \boldsymbol{A})^{\perp}. \tag{PA.9.2}$$

(b) If $\boldsymbol{B}$ is symmetric positive semidefinite then $\boldsymbol{A}+\boldsymbol{B}$ is so as well,

$$\operatorname{Ker}(\boldsymbol{A}+\boldsymbol{B}) \subseteq \operatorname{Ker}\boldsymbol{A} \tag{PA.9.3}$$

and

$$\operatorname{Ran}(\boldsymbol{A}+\boldsymbol{B}) \supseteq \operatorname{Ran}\boldsymbol{A}. \tag{PA.9.4}$$

Problem A.10. If a matrix is unitary then its determinant has absolute value 1. Is the converse true?

Problem A.11. Show that the only linear transformations $\mathcal{A} \in \operatorname{Lin}\left(\mathbb{R}^2 \to \mathbb{R}^2\right)$ that preserve the scalar product up to a multiplicative factor, i.e., for some $d \in \mathbb{R}$

$$(\mathcal{A}\boldsymbol{z})^{\mathsf{T}} \mathcal{A}\boldsymbol{w} = d^2 \boldsymbol{z}^{\mathsf{T}} \boldsymbol{w} \quad \forall\, z, w \in \mathbb{R}^2, \tag{PA.11.1}$$

are those and only those represented with respect to the canonical basis $\left[\mathbf{e}^1\ \mathbf{e}^2\right]$ by a matrix of the form

$$\begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix} \text{ or } \begin{bmatrix} \alpha & \beta \\ \beta & -\alpha \end{bmatrix} \tag{PA.11.2}$$

for $\left(\alpha, \beta\right) \in \mathbb{R}^2$.

Problem A.12. Consider a zero-diagonal symmetric (square) matrix $\boldsymbol{A}$ of order n

$$\boldsymbol{A} = \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} & \cdots & a_{1n} \\ a_{21} & 0 & a_{23} & a_{24} & \cdots & a_{2n} \\ a_{31} & a_{32} & 0 & a_{34} & \cdots & a_{3n} \\ a_{41} & a_{42} & a_{43} & 0 & \cdots & a_{4n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & a_{n4} & \cdots & 0 \end{bmatrix} \tag{PA.12.1}$$

Denote A's eigenvalues $\lambda_i, i = 1, \cdots, n$.
Consider two matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ with eigenvalues μ_i and $\nu_i, i = 1, 2, \cdots, n$,which are obtained from matrix A by replacing a_{11} entry to 1 and -1 respectively
i.e

$$B = \begin{bmatrix} 1 & a_{12} & a_{13} & a_{14} & \cdots & a_{1n} \\ a_{21} & 0 & a_{23} & a_{24} & \cdots & a_{2n} \\ a_{31} & a_{32} & 0 & a_{34} & \cdots & a_{3n} \\ a_{41} & a_{42} & a_{43} & 0 & \cdots & a_{4n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & a_{n4} & \cdots & 0 \end{bmatrix}_n$$

$$C = \begin{bmatrix} -1 & a_{12} & a_{13} & a_{14} & \cdots & a_{1n} \\ a_{21} & 0 & a_{23} & a_{24} & \cdots & a_{2n} \\ a_{31} & a_{32} & 0 & a_{34} & \cdots & a_{3n} \\ a_{41} & a_{42} & a_{43} & 0 & \cdots & a_{4n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & a_{n4} & \cdots & 0 \end{bmatrix}_n$$

My questions are
1.Can we find any relation between λ_i and μ_i?
2.Can we find any relation between λ_i and ν_i?
3.Can we find any $\sum_{i=1}^{n} \left|\mu_i\right|$ in terms of λ_i?
4.Can we find any $\sum_{i=1}^{n} |\nu_i|$ in terms of λ_i?

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 tha 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

$$\forall \epsilon > 0 : \exists N \in \mathbb{N} : n \geq N \Rightarrow |x - x_n| < \epsilon. \tag{B.1.1}$$

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 $\lim_{n\to\infty} x_n$. 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* $n \in \mathbb{N}$,

$$|\hat{x} - x| \leq |\hat{x} - x_n| + |x_n - \bar{n}|. \tag{B.1.2}$$

Therefore for any given ϵ there exists $\hat{N}$ and $\bar{N}$ such that

$$n \geq \hat{N} \vee \bar{N} \Rightarrow |\hat{x} - x_n| \vee |x - x_n| \leq \frac{\epsilon}{2}. \tag{B.1.3}$$

It follows from (B.1.2) and by choosing appropriately n that

$$|\hat{x} - x| \leq \epsilon. \tag{B.1.4}$$

(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,$\lim_{n\to\infty} x_n$ is unique and the equal sign can safely be used to write:

$$\lim_{n\to\infty} x_n = x. \tag{B.1.5}$$

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, $\lim_{n\to\infty} x_n$.

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*

$$\forall\, \epsilon > 0 : \exists\, N \in \mathbb{N} : n \geq N, k \in \mathbb{N}_0 \Rightarrow |x_n - x_{n+k}| < \epsilon. \tag{B.1.6}$$

Proof The proof can be found in some elementary analysis text, e.g.,. □

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 $\mathrm{C}^0(\Omega)$ of continuous functions on an open set Ω, the set $\mathrm{C}^1(\Omega)$ 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

$$\|\cdot\| : X \to \mathbb{R} \tag{B.2.1}$$

which satisfies

$$\|x\| \geq 0 \text{ and } \|x\| = 0 \iff x = 0 \qquad (\textit{positivity}), \tag{B.2.2}$$

$$\|\lambda x\| = |\lambda|\,\|x\| \qquad (\textit{homogeneity}), \tag{B.2.3}$$

$$\text{and } \left\|x + y\right\| \leq \|x\| + \left\|y\right\| \qquad (\textit{triangle inequality}), \tag{B.2.4}$$

for all $x, y \in X$ and $\lambda \in \mathbb{K}$.

[1] Even 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 (1970).

B.2.3. Common Vector Norms. The most widely used norm in $\mathbb{K}^m$ is the Euclidean norm

$$|\boldsymbol{x}| = |\boldsymbol{x}|_2 := \left({x_1}^2 + {x_2}^2 + \cdots + {x_m}^2\right)^{1/2}. \tag{B.2.5}$$

This norm is the most natural, as $\left|\boldsymbol{x} - \boldsymbol{y}\right|$ is the usual physical distance between $\boldsymbol{x}$ and $\boldsymbol{y}$; and because of this it takes the name of *Euclidean norm*. What distinguishes this norm (among all norms on $\mathbb{K}^n$) is that it can expressed in terms of the usual *inner* (*scalar* or *dot*) *product*:

$$|\boldsymbol{x}|^2 = \boldsymbol{x}^*\boldsymbol{x} \tag{B.2.6}$$

where $\boldsymbol{x}^*$ denotes the adjoint vector (transpose–conjugate) of $\boldsymbol{x}$

$$\boldsymbol{x}^*\boldsymbol{y} = {x_1}^* y_1 + \cdots + {x_m}^* y_m \qquad \left(\text{where } z^* = \overline{z}, \text{ for } z \in \mathbb{C}\right). \tag{B.2.7}$$

Arguably, the most useful property of this norm, is the Cauchy–Bunyakovski–Schwarz inequality[2]

$$\left|\boldsymbol{x}^*\boldsymbol{y}\right| \leq |\boldsymbol{x}| \left|\boldsymbol{y}\right|. \tag{B.2.8}$$

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

$$|\boldsymbol{x}|_p = \left(|x_1|^p + |x_2|^p + \cdots + |x_m|^p\right)^{1/p}. \tag{B.2.9}$$

- ⋆ Choosing $p = 1$ gives the so-called the *Manhattan taxi-cab norm* or L_1 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 $p \to \infty$ in the definition we obtain

$$|u|_\infty = \max_{1 \leq i \leq m} |u_i|. \tag{B.2.10}$$

B.2.4. Exercise. *Show that $|\cdot|_p$ defines a norm on $\mathbb{K}^m$.*

B.2.5. Vector norm inequalities. Besides the triangle inequality, vector norms satisfy many other useful inequalities. For example if $p, q \in [1, \infty]$ with $1/p + 1/q = 1$, we have the *Hölder inequality*

$$\left|\boldsymbol{x}^*\boldsymbol{y}\right| \leq \sum_{i=1}^{d} \left|x_i y_i\right| \leq |\boldsymbol{x}|_p \left|\boldsymbol{y}\right|_q \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{K}^d. \tag{B.2.11}$$

[2]This 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.

It follows that the Euclidean norm satisfies the *Cauchy–Bunyakovsky–Schwarz inequality*[3]

$$\left|\boldsymbol{x}^*\boldsymbol{y}\right| \le \sum_{i=1}^{d} \left|x_i y_i\right| \le |\boldsymbol{x}|_2 \left|\boldsymbol{y}\right|_2 \quad \forall \boldsymbol{x},\boldsymbol{y} \in \mathbb{K}^d. \tag{B.2.12}$$

B.3. Matrix analysis

B.3.1. Matrix norms. The notion of size of, or distance between, matrixes will also be required, and there are special 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, matrices.
Suppose that $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices in $\mathbb{K}^{m\times m}$, a *matrix norm* is a function that associates to each matrix a real number and that satisfies the following properties:

(1) positivity: $|\boldsymbol{A}| \ge 0$, and $|\boldsymbol{A}| = 0 \Rightarrow \boldsymbol{A} = \mathbf{O}$ ($\mathbf{O}$ is the zero matrix).
(2) homogeneity: $\left|\mu\boldsymbol{A}\right| = \left|\mu\right| |\boldsymbol{A}|, \forall\mu \in \mathbb{K}$.
(3) subadditvity (also known as norm triangle inequality): $|\boldsymbol{A}+\boldsymbol{B}| \le |\boldsymbol{A}| + |\boldsymbol{B}|$.
(4) submultiplicativity: $|\boldsymbol{A}\boldsymbol{B}| \le |\boldsymbol{A}||\boldsymbol{B}|$.

The first three properties are the same as those defining a vector norm (so a matrix norm is always a plain norm), while the fourth property is typical of matrix norms only, since the matrix product of two vectors in $\mathbb{R}^m$ is not defined. Furthermore there are norms on $\mathbb{K}^{m\times m}$ that satisfy

B.3.2. Example (maximum row-sum norm). Often we will consider the *maximum row-sum* norm

$$|\boldsymbol{A}|_{\max} = \max_{i=1,\dots,m} \sum_{j=1}^{m} \left|a_i^j\right|. \tag{B.3.1}$$

This constitutes a matrix norm (see B.3.3).
The maximum row-sum norm is "compatible" with the L_∞ norm of $\mathbb{R}^m$, in that it the following property is satisfied

$$|\boldsymbol{A}\boldsymbol{x}|_\infty \le |\boldsymbol{A}|_{\max} |\boldsymbol{x}|_\infty \tag{B.3.2}$$

for all $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{x} \in \mathbb{R}^m$.
Due to this compatibility property, the same symbol $|\cdot|_\infty$ is used for both the vector and the matrix norm.

B.3.3. Exercise (max row sum is a norm). *Check that the maximum row-sum norm satisfies indeed the properties for a matrix norm and deserves the name "norm".*

[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 inquality 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.3.4. Definition of induced bounded-linear operator norm. Given two normed vector spaces $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ and a linear transformation $T : X \to Y$, we say that T is a *bounded-linear operator* if and only if[4] [5]

$$\sup_{x \in X \setminus \{0\}} \frac{\|Tx\|_Y}{\|x\|_X} < \infty. \tag{B.3.3}$$

If this happens we define the *operator norm* of T *induced* by the norms of X and Y as the left-hand side above

$$\|T\|_{X \to Y} := \sup_{x \in X \setminus \{0\}} \frac{\|Tx\|_Y}{\|x\|_X}. \tag{B.3.4}$$

If $X = Y$ and $\|\cdot\|_X = \|\cdot\|_Y$ then we write $\|T\|_X$ instead of $\|T\|_{X \to X}$ and when X and Y are clear from the context we omit the subscript and simply write $\|T\|$.

