# Mathematical Tripos: IB Numerical Analysis

## Contents


**0 Numerical Analysis: Introduction** **i**

0.1 The course . . . i

0.2 What is Numerical Analysis? . . . i

**1 Polynomial Interpolation** **1**

1.1 The interpolation problem . . . 1

1.2 The Lagrange interpolation formula . . . 1

1.3 The Newton interpolation formula . . . 2

1.3.1 Calculating the Newton divided differences . . . 3

1.4 Examples (*Unlectured*) . . . 5

1.5 A property of divided differences . . . 6

1.6 Error bounds for polynomial interpolation . . . 7

1.6.1 Optimal choice of interpolation points . . . 9

**2 Orthogonal Polynomials and Least-squares Approximation** **11**

2.1 Scalar products . . . 11

2.2 Orthogonal polynomials – definition, existence, uniqueness . . . 12

2.3 The three-term recurrence relation . . . 13

2.4 Examples . . . 13

2.5 Least-squares polynomial approximation . . . 14

**3 Approximation of Linear Functionals** **17**

3.1 Linear functionals . . . 17

3.2 Numerical integration . . . 18

3.2.1 Gaussian quadrature . . . 20

3.2.2 Examples . . . 22

3.3 Numerical differentiation . . . 23

3.3.1 Examples (*Unlectured*) . . . 24

3.4 Error for approximation of linear functionals . . . 25

3.4.1 Taylor series expansion . . . 26

3.4.2 Exchanging the order of $\lambda$ and integration . . . 26

3.4.3 Formula for $e_L(f)$ when $K(\theta)$ does not change sign . . . 28

3.4.4 Bounds for $|e_L(f)|$ . . . 29

**4 Initial Value Ordinary Differential Equations** **31**

4.1 One-step methods . . . 31

4.1.1 The Euler method . . . 32

4.1.2 Theta methods . . . 34

4.2 Multistep methods . . . 35

4.2.1 The order of a multistep method . . . 35

4.2.2 The convergence of multistep methods . . . 37

4.2.3 Adams and BDF methods . . . 38

4.3 Runge–Kutta methods . . . 41

4.3.1 Quadrature formulae . . . 41

4.3.2 General Runge–Kutta methods . . . 42

4.3.3 Examples of Runge–Kutta methods . . . 43

4.3.4 Implicit Runge–Kutta methods (*Unlectured*) . . . 46

4.4 Stiff equations . . . 47

4.4.1 Stiffness: the problem . . . 47

4.4.2 Linear stability domains and A-stability . . . 48

4.4.3 A-stability and the maximum principle . . . 50

4.5 Implementation of ODE methods . . . 52

4.5.1 Error control for multistep methods . . . 52

4.5.2 Error control for Runge–Kutta methods . . . 54

4.5.3 The Zadunaisky device . . . 54

4.5.4 Not the last word . . . 55

4.5.5 Solving nonlinear algebraic systems . . . 55

4.5.6 *A distraction* . . . 56

**5 Square Linear Systems and the LU factorisation** **57**

5.1 Triangular matrices . . . 57

5.2 LU factorization and its generalization . . . 58

5.2.1 The construction of an LU factorization . . . 59

5.2.2 Relation to Gaussian elimination . . . 61

5.2.3 Pivoting to avoid breakdown . . . 62

5.2.4 Pivoting to maintain accuracy . . . 64

5.2.5 Further examples (*Unlectured*) . . . 65

5.3 LU factorization theory and application to structured A . . . 66

5.3.1 Existence and uniqueness of the LU factorization . . . 66

5.3.2 Symmetric matrices . . . 69

5.3.3 Positive definite matrices . . . 70

5.3.4 Symmetric positive definite matrices . . . 70

5.3.5 Sparse matrices . . . 71

**6 Linear Least Squares and the QR factorisation** **74**

6.1 The normal equations . . . . . 74

6.2 Orthogonal matrices . . . . . 77

6.3 The QR factorization . . . . . 78

6.4 Constructing a QR factorization . . . . . 80

6.4.1 The Gram–Schmidt orthogonalisation process . . . . . 80

6.4.2 Givens rotation matrices . . . . . 84

6.4.3 Householder reflection matrices . . . . . 89

6.5 Solving least squares problems with the QR factorisation . . . . . 95

6.5.1 Examples . . . . . 96

# 0 Numerical Analysis: Introduction

## 0.1 The course

**The Structure.** The lectures will be mainly pedagogical, covering the theory, with relatively few concrete examples. The *nuts and bolts* of the implementation of the methods are mainly covered in unlectured examples and in the Examples Sheets.

**The Notes.** These notes are *heavily* based on those of Arieh Iserles and Alexei Shadrin; however, the lecturer takes responsibility for errors! Any corrections and suggestions should be emailed to `G.Moore@maths.cam.ac.uk`. Previous versions are:

- Stephen Cowley's notes (used upto 2016)

  `http://www.damtp.cam.ac.uk/user/sjc1/teaching/NAIB/notes.pdf`

- Arieh Iserles' handouts (an excellent summary in 32 pages instead of 76 are available at

  `http://www.damtp.cam.ac.uk/user/na/PartIB/`

- Alexei Shadrin's notes (which are for the old 12-lecture schedule, and cover about two-thirds of the course) are available at

  `http://www.damtp.cam.ac.uk/user/na/PartIB_03/na03.html`

**The Book.** There is also a book which covers most of the course:

Süli & Mayers, *An Introduction to Numerical Analysis*, CUP 2003 (ISBN 0 521 81026 4; ISBN 0 521 00794 1)

The following book is especially useful for differential equations.

Arieh Iserles, *A First Course in the Numerical Analysis of Differential Equations*, CUP 2008 (ISBN-10: 0521734908; ISBN-13: 978-0521734905)

**Demonstrations.** There are a number of MATLAB demonstrations illustrating the course. These are available at

`http://www.maths.cam.ac.uk/undergrad/course/na/`.

You are encouraged to download them and try them out.

## 0.2 What is Numerical Analysis?

Numerical Analysis is the study of algorithms that use *numerical approximation* (as opposed to *symbolic manipulation*) to solve problems of *mathematical analysis* (as distinguished from *discrete mathematics*).[1] The subject predates computers and is application driven, e.g. the Babylonians had calculated $\sqrt{2}$ to about six decimal places sometime between 1800BC and 1600BC.

Numerical Analysis is often about obtaining **approximate** answers. Concern with **error** is therefore a recurring theme. *Rounding errors* arise because 'computers' (human as well as machine) use finite-precision arithmetic (even if 16-digit), but there are other errors as well that are associated with approximations in the solution, e.g. *discretization errors*, *truncation errors*. Other recurring themes include

- **stability**, a concept referring to the sensitivity of the solution of a given problem to small changes in the data or the given parameters of the problem, and
- **efficiency**, or more generally **computational complexity**.

You will have already touched on the latter point in *Vectors & Matrices* where you saw that solving a $n \times n$ linear system in general requires $\mathcal{O}((n+1)!)$ operations by Cramer's rule, but only $\mathcal{O}(n^3)$ operations by Gaussian elimination. However, efficiency is not necessarily a straightforward concept in that its measure can depend on the type of computer in use (e.g. the structure of computer memory, and/or whether the computer has a *parallel* architecture capable of multiple calculations simultaneously).

[1] See *Wikipedia*. Those of who are interested in algorithms for discrete mathematics might like to consult the 2010 on-line *Algorithms* course (see `http://algorithms.soc.srcf.net/`), and/or the follow-up 2011 on-line *Data Structures and Computational Complexity* course (see `http://algorithmstwo.soc.srcf.net/`).

# 1 Polynomial Interpolation

First we must be clear about the two words in the title. We denote by $\mathbb{P}_n[x]$ the *real linear space* of all real polynomials having degree at most $n$ and we observe that each element of $\mathbb{P}_n[x]$ is uniquely defined by its $n+1$ real coefficients. Hence such polynomials have $n+1$ degrees of freedom and $\dim(\mathbb{P}_n[x]) = n+1$. Interpolation is a particular example of curve fitting: i.e. suppose that we have a number of given data values at given distinct data points, then *curve fitting* tries to construct a function which closely fits these values. (The key advantage of such a function being that it is not just defined at the data points.) *Interpolation* is a specific case of curve fitting, in which the function must go exactly through the data values at the data points, and in this case we usually talk about interpolation points.[2] Crucially, with polynomial interpolation, the function we try to construct will be restricted to be a polynomial.

## 1.1 The interpolation problem

Suppose that we are given $n+1$ distinct real points $x_0, x_1, \ldots, x_n$, together with $n+1$ real values $f_0, f_1, \ldots, f_n$. We seek a $p \in \mathbb{P}_n[x]$ such that

$$p(x_i) = f_i, \qquad i = 0, 1, \ldots, n.$$

Such a function is called a *polynomial interpolant*. Note that interpolation at $x_0, x_1, \ldots, x_n$ imposes $n+1$ conditions and this intuitively justifies using a linear space of functions with dimension $n+1$. However this is not enough to guarantee that the polynomial interpolation problem is solvable and our first task is to establish existence, uniqueness and constructability for solutions of this problem.

*Remark.* If $[a, b] \subset \mathbb{R}$ denotes some interval, it is not uncommon for the $\{f_i\}_{i=0}^n$ to be the values at $\{x_i\}_{i=0}^n \subset [a, b]$ of some given continuous function $f : [a, b] \to \mathbb{R}$. In this case we will be interested in how well $p$ approximates $f$ at other points in $[a, b]$.

## 1.2 The Lagrange interpolation formula

Although we may set up (and then hopefully solve) a linear problem with $n+1$ unknowns in order to construct a polynomial interpolant, this approach has three disadvantages. The first theoretical disadvantage is that our $(n+1) \times (n+1)$ coefficient matrix is not obviously non-singular, although in fact it is possible to prove this. The second practical disadvantage is that, if we choose our unknowns to be the coefficients of $x^k$ for $k = 0, 1, \ldots, n$, then our coefficient matrix can be arbitrarily close to being singular. (The famous matrices that arise are called *Vandermonde matrices*.) The third practical disadvantage is that solving an $(n+1) \times (n+1)$ linear system requires $\mathcal{O}(n^3)$ operations in general, as we shall see later in the course, whereas it is actually possible to solve the special polynomial interpolation problem in only $\mathcal{O}(n^2)$ operations using the explicit *Lagrange formula* constructed below. To achieve this, we use a special basis for $\mathbb{P}_n[x]$, i.e. the *Lagrange cardinal polynomials* for the points $x_0, x_1, \ldots, x_n$ defined by

$$\ell_k(x) \equiv \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}, \qquad k = 0, 1, \ldots, n. \tag{1.1a}$$

Note that these polynomials are independent of the data values $\{f_i\}_{i=0}^n$, are exactly of degree $n$ and satisfy the key condition

$$\ell_k(x_j) = \delta_{kj} \qquad j, k = 0, 1, \ldots, n.$$

**Theorem 1.1** (Existence, uniqueness and construction)**.** *Given $n+1$ distinct real points $\{x_i\}_{i=0}^n$ and $n+1$ real numbers $\{f_i\}_{i=0}^n$, there is exactly one polynomial $p \in \mathbb{P}_n[x]$, namely that given below by* (1.1b)*, such that $p(x_i) = f_i$ for $i = 0, \ldots, n$.*

[2] A different problem, which is closely related to interpolation, is the approximation of a complicated function by a simple function, e.g. see *Chebfun* at http://www.maths.ox.ac.uk/chebfun/.

*Proof.* If we define $p \in \mathbb{P}_n[x]$ by

$$p(x) \equiv \sum_{k=0}^{n} f_k \ell_k(x) \qquad x \in \mathbb{R}\,, \tag{1.1b}$$

then existence and construction are immediately verified by

$$p(x_j) = \sum_{k=0}^{n} f_k \ell_k(x_j) = f_j, \qquad j = 0, 1, \ldots, n. \tag{1.1c}$$

In order to establish uniqueness, we employ a proof by contradiction and suppose that both $p \in \mathbb{P}_n[x]$ and $q \in \mathbb{P}_n[x]$ solve the interpolation problem. But then $r \equiv p - q \in \mathbb{P}_n[x]$ vanishes at the $n+1$ distinct interpolation points and the only element of $\mathbb{P}_n[x]$ that can achieve this is the zero polynomial. Hence $p - q \equiv 0$ and the solution of the interpolation problem is unique. □

*Remarks.*

(i) Let us introduce the so-called *nodal* polynomial

$$\omega(x) \equiv \prod_{i=0}^{n} (x - x_i). \tag{1.2a}$$

Then, in the expression (1.1a) for $\ell_k$, the numerator is simply $\omega(x)/(x - x_k)$ while the denominator is equal to $\omega'(x_k)$. With that, we arrive at a compact Lagrange form

$$p(x) = \sum_{k=0}^{n} f_k \, \ell_k(x) = \sum_{k=0}^{n} \frac{f_k}{\omega'(x_k)} \, \frac{\omega(x)}{x - x_k}. \tag{1.2b}$$

(ii) The Lagrange forms (1.1b) and (1.2b) for the unique interpolating polynomial are often the appropriate forms to use when we wish to manipulate this polynomial as part of a larger mathematical expression. We will see an example of this in section §3.2.1 when we discuss *Gaussian quadrature.* However they are not ideal for numerical evaluation, both because of speed of calculation (i.e. complexity) and because of the accumulation of rounding error: e.g. see the *Newton vs Lagrange* demonstration at

http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php

An alternative is the *Newton form* of the interpolating polynomial, which has an *adaptive* (or *recurrent*) nature: i.e. if an extra data point and data value is added, then the new interpolant (say $p_{n+1}$) can be constructed efficiently from the existing interpolant $p_n$ (rather than starting again from scratch).

## 1.3 The Newton interpolation formula

Theoretically, the Lagrange formula in the previous subsection provides everything we need. We now derive, however, an equally famous alternative representation of the interpolating polynomial, which has important practical advantages (as we shall see).

We again suppose that real distinct $\{x_i\}_{i=0}^n$ and real $\{f_i\}_{i=0}^n$ are given and that we seek the unique $p \in \mathbb{P}_n[x]$ such that $p(x_i) = f_i$ for $i = 0, \ldots, n$. To construct our new interpolation formula, we first introduce the following additional polynomial interpolation problems: for $k = 0, 1, ..., n$, let $p_k \in \mathbb{P}_k[x]$ satisfy

$$p_k(x_i) = f_i, \quad i = 0, \ldots, k.$$

We know from the previous subsection that each of these problems has a unique solution and now our new formula is obtained by writing

$$p(x) \equiv p_n(x) = p_0(x) + \{p_1(x) - p_0(x)\} + \{p_2(x) - p_1(x)\} + \cdots + \{p_n(x) - p_{n-1}(x)\}\,. \tag{1.3}$$

The key property of (1.3) is that both $p_{k-1}$ and $p_k$ interpolate the same values $\{f_i\}_{i=0}^{k-1}$ at $\{x_i\}_{i=0}^{k-1}$ and hence their difference is an element of $\mathbb{P}_k[x]$ that vanishes at these $k$ distinct points. Thus we may write

$$p_k(x) - p_{k-1}(x) = A_k \prod_{i=0}^{k-1}(x - x_i) \qquad k = 1, \ldots, n, \tag{1.4}$$

for some real constants $\{A_k\}_{k=1}^n$, and we note that these constants are equal to the *leading* coefficients of $\{p_k\}_{k=1}^n$. It follows that $p \equiv p_n$ can be built step-by-step as one constructs the sequence $\{p_0, p_1, \ldots, p_n\}$, with $p_k$ obtained from $p_{k-1}$ by the addition of the term on the right-hand side of (1.4): thus we finally obtain

$$p(x) \equiv p_n(x) = p_0(x) + \sum_{k=1}^{n}\{p_k(x) - p_{k-1}(x)\} = A_0 + \sum_{k=1}^{n} A_k \prod_{i=0}^{k-1}(x - x_i), \tag{1.5}$$

where we start with $p_0 = A_0 = f_0$.

In the next section, we describe an efficient algorithm for computing the constants $\{A_k\}_{k=0}^n$, which are known as *Newton divided differences* and denoted

$$A_k \equiv f[x_0, \ldots, x_k] \qquad k = 0, 1, \ldots, n. \tag{1.6}$$

This notation is appropriate, since $A_k$ only depends on $\{x_i\}_{i=0}^k$ and $\{f_i\}_{i=0}^k$ as stated in the following definition.

**Definition 1.2** (Divided difference)**.** Given the $k+1$ distinct real points $\{x_i\}_{i=0}^k$ and the $k+1$ real values $\{f_i\}_{i=0}^k$ the Newton divided difference $f[x_0, ..., x_k]$ is defined to be the leading coefficient of the unique $p_k \in \mathbb{P}_k$ which solves the interpolation problem for this data. This divided difference is said to be of *degree* (or *order*) $k$.

Note that a divided difference $f[x_0, \ldots, x_k]$ is a symmetric function of its arguments and that (1.5) and (1.6) allow us to state the following Newton formula for the unique interpolating polynomial.

**Theorem 1.3** (Newton formula)**.** *Given $n+1$ distinct real points $\{x_i\}_{i=0}^n$ and $n+1$ real values $\{f_i\}_{i=0}^n$, we may express the unique solution $p_n \in \mathbb{P}_n$ of the polynomial interpolation problem in the Newton form*

$$p_n(x) = \sum_{k=0}^{n} f[x_0, ..., x_k] \prod_{i=0}^{k-1}(x - x_i). \tag{1.7}$$

### 1.3.1 Calculating the Newton divided differences

We now derive an efficient recurrence relation for calculating (1.6) by first generalising our divided differences to

$$f[x_j, \ldots, x_k] \qquad 0 \leqslant j \leqslant k \leq n :$$

the natural extension of Definition 1.2 being that these expressions denote the leading coefficients of the unique $q \in \mathbb{P}_{k-j}[x]$ which solve the polynomial interpolation problems

$$q(x_i) = f_i \qquad i = j, \ldots, k.$$

This then allows us to establish the following theorem.

**Theorem 1.4** (Recurrence relation)**.** *Given $n+1$ distinct real points $\{x_i\}_{i=0}^n$ and $n+1$ real values $\{f_i\}_{i=0}^n$, it follows that*

$$f[x_j, x_{j+1}, \ldots, x_k] = \frac{f[x_{j+1}, \ldots, x_k] - f[x_j, \ldots, x_{k-1}]}{x_k - x_j} \qquad 0 \leqslant j < k \leqslant n. \tag{1.8}$$

*Proof.* Let $q_0, q_1 \in \mathbb{P}_{k-j-1}[x]$ and $q_2 \in \mathbb{P}_{k-j}[x]$ be the unique solutions of the the following polynomial interpolation problems:

$$\begin{aligned} q_0(x_i) &= f_i \qquad i = j, \ldots, k-1 \\ q_1(x_i) &= f_i \qquad i = j+1, \ldots, k \\ q_2(x_i) &= f_i \qquad i = j, \ldots, k. \end{aligned}$$

Hence we have the relationship

$$q_2(x) = \frac{x - x_j}{x_k - x_j}\, q_1(x) + \frac{x_k - x}{x_k - x_j}\, q_0(x) \tag{1.9}$$

and comparing the leading coefficients of this expression gives the required result. □

This recurrence relation now allows us to efficiently construct (1.6) from the following famous *Newton divided difference table*.

**Method 1.5.** Recalling that $f[x_i] = f_i \quad i = 0, \ldots, n$, the recursive formula (1.8) is used to rapidly evaluate each column of the following table in turn.

| $x_i$ | $f[*] = f(*)$ | $f[*,*]$ | $f[*,*,*]$ | $\ldots$ | $f[*,*,\ldots,*]$ |
|---|---|---|---|---|---|
| $x_0$ | $\to f[x_0]$ | $f[x_0,x_1]$ | $f[x_0,x_1,x_2]$ | $\ddots$ | |
| $x_1$ | $\to f[x_1]$ | $f[x_1,x_2]$ | $f[x_1,x_2,x_3]$ | $\ddots$ | |
| $x_2$ | $\to f[x_2]$ | $f[x_2,x_3]$ | | | |
| $x_3$ | $\to f[x_3]$ | $\ddots$ | | | |
| $\vdots$ | | | | | $\ddots\ f[x_0,x_1,\ldots,x_n]$ |
| $x_{n-1}$ | $\to f[x_{n-1}]$ | $f[x_{n-1},x_n]$ | $f[x_{n-2},x_{n-1},x_n]$ | | |
| $x_n$ | $\to f[x_n]$ | | | | |

**Table 1:** *The divided difference table*

*Remarks.*

(i) The whole table can be evaluated in $\mathcal{O}(n^2)$ operations. Only $\{f[x_0, \ldots, x_k]\}_{k=0}^n$ are needed in (1.7), i.e. the leading diagonal of the table, but their computation requires the other elements as well.

(ii) While it is usual for the points $\{x_i\}_{i=0}^n$ to be in ascending order, there is no need for this condition to be imposed. It also turns out that the cancellation that occurs when divided differences are formed does not lose 'information', although it does reduce the number of leading digits that are reliable in successive columns of the divided difference table (see question 5 on Example Sheet 1).

(iii) An important practical consideration arises if an extra interpolation point, say $x_{n+1}$, and an extra data value, say $f_{n+1}$, are added at the bottom of the table in the first two columns. Then only the additional diagonal

$$f[x_k, \ldots, x_{n+1}] \qquad k = n, \ldots, 0$$

needs to be computed in order to update the table and this merely requires $\mathcal{O}(n)$ operations using the recurrence relation.

We end this subsection by reminding you how, once $\{f[x_0, \ldots, x_k]\}_{k=0}^n$ have been computed, the famous *Horner's scheme* enables $p_n$ in (1.7) to be evaluated at any given point $\hat{x} \in \mathbb{R}$ in only $\mathcal{O}(n)$ operations. Thus the sequence of calculations

$$\begin{aligned} &\sigma \ \leftarrow\ f[x_0, \ldots, x_n] \\ &\textbf{for} \quad k = n-1, \ldots, 0 \\ &\qquad \sigma \ \leftarrow\ \sigma\,(\hat{x} - x_k) + f[x_0, x_1, \ldots, x_k] \\ &\textbf{end} \end{aligned}$$

leads to $\sigma$ finally containing $p_n(\hat{x})$.

*Remarks.*

(i) Horner's scheme can be used similarly to evaluate efficiently any polynomial $p(x) = \sum_{k=0}^{n} c_k x^k$ for any given $\hat{x} \in \mathbb{R}$.

(ii) An advantage of Newton's formula over Lagrange's formula is now evident. Suppose an extra interpolation point, say $x_{n+1}$, and an extra data value, say $f_{n+1}$ is added in order to improve accuracy. Then in order to calculate the extra coefficient in (1.7), only an extra diagonal of the divided difference table need be calculated in $\mathcal{O}(n)$ operations, while to evaluate Newton's formula at any given point $\hat{x}$ takes only $\mathcal{O}(n)$ operations. This is to be compared with $\mathcal{O}(n^2)$ operations to obtain the equivalent result using Lagrange's formula.

(iii) The effect of rounding error on the evaluation of the Newton form compared with the Lagrange form of the interpolating polynomial may be investigated using the *Newton vs Lagrange* demonstration at

http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php

## 1.4 Examples (*Unlectured*)

Given the data

| $x_i$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $f(x_i)$ | $-3$ | $-3$ | $-1$ | 9 |

,

find the interpolating polynomial $p \in \mathbb{P}_3[x]$ in both Lagrange and Newton forms.

*Fundamental Lagrange polynomials:*

$$\begin{aligned}
\ell_0(x) &= \tfrac{(x-1)(x-2)(x-3)}{-6} &&= -\tfrac{1}{6}(x^3 - 6x^2 + 11x - 6),\\
\ell_1(x) &= \tfrac{x(x-2)(x-3)}{2} &&= \tfrac{1}{2}(x^3 - 5x^2 + 6x),\\
\ell_2(x) &= \tfrac{x(x-1)(x-3)}{-2} &&= -\tfrac{1}{2}(x^3 - 4x^2 + 3x),\\
\ell_3(x) &= \tfrac{x(x-1)(x-2)}{6} &&= \tfrac{1}{6}(x^3 - 3x^2 + 2x).
\end{aligned}$$

*Lagrange form:*

$$\begin{aligned}
p(x) &= (-3)\cdot\ell_0(x) + (-3)\cdot\ell_1(x) + (-1)\cdot\ell_2(x) + 9\cdot\ell_3(x)\\
&= \left(\tfrac{1}{2} - \tfrac{3}{2} + \tfrac{1}{2} + \tfrac{3}{2}\right)x^3 + \left(-3 + \tfrac{15}{2} - 2 - \tfrac{9}{2}\right)x^2 + \left(\tfrac{11}{2} - 9 + \tfrac{3}{2} + 3\right)x - 3\\
&= x^3 - 2x^2 + x - 3\,.
\end{aligned}$$

*Divided differences:*

$$\begin{array}{c|ccccc}
0 & \mathbf{-3} & \searrow & & & \\
 & & \tfrac{(-3)-(-3)}{1-0} = \mathbf{0} & \searrow & & \\
1 & -3 & \nearrow\ \searrow & & \tfrac{2-0}{2-0} = \mathbf{1} & \searrow \\
 & & \tfrac{(-1)-(-3)}{2-1} = 2 & \nearrow\ \searrow & & \tfrac{4-1}{3-0} = \mathbf{1} \\
2 & -1 & \nearrow\ \searrow & & \tfrac{10-2}{3-1} = 4 & \nearrow \\
 & & \tfrac{9-(-1)}{3-2} = 10 & \nearrow & & \\
3 & 9 & \nearrow & & &
\end{array}$$

*Newton form:*

$$p(x) = \mathbf{-3} + \mathbf{0}\cdot(x-0) + \mathbf{1}\cdot(x-0)(x-1) + \mathbf{1}\cdot(x-0)(x-1)(x-2)\,.$$

*Horner scheme:*

$$p(x) = \Big\{ \big[ \mathbf{1} \cdot (x - 2) + \mathbf{1} \big] \cdot (x - 1) + \mathbf{0} \Big\} \cdot (x - 0) - \mathbf{3} \,.$$

*Exercise:* Add a $5^{\text{th}}$ point, $x_4 = 4$, $f(x_4) = 0$, and compare the effort to calculate the new interpolating polynomial by the Lagrange and Newton formulae.

## 1.5 A property of divided differences

Suppose that our data values are derived from a particular function, i.e.

$$f_i = f(x_i) \qquad i = 0, \ldots, n$$

for some smooth $f$. In the next subsection, we want to consider how the error

$$e_n(x) \equiv f(x) - p_n(x)$$

depends on $n$ and the choice of $\{x_i\}_{i=0}^n$. (Of course this error is zero at the interpolation points, but how about at other points?) In the present subsection, we will derive a useful preliminary result connecting divided differences with derivatives and we start by reminding you of some standard notation for smooth functions.

**Definition 1.6** ($C^s[a,b]$)**.** Let $[a,b]$ be a closed interval of $\mathbb{R}$. We denote by $C[a,b]$ the linear space of all continuous functions from $[a,b]$ to $\mathbb{R}$ and let $C^s[a,b]$, where $s$ is a positive integer, stand for the linear space of all functions in $C[a,b]$ that possess $s$ continuous derivatives on $[a,b]$.

First we need a preparatory lemma.

**Lemma 1.7.** *If $g \in C^m[a,b]$ is zero at $m + \ell$ distinct points in $[a,b]$, then $g^{(m)}$ has at least $\ell$ distinct zeros in $(a,b)$.*

*Proof.* We just need to apply Rolle's theorem $m$ times. Firstly to deduce that $g' \in C^{(m-1)}[a,b]$ has at least $(m-1)+\ell$ distinct zeros in $(a,b)$ and lastly to deduce that $g^{(m)} \in C[a,b]$ has at least $\ell$ distinct zeros in $(a,b)$. □

Now we can connect divided differences for smooth functions with their derivatives.

**Theorem 1.8** (divided differences and derivatives)**.** *If $f \in C^n[a,b]$ and $\{x_i\}_{i=0}^n \subset [a,b]$ are a set of distinct points then there exists $\xi \in (a,b)$ such that*

$$f[x_0, x_1, \ldots, x_n] = \tfrac{1}{n!} f^{(n)}(\xi). \tag{1.10a}$$

*Proof.* With $p \in \mathbb{P}_n[x]$ being the unique solution of the polynomial interpolation problem, we just apply Lemma 1.7 to $e \equiv f - p_n$. Since $e$ has distinct zeros $\{x_i\}_{i=0}^n \subset [a,b]$, we deduce that $e^{(n)} \equiv f^{(n)} - p^{(n)}$ must vanish for some $\xi \in (a,b)$. Result (1.10a) then follows from

$$p^{(n)}(\xi) = n!\, f[x_0, \ldots, x_n] \,,$$

because (by definition) $f[x_0, \ldots, x_n]$ is the leading coefficient of $p_n$. □

*Application.* A method of estimating a derivative, say $f^{(n)}(\xi)$ where $\xi$ is now given, is to let the distinct points $\{x_i\}_{i=0}^n$ be suitably close to $\xi$, and to make the approximation

$$f^{(n)}(\xi) \approx n!\, f[x_0, x_1, \ldots, x_n] \,. \tag{1.10b}$$

However, a drawback is that, although one achieves good accuracy in theory by picking such close interpolation points, if $f$ is smooth and if the precision of the arithmetic is finite, significant loss of accuracy may occur due to cancellation of the leading digits of the function values.