B.3.5. Induced matrix p-norms. There is a general procedure to define matrix norms in $\mathbb{K}^{d \times k}$ based on Definition B.3.4. Indeed, a matrix $\boldsymbol{A} \in \mathbb{K}^{d \times k}$ can be identified with the *linear map* $T_{\boldsymbol{A}} : \mathbb{K}^k \to \mathbb{K}^d$ defined by

$$T_{\boldsymbol{A}}(\boldsymbol{y}) := \boldsymbol{A}\boldsymbol{y} \in \mathbb{K}^d \quad \forall \boldsymbol{y} \in \mathbb{K}^k. \tag{B.3.5}$$

Since this identification (i.e., a one-to-one surjective map) T is so common, it is customary to write $\boldsymbol{A}$ for both the matrix and the transformation $T_{\boldsymbol{A}}$.
Given two norms, one on $\mathbb{K}^d$ and one on $\mathbb{K}^k$, the *induced matrix norm* of $\boldsymbol{A}$ 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 $|\boldsymbol{A}|_p$ too. Namely, for a matrix $\boldsymbol{A} \in \mathbb{K}^{d \times k}$, we define

$$|\boldsymbol{A}|_p := \sup_{\boldsymbol{y} \in \mathbb{K}^k \setminus \{\boldsymbol{0}\}} \frac{\left|\boldsymbol{A}\boldsymbol{y}\right|_p}{\left|\boldsymbol{y}\right|_p}, \ \forall\, p \in [1, \infty]. \tag{B.3.6}$$

B.3.6. Exercise (matrix norms). *Check that this definition makes $|\cdot|_p$ a matrix norm, in the sense of §B.3.1, on $\mathbb{K}^{d \times k}$.*

B.3.7. Proposition (matrix p-norm dual inequalities). *Let $p, q \in [1, \infty]$ such that $1/p + 1/q = 1$. Given $\boldsymbol{A} \in \mathbb{K}^{d \times k}$, and denoting by $\boldsymbol{a}_i$ the i-th row (vector) of $\boldsymbol{A}$ for $i \in [1 \dots d]$, we have that*

$$|\boldsymbol{A}|_p \le \left| \left(|\boldsymbol{a}_i|_q \right)_{i=1,\dots,d} \right|_p, \tag{B.3.7}$$

where the norms $|\cdot|_q$ and $|\cdot|_p$ appearing on the right hand side are taken, respectively, on $\mathbb{R}^k$ and $\mathbb{R}^d$.

[4] The definition of bounded-linear operator is truly interesting when one of the space X is infinite dimensional. If X is finite dimensional then, by compactness, it can be shown that all linear maps from X into Y are bounded-linear.

[5] Note that the definition of T being a bounded-linear operator is *not* equivalent to say T is a bounded function and a linear map. Another unfortunate terminology that is often overlooked in many textbooks and leading to confusion amongst beginners.

Further, for $p = 1$, or ∞ the equality is satisfied (but for $p \in (1,\infty)$ the equality may fail). Namely,

$$
\begin{aligned}
|\boldsymbol{A}|_1 &= \sum_{i=1}^{d} \max_{j=1,\dots,k} \left| a_i^j \right| \\
|\boldsymbol{A}|_\infty &= \max_{i=1,\dots,d} \sum_{j=1}^{k} \left| a_i^j \right|
\end{aligned}
\tag{B.3.8}
$$

A proof of this result is obtained by applying Hölder's inequality (B.2.11).

B.3.8. Definition of Frobenius norm. Given a matrix $\boldsymbol{A}$ its *Frobenius norm* is given by

$$
|\boldsymbol{A}|_{(2,2)} := \sqrt{\sum_{i,j=1}^{d,k} \left| a_i^j \right|^2}. \tag{B.3.9}
$$

B.3.9. Neumann's series. A useful result that makes use of matrix norms goes by the name of Neumann's series lemma. It can be formulated as follows.

LEMMA. *Given a matrix $\boldsymbol{M} \in \mathbb{K}^{d\times d}$, such that $|\boldsymbol{M}| < 1$—where $|\cdot|$ is any matrix (vector-induced) norm—then the inverse of $\mathbf{I} - \boldsymbol{M}$ exists and*

$$
[\mathbf{I} - \boldsymbol{M}]^{-1} = \sum_{k=0}^{\infty} \boldsymbol{M}^k. \tag{B.3.10}
$$

B.3.10. 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

$$
\mathrm{C}^0(D) := \left\{ f : D \to \mathbb{K} : f \text{ is continuous on } D \right\}, \tag{B.3.11}
$$

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

$$
\operatorname{Span} S := \left\{ \sum_{s\in V} \lambda_s s : V \text{ finite subset of } S \text{ and } (\lambda_s)_{s\in V} \in \mathbb{K}^s \right\}, \tag{B.3.12}
$$

in words $\operatorname{Span} S$ 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 $X = \operatorname{Span} G$; 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

$$C_1 \|x\| \leq \|x\|' \leq C_2 \|x\| \quad \forall x \in X. \tag{B.3.13}$$

B.3.11. 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 $f : X \to X$ 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.3.12. 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

$$\mathrm{C}^0(\overline{\Omega}) := \left\{ f : \Omega \to \mathbb{K} : \ f \text{ is continuous at each } x \in \Omega \text{ and } f \text{ is bounded on } \Omega \right\}. \tag{B.3.14}$$

We can equipping X with either $\|\cdot\|_{\mathrm{L}_\infty(\Omega)}$ or $\|\cdot\|_{\mathrm{L}_2(\Omega)}$, where

$$\left\| f \right\|_{\mathrm{L}_\infty(\Omega)} := \sup_{x\in\Omega} \left| f(\boldsymbol{x}) \right| \tag{B.3.15}$$

$$\left\| f \right\|_{\mathrm{L}_2(\Omega)} := \sqrt{\int_\Omega \left| f(\boldsymbol{x}) \right|^2 \mathrm{d}\boldsymbol{x}}. \tag{B.3.16}$$

It is a good exercise to check that both define a norm on $\mathrm{C}^0(\overline{\Omega})$. It is also possible to show, by using Hölder's inequality for function,

$$\int_\Omega \left| f(\boldsymbol{x}) g(\boldsymbol{x}) \right| \mathrm{d}\boldsymbol{x} = \sup_\Omega \left| f \right| \int_\Omega \left| g(\boldsymbol{x}) \right| \mathrm{d}\boldsymbol{x}, \tag{B.3.17}$$

that every sequence of functions in X which converges with respect to the $\|\cdot\|_{\mathrm{L}_\infty(\Omega)}$ converges with respect to $\|\cdot\|_{\mathrm{L}_2(\Omega)}$, 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.3.13. Function-valued functions. Ordinary differential equations (ODEs) typically involve functions (or fields) of the form

$$\begin{array}{rccc} \boldsymbol{f}: & [0,T] \times D & \to & D \\ & (t,\boldsymbol{y}) & \mapsto & \boldsymbol{f}(t,\boldsymbol{y}) \end{array}, \tag{B.3.18}$$

with $D \subseteq \mathbb{R}^d$ an open domain. Often, we denote by $\boldsymbol{f}(t)$ the function $\boldsymbol{f}$ *frozen at time* t given by

$$\begin{array}{rccc} \boldsymbol{f}(t): & D & \to & D \\ & \boldsymbol{y} & \mapsto & \boldsymbol{f}(t,\boldsymbol{y}) \end{array} \tag{B.3.19}$$

If we then set t free, we could view $\boldsymbol{f}$ not as a function mapping $[0,T] \times D$ into D but as a function mapping $[0,T]$ into $\mathrm{Map}(D,D)$, 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 $\boldsymbol{f}(t)$, as $t \in [0,T]$. For example, we may ask that $\boldsymbol{f}(t) \in \operatorname{Lip}(D;D)$, i.e., the set of all operators that satisfy a (global) Lipschitz condition on D, or that $\boldsymbol{f}(t) \in \operatorname{Lip}_{\text{loc}}(D,D)$ which means it is merely locally Lipschitz.

B.3.14. 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

$$\begin{aligned} \mathrm{B}_r^{\|\cdot\|}(x) &:= \left\{ y \in X : \left\| x - y \right\| < r \right\}, \\ \overline{\mathrm{B}}_r^{\|\cdot\|}(x) &:= \left\{ y \in X : \left\| x - y \right\| \leq r \right\}. \end{aligned} \tag{B.3.20}$$

When the norm $\|\cdot\|$ is obvious from the contex, 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

$$\overline{\mathrm{B}_r(x)} = \overline{\mathrm{B}}_r(x). \tag{B.3.21}$$

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 $\mathrm{B}_r(x) \subseteq N$.
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 $\mathrm{B}_{r(x)}(x) \subseteq G$.

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 $\boldsymbol{f} : S \subseteq \mathbb{K}^k \to \mathbb{K}^d$ where S is any subset of $\mathbb{K}^k$ (not necessarily open!) is said to be *continuous at a point $\boldsymbol{x} \in S$* if and only if for each $\varepsilon > 0$ there exists a $\delta = \delta_{x,\varepsilon} > 0$ such that

$$\boldsymbol{y} \in S \text{ and } \left| \boldsymbol{y} - \boldsymbol{x} \right| < \delta \Rightarrow \left| \boldsymbol{f}(\boldsymbol{y}) - \boldsymbol{f}(\boldsymbol{x}) \right| < \varepsilon. \tag{B.4.1}$$

Note that, if $\boldsymbol{x}$ if fixed, then the δ will generally change if ε changes. This is sometimes emphasised by writing δ_ϵ instead of δ. Also, if $\boldsymbol{f}$ is continuous at two different point, say $\boldsymbol{x} \neq \boldsymbol{z}$ in S, then the δ will change with the point under scrutiny, even if the ϵ is the same, i.e., denoting δ with $\delta_{\varepsilon,\boldsymbol{x}}$ in (B.4.1), $\delta_{\varepsilon,\boldsymbol{x}}$ may be different from $\delta_{\varepsilon,\boldsymbol{z}}$.
If $\boldsymbol{f}$ is continuous at all points of $R \subseteq S$, we say that $\boldsymbol{f}$ is *continous on (the set) R*. If for each ϵ it is possible to choose δ_ε independently of $x \in R$, we say that $\boldsymbol{f}$ is *uniformly continuous* on R.

B.4.2. Theorem (characterisation of continuity at a point via sequences). *A function $\boldsymbol{f} : S \to \mathbb{K}^d$ is continous at $x \in S$ if and only if for each sequence $(x_k)_{k \in \mathbb{N}}$*

$$(x_k \to x) \Rightarrow \left(\boldsymbol{f}(x_k) \to \boldsymbol{f}(x) \right). \tag{B.4.2}$$

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. □

B.4.3. Theorem (differentiable implies continuous). *If a function $\boldsymbol{f} : \Omega \overset{open}{\subseteq} \mathbb{K}^k \to \mathbb{K}^d$ is differentiable at a point $\boldsymbol{x} \in \Omega$, then $\boldsymbol{f}$ is continuous at that point.*
Thus, if $\boldsymbol{f}$ is differentiable (at all the points) in Ω, then $\boldsymbol{f}$ is continuous (at each point) on Ω.
Proof This is a standard result in calculus. □

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 $\boldsymbol{f}, \boldsymbol{g} : S \to \mathbb{K}^d$ are continuous and $\lambda, \mu \in \mathbb{K}$, then the linear combination function $\lambda \boldsymbol{f} + \mu \boldsymbol{g}$ 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 $\boldsymbol{f}$ and $\boldsymbol{g}$ are composable functions (i.e.,* $\operatorname{Cod} \boldsymbol{f} \subseteq \operatorname{Dom} \boldsymbol{g}$*). If $\boldsymbol{f}$ is continuous on a set S and $\boldsymbol{g}$ is continuous on* $\operatorname{Dom} \boldsymbol{g}$*, then $S \ni \boldsymbol{x} \mapsto \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}))$ 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 $X \in \mathbb{K}^d$ is called a* closed set*, if and only if for each sequence $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ and $\boldsymbol{x} \in \mathbb{K}^d$*

$$\big((x_k \in X \; \forall \, k \in \mathbb{N}) \text{ and } x_k \to x \text{ in } \mathbb{K}^d\big) \Rightarrow x \in X. \tag{B.4.3}$$

A set $X \in \mathbb{K}^d$ is called bounded set*, if and only if there exists a number $r > 0$ such that*

$$\boldsymbol{x} \in X \Rightarrow |\boldsymbol{x}| < r. \tag{B.4.4}$$

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 $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ such that $\{\boldsymbol{x}_k\}_{k\in\mathbb{N}} \subseteq K$ has a subsequence $\left(\boldsymbol{x}_{k_j}\right)_{j\in\mathbb{N}}$ ($j \mapsto k_j$ strictly increasing) such that*

$$\lim_{j\to\infty} \boldsymbol{x}_{k_j} \text{ exists and belongs to } K. \tag{B.4.5}$$

B.4.9. Theorem (Weierstrass compactness). *If a function $\boldsymbol{f} : S \subseteq \mathbb{K}^k \to \mathbb{K}^d$ is continuous, then for each compact set $K \subseteq S$ the image $\boldsymbol{f}(K)$ is a compact set. In particular, $\boldsymbol{f}$ is bounded on K, i.e.,*

$$\exists\, C > 0 : \forall\, x \in K : \left|\boldsymbol{f}(\boldsymbol{x})\right| < K, \tag{B.4.6}$$

and there exists a point $\boldsymbol{x}_* \in K$ such that

$$\boldsymbol{f}(\boldsymbol{x}_*) = \sup_{x\in K} \left|\boldsymbol{f}(\boldsymbol{x})\right|. \tag{B.4.7}$$

B.4.10. Theorem (uniform limit). *Let Ω be an open set and $f_n : \Omega \to \mathbb{R}$ a continuous function for each $n \in \mathbb{N}$. Suppose the sequence $(f_n)_{n\in\mathbb{N}}$ converges uniformly to f, i.e.,*

$$\left\|f - f_n\right\|_{\mathrm{L}_\infty(\Omega)} := \sup_{x\in\Omega}\left|f(x) - f_n(x)\right| \to 0 \text{ as } n \to \infty, \tag{B.4.8}$$

then the limit f is continuous.