## 1.6 Error bounds for polynomial interpolation

We are now able to study the *interpolation error*

$$e_n(x) \equiv f(x) - p_n(x) \qquad x \in [a,b]\,, \tag{1.11}$$

when our distinct interpolation points $\{x_i\}_{i=0}^n$ are contained in $[a,b]$ and our data values $\{f_i\}_{i=0}^n$ are derived from a smooth function $f$ over $[a,b]$.

Most of the work in obtaining a useful expression for the interpolation error $e_n \equiv f - p_n$ is contained in the following theorem, which basically states that the error is 'like the next term' in the Newton formula.

**Theorem 1.9.** *Assume that our distinct interpolation points $\{x_i\}_{i=0}^n \subset [a,b]$ and also that $\bar{x} \in [a,b]$ with $\bar{x} \notin \{x_i\}_{i=0}^n$. Also assume that our data values $\{f_i\}_{i=0}^n$ are derived from a function $f \in C[a,b]$. Then*

$$e_n(\bar{x}) = f[x_0, ..., x_n, \bar{x}]\,\omega(\bar{x}), \tag{1.12}$$

*where $\omega \in \mathbb{P}_n[x]$ was defined in* (1.2a).

*Proof.* We think of $x_{n+1} \equiv \bar{x}$ as an 'extra' interpolation point and (like (1.4) and (1.6)) write

$$p_{n+1}(x) = p_n(x) + f[x_0, ..., x_n, \bar{x}]\,\omega(x) \qquad \forall x \in \mathbb{R}\,,$$

where $p_{n+1} \in \mathbb{P}_{n+1}[x]$ interpolates $f$ at $\{x_i\}_{i=0}^{n+1}$. Setting $x = \bar{x}$ and using $p_{n+1}(\bar{x}) = f(\bar{x})$ then gives the required result. □

By assuming sufficient smoothness on $f$, we can then simplify (1.12).

**Theorem 1.10.** *Given $f \in C^{n+1}[a,b]$ and distinct interpolation points $\{x_i\}_{i=0}^n \subset [a,b]$, let $p_n \in \mathbb{P}_n[x]$ be the unique solution of the polynomial interpolation problem for data values $\{f(x_i)\}_{i=0}^n$. Then, for every $x \in [a,b]$, $\exists\, \xi_x \in (a,b)$ such that*

$$e_n(x) \equiv f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi_x)\omega(x). \tag{1.13}$$

*Proof.* The result is immediate if $x$ is an interpolation point. Otherwise we just combine the results of Theorem 1.9 and Theorem 1.8. (Note that $\xi_x$ varies with $x$.) □

*Alternative Proof.* (*Unlectured, but useful for the Examples Sheet.*) The formula (1.13) is true when $x = x_j$ for $j \in \{0, 1, \ldots, n\}$, since both sides of the equation vanish. Let $x \in [a,b]$ be any other point and define

$$\phi(t) = [f(t) - p(t)] \prod_{i=0}^{n} (x - x_i) - [f(x) - p(x)] \prod_{i=0}^{n} (t - x_i), \qquad t \in [a,b].$$

*We emphasise that the variable in $\phi$ is $t$, whereas $x$ is a fixed parameter.* Next, note that $\phi(x_j) = 0$, $j = 0, 1, \ldots, n$, and $\phi(x) = 0$. Hence, $\phi$ has at least $n+2$ distinct zeros in $[a,b]$. Moreover, $\phi \in C^{n+1}[a,b]$.

We now apply Lemma 1.7. We deduce that $\phi'$ has at least $n+1$ distinct zeros in $(a,b)$, that $\phi''$ vanishes at $n$ points in $(a,b)$, etc. We conclude that $\phi^{(s)}$ vanishes at $n+2-s$ distinct points of $(a,b)$ for $s = 0, 1, \ldots, n+1$. Letting $s = n+1$, we have $\phi^{(n+1)}(\xi_x) = 0$ for some $\xi_x \in (a,b)$ and hence

$$0 = \phi^{(n+1)}(\xi_x) = [f^{(n+1)}(\xi_x) - p^{(n+1)}(\xi_x)] \prod_{i=0}^{n} (x - x_i) - [f(x) - p(x)](n+1)!\,.$$

Since $p^{(n+1)} \equiv 0$, we obtain (1.13). □

Making use of the $L_\infty$-norm (or *max*-norm)

$$\|g\|_\infty \equiv \max_{t\in[a,b]} |g(t)| \qquad g \in C[a,b]$$

enables us to replace (1.13) with the more useful *bound*

$$|f(x) - p(x)| \leqslant \frac{1}{(n+1)!}|\omega(x)|\,\|f^{(n+1)}\|_\infty \qquad \forall x \in [a,b]\,. \tag{1.14a}$$

In particular, we can now keep $n$ and $[a,b]$ fixed while investigating how this bound can be minimised through the choice of distinct interpolation points $\Delta \equiv \{x_i\}_{i=0}^n \subset [a,b]$, i.e.

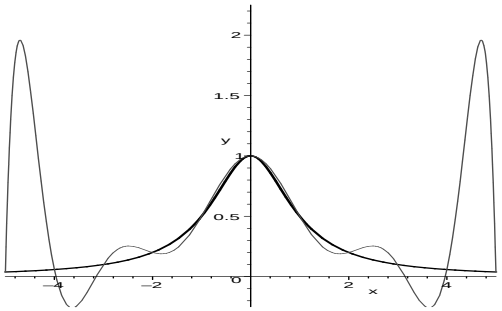
$$\|f - p_\Delta\|_\infty \leqslant \frac{1}{(n+1)!}\|\omega_\Delta\|_\infty\,\|f^{(n+1)}\|_\infty\,. \tag{1.14b}$$

Introducing the lower index $\Delta$ emphasizes the dependence on $\{x_i\}_{i=0}^n$ and we are now trying to minimise $\|\omega_\Delta\|_\infty$ with respect to $\Delta$. The choice of $\Delta$ can make a big difference!

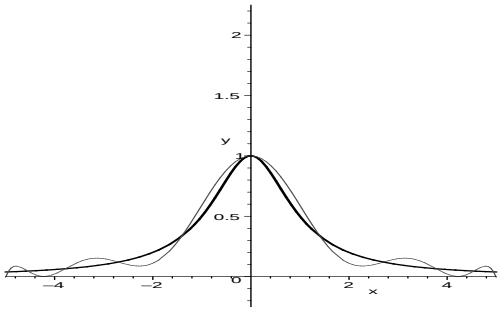
*Runge's Example.* We interpolate

$$f(x) = \frac{1}{1+x^2}, \quad x \in [-5,5], \tag{1.15}$$

first at the equally-spaced *knots* $x_j = -5 + 10j/n$, $j = 0,1,\ldots,n$, and then at the Chebyshev knots $x_j = -5\cos\frac{2j+1}{2(n+1)}\pi$, $j = 0,1,\ldots,n$.




Interpolation at uniform knots $\{-5+j\}_{j=0}^{10}$.



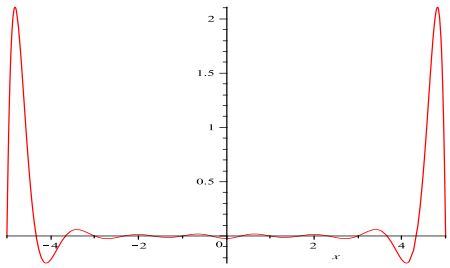

Interpolation at Chebyshev knots $\{-5\cos\frac{2j+1}{22}\pi\}_{j=0}^{10}$.

In the case of equi-spaced points, note the growth in the error which occurs towards the end of the range. As illustrated in the rightmost column of the table below, this arises from the nodal polynomial term in (1.13). Moreover, adding more interpolation points makes the largest error

$$\|f-p\|_\infty = \max\{|f(x) - p(x)| : -5 \leqslant x \leqslant 5\}\,,$$

even worse, as may be investigated using the *Lagrange Interpolation* demonstration at

http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php




Errors in interpolating (1.15) at uniform knots: $n = 15$.

| $x$ | $f(x) - p(x)$ | $\prod_{i=0}^n (x - x_i)$ |
|---|---|---|
| 0.75 | $3.2 \times 10^{-3}$ | $-2.5 \times 10^{6}$ |
| 1.75 | $7.7 \times 10^{-3}$ | $-6.6 \times 10^{6}$ |
| 2.75 | $3.6 \times 10^{-2}$ | $-4.1 \times 10^{7}$ |
| 3.75 | $5.1 \times 10^{-1}$ | $-7.6 \times 10^{8}$ |
| 4.75 | $4.0 \times 10^{+2}$ | $-7.3 \times 10^{10}$ |

Errors in interpolating (1.15) at uniform knots: $n = 20$.

A remedy to this state of affairs is to cluster points towards the end of the range. As illustrated in the second figure above, a considerably smaller error is attained for $x_j = 5\cos\frac{(n-j)\pi}{n}$, $j = 0,1,\ldots,n$, the so-called *Chebyshev points*. It is possible to prove that this choice of points minimizes the magnitude of $\max_{x\in[-5,5]} |\prod_{i=0}^n (x - x_i)|$.

### 1.6.1 Optimal choice of interpolation points

We now begin to establish the results we want, by first minimising $\|\omega\|_\Delta$ in (1.14b) for the special case $[a,b] = [-1,1]$.

**Definition 1.11.** For $n \geqslant 0$, the *Chebyshev*[3] polynomial of degree $n$ on $[-1,1]$ is defined by

$$T_n(x) = \cos n\theta, \qquad \text{where } x = \cos\theta \text{ with } \theta \in [0,\pi]. \tag{1.16}$$

Elementary trigonometrical formulae show that $T_n \in \mathbb{P}_n[x]$ and we also have the following key properties of $T_n$ on $[-1,1]$.

(i) $T_n$ takes its maximal absolute value 1 (with alternating signs) $n+1$ times:

$$\|T_n\|_\infty = 1, \quad T_n(Y_k) = (-1)^k, \quad Y_k = \cos\tfrac{\pi k}{n}, \quad k = 0,1,\ldots,n. \tag{1.17a}$$

(ii) $T_n$ has $n$ distinct zeros:

$$T_n(y_k) = 0, \quad y_k = \cos\tfrac{2k-1}{2n}\pi, \quad k = 1,2,\ldots,n. \tag{1.17b}$$

Like all sets of orthogonal polynomials (as we shall see in the next section), Chebyshev polynomials satisfy a three-term recurrence relation.

**Lemma 1.12.** *The Chebyshev polynomials $T_n$ satisfy the recurrence relation*

$$T_0(x) \equiv 1, \quad T_1(x) = x, \tag{1.18a}$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geqslant 1. \tag{1.18b}$$

*Proof.* (1.18a) follows directly from (1.16) and (1.18b) is derived from the trigonometrical identity

$$\cos(n+1)\theta + \cos(n-1)\theta = 2\cos\theta\cos n\theta$$

via the substitution $x = \cos\theta$. □

*Remark.* (1.18a) and (1.18b) confirm that $T_n \in \mathbb{P}_n[x]$ and that the leading coefficient is $2^{n-1}$ (for $n \geqslant 1$).

The reason for introducing Chebyshev polynomials now is that it is easy to show that they solve our minimisation problem.

**Theorem 1.13.** *On the interval $[-1,1]$, among all $p \in \mathbb{P}_n[x]$ with leading coefficient 1, it is $\frac{1}{2^{n-1}}T_n$ that satisfies*

$$\min_{(a_i)} \left\|x^n + a_{n-1}x^{n-1} + \cdots + a_0\right\|_\infty = \tfrac{1}{2^{n-1}}\|T_n\|_\infty = \tfrac{1}{2^{n-1}}. \tag{1.19}$$

*Proof.* Suppose (by contradiction) $\exists\, q_n \in \mathbb{P}_n[x]$ with leading coefficient 1 and $\|q_n\|_\infty < \frac{1}{2^{n-1}}$. If we define

$$r \equiv \tfrac{1}{2^{n-1}}T_n - q_n \in \mathbb{P}_{n-1}[x]$$

then, because (1.17a) states that $T_n(Y_k) = \pm 1$ and $|q_n(Y_k)| < \frac{1}{2^{n-1}}$ for $k = 0,\ldots,n$, it follows that $r$ alternates in sign at the distinct points $\{Y_k\}_{k=0}^n \subset [-1,1]$. Hence, by the intermediate value theorem, $r$ has at least $n$ distinct zeros in $(-1,1)$ and this contradicts $r \neq 0$. □

Now it is clear that our 'best' choice of interpolation points is derived from Chebyshev polynomials.

[3] Alternative transliterations of Chebyshev include Chebyshov, Tchebycheff and Tschebyscheff: hence $T_n$.

**Corollary 1.14.** *Consider $\omega_\Delta \in \mathbb{P}_n[x]$ for any distinct $\Delta = \{x_i\}_{i=0}^n \subset [-1,1]$: then*

$$\min_\Delta \|\omega_\Delta\|_\infty = \tfrac{1}{2^n} \tag{1.20a}$$

*and this is achieved by the zeros of $T_{n+1}$ in* (1.17b)*, i.e.*

$$x_k = \cos \tfrac{2k+1}{2n+2}\pi \qquad k = 0, \ldots, n\,. \tag{1.20b}$$

Thus we end up with our smallest error bound for polynomial interpolation.

**Theorem 1.15.** *For $f \in C^{n+1}[-1,1]$, the Chebyshev choice of interpolating points, as defined in* (1.20b)*, gives the best bound of the form* (1.14b)*, i.e.*

$$\|f - p_n\|_\infty \leqslant \tfrac{1}{2^n} \tfrac{1}{(n+1)!} \|f^{(n+1)}\|_\infty\,. \tag{1.21}$$

*Example.* For $f(x) = \mathrm{e}^x$, and $x \in [-1,1]$, the error of approximation provided by interpolating polynomial of degree 9 with 10 Chebyshev knots is bounded by

$$|\mathrm{e}^x - p_9(x)| \leqslant \tfrac{1}{2^9} \tfrac{1}{10!}\,\mathrm{e} \leqslant 1.5 \cdot 10^{-9}$$

Finally, we explain how it is easy to generalise the above theorems to an arbitrary finite interval $[a,b]$, rather than just the special interval $[-1,1]$: thus obtaining the results stated in Runge's example. By using the linear transformation

$$x = \tfrac{b+a}{2} + \tfrac{b-a}{2} t\,, \tag{1.22}$$

which maps $t \in [-1,1]$ to $x \in [a,b]$, we see that Theorem 1.13, Corollary 1.14 and Theorem 1.15 can be generalised to $[a,b]$.

a) The analogue of Theorem 1.13 for the general interval $[a,b]$ has solution

$$\tfrac{(b-a)^n}{2^{2n-1}} T_n\left(\tfrac{2}{b-a}x - \tfrac{b+a}{b-a}\right) :$$

i.e. this is the element of $\mathbb{P}_n[x]$, with leading coefficient 1, which has minimal $\infty$-norm over $[a,b]$. Note how the inverse of the transformation (1.22) is used, so that the proof will take a similar form.

b) The analogue of Corollary 1.14 for the general interval $[a,b]$ has

$$\min_\Delta \|\omega_\Delta\|_\infty = \tfrac{(b-a)^{n+1}}{2^{2n+1}}$$

and this is achieved by the zeros of $T_{n+1}$ under the mapping (1.22), i.e.

$$x_k = \tfrac{b+a}{2} + \tfrac{b-a}{2} \cos \tfrac{2k+1}{2n+2}\pi \qquad k = 0, \ldots, n\,.$$

c) The analogue of Theorem 1.15 for the general interval $[a,b]$ is

$$\|f - p_n\|_\infty \leqslant \tfrac{(b-a)^{n+1}}{2^{2n+1}} \tfrac{1}{(n+1)!} \|f^{(n+1)}\|_\infty\,,$$

where the transformed zeros of $T_{n+1}$ are used as interpolation points.

# 2 Orthogonal Polynomials and Least-squares Approximation

There are many other ways, apart from interpolation, of constructing polynomials with good approximation properties. In this section we look at polynomials which are optimal in a *least-squares* sense. We shall see that this requires the construction of polynomials that are *orthogonal* with respect to the scalar product that defines the least-squares problem.

## 2.1 Scalar products

We first need to significantly generalise the idea of a scalar product. Recall (e.g. from *Vectors & Matrices*) that the simplest scalar product between two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ is defined by

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle \equiv \sum_{i=1}^{n} x_i y_i \,. \tag{2.1a}$$

Given arbitrary fixed *weights* $w_1, w_2, \ldots, w_n > 0$, we may broaden this definition to

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle \equiv \sum_{i=1}^{n} w_i x_i y_i \,. \tag{2.1b}$$

In general, given a real linear vector space $\mathbb{V}$, a *scalar* (or *inner*) *product* is any function $\mathbb{V} \times \mathbb{V} \mapsto \mathbb{R}$ which satisfies the following axioms:

$$\left.\begin{array}{lll} \text{symmetry:} & \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle & \forall\ \boldsymbol{x}, \boldsymbol{y} \in \mathbb{V}\,, \\ \text{linearity:} & \langle \alpha\boldsymbol{x} + \beta\boldsymbol{y}, \boldsymbol{z} \rangle = \alpha \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \beta \langle \boldsymbol{y}, \boldsymbol{z} \rangle & \forall\ \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{V}\ \&\ \alpha, \beta \in \mathbb{R}\,, \\ \text{non-negativity:} & \langle \boldsymbol{x}, \boldsymbol{x} \rangle \geqslant 0 & \forall\ \boldsymbol{x} \in \mathbb{V}\,, \\ \text{non-degeneracy:} & \langle \boldsymbol{x}, \boldsymbol{x} \rangle = 0 \quad \text{iff} \quad \boldsymbol{x} = 0\,. & \end{array}\right\} \tag{2.2}$$

In this section, we shall consider special scalar products defined on particular real vector spaces of functions. Since we will be concerned with orthogonal polynomials, our vector spaces must contain $\mathbb{P}_n[x]$, at least up to a certain degree. We must also make sure that the above axioms hold, especially the non-degeneracy condition.

a) For $s \geqslant 0$ and $[a, b]$ finite, let $\mathbb{V} \equiv C^s[a, b]$. The scalar products we are interested in are defined by

$$\langle f, g \rangle = \int_a^b w(x) f(x) g(x) \,\mathrm{d}x \,, \tag{2.3}$$

where

- $w \in C(a, b)$ is a fixed *weight* function,
- $w(x) > 0$ for $x \in (a, b)$,
- $w$ is integrable over $[a, b]$.

In particular, this allows $w$ to be zero at the endpoints or mildly unbounded there. (Many famous sets of orthogonal polynomials, including Chebyshev, require this generality!) Since $\mathbb{P}_n[x]$ is a vector subspace of $C^s[a, b]$, these scalar products are also suitable for $\mathbb{V} \equiv \mathbb{P}_n[x]$ for any $n$.

b) If our interval is infinite, e.g. $[0, \infty)$ or $(-\infty, \infty)$, we can only use (2.3) if we put additional restrictions on the weight function. (Again, this generality is required by some famous sets of orthogonal polynomials!) Even for $\mathbb{V} \equiv \mathbb{P}_n[x]$, we must also require that

$$\int_a^b w(x) x^k \,\mathrm{d}x \quad \text{exists for } 1 \leqslant k \leqslant 2n.$$

For example, if

$$w(x) \equiv \mathrm{e}^{-x} \quad \text{for } [0, \infty) \qquad \text{or} \qquad w(x) \equiv \mathrm{e}^{-x^2} \quad \text{for } (-\infty, \infty)$$

then these scalar products are suitable for $\mathbb{V} \equiv \mathbb{P}_n[x]$ without any restriction on $n$. (Note however that letting $\mathbb{V}$ be a vector space of smooth functions would require some restriction on their behaviour at $\pm\infty$.)

c) If $\mathbb{V} \equiv \mathbb{P}_n[x]$, a somewhat different possibility is the *discrete* scalar product

$$\langle f, g \rangle = \sum_{j=1}^{m} w_j f(\xi_j)\, g(\xi_j) \qquad f, g \in \mathbb{V}\,, \tag{2.4}$$

where $m \geqslant n+1$, $\{\xi_j\}_{j=1}^m$ are fixed distinct points and $\{w_j\}_{j=1}^m$ are fixed positive weights.

Finally, we state an obvious definition.

**Definition 2.1.** *Having chosen a space $\mathbb{V}$ of functions and a scalar product for the space, we say that $f, g \in \mathbb{V}$ are orthogonal if and only if $\langle f, g \rangle = 0$.*

## 2.2 Orthogonal polynomials – definition, existence, uniqueness

If $\mathbb{V}$ is a vector space of functions that contains $\mathbb{P}_n[x]$ $\forall n \geqslant 0$, together with a given scalar product, we say that $p_n \in \mathbb{P}_n[x]$ is the $n^{\text{th}}$ *orthogonal polynomial* if

$$\langle p_n, p \rangle = 0 \quad \forall p \in \mathbb{P}_{n-1}[x]\,. \tag{2.5}$$

This definition implies that $\langle p_m, p_n \rangle = 0$ if $m \neq n$, i.e. orthogonal polynomials of different degrees are orthogonal to each other.

*Remarks.*

(i) Different inner products lead to different sets of orthogonal polynomials.

(ii) Note that, if our vector space $\mathbb{V}$ only contains polynomials up to a certain degree, then we will only be able to construct orthogonal polynomials up to this degree.

In order to establish uniqueness, we must impose some kind of normalisation on our orthogonal polynomials.

**Definition 2.2.** A polynomial in $\mathbb{P}_n[x]$ is said to be *monic* if its coefficient of $x^n$ is one.

*Remark.* For theoretical simplicity, the usual normalisation for orthogonal polynomials is that they are monic. However the standard definitions of some famous orthogonal polynomials (e.g. the Chebyshev polynomials) satisfy other scalings.

**Theorem 2.3.** *Given a vector space $\mathbb{V}$ containing $\mathbb{P}_n[x]$ $\forall n \geqslant 0$ and a scalar product defined on $\mathbb{V}$, there exists a unique monic orthogonal polynomial of degree $n$ $\forall n \geqslant 0$. In addition, $\{p_k\}_{k=0}^n$ form a basis for $\mathbb{P}_n[x]$ $\forall n \geqslant 0$.*

*Proof.* We prove both parts of the theorem simultaneously by induction over $n$: first noting that $p_0(x) \equiv 1$ starts the induction for $n = 0$.

Suppose $\{p_k\}_{k=0}^n$ satisfies the induction hypothesis.

a) By choosing any monic $q_{n+1} \in \mathbb{P}_{n+1}[x]$, e.g. $q_{n+1}(x) \equiv x^{n+1}$, we can construct $p_{n+1} \in \mathbb{P}_{n+1}[x]$ by the *Gram–Schmidt algorithm*, i.e.

$$p_{n+1} = q_{n+1} - \sum_{k=0}^{n} \frac{\langle q_{n+1}, p_k \rangle}{\langle p_k, p_k \rangle} p_k\,. \tag{2.6}$$

Since (by construction) this $p_{n+1}$ is monic and satisfies $\langle p_{n+1}, p_m \rangle$ $\forall m \leqslant n$, it must also satisfy $\langle p_{n+1}, p \rangle$ $\forall p \in \mathbb{P}_n[x]$ as required.

b) For uniqueness, we follow the usual contradiction argument and assume that $p_{n+1}, \hat{p}_{n+1} \in \mathbb{P}_{n+1}[x]$ are both monic orthogonal polynomials. But then $r \equiv p_{n+1} - \hat{p}_{n+1} \in \mathbb{P}_n[x]$ and so

$$\langle r, r \rangle = \langle p_{n+1}, r \rangle - \langle \hat{p}_{n+1}, r \rangle = 0$$

shows that $r \equiv 0$.

c) Finally, to show that $\{p_k\}_{k=0}^{n+1}$ is a basis for $\mathbb{P}_{n+1}[x]$, we note that each $p \in \mathbb{P}_{n+1}[x]$ can be written uniquely in the form

$$p = c\,p_{n+1} + q \qquad c \in \mathbb{R}, q \in \mathbb{P}_n[x]\,,$$

where $c$ is the coefficient of $x^{n+1}$ in $p$. But, according to the induction hypothesis, $\{p_k\}_{k=0}^{n}$ is already a basis for $\mathbb{P}_n[x]$ and so our required result follows.

□

## 2.3 The three-term recurrence relation

As a practical method of construction, the Gram–Schmidt algorithm (2.6) in the proof of Theorem 2.3 can be considerably improved by making the clever choice of $q_{n+1}(x) \equiv xp_n(x)$ there. (Note that the monic orthogonal polynomial $p_n \in \mathbb{P}_n$ is available before $q_{n+1}$ is required.) We also need to use the fact that both (2.3) and (2.4), the scalar products we are interested in, satisfy

$$\langle xf, g\rangle = \langle f, xg\rangle \qquad \forall f, g, xf, xg \in \mathbb{V}\,. \tag{2.7}$$

**Theorem 2.4.** *Under the conditions of Theorem 2.3, our unique monic orthogonal polynomials can be generated by the three-term recurrence relation*

$$p_0(x) \equiv 1\,, \quad p_1(x) = [x - \alpha_0]p_0(x)\,, \tag{2.8a}$$

$$p_{n+1}(x) = [x - \alpha_n]p_n(x) - \beta_n p_{n-1}(x) \qquad n \geqslant 1\,, \tag{2.8b}$$

*where*

$$\alpha_n \equiv \frac{\langle p_n, xp_n\rangle}{\langle p_n, p_n\rangle} \quad n \geqslant 0 \qquad \textit{and} \qquad \beta_n \equiv \frac{\langle p_n, p_n\rangle}{\langle p_{n-1}, p_{n-1}\rangle} > 0 \quad n \geqslant 1\,. \tag{2.8c}$$

*Proof.* Obviously the unique monic polynomial of degree 0 must be $p_0(x) \equiv 1$. Also, by construction, our given $p_1 \in \mathbb{P}_1[x]$ is monic and satisfies

$$\langle p_1, p_0\rangle = \langle xp_0, p_0\rangle - \frac{\langle p_0, xp_0\rangle}{\langle p_0, p_0\rangle}\langle p_0, p_0\rangle = 0\,,$$

so it must be the unique monic orthogonal polynomial for $n = 1$.