B.4.11. Problem. *Show by producing a counterexample that the uniform limit theorem B.4.10 is invalid if "uniform" is replaced with "pointwise". Namely, construct a sequence of continuous functions such that $f_n \to f$ pointwisely with f discontinuous.*
Hint. $\sin(nx)$ *for* $x \in (-1, 1)$

B.4.12. Problem. *Suppose $(f_n)_{n\in\mathbb{N}}$ converges uniformy to f, show that for some given $N \in \mathbb{N}$, all the functions f_n for $n \geq N$, are each uniformly continuous.*

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 and Hölder 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 $f : D \to \mathbb{R}$, we say that f is *Lipschitz-continuous* on D if and only if

$$\sup_{\substack{x,y\in D\\ x\neq y}} \frac{\left|f(x) - f(y)\right|}{\left|x - y\right|} =: \left|f\right|_{\mathrm{Lip}\,1(D)} \in \mathbb{R}_0^+. \tag{B.6.1}$$

B.6.2. Proposition. *If f is Lipschitz continuous, then f is (sequentially) continuous, i.e.,*

$$x_n \to \hat{x} \Rightarrow f(x_n) \to f(\hat{x}). \tag{B.6.2}$$

B.6.3. Remark. The square root function

$$\begin{array}{rccc} \sqrt{\cdot}: & \mathbb{R}_0^+ & \to & \mathbb{R}_0^+ \\ & x & \mapsto & \sqrt{x} \end{array} \tag{B.6.3}$$

is continous 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

$$\frac{\left|\sqrt{x} - \sqrt{0}\right|}{|x - 0|} = \frac{1}{\sqrt{x}} \to \infty, \text{ as } x \to 0 \in \mathbb{R}_0^+. \tag{B.6.4}$$

Hence

$$\sup_{\substack{x,y\in\mathbb{R}_0^+\\ x\neq y}} \frac{\left|\sqrt{x} - \sqrt{y}\right|}{\left|x - y\right|} \tag{B.6.5}$$

B.6.4. Proposition (differentiability and Lipschitz continuity). *Suppose a function $f : D \to \mathbb{R}$ is differentiable on D. Then f is Lipschitz if and only if its derivative is bounded, i.e.,*

$$\left\|f'\right\|_{\mathrm{L}_\infty(D)} := \sup_{x\in D}\left|f'(x)\right| < \infty. \tag{B.6.6}$$

Furthermore the Lipschitz constant of f coincides with $\left\|f'\right\|_{\mathrm{L}_\infty(D)}$.

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 $[-1,1)$.[6]

B.6.6. Definition of Hölder continuous function. For a given real number $\alpha \in (0,1]$, a function $f : D \to \mathbb{R}$ is called *Hölder continuous of order α* or simply *α-Hölder continuous* if and only if

$$\sup_{\substack{x,y\in D\\ x\neq y}} \frac{\left|f(x)-f(y)\right|}{\left|x-y\right|^\alpha} =: \left|f\right|_{\mathrm{C}^{0,\alpha}(D)} \in \mathbb{R}_0^+. \tag{B.6.7}$$

The set of all α-Hölder continuous functions on D is denoted by

$$\mathrm{C}^{0,\alpha}(D) := \left\{f : D \to \mathbb{R} : \left|f\right|_{\mathrm{C}^{0,\alpha}(D)} \in \mathbb{R}_0^+\right\}. \tag{B.6.8}$$

B.6.7. Proposition. *If $f \in \mathrm{C}^{0,\alpha}(\Omega)$ then f is continuous. This justifies using "continuous" in "α-Hölder continuous".*
Proof Exercise. □

B.6.8. Exercise (Hölder space). *Show that $\mathrm{C}^{0,\alpha}(D)$ is a vector space, that the function*

$$\mathrm{C}^{0,\alpha}(D) \ni f \mapsto \left|f\right|_{\mathrm{C}^{0,\alpha}(D)} \tag{B.6.9}$$

is a seminorm on it, and the expression

$$\left\|f\right\|_{\mathrm{C}^{0,\alpha}(D)} := \left\|f\right\|_{\mathrm{L}_\infty(D)} + \left|f\right|_{\mathrm{C}^{0,\alpha}(D)} \tag{B.6.10}$$

defines a (full) norm on $\mathrm{C}^{0,\alpha}(D)\cap \mathrm{L}_\infty(\Omega)$.

B.6.9. Remark (What about 0-Hölder?) It is possible to extend the definition of α-Hölder continuity to the case $\alpha = 0$. As an exercise one can check that an 0-Hölder continuous function f is a bounded function but not necessarily continuous and in that case the seminorm $\left|f\right|_{\mathrm{C}^{0,0}(D)}$ measures the maximal oscillation of a function f.

B.6.10. Remark (1-Hölder is same as Lipschitz). This follows immediately from the definition.

B.6.11. Proposition (Hölder space hierarchy). *The higher the Hölder exponent, the smaller the space*

$$0 < \alpha \leq \beta \leq 1 \Rightarrow \mathrm{C}^{0,\beta}(D) \subseteq \mathrm{C}^{0,\alpha}(D). \tag{B.6.11}$$

[6]Note 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.

B.6.12. Example. In general, the Hölder spaces are strictly different

$$0 < \alpha < \beta \leq 1 \Rightarrow \mathrm{C}^{0,\beta}(D) \subsetneq \mathrm{C}^{0,\alpha}(D). \tag{B.6.12}$$

For example, if $D = \mathrm{B}_{\rho}(\mathbf{0})$, the function $\boldsymbol{x} \mapsto |\boldsymbol{x}|^{\alpha}$ belongs to $\mathrm{C}^0\,\alpha(D)$ but not to $\mathrm{C}^0\,\beta$ for any $\beta > \alpha$.

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.[7]

B.7.1. Definition of contraction mapping. Let X be a Banach space and $Y \subseteq X$ where Y is closed. A mapping $\mathscr{T} : Y \to Y$ is called a *contraction* on Y if there exists $\kappa \in \mathbb{R}$ such that

$$0 \leq \kappa < 1 \text{ and } \forall\, x, y \in Y : \left\|\mathscr{T} x - \mathscr{T} y\right\| \leq \kappa \left\|x - y\right\|. \tag{B.7.1}$$

B.7.2. Remark. A contraction mapping on Y reduces the distance between points of Y, whence the name.

B.7.3. Remark (continuity of contractions). Note that a contraction $\mathscr{T} : Y \to Y$, $Y \subseteq X$ 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

$$\|\mathscr{T} x - \mathscr{T} x_0\| \leq \kappa \|x - x_0\| < \varepsilon. \tag{B.7.2}$$

Continuity of $\mathscr{T}$ is crucial for the next result's validity.[8]

B.7.4. Theorem (Banach–Caccioppoli contraction principle). *Let X be a Banach space and Y a closed subset (bu not necessarily a subspace) of X. If $\mathscr{T} : Y \to Y$ is a contraction mapping, then $\mathscr{T}$ has a unique fixed point, namely there exists exactly one $x_* \in Y$ such that*

$$x_* = \mathscr{T} x_*. \tag{B.7.3}$$

Furthermore, x_* can be approximated using the fixed-point iteration starting with any $x_0 \in Y$ and building the sequence $(x_k)_{k \in \mathbb{N}_0}$

$$x_k := \mathscr{T} x_{k-1}, \textit{ for } k \in \mathbb{N}. \tag{B.7.4}$$

Then this sequence converges and

$$\lim_{k \to \infty} x_k = x_*. \tag{B.7.5}$$

[7]The theory described here works also in *complete metric spaces*, which are more general structures than Banach spaces, but we will not need them here.

[8]In 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.

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

$$\begin{aligned}\left\|x_k - x_{k+j}\right\| &= \left\|x_k - x_{k+1} + x_{k+1} + \cdots - x_{k+j-1} + x_{k+j-1} - x_{k+j}\right\| \\ &\leq \sum_{i=1}^{j} \|x_{k+i-1} - x_{k+i}\|.\end{aligned} \tag{B.7.6}$$

Furthermore for each $l \geq 1$ we have

$$\|x_{l-1} - x_l\| = \|\mathscr{T} x_{l-2} - \mathscr{T} x_{l-1}\| \leq \kappa \|x_{l-2} - x_{l-1}\| \leq \cdots \leq \kappa^{l-1} \|x_0 - x_1\|, \tag{B.7.7}$$

and thus, defining $d_0 := \|x_0 - x_1\|$,

$$\left\|x_k - x_{k+j}\right\| \leq d_0 \sum_{i=1}^{j} \kappa^{k+i-1} = d_0 \kappa^k \sum_{i=0}^{j-1} \kappa^i \leq \frac{d_0}{1-\kappa} \kappa^k. \tag{B.7.8}$$

But $\kappa^k \to 0$ as $k \to \infty$ because $0 \leq \kappa < 1$. It follows that for each $\varepsilon > 0$ there exists $k = k_\varepsilon \in \mathbb{N}$ such that

$$\left\|x_k - x_{k+j}\right\| < \varepsilon \quad \forall j > 0. \tag{B.7.9}$$

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 $x_k \to x_*$ as $k \to \infty$. 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

$$x_* = \lim_{k\to\infty} x_k = \lim_{k\to\infty} \mathscr{T} x_{k-1} = \mathscr{T} \lim_{k\to\infty} x_{k-1} = \mathscr{T} x_*, \tag{B.7.10}$$

where the second last step is due to the continuity of $\mathscr{T}$. Thus we have proved that $\mathscr{T}$ has a fixed point x_*.
To close, we need to acertain the uniqueness of x_*. Suppose another point $x_{**} \in Y$ is also a fixed point, it follows that

$$0 \leq \|x_* - x_{**}\| = \|\mathscr{T} x_* - \mathscr{T} x_{**}\| \leq \kappa \|x_* - x_{**}\|. \tag{B.7.11}$$

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. Landau's big O and small o notation

This section contains some analysis theory and definitions to explain the use of the symbol $\mathrm{O}(h^p)$, with particular focus on its use in the analysis of numerical methods for differential equations.

B.8.1. Big O. $\mathrm{O}(x)$ is read as "Big O of x" or "order x," and similarly $\mathrm{O}(x^p)$ is "Big O of x^p" or "order x^p". We will always take p to be a positive integer. This notation is commonly known as the *Landau* O *notation* in honour of the German mathematician Edmund Landau who first introduced it.
Most text books either carefully avoid using this notation or use it without explanation. The reason for this is that it has two nearly but not quite equivalent uses. Applied mathematicians tend to switch between these uses when it suits them. This handout sets out these two uses and the relationship between them.

B.8.2. Definition of the rigourous Landau O **notation.** Let $g : \mathbb{R} \to \mathbb{R}$. If there exists a constant C and an $\epsilon > 0$ such that

$$|g(x)| \leq C|x|^p, \qquad \forall |x| < \epsilon, \tag{B.8.1}$$

then we say that

$$g(x) = \mathrm{O}(x^p). \tag{B.8.2}$$

Notice that the equality in (B.8.2) is a slight abuse of notation, as $\mathrm{O}(x^p)$ is not a function, but rather a class of functions. Strictly speaking, we should write $g \in \mathrm{O}(\cdot^p)$ (or $g(x) \equiv \mathrm{O}(x^p)$), but the equality sign is almost universally used and we comply with this use.

B.8.3. The Formal Landau O **Notation.** There is a second nearly (but not quite) equivalent usage of $\mathrm{O}(x^p)$ to that in Definition B.8.2.
Suppose that $g(0) = 0$ and that $g(x)$ is given by the series

$$g(x) = \sum_{j=0}^{\infty} a_j x^j, \tag{B.8.3}$$

where $a_0 = a_1 = \ldots = a_{p-1} = 0$, i.e., in complex analytic terminology g has a zero of order p.
In this case, we also say that $g(x) = \mathrm{O}(x^p)$ since

$$g(x) = \sum_{j=0}^{\infty} a_j x^j = \sum_{j=p}^{\infty} a_j x^j = x^p \sum_{j=0}^{\infty} a_{j+p} x^j. \tag{B.8.4}$$

This second usage is known as the *formal Landau* O *notation.* The next result relate these two uses of $\mathrm{O}(x^p)$.