For $n \geqslant 1$, using $q_{n+1} \equiv xp_n$ in the proof of Theorem 2.3 gives

$$\begin{aligned} p_{n+1} &= x\,p_n - \sum_{k=0}^{n} \frac{\langle x\,p_n, p_k\rangle}{\langle p_k, p_k\rangle}\,p_k \\ &= [x - \alpha_n]\,p_n - \frac{\langle p_n, x\,p_{n-1}\rangle}{\langle p_{n-1}, p_{n-1}\rangle}\,p_{n-1}\,, \end{aligned}$$

since $x\,p_k \in \mathbb{P}_{k+1}[x]$ means that the other terms in the sum are zero. But then

$$\begin{aligned} \langle p_n, x\,p_{n-1}\rangle &= \langle p_n, p_n + q\rangle \quad q \in \mathbb{P}_{n-1}[x] \\ &= \langle p_n, p_n\rangle \end{aligned}$$

gives the required result. □

## 2.4 Examples

Four of the most famous sets of orthogonal polynomials are illustrated in Table 2; note that none of them are monic. In particular, the table lists the different intervals $[a, b] \subset \mathbb{R}$ and weight functions $w$ that define the scalar product (2.3). For plots of these polynomials, see the *Orthogonal Polynomials* demonstration at

http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php

| Name | Notation | Interval | $w(x)$ | Recurrence |
|---|---|---|---|---|
| Legendre | $P_n$ | $[-1,1]$ | $1$ | $(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)$ |
| Chebyshev | $T_n$ | $[-1,1]$ | $\frac{1}{\sqrt{1-x^2}}$ | $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$ |
| Laguerre | $L_n$ | $[0,\infty)$ | $\mathrm{e}^{-x}$ | $(n+1)L_{n+1}(x) = (2n+1-x)L_n(x) - nL_{n-1}(x)$ |
| Hermite | $H_n$ | $(-\infty,\infty)$ | $\mathrm{e}^{-x^2}$ | $H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x)$ |

**Table 2:** *Some famous orthogonal polynomials*

We also note how the Chebyshev polynomials based on the scalar product

$$\langle f, g\rangle \equiv \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)g(x)\,\mathrm{d}x \tag{2.9}$$

link up with the previous definition

$$T_n(x) = \cos n\theta, \quad x = \cos\theta, \quad \theta \in [0,\pi] \tag{2.10}$$

in (1.16). This simply requires the change of variable $x = \cos\theta$ in (2.9), i.e.

$$\begin{aligned} \langle T_n, T_m\rangle &\equiv \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_n(x)T_m(x)\,\mathrm{d}x \\ &= \int_0^{\pi} \cos n\theta \cos m\theta\,\mathrm{d}\theta \\ &= \tfrac{1}{2}\int_0^{\pi} \{\cos(n+m)\theta + \cos(n-m)\theta\}\,\mathrm{d}\theta = 0 \quad \text{if } m \neq n\,. \end{aligned}$$

## 2.5 Least-squares polynomial approximation

Let us return to our original problem of curve fitting: suppose that we wish to fit a polynomial $p \in \mathbb{P}_n[x]$ to a function $f(x)$ for $a \leqslant x \leqslant b$ or to function/data values $f(\xi_j)$ for $j = 1, 2, \ldots, m > n+1$. Then it is often a good idea to choose $p \in \mathbb{P}_n[x]$ by minimizing the least-squares expression

$$\int_a^b w(x)[f(x) - p(x)]^2\,\mathrm{d}x \quad \text{or} \quad \sum_{j=1}^{m} w_j[f(\xi_j) - p(\xi_j)]^2 \tag{2.11}$$

respectively, which are linked to the scalar products (2.3) and (2.4). This is an alternative to polynomial interpolation and is called (*weighted*) *least-squares approximation*. The generality of the scalar product is a useful flexibility in both the continuous and discrete cases. Note that our $p \in \mathbb{P}_n[x]$ achieves the least value of the distance (or *norm*) $\|f - p\| \equiv \langle f - p, f - p\rangle^{1/2}$.

The next theorem shows us that, once we have available the set of orthogonal polynomials with respect to the scalar product that we are using, it is simple to write down a formula for the unique solution of the least-squares approximation problem.

**Theorem 2.5.** *Let $\{p_k\}_{k=0}^n$ be the set of orthogonal polynomials with respect to the scalar product chosen in* (2.11)*, then the least-squares approximant $\hat{p}_n \in \mathbb{P}_n[x]$ is unique and given by the formula*

$$\hat{p}_n = \sum_{k=0}^{n} \hat{c}_k p_k\,, \qquad \textit{where} \quad \hat{c}_k \equiv \frac{\langle f, p_k\rangle}{\|p_k\|^2} \quad \textit{for } k = 0, \ldots, n. \tag{2.12a}$$

*Additionally, the minimum error achieved is*

$$\|f - \hat{p}_n\|^2 = \|f\|^2 - \sum_{k=0}^{n} \frac{\langle f, p_k\rangle^2}{\|p_k\|^2}\,. \tag{2.12b}$$

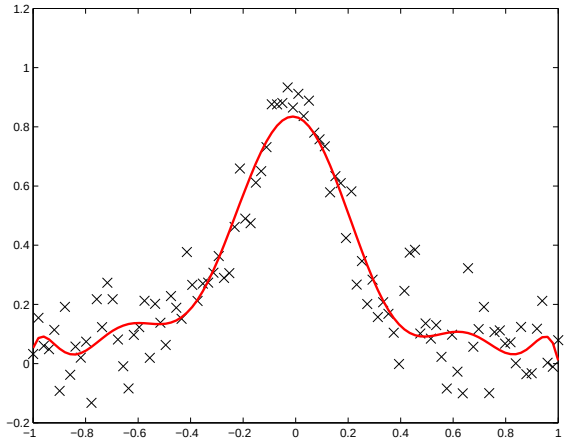



**Figure 2.1:** Least squares fit of a polynomial to discrete data.

*Proof.* Since $\{p_k\}_{k=0}^n$ is a basis for $\mathbb{P}_n[x]$, our least-squares approximant is obtained by substituting $q \equiv \sum_{k=0}^n c_k p_k$ into $\|f-q\|^2$ and minimising with respect to $\{c_k\}_{k=0}^n \subset \mathbb{R}$; i.e.

$$\begin{aligned} \langle f-q, f-q\rangle &= \langle f,f\rangle - 2\langle f,q\rangle + \langle q,q\rangle \\ &= \|f\|^2 - 2\sum_{k=0}^n c_k\langle f,p_k\rangle + \sum_{k=0}^n c_k^2\|p_k\|^2\,, \end{aligned} \tag{2.13}$$

where we note that orthogonality removes the cross-terms from $\langle q,q\rangle$.

This last expression in (2.13) is a sum of quadratic functions for each $c_k$ separately and so is minimised by setting

$$\frac{\partial}{\partial c_k}\langle f-q,f-q\rangle = -2\langle p_k,f\rangle + 2c_k\langle p_k,p_k\rangle,$$

to zero for $k=0,1,\ldots,n$; hence we obtain our formula (2.12a). (What we have constructed must be a unique minimum since the coefficients of $\{c_k^2\}_{k=0}^n$ in (2.13) are all strictly positive.) Finally, by substituting $\{\hat{c}_k\}_{k=0}^n$ into (2.13) we obtain (2.12b). □

*Remarks.*

(i) Because our least-squares approximation (2.12a) satisfies

$$\|\hat{p}_n\|^2 \equiv \langle \hat{p}_n,\hat{p}_n\rangle = \sum_{k=0}^n \hat{c}_k^2\,\langle p_k,p_k\rangle = \sum_{k=0}^n \frac{\langle f,p_k\rangle^2}{\|p_k\|^2}\,, \tag{2.14a}$$

another way of understanding the error formula (2.12b) is that it corresponds to

$$\|f-\hat{p}_n\|^2 + \|\hat{p}_n\|^2 = \|f\|^2\,. \tag{2.14b}$$

This is an analogue of the theorem of Pythagoras because

$$\langle f-\hat{p}_n,p_k\rangle = 0 \quad k=0,\ldots,n \qquad \rightarrow \qquad \langle f-\hat{p}_n,q\rangle = 0 \quad \forall q\in\mathbb{P}_n[x] \tag{2.14c}$$

and so in particular $\langle f-\hat{p}_n,\hat{p}_n\rangle = 0$.

(ii) Our error formula (2.12b) also helps us choose $n$ so that $\|f-\hat{p}_n\|^2 < \varepsilon$, where $\varepsilon>0$ is some desired *error tolerance*. Since the sequence $\{\sigma_n\}_{n=0}^\infty$, where

$$\sigma_n \equiv \sum_{k=0}^n \frac{\langle f,p_k\rangle^2}{\langle p_k,p_k\rangle} \quad n=0,1,\ldots\,, \tag{2.15}$$

is monotonic, we can gradually increase $n$ until (hopefully)

$$\sigma_n > \|f\|^2 - \varepsilon\,.$$

(Note that the construction and evaluation of both $p_k$ and $\hat{c}_k$ depends only on orthogonal polynomials of degree $k$ or less. It follows that it is not necessary to know the final value of $n \equiv n(\varepsilon)$ when one begins to form the sums in either (2.12a) or (2.15).)

(iii) Without further technical restrictions, we cannot deduce the *Parseval identity*

$$\sum_{k=0}^{\infty} \frac{\langle f, p_k \rangle^2}{\langle p_k, p_k \rangle} = \langle f, f \rangle \,; \tag{2.16}$$

which is necessary for the above error tolerance to be achievable $\forall \varepsilon > 0$. (Although monotonicity of $\{\sigma_n\}$ means that the infinite sum in (2.16) must converge.) However, when our vector space $\mathbb{V} \equiv C[a,b]$ for a finite interval and our scalar product is (2.3), this result follows easily from a famous theorem of Weierstrass (unproved).

**Theorem 2.6** (Weierstrass approximation theorem)**.** *Given $f \in C[a,b]$ and $\varepsilon > 0$, we can find $\tilde{p}_n \in \mathbb{P}_n[x]$ (for $n$ sufficiently large) so that*

$$\|f - \tilde{p}_n\|_\infty \equiv \max_{x \in [a,b]} |f(x) - \tilde{p}_n(x)| < \varepsilon \,.$$

Combining this theorem with the scalar product (2.3), we know that (for any given $\varepsilon > 0$) it is always possible to find $\tilde{p}_n \in \mathbb{P}_n[x]$ (for $n$ sufficiently large) such that

$$\langle f - \tilde{p}_n, f - \tilde{p}_n \rangle^{1/2} \leqslant \left( \|w\|_1 \|f - \tilde{p}_n\|_\infty^2 \right)^{1/2} \leqslant \|w\|_1^{1/2} \varepsilon \,,$$

where $\|w\|_1 \equiv \int_a^b w(x)\,\mathrm{d}x$. If however the Parseval identity (2.16) is not true, we can choose $\varepsilon > 0$ so that

$$0 < \|w\|_1^{1/2} \varepsilon < \left( \|f\|^2 - \sum_{k=0}^{\infty} \frac{\langle f, p_k \rangle^2}{\|p_k\|^2} \right)^{1/2} .$$

But then we have a contradiction with the minimising property of $\hat{p}_n$, i.e.

$$\left( \|f\|^2 - \sum_{k=0}^{\infty} \frac{\langle f, p_k \rangle^2}{\|p_k\|^2} \right)^{1/2} \leqslant \langle f - \hat{p}_n, f - \hat{p}_n \rangle^{1/2} \leqslant \langle f - \tilde{p}_n, f - \tilde{p}_n \rangle^{1/2} \leqslant \|w\|_1^{1/2} \varepsilon \,.$$

(iv) We have spent most of this subsection looking at least-squares polynomial approximation with respect to the continuous scalar product (2.3). Least-squares problems with respect to the discrete scalar product (2.4) often arise in practice for a different reason: i.e. $\{f_i\}_{i=1}^m$ correspond to *inexact experimental* data and we wish to construct a low-degree ($n \ll m$) polynomial that *smooths* out the data errors. An example of an acceptable solution is shown in Figure 2.1.

# 3 Approximation of Linear Functionals

In the previous two sections, we have considered the approximation of functions by polynomials. In the present section, we shall again use polynomials to construct approximation formulae, but now for *linear functionals*.

## 3.1 Linear functionals

In pure functional analysis, a linear functional is a linear mapping from a given vector space to the underlying field of scalars. In this course, we shall be slightly more restrictive.

**Definition 3.1.** If $\mathbb{V}$ is a given real vector space of functions, a linear functional on $\mathbb{V}$ is a linear mapping $L : \mathbb{V} \mapsto \mathbb{R}$.

Below are some simple examples, where it is easy to verify that $L$ is a linear functional on $\mathbb{V}$.

(i) $\mathbb{V} = C[a, b]$ and $L(f) \equiv f(\xi)$ for some fixed $\xi \in [a, b]$.

(ii) $\mathbb{V} = C[a, b]$ with $[a, b]$ finite and

$$L(f) \equiv \int_a^b f(x)w(x)\,\mathrm{d}x\,, \tag{3.1a}$$

where the fixed weight function $w$ is integrable over $[a, b]$.

(iii) $\mathbb{V} = C^1[a, b]$ and $L(f) \equiv f'(\eta)$ for some fixed $\eta \in [a, b]$.

(iv) $\mathbb{V} = C^1[a, b]$ and

$$L(f) \equiv f(\beta) - f(\alpha) - \tfrac{\beta-\alpha}{2}\,[f'(\beta) + f'(\alpha)] \tag{3.1b}$$

for some fixed distinct $\alpha, \beta \in [a, b]$.

We shall be especially interested in approximating more complicated linear functionals (usually definite integrals or derivative point-values of functions) by simpler linear functionals (usually linear combinations of function point-values): e.g.

$$L(f) \approx \sum_{i=0}^{N} a_i f(x_i)\,, \tag{3.2}$$

where $\mathbb{V} = C^p[a, b]$ for some $p \geqslant 0$ and $\{x_i\}_{i=0}^N$ are given or chosen distinct points in $[a, b]$. $\{a_i\}_{i=0}^n$ are always chosen in some way so that the approximation (3.2) is acceptable $\forall f \in \mathbb{V}$.

*Remark.* If $L$ is defined by (3.1a), (3.2) is called *numerical integration* (or *quadrature*). (Sometimes we can also consider approximating integrals over infinite intervals.) If $L$ is defined by

$$L(f) \equiv f^{(k)}(\xi)\,, \qquad \text{for fixed } \xi \in [a, b] \text{ and } 1 \leqslant k \leqslant p\,, \tag{3.3}$$

then (3.2) is called *numerical differentiation*.

We can now describe two sensible methods for choosing the $\{a_i\}_{i=0}^N$ (and possibly also the $\{x_i\}_{i=0}^N$) so that the approximation (3.2) is "good".

**Method 3.2.** *Interpolating formulae.* $\{x_i\}_{i=0}^N \subset [a, b]$ are allowed to be an arbitrary set of distinct points, but then $\{a_i\}_{i=0}^N$ are derived from the Lagrange interpolation formula (1.1b): i.e.

$$L(f) \approx L\left(\sum_{i=0}^{N} f(x_i)\ell_i\right) = \sum_{i=0}^{N} L(\ell_i)f(x_i). \tag{3.4}$$

Here $\{\ell_i\}_{i=0}^N$ are the Lagrange cardinal polynomials (1.1a) with respect to $\{x_i\}_{i=0}^N$ and so (by linearity) we obtain the formulae $a_i = L(\ell_i)$ for $i = 0, \ldots, N$. The accuracy of (3.4) relies on the accuracy of polynomial interpolation and an immediate result is that

$$f \in \mathbb{P}_N[x] \quad \Rightarrow \quad L(f) = \sum_{i=0}^{N} a_i f(x_i) \ :$$

i.e. (3.4) is *exact* for $f \in \mathbb{P}_N[x]$.

**Method 3.3.** *Superaccurate formulae.* We try to choose both $\{x_i\}_{i=0}^N$ and $\{a_i\}_{i=0}^N$ so that (3.2) is exact for $f \in \mathbb{P}_{2N+1}[x]$. Since we have $2N+2$ free parameters available and $\dim(\mathbb{P}_{2N+1}[x]) = 2N+2$, this approach is feasible; but whether it is actually achievable or not depends on the particular linear functional $L$ we are trying to approximate.

## 3.2 Numerical integration

Let us first consider the simpler idea of Method 3.2 above applied to (3.1a): thus we have $\mathbb{V} = C[a,b]$, for some finite interval $[a,b]$, and

$$L(f) \equiv \int_a^b w(x) f(x)\,\mathrm{d}x\,, \tag{3.5}$$

with $w$ integrable over $[a,b]$. If $\{x_i\}_{i=0}^N \subset [a,b]$ are distinct points, then

$$a_i \equiv \int_a^b w(x)\ell_i(x)\,\mathrm{d}x \quad i = 0, \ldots, N\,, \tag{3.6}$$

where $\{\ell_i\}_{i=0}^N$ are the Lagrange cardinal polynomials with respect to $\{x_i\}_{i=0}^N$. Hence our approximation

$$\int_a^b w(x) f(x)\,\mathrm{d}x \approx \sum_{i=0}^{N} a_i f(x_i) \tag{3.7}$$

will be exact when $f \in \mathbb{P}_N[x]$. In certain commonly-occurring situations, we can slightly improve this result.

**Theorem 3.4.** *We make the following three restrictions.*

- *$N$ is even.*
- *The weight function $w$ is an even function with respect to $[a,b]$, i.e. $w(x - \frac{a+b}{2})$ is an even function for $x \in [\frac{a-b}{2}, \frac{b-a}{2}]$.*
- *$\{x_i\}_{i=0}^N$ are symmetrically placed in $[a,b]$, i.e.*

$$x_{N/2} = \tfrac{a+b}{2} \qquad \text{and} \qquad x_k + x_{N-k} = a+b \quad \text{for} \quad k = 0, \ldots, N/2-1\,.$$

*Then our coefficients in* (3.6) *satisfy*

$$a_k = a_{N-k} \qquad k = 0, \ldots, N/2-1 \tag{3.8}$$

*and our approximation* (3.7) *is exact for $f \in \mathbb{P}_{N+1}[x]$.*

*Proof.* Since $\{x_i\}_{i=0}^N$ are symmetrically placed in $[a,b]$, the nodal polynomial

$$\omega(x) \equiv \prod_{i=0}^{N} (x - x_i)\,,$$

as defined in (1.2a), is an odd function with respect to $[a,b]$; thus $w'$ is an even function with respect to $[a,b]$. If we then use (1.2b) to write our Lagrange cardinal polynomials in terms of $\omega$, (3.6) becomes

$$a_i \equiv \int_a^b w(x) \tfrac{\omega(x)}{\omega'(x_i)[x-x_i]} \,\mathrm{d}x \quad i = 0, \ldots, N\,.$$

Thus for $k = 0, \ldots, N/2-1$,

$$\begin{aligned} a_{N-k} - a_k &= \int_a^b w(x) \left\{ \tfrac{\omega(x)}{\omega'(x_{N-k})[x-x_{N-k}]} - \tfrac{\omega(x)}{\omega'(x_k)[x-x_k]} \right\} \mathrm{d}x \\ &= \tfrac{x_{N-k}-x_k}{d_k} \int_a^b w(x) \tfrac{\omega(x)}{(x-x_{N-k})(x-x_k)} \,\mathrm{d}x \qquad \text{where} \quad d_k \equiv \omega'(x_k) = \omega'(x_{N-k}) \end{aligned}$$

must be zero because the integrand is odd with respect to $[a,b]$.

The main part of the theorem is now simple and relies on the following decomposition: given any $f \in \mathbb{P}_{N+1}[x]$, there is a unique $c \in \mathbb{R}$ and unique $q \in \mathbb{P}_N[x]$ such that

$$f(x) = c\,(x - \tfrac{a+b}{2})^{N+1} + q(x)\,, \tag{3.9}$$

where $c$ is the leading coefficient of $f$. Hence we only have to combine the following three results:

$$\int_a^b w(x) f(x) \,\mathrm{d}x = \int_a^b w(x) q(x) \,\mathrm{d}x\,, \tag{3.10a}$$

$$\sum_{i=0}^N a_i f(x_i) = \sum_{i=0}^N a_i q(x_i)\,, \tag{3.10b}$$

$$\int_a^b w(x) q(x) \,\mathrm{d}x = \sum_{i=0}^N a_i q(x_i)\,. \tag{3.10c}$$

(3.10a) is true because the first term on the right-hand side of (3.9) has been chosen to be odd with respect to $[a,b]$. This fact, together with (3.8), is also used to establish (3.10b). Finally, (3.10c) is true because we already know that (3.7) is exact on $\mathbb{P}_N[x]$. □

In practice, the restriction that the weight function $w$ must be even usually occurs because it is constant. The midpoint and Simpson rules in §3.2.2 rely on this theorem, with $N = 0$ and $N = 2$ respectively.

Not surprisingly, there is an analogous result with the properties of $w$ and $\omega$ reversed.

**Theorem 3.5.** *We make the following three restrictions.*

- *$N$ is odd.*
- *The weight function $w$ is an odd function with respect to $[a,b]$, i.e. $w(x - \frac{a+b}{2})$ is an odd function for $x \in [\frac{a-b}{2}, \frac{b-a}{2}]$.*
- *$\{x_i\}_{i=0}^N$ are symmetrically placed in $[a,b]$, i.e.*

$$x_k + x_{N-k} = a + b \quad \textit{for} \quad k = 0, \ldots, (N-1)/2\,.$$

*Then our coefficients in* (3.6) *satisfy*

$$a_k = -a_{N-k} \qquad k = 0, \ldots, (N-1)/2$$

*and our approximation* (3.7) *is exact for $f \in \mathbb{P}_{N+1}[x]$.*

We do not give a proof of this second theorem, because it is so similar to the proof of the first, e.g. the nodal polynomial $\omega$ is now even with respect to $[a,b]$. The second theorem appears less often in practice, because the condition on the weight function $w$ occurs less often.

### 3.2.1 Gaussian quadrature

In the present subsection, we explain how the superaccurate formulae of Method 3.3 are often achievable for numerical integration. To do this we require the theory of orthogonal polynomials developed in §2 and so we must have an appropriate scalar product. Thus the linear functional we are now trying to approximate is

$$\int_a^b w(x)f(x)\,\mathrm{d}x \tag{3.11}$$

and the weight function $w$ is restricted so that

$$\langle f,g\rangle \equiv \int_a^b w(x)f(x)g(x)\,\mathrm{d}x \tag{3.12}$$

is a scalar product on $\mathbb{P}_n[x]$ for all $n \geqslant 0$. (The restrictions necessary on $w$, both for finite and infinite intervals, are discussed in §2.1.) Of course, we can also only consider functions $f$ such that the integral in (3.11) exists.

We shall also slightly change our notation in this subsection, in order to conform to standard practice for Gauss quadrature formulae: i.e. our approximation is

$$\int_a^b w(x)f(x)\,\mathrm{d}x \approx \sum_{k=1}^{\nu} b_k f(c_k)\,, \tag{3.13}$$

with *nodes* (or *knots*) $\{c_k\}_{k=1}^{\nu}$ and *weights* $\{b_k\}_{k=1}^{\nu}$. Thus to achieve Method 3.3, we must show how the nodes and weights can be chosen so that (3.13) is exact for $f \in \mathbb{P}_{2\nu-1}[x]$. Before proceeding with this, however, we prove that it is impossible to do any better.

**Claim 3.6.** *No choice of* $\{c_k\}_{k=1}^{\nu} \subset \mathbb{R}$ *can make* (3.13) *exact for* $f \in \mathbb{P}_{2\nu}[x]$.

*Proof.* There is a simple proof by contradiction. Let $c_1,\ldots,c_\nu$ be arbitrary nodes and define $q \in \mathbb{P}_\nu[x]$ by

$$q(x) \equiv \prod_{k=1}^{\nu}(x-c_k)\,.$$

But then

$$\int_a^b w(x)q^2(x)\,\mathrm{d}x > 0 \quad \text{and} \quad \sum_{k=1}^{\nu} b_k q^2(c_k) = 0$$

for any choice of weights $b_1,\ldots,b_\nu$. Hence the integral and the quadrature do not match. □

It will turn out that the nodes $\{c_k\}_{k=1}^{\nu}$ necessary to achieve Method 3.3 are the zeros of $p_\nu$, the orthogonal polynomial of degree $\nu$ with respect to the scalar product (3.12). We must first show, however, that these zeros have the appropriate properties. (The following theorem is a generalisation of (1.17b).)

**Theorem 3.7.** *For* $n \geqslant 1$, *all the zeros of* $p_n$ *are real, distinct and lie in the interval* $(a,b)$.

*Proof.* Since $p_0$ is constant, we know that

$$\int_a^b w(x)p_n(x)\,\mathrm{d}x = \int_a^b w(x)p_0(x)p_n(x)\,\mathrm{d}x = \langle p_0,p_n\rangle = 0$$

for $n \geqslant 1$ by orthogonality: hence $p_n$ has at least one change of sign, i.e. at least one zero, in $(a,b)$. Now we assume that $p_n$ changes sign at $\{\xi_j\}_{j=1}^{m} \subset (a,b)$ and prove by contradiction that $m < n$ is impossible. If we define

$$q(x) \equiv \prod_{j=1}^{m}(x-\xi_j) \in \mathbb{P}_m[x]\,.$$

then $q(x)p_n(x)$ does not change sign in $(a,b)$ and so $\langle q,p_n\rangle$ cannot be zero. (Since it is a non-zero polynomial, $qp_n$ can only be zero at a finite number of points.) On the other hand, if $m < n$ we must have $\langle q,p_n\rangle = 0$ by orthogonality: thus we have our contradiction.

Since $p_n$ can have at most $n$ real zeros, we conclude that $p_n$ changes sign $n$ times in $(a,b)$ and that these points are real, distinct zeros. □

Given any choice of nodes $\{c_k\}_{k=1}^{\nu}$, we already know from (3.6) how to choose the weights $\{b_k\}_{k=1}^{\nu}$ so that (3.13) is exact for $f \in \mathbb{P}_{\nu-1}[x]$. The following theorem simply restates this result using our current notation.

**Theorem 3.8.** *If $\{\ell_k\}_{k=1}^{\nu}$ are the Lagrange cardinal polynomials with respect to the nodes $\{c_k\}_{k=1}^{\nu}$, then choosing*

$$b_k \equiv \int_a^b w(x)\ell_k(x)\,\mathrm{d}x, \qquad k = 1, \ldots, \nu \tag{3.14}$$

*means that* (3.13) *is exact when $f \in \mathbb{P}_{\nu-1}[x]$.*

*Proof.* If $f \in \mathbb{P}_{\nu-1}[x]$ then

$$\int_a^b w(x)f(x)\,\mathrm{d}x = \int_a^b w(x)\left\{\sum_{k=1}^{\nu} f(c_k)\ell_k(x)\right\}\,\mathrm{d}x = \sum_{k=1}^{\nu} b_k f(c_k)\,.$$

□

*Example: Trapezoidal Rule.* When $[a, b]$ is finite and $w \equiv 1$ in (3.12), then, for $\nu = 2$ with $c_1 \equiv a$ and $c_2 \equiv b$, we obtain

$$b_1 \equiv \int_a^b \frac{x-b}{a-b}\,\mathrm{d}x = \tfrac{1}{2}(b-a) = \int_a^b \frac{x-a}{b-a}\,\mathrm{d}x \equiv b_2\,.$$

We finally show that the result in Theorem 3.8 can be much improved (i.e. can achieve Method 3.3) when the nodes are specially chosen.

**Theorem 3.9.** *If $\{c_k\}_{k=1}^{\nu}$ are the zeros of $p_\nu$ and $\{b_k\}_{k=1}^{\nu}$ are chosen as in Theorem 3.8, then* (3.13) *is exact for $f \in \mathbb{P}_{2\nu-1}[x]$. In addition, $b_k > 0$ for $k = 1, \ldots, \nu$ (i.e. all the weights are positive).*

*Proof.* We make use of the following key decomposition: for any $f \in \mathbb{P}_{2\nu-1}[x]$, there exist unique $p, q \in \mathbb{P}_{\nu-1}[x]$ such that $f = qp_\nu + r$. On the one hand, this decomposition simplifies the integral because

$$\int_a^b w(x)f(x)\,\mathrm{d}x = \int_a^b w(x)\left\{q(x)p_\nu(x) + r(x)\right\}\,\mathrm{d}x = \int_a^b w(x)r(x)\,\mathrm{d}x$$

by orthogonality. On the other hand, it also simplifies the quadrature because

$$\sum_{k=1}^{\nu} b_k f(c_k) = \sum_{k=1}^{\nu} b_k\left\{q(c_k)p_\nu(c_k) + r(c_k)\right\} = \sum_{k=1}^{\nu} b_k r(c_k)$$

by the special choice of nodes. Hence (3.13) is exact for $f$, because we already know from Theorem 3.8 that it is exact for $r$.

The final part of the theorem follows from (3.13) being exact when $f$ is replaced by any of $\{\ell_k^2\}_{k=1}^{\nu} \in \mathbb{P}_{2\nu-2}[x]$, where $\{\ell_k\}_{k=1}^{\nu}$ are the Lagrange cardinal polynomials with respect to our special choice of nodes: i.e. for $k = 1, \ldots, \nu$ we have

$$0 < \int_a^b w(x)\ell_k^2(x)\,\mathrm{d}x = \sum_{j=1}^{\nu} b_j \ell_k^2(c_j) = \sum_{j=1}^{\nu} b_j \delta_{jk} = b_k\,.$$

□

**Definition 3.10.** A quadrature with $\nu$ nodes that is exact on $\mathbb{P}_{2\nu-1}$ is called *Gaussian quadrature.*

### 3.2.2 Examples

The four famous sets of orthogonal polynomials mentioned in §2.4 (Legendre, Chebyshev, Laguerre, Hermite) all generate important Gauss quadrature formulae for different intervals and weight functions. Many other well-known numerical integration methods are not Gaussian, e.g. trapezoidal and Simpson's rules.