B.8.4. Theorem (equivalence of rigorous and formal O **notation for analytic functions).** *Suppose that* $g(x)$ *is infinitely many times differentiable at* 0 *and that its Taylor series has radius of convergence* $R > 0$*. Then* $g(x) = \mathrm{O}(x^p)$ *following definition* (B.8.1) *if and only if* $g(x) = \mathrm{O}(x^p)$ *following definition (B.8.4).*
Proof We can write down a Taylor series expansion for $g(x)$:

$$g(x) = \sum_{j=0}^{\infty} \frac{1}{j!} x^j g^{(j)}(0). \tag{B.8.5}$$

From (B.8.1), $g(0) = 0$ and so

$$g(x) = \sum_{j=1}^{\infty} \frac{1}{j!} x^j g^{(j)}(0). \tag{B.8.6}$$

Suppose that $p \geq 2$. We have that

$$g'(0) = \lim_{x \to 0} \frac{g(x) - g(0)}{x} = \lim_{x \to 0} \frac{g(x)}{x}. \tag{B.8.7}$$

By (B.8.1),

$$\left| \frac{g(x)}{x} \right| \leq C|x|^{p-1} = C|x|^{p-2}|x| \leq C\varepsilon^{p-2}|x| \quad \forall |x| < \varepsilon. \tag{B.8.8}$$

Hence

$$\lim_{x \to 0} \left| \frac{g(x)}{x} \right| = 0 \quad \Rightarrow \quad g'(0) = \lim_{x \to 0} \frac{g(x)}{x} = 0. \tag{B.8.9}$$

We can go on to show that

$$g^{(j)}(0)=0 \quad \forall j=0,1,\dots,p-1. \tag{B.8.10}$$

It follows that

$$g(x)=\sum_{j=p}^{\infty}\frac{1}{j!}x^j g^{(j)}(0), \tag{B.8.11}$$

and so $g(x)=\mathrm{O}(x^p)$ by the usage in (B.8.4).
Suppose that $g(x)=\mathrm{O}(x^p)$ by the usage in (B.8.4), and that we can find $\epsilon>0$ and a constant C such that

$$\sum_{j=0}^{\infty}a_{j+p}x^j$$

is absolutely convergent for all $|x|<\epsilon$, with

$$\sum_{j=0}^{\infty}|a_{j+p}x^j|\le C \quad \forall |x|<\epsilon. \tag{B.8.12}$$

It follows from (B.8.4) that

$$|g(x)|\le C|x|^p \quad \forall |x|<\epsilon.$$

Hence $g(x)=\mathrm{O}(x^p)$ by the usage in (B.8.1).
It follows that if $g(x)$ is infinitely differentiable at 0 and there exist $\epsilon>0$ and a constant C such that

$$\sum_{j=0}^{\infty}\left|\frac{1}{(j+p)!}x^j g^{(j+p)}(0)\right|\le C \quad \forall |x|<\epsilon, \tag{B.8.13}$$

then $g(x)=\mathrm{O}(x^p)$ by the usage in (B.8.1) if, and only if, $g(x)=\mathrm{O}(x^p)$ by the usage in (B.8.4). □
A Theorem, originally due to Weierstrass, states that if

$$|g^{(j)}(0)|\le M, \quad \forall j\ge 0, \tag{B.8.14}$$

and thus in particular if g is a polynomial in x, then this series has infinite radius of convergence. It can be shown to have radius of convergence $R>0$ under much weaker conditions on the derivatives than (B.8.14). More on analytic functions and radiuses of convergence can be found in Ahlfors, 1978.
For functions $g:\mathbb{R}\to\mathbb{R}$ which are analytic, the two uses of $\mathrm{O}(x^p)$ are always equivalent. The reader should beware that for functions that are not analytic, definitions (B.8.1) and (B.8.4) are not always equivalent.

B.8.5. Example (smooth function which is not analytic). Consider the following function $f:\mathbb{R}\to\mathbb{R}$:

$$f(x)=\begin{cases}\exp(-1/x) & \text{for } x>0,\\ 0 & \text{otherwise.}\end{cases} \tag{B.8.15}$$

It can be shown by induction that for $j\in\mathbb{N}_0$,

$$f^{(j)}(x)=\begin{cases}q_j(1/x)\exp(-1/x) & \text{for } x>0,\\ 0 & \text{for } x<0;\end{cases} \tag{B.8.16}$$

where $q_j(\xi)$ is a polynomial in ξ recursively defined via

$$q_0 \equiv 1, \qquad q_j(\xi) = \xi^2(q_{j-1}(\xi) - q'_{j-1}(\xi)) \quad \forall j \in \mathbb{N}, \tag{B.8.17}$$

It follows that for each $j \in \mathbb{N}_0$, $f^{(j)}(x) \in \mathrm{C}^0(\mathbb{R})$. Hence $f \in \mathrm{C}^\infty(\mathbb{R})$.
To see that f is not analytic at 0, we argue by contradiction. Since $f^{(j)}(0) = 0$ for all $j \in \mathbb{N}_0$, the Taylor series of f about 0 is identically 0. If f is analytic at 0, then there exists a neighbourhood of 0 in which the function f equals this Taylor series, i.e. there exists $R > 0$ such that

$$f(x) = 0 \quad \forall |x| < R, \tag{B.8.18}$$

which is false.

Exercises and problems on Analysis basics

Exercise B.1. Let $(a_n)_{n\in\mathbb{N}}$ be a sequence of positive terms that converges to 0 but whose sum diverges, e.g., $a_n = {}^1\!/\!_n$, i.e.,

$$\lim_{n\to\infty} a_n = 0 \text{ and } \sum_{n\in\mathbb{N}} a_n = \infty. \tag{PB.1.1}$$

(1) Show that for any $x \in \mathbb{R}_0^+$ there exists a subsequence $(a_n)_{n\in K}$ of $(a_n)_{n\in\mathbb{N}}$, $K \subseteq \mathbb{N}$, such that

$$x = \sum_{n\in K} a_n. \tag{PB.1.2}$$

Note that K may be finite for some x, but if $x > 0$ then it is possible to find it infinite.

(2) Show that for a $\mathcal{M}$-measurable function $f : \Omega \to \mathbb{R}_0^+$ there exists a sequence of $\mathcal{M}$-measurable sets Ω_n, such that

$$f(x) = \sum_{n=1}^{\infty} a_n \mathbb{1}_{\Omega_n}(x), \text{ for } \mathcal{M}\text{-almost all } x \in \Omega. \tag{PB.1.3}$$

In particular, for $a_n = {}^1\!/\!_n$, any $\mathcal{M}$-measurable function f can be written as a series of scaled characteristic functions

$$f = \sum_{n=1}^{\infty} \frac{1}{n} \mathbb{1}_{\Omega_n}. \tag{PB.1.4}$$

Exercise B.2. Let $|\cdot|_m$ and $|\cdot|_n$ be norms on $\mathbb{K}^m$ and $\mathbb{K}^n$, respectively, and let $\boldsymbol{A} \in \mathbb{K}^{m\times n}$. Show that the induced matrix norm

$$|\boldsymbol{A}|_{mn} := \sup_{\substack{\boldsymbol{x}\in\mathbb{K}^n,\\ \boldsymbol{x}\neq 0}} \frac{|\boldsymbol{A}\boldsymbol{x}|_m}{|\boldsymbol{x}|_n} = \sup_{\substack{\boldsymbol{x}\in\mathbb{K}^n,\\ |\boldsymbol{x}|_n=1}} |\boldsymbol{A}\boldsymbol{x}|_m \tag{PB.2.1}$$

satisfies the properties of a norm.
Hint. Exploit the norm properties of $|\cdot|_m$.

Exercise B.3. For the matrix (zero entries omitted)

$$\boldsymbol{A} := \begin{bmatrix} {}^3\!/\!_2 & & {}^1\!/\!_2 \\ & 3 & \\ {}^1\!/\!_2 & & {}^3\!/\!_2 \end{bmatrix}, \tag{PB.3.1}$$

compute the induced matrix p-norms for $p \in \{1, 2, \infty\}$ and the Frobenius norm. Compute also the spectral radius of A.

Exercise B.4 (Cauchy–Bunyakovsky–Schwarz inequality inequality)**.** Suppose a vector space $\mathcal{V}$ has an inner product $\langle\cdot,\cdot\rangle$, show that

$$\langle x, y\rangle^2 \leq \langle x, x\rangle \langle y, y\rangle \quad \forall\, x, y \in \mathcal{V}. \tag{PB.4.1}$$

Exercise B.5. Let $\mathcal{V}$ be a vector space.

(1) Suppose $\langle\cdot,\cdot\rangle$ is an inner product on $\mathcal{V}$ and let

$$\|x\| := \sqrt{\langle x, x\rangle} \text{ for all } x \in \mathcal{V}. \tag{PB.5.1}$$

Show that the *parallelogram identity* (also known as *parallelogram law*) holds

$$2\left(\|x\|^2 + \|y\|^2\right) = \|x+y\|^2 + \|x-y\|^2. \tag{PB.5.2}$$

(2) Under the same conditions as (1) show the *Pythagoras identity* (also known as *Pythagoras theorem*)

$$\langle x, y\rangle = 0 \Rightarrow \|x\|^2 + \|y\|^2 = \|x \pm y\|^2. \qquad \text{(PB.5.3)}$$

(3)

Problem B.6 (Fréchet differentiability implies continuity)**.** Let $\boldsymbol{f} : D \to \mathbb{R}^k$ be a function that is Fréchet differentiable at a given point $\boldsymbol{x} \in D$. Show that $\boldsymbol{f}$ is continuous at $\boldsymbol{x}$.

Exercise B.7 (Gâteaux is weaker than Fréchet)**.** Show that the function

$$g(x, y) := \begin{cases} 0 & \text{if } x = 0 \text{ and } y = 0 \\ \frac{x^4 y}{x^6 + y^3} & \text{otherwise.} \end{cases} \qquad \text{(PB.7.1)}$$

has a Gâteaux derivative at $(0,0)$. Show that $\boldsymbol{g}$ is discontinuous at $(0,0)$. Deduce it has no Fréchet derivative thereon.

Problem B.8 (derivative of the inverse matrix function)**.** Denote by $\mathrm{GL}(\mathbb{R}^n)$ the set of all invertible matrices in $\mathbb{R}^{n\times n}$ and by inv the inverse matrix function

$$\begin{array}{rccc} \mathrm{inv}: & \mathrm{GL}(\mathbb{R}^n) & \to & \mathrm{GL}(\mathbb{R}^n) \\ & \boldsymbol{X} & \mapsto & \boldsymbol{X}^{-1} \end{array}. \qquad \text{(PB.8.1)}$$

Show that for any matrix $\boldsymbol{Z} \in \mathbb{R}^{n\times n}$ sufficiently small ϵ the Neumann series

$$\mathbf{I} - \epsilon \boldsymbol{Z} + \epsilon^2 \boldsymbol{Z}^2 - \ldots = \sum_{k=0}^{\infty} (-\epsilon \boldsymbol{Z})^k \qquad \text{(PB.8.2)}$$

converges absolutely, the matrix $\mathbf{I} + \epsilon \boldsymbol{Z}$ is invertible, and

$$(\mathbf{I} + \epsilon \boldsymbol{Z})^{-1} = \sum_{k=0}^{\infty} (-\epsilon \boldsymbol{Z})^k. \qquad \text{(PB.8.3)}$$

Use this result to show the existence and compute the directional derivative of inv at a point $\boldsymbol{X} \in \mathrm{GL}(\mathbb{R}^n)$ in the direction $\boldsymbol{Y} \in \mathbb{R}^{n\times n}$. Check that your result agrees with what you know from basic calculus when $n = 1$.

Problem B.9. Let $\boldsymbol{v} \in \mathrm{C}^{k,\alpha}(\Omega)$ for some $k \in \mathbb{N}_0$ and $0 \le \alpha \le 1$ and define Taylor's remainder of v at $\boldsymbol{x}$ of order k as

$$\mathrm{R}^{(k)}_{\boldsymbol{v},\boldsymbol{x}}(\boldsymbol{h}) := \boldsymbol{v}(\boldsymbol{x} + \boldsymbol{h}) - \sum_{j=0}^{k} \frac{1}{j!} \mathrm{D}^j \boldsymbol{v}(\boldsymbol{x}) \boldsymbol{h} \cdots \boldsymbol{h} \qquad \text{(PB.9.1)}$$

for a fixed point $\boldsymbol{x} \in \Omega$ and all $\boldsymbol{h}$ such that $|\boldsymbol{h}| < \rho_{\boldsymbol{x}}$ (as a by way, make sure you argue why such a $\rho_{\boldsymbol{x}} > 0$ exists and think about why you need it). Prove that

$$\left|\mathrm{R}^{(k)}_{\boldsymbol{v},\boldsymbol{x}}(\boldsymbol{h})\right| \le C_{\mathrm{PB.9.2},\boldsymbol{v},\boldsymbol{x}} |\boldsymbol{h}|^{k+\alpha} \qquad \text{(PB.9.2)}$$

Assume only the fundamental theorem of calculus, and the definition of Hölder continuity, and work from first principles.