(i) If $[a,b]=[-1,1]$ and $w(x)\equiv 1$, the underlying orthogonal polynomials are the *Legendre polynomials*. The first few polynomials are, with the traditional non-monic normalization,

$$\begin{aligned} P_0(x) &= 1\,, \\ P_1(x) &= x\,, \\ P_2(x) &= \tfrac{3}{2}x^2-\tfrac{1}{2}\,, \\ P_3(x) &= \tfrac{5}{2}x^3-\tfrac{3}{2}x\,, \\ P_4(x) &= \tfrac{35}{8}x^4-\tfrac{15}{4}x^2+\tfrac{3}{8}\,. \end{aligned}$$

It follows that the Gaussian quadrature nodes/weights for $[a,b]=[-1,1]$ and $w(x)\equiv 1$ are

| $\nu$ | $b_k$ | $c_k\in[-1,1]$ | *Exact For* |
|---|---|---|---|
| 1 | $b_1=2$ | $c_1=0$ | $\mathbb{P}_1$ |
| 2 | $b_1=1,\ b_2=1$ | $c_1=-\sqrt{\frac{1}{3}},\ c_2=\sqrt{\frac{1}{3}}$ | $\mathbb{P}_3$ |
| 3 | $b_1=\frac{5}{9},\ b_2=\frac{8}{9},\ b_3=\frac{5}{9}$ | $c_1=-\sqrt{\frac{3}{5}},\ c_2=0,\ c_3=\sqrt{\frac{3}{5}}$ | $\mathbb{P}_5$ |
| 4 | $b_1=b_4=\frac{1}{2}+\frac{1}{6}\sqrt{\frac{5}{6}}$<br>$b_2=b_3=\frac{1}{2}-\frac{1}{6}\sqrt{\frac{5}{6}}$ | $c_1=-c_4,\ c_4=\left(\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}\right)^{\frac{1}{2}}$<br>$c_2=-c_3,\ c_3=\left(\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}\right)^{\frac{1}{2}}$ | $\mathbb{P}_7$ |

These Gauss quadrature rules can also be applied on a general finite interval $[a,b]$, by making use of the linear mapping (1.22).

(ii) For $[a,b]=[-1,1]$ and $w(x)=(1-x^2)^{-1/2}$, the orthogonal polynomials are the *Chebyshev polynomials* and the quadrature rule is

$$T_\nu(x)=\cos(\nu\arccos x),\qquad b_k=\frac{\pi}{\nu},\qquad c_k=\cos\frac{2k-1}{2\nu}\pi,\qquad k=1,\ldots,\nu.$$

Now generalising to $[a,b]$ is no longer so simple, since (1.22) alters the weight function.

(iii) Here are some simple well-known, not necessarily Gaussian, quadrature methods for a finite interval $[a,b]$ with $w\equiv 1$.

| *Rule* | $\nu$ | $b_k$ | $c_k\in[a,b]$ | *Exact For* | *Comment* |
|---|---|---|---|---|---|
| Rectangle | 1 | $b_1=(b-a)$ | $c_1=a$ or $c_1=b$ | $\mathbb{P}_0=\mathbb{P}_{\nu-1}$ | 1-point, non-Gaussian, exact on constants. |
| Midpoint | 1 | $b_1=(b-a)$ | $c_1=\frac{1}{2}(a+b)$ | $\mathbb{P}_1=\mathbb{P}_{2\nu-1}$ | 1-point, Gaussian, exact on linear fns. |
| Trapezoid(al) | 2 | $b_1=b_2=\frac{1}{2}(b-a)$ | $c_1=a,\ c_2=b$ | $\mathbb{P}_1=\mathbb{P}_{\nu-1}$ | 2-point, non-Gaussian, exact on linear fns. |
| Simpson's | 3 | $b_1=b_3=\frac{1}{6}(b-a)$<br>$b_2=\frac{2}{3}(b-a)$ | $c_1=a,\ c_3=b$<br>$c_2=\frac{1}{2}(a+b)$ | $\mathbb{P}_3$ | 3-point, non-Gaussian, exact on cubics. |

*Demonstration.* For numerical examples, see the *Gaussian Quadrature* demonstration at

http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php

*Practicalities.* If an approximation is required to an integral over a 'large' interval $[a, b]$, then often the interval will be split into $M$ sub-intervals, $[x_{i-1}, x_i]$ for $i = 1, \ldots, M$, with $x_0 = a$ and $x_M = b$. Gaussian quadrature, or another approximation, is then used in each sub-interval.

## 3.3 Numerical differentiation

In this subsection our vector space is $\mathbb{V} \equiv C^k[a, b]$, for a finite interval and some $k \geqslant 1$, and our linear functional is

$$L(f) \equiv f^{(k)}(\xi) \tag{3.15}$$

for some fixed $\xi \in [a, b]$. For $N \geqslant k$, we seek distinct $\{x_i\}_{i=0}^N \subset [a, b]$ and $\{a_i\}_{i=0}^N \subset \mathbb{R}$ so that

$$f^{(k)}(\xi) \approx \sum_{i=0}^{N} a_i f(x_i) \tag{3.16}$$

is a "good" approximation. Just as in §3.2, we can apply Method 3.2 for any choice of distinct $\{x_i\}_{i=0}^N$ by setting

$$a_i \equiv \ell_i^{(k)}(\xi) \qquad i = 0, \ldots, N\,, \tag{3.17}$$

where $\{\ell_i\}_{i=0}^N$ are the Lagrange cardinal polynomials with respect to $\{x_i\}_{i=0}^N$. Then our approximation (3.16) is exact when $f \in \mathbb{P}_N[x]$. (Note that our approximation takes a particularly simple form when $N = k$: because then (3.4) gives

$$\sum_{i=0}^{k} a_i f(x_i) = p^{(k)}(\xi) = k! f[x_0, \ldots, x_k]\,,$$

where $p \in \mathbb{P}_k[x]$ is the interpolating polynomial for $f$ with respect to $\{x_i\}_{i=0}^k$ and so we have made use of (1.10a).)

As in §3.2, we can also slightly improve the above result in special cases.

**Theorem 3.11.** *We make the following restrictions:*

- *$k$ is even and $\xi \equiv \frac{a+b}{2}$;*
- *$N$ is even and $\{x_i\}_{i=0}^N$ are symmetrically placed in $[a, b]$, i.e.*

$$x_{N/2} = \tfrac{a+b}{2} \qquad \text{and} \qquad x_i + x_{N-i} = a + b \quad \text{for} \quad i = 0, \ldots, N/2 - 1\,.$$

*Then our coefficients in* (3.17) *satisfy*

$$a_i = a_{N-i} \qquad i = 0, \ldots, N/2 - 1 \tag{3.18}$$

*and our approximation* (3.16) *is exact for $f \in \mathbb{P}_{N+1}[x]$.*

*Proof.* The proof has the same pattern as the proof of Theorem 3.4, so we do not go into detail. The nodal polynomial $\omega$ is odd with respect to $[a, b]$ and hence

$$\begin{aligned} a_{N-i} - a_i &= \frac{\mathrm{d}^k}{\mathrm{d}x^k} \left\{ \frac{\omega(x)}{\omega'(x_{N-i})[x - x_{N-i}]} - \frac{\omega(x)}{\omega'(x_i)[x - x_i]} \right\}_{x=\frac{a+b}{2}} \\ &= \frac{x_{N-i} - x_i}{d_i} \frac{\mathrm{d}^k}{\mathrm{d}x^k} \left\{ \frac{\omega(x)}{(x - x_{N-i})(x - x_i)} \right\}_{x=\frac{a+b}{2}}, \end{aligned}$$

where $d_i \equiv \omega'(x_i) = \omega'(x_{N-i})$, is zero for $i = 0, \ldots, N/2 - 1$.

We again use the decomposition (3.9) and combine the following three results:

$$f^{(k)}(\tfrac{a+b}{2}) = q^{(k)}(\tfrac{a+b}{2})\,, \tag{3.19a}$$

$$\sum_{i=0}^{N} a_i f(x_i) = \sum_{i=0}^{N} a_i q(x_i)\,, \tag{3.19b}$$

$$q^{(k)}(\tfrac{a+b}{2}) = \sum_{i=0}^{N} a_i q(x_i)\,. \tag{3.19c}$$

□

The central difference formula (3.20c) is an example of this theorem.

We just state the analogous theorem.

**Theorem 3.12.** *We make the following restrictions:*

- *$k$ is odd and $\xi \equiv \frac{a+b}{2}$;*
- *$N$ is odd and $\{x_i\}_{i=0}^N$ are symmetrically placed in $[a,b]$, i.e.*

$$x_i + x_{N-i} = a + b \quad \textit{for} \quad i = 0, \ldots, (N-1)/2\,.$$

*Then our coefficients in* (3.17) *satisfy*

$$a_i = -a_{N-i} \qquad i = 0, \ldots, (N-1)/2$$

*and our approximation* (3.16) *is exact for $f \in \mathbb{P}_{N+1}[x]$.*
The central difference formula (3.20b) is an example of this theorem.

Finally, we have to admit that no Method 3.3 formulae exist for numerical differentiation: i.e. there is no analogue of Gaussian quadrature.

### 3.3.1 Examples (*Unlectured*)

$N = k$

Forward difference, 2-point, exact on linear fns:
$$f'(x) \approx f[x, x+h] = \frac{f(x+h) - f(x)}{h}\,, \tag{3.20a}$$

Central difference, 2-point, exact on quadratics:
$$f'(x) \approx f[x-h, x+h] = \frac{f(x+h) - f(x-h)}{2h}\,, \tag{3.20b}$$

$2^{\text{nd}}$-order central difference, 3-point, exact on cubics:
$$f''(x) \approx 2f[x-h, x, x+h] = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}\,. \tag{3.20c}$$

$2 = N > k = 1$. Suppose $[a,b] = [0,2]$, although one can, of course, transform any formula to any interval.
We claim that

$$f'(0) \approx p_2'(0) = -\tfrac{3}{2} f(0) + 2f(1) - \tfrac{1}{2} f(2)\,. \tag{3.21}$$

Given the nodes $\{x_i\}_{i=0}^2$, in our case $(0,1,2)$, we can find the corresponding coefficients $\{a_i\}_{i=0}^2$ in two ways.

(i) First determine the fundamental Lagrange polynomials $\ell_i$

$$\ell_0(x) = \tfrac{1}{2}(x-1)(x-2), \quad \ell_1(x) = -x(x-2), \quad \ell_2(x) = \tfrac{1}{2}x(x-1),$$

and then use (3.17) to obtain

$$a_0 = \ell_0'(0) = -\tfrac{3}{2}, \quad a_1 = \ell_1'(0) = 2, \quad a_2 = \ell_2'(0) = -\tfrac{1}{2}.$$

(ii) However, sometimes it is easier to solve the system of linear equations which arises if we require the formula to be exact on monomials $x^j$, $j = 0, \ldots, N$ (or elements of any other basis for $\mathbb{P}_N$), i.e. if we require

$$f^{(k)}(\xi) = \sum_{i=0}^{N} a_i f(x_i) \quad \text{for} \quad f = x^j,\ j = 0, \ldots, n.$$

Hence for (3.21) and $x_i = 0, 1, 2$

$$\begin{cases} f(x) = 1: & 0 = a_0 + a_1 + a_2 \\ f(x) = x: & 1 = a_1 + 2a_2 \\ f(x) = x^2: & 0 = a_1 + 4a_2 \end{cases} \quad \Rightarrow \quad a_0 = -\tfrac{3}{2}, \quad a_1 = 2, \quad a_2 = -\tfrac{1}{2}.$$

## 3.4 Error for approximation of linear functionals

The general problem we have been considering throughout this section is the approximation of more complicated linear functionals by linear combinations of simpler linear functionals; i.e.

$$L(f) \approx \sum_{i=0}^{N} a_i L_i(f) \tag{3.22}$$

where $\{L_i\}_{i=0}^N$ and $\{a_i\}_{i=0}^N \subset \mathbb{R}$ are chosen so that (3.22) is exact when $f \in \mathbb{P}_k[x]$ for some $k \geqslant 0$. Hence, if we introduce the error for our approximation, i.e.

$$e_L(f) \equiv L(f) - \sum_{i=0}^{N} L_i(f) \tag{3.23}$$

so that $e_L$ is also a linear functional, then (3.23) will be zero when $f \in \mathbb{P}_k[x]$.

*Example.* As a very simple example that can be used for illustrative purposes throughout this section, we assume that $\alpha, \beta \in \mathbb{R}$ are two distinct points and let $L(f) \equiv f(\beta)$ be approximated by

$$f(\alpha) + \frac{\beta - \alpha}{2} \{f'(\beta) + f(\alpha)\} .$$

Hence our error functional is

$$e_L(f) \equiv f(\beta) - f(\alpha) - \frac{\beta - \alpha}{2} \{f'(\beta) + f'(\alpha)\} \tag{3.24}$$

(cf. (3.1b)) and it is easy to check that $k = 2$, i.e.

$$f \in \mathbb{P}_2[x] \Rightarrow e_L(f) = 0 .$$

Our aim in this section is to find useful formulae and bounds for our error functional $e_L$. To achieve this, we assume that our functionals $L$ and $\{L_i\}_{i=0}^N$ can be defined on the vector space $\mathbb{V} \equiv C^{k+1}[a, b]$ for a suitable finite interval $[a, b]$. (In the above example, any finite interval containing the two points $\alpha$ and $\beta$ is acceptable.) In this case, one of our error bounds will be

$$|e_L(f)| \leqslant c_L \|f^{(k+1)}\|_\infty \qquad \forall f \in C^{k+1}[a, b] \tag{3.25}$$

for some constant $c_L$; although we will also use other norms. It is also of interest whether $c_L$ is optimal, in the sense of the following definition.

**Definition 3.13.** $c_L$ is said to be *achievable* if $\exists f \in C^{k+1}[a, b]$ such that (3.25) is an equality. $c_L$ is said to be *sharp* if, for any $\epsilon > 0$, $\exists f_\epsilon \in C^{k+1}[a, b]$ such that

$$|e_L(f_\epsilon)| > (c_L - \epsilon) \|f_\epsilon^{(k+1)}\|_\infty .$$

### 3.4.1 Taylor series expansion

In order to obtain a simple formula for $e_L$, we need to use the Taylor series with integral remainder for $f \in C^{k+1}[a,b]$, i.e.

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + \cdots + \frac{(x-a)^k}{k!}f^{(k)}(a) + \frac{1}{k!}\int_a^x (x-\theta)^k f^{(k+1)}(\theta)\,\mathrm{d}\theta \quad (3.26a)$$

for $x \in [a,b]$. It is convenient to make the range of integration independent of $x$ by introducing the notation

$$(x-\theta)_+^k \equiv \begin{cases} (x-\theta)^k & x \geqslant \theta \\ 0 & x < \theta \end{cases}, \quad (3.26b)$$

so that we then have

$$f(x) = \sum_{r=0}^{k} \frac{1}{r!}(x-a)^r f^{(r)}(a) + \frac{1}{k!}\int_a^b (x-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta \qquad \forall x \in [a,b]\,. \quad (3.26c)$$

(Note that, when $k=0$, we may write $(x-\theta)_+ = H(x-\theta)$, where $H$ is the Heaviside/step function.) If $\lambda$ is a linear functional such that $\lambda(p) = 0$ for $p \in \mathbb{P}_k$, it then follows by linearity that

$$\lambda(f) = \lambda\left(\frac{1}{k!}\int_a^b (x-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta\right) \qquad \forall f \in C^{k+1}[a,b]\,. \quad (3.27)$$

So if we identify $\lambda$ with our error functional $e_L$, then we have obtained an exact error formula.

*Example.* Since $k=2$ for (3.24), in this case we obtain

$$e_L(f) = e_L\left(\tfrac{1}{2}\int_a^b (x-\theta)_+^2 f'''(\theta)\,\mathrm{d}\theta\right) \qquad \forall f \in C^3[a,b]\,. \quad (3.28)$$

### 3.4.2 Exchanging the order of $\lambda$ and integration

We will now assume that the order of action of $\int$ and $\lambda$ in (3.27) can be exchanged. Since our error functionals typically involve linear combinations of integrals and point values of functions and derivatives, this assumption will hold. The following results, involving the remainder term

$$g(x) \equiv \int_a^b (x-\theta)_+^k\, f^{(k+1)}(\theta)\;\mathrm{d}\theta\,, \quad (3.29)$$

are stated without proof.

(i) Suppose that $\lambda(f) \equiv f(\xi)$ with $\xi \in [a,b]$: then

$$\lambda(g) \equiv \left(\int_a^b (\xi-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta\right)\Bigg|_{x=\xi} = \int_a^b (\xi-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta$$
$$\equiv \int_a^b \lambda\left((x-\theta)_+^k\right) f^{(k+1)}(\theta)\,\mathrm{d}\theta\,.$$

(ii) Suppose that $\lambda(f) \equiv \int_a^b \beta(x)\,f(x)\,\mathrm{d}x$, where $\beta \in C[a,b]$: then

$$\lambda(g) \equiv \int_a^b \beta(x)\left(\int_a^b (x-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta\right)\mathrm{d}x = \int_a^b \left(\int_a^b \beta(x)(x-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}x\right)\mathrm{d}\theta$$
$$\equiv \int_a^b \lambda\left((x-\theta)_+^k\right) f^{(k+1)}(\theta)\,\mathrm{d}\theta\,.$$

(iii) Suppose that $\lambda(f) \equiv \frac{\mathrm{d}^\ell f}{\mathrm{d}x^\ell}(\eta)$, with $1 \leqslant \ell \leqslant k-1$ and $\eta \in [a,b]$: then

$$\lambda(g) \equiv \frac{\mathrm{d}^\ell}{\mathrm{d}x^\ell}\left(\int_a^b (x-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta\right)\Bigg|_{x=\eta} = \int_a^b \frac{\mathrm{d}^\ell}{\mathrm{d}x^\ell}\left((x-\theta)_+^k\right)\Big|_{x=\eta} f^{(k+1)}(\theta)\,\mathrm{d}\theta$$
$$\equiv \int_a^b \lambda\left((x-\theta)_+^k\right) f^{(k+1)}(\theta)\,\mathrm{d}\theta\,.$$

Then, for our error functionals, we can simplify (3.27) to

$$\lambda(f) = \frac{1}{k!}\int_a^b \lambda\left((x-\theta)_+^k\right) f^{(k+1)}(\theta)\,\mathrm{d}\theta \qquad \forall f \in C^{k+1}[a,b]\,, \tag{3.30}$$

noting that $\lambda$ acts on $(x-\theta)_+^k \in C^{k-1}[a,b]$ as a function of $x$ with $\theta$ held constant, and arrive at the following famous theorem.

**Theorem 3.14** (Peano kernel theorem)**.** *Let $\lambda$ be a linear functional on $C^{k+1}[a,b]$ such that $\lambda(f) = 0$ for all $f \in \mathbb{P}_k[x]$. Suppose also that $\lambda$ applied to $\int_a^b (x-\theta)_+^k f^{(k+1)}(\theta)\,d\theta$ commutes with the integration sign. Then we may write*

$$\lambda(f) = \frac{1}{k!}\int_a^b K(\theta) f^{(k+1)}(\theta)\,\mathrm{d}\theta \qquad \forall f \in C^{k+1}[a,b]\,, \tag{3.31a}$$

*where the* Peano kernel function

$$K(\theta) \equiv \lambda((x-\theta)_+^k) \quad \textit{for} \quad a \leqslant \theta \leqslant b \tag{3.31b}$$

*is independent of $f$.*

*Proof.* Since it is assumed that

$$\lambda\left(\int_a^b (x-\theta)_+^k f^{(k+1)}(\theta)\,d\theta\right) = \int_a^b \lambda\left((x-\theta)_+^k f^{(k+1)}(\theta)\right)\,d\theta\,, \tag{3.32a}$$

it follows from (3.27) and the linearity of $\lambda$ that

$$\lambda(f) = \frac{1}{k!}\int_a^b \lambda\left((x-\theta)_+^k f^{(k+1)}(\theta)\right)\,\mathrm{d}\theta = \frac{1}{k!}\int_a^b \lambda\left((x-\theta)_+^k\right) f^{(k+1)}(\theta)\,\mathrm{d}\theta\,. \tag{3.32b}$$

The formula (3.31a) follows from the definition of $K(\theta)$. □

*Examples.*

(i) We continue our previous example from (3.24) and (3.28). If we assume (without loss of generality) that $\alpha < \beta$, we may use the obvious result

$$\frac{\mathrm{d}}{\mathrm{d}x}(x-\theta)_+^k = k(x-\theta)_+^{k-1} \qquad k \geqslant 1$$

to obtain

$$\begin{aligned} K(\theta) &\equiv e_L\left((x-\theta)_+^2\right) \\ &= (\beta-\theta)_+^2 - (\alpha-\theta)_+^2 - \tfrac{1}{2}(\beta-\alpha)\,\left(2\,(\beta-\theta)_+ + 2\,(\alpha-\theta)_+\right) \\ &= \begin{cases} 0 & a \leqslant \theta \leqslant \alpha \quad \text{and} \quad \beta \leqslant \theta \leqslant b \\ (\alpha-\theta)\,(\beta-\theta) & \alpha \leqslant \theta \leqslant \beta \end{cases}. \end{aligned} \tag{3.33}$$

Hence our error formula from the Peano kernel theorem is

$$e_L(f) = \tfrac{1}{2}\int_\alpha^\beta (\alpha-\theta)\,(\beta-\theta)\,f'''(\theta)\;\mathrm{d}\theta \tag{3.34a}$$

for $f \in C^3[\alpha,\beta]$ and, after integrating by parts, we also obtain

$$e_L(f) = \tfrac{1}{2}\int_\alpha^\beta (\alpha+\beta-2\,\theta)\, f''(\theta)\ \mathrm{d}\theta\,. \tag{3.34b}$$

This last result could also have been derived by applying the Peano kernel theorem with $k=1$.

(ii) Consider the approximation (3.21), i.e.

$$f'(0) \approx -\tfrac{3}{2}f(0) + 2f(1) - \tfrac{1}{2}f(2)\,.$$

The error of this approximation is the linear functional

$$e_L(f) \equiv f'(0) + \tfrac{3}{2}f(0) - 2f(1) + \tfrac{1}{2}f(2)$$

and (as may be verified by trying $f(x) = 1, x, x^2$) $e_L(f) = 0$ for $f \in \mathbb{P}_2[x]$. Hence the Peano kernel theorem tells us that, for $f \in C^3[0,2]$,

$$e_L(f) = \tfrac{1}{2}\int_0^2 K(\theta) f'''(\theta)\,\mathrm{d}\theta\,,$$

where

$$\begin{aligned} K(\theta) &\equiv e_L((x-\theta)_+^2) \\ &= 2(0-\theta)_+ + \tfrac{3}{2}(0-\theta)_+^2 - 2(1-\theta)_+^2 + \tfrac{1}{2}(2-\theta)_+^2 \\ &= \begin{cases} -2(1-\theta)^2 + \tfrac{1}{2}(2-\theta)^2 \;=\; 2\theta - \tfrac{3}{2}\theta^2 & 0 \leqslant \theta \leqslant 1 \\ \tfrac{1}{2}(2-\theta)^2 & 1 \leqslant \theta \leqslant 2 \end{cases}. \end{aligned} \tag{3.35}$$

*Remark.* It is clear from the above examples that we can always evaluate $K(\theta)$ for any $\theta \in \mathbb{R}$. This simple idea provides a check on our calculations, because we now explain why $K(\theta) = 0$ for $\theta \notin (a,b)$. On the one hand, if $\theta \leqslant a$ then

$$K(\theta) \equiv \lambda\left((x-\theta)_+^k\right) = \lambda\left((x-\theta)^k\right) \tag{3.36}$$

and so $K(\theta) = 0$ because $f \in \mathbb{P}_k[x] \Rightarrow \lambda(f) = 0$. On the other hand, if $\theta \geqslant b$ then $\left((x-\theta)_+^k\right)$ is the zero function and so (by linearity of $\lambda$) $K(\theta) = 0$.

Indeed, we can extend this idea by letting $[\alpha,\beta] \subset [a,b]$ be the smallest subinterval such that $\lambda(f)$ is independent of $\{f(x) : a \leqslant x < \alpha\}$ and $\{f(x) : \beta < x \leqslant b\}$. By the same reasoning as above, we can conclude that $K(\theta) = 0$ for $\theta \notin (\alpha,\beta)$.

### 3.4.3 Formula for $e_L(f)$ when $K(\theta)$ does not change sign

In this commonly occurring situation, we obtain two useful extra results.

**Theorem 3.15.** *If $K$ does not change sign in $(a,b)$, then for each $f \in C^{k+1}[a,b]$ we have*

$$|e_L(f)| \leqslant \frac{1}{k!}\int_a^b |K(\theta)|\,\mathrm{d}\theta\, \|f^{(k+1)}\|_\infty \tag{3.37a}$$

*and*

$$e_L(f) = \frac{1}{k!}\int_a^b K(\theta)\,\mathrm{d}\theta\, f^{(k+1)}(\xi)\,. \tag{3.37b}$$

*In* (3.37a) *the constant is achievable (cf. Definition 3.13) and in* (3.37b) *the point $\xi \in (a,b)$ depends on $f$.*

*Proof.* The first result follows directly from (3.31a) and the constant is achieved by $f(x) \equiv x^{k+1}$ because then $f^{(k+1)}$ is constant. The second result is just a form of the integral mean value theorem: i.e. we can assume (without loss of generality) that $K \geqslant 0$ and so

$$\frac{m}{k!}\int_a^b K(\theta)\,\mathrm{d}\theta \leqslant \lambda(f) \leqslant \frac{M}{k!}\int_a^b K(\theta)\,\mathrm{d}\theta\,,$$

where

$$m \equiv \min_{x\in[a,b]} f^{(k+1)}(x) \qquad \text{and} \qquad M \equiv \max_{x\in[a,b]} f^{(k+1)}(x)\,.$$

Hence

$$m \leqslant \frac{k!\lambda(f)}{\int_a^b K(\theta)\,\mathrm{d}\theta} \leqslant M$$

and the intermediate value theorem gives the result. □

*Examples.*

(i) We continue the previous example from (3.33) and so $K \leqslant 0$ on $[\alpha,\beta]$ with $\int_\alpha^\beta K(\theta)\,\mathrm{d}\theta = -\frac{1}{6}(\beta-\alpha)^3$. Hence

$$|e_L(f)| \leqslant \tfrac{1}{12}(\beta-\alpha)^3\|f'''\|_\infty \quad \text{is achievable} \tag{3.38a}$$

and

$$e_L(f) = -\tfrac{1}{12}(\beta-\alpha)^3 f'''(\xi) \quad \text{for some} \quad \xi\in(\alpha,\beta)\,. \tag{3.38b}$$

(ii) We continue the previous example from (3.35) and so $K \geqslant 0$ with

$$\int_0^2 K(\theta)\,\mathrm{d}\theta = \int_0^1 \left(2\theta - \tfrac{3}{2}\theta^2\right)\,\mathrm{d}\theta + \int_1^2 \tfrac{1}{2}(2-\theta)^2\,\mathrm{d}\theta = \tfrac{1}{2}+\tfrac{1}{6} = \tfrac{2}{3}\,.$$

Consequently

$$!e_L(f)| \leqslant \tfrac{1}{3}\|f'''\|_\infty \quad \text{is achievable} \tag{3.39a}$$

and

$$e_L(f) = \tfrac{1}{3}f'''(\xi) \quad \text{for some} \quad \xi\in(0,2)\,. \tag{3.39b}$$

*Comment.* As an alternative to determining $\int_a^b K(\theta)\,\mathrm{d}\theta$ analytically, we can use the fact that (3.31a) is valid for all $f\in C^{k+1}[a,b]$. Hence, for the particular case $f(x)\equiv x^{k+1}$ (so that $f^{(k+1)}(x) = (k+1)!$) we deduce that

$$\int_a^b K(\theta)\;\mathrm{d}\theta = \frac{1}{k+1}\,e_L\left(x^{k+1}\right)\,. \tag{3.40}$$

Furthermore, since $e_L(p)=0 \quad \forall p\in\mathbb{P}_k[x]$, (3.40) remains true if $x^{k+1}$ is replaced by any monic $q\in\mathbb{P}_{k+1}[x]$: hence we can choose such a $q$ for which the evaluation of $e_L(q)$ is straightforward.

### 3.4.4 Bounds for $|e_L(f)|$

For a finite interval $[a,b]$, the three most important norms for $g\in C[a,b]$ are

1-norm: $$\|g\|_1 = \int_a^b |g(x)|\,\mathrm{d}x\,; \tag{3.41a}$$

2-norm: $$\|g\|_2 = \left\{\int_a^b [g(x)]^2\,\mathrm{d}x\right\}^{1/2}\,; \tag{3.41b}$$

$\infty$-norm: $$\|g\|_\infty = \max_{x\in[a,b]} |g(x)|\,. \tag{3.41c}$$

By employing the *Cauchy–Schwarz inequality*

$$\left| \int_a^b f(x)g(x)\,\mathrm{d}x \right| \leqslant \|f\|_2 \|g\|_2$$

and

$$\left| \int_a^b f(x)g(x)\,\mathrm{d}x \right| \leqslant \|f\|_\infty \int_a^b |g(x)|\,\mathrm{d}x = \|f\|_\infty \|g\|_1\,,$$

which are both valid $\forall f, g \in C[a,b]$, we may use (3.31a) to derive the three different bounds

$$|e_L(f)| \leqslant \frac{1}{k!}\|K\|_2 \|f^{(k+1)}\|_2 \tag{3.42a}$$

$$|e_L(f)| \leqslant \frac{1}{k!}\,\|K\|_\infty \int_a^b |f^{(k+1)}(\theta)|\,\mathrm{d}\theta = \frac{1}{k!}\|K\|_\infty\,\|f^{(k+1)}\|_1 \tag{3.42b}$$

$$|e_L(f)| \leqslant \frac{1}{k!}\int_a^b |K(\theta)|\,\mathrm{d}\theta\,\|f^{(k+1)}\|_\infty = \frac{1}{k!}\|K\|_1\,\|f^{(k+1)}\|_\infty\,. \tag{3.42c}$$

No sign condition on $K$ was needed to deduce these inequalities. If however $K(\theta)$ does not change sign on $[a,b]$ then the constants in (3.42a) and (3.42c) are achieved by $f(x) \equiv x^{k+1}$.

*Example.* Simpson's rule (see the table on page 22)

$$L(f) \equiv \int_{-1}^1 f(t)\;\mathrm{d}t \approx \tfrac{1}{3}f(-1) + \tfrac{4}{3}f(0) + \tfrac{1}{3}f(1)$$

is exact for quadratics (as well as cubics) and so the error formula

$$e_L(f) = \int_{-1}^1 f(t)\;\mathrm{d}t - \tfrac{1}{3}f(-1) - \tfrac{4}{3}f(0) - \tfrac{1}{3}f(1) = \tfrac{1}{2!}\int_{-1}^1 K(\theta)f'''(\theta)\;\mathrm{d}\theta$$

is valid for all $f \in C^3[-1,1]$. The Peano kernel

$$\begin{aligned} K(\theta) = e_L((t-\theta)_+^2) &= \int_{-1}^1 (t-\theta)_+^2\,\mathrm{d}x - \tfrac{1}{3}(-1-\theta)_+^2 - \tfrac{4}{3}(0-\theta)_+^2 - \tfrac{1}{3}(1-\theta)_+^2 \\ &= \begin{cases} \frac{1}{3}(1-\theta)^3 - \frac{4}{3}\theta^2 - \frac{1}{3}(1-\theta)^2 \\ \frac{1}{3}(1-\theta)^3 \qquad\quad - \frac{1}{3}(1-\theta)^2 \end{cases} = \begin{cases} -\frac{1}{3}\,\theta\,(1+\theta)^2, & \theta \in [-1,0] \\ -\frac{1}{3}\,\theta\,(1-\theta)^2, & \theta \in [0,1] \end{cases} \end{aligned}$$

changes sign at $\theta = 0$ and so

$$\|K\|_1 \equiv \int_0^2 |K(\theta)|\;\mathrm{d}\theta = \int_{-1}^0 |-\tfrac{1}{3}\,\theta\,(1+\theta)^2|\,\mathrm{d}\theta + \int_0^1 |-\tfrac{1}{3}\,\theta\,(1-\theta)^2|\,\mathrm{d}\theta = 2\cdot\tfrac{1}{3}\big(\tfrac{1}{2} - \tfrac{2}{3} + \tfrac{1}{4}\big) = \tfrac{1}{18}\,,$$

gives the error bound

$$|e_L(f)| \leqslant \tfrac{1}{2!}\,\tfrac{1}{18}\,\|f'''\|_\infty = \tfrac{1}{36}\,\|f'''\|_\infty$$

from (3.42c).

# 4 Initial Value Ordinary Differential Equations

The aim of this section is to discuss some elementary numerical methods for approximating the exact solutions of the *ordinary differential equation (ODE)*

$$\boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y}(t)), \qquad 0 \leqslant t \leqslant T \ : \tag{4.1}$$

here our data is $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^N \mapsto \mathbb{R}^N$ and $T > 0$, and we seek a solution $\boldsymbol{y} : [0, T] \mapsto \mathbb{R}^N$. (Later on, in §4.4, we shall consider some of the difficulties connected with letting $T \to \infty$.)

In order to guarantee existence and uniqueness for $\boldsymbol{y}$ in (4.1), we need both a smoothness restriction on $\boldsymbol{f}$ and to augment (4.1) with suitable side conditions. The assumption we shall make on $\boldsymbol{f}$ is slightly (but crucially!) stronger than continuity.

**Definition 4.1** (*Lipschitz continuity*)**.** $\boldsymbol{f}$ is continuous on $[0, T] \times \mathbb{R}^N$ and $\exists\, \lambda \geqslant 0$ such that

$$\|\boldsymbol{f}(t, \boldsymbol{x}) - \boldsymbol{f}(t, \hat{\boldsymbol{x}})\| \leqslant \lambda \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \qquad \forall t \in [0, T]\,, \quad \forall \boldsymbol{x}, \hat{\boldsymbol{x}} \in \mathbb{R}^N\,. \tag{4.2}$$

Any norm on $\mathbb{R}^N$ can be used in this definition, since only the value of the constant $\lambda$ would be changed.

The side condition that we add to (4.1) will always be

$$\boldsymbol{y}(0) = \boldsymbol{y}_0 \tag{4.3}$$

for a given $\boldsymbol{y}_0 \in \mathbb{R}^N$, and this is why the words "initial value" appear in the title of this section. Thus we know both $\boldsymbol{y}(t)$ and $\boldsymbol{y}'(t)$ at $t = 0$.

The numerical approximations for (4.1) and (4.3) that we will consider all have the same form: i.e. given a small time-step $h > 0$, we will construct

$$\boldsymbol{y}_n \approx \boldsymbol{y}(t_n) \qquad n = 1, 2, \ldots \tag{4.4}$$

where $t_n \equiv nh$. Thus we start with the exact solution $\boldsymbol{y}_0 = \boldsymbol{y}(0)$ and can continue while $n \leqslant \lfloor T/h \rfloor$, the integral part of $T/h$. Our hope is that the approximations in (4.4) will become more accurate if we choose a smaller value for $h$. For the sake of simplicity, $t_n - t_{n-1}$ will always be held constant in this course. For the sake of efficiency, $t_n - t_{n-1}$ is always allowed to vary in software packages: hence this will be briefly discussed in §4.5.

Finally we note that, although (4.2) is a sufficient condition for existence and uniqueness of solutions for (4.1) plus (4.3), we shall always assume that both $\boldsymbol{f}$ and the exact solution $\boldsymbol{y}$ are as smooth as we like: i.e. $\boldsymbol{f}$ and $\boldsymbol{y}$ can always be expanded in a Taylor series as desired.

## 4.1 One-step methods

In principle $\boldsymbol{y}_n$ in (4.4) could depend on $\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n-1}$ and we will consider some such schemes later. However we start by studying the relatively simple *one-step methods*.

**Definition 4.2** (*Explicit one-step methods*)**.** This is a map

$$\boldsymbol{y}_{n+1} = \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}_n) \qquad \boldsymbol{\varphi}_h : \mathbb{R} \times \mathbb{R}^N \mapsto \mathbb{R}^N \ : \tag{4.5}$$

i.e. an algorithm which allows us to compute $\boldsymbol{y}_{n+1}$ from $t_n$, $\boldsymbol{y}_n$, $h$ and $\boldsymbol{f}$ (through the ODE (4.1)).

Note that, since $\boldsymbol{y}_0$ is known from (4.3), (4.5) allows us to compute $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$.

### 4.1.1 The Euler method

This is the simplest numerical method for approximating (4.1) and (4.3), and is defined by

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n), \qquad n = 0, 1, \ldots . \tag{4.6}$$

We can see the basic idea for the case $n = 0$: the exact solution

$$\boldsymbol{y}(h) = \boldsymbol{y}(0) + h\boldsymbol{y}'(0) + \tfrac{1}{2}h^2\boldsymbol{y}''(0) + \cdots$$

is approximated by the truncated Taylor series

$$\boldsymbol{y}_1 = \boldsymbol{y}_0 + h\boldsymbol{f}(t_0, \boldsymbol{y}_0)\,,$$

because $\boldsymbol{y}'(0) = \boldsymbol{f}(t_0, \boldsymbol{y}_0)$. The same idea is used for $n \geqslant 1$ but, because $\boldsymbol{y}_1$ only approximates $\boldsymbol{y}(h)$, we must expect $\{\boldsymbol{y}_n\}$ to gradually "drift away" from $\{\boldsymbol{y}(t_n)\}$ as $n$ increases.

We now give a proof of convergence for the Euler method, but first we have to be clear what is meant by convergence (as $h \to 0$) of a general numerical method, like (4.5), to the exact solution $\boldsymbol{y}(t)$ of (4.1) and (4.3).

**Definition 4.3** (*Convergence*)**.** For each $h > 0$, our numerical method produces $\boldsymbol{y}_n$ for $n = 0, 1, \ldots, \lfloor T/h \rfloor$: the method is said to *converge* if, as $h \to 0$ and $nh \to t$,

$$\boldsymbol{y}_n \to \boldsymbol{y}(t) \qquad \text{uniformly for } t \in [0, T].$$

The key point in this definition is that we have two limiting processes: $h \to 0$ and $n \to \infty$, but constrained so that $nh \to t$.

**Theorem 4.4** (Convergence of Euler)**.** *If $\boldsymbol{e}_n \equiv \boldsymbol{y}_n - \boldsymbol{y}(t_n)$ denotes the error for Euler's method then $\exists\, c \geqslant 0$ (independent of $h$, $n$ and $t$) such that*

$$\|\boldsymbol{e}_n\| \leqslant ch\,\frac{\mathrm{e}^{\lambda T} - 1}{\lambda} \qquad 0 \leqslant n \leqslant \lfloor T/h \rfloor\,. \tag{4.7}$$

*Hence Euler's method converges.*

*Proof.* To derive (4.7), we split the proof into two parts. (This is the correct approach for any numerical approximation of initial value ordinary differential equations, not just for Euler's method!)

a) We insert the exact solution into Euler's method and obtain

$$\boldsymbol{y}(t_{n+1}) = \boldsymbol{y}(t_n) + h\boldsymbol{f}(t_n, \boldsymbol{y}(t_n)) + \boldsymbol{R}_n\,,$$

where

$$\boldsymbol{R}_n \equiv \int_{t_n}^{t_{n+1}} (t_{n+1} - \theta)\boldsymbol{y}''(\theta)\,\mathrm{d}\theta$$

is the integral remainder term in Taylor series because $\boldsymbol{y}'(t_n) = \boldsymbol{f}(t_n, \boldsymbol{y}(t_n))$. Hence

$$\|\boldsymbol{R}_n\|_\infty \leqslant ch^2 \qquad \text{with} \quad c \equiv \tfrac{1}{2}\|\boldsymbol{y}''\|_\infty\,,$$

where

$$\|\boldsymbol{y}''\|_\infty \equiv \max_{t\in[0,T]} \|\boldsymbol{y}''(t)\|_\infty\,.$$

b) Now we construct a formula for $\boldsymbol{e}_{n+1}$ in terms of $\boldsymbol{e}_n$, i.e.

$$\begin{aligned}\boldsymbol{e}_{n+1} &\equiv \boldsymbol{y}_{n+1} - \boldsymbol{y}(t_{n+1}) \\ &= \{\boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n)\} - \{\boldsymbol{y}(t_n) + h\boldsymbol{f}(t_n, \boldsymbol{y}(t_n) + \boldsymbol{R}_n\}\,.\end{aligned}$$

Hence, using the Lipschitz condition on $\boldsymbol{f}$,

$$\|\boldsymbol{e}_{n+1}\|_\infty \leqslant (1+h\lambda)\|\boldsymbol{e}_n\|_\infty + ch^2 \qquad n = 0, \ldots, \lfloor T/h \rfloor - 1$$

and so, since $\boldsymbol{e}_0 = \boldsymbol{0}$,

$$\|\boldsymbol{e}_n\|_\infty \leqslant ch^2 \sum_{j=0}^{n-1} (1+h\lambda)^j = \frac{ch}{\lambda}\left\{(1+h\lambda)^n - 1\right\} \qquad n = 0, \ldots, \lfloor T/h \rfloor .$$

Finally, we use the bound

$$1 + h\lambda \leqslant \mathrm{e}^{\lambda h} \quad \Rightarrow \quad (1+h\lambda)^n \leqslant \mathrm{e}^{\lambda h n} \leqslant \mathrm{e}^{\lambda T}$$

to obtain

$$\|\boldsymbol{e}_n\|_\infty \leqslant ch\,\frac{\mathrm{e}^{\lambda T} - 1}{\lambda} \qquad n = 0, \ldots, \lfloor T/h \rfloor .$$

The final part of the theorem relies on the *uniform continuity* of the exact solution $\boldsymbol{y}$ on $[0, T]$: i.e. given $\epsilon > 0$, $\exists\, \delta > 0$ such that for any $s, t \in [0, T]$

$$|s - t| < \delta \quad \Rightarrow \quad \|\boldsymbol{y}(s) - \boldsymbol{y}(t)\|_\infty < \epsilon .$$

Hence the triangle inequality

$$\|\boldsymbol{y}_n - \boldsymbol{y}(t)\|_\infty \leqslant \|\boldsymbol{e}_n\|_\infty + \|\boldsymbol{y}(t) - \boldsymbol{y}(t_n)\|_\infty$$

means that the limit

$$\lim_{\substack{h\to 0 \\ nh \to t}} \|\boldsymbol{y}_n - \boldsymbol{y}(t)\|_\infty = \boldsymbol{0} \tag{4.8}$$

is uniform for $t \in [0, T]$. □

We make two remarks about this proof.

- We have used the $\infty$-norm on $\mathbb{R}^N$, but any other vector norm is possible.
- We have tacitly assumed $\lambda > 0$: since $\lambda = 0$ corresponds to the relatively trivial case when $\boldsymbol{f}$ is independent of $\boldsymbol{y}$. It is easy to work through the proof in this special case and end up with

  $$\|\boldsymbol{e}_n\|_\infty \leqslant chT \qquad n = 0, \ldots, \lfloor T/h \rfloor .$$

  This links up with

  $$\lim_{\lambda \to 0} \frac{\mathrm{e}^{\lambda T} - 1}{\lambda} = T .$$

We can use the above convergence proof for Euler's method to illustrate two important pieces of terminology which apply to all numerical methods for initial value ordinary differential equations.

(i) The *local truncation error* (l.t.e.) for a general multistep method (see §4.2)

$$\boldsymbol{y}_{n+1} = \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) \tag{4.9a}$$

is obtained by inserting the exact solution $\boldsymbol{y}(t)$ into the numerical method; thus obtaining

$$\boldsymbol{\eta}_{n+1} \equiv \boldsymbol{y}(t_{n+1}) - \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}(t_0), \boldsymbol{y}(t_1), \ldots, \boldsymbol{y}(t_n)) . \tag{4.9b}$$

This is the same as the specific notation $\boldsymbol{R}_n$ used in the convergence proof for Euler's method and denotes the inexactness of our numerical method over the single time-step $[t_n, t_{n+1}]$.

(ii) The *order* of a numerical method is the largest integer $p \geqslant 1$ such that

$$\boldsymbol{\eta}_{n+1} \equiv \boldsymbol{y}(t_{n+1}) - \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}(t_0), \boldsymbol{y}(t_1), \ldots, \boldsymbol{y}(t_n)) = \mathcal{O}\big(h^{p+1}\big) : \tag{4.10}$$

here $h > 0$ and $t_{n+1} \in [0, T]$, but $\boldsymbol{f}$ in (4.1) is assumed to be as smooth as desired. Thus we see that Euler's method has order 1 and the convergence proof shows that our uniform global error is $\|\boldsymbol{e}_n\|_\infty = \mathcal{O}(h)$. In general, the uniform global error of a numerical method differs by one power of $h$ from the local truncation error for that method. Later we shall see that, unless $p \geqslant 1$, a 'method' is an unsuitable approximation for (4.1): in particular, $p \geqslant 1$ is necessary for convergence (see Theorem 4.10).

### 4.1.2 Theta methods

These are the next simplest one-step methods beyond Euler's method.

**Definition 4.5** (*Theta methods).* For fixed $\theta \in [0,1]$, a theta method is defined by

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left\{\theta \boldsymbol{f}(t_n, \boldsymbol{y}_n) + (1-\theta)\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})\right\} \qquad n = 0, 1, \ldots . \tag{4.11}$$

For $\theta = 1$, we just recover Euler's method. The two other most important choices are $\theta = 0$ and $\theta = \frac{1}{2}$, which are known respectively as

$$\text{Backward Euler:} \qquad \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})\,, \tag{4.12a}$$

$$\text{Trapezoidal rule:} \qquad \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \tfrac{1}{2}h[\boldsymbol{f}(t_n, \boldsymbol{y}_n) + \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})]\,. \tag{4.12b}$$

For $\theta \in [0,1)$ the most important fact about (4.11) is that these methods are *implicit*: i.e. the unknown vector $\boldsymbol{y}_{n+1}$ appears on the right-hand side and so we should replace (4.5) with

$$\boldsymbol{y}_{n+1} = \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}_n, \boldsymbol{y}_{n+1})\,. \tag{4.13}$$

Thus, in contrast to Euler's method, we now need to solve a (in general, nonlinear) algebraic system of $N$ equations in order to determine $\boldsymbol{y}_{n+1}$. This requires an iterative method and we illustrate the three most common choices; in each case starting the iteration with $\boldsymbol{y}_{n+1}^{[0]} \equiv \boldsymbol{y}_n$.

$$\textit{Direct iteration:} \qquad \boldsymbol{y}_{n+1}^{[j+1]} = \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}_n, \boldsymbol{y}_{n+1}^{[j]})\,;$$

$$\textit{Newton:} \qquad \boldsymbol{y}_{n+1}^{[j+1]} = \boldsymbol{y}_{n+1}^{[j]} - \left[I - \nabla\boldsymbol{\varphi}_h(t_n, \boldsymbol{y}_n, \boldsymbol{y}_{n+1}^{[j]})\right]^{-1}\left\{\boldsymbol{y}_{n+1}^{[j]} - \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}_n, \boldsymbol{y}_{n+1}^{[j]})\right\}\,;$$

$$\textit{Modified Newton:} \qquad \boldsymbol{y}_{n+1}^{[j+1]} = \boldsymbol{y}_{n+1}^{[j]} - \left[I - \nabla\boldsymbol{\varphi}_h(t_n, \boldsymbol{y}_n, \boldsymbol{y}_{n+1}^{[0]})\right]^{-1}\left\{\boldsymbol{y}_{n+1}^{[j]} - \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}_n, \boldsymbol{y}_{n+1}^{[j]})\right\}\,.$$

(Here $\nabla\boldsymbol{\varphi}_h$ denotes the $N \times N$ Jacobian matrix of $\boldsymbol{\varphi}_h$ with respect to the argument $\boldsymbol{y}_{n+1}$.) We will return to this topic later (in §4.5.5), but emphasise here that the advantages of implicit methods (including higher order) often outweigh the extra work involved.

Finally, we show how a simple Taylor series expansion about $t = t_n$ enables us to determine the order of theta methods: i.e. (4.10) and (4.11) give

$$\begin{aligned}\boldsymbol{\eta}_{n+1} &= \boldsymbol{y}(t_{n+1}) - \boldsymbol{y}(t_n) - h\left\{\theta\boldsymbol{y}'(t_n) + [1-\theta]\boldsymbol{y}'(t_{n+1})\right\} \\ &= [\theta - \tfrac{1}{2}]h^2\boldsymbol{y}''(t_n) + [\tfrac{1}{2}\theta - \tfrac{1}{3}]h^3\boldsymbol{y}'''(t_n) + \mathcal{O}\big(h^4\big)\,.\end{aligned} \tag{4.14}$$

Thus theta methods are in general of order 1, except that the trapezoidal rule ($\theta = \frac{1}{2}$) is of order 2. In Figures 4.2 and 4.3, we see how accuracy can be improved by decreasing $h$ and/or increasing order.

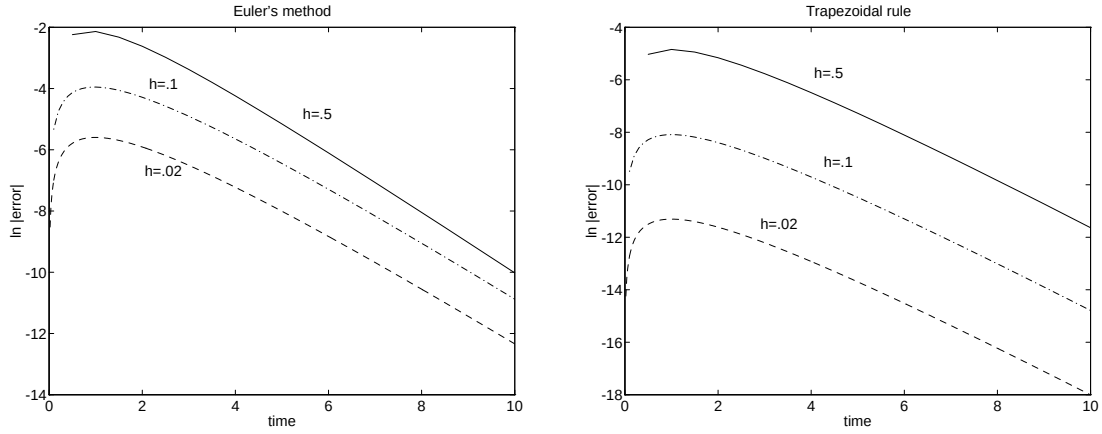



**Figure 4.2:** Error between the numerical solution and the exact solution of the equation $y' = -y$, $y(0) = 1$ for both Euler's method (first order) and the trapezoidal method (second order).

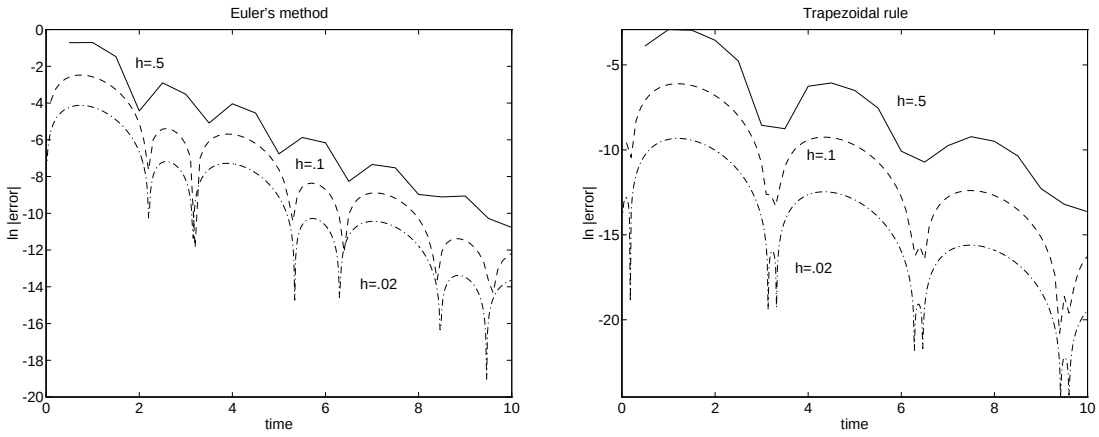



**Figure 4.3:** Error between the numerical solution and the exact solution of the equation $y' = -y+2e^{-t}\cos 2t$, $y(0) = 0$ for both Euler's method (first order) and the trapezoidal method (second order).

## 4.2 Multistep methods

One-step methods just use the previously computed approximate solution $\boldsymbol{y}_n \approx \boldsymbol{y}(t_n)$ in order to calculate the next approximate solution $\boldsymbol{y}_{n+1} \approx \boldsymbol{y}(t_{n+1})$: in contrast multistep methods aim for higher accuracy (i.e. higher order) by making use of extra past solution approximations, e.g. $\boldsymbol{y}_{n-1} \approx \boldsymbol{y}(t_{n-1})$. An example of a 2-step method (as we shall see in §4.2.3) is the *Adams–Bashforth*[4] method

$$\boldsymbol{y}_{n+2} = \boldsymbol{y}_{n+1} + \tfrac{1}{2}h\left\{3\boldsymbol{f}(t_{n+1},\boldsymbol{y}_{n+1}) - \boldsymbol{f}(t_n,\boldsymbol{y}_n)\right\}. \tag{4.15}$$

Note that it is standard notation for the next approximate solution to be $\boldsymbol{y}_{n+2} \approx \boldsymbol{y}(t_{n+2})$, while the two previously computed solution approximations are $\boldsymbol{y}_{n+1} \approx \boldsymbol{y}(t_{n+1})$ and $\boldsymbol{y}_n \approx \boldsymbol{y}(t_n)$. (We shall use (4.15) repeatedly as a simple example of a multistep method.) More generally, we can consider $s$-step methods.

**Definition 4.6** (*Multistep methods).* Assuming that $\boldsymbol{y}_n, \boldsymbol{y}_{n+1}, \ldots, \boldsymbol{y}_{n+s-1}$ are available, where $s \geqslant 1$, we say that

$$\sum_{\ell=0}^{s} \rho_\ell \boldsymbol{y}_{n+\ell} = h\sum_{\ell=0}^{s} \sigma_\ell \boldsymbol{f}(t_{n+\ell},\boldsymbol{y}_{n+\ell}), \qquad n = 0,1,\ldots, \tag{4.16}$$

where $\rho_s \equiv 1$, is an $s$-step method. If $\sigma_s = 0$, the method is *explicit,* otherwise it is *implicit.*

Since we can multiply through (4.16) by an arbitrary non-zero constant, the insistence on $\rho_s \equiv 1$ is just the traditional scaling for multistep methods. Note that our example (4.15) is an explicit method.

*Remark.* An important practical point is that, if $s \geqslant 2$, we cannot start using an $s$-step method until we obtain extra *starting values* $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{s-1}$ in some way. One possibility is to use the higher-order one-step methods in §4.3.

Finally, we emphasise that proving convergence for multistep methods (as we shall see in section §4.2.2) is more complicated than for one-step methods.

### 4.2.1 The order of a multistep method

When choosing the coefficients $\{\rho_\ell\}_{\ell=0}^{s-1}$ and $\{\sigma_\ell\}_{\ell=0}^{s}$ in the general $s$-step method (4.16), ones first thought is to make the order as high as possible. As described in (4.10), the order of a multistep method is determined by inserting the exact solution $\boldsymbol{y}(t)$ into it.

**Theorem 4.7.** *The multistep method* (4.16) *is of order* $p \geqslant 1$ *if and only if* [5]

$$\sum_{\ell=0}^{s} \rho_\ell = 0 \qquad \textit{and} \qquad \sum_{\ell=0}^{s} \rho_\ell \ell^k = k\sum_{\ell=0}^{s} \sigma_\ell \ell^{k-1} \quad \textit{for} \quad k = 1,\ldots,p. \tag{4.17}$$

[4] Adams, as in Adams Road.

[5] With the standard convention that $0^0 = 1$.

*Proof.* Substituting the exact solution and expanding in Taylor series about $t_n$, we obtain

$$\sum_{\ell=0}^{s} \rho_\ell \boldsymbol{y}(t_{n+\ell}) - h\sum_{\ell=0}^{s} \sigma_\ell \boldsymbol{y}'(t_{n+\ell}) = \sum_{\ell=0}^{s} \rho_\ell \sum_{k=0}^{\infty} \frac{(\ell h)^k}{k!} \boldsymbol{y}^{(k)}(t_n) - h\sum_{\ell=0}^{s} \sigma_\ell \sum_{k=1}^{\infty} \frac{(\ell h)^{k-1}}{(k-1)!} \boldsymbol{y}^{(k)}(t_n)$$
$$= \left(\sum_{\ell=0}^{s} \rho_\ell\right) \boldsymbol{y}(t_n) + \sum_{k=1}^{\infty} \frac{h^k}{k!} \left(\sum_{\ell=0}^{s} \rho_\ell \ell^k - k\sum_{\ell=0}^{s} \sigma_\ell \ell^{k-1}\right) \boldsymbol{y}^{(k)}(t_n)\,.$$

Thus, to obtain a local truncation error of order $\mathcal{O}\big(h^{p+1}\big)$ regardless of the choice of $\boldsymbol{y}$, it is necessary and sufficient that the coefficients of $h^k$ vanish for $k \leqslant p$, i.e. that (4.17) is satisfied. □

*Remarks.*

(i) Since the Taylor series expansion of polynomials of degree $p$ contains only terms of $\mathcal{O}\big(h^k\big)$ with $k \leqslant p$, the multistep method (4.16) is of order $p$ iff

$$\sum_{\ell=0}^{s} \rho_\ell\, Q(t_{n+\ell}) = h\sum_{\ell=0}^{s} \sigma_\ell Q'(t_{n+\ell})\,, \quad \forall\ Q \in \mathbb{P}_p[x]\,. \tag{4.18}$$

In particular taking $Q(x) = x^k$ for $k = 0, \ldots, p$, $t_{n+\ell} = \ell$ and $h = 1$, we obtain (4.17).