APPENDIX C

Polynomial interpolation

C.1. Polynomials and approximation of functions

Generally speaking a *polynomial* of degree n with coefficients $a_0, \dots, a_n \in \mathbb{K}$ is an algebraic expression of the form

$$p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n, \tag{C.1.1}$$

where x is called the "indeterminate". Note that x does not need to be restricted to numbers, it can be also a matrix, or any kind of object for which $\mathbb{K}$-scaling, addition and power operations make sense. If we do restrict x in to $\mathbb{R}$ we recover the usual high-school notion of *polynomial.* Then the polynomial acquires the meaning of a function $p : \mathbb{R} \to \mathbb{R}$.
Because polynomials are simple functions that are easily computed (sum, scaling and power of scalars have all immediate floating point realisation), they are the prime choice for conducting so-called "constructive approximation". So the main problem in this chapter:

> Given a function $f : D \to \mathbb{R}$ *numerically* find a polynomial p such that p approximates f.

The fact that our quest is not hopeless is backed by the following well-known approximation result.

C.1.1. Theorem (Weierstrass's polynomial approximation). *Let f be a continuous function on the interval $[a, b] \subseteq \mathbb{R}$ and given any $\epsilon > 0$, there exists a polynomial p (with sufficiently high degree n) such that*

$$\max_{x \in [a,b]} \left| f(x) - p(x) \right| < \epsilon. \tag{C.1.2}$$

C.1.2. Remark (Weierstrass is not very practical). We here omit Weierstrass Theorem's Proof, which can be found in many analysis books, for example, in Friedman, 1970. The problem with any known Proof of this result is that none is constructive. So there is no recipe for building the polynomial p.

C.1.3. Taylor's polynomial. If we are ready to do with generality for the sake of practicality, a useful polynomial approximation to a *sufficiently differentiable function f* (continuous is not enough), is given by the *Taylor polynomial* $T_n(x)$ of degree n. We also know from Taylor's theorem that the error between f and T_n is given by

$$f(x) - T_n(x) = \frac{(x - x_0)^{n+1}}{(n+1)!} f^{(n+1)}(\eta), \tag{C.1.3}$$

for some η lying between x and x_0, this is known as the *remainder term.*

C.1.4. Definition of Taylor's polynomial. Given a function f that is n-times differentiable with continuous derivatives. The Taylor polynomial interpolates f and its first n derivatives at a single point x_0.

$$\begin{aligned} T_{f,n}(x_0;x) := T_n(x) &:= \sum_{k=0}^{n} \frac{(x-x_0)^k}{k!} f^{(k)}(x_0) \\ &= f(x_0) + \frac{(x-x_0)}{1!} f'(x_0) + \cdots + \frac{(x-x_0)^n}{n!} f^{(n)}(x_0). \end{aligned} \tag{C.1.4}$$

C.1.5. Theorem (Taylor's polynomial approximation error). *Let $f \in \mathrm{C}^n[a,b]$ with $f^{(n+1)}$ defined and such that*

$$\max_{\eta\in[a,b]} \left|f^{(n+1)}(\eta)\right| \le M, \tag{C.1.5}$$

then for each $x \in [a,b]$

$$\left|f(x) - T_n(x)\right| \le \frac{M|x-x_0|^{n+1}}{(n+1)!}. \tag{C.1.6}$$

C.1.6. Remark (Taylor's good news). Hence, if x is close to x_0 we get a good approximation.

C.1.7. Remark (Taylor's bad news). The major pitfalls of this approximation are:

(1) As x moves away from x_0, $|x-x_0|^{n+1}$ becomes larger. For $|x-x_0| > 1$, the error thus becomes larger if n is larger; a large n should on the contrary mean that the approximation becomes better.
(2) The accuracy depends on the smoothness of the function and the amount of derivatives taken, many continuous functions do not have first derivatives everywhere.

C.1.8. Example. Consider $f(x) = \sin x$ and $x_0 = 0$. Then

$$p_5(x) = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5. \tag{C.1.7}$$

In Figure 1 we compare the sine-function with p_1, p_3 and p_5.

C.2. Interpolation

Polynomial interpolation is and alternative, but somewhat related, method to Taylor's polynomial. Instead of looking at approximating the function and via its value and its derivatives at one point, we approximate the function value at distinct points.

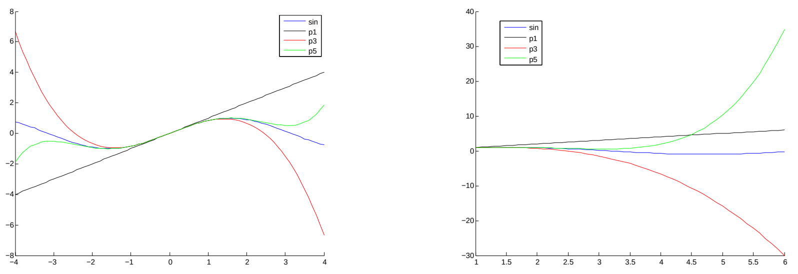


FIGURE 1. The figure shows $\sin x$ and three of its Taylor approximations p_1, p_3, p_5. Left: $x \in [-4,4]$, right: $x \in [1,6]$.

C.2.1. Example (linear Lagrange interpolation). To illustrate the idea behind interpolation, consider the simple situation where a twice differentiable function f defined on an interval $[a,b]$, with $h = b-a$, is approximated by its Taylor polynomial at $x_0 := (b+a)/2$. This requires access to $f(x_0)$ and $f'(x_0)$ and provides an approximation error of

$$\begin{aligned} \left|f(x) - T_1(x)\right| &\leq \frac{1}{2}\sup_{[a,b]}\left|f''\right| |x-x_0|^2 \\ &\leq C[f]h^2 \\ \text{with } C[f] &:= \frac{1}{8}\left\|f''\right\|_{\mathrm{L}_\infty(a,b)}. \end{aligned} \tag{C.2.1}$$

Alternatively, we may use $f(a)$ and $f(b)$ as data instead and consider the line passing through $(a, f(a))$ and $(b, f(b))$ as an approximation to f. This line is the graph of the function

$$x \longmapsto p(x) := \frac{f(b)-f(a)}{h}(x-a) + f(a) = f(a)l_a(x) + f(b)l_b(x), \tag{C.2.2}$$

where

$$l_a(x) := \frac{x-b}{a-b} = \frac{b-x}{h} \text{ and } l_b(x) := \frac{x-a}{b-a} = \frac{x-a}{h}. \tag{C.2.3}$$

By the mean value theorem we know that for a $\xi \in (a,b)$ we have

$$\begin{aligned} \left|f(x) - p(x)\right| &= \left|f(x) - f(a) - f'(\xi)(x-a)\right| \\ \text{(mean value implies } \eta \text{ exists in } (a,x)) \quad &= \left|f'(\eta)(x-a) - f'(\xi)(x-a)\right| \\ &= \left|f'(\eta) - f'(\xi)\right| |x-a| \\ \text{(mean value again applied to } f') \quad &= \left|f''(\zeta)\right| \left|\eta - \xi\right| |x-a| \\ &\leq C'[f]h^2 \\ \text{where } C'[f] &:= \left\|f''\right\|_{\mathrm{L}_\infty[a,b]}. \end{aligned} \tag{C.2.4}$$

We conclude that in terms of h both approximations have the *same order*.

C.2.2. Lagrange polynomials. Suppose we are given the values of a continuous function, $f(x)$, at $n+1$ pairwise distinct points, $x_0, \ldots, x_n$. We now wish to construct a polynomial p_n, such that,

$$\deg p = n \text{ and } p(x_i) = f(x_i) \quad \forall\, i = 0, \ldots, n. \tag{C.2.5}$$

This procedure is called *interpolation.*
Assuming

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n, \tag{C.2.6}$$

with unknown $(a_i)_{i\in[i\ldots 0]n}$, we may derive the following linear system, known as Vandermonde's equation,

$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= f(x_0) \\ &\vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= f(x_n) \end{aligned} \tag{C.2.7}$$

We may write this in a matrix-vector form:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}. \tag{C.2.8}$$

The matrix appearing in (C.2.8) is known as the *Vandermonde matrix.* If the set of interpolation points x_i consists of pairwise distinct points, then the matrix is non-singular (exercise sheet 13) and thus the problem has a unique solution.

C.2.3. Theorem (interpolant's uniqueness). *Suppose we are given the value of a continuous function, f, at $n+1$ pairwise distinct points, $x_0, \ldots, x_n$. Then there is one and only one polynomial p_n that satisfies*

$$\deg p_n \le n \text{ and } p_n(x_i) = f(x_i), \quad \forall\, i = 0, \ldots, n. \tag{C.2.9}$$

Proof This is a direct application of the polynomial identity principle (PIP) explained in Appendix §A.4.5. Suppose there is another polynomial, say q_n, such that

$$\deg q_n \le n \text{ and } q_n(x_i) = f(x_i), \quad \forall\, i = 0, \ldots, n. \tag{C.2.10}$$

Then the polynomial $p_n - q_n$ has degree no more than n and $n+1$ distinct roots (the points x_i, $i = 0, \ldots, n$). By PIP we conclude that $p_n - q_n = 0$, which means uniqueness. $\square$

We can now calculate the coefficients $a_0, \ldots, a_n$ and thus obtaining the interpolating polynomial p by solving the above system of linear equation. We will study three ways of calculating the same unique interpolation polynomial:

1. Solve the system of linear equations
2. Lagrange basis polynomials
3. Newton's divided differences

C.2.4. Solve the Vandermonde linear system (C.2.8). This can be implemented as follows.

Printout of file `Code/lagrange1.m`

```
function [p,y] = lagrange1(x,f,a,b)
%% function [p,y] = lagrange1(x,f,a,b)
%% Lagrange interpolation polynomial for the data set f(x)
%%  x, f are column vectors of the same length
%% polynomial is evaluated at points in the region [a,b]
N = length(x);
% construct Vandermonde matrix
V = zeros(N,N);
for i = 1:N
  for j = 1:N
    V(i,j) = x(i)^(j-1);
  end
end
% obtain the polynomial coefficients
coeff = V\f;
% polynomial evaluation points.
y = [a:0.01*(b-a):b];
% evaluate the polynomial at the data points
p = zeros(1,101);
for i = 0:N-1
  p = p + coeff(i+1)*y.^i;
end
return
```

C.2.5. Example. To interpolate the data set $f(0) = 1$, $f(1) = 5$, $f(1.5) = 2$, $f(3) = 6$, $f(4.5) = -1$, we consider the following driver code

Printout of file `lagrange1Driver1.m`

```
function [] = lagrange1Driver1()

x = [0; 1; 1.5; 3; 4.5];
f = [1; 5; 2; 6; -1];
hold on
plot(x,f,' s','MarkerSize',10)
[p,y] = lagrange1(x,f,-0.1,4.6);
plot(y,p,'k')
```

and later from -1 to 5, cf. Figure 2.

C.2.6. Definition of Lagrange basis. Instead of solving the system of linear equations, we can also compute p using the Lagrange basis polynomial.
For the pairwise different points $x_0,\dots,x_n$ we define the *Lagrange basis polynomials* $l_j(x)$, for $j = 0,\dots,n$, as the (unique) polynomials of maximal degree n such that

$$l_j(x_k) = \delta_k^j. \tag{C.2.11}$$

C.2.7. Remark (interpolation property of Lagrange polynomials). The Lagrange basis polynomial l_j, $j = 0,\dots,n$, is the solution of the interpolation problem with the function f given by $f(x_k) = \delta_{jk}$.

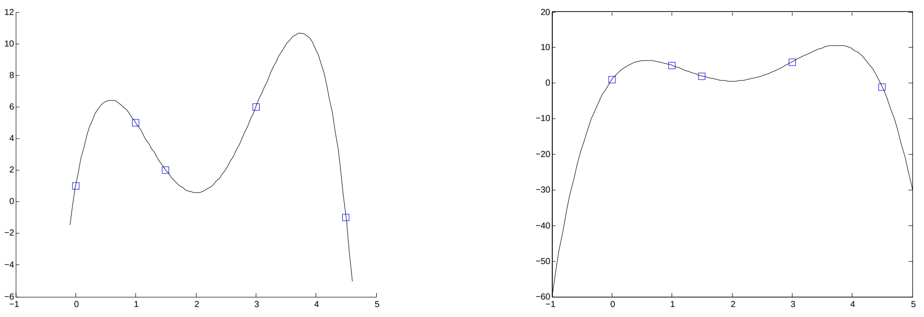


FIGURE 2. The figures show the polynomial interpolation of the five data points: for x-values outside the interval of the data points, the values of the polynomial are large.