(ii) If the desire is to have a order $p$ method, then (4.17) might be viewed as $p+1$ equations for the $2s+1$ variables $\rho_\ell$ $(\ell = 0, \ldots, s-1)$ and $\sigma_\ell$ $(\ell = 0, \ldots, s)$. A key question is how to choose the $\rho_\ell$ and $\sigma_\ell$ (given that if $2s > p$ there is some wriggle room).

*Example: the 2-step Adams–Bashforth method.* The coefficients for (4.15) are $\rho_0 = 0$, $\rho_1 = -1$, $\rho_2 = 1$, $\sigma_0 = -\frac{1}{2}$, $\sigma_1 = \frac{3}{2}$, $\sigma_2 = 0$. Hence (4.17) gives

$$\sum_{\ell=0}^{2} \rho_\ell = 0 - 1 + 1 = 0\,,$$
$$\sum_{\ell=0}^{2} \rho_\ell \ell - \sum_{\ell=0}^{2} \sigma_\ell \ell^0 = (0 - 1 + 2) - \big(-\tfrac{1}{2} + \tfrac{3}{2} + 0\big) = 0\,,$$
$$\sum_{\ell=0}^{2} \rho_\ell \ell^2 - 2\sum_{\ell=0}^{2} \sigma_\ell \ell = \big(0 - 1 + 2^2\big) - 2\big(0 + \tfrac{3}{2} + 0\big) = 0\,,$$
$$\sum_{\ell=0}^{2} \rho_\ell \ell^3 - 3\sum_{\ell=0}^{2} \sigma_\ell \ell^2 = \big(0 - 1 + 2^3\big) - 3\big(0 + \tfrac{3}{2} + 0\big) = \tfrac{5}{2} \neq 0\,.$$

and the 2-step Adams–Bashforth method is of order 2.

The conditions (4.17) are algebraic: equivalent analytic conditions for order can be defined in terms of the two polynomials

$$\rho(w) \equiv \sum_{\ell=0}^{s} \rho_\ell w^\ell \qquad \text{and} \qquad \sigma(w) \equiv \sum_{\ell=0}^{s} \sigma_\ell w^\ell\,. \tag{4.19}$$

Note that these polynomials just depend on the coefficients of our general $s$-step method (4.16).

**Theorem 4.8.** *The multistep method* (4.16) *is of order* $p \geqslant 1$ *iff*

$$\rho(\mathrm{e}^z) - z\sigma(\mathrm{e}^z) = \mathcal{O}\big(z^{p+1}\big) \quad \textit{as} \quad z \to 0. \tag{4.20}$$

*Proof.* Expanding in Taylor series about $z = 0$ gives

$$\begin{aligned}
\rho(\mathrm{e}^z) - z\sigma(\mathrm{e}^z) &\equiv \sum_{\ell=0}^{s} \rho_\ell \mathrm{e}^{\ell z} - z \sum_{\ell=0}^{s} \sigma_\ell \mathrm{e}^{\ell z} \\
&= \sum_{\ell=0}^{s} \rho_\ell \left( \sum_{k=0}^{\infty} \frac{1}{k!} \ell^k z^k \right) - z \sum_{\ell=0}^{s} \sigma_\ell \left( \sum_{k=0}^{\infty} \frac{1}{k!} \ell^k z^k \right) \\
&= \sum_{k=0}^{\infty} \frac{1}{k!} \left( \sum_{\ell=0}^{s} \ell^k \rho_\ell \right) z^k - \sum_{k=1}^{\infty} \frac{1}{(k-1)!} \left( \sum_{\ell=0}^{s} \ell^{k-1} \sigma_\ell \right) z^k \\
&= \left( \sum_{\ell=0}^{s} \rho_\ell \right) + \sum_{k=1}^{\infty} \frac{1}{k!} \left( \sum_{\ell=0}^{s} \ell^k \rho_\ell - k \sum_{\ell=0}^{s} \ell^{k-1} \sigma_\ell \right) z^k .
\end{aligned}$$

The theorem then follows from (4.17). □

Note that our order conditions include $\rho(1) = 0$ and so the polynomial $\rho(w)$ must have a root at $w = 1$.

*Remarks*

(i) The reason that (4.20) is equivalent to (4.17) (and their proofs are almost identical) is because of the relation between Taylor series and the exponent $\mathrm{e}^{hD}$, where $D$ is the differentiation operator. Namely

$$\boldsymbol{f}(x+h) = (I + hD + \tfrac{1}{2}h^2 D^2 + \ldots)\boldsymbol{f}(x) = \mathrm{e}^{hD}\boldsymbol{f}(x)\,. \tag{4.21}$$

(ii) By using the change of variable $w = \mathrm{e}^z$, an equivalent statement of the theorem is that the multistep method (4.16) is of order $p \geqslant 1$ iff

$$\rho(w) - \log(w)\sigma(w) = \mathcal{O}\big(|w-1|^{p+1}\big)\,, \quad \text{as} \quad w \to 1. \tag{4.22}$$

*Example: the 2-step Adams–Bashforth method.* The polynomials for (4.15) are

$$\rho(w) \equiv w^2 - w \quad \text{and} \quad \sigma(w) \equiv \tfrac{3}{2}w - \tfrac{1}{2} \tag{4.23a}$$

and hence

$$\begin{aligned}
\rho(\mathrm{e}^z) - z\sigma(\mathrm{e}^z) &= \big[1 + 2z + 2z^2 + \tfrac{4}{3}z^3\big] - \big[1 + z + \tfrac{1}{2}z^2 + \tfrac{1}{6}z^3\big] - \tfrac{3}{2}z\left[1 + z + \tfrac{1}{2}z^2\right] + \tfrac{1}{2}z + \mathcal{O}\big(z^4\big) \\
&= \tfrac{5}{12}z^3 + \mathcal{O}\big(z^4\big)\,.
\end{aligned} \tag{4.23b}$$

As before, we conclude that the 2-step Adams–Bashforth method is of order 2.

### 4.2.2 The convergence of multistep methods

Proving convergence for multistep methods is more complicated than for one-step methods and we shall only state the required conditions in Theorem 4.10 below. It is however possible to illustrate the problem with a very simple example.

*Example: absence of convergence.* The explicit 2-step method

$$\boldsymbol{y}_{n+2} + 4\boldsymbol{y}_{n+1} - 5\boldsymbol{y}_n = h\left\{4\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}) + 2\boldsymbol{f}(t_n, \boldsymbol{y}_n)\right\} \tag{4.24}$$

has order 3; as may be verified by using the two polynomials

$$\rho(w) \equiv w^2 + 4w - 5 \quad \text{and} \quad \sigma(w) \equiv 4w + 2\,.$$

If we apply this method to the trivial scalar initial value ODE

$$y'(t) = 0 \quad y(0) = 1\,, \tag{4.25}$$

which of course has exact solution $y(t) = 1$, we obtain the second-order difference equation

$$y_{n+2} + 4y_{n+1} - 5y_n = 0 \qquad n \geqslant 0\,.$$

Since the roots of the polynomial $\rho$ are 1 and $-5$, the general solution of this difference equation is

$$y_n = c_1(1)^n + c_2(-5)^n \quad \text{for} \quad n = 0, 1, \ldots\,,$$

where $c_1, c_2$ are arbitrary constants. Our initial condition $y_0 = 1$ will force these constants to satisfy $c_1 + c_2 = 1$, but the second equation for $c_1$ & $c_2$ (which is $y_1 = c_1 - 5c_2$) depends on the value of $y_1$ used to start (4.24). (See the remark after Definition 4.6.) In general we cannot expect $y_1$ to be exact (i.e. $y_1 = 1$), but merely satisfy $y_1 = 1 + \mathcal{O}(h^p)$ for some $p \geqslant 1$: hence $c_2$ will not be zero, but will depend on $h$ and merely satisfy $\lim_{h\to 0} c_2(h) = 0$. Because our second root of $\rho$ is greater than 1 in modulus, this is insufficient for our numerical method to converge: e.g. we do not satisfy

$$\lim_{\substack{h\to 0\\ nh\to t}} y_n = 1\,.$$

*Exercise (with a large amount of algebra).* Consider the ODE $y'(t) = -y(t)$, $y(0) = 1$, which has the exact solution $y(t) = \mathrm{e}^{-t}$. Show that if $y_1 = \mathrm{e}^{-h}$, the sequence $\{y_n\}_{n=0}^{\infty}$ grows like $h^4(-5)^n$.

The extra condition that a multistep method must satisfy in order to be convergent is contained in the following definition. If a multistep method does not satisfy this condition then, no matter how high its order, it is not usable.

**Definition 4.9** (*root condition*)**.** A multistep method satisfies the *root condition* if all the zeros of its polynomial $\rho(w)$ lie in the unit disc $|w| \leqslant 1$ and any zeros of unit modulus are simple. (Sometimes one just says that $\rho$ satisfies the root condition.)

In section 4.2.1 we saw that a multistep method of order $p \geqslant 1$ must have its polynomial satisfy $\rho(1) = 0$. The other roots of $\rho$ have nothing to do with the solution of our initial value ODE and Definition 4.9 ensures that they do not "swamp" it.

Now we can state the most famous theorem on the convergence of multistep methods.

**Theorem 4.10** (The Dahlquist equivalence theorem)**.** *The multistep method* (4.16) *is convergent iff it is of order $p \geqslant 1$ and the polynomial $\rho$ obeys the root condition.*

*Proof.* See Part III. □

*Definition.* If $\rho$ obeys the root condition, the multistep method (4.16) is sometimes said to be *zero-stable:* we will not use this terminology.

*Examples.* For the Adams–Bashforth method (4.15) we have $\rho(w) \equiv (w-1)w$ and the root condition is obeyed. However, for (4.24) we obtain $\rho(w) \equiv (w-1)(w+5)$, the root condition fails and it follows that there is no convergence.

### 4.2.3 Adams and BDF methods

A sensible strategy for constructing convergent multistep methods is to first satisfy the root condition in Definition 4.9 and secondly to maximise the order. For example, if we aim to construct an $s$-step implicit method then:-

- choose the polynomial $\rho$ so that $\rho(1) = 0$ and the root condition holds;
- choose the polynomial $\sigma$ so that

$$\sigma(w) = \frac{\rho(w)}{\log w} + \mathcal{O}\big(|w-1|^{s+1}\big)\,. \tag{4.26a}$$

Since an implicit method gives us $s+1$ free coefficients in $\sigma$, (4.26a) can be satisfied by expanding in a Taylor series about $w=1$. (Note that the simple root of $\log w$ at $w=1$ is no problem, since $\rho(1)=0$.) If we change variable with $w=\mathrm{e}^z$ then (4.26a) becomes

$$\rho(\mathrm{e}^z) - z\sigma(\mathrm{e}^z) = \mathcal{O}\big(z^{s+2}\big) \tag{4.26b}$$

and we see from (4.20) that we have constructed an $s$-step method of order at least $s+1$.

If our aim is to construct an $s$-step explicit method then (because $\sigma_s = 0$) we only have $s$ free coefficients in $\sigma$. Hence we can only expect to construct an $s$-step explicit method of order $s$.

The most popular class of multistep methods that utilise the above strategy are *Adams methods*, which choose $\rho(w) \equiv w^{s-1}(w-1)$. Thus $\rho$ has a simple root at 1 (as required) but all its other roots are zero. (One could therefore argue that Adams methods "best" satisfy the root condition!) Thus Adams methods have the form

$$\boldsymbol{y}_{n+s} - \boldsymbol{y}_{n+s-1} = h\sum_{\ell=0}^{s} \sigma_\ell \boldsymbol{f}(t_{n+\ell}, \boldsymbol{y}_{n+\ell}), \qquad n = 0, 1, \ldots : \tag{4.27}$$

they are called Adams–Bashforth if explicit and Adams–Moulton if implicit.

*Example.* The 2-step, 3rd-order Adams–Moulton method

$$\boldsymbol{y}_{n+2} - \boldsymbol{y}_{n+1} = h\left\{-\tfrac{1}{12}\boldsymbol{f}(t_n, \boldsymbol{y}_n) + \tfrac{2}{3}\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}) + \tfrac{5}{12}\boldsymbol{f}(t_{n+2}, \boldsymbol{y}_{n+2})\right\}$$

is derived from

$$\begin{aligned}
\frac{w(w-1)}{\log w} &= \frac{\xi+\xi^2}{\log(1+\xi)} = \frac{\xi+\xi^2}{\xi - \frac{1}{2}\xi^2 + \frac{1}{3}\xi^3 - \cdots} = \frac{1+\xi}{1 - \frac{1}{2}\xi + \frac{1}{3}\xi^2 - \cdots} \\
&= (1+\xi)[1 + (\tfrac{1}{2}\xi - \tfrac{1}{3}\xi^2) + (\tfrac{1}{2}\xi - \tfrac{1}{3}\xi^2)^2 + \mathcal{O}\big(\xi^3\big)] = 1 + \tfrac{3}{2}\xi + \tfrac{5}{12}\xi^2 + \mathcal{O}\big(\xi^3\big) \\
&= 1 + \tfrac{3}{2}(w-1) + \tfrac{5}{12}(w-1)^2 + \mathcal{O}\big(|w-1|^3\big) \\
&= -\tfrac{1}{12} + \tfrac{2}{3}w + \tfrac{5}{12}w^2 + \mathcal{O}\big(|w-1|^3\big)\ ,
\end{aligned}$$

where $\xi \equiv w-1$. Thus the coefficients of $\sigma(w)$ are calculated from the Taylor series expansion of $w(w-1)/log(w)$ about $w=1$.

Note that we have already shown that (4.15) is a 2-step Adams–Bashforth method of order 2.

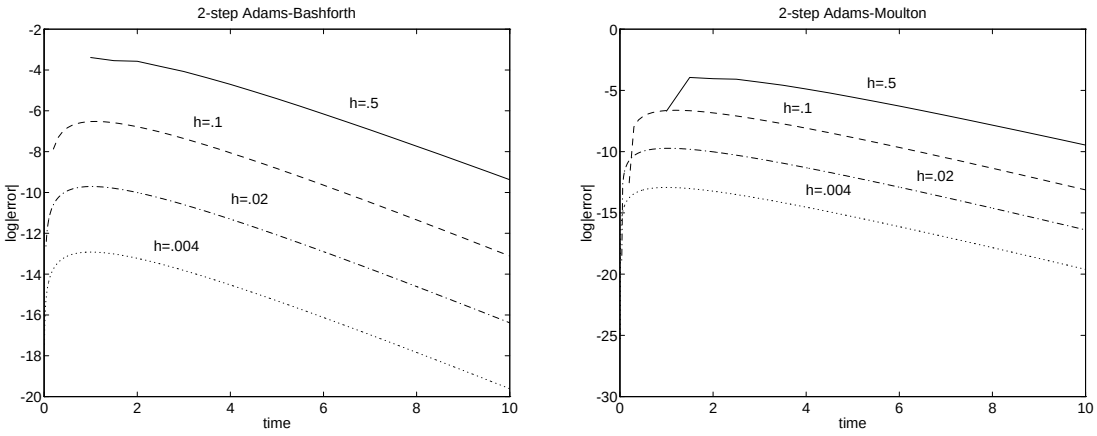



**Figure 4.4:** Error between the numerical solution and the exact solution of the equation $y' = -y$, $y(0) = 1$ for both the 2-step Adams–Bashforth method (second order) and the 2-step Adams–Moulton method (third order).

Another popular class of implicit multistep methods is based on a different philosophy to Adams methods. Instead of insisting that the polynomial $\rho$ has a simple form, they choose an $s$-step method with $\sigma(w) \equiv$

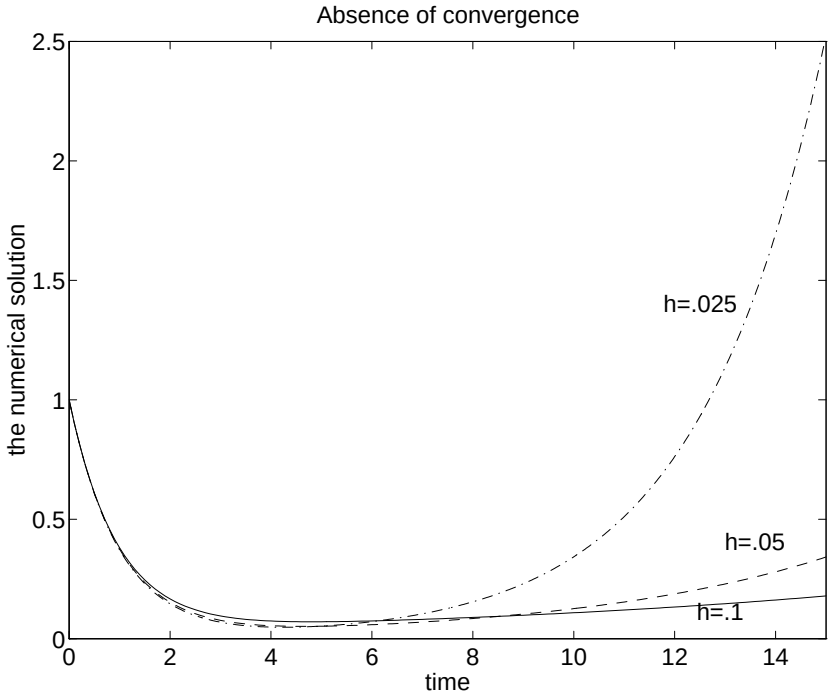



**Figure 4.5:** The numerical solution to the equation is $y' = -y$, $y(0) = 1$ using the 2-step method with $\rho(w) = w^2 - 2.01w + 1.01$, $\sigma(w) = 0.995w - 1.005$. Thus, $\rho$ has a zero at 1.01, the method is not convergent, and the smaller $h$ the worse the solution.

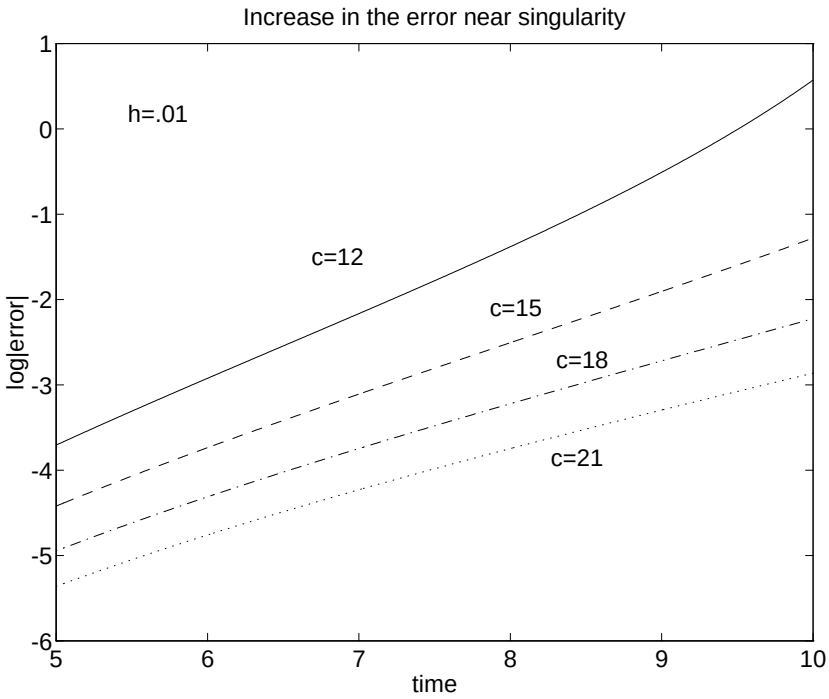



**Figure 4.6:** Plot showing that accuracy may deteriorate near a singularity. Plotted is the error to the solution of $y' = 2y/t + (y/t)^2$, $y(1) = 1/(c-1)$ using the 2-step Adams–Bashforth method, for various values of $c$. The exact solution is $y(t) = t^2/(c-t)$, with singularity at $c$. Accuracy worsens the nearer we are to the singularity.

$\sigma_s w^s$ for some $\sigma_s \neq 0$ (see section §4.4.2 for the justification). These are called *backward differentiation* (BDF) methods and take the form

$$\sum_{\ell=0}^{s} \rho_\ell \boldsymbol{y}_{n+\ell} = h\sigma_s \boldsymbol{f}(t_{n+s}, \boldsymbol{y}_{n+s}), \qquad n = 0, 1, \ldots . \tag{4.28}$$

The coefficients of $\rho$ are now chosen to maximise the order and the very simplest example of a BDF method occurs when $s = 1$: i.e. we then obtain the one-step backward Euler method of Definition 4.11.

**Lemma 4.11.** *An s-step BDF method of order s is obtained by choosing*

$$\rho(w) \equiv \sigma_s \sum_{\ell=1}^{s} \frac{1}{\ell} w^{s-\ell}(w-1)^\ell, \qquad \text{with} \quad \sigma_s = \left( \sum_{\ell=1}^{s} \frac{1}{\ell} \right)^{-1} . \tag{4.29}$$

*This value of $\sigma_s$ being necessary to force the standard scaling $\rho_s \equiv 1$ of Definition 4.6.*

*Proof.* Using (4.22), we need to construct $\rho$ so that

$$\rho(w) = \sigma_s w^s \log w + \mathcal{O}\big(|w-1|^{s+1}\big) . \tag{4.30}$$

This easy if we write

$$\log w = -\log\left(\frac{1}{w}\right) = -\log\left(1 - \frac{w-1}{w}\right) = \sum_{\ell=1}^{\infty} \frac{1}{\ell} \left(\frac{w-1}{w}\right)^\ell ,$$

because then we see that choosing $\rho$ as in (4.29) means that (4.30) is satisfied. □

*Examples*

(i) Let $s = 2$. Then substitution in (4.29) yields $\sigma_2 = \frac{2}{3}$, and some straightforward algebra results in $\rho(w) = w^2 - \frac{4}{3}w + \frac{1}{3} = (w-1)(w-\frac{1}{3})$. Hence $\rho$ satisfies the root condition, and the 2-step BDF is

$$\boldsymbol{y}_{n+2} - \tfrac{4}{3}\boldsymbol{y}_{n+1} + \tfrac{1}{3}\boldsymbol{y}_n = \tfrac{2}{3}h\boldsymbol{f}(t_{n+2}, \boldsymbol{y}_{n+2}) . \tag{4.31a}$$

(ii) Similarly for $s = 3$ we find that $\rho$ satisfies the root condition, and that the 3-step BDF is

$$\boldsymbol{y}_{n+3} - \tfrac{18}{11}\boldsymbol{y}_{n+2} + \tfrac{9}{11}\boldsymbol{y}_{n+1} - \tfrac{2}{11}\boldsymbol{y}_n = \tfrac{6}{11}h\boldsymbol{f}(t_{n+3}, \boldsymbol{y}_{n+3}) . \tag{4.31b}$$

*Convergence of BDF methods.* There is no guarantee that BDF methods satisfy the root condition and hence no guarantee that BDF methods converge. For $s \leqslant 6$, it can be verified that the root condition actually does hold. BDF methods with larger $s$ should not be used!

## 4.3 Runge–Kutta methods

These are sophisticated one-step methods: capable of much higher accuracy than the one-step methods of §4.1, but also much more complicated to derive and work with. In the pre-computer and early computer days (1960's), these methods were almost completely ignored; because multistep methods were much easier to use in practice. Nowerdays they are very popular and the optimal order implicit Runge–Kutta methods have very strong theoretical properties.

### 4.3.1 Quadrature formulae

As motivation for the structure of Runge–Kutta methods, we recall the quadrature formulae of §3.2; i.e.

$$\int_0^1 f(t)\mathrm{d}t \approx h \sum_{\ell=1}^{\nu} b_\ell f(c_\ell) , \tag{4.32}$$

where the nodes $\{c_\ell\}_{\ell=1}^{\nu}$ and the weights $\{b_\ell\}_{\ell=1}^{\nu}$ are chosen so that (4.32) is exact for polynomials of a certain degree. In particular, we restate the following two results from §3.2.

- For any set of distinct nodes, we can choose weights so that (4.32) is exact for $f \in \mathbb{P}_{\nu-1}[x]$.
- If the nodes are chosen to be the zeros of the Legendre polynomial $P_\nu$ (transformed from $[-1,1]$ to $[0,1]$ by the linear mapping (1.22)), then we can choose the weights so that (4.32) is exact for $f \in \mathbb{P}_{2\nu-1}[x]$.

These results can be immediately applied to the "trivial" initial value ODE

$$\boldsymbol{y}'(t) = \boldsymbol{f}(t)\,, \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0 \tag{4.33a}$$

(i.e. $\boldsymbol{f}$ has no dependence on $\boldsymbol{y}$) because integrating (4.33a) over a single time step gives

$$\boldsymbol{y}(t_{n+1}) = \boldsymbol{y}(t_n) + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(t)\mathrm{d}t\,. \tag{4.33b}$$

Hence we can use the quadrature formula (4.32) (transformed from $[0,1]$ to $[t_n, t_{n+1}]$) to generate the one-step method

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{\ell=1}^{\nu} b_\ell \boldsymbol{f}(t_n + c_\ell h) \qquad n = 0, 1, \ldots\,. \tag{4.33c}$$

This would be an example of a $\nu$-stage Runge–Kutta method. Note that, in order for (4.32) to be exact when $f$ is a constant, we must have $\sum_{\ell=1}^{\nu} b_\ell = 1$.

This is all well and good, but how do we extend the idea from the special case (4.33a) to the general ODE (4.1)? Integrating (4.1) over a single time step gives

$$\boldsymbol{y}(t_{n+1}) = \boldsymbol{y}(t_n) + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(t, \boldsymbol{y}(t))\mathrm{d}t \tag{4.34a}$$

and, after applying the quadrature formula (4.32), we arrive at the "one-step method"

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{\ell=1}^{\nu} b_\ell \boldsymbol{f}(t_n + c_\ell h, \boldsymbol{y}(t_n + c_\ell h))\,. \tag{4.34b}$$

But we certainly don't know the exact values $\{\boldsymbol{y}(t_n + c_\ell h)\}_{\ell=1}^{\nu}$, so in some way we have to approximate these values and/or $\{\boldsymbol{f}(t_n + c_\ell h, \boldsymbol{y}(t_n + c_\ell h)\}_{\ell=1}^{\nu}$. In the next section, we explain the Runge–Kutta philosophy behind making these approximations.

### 4.3.2 General Runge–Kutta methods

$\nu$-stage Runge–Kutta methods for (4.1) over $[t_n, t_{n+1}]$, i.e. for approximating (4.34b), have the general form

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{\ell=1}^{\nu} b_\ell \boldsymbol{k}_\ell\,, \tag{4.35a}$$

where

$$\boldsymbol{k}_\ell = \boldsymbol{f}\left(t_n + c_\ell h, \boldsymbol{y}_n + h\sum_{j=1}^{\nu} a_{\ell,j}\boldsymbol{k}_j\right) \qquad \ell = 1, 2, \ldots, \nu\,. \tag{4.35b}$$

To explain what is going on here, we list the following points.

a) Different Runge–Kutta methods are obtained by different choices for the real parameters $\{b_\ell\}_{\ell=1}^{\nu}$, $\{c_\ell\}_{\ell=1}^{\nu}$ and $\{a_{\ell,j}\}_{\ell,j=1}^{\nu}$. Thus we need the extra parameters $\{a_{\ell,j}\}_{\ell,j=1}^{\nu}$ in order to approximate the general ODE (4.1) rather than just the special case (4.33a).

b) $\{\boldsymbol{k}_\ell\}_{\ell=1}^{\nu} \subset \mathbb{R}^N$ have to be computed from (4.35b). We can think of $\{\boldsymbol{k}_\ell\}_{\ell=1}^{\nu}$ approximating $\{\boldsymbol{f}(t_n + c_\ell h, \boldsymbol{y}(t_n + c_\ell h)\}_{\ell=1}^{\nu}$ in (4.35a), as well as the approximations

$$\boldsymbol{y}_n + h\sum_{j=1}^{\nu} a_{\ell,j}\boldsymbol{k}_j \approx \boldsymbol{y}(t_n + c_\ell h) \qquad \ell = 1, \ldots, \nu$$

in (4.35b).

c) In general we are describing implicit Runge–Kutta methods, because (4.35b) are a set of nonlinear algebraic equations that need to be solved for $\{\boldsymbol{k}_\ell\}_{\ell=1}^{\nu}$ simultaneously. If, however, we restrict our choice of parameters so that

$$1 \leqslant \ell \leqslant j \leqslant \nu \quad \Rightarrow \quad a_{\ell,j} = 0\,, \tag{4.36}$$

then these Runge–Kutta methods are explicit because (4.35b) just gives formulae for $\{\boldsymbol{k}_\ell\}_{\ell=1}^{\nu}$ one-by-one. (No solution of algebraic equations is necessary!)

d) The accuracy of any Runge–Kutta method depends on its order and this is determined as in (4.10): i.e. we insert the exact solution $\boldsymbol{y}(t)$ of our initial value ODE into (4.35a) and (4.35b). Thus our local truncation error is

$$\boldsymbol{\eta}_{n+1} \equiv \boldsymbol{y}(t_{n+1}) - \boldsymbol{y}(t_n) - h\sum_{\ell=1}^{\nu} b_\ell \boldsymbol{k}_\ell\,, \tag{4.37a}$$

where

$$\boldsymbol{k}_\ell = \boldsymbol{f}\left(t_n + c_\ell h, \boldsymbol{y}(t_n) + h\sum_{j=1}^{\nu} a_{\ell,j}\boldsymbol{k}_j\right) \qquad \ell = 1, 2, \ldots, \nu\,, \tag{4.37b}$$

and if $\boldsymbol{\eta}_{n+1} = \mathcal{O}\big(h^{p+1}\big)$ then the order is $p$. Because the parameters for Runge–Kutta methods (unlike the parameters for multistep methods) appear inside $\boldsymbol{f}$, the Taylor series expansions required to determine order rapidly become unmanageable. (Nowerdays symbolic algebra packages do this job!)