C.2.8. Definition of Lagrange basis. The Lagrange basis polynomials (also known as Lagrange polynomials) are given by

$$l_j(x) = \prod_{k \neq j} \frac{(x - x_k)}{(x_j - x_k)}. \tag{C.2.12}$$

By the interpolant's uniqueness (theorem C.2.3), the equations (C.2.12) define uniquely the polynomial l_j.

C.2.9. Theorem (Lagrange interpolation). *Given a set of nodes x_i, for $i = 0, \dots, n$, and a correspondingly labelled set of intepolation values f^i, for $i = 0, \dots, n$, the polynomial p, such that*

$$\deg p \leq n \text{ and } p(x_i) = f^i \quad \forall\, i = 0, \dots, n \tag{C.2.13}$$

is given by

$$p(x) = \sum_{j=0}^{n} f^j l_j(x). \tag{C.2.14}$$

Proof First consider the formula for the Lagrange basis polynomials. For each j, the polynomial l_j defined in C.2.6 has degree n and satisfies $l_j(x_i) = \delta_i^j$. This means that $l_0, \dots, l_n$ forms a basis of $\mathbb{P}^n$ (and hence the name Lagrange basis is justified). From the $\deg p \leq n$ requirement, it follows that $p \in \mathbb{P}^n$ and hence

$$p(x) = \sum_{i=0}^{n} c_i l_i(x) \text{ for some } \boldsymbol{c} = (c_0, \dots, c_n) \in \mathbb{R}^{n+1} \tag{C.2.15}$$

By evaluating p at each of the points $x_0, \dots, x_n$ we obtain

$$f^j = \sum_{i=0}^{n} c_i \delta_i^j = c_j. \tag{C.2.16}$$

And thus the polynomial $p := \sum_{j=0}^{n} f^j l_j$ is the interpolating polynomial. ☐

C.2.10. Remark. Strictly speaking statement of theorem C.2.9 does not involve any function f, except for its values at the nodes x_i, for $i = 0, \dots, n,$. In fact there needs not be a function f to find the interpolating polynomial p which is why the theorem is stated without assuming. However, in most practical applications one is after an approximation of a function f of which the values are known only at the nodes x_i, for $i = 0, \dots, n$, and one puts

$$f(x_j) =: f^j. \tag{C.2.17}$$

C.2.11. Example. We calculate the Langrage interpolation polynomial for the data $f(1) = 3$, $f(2) = 2$, $f(4) = 1$. The Lagrange basis polynomials are given by

$$\begin{aligned} l_0(x) &= \frac{(x-2)(x-4)}{(1-2)(1-4)} = \frac{x^2-6x+8}{3}, \\ l_1(x) &= \frac{(x-1)(x-4)}{(2-1)(2-4)} = -\frac{x^2-5x+4}{2}, \\ l_2(x) &= \frac{(x-1)(x-2)}{(4-1)(4-2)} = \frac{x^2-3x+2}{6}. \end{aligned} \tag{C.2.18}$$

Note that they do not depend on the f-values, but only on the x-values of the data. Then the interpolating polynomial is given by

$$\begin{aligned} p(x) &= 3l_0(x) + 2l_1(x) + l_2(x) \\ &= x^2 - 6x + 8 - (x^2 - 5x + 4) + \frac{1}{6}(x^2 - 3x + 2) \\ &= \frac{1}{6}x^2 - \frac{3}{2}x + \frac{13}{3}. \end{aligned} \tag{C.2.19}$$

We calculate $p(3/2) = 59/24$.

C.2.12. Theorem (Lagrange interpolation error estimate). *Suppose a function f is $n+1$ times continuously differentiable on the interval $[a,b]$ which contains the pairwise different interpolation points $x_0, x_1, x_2, \dots, x_n$. Let $p(x)$ be the polynomial interpolant with $p(x_i) = f(x_i)$ for $i = 0, \dots, n$. Then, for any $x \in [a,b]$,*

$$f(x) - p(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!} f^{(n+1)}(\xi), \tag{C.2.20}$$

for some $\xi \in [a,b]$.
Proof Denote

$$L(x) = \Pi_{k=0}^{n}(x - x_k).$$

Fix $x \in [a,b]$ with $x \neq x_i$ for all $i = 0, \dots, n$. (For $x = x_i$ the formula holds true anyway). Define

$$g(t) := f(t) - p(t) - cL(t)$$

where $c = \frac{f(x)-p(x)}{L(x)}$ so that $g(x) = 0$. Hence, $g(t) = 0$ for the $n+2$ pairwise distinct points $x_0, x_1, \ldots, x_n, x$. By Rolle's Theorem, between each successive pair there is a point where g' vanishes. Repeating this argument for $g', g'', \ldots, g^{(n)}$ there is a point $\xi \in [a, b]$ such that $g^{(n+1)}(\xi) = 0$, i.e.

$$f^{(n+1)}(\xi) - p^{(n+1)}(\xi) - c\, L^{(n+1)}(\xi) = 0.$$

Since $p(x)$ is a polynomial of degree at most n, $p^{(n+1)}(x) = 0$ for all x. Since $L^{(n+1)}(x) = (n+1)!$ for all x we have

$$c = \frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

Since $g(x) = 0$, this completes the Proof. $\square$

C.2.13. Remark (Lagrange vs. Taylor). The error estimate (C.2.20) looks very badly like the error estimate for the error of Taylor's polynomial, T_n of order n at one of the points x_i, say x_0. This is not surprising as, formally at least, if $x_i(\theta) \to x_0$, for all $i = 1, \ldots, n$ and a given parameter $\theta \to 0$ then $p_n \to T_n$. This formal argument can be made rigorous, but that is beyond the scope of these notes.
So one may ask, why insist with Lagrange, if Taylor gives the same rate of approximation. And the answer to this lies in that the Lagrange polynomials do not require any information about the derivative of the function, just the value of the function at given points, whereas Taylor requires access to all the derivatives (and thus a formula for them). But in many applications, e.g., scientists taking data of the growth of bacteria may take measurements at certain times and then try to fit a function to this data. This is a naive, but simple case where access to the function is easy, but access to derivatives is in fact impossible. Note however that polynomial interpolation (including Lagrange polynomial-based one) has limited direct practical use because of *stability* problems for large data sets and is generally eschewed as an interpolation tool for more robust methods, such as *least squares.*
On the other hand ideas behind polynomial interpolation, including Lagrange interpolation, lead to powerful methods, e.g., Galerkin and finite element methods, for the numerical approximation of differential equation solutions.

C.3. Newton's divided differences

From a coding point of view, the form we have described for the Lagrange polynomials has a serious drawback. If we want to ignore a certain interpolation point of the initial set, or add another interpolation point then we must start over, and construct a new set of Lagrange basis polynomials. *Newton's divided difference* method fixes this problem.

C.3.1. Newton's basis. Newton's method[1] allows this to be done by using the Netwon basis polynomials:

$$N_0(x) = 1,\ N_1(x) = (x - x_0),\ N_2(x) = (x - x_0)(x - x_1),\ \text{etc.} \tag{C.3.1}$$

[1]Also, if our goal is to calculate the value of the interpolant at one point x only, Neville's method, which is based on Newton's finite differences, can achieve that in $O(n^2)$ time, which is faster than computing the coefficients, the Lagrange basis and then evaluating at just the point x.

The guess for the polynomial interpolating $(x_i, f(x_i))$, for $i = 0,\dots,n$, is $p = \sum_{i=0}^{n} d_i N_i$, which writes explicitly as

$$p(x) = d_0 + d_1(x-x_0) + \cdots + d_n(x-x_0)(x-x_1)\cdots(x-x_{n-1}) \tag{C.3.2}$$

The coefficients $d_0,\dots,d_n$ can hence be calculated by solving the linear system

$$\begin{aligned} p(x_0) &= d_0 &&= f^0\\ p(x_1) &= d_0 + d_1(x_1-x_0) &&= f^1\\ &\vdots\\ p(x_n) &= d_0 + d_1(x_1-x_0) + \cdots + d_n(x_1-x_0)\cdots(x_n-x_{n-1}) &&= f^n \end{aligned} \tag{C.3.3}$$

This system of linear equations has a unique solution supposed that all points x_i are pairwise different: the system with lower triangular matrix is given by

$$\begin{bmatrix} 1 & 0 & \dots & 0\\ 1 & x_1-x_0 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 1 & x_n-x_0 & \dots & (x_n-x_0)\cdots(x_n-x_{n-1}) \end{bmatrix} \begin{bmatrix} d_0\\ d_1\\ \dots\\ d_n \end{bmatrix} = \begin{bmatrix} f(x_0)\\ f(x_1)\\ \vdots\\ f(x_n) \end{bmatrix}. \tag{C.3.4}$$

Since d_k only depends on $x_0,\dots,x_k$ and the function values $f(x_0),\dots,f(x_k)$, we denote

$$d_k := f[x_0, x_1\dots, x_k].$$

Hence, if we add a point x_{n+1}, then all previous coefficients $d_0,\dots,d_n$ stay the same and $p_{n+1}(x) = p_n(x) + d_{n+1}(x-x_0)(x-x_1)\dots(x-x_n)$.

C.3.2. Lemma (recursive property of Newton's polynomials). *Let $n \geq 0$, $x_0,\dots,x_n$ be distinct points in $\mathbb{R}$, $n \geq k \geq 0$, $n-k \geq m \geq 0$, and denote by $p_{k,m}$ the polynomial such that*

$$\deg p_{k,m} \leq m \text{ and } p_{k,m}(x_i) = f(x_i) \quad \forall\, i = k,\dots,k+m. \tag{C.3.5}$$

Then we have

$$p_{k,m} = \frac{p_{k+1,m-1} - p_{k,m-1}}{x_{k+m} - x_k}. \tag{C.3.6}$$

Proof Let q be the right hand side of (C.3.6), then

$$\begin{aligned} q(x_i) &= \frac{p_{k+1,m-1}(x_i) - p_{k,m-1}(x_i)}{x_{k+m}-x_k}\\ &= \begin{cases} 0 & \text{if } k+1 \leq i \leq k+m-1,\\ \dfrac{p_{k+1,m-1}(x_k) - p_{k,m-1}(x_k)}{x_{k+m}-x_k} & \text{if } i = k,\\ \dfrac{p_{k+1,m-1}(x_{k+m}) - p_{k,m-1}(x_{k+m})}{x_{k+m}-x_k} & \text{if } i = k+m. \end{cases} \end{aligned} \tag{C.3.7}$$

We show that the right-hand-side is the unique interpolation polynomial of the interpolation problem with the $m+1$ points $x_k,\dots,x_{k+m}$, cf. Theorem C.2.3. Since the degree of both $p_{k+1,m-1}$ and $p_{k,m-1}$ is at most $m-1$, the degree of the right-hand side is at most m.