The two simplest order conditions on parameter choice are

$$\sum_{\ell=1}^{\nu} b_\ell = 1 \qquad \text{and} \qquad c_\ell = \sum_{j=1}^{\nu} a_{\ell,j} \quad \ell = 1, \ldots, \nu\,.$$

The optimal choice of parameters can achieve order $2\nu$ and we show how this can be done in §4.3.4: of course this is an implicit method and can be regarded as the ODE equivalent of Gauss quadrature in §3.2.1.

e) Because Runge–Kutta methods are one-step methods, the only criterion for convergence is order $p \geqslant 1$. There is no extra restriction, like the root condition for multistep methods.

The standard notation used to describe Runge–Kutta methods is the *Butcher table*:-

$$\begin{array}{c|c} \boldsymbol{c} & \mathsf{A} \\ \hline & \boldsymbol{b}^{\mathrm{T}} \end{array} \quad = \quad \begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,\nu} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,\nu} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_\nu & a_{\nu,1} & a_{\nu,2} & \cdots & a_{\nu,\nu} \\ \hline & b_1 & b_2 & \cdots & b_\nu \end{array}\,, \tag{4.38}$$

where $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^\nu$ and $\mathsf{A} \in \mathbb{R}^{\nu\times\nu}$. Note that an *explicit* RK method corresponds to the case when $a_{\ell,j} = 0$ for $\ell \leqslant j$, i.e. the matrix $\mathsf{A}$ is strictly lower triangular.

### 4.3.3 Examples of Runge–Kutta methods

1) We first show how the parameters in a 2-stage explicit RK method can be chosen to maximise order. Thus we have three free parameters and our method is

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\,\{b_1\boldsymbol{k}_1 + b_2\boldsymbol{k}_2\}\,, \tag{4.39a}$$

where

$$\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}(t_n, \boldsymbol{y}_n) && \text{(4.39b)} \\ \boldsymbol{k}_2 &= \boldsymbol{f}(t_n + c_2 h, \boldsymbol{y}_n + c_2 h \boldsymbol{k}_1)\,. && \text{(4.39c)} \end{aligned}$$

(Note that we have already introduced the order condition $a_{21} = c_2$.) Inserting the exact solution $\boldsymbol{y}(t_n)$ into (4.39b) and (4.39c) gives

$$\boldsymbol{k}_1 = \boldsymbol{y}'(t_n)$$

and

$$\begin{aligned} \boldsymbol{k}_2 &= \boldsymbol{f}(t_n + c_2 h, \boldsymbol{y}(t_n) + c_2 h \boldsymbol{y}'(t_n)) \\ &= \boldsymbol{f}(t_n, \boldsymbol{y}(t_n)) + c_2 h \left\{ \frac{\partial \boldsymbol{f}}{\partial t}(t_n, \boldsymbol{y}(t_n)) + \boldsymbol{\nabla} \boldsymbol{f}(t_n, \boldsymbol{y}(t_n)) \boldsymbol{y}'(t_n) \right\} + \mathcal{O}\big(h^2\big) \\ &= \boldsymbol{y}'(t_n) + c_2 h \boldsymbol{y}''(t_n) + \mathcal{O}\big(h^2\big)\,. \end{aligned}$$

Hence inserting these results back into (4.39a) shows that our local truncation error is

$$\begin{aligned} \boldsymbol{\eta}_{n+1} &\equiv \boldsymbol{y}(t_{n+1}) - \boldsymbol{y}(t_n) - h\left\{b_1 \boldsymbol{k}_1 + b_2 \boldsymbol{k}_2)\right\} \\ &= [1 - b_1 - b_2] h \boldsymbol{y}'(t_n) + \left[\tfrac{1}{2} - b_2 c_2\right] h^2 \boldsymbol{y}''(t_n) + \mathcal{O}\big(h^3\big)\,. \end{aligned} \qquad \text{(4.40)}$$

Thus we have a 1-parameter family of order 2 methods under the conditions

$$b_1 + b_2 = 1 \quad \text{and} \quad b_2 c_2 = \frac{1}{2}\,.$$

(That no explicit 2-stage RK method can be of third order or greater can be demonstrated by applying it to the scalar ODE $y' = \lambda y$.) A popular choice is $b_1 = 0$, $b_2 = 1$ and $c_2 = \frac{1}{2}$ (e.g. see Figure 4.7).

2) The most famous explicit RK method is the 4-stage order 4 method

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \tfrac{h}{6}\left\{\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right\}\,,$$

where

$$\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}(t_n, \boldsymbol{y}_n) \\ \boldsymbol{k}_2 &= \boldsymbol{f}(t_n + \tfrac{1}{2}h, \boldsymbol{y}_n + \tfrac{1}{2}h\boldsymbol{k}_1) \\ \boldsymbol{k}_3 &= \boldsymbol{f}(t_n + \tfrac{1}{2}h, \boldsymbol{y}_n + \tfrac{1}{2}h\boldsymbol{k}_2) \\ \boldsymbol{k}_4 &= \boldsymbol{f}(t_n + h, \boldsymbol{y}_n + h\boldsymbol{k}_3)\,. \end{aligned}$$

Its Butcher table is

| | | | | |
|---|---|---|---|---|
| 0 | | | | |
| $\frac{1}{2}$ | $\frac{1}{2}$ | | | |
| $\frac{1}{2}$ | 0 | $\frac{1}{2}$ | | |
| 1 | 0 | 0 | 1 | |
| | $\frac{1}{6}$ | $\frac{1}{3}$ | $\frac{1}{3}$ | $\frac{1}{6}$ |

.

3) Consider the 2-stage method implicit method

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \tfrac{1}{4}h\left\{\boldsymbol{k}_1 + 3\boldsymbol{k}_2\right\}\,, \qquad \text{(4.41a)}$$

where

$$\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}\left(t_n, \boldsymbol{y}_n + \tfrac{1}{4}h[\boldsymbol{k}_1 - \boldsymbol{k}_2]\right) && \text{(4.41b)} \\ \boldsymbol{k}_2 &= \boldsymbol{f}\left(t_n + \tfrac{2}{3}h, \boldsymbol{y}_n + \tfrac{1}{12}h[3\boldsymbol{k}_1 + 5\boldsymbol{k}_2]\right)\,. && \text{(4.41c)} \end{aligned}$$

In order to analyse the order of this method, we restrict our attention to scalar, *autonomous* equations of the form $y' = f(y)$ (although this procedure might lead to loss of generality for methods of order

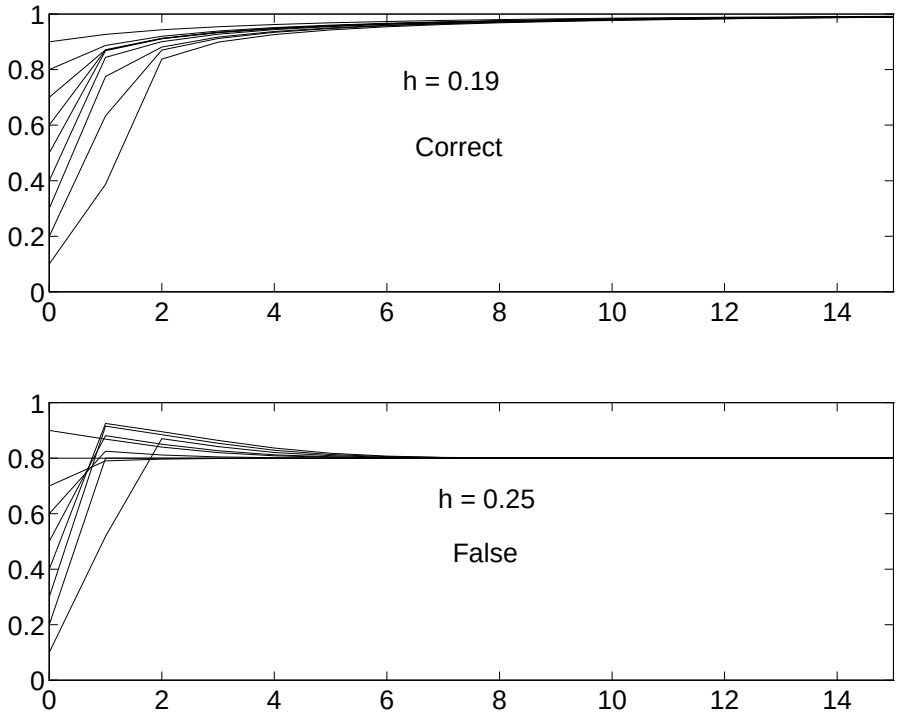



**Figure 4.7:** Solutions to the equation $y' = \alpha y(1-y)$ using the 2nd order RK method

$$k_1 = f(y_n)\,, \quad k_2 = f\left(y_n + \tfrac{1}{2}hk_1\right)\,, \quad y_{n+1} = y_n + hk_2 \quad \text{(i.e. } b_1 = 0,\, b_2 = 1 \text{ and } c_2 = \tfrac{1}{2}\text{)}.$$

For every $\alpha > 0$ and every $y_0 > 0$ it is straightforward to verify that $\lim_{t\to\infty} y(t) = 1$. However, an unwise choice of $h$ in the RK method can produce trajectories that tend to wrong limits. Here $\alpha = 10$ and it can be seen that quite small 'excess' in the value of $h$ can be disastrous: $h = 0.19$ is good, whilst $h = 0.25$ is bad. An important aspect of this example is that, although the second graph displays a wrong solution trajectory, it looks 'right' – there are no apparent instabilities, no chaotic components, no oscillation on a grid scale, no tell-tale signs that something has gone astray. Thus – and this is lost on many users of numerical methods – it is not enough to use your eyes. Use your brain as well!

greater than or equal to 5). For brevity, we use the convention that all functions are evaluated at $y = y(t_n)$, e.g. $f_y = \frac{\mathrm{d}f}{\mathrm{d}y}(y(t_n))$. Thus,

$$\begin{aligned} k_1 &= f + \tfrac{1}{4}h(k_1 - k_2)f_y + \tfrac{1}{32}h^2(k_1 - k_2)^2 f_{yy} + \mathcal{O}\big(h^3\big)\,, \\ k_2 &= f + \tfrac{1}{12}h(3k_1 + 5k_2)f_y + \tfrac{1}{288}h^2(3k_1 + 5k_2)^2 f_{yy} + \mathcal{O}\big(h^3\big)\,. \end{aligned}$$

Hence $k_1, k_2 = f + \mathcal{O}(h)$, and so substitution in the above equations yields

$$k_1 = f + \mathcal{O}\big(h^2\big) \quad \text{and} \quad k_2 = f + \tfrac{2}{3}hff_y + \mathcal{O}\big(h^2\big)\,.$$

Substituting again, we obtain

$$\begin{aligned} k_1 &= f - \tfrac{1}{6}h^2 f f_y^2 + \mathcal{O}\big(h^3\big)\,, \\ k_2 &= f + \tfrac{2}{3}hff_y + h^2\left(\tfrac{5}{18}ff_y^2 + \tfrac{2}{9}f^2 f_{yy}\right) + \mathcal{O}\big(h^3\big) \end{aligned}$$

and hence

$$y_{n+1} = y + hf + \tfrac{1}{2}h^2 f f_y + \tfrac{1}{6}h^3(f f_y^2 + f^2 f_{yy}) + \mathcal{O}\big(h^4\big)\,.$$

But $y' = f$, and hence

$$y'' = f f_y \quad \text{and} \quad y''' = f f_y^2 + f^2 f_{yy}\,.$$

We deduce from Taylor's theorem that the method is at least of order 3.

*Remark.* It is possible to verify that it is not of order 4, for example applying it to the equation $y' = \lambda y$.

*Remark.* A better way of deriving the order of Runge–Kutta methods is based on graph-theoretic approaches.

4) The optimal 2-stage order 4 implicit RK method has Butcher table

$$\begin{array}{c|cc} \frac{1}{6}\left(3-\sqrt{3}\right) & \frac{1}{4} & \frac{1}{12}\left(3-2\sqrt{3}\right) \\ \frac{1}{6}\left(3+\sqrt{3}\right) & \frac{1}{12}\left(3+2\sqrt{3}\right) & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{4.42}$$

In the next section, we shall see how these parameters can be obtained.

### 4.3.4 Implicit Runge–Kutta methods (*Unlectured*)

In this section we explain (without proof) how the parameters in implicit $\nu$-stage RK methods can be chosen to achieve high order. It is only necessary to consider the scalar ODE

$$y'(t) = f(t, y(t)) \tag{4.43}$$

over the single time step $[t_n, t_{n+1}]$.

For any choice of distinct points $\{c_i\}_{i=1}^{\nu} \subset [0,1]$, the *collocation* method for approximately solving (4.43) is to construct $p \in \mathbb{P}_\nu[x]$ satisfying

$$p(t_n) = y_n \qquad \text{and} \qquad p'(t_n + c_i h) = f(t_n + c_i h, p(t_n + c_i h)) \quad i = 1, \ldots, \nu\,. \tag{4.44}$$

(Thus we can think of collocation as "interpolating" our ODE!) If $h \equiv t_{n+1} - t_n$ is sufficiently small, relative to the Lipschitz constant for $f$, it can be shown that this set of $\nu+1$ equations in $\nu+1$ unknowns has a unique solution. The approximation to $y(t_{n+1})$ is then provided by $y_{n+1} \equiv p(t_{n+1})$.

In fact it is easy to verify that collocation methods are equivalent to implicit RK methods.

**Lemma 4.12.** *Let $\{\ell_i\}_{i=1}^{\nu} \subset \mathbb{P}_{\nu-1}[x]$ be the Lagrange cardinal polynomials with respect to $\{c_i\}_{i=1}^{\nu}$. Then the above collocation method is identical to the $\nu$-stage implicit RK method with parameters*

$$b_i \equiv \int_0^1 \ell_i(\tau)\,\mathrm{d}\tau \quad i = 1, \ldots, \nu \qquad \textit{and} \qquad a_{ij} \equiv \int_0^{c_i} \ell_j(\tau)\,\mathrm{d}\tau \quad i, j = 1, \ldots, \nu\,. \tag{4.45}$$

*Proof.* If $p$ satisfies (4.44), we show that (4.35a) and (4.35b) are satisfied by

$$k_i \equiv p'(t_n + c_i h) \quad i = 1, \ldots, \nu\,.$$

Firstly, using the change of variable $t = t_n + \tau h$, (4.35a) is obtained from

$$\begin{aligned} y_{n+1} - y_n &= \int_{t_n}^{t_{n+1}} p'(t)\,\mathrm{d}t \\ &= h \int_0^1 p'(t_n + \tau h)\,\mathrm{d}\tau \\ &= h \int_0^1 \sum_{i=1}^{\nu} p'(t_n + c_i h)\ell_i(\tau)\,\mathrm{d}\tau \\ &= h \sum_{i=1}^{\nu} b_i k_i\,. \end{aligned}$$

Secondly, for (4.35b) it is sufficient to show that

$$p(t_n + c_i h) = y_n + h \sum_{j=1}^{\nu} a_{ij} k_j \qquad i, j = 1, \ldots, \nu\,.$$

This follows from

$$\begin{aligned} p(t_n + c_i h) &= p(t_n) + \int_{t_n}^{t_n + c_i h} p'(t)\,\mathrm{d}t \\ &= y_n + h \int_0^{c_i} p'(t_n + \tau h)\,\mathrm{d}\tau \\ &= y_n + h \int_0^{c_i} \sum_{j=1}^{\nu} p'(t_n + c_j h)\ell_j(\tau)\,\mathrm{d}\tau \\ &= y_n + h \sum_{j=1}^{\nu} a_{ij} k_j \,. \end{aligned}$$

□

Now we can state our required result.

**Theorem 4.13** (No proof)**.** *Let $\{c_i\}_{i=1}^{\nu}$ be chosen so that the nodal polynomial $\omega(\tau) \equiv \prod_{i=1}^{\nu}(\tau - c_i)$ is orthogonal on the interval $[0,1]$ to all $q \in \mathbb{P}_s[x]$, where we must have $s \leqslant \nu - 1$. Then the $\nu$-stage implicit RK method with parameters* (4.45) *is of order $\nu + s + 1$.*

The highest order of a $\nu$-stage implicit RK method is $2\nu$, and corresponds to collocation at zeros of the transformed Legendre polynomial $P_\nu$ (Gauss–Legendre RK method).

## 4.4 Stiff equations

When approximating an initial value problem for the ODE (4.1), one is not only interested in the accuracy of our numerical approximation. Since the 1960's, another important consideration has been: does the approximation preserve any important "structure" of (4.1)? Of course this question is too vague as it stands and we need to state precisely what kind of structure we are interested in. This section on stiff equations is interested is concerned with the preservation of one particular structural property.

### 4.4.1 Stiffness: the problem

Suppose that the exact solutions of (4.1), independent of the initial value $\boldsymbol{y}(0) = \boldsymbol{y}_0$, exist for $[0,T]$ $\forall T > 0$ and satisfy

$$\lim_{t\to\infty} \boldsymbol{y}(t) = \boldsymbol{0} \;:$$

will the approximate solutions for our numerical method satisfy the analogous property

$$\lim_{n\to\infty} \boldsymbol{y}_n = \boldsymbol{0}$$

without imposing severe retrictions on the size of $h$? Of course one will guess immediately that the answer depends on our choice of numerical method.

To illustrate what can happen, we look at two simple numerical methods applied to two linear scalar constant-coefficient examples.

a) Consider the initial value ODE

$$y'(t) = -0.1y(t) \qquad y(0) = 1\,, \tag{4.46a}$$

whose exact solution is

$$y(t) \equiv \mathrm{e}^{-0.1t} \quad \forall t \geqslant 0\,.$$

If one applies Euler's method (4.6) to (4.46a), then our numerical approximation is $y_n = (1 - 0.1h)^n$ $n \geqslant 0$. Thus the property $\lim_{n\to\infty} y_n = 0$ is preserved, provided the weak restriction $0 < h < 20$ holds. If one applies the backward Euler method (4.12a) to (4.46a), then our numerical approximation is $y_n = (1 + 0.1h)^{-n}$ $n \geqslant 0$. Thus the property $\lim_{n\to\infty} y_n = 0$ is preserved $\forall h > 0$.

b) Consider the initial value ODE

$$y'(t) = -100y(t) \qquad y(0) = 1\,, \tag{4.46b}$$

whose exact solution is

$$y(t) \equiv \mathrm{e}^{-100t} \quad \forall t \geqslant 0\,.$$

If one applies Euler's method to (4.46b), then our numerical approximation is $y_n = (1 - 100h)^n \;\; n \geqslant 0$. Thus the property $\lim_{n\to\infty} y_n = 0$ is now only preserved, provided the severe restriction $0 < h < 0.02$ holds. If one applies the backward Euler method to (4.46b), then our numerical approximation is $y_n = (1 + 100h)^{-n} \;\; n \geqslant 0$. Thus the property $\lim_{n\to\infty} y_n = 0$ is again preserved $\forall h > 0$.

Thus as the exact solution of our ODE converges faster to 0 as $t \to \infty$ ($\mathrm{e}^{-100t}$ compared to $\mathrm{e}^{-0.1t}$), we need a much more severe restriction on the steplength $h$ ($h < 0.02$ compared to $h < 20$) for the explicit Euler method to ensure $\lim_{n\to\infty} y_n = 0$. On the other hand, the implicit backward Euler method achieves $\lim_{n\to\infty} y_n = 0$ without any restriction on $h$.

In this course, we shall only discuss the problem of stiffness for scalar linear constant-coefficient equations of the form

$$y'(t) = \lambda y(t) \qquad y(0) = 1 \tag{4.47}$$

for $\lambda \in \mathbb{C}$. This is not quite such a limitation as it sounds, because our analysis will also be valid for linear constant-coefficient systems of the form

$$\boldsymbol{y}'(t) = \mathsf{A}\boldsymbol{y}(t) \tag{4.48}$$

where $\mathsf{A} \in \mathbb{R}^{N\times N}$. (If $\mathsf{A}$ is diagonalisable, this follows directly from an eigenvector decomposition. Otherwise we have to use the matrix exponential function defined by

$$\mathrm{e}^{t\mathsf{A}} \equiv \sum_{k=0}^{\infty} \frac{1}{k!} t^k \mathsf{A}^k\,. \tag{4.49a}$$

Since

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathrm{e}^{t\mathsf{A}}\right) = \sum_{k=1}^{\infty} \frac{1}{(k-1)!} t^{k-1}\mathsf{A}^k = \mathsf{A}\mathrm{e}^{t\mathsf{A}}\,, \tag{4.49b}$$

it follows that the solutions of (4.48) are

$$\boldsymbol{y}(t) \equiv \mathrm{e}^{t\mathsf{A}}\boldsymbol{y}(0)\,. \tag{4.49c}$$

Note that all the exact solutions of (4.48) satisfy $\lim_{t\to\infty} \boldsymbol{y}(t) = \boldsymbol{0}$ if and only if all the eigenvalues $\lambda$ of $\mathsf{A}$ satisfy $\mathrm{Re}\,\lambda < 0$. That is why we must consider (4.47) with $\lambda \in \mathbb{C}$.) However the problem of stiffness for general linear systems of the form

$$\boldsymbol{y}'(t) = \mathsf{A}(t)\boldsymbol{y}(t)\,,$$

and even more so for (4.1), is a subject of current research and well beyond the scope of this course.

### 4.4.2 Linear stability domains and A-stability

The exact solution of (4.47) satisfies $\lim_{t\to\infty} y(t) = 0$ if and only if $\mathrm{Re}\,\lambda < 0$. If we apply a chosen numerical method to (4.47) then we obtain the following analogous definition.

**Definition 4.14** (*Linear stability domain*)**.** The *linear stability domain* for a numerical method applied to (4.47) is the subset of $\mathbb{C}$ defined by

$$\mathcal{D} \equiv \left\{ h\lambda \in \mathbb{C} \,:\, \lim_{n\to\infty} y_n = 0 \right\}.$$

*Examples.*

(i) Euler's method applied to (4.47) has solution $y_n = (1 + h\lambda)^n \quad n \geqslant 0$ and so its linear stability domain is

$$\mathcal{D} = \{z \in \mathbb{C} : |1 + z| < 1\} ;$$

i.e. the domain inside the unit circle centered on -1, as illustrated in Figure 4.8.

(ii) The backward Euler method applied to (4.47) has solution $y_n = (1-h\lambda)^{-n}$ and so its linear stability domain is

$$\mathcal{D} = \{z \in \mathbb{C} : |1 - z| > 1\} ;$$

i.e. the domain outside the unit circle centred on 1.

(iii) The trapezoidal rule (4.12b) applied to (4.47) has solution

$$y_n = \left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}\right)^n ,$$

provided that $\lambda h \neq 2$, and so its linear stability domain is

$$\mathcal{D} = \left\{z \in \mathbb{C} : \left|\frac{2 + z}{2 - z}\right| < 1\right\} .$$

Since this inequality is merely saying that $z$ must be closer to -2 than it is to 2; we have $\mathcal{D}$ equal to the left half-plane $\mathbb{C}^- = \{z \in \mathbb{C} : \operatorname{Re} z < 0\}$.

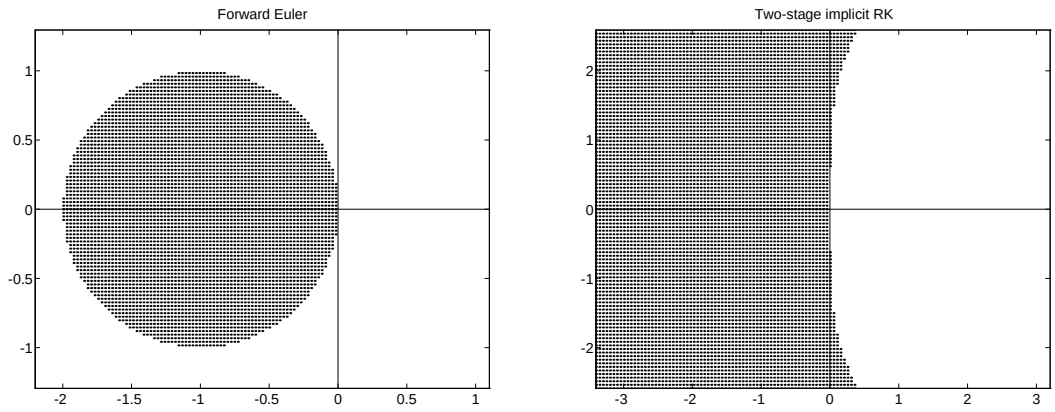



**Figure 4.8:** Linear stability domains: Euler's method and a 2-stage implicit RK. See also the *A-Stabilty* demonstration at http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php.

Now we introduce a definition that tests whether a numerical method preserves the structure of solutions for (4.47).

**Definition 4.15** (*A-stability*)**.** A numerical method is *A-stable* if $\mathbb{C}^- \subseteq \mathcal{D}$.

Thus we have shown that both the backward Euler method and the trapezoidal rule are A-stable, but Euler's method is not. (Note that A-stability does not mean that *any* step size will do! We still need to choose $h$ small enough to ensure the required accuracy.)

In general it is very difficult for either multistep methods or explicit methods to be A-stable, as the following two famous results state.

(i) No multistep method of order $p \geqslant 3$ can be A-stable. (This is called the second Dahlquist barrier theorem.) The $p = 2$ barrier is attained by the trapezoidal rule, which can be viewed as the 1-step Adams-Moulton method.

(ii) No explicit RK method can be A-stable. (Although, as we shall see in the next section, implicit RK methods may be A-stable.)

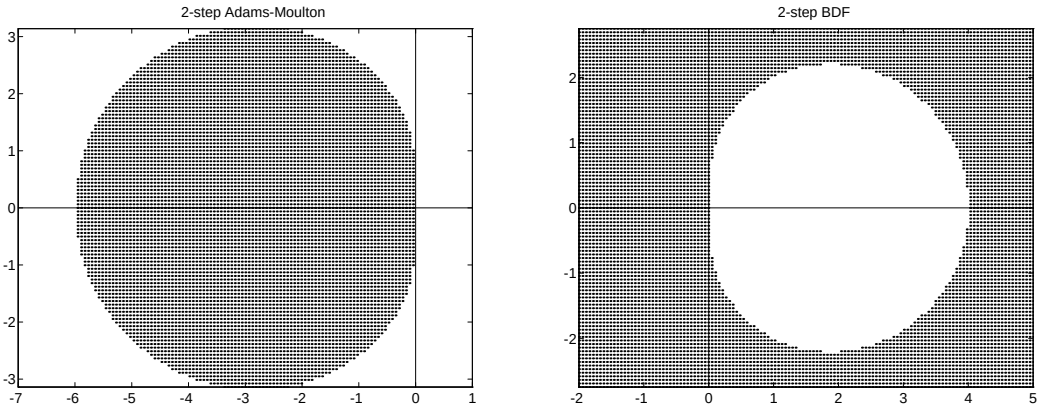



**Figure 4.9:** Linear stability domains: 2-step Adams-Moulton and 2-step BDF. See also the *A-Stabilty* demonstration at http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php.

Although no multistep method of order $p \geqslant 3$ may be A-stable, the convergent BDF methods of section 4.2.3 are often satisfactory for many stiff equations; even though only the 2-step BDF method is A-stable. This is because in many 'real-life' stiff *linear*[6] systems, the eigenvalues are not just in $\mathbb{C}^-$ but also well away from $\mathrm{i}\mathbb{R}$ and often near the negative real axis. Hence numerical schemes whose linear stability domains only include this part of $\mathbb{C}^-$ can still be useful. This is the case for all BDF methods of order $p \leqslant 6$ (i.e. all convergent BDF methods), since their linear stability domains include a wedge about $(-\infty, 0)$ in $\mathbb{C}^-$. (Such methods are said to be $A_0$*-stable*.)

### 4.4.3 A-stability and the maximum principle

Often the easiest way of proving that a numerical method is A-stable is by making use of the *maximum principle* in complex analysis. This supposes that our numerical method applied to (4.47) has solution

$$y_n = [r(h\lambda)]^n \quad n \geqslant 0\,,$$

where $r$ is a rational function. Thus the linear stability domain is

$$\mathcal{D} = \{z \in \mathbb{C} \,:\, |r(z)| < 1\}$$

and so our numerical method is A-stable if we can prove that

$$z \in \mathbb{C}^- \quad \Rightarrow \quad |r(z)| < 1\,.$$

We need the following form of the maximum principle from *Complex Analysis*[7].

**Theorem 4.16** (Maximum modulus principle)**.** *Let $\Omega \subset \mathbb{C}$ be an open set and suppose $g : \Omega \mapsto \mathbb{C}$ is analytic and non-constant: then $|g|$ has no maximum in $\Omega$.*

We the apply this theorem with $\Omega \equiv \mathbb{C}^-$ and $g \equiv r$, checking the following conditions:

- $r$ is analytic in $\mathbb{C}^-$;
- $|r(\mathrm{i}t)| \leqslant 1 \quad \forall t \in \mathbb{R}$;
- $\lim_{|z|\to\infty\,,\mathrm{Re}\,z<0} |r(z)| \leqslant 1$.

We have stated that, unlike multistep methods, implicit high-order RK methods may be A-stable; thus we give two simple examples where Theorem 4.16 can easily establish this result.

[6] The analysis of *nonlinear* stiff equations is difficult and well outside the scope of this course.
[7] If you are taking *Complex Methods* then this is your chance to learn the statement of the *maximum modulus principle*.

1) Consider the 2-stage 3rd-order method (4.41a)-(4.41c): so applying this scheme to (4.47) gives

$$hk_1 = h\lambda\left(y_n + \tfrac{1}{4}hk_1 - \tfrac{1}{4}hk_2\right),$$
$$hk_2 = h\lambda\left(y_n + \tfrac{1}{4}hk_1 + \tfrac{5}{12}hk_2\right).$$

This is a linear system for $hk_1$ and $hk_2$, whose solution is

$$\begin{bmatrix} hk_1 \\ hk_2 \end{bmatrix} = \begin{bmatrix} 1-\frac{1}{4}h\lambda & \frac{1}{4}h\lambda \\ -\frac{1}{4}h\lambda & 1-\frac{5}{12}h\lambda \end{bmatrix}^{-1} \begin{bmatrix} h\lambda y_n \\ h\lambda y_n \end{bmatrix} = \frac{h\lambda y_n}{1-\frac{2}{3}h\lambda+\frac{1}{6}(h\lambda)^2} \begin{bmatrix} 1-\frac{2}{3}h\lambda \\ 1 \end{bmatrix},$$

and therefore

$$y_{n+1} = y_n + \tfrac{1}{4}hk_1 + \tfrac{3}{4}hk_2 = \frac{1+\frac{1}{3}h\lambda}{1-\frac{2}{3}h\lambda+\frac{1}{6}h^2\lambda^2} y_n\,.$$

Thus we have $y_{n+1} = r(h\lambda)y_n$, where

$$r(z) \equiv \frac{1+\frac{1}{3}z}{1-\frac{2}{3}z+\frac{1}{6}z^2}\,,$$

and so

$$y_n = [r(h\lambda)]^n \quad n \geqslant 0\,.$$

Since the poles of the rational function $r$ are $2 \pm \sqrt{2}\mathrm{i}$, we know that $r$ is analytic in $\mathbb{C}^-$. $|r| \leqslant 1$ on the imaginary axis, because

$$|r(\mathrm{i}t)|^2 = \frac{36+4t^2}{36+4t^2+t^4}\,.$$

$|r|$ tends to zero on the rest of the boundary of $\mathbb{C}^-$, because the denominator is of higher degree than the numerator. Hence we can apply Theorem 4.16 to prove that $|r| < 1$ in $\mathbb{C}^-$ and so the method is A-stable.

2) Consider the 2-stage order 4 Gauss–Legendre method in (4.42), so applying this numerical scheme to (4.47) gives

$$hk_1 = z\Big(y_n + \tfrac{1}{4}hk_1 + \big(\tfrac{1}{4}-\tfrac{\sqrt{3}}{6}\big)hk_2\Big), \tag{4.50a}$$
$$hk_2 = z\Big(y_n + \big(\tfrac{1}{4}+\tfrac{\sqrt{3}}{6}\big)hk_1 + \tfrac{1}{4}hk_2\Big). \tag{4.50b}$$

This is a linear system for $hk_1$ and $hk_2$, whose solution is

$$\begin{bmatrix} hk_1 \\ hk_2 \end{bmatrix} = \begin{bmatrix} 1-\frac{1}{4}z & -(\frac{1}{4}-\frac{\sqrt{3}}{6})z \\ -(\frac{1}{4}+\frac{\sqrt{3}}{6})z & 1-\frac{1}{4}z \end{bmatrix}^{-1} \begin{bmatrix} zy_n \\ zy_n \end{bmatrix}$$
$$= \frac{1}{\det(\mathsf{B})} \begin{bmatrix} 1-\frac{1}{4}z & (\frac{1}{4}+\frac{\sqrt{3}}{6})z \\ (\frac{1}{4}-\frac{\sqrt{3}}{6})z & 1-\frac{1}{4}z \end{bmatrix} \begin{bmatrix} zy_n \\ zy_n \end{bmatrix},$$

where

$$\mathsf{B} \equiv \begin{bmatrix} 1-\frac{1}{4}z & -(\frac{1}{4}-\frac{\sqrt{3}}{6})z \\ -(\frac{1}{4}+\frac{\sqrt{3}}{6})z & 1-\frac{1}{4}z \end{bmatrix}.$$

Since we only need

$$hk_1 + hk_2 = \tfrac{2z}{\det(\mathsf{B})}\, y_n = \tfrac{2z}{1-\frac{1}{2}z+\frac{1}{12}z^2}\, y_n\,,$$

it is easy to obtain

$$y_{n+1} = y_n + \tfrac{1}{2}(hk_1+hk_2) = \frac{1+\frac{1}{2}z+\frac{1}{12}z^2}{1-\frac{1}{2}z+\frac{1}{12}z^2}\, y_n\,; \tag{4.51}$$

thus $y_n = [r(z)]^n$, where

$$r(z) \equiv \frac{1+\frac{1}{2}z+\frac{1}{12}z^2}{1-\frac{1}{2}z+\frac{1}{12}z^2}\,. \tag{4.52}$$

$r$ is a rational function, and its only singularities are the poles $3 \pm \mathrm{i}\sqrt{3}$ which lie in the right half-lane; hence $r$ is analytic in $\mathbb{C}^-$. On the imaginary axis, we have $|r| = 1$: similarly

$$\lim_{|z|\to\infty} |r(z)| = 1\,.$$

Hence we can apply Theorem 4.16 to prove that $|r| < 1$ in $\mathbb{C}^-$ and so this Gauss–Legendre method is A-stable. (Moreover, since $r(z) = \frac{1}{r(-z)}$, we deduce the equality $\mathcal{D} = \mathbb{C}^-$.)

## 4.5 Implementation of ODE methods

Throughout this chapter on numerical methods for computing approximations to exact solutions of initial-value ODEs, we have simplified matters by assuming that the step size $h$ is constant. In practice, and especially under control of a well-written software package, this will not be the case: i.e. the step size $h_n \equiv t_{n+1} - t_n$ will vary with $n$. Moreover, the step size $h$ is not a pre-ordained quantity chosen by the user, but a parameter of the method. More precisely, the basic input for a software package (i.e. chosen by the user) is *not* the step size but rather an *error tolerance:* this being the accuracy of the numerical approximation that the user requires. The software package then chooses the step length $h_n$ (varying with $n$ in general) to bound a local estimate of the error by the user-given error tolerance. It is this strategy that is called a *time-stepping algorithm.* In the present section, we shall briefly describe a few of the key ideas behind such algorithms for *error control.*

### 4.5.1 Error control for multistep methods

As a simple example, suppose that we wish to monitor the error for the trapezoidal rule

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \tfrac{1}{2}h\left\{\boldsymbol{f}(t_n, \boldsymbol{y}_n) + \boldsymbol{f}(t_{n+1}\boldsymbol{y}_{n+1})\right\}\,, \tag{4.53a}$$

which we already know has order 2. If we substitute the exact solution $\boldsymbol{y}(t)$ into (4.53a) then (as in (4.14) with $\theta = \frac{1}{2}$) we deduce that

$$\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}(t_n) - \tfrac{1}{2}h\left\{\boldsymbol{y}'(t_n) + \boldsymbol{y}'(t_{n+1})\right\} = c_{\mathrm{TR}}h^3\boldsymbol{y}'''(t_n) + \mathcal{O}\big(h^4\big)\,, \tag{4.53b}$$

where

$$c_{\mathrm{TR}} = -\tfrac{1}{12}\,. \tag{4.53c}$$

To estimate the *local* error in this single step from $t_n$ to $t_{n+1}$, we *assume* that $\boldsymbol{y}_n = \boldsymbol{y}(t_n)$ (i.e. *no* error has been committed so far) and use (4.53a) and (4.53b) to produce the approximation

$$\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}_{n+1}^{\mathrm{TR}} \approx c_{\mathrm{TR}}h^3\boldsymbol{y}'''(t_n) + \mathcal{O}\big(h^4\big)\,. \tag{4.53d}$$

Even this estimate does not seem much help, since the value $\boldsymbol{y}'''(t_n)$ is unknown. We will see however that this objection can be circumvented.

*Definition.* $c_{\mathrm{TR}}$ is called the *error constant* for the trapezoidal rule.

Each multistep method (but not RK!) has its own error constant, and this can be obtained from its local truncation error. For example, the 2nd order 2-step Adams–Bashforth method of (4.15),

$$\boldsymbol{y}_{n+1} - \boldsymbol{y}_n = \tfrac{1}{2}h\left\{3\boldsymbol{f}(t_n, \boldsymbol{y}_n) - \boldsymbol{f}(t_{n-1}, \boldsymbol{y}_{n-1})\right\}\,, \tag{4.54a}$$

has error constant $c_{\mathrm{AB}} \equiv \frac{5}{12}$ in (4.23b), i.e.

$$\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}_{n+1}^{\mathrm{AB}} \approx c_{\mathrm{AB}}h^3\boldsymbol{y}'''(t_n) + \mathcal{O}(h^4)\,. \tag{4.54b}$$

*Milne's device* is a technique for estimating the local error and thus choosing the step size $h$ to achieve a given error tolerance. It uses two multistep methods of the same order, one explicit and one implicit. It is the implicit method that produces our approximation for the exact solution $\boldsymbol{y}(t_{n+1})$; the explicit method

is just used to provide a local error estimate for this approximation. We illustrate the use of Milne's device with the above two 2nd order multistep methods; the explicit Adams–Bashforth method (4.54a) and the implicit trapezoidal rule (4.53a). For both of these we have the local error estimates (4.54b) and (4.53d) so that, if we neglect the higher powers of $h$, we may eliminate $\boldsymbol{y}'''(t_n)$ and obtain a *computable* local error estimate for the trapezoidal rule: i.e.

$$\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}_{n+1}^{\mathrm{TR}} \approx -\frac{c_{\mathrm{TR}}}{c_{\mathrm{AB}} - c_{\mathrm{TR}}} \left[\boldsymbol{y}_{n+1}^{\mathrm{AB}} - \boldsymbol{y}_{n+1}^{\mathrm{TR}}\right] = \tfrac{1}{6}\left(\boldsymbol{y}_{n+1}^{\mathrm{AB}} - \boldsymbol{y}_{n+1}^{\mathrm{TR}}\right) . \tag{4.55}$$

In the next paragraph, we shall explain how this estimate can be used to choose the step length $h$.

*Remark.* TR is a far better method than AB: it is A-stable, hence its *global* behaviour is superior. Employing AB to estimate the local error adds very little to the overall cost of TR, since AB is an explicit method.

The usual way of setting up error control for multistep methods is to employ a *predictor-corrector pair*: that is, we use two multistep methods of the same order, one explicit (the predictor) and the other implicit (the corrector). For example, the 2nd order Adams–Bashforth method and the trapezoidal rule just discussed. A slightly less simple example would be the third-order Adams–Bashforth and Adams–Moulton pair, i.e.

$$\text{Predictor}: \quad \boldsymbol{y}_{n+2}^{\mathrm{P}} = \boldsymbol{y}_{n+1} + h\left[\tfrac{5}{12}\boldsymbol{f}(t_{n-1},\boldsymbol{y}_{n-1}) - \tfrac{4}{3}\boldsymbol{f}(t_n,\boldsymbol{y}_n) + \tfrac{23}{12}\boldsymbol{f}(t_{n+1},\boldsymbol{y}_{n+1})\right], \tag{4.56a}$$

$$\text{Corrector}: \quad \boldsymbol{y}_{n+2}^{\mathrm{C}} = \boldsymbol{y}_{n+1} + h\left[-\tfrac{1}{12}\boldsymbol{f}(t_n,\boldsymbol{y}_n) + \tfrac{2}{3}\boldsymbol{f}(t_{n+1},\boldsymbol{y}_{n+1}) + \tfrac{5}{12}\boldsymbol{f}(t_{n+2},\boldsymbol{y}_{n+2})\right] . \tag{4.56b}$$

The predictor is employed not just to estimate the local error for the corrector using Milne's device, but also to provide a *good initial guess* for the solution of the implicit corrector equations. (This topic is briefly discussed in §4.5.5.) Typically, for nonstiff ODEs we iterate the correction equations at most twice; while stiff ODEs require *iteration to convergence*, otherwise the usual superior features of the corrector are lost.

Depending on whether an error tolerance has been achieved, we amend the step size $h$. To this end let $\varepsilon_{\mathrm{TOL}} > 0$ be a user-specified *tolerance:* the maximal error that we wish to allow in approximating the ODE. Having completed a single step and estimated the error, there are three possibilities:

(a) $\alpha\,\varepsilon_{\mathrm{TOL}} \leqslant \|\,\mathrm{error}\,\| \leqslant \varepsilon_{\mathrm{TOL}}$, say with $\alpha = \frac{1}{10}$: accept the step, continue to $t_{n+2}$ with the same step size;

(b) $\|\,\mathrm{error}\,\| < \alpha\,\varepsilon_{\mathrm{TOL}}$: accept the step and increase the step length;

(c) $\|\,\mathrm{error}\,\| > \varepsilon_{\mathrm{TOL}}$: reject the step, recommence integration from $t_n$ with smaller $h$.

In the case of (b) and (c), we have to ask the question: how are the unknown previous approximations (e.g. $\boldsymbol{y}_n$ and $\boldsymbol{y}_{n-1}$ in (4.56a)) obtained if the step length is changed? The answer is that they are obtained by suitable polynomial interpolation from the other approximations calculated with different step lengths.

*Error estimation per unit step.* Let $\boldsymbol{e}$ be our estimate of *local* error. Then $\boldsymbol{e}/h$ is our estimate for the global error in an interval of unit length. It is usual to require the latter quantity not to exceed $\varepsilon_{\mathrm{TOL}}$ since good implementations of numerical ODEs should monitor the accumulation of *global* error. This is called *error estimation per unit step.*

*Demonstration.* See also the *Predictor-Corrector Methods* demonstration at

http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php.

### 4.5.2 Error control for Runge–Kutta methods

The strategy used for error control of multistep methods (i.e. Milne's device with predictor-corrector pairs) cannot be applied to RK methods. This is because the nonlinear nature of RK methods means that the leading term in the local truncation error is no longer simply an error constant multiplying a derivative of the exact solution. We replace the idea of predictor-corrector pairs by *embedded Runge–Kutta* methods, where a lower-order RK method is "hidden inside" a higher-order RK method.

In slightly more detail, the embedded RK approach requires two (typically explicit) RK methods: one having $\nu$ stages and order $p$, while the other has $\nu+\ell$ stages (with $\ell \geqslant 1$) and order $p+1$. The key restriction is that the *first $\nu$ stages* of both methods must be identical. This restriction ensures that the cost of implementing the higher-order method is marginal, once we have computed the lower-order approximation. A simple example would be

$$\left.\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}(t_n, \boldsymbol{y}_n), \\ \boldsymbol{k}_2 &= \boldsymbol{f}\big(t_n + \tfrac{1}{2}h, \boldsymbol{y}_n + \tfrac{1}{2}h\boldsymbol{k}_1\big), \\ \boldsymbol{y}^{[1]}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{k}_2; \end{aligned}\right\} \text{order 2, local error } \mathcal{O}\big(h^3\big)$$

$$\left.\begin{aligned} \boldsymbol{k}_3 &= \boldsymbol{f}(t_n + h, \boldsymbol{y}_n - h\boldsymbol{k}_1 + 2h\boldsymbol{k}_2), \\ \boldsymbol{y}^{[2]}_{n+1} &= \boldsymbol{y}_n + \tfrac{1}{6}h(\boldsymbol{k}_1 + 4\boldsymbol{k}_2 + \boldsymbol{k}_3)\,. \end{aligned}\right\} \text{order 3, local error } \mathcal{O}\big(h^4\big)$$

Here we can estimate the error for the order 2 method by

$$\boldsymbol{y}^{[1]}_{n+1} - \boldsymbol{y}(t_{n+1}) \approx \boldsymbol{y}^{[1]}_{n+1} - \boldsymbol{y}^{[2]}_{n+1}\,. \tag{4.57}$$

(While it might look paradoxical, at least at first glance, the only purpose of the higher-order method is to provide an error estimate for the lower-order one.) Once a local error estimate is available, so that it can be compared with the user-supplied error tolerance, the time-stepping strategy described in the last section can be carried out.

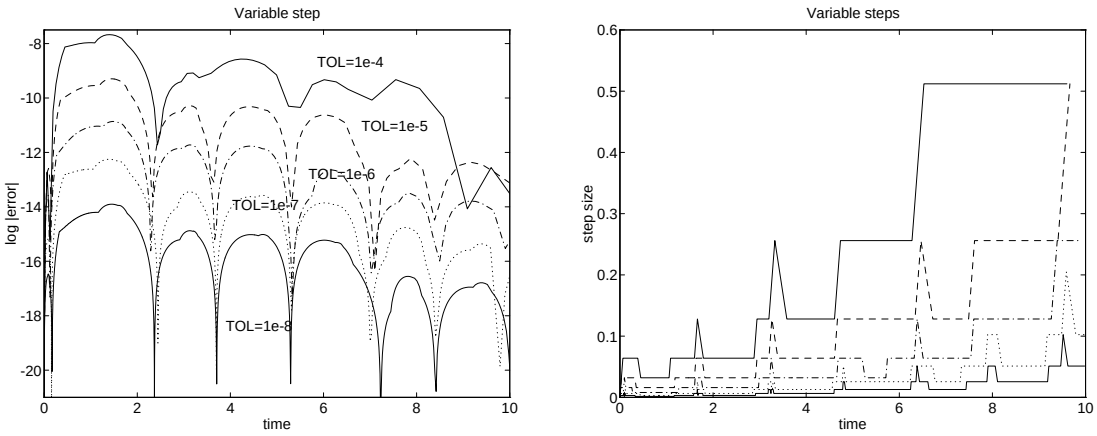



**Figure 4.10:** Adjustments of the step size in the solution of the equation $y' = -y + 2e^{-t}\cos 2t$, $y(0) = 0$.

### 4.5.3 The Zadunaisky device

This is a general technique for obtaining error estimates for numerical approximations of initial-value ODEs.

Suppose we have used an arbitrary numerical method of order $p$ to approximate (4.1) and that we have stored the previously computed values $\boldsymbol{y}_n, \boldsymbol{y}_{n-1}, \ldots, \boldsymbol{y}_{n-p}$. (The time steps $h_i \equiv t_{i+1} - t_i$ $i = n-p, \ldots, n-1$ need not be equal.) We construct the $p^{\text{th}}$ degree interpolating polynomial (with vector coefficients) $\boldsymbol{d}$, such that

$$\boldsymbol{d}(t_{n-i}) = \boldsymbol{y}_{n-i} \quad i = 0, 1, \ldots, p\,,$$

and consider the initial-value ODE

$$\boldsymbol{z}'(t) = \boldsymbol{f}(t, \boldsymbol{z}(t)) + \big\{\boldsymbol{d}'(t) - \boldsymbol{f}(t, \boldsymbol{d})\big\} \qquad t \in [t_n, t_{n+1}] \tag{4.58}$$

with $\boldsymbol{z}(t_n) = \boldsymbol{y}_n$. There are two important observations with regard to (4.58):

(i) Since $\boldsymbol{d}(t) - \boldsymbol{y}(t) = \mathcal{O}\big(h^{p+1}\big)$ and $\boldsymbol{y}'(t) \equiv \boldsymbol{f}(t, \boldsymbol{y}(t))$, the term $\boldsymbol{d}'(t) - \boldsymbol{f}(t, \boldsymbol{d})$ is usually small. Therefore, (4.58) is a small perturbation of the original ODE.

(ii) The exact solution of (4.58) is $\boldsymbol{z}(t) = \boldsymbol{d}(t)$.

So, having applied our numerical method to (4.1) to produce $\boldsymbol{y}_{n+1}$, we apply *exactly the same numerical method and implementation details* to (4.58) to produce $\boldsymbol{z}_{n+1}$. We then evaluate the error in $\boldsymbol{z}_{n+1}$, namely $\boldsymbol{z}_{n+1} - \boldsymbol{d}(t_{n+1})$, and use it as an estimate of the error in $\boldsymbol{y}_{n+1}$.

### 4.5.4 Not the last word

There is still very much more that we could say on the numerical solution of ODEs (let alone PDEs). We have tended to concentrate on the accuracy of solutions and error control. However, many equations have extra properties that we have not addressed, e.g. the solutions might be constrained to surfaces, or in physics, the system might conserve energy (in which case it might be good if the numerical scheme conserved 'energy'). A preliminary discussion of one of these points is given on the *Symplectic Integrators* page at

http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php.

There it is shown that in certain circumstances a modified first-order Euler method can have advantages over a higher-order adaptive RK method.

### 4.5.5 Solving nonlinear algebraic systems

We have already observed that the implementation of an *implicit* ODE method, whether multistep or RK, requires the solution of a nonlinear (in general) system of algebraic equations at each step. We use the simple backward Euler method to illustrate the two commonest types of iterative nonlinear solver.

a) Functional iteration

$$\boldsymbol{y}_{n+1}^{[k+1]} = \boldsymbol{y}_n + h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}^{[k]}) \qquad \boldsymbol{y}_{n+1}^{[0]} = \boldsymbol{y}_n\,. \tag{4.59}$$

b) Newton's method

$$\Big[\mathsf{I} - h\mathsf{J}^{[k]}\Big]\,\boldsymbol{z}^{[k]} = \boldsymbol{y}_{n+1}^{[k]} - \Big\{\boldsymbol{y}_n + h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}^{[k]})\Big\} \qquad \boldsymbol{y}_{n+1}^{[0]} = \boldsymbol{y}_n\,, \tag{4.60}$$

where $\boldsymbol{y}_{n+1}^{[k+1]} \equiv \boldsymbol{y}_n^{[k]} - \boldsymbol{z}^{[k]}$ and

$$\mathsf{J}^{[k]} \equiv \boldsymbol{\nabla}\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}^{[k]}) \in \mathbb{R}^{N\times N}$$

is the *Jacobian matrix* for $\boldsymbol{f}$.

We make the following remarks about the suitability of these two iterative methods for general implicit numerical methods applied to (4.1).

1) Unlike Newton's method, functional iteration requires neither the solution of $N \times N$ linear systems nor the computation of Jacobian matrices for $\boldsymbol{f}$. Hence it has a much lower computational cost than Newton's method.

2) Functional iteration can only be guaranteed to converge under the step length restriction $\lambda h < 1$, where $\lambda$ is the Lipschitz constant for $\boldsymbol{f}$ in (4.2). Especially for stiff equations, this restriction can lead to very small time-steps and a large amount of computation.

3) It is possible to use variants of Newton's method, which may be more efficient. For example, the so-called *modified* Newton's method replaces (4.60) with

$$\left[\mathsf{I} - h\mathsf{J}^{[0]}\right] \boldsymbol{z}^{[k]} = \boldsymbol{y}_{n+1}^{[k]} - \left\{\boldsymbol{y}_n + h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}^{[k]})\right\} \qquad \boldsymbol{y}_{n+1}^{[0]} = \boldsymbol{y}_n \,. \tag{4.61}$$

Thus only *one* Jacobian matrix needs to be evaluated at each time-step and only $N \times N$ linear systems with the *same* coefficient matrix (but different rhs's!) need to be solved. As we shall see in §5.2, this can lead to substantial savings. Hence in some cases, (4.61) may be preferred to (4.60): despite the fact that, in general, (4.61) requires more iterations to converge than (4.60).

4) The only role the Jacobian matrix plays in (4.60) and (4.61) is to ensure convergence: its precise value makes no difference to $\lim_{k\to\infty} \boldsymbol{y}_n^{[k]}$. Therefore we might replace it with a finite-difference approximation and/or evaluate it once every several steps.

### 4.5.6 *A distraction*

In Part IB we only discuss the numerical solution of ordinary differential equations; in Part II the numerical solution of partial differential equations will be touched upon.

One of the most well-known nonlinear partial differential equations is the Navier-Stokes equation which describes Newtonian viscous flow. There are very few analytical solutions for this important equation, with the result that numerical solutions play a crucial role in fields ranging from the motion of bacteria and other living organisms, aerodynamics and climate change.

To see numerical solutions of the Navier-Stokes equation in real time you can download `FI1.2.zip` to a Windows computer from

`www.imperial.ac.uk/aeronautics/fluiddynamics/FI/InteractiveFlowIllustrator.htm`

More information can be found at this URL, but one of the main goals of this *Interactive Flow Illustrator* is easiness to use, so, rather than reading manuals etc. one should just download it, unzip it, click on `IFI.exe`, and see if one can make sense of what one gets.

*Some technical details.* Flow Illustrator solves the Navier-Stokes equations on a uniform Cartesian grid, with the grid step equal to one pixel of the input bitmap. Finite differences are used. The embedded boundary method is used to represent the body shape. This means that the equations are solved in a rectangular domain including the areas inside the body, and a body force is added inside the body so as to make the velocity of the fluid equal to the velocity of the body. Both viscous and inviscid terms are modelled implicitly, so that there are no stability constraints on the time step. The pressure equation is solved (or, rather, a projection on a solenoidal subspace is done) using fast Fourier transforms. Velocity is prescribed on the left boundary of the computational domain, and soft boundary conditions are applied on other boundaries.

# 5 Square Linear Systems and the LU factorisation

In this section, we shall be concerned with one of the most basic problems in computational linear algebra.

**Problem 5.1.** Given $\mathsf{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$, find $\boldsymbol{x} \in \mathbb{R}^n$ to satisfy

$$\mathsf{A}\boldsymbol{x} = \boldsymbol{b}\,. \tag{5.1}$$

*Remark.* This is the fourth time that the solution of linear equations has been addressed in the Tripos, and it will not be the last. This level of attention gives an indication of the importance of the subject (if not its excitement).

As discussed in *Vectors & Matrices*, the theoretical situation for Problem 5.1 is clear.

- If $\mathsf{A}$ is non-singular then, for each $\boldsymbol{b} \in \mathbb{R}^n$, there is a unique $\boldsymbol{x} \in \mathbb{R}^n$ that satisfies (5.1).
- If $\mathsf{A}$ is singular then, depending on $\boldsymbol{b} \in \mathbb{R}^n$, either there is *no* $\boldsymbol{x} \in \mathbb{R}^n$ that satisfies (5.1) or there are *infinitely many* $\boldsymbol{x} \in \mathbb{R}^n$ that satisfy (5.1).

We shall mainly be interested in the case of non-singular $\mathsf{A}$, although some results for singular $\mathsf{A}$ will be mentioned. Moreover, our primary concern is the description of efficient practical algorithms for solving (5.1). As also discussed