The right-hand side evaluated at x_k gives

$$\frac{-(x_k - x_{k+m})f(x_k)}{x_{k+m} - x_k} = f(x_k) \tag{C.3.8}$$

At x_{k+m} we obtain

$$\frac{(x_{k+m} - x_k)f(x_{k+m})}{x_{k+m} - x_k} = f(x_{k+m}) \tag{C.3.9}$$

For $k+1 \le j \le k+m-1$ we have for the right-hand side evaluated at x_j:

$$\frac{(x_j - x_k)f(x_j) - (x_j - x_{k+m})f(x_j)}{x_{k+m} - x_k} = f(x_j). \tag{C.3.10}$$

This concludes the proof. □

C.3.3. Corollary (Newton's divided difference formula). *Let $p_{k,m}$ be the polynomial interpolating the values f^k, for $k = i,\dots,n$, at the nodes x_i, for $i = k,\dots,m$,. The coefficients $f[x_k,\dots,x_{k+m}]$ of*

$$\begin{aligned} p_{k,m}(x) =& f[x_k] + f[x_k, x_{k+1}](x - x_k) + f[x_k, x_{k+1}, x_{k+2}](x - x_k)(x - x_{k+1}) \\ &+ \dots + f[x_k, x_{k+1}, \dots, x_{k+m}](x - x_k)\dots(x - x_{k+m-1}) \end{aligned} \tag{C.3.11}$$

are determined by the following recursive divided difference formula

$$\begin{aligned} f[x_k] &:= f^k \\ f[x_k,\dots,x_{k+m}] &:= \frac{f[x_{k+1},\dots,x_{k+m}] - f[x_k,\dots,x_{k+m-1}]}{x_{k+m} - x_k}. \end{aligned} \tag{C.3.12}$$

Proof We prove this by induction with respect to m. For $m = 0$, the constant polynomial $p_{k,0} :\equiv f[x_k] := f^k$ interpolates the node-value data (x_k, f^k).
Next, suppose the inductive hypothesis is satisfied, that is the coefficients of $p_{k,m-1}$ and $p_{k+1,m-1}$ are given by formula (C.3.11). Upon "adding" the point x_{k+m} to the interpolation polynomial $p_{k,m-1}$, we obtain $p_{k,m}$ which, according to lemma C.3.2, that

$$p_{k,m} \tag{C.3.13}$$

we have noticed before, that the coefficients except for the last one do not change. For the last coefficient the recursion formula follows from the highest-order terms in the equation of Lemma C.3.2. □

C.3.4. Theorem (divided differences interpolation). *The polynomial*

$$p_N(x) = f[x_0] + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] \tag{C.3.14}$$
$$+ (x - x_0)(x - x_1)\cdots(x - x_{N-1})f[x_0, x_1, \dots, x_N], \tag{C.3.15}$$

where $f[x_k,\dots,x_{k+m}]$ is given by Corollary C.3.3, satisfies the interpolation condition

$$p_N(x_k) = f^k, \quad \forall\, k = 0,\dots,N. \tag{C.3.16}$$

Hence, this polynomial is just a rearrangement of the Lagrange polynomial. The advantages of this construction are:

(1) Extra data points can be introduced without any waste of effort that the Lagrange formula requires.

(2) Since the interpolation polynomial is unique, NO ordering of the interpolation points is necessary (this is also the case for the Lagrange polynomial).

C.3.5. Example. Use Newton's divided difference formula to estimate $f(3)$ using the following data:

$$\begin{array}{c|ccc} x & 1 & 2 & 4 \\ \hline f(x) & -1 & 1 & 3 \end{array} \tag{C.3.17}$$

We can now build a difference table:

k	x_k	$f[x_k]$	$f[x_k, x_{k+1}]$	$f[x_k, x_{k+1}, x_{k+2}]$
0	1	−1	2	−1/3
1	2	1	1	
2	4	3		

Hint. Build the table in successive columns.
Now we are able to construct polynomial approximations to $f(3)$ as follows

(1) Linear, using x_0 and x_1:

$$p_1(x) = f[x_0] + (x - x_0) f[x_0, x_1] = -1 + 2(x-1) = 2x - 3.$$

This gives the approximation

$$f(3) \approx p_1(3) = 3.$$

(2) Quadratic:

$$\begin{aligned} p_2(x) &= f[x_0] + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] \\ &= p_1(x) + (x - x_0)(x - x_1) f[x_0, x_1, x_2] \\ &= 2x - 3 - \frac{1}{3}(x-1)(x-2) = -\frac{1}{3}x^2 + 3x - \frac{11}{3} \end{aligned}$$

This gives the approximation

$$f(3) \approx p_2(3) = -3 + 9 - \frac{11}{3} = \frac{7}{3}.$$

C.3.6. Example. Use Newton's divided difference formula to estimate log(1.2) using the following data:

$x =$	1.0	1.4	1.5	2.0
$\log x$	0.0000	0.3365	0.4055	0.6931

We can now build a difference table:

k	x_k	$f[x_k]$	$f[x_k, x_{k+1}]$	$f[x_k, x_{k+1}, x_{k+2}]$	$f[x_k, x_{k+1}, x_{k+2}, x_{k+3}]$
0	1.0	0.0000	0.8413	−0.3026	0.1113
1	1.4	0.3365	0.6900	−0.1913	
2	1.5	0.4055	0.5752		
3	2.0	0.6931			

Hint. Build this table in successive columns.
Now we are able to construct polynomial approximations to $\log x$ as follows

(1) Linear, using x_0 and x_1:

$$p_1(x) = f[x_0] + (x - x_0) f[x_0, x_1] = 0.0000 + 0.8413(x - 1.0).$$

This gives the approximation

$$\log(1.2) \approx p_1(1.2) = 0.1683.$$

(2) Quadratic, using x_0, x_1 and x_2:

$$\begin{aligned} p_2(x) &= p_1(x) + (x - x_0)(x - x_1) f[x_0, x_1, x_2] \\ &= 0.0000 + 0.8413(x - 1.0) - 0.3026(x - 1.0)(x - 1.4) \end{aligned}$$

This gives the approximation

$$\log(1.2) \approx p_2(1.2) = 0.1804.$$

(3) Cubic:

$$\begin{aligned} p_3(x) &= p_2(x) + (x - x_0)(x - x_1)(x - x_2) f[x_0, x_1, x_2, x_3] \\ &= 0.0000 + 0.8413(x - 1.0) - 0.3026(x - 1.0)(x - 1.4) \\ &\quad + 0.1113(x - 1.0)(x - 1.4)(x - 1.5) \end{aligned}$$

This gives the approximation

$$\log(1.2) \approx p_3(1.2) = 0.1817.$$

Correct up to 4 decimal places we have $\log(1.2) \approx 0.1823$.
Exercise: How may we change the order to build improving approximations to the value $\log(1.46)$?

Exercises and problems on polynomial interpolation

Exercise C.1. Consider the data points $x_0 = -1$, $x_1 = 1$, $x_2 = 2$.

(1) Calculate the Lagrange basis polynomials for the above points.
(2) Calculate the interpolation polynomial $p_2(x)$ for the data $f(-1) = -1$, $f(1) = 1$, $f(2) = -1$.
(3) Can you determine the interpolation polynomial $p_3(x)$ for the data $f(-1) = -1$, $f(0) = 1$, $f(1) = 1$, $f(2) = -1$ without repeating the whole process?

Exercise C.2. Show that the Vandermonde matrix

$$\boldsymbol{V} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \tag{PC.2.1}$$

is invertible if the points $x_0, \ldots, x_n$ are pairwise different.
Hint. Assume in contradiction that $\boldsymbol{V}$ is singular, i.e. there is a $(n+1)$-vector $\boldsymbol{a} \neq \boldsymbol{0}$ such that $\boldsymbol{V}\boldsymbol{a} = \boldsymbol{0}$. A more brute-force approach is to calculate the determinant of $\boldsymbol{V}$.

Exercise C.3. At times x_0, x_1, x_2, x_3 a scientist measures the mass, $f(x)$, of a tumour subjected to radiation. He obtains the results

$$f(0) = 1, \quad f(0.5) = 3, \quad f(1.5) = 4, \quad f(3) = 2.$$

(1) Complete the divided difference table:

k	x_k	$f[x_k]$	$f[x_k, x_{k+1}]$	$f[x_k, x_{k+1}, x_{k+2}]$	$f[x_k, x_{k+1}, x_{k+2}, x_{k+3}]$
0	0	1			
1	0.5	3			
2	1.5	4			
3	3.0	2			

(2) Find the quadratic polynomial $p_2(x)$ that interpolates the data at times $0, 0.5, 1.5$.
(3) Find the cubic polynomial $p_3(x)$ that interpolates at all the given data points.
(4) Plot the four data points as well as $p_2(x)$ and $p_3(x)$ for $x \in [0,4]$ into one figure using Matlab.

Exercise C.4. (a) Calculate the Lagrange basis polynomials $l_j(x)$ for the points

$$x_0 = 0, x_1 = 1 \text{ and } x_2 = 3.$$

(b) Calculate the interpolating polynomial $p_2(x)$ for the data

$$f(0) = 1, f(1) = 0, f(3) = -1.$$

(c) Calculate the interpolating polynomial $q_2(x)$ for the data

$$f(0) = 1, f(1) = 0, f(3) = -2.$$

(d) Calculate the interpolating polynomial $r_2(x)$ for the data

$$f(0) = 1, f(1) = 0, f(2) = -1.$$

(e) Calculate $p_2(x)$, $q_2(x)$ and $r_2(x)$ using Newton's divided differences. Show that you obtain the same polynomials. Describe the differences between the Lagrange and Newton method.

Exercise C.5. We interpolate the function $f(x) = \frac{1}{1+x^2}$ in the interval $[-5,5]$ using 13 interpolation points.

(a) Use the 13 equidistant interpolation points $x_k = -5 + \frac{10}{12}k$, $k = 0, \dots, 12$.
(b) Use the 13 non-equidistant interpolation points

$$x_k = 5\cos\left(\frac{12-k}{12}\pi\right) \text{ for } k = 0, \dots, 12. \qquad \text{(PC.5.1)}$$

They are the extrema of the 12-th Chebyshev polynomial, transformed to the interval $[-5,5]$.

Use a code to compute both interpolation polynomials and plot them both together with the function f in one figure. Also plot the non-equidistant interpolation points of (b) – describe where they are dense, where sparse. Which choice of interpolation points (a) or (b) gives a better approximation?

APPENDIX D

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.

D.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.

D.2. Matlab and Octave

D.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,

[1]I like to use "hacker" and "to hack" in their original English meaning which mean, respectively, someone who likes devlopping 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.

[2]Matlab® 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 comptuing 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 insitutions 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 sofwared 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.

D.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.

D.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

D.3. Octave

In this tutorial we focus on writing scripts and functions rather than interactive manipulations in Octave. Nevertheless some use of the interactive session are necessary to run scripts and functions.

D.3.1. First steps. If your operating system is on a Unix, e.g., Mac OS X, or Linux[3] the best way to use Octave is from a shall (also known as terminal) window.[4]
If you are working on a Windows machine then to take best advantage of these notes is to install a Linux emulator (and when you gain experience you may even install a whole Linux distribution alongside your Windows machine. Refer to, e.g., Agarwal (2015).
Create a directory (we name ours `Code`) and `cd` into it:

```
bash» mkdir Code
```

```
bash» cd Code
```

Unless otherwise stated, all files are saved in that directory. Now launch Octave by issuing

```
bash» octave
```

Octave scalars. Scalars are the most basic type in Octave which does not distinguish between integers and not. For example

```
octave:*> a = 2; b = 3.5; a+b
```

yields `5.5000`.

Octave vectors. The basic datatype in Octave, as in Matlab®, is the *vector*, which is either a row or a column. The default vector is a row.[5] A (row) vector can be built in several ways as follows:

explicitly: by issuing

```
octave:*> v = [1, 3.3, 5, -1]
```

as an equally spaced sequence: issuing

[3]We assume some (very basic) familiarity with the most simple Unix manipulations. You should be able to find some tutorial with an online search. Good staring points are Veselosky (2015) and Westwind computing (2015) for Linux and Mac OS X (also BSD) respectively.

[4]Some people borrow the microsoftism "command line" to talk about a shell.

[5]This is in contrast with the linear algebra approach where a vector is usually thought of as a column; so beware.

```
octave:*> w = [0:.1:1]
```

using Octave's built-in vector constructors: for example,

```
octave:*> j = ones(1,8)
```

transposing a column: (silly example, but should give the idea)

```
octave:*> column = ones(6,1); row = column'
```

The following function uses a vector to build a Fibonacci sequence.

Printout of file `fibonacci0.m`

```
%% -*- mode: octave -*-
function [f] = fibonacci0(a,b,n)
%% function [f] = fibonacci0(a,b,n)
%% builds the Fibonacci sequence of length n with initial seeds a,b
f = [a,b];
k = length(f);
while k<n
  f(k+1)=f(k)+f(k-1);
  k = length(f);
end
```

Octave matrices. To be useful, vectors need matrices to operate on them. Octave provides a very easy way to build matrices. For example,

```
octave:*> A = [1 2; 3 4]
```

builds a computer version of the matrix

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}. \tag{D.3.1}$$

A scalar is viewed by Octave of as a 1×1 matrix.

D.3.2. Plotting. A most simple plotting function is given by the following.

Printout of file `plotfun0.m`

```
%% -*- mode: octave -*-
function [] = plotfun0()
%% function [] = plotfun0()
%% a simple plot function
x = [pi/8:pi/8:2*pi]
y = sin(x)./x
plot(x,y);
```

D.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 comminity,
⋆ 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 progammers and developers which makes other people's code readable.

D.4.1. References. The literature on Python is huge, including that specialised for numerical analysis and scientific comptuation. 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.

D.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 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 keword, 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()
```

D.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
```

There are two ways to run the script, in both ways you need to `cd` into the directory where the code is located[6]

run from shell:

```
bash» ./fibonacci.py
```

run interactively:

```
>>> import fibonacci
```

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
```

[6]For more “advanced” hackers, this is actually falst, all you need to prepent or tweak path appropriately.

```
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
```

D.4.4. 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 is 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 "deimported" (such and operation does not exist 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.

[2] Amit Agarwal. *How to Install Linux on your Windows Computer*. URL: http://www.labnol.org/software/run-linux-with-windows/19746/ (visited on 23/10/2015).

[3] Lars V. Ahlfors. *Complex analysis*. Third. An introduction to the theory of analytic functions of one complex variable, International Series in Pure and Applied Mathematics. New York: McGraw-Hill Book Co., 1978, pp. xi+331. ISBN: 0-07-000657-1.

[4] Kendall E. Atkinson. *An introduction to numerical analysis*. Second. New York: John Wiley & Sons Inc., 1989, pp. xvi+693. ISBN: 0-471-62489-6.

[5] L. K. Babadzanjanz. "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).

[6] Ronald A. DeVore and George G. Lorentz. *Constructive approximation*. Vol. 303. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Berlin: Springer-Verlag, 1993, pp. x+449. ISBN: 3-540-50627-6.

[7] 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).

[8] 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.

[9] Wendell Fleming. *Functions of several variables*. Second. Undergraduate Texts in Mathematics. New York: Springer-Verlag, 1977, pp. xi+411.

[10] Avner Friedman. *Foundations of modern analysis*. Rinehart and Winston, Inc., New York: Holt, 1970, pp. vi+250.

[11] Joseph F. Grcar. "How ordinary elimination became Gaussian elimination". In: *Historia Mathematica* 38.2 (May 2011), pp. 163–218. ISSN: 0315-0860. DOI: 10.1016/j.hm.2010.06.003. URL: http://www.sciencedirect.com/science/article/pii/S0315086010000376 (visited on 26/10/2015).

[12] 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.

[13] Paul R. Halmos. *Finite-dimensional vector spaces.* second. Undergraduate Texts in Mathematics. New York: Springer-Verlag, 1974, pp. viii+200.

[14] John Hamal Hubbard and Barbara Burke Hubbard. *Vector calculus, linear algebra, and differential forms: a unified approach with Maple 10 VP.* en. Pearson Education, Limited, Mar. 2006. ISBN: 978-1-4058-3615-9.

[15] Vasile I. Istrăţescu. *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).

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] python.org. *The Python Tutorial — Python 2.7.9 documentation.* URL: https://docs.python.org/2/tutorial/index.html (visited on 02/01/2015).

[22] 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).

[23] 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.

[24] Gilbert Strang. *Introduction to Linear Algebra.* en. Google-Books-ID: M19gPgAACAAJ. Wellesley-Cambridge Press, Feb. 2009. ISBN: 978-0-9802327-1-4.

[25] Gilbert Strang and MIT-OCW. *“Linear Algebra.” MIT OpenCourseWare.* 2nd Oct. 2016. URL: https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/.

[26] A. M. Stuart and A. R. Humphries. *Dynamical systems and numerical analysis.* Vol. 2. Cambridge Monographs on Applied and Computational Mathematics. Cambridge: Cambridge University Press, 1996, pp. xxii+685. ISBN: 0-521-49672-1.

[27] 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).

[28] Tomonari Suzuki and Badriah Alamri. “A sufficient and necessary condition for the convergence of the sequence of successive approximations to a unique fixed point II”. en. In: *Fixed Point Theory and Applications* 2015.1 (Apr. 2015), p. 59. ISSN: 1687-1812. DOI: 10.1186/s13663-015-0302-9. URL: http://www.fixedpointtheoryandapplications.com/content/2015/1/59/abstract (visited on 28/04/2015).

[29] Noel M. Swerdlow. “Kepler’s Iterative Solution to Kepler’s Equation”. In: *Journal for the History of Astronomy* 31 (Nov. 2000), p. 339.

[30] Sergei Treil. *Linear Algebra Done Wrong*. online book. Department of Mathematics, Brown University, Sept. 2015. URL: https://www.math.brown.edu/~treil/papers/LADW/LADW.html.

[31] Vince Veselosky. *Bootstrap – A Starter for the Impatient — Guide to Linux for Beginners*. URL: http://www.control-escape.com/linux/bootstrap.html (visited on 23/10/2015).

[32] 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).

[33] Westwind computing. *How-tos*. URL: http://www.westwind.com/reference/tips.html (visited on 23/10/2015).

Index

$\mathbb{C}$-diagonalisable, 103
$\mathbb{K}$-diagonalisable, 103
k-linear, 100
n-body problem, 74
(Hamel) basis, 118
$C^{0,\alpha}(D)$, 123
$\left|f\right|_{C^{0,\alpha}(D)}$, 123
apriori, 66, 67

Abelian, 7, 87
Abelian group, 87
adjoint, 92
adjoint of a matrix, 34
adjoint operator, 109
algebra, 29, 30, 93
algebraic basis, 93
algebraic multiplicity, 103
algebraically closed field, 95
alternating, 100
alternating form,form
  alternating, 100
analysis, 130
angle-cosine, 33
antisymmetric, 100
antisymmetric forms, 100
aposteriori, 66, 67
approximate, 51
approximation, 7
approximation algorithms, 2
Approximation Theory, 51
attribute, 146

back-multiply, 48
back-substitution, 3
backwards, 10
Banach space, 124
Banach spaces, 114
basis, 93
best fit, 51
bidual space, 97
bilinear forms, 100
blockwise matrix algebra, 15
Bolzano–Weierstrass, 10, 121
  Theorem, 121
bound, 130
bounded set, 121
bounded-linear operator, 117

calculus, 130
Cauchy criterion, 114
Cauchy sequence, 70
Cauchy–Bunyakovsky–Schwarz inequality, 116
Cholesky, 44
Cholesky's algorithm, 44
class definition, 146
closed ball, 120
closed set, 121
coercivity, 39
column vector, 90
column-index, 8
commutative, 7, 87
commutative group, 87
compact set, 121
complete, 70
complete metric space, 6, 70
complete metric spaces, 124
complex, 111
complex Cholesky, 46
complex diagonalisable, 103
complexification, 102
complexified operator, 102
complexity, 14
computational cost, 14
condition number, 25
conformal, 111
constant polynomial, 94
constructive proof, 16
continous on (the set), 120
continuous at a point, 120
contraction, 68, 69, 77, 124
contractive, 69
converge linearly, 72
converge quadratically, 72
converge to, 113
converge with rate p, 72
convergence

quadratic, 72
convergence rate, 72
convergent, 106
convergent sequence of real numbers, 9, 113
converges superlinearly, 72

defective, 103
determinants, 100
diagonal, 8
diagonalisable, 103
diagonally normalised matrix, 8
dimension, 118
Dirac delta, 98
Dirac mass, 98
direct method, 7
direct solvers, 7
distance, 114
diverge, 114
divided difference formula, 140
divided difference(s), 138
dot product notation, 92
duality, 114
duality bracket notation, 96

eigenequation, 103
eigenpair, 103
eigenspace, 103
eigenvalue, 102, 103
eigenvector, 103
elimination, 29, 30
empty summation, 10
entries, 90
entry, 90
equivalent, 119
error, 64
error analysis, 64
error bound, 67
error bounds, 66
error estimate, 67
error reduction, 66, 67
linear, 72
quadratic, 72
errors, 70
Euclidean algorithm, 1, 3
Euclidean inner product, 33
Euclidean length, 33
Euclidean norm, 107, 115
event, 146
exact, 2
exact method, 3
exact solution, 1
expansion, 130
exterior forms, 100

field, 88, 111
finite basis, 93
finite dimensional, 118
finite dimensional vector space, 93
finite element methods, 138
finite linear combinations, 118
fixed-point, 65
fore-substitution, 9
Frobenius norm, 118
Functional Analysis, 109
functional analysis, 114
functional language, 150

Galerkin, 138
Gaussian, 29, 30
Gaussian elimination, 2, 3
geometric multiplicity, 103
geometric series, 3
geometric series method, 3, 5
Grammian, 40
greatest common divisor, 1
group, 7, 87

Hölder inequality, 115
Hölder continuous of order α, 123
Hamel basis, 93
Hankel matrix, 52
Hermitian, 4, 39, 42
highest common factor, 1
Hilbert dimension, 108
Hilbert space, 108, 129
Hilbert spaces, 114
Hilbertian norm, 107
homogeneity, 114, 116
Householder matrix, 37, 41

induced, 117
induced bounded-linear operator norm, 9, 117
induced matrix norm, 117
induced operator norm, 117
induction, 15
inner product, 107, 129
inner product space, 107
inner products, 100
input, 2
intepolation values, 136
interactive mode, 12, 149
interpolate exactly, 51
interpolation, 134
invariant (sub)set, 103
invertible, 11, 27
involution, 37
involutive, 37
iterates, 66
iteration, 5

iterative solvers, 7

Kronecker's delta notation, 91

Lagrange basis, 11, 135, 136
Lagrange basis polynomials, 135
Lagrange polynomial, 135
Lagrange polynomials, 136
least squares, 138
least squares best fit, 52
least-squares polynomial approximation, 51
left inverse, 99
left invertible, 99
length, 33
limit, 106
limit of a sequence, 9, 106
linear, 29, 30
linear algebra, 39, 42
linear combination, 92
linear convergence, 72
linear convergence rate, 70
linear error reduction, 72
linear form, 96
linear forms, 100
linear Lagrange interpolation, 11, 133
linear least squares, 52
linear map, 96, 117
linear operator, 96
linear spaces, 114
Lipschitz constant, 123
Lipschitz continuous, 67
Lipschitz-continuous, 122
listener, 146
LLT factorisation, 46
lower triangular, 8
LU, 29, 30
LU factorisation, 16, 28
LU-cky, 15, 28

Manhattan taxi-cab norm, 115
many body problem, 74
matrix, 29, 30, 39, 42, 90
matrix inverse, 12
matrix norm, 116
matrix notation, 90
matrix-matrix multiplication, 91
maximum row-sum, 116
method, 7, 146
metric, 114
metric space, 70
multilinear form, 100
multiple root, 81

neighbourhood, 120
Neumann series method, 3
Newton's divided difference, 138
Newton–Raphson iteration, 80
nodes, 136
noise, 51
nondefective, 103
norm, 106, 114
norm triangle inequality, 116
normal equations, 52
normal matrix, 5, 40
normalised, 8
normalised lower triangular matrix, 50
numerical, 2

object, 147
open ball, 120
open set, 120
operation count, 14
operator, 39, 42
operator norm, 117
operator theory, 114
operators, 119
orthogonal, 35, 55
orthogonal pair, 34
orthogonality
  with respect to points, 56
orthonormal system, 35
output, 2

parallelogram, 129
parallelogram identity, 129
parallelogram law, 129
permutation, 29, 30
pivot, 29, 30
PLU, 29, 30
point spectrum, 103
polynomial, 94
polynomial fitting, 51
polynomial-zero, 94
positive, 39
positive definite, 39
positive semidefinite, 39
positively definite, 39
positivity, 114, 116
postinverse, 99
postinvertible, 99
preinvertible, 99
principal block, 42
product, 115
property, 146
pseudocode, 11
pull-back, 101
push-forward, 101
Pythagoras identity, 130
Pythagoras theorem, 130

Q-less QR factorisation, 50
QR factorisation, 47
quadratic convergence, 72
quadratic convergence rate, 70
quadratic error reduction, 72

Rademacher's theorem, 123
rate p error reduction, 72
real Cholesky, 46
recursion, 15
recursive, 13
recursive algorithm, 13
recursive back-substitution, 13
reflection, 111
reloaded, 151
remainder, 130
remainder term, 131
RIDCIU, 91
RIFCIL, 91
right invertible, 99
rotation, 111
row vectors, 90
row-index, 8
row-index-down-col-index-up, 91
row-index-first-col-index-last, 91

scalar–vector multiplication, 89
self-adjoint, 4, 39, 42
separable Hilbert space, 108
sequence, 113
sesquilinear form, 102
simple root, 81
singular value, 104
skew symmetric forms, 100
solver, 7
span, 92
spanned space, 92
SPD, 39
spectral radius, 104
spectral value, 104
spectrum, 104
stability, 82, 138
subadditvity, 116
subclass, 147
submultiplicativity, 116
substituting, 10
superclass, 147
superlinear convergence, 72
symmetric, 4, 39, 42, 100

table, 90
Taylor, 130
Taylor polynomial, 131
Taylor's polynomial, 11, 132
term
  of a sequence, 113
terminate, 2
tester code, 12
theorem, 130
topological spaces, 122
topology, 114
triangle inequality, 114
triangular, 8

UHU factorisation, 46
un-LU-cky, 15
uniformly continuous, 120
uniqueness of the limit, 113
unitary, 35
upper triangular, 8
upper triangular matrix, 10
upwards, 10

Vandermonde matrix, 134
Vandermonde's equation, 134
vector, 147
vector of $\mathbb{K}^n$, 90
vector space, 89
vector spaces, 114

weak convergence, 114
well-defined output, 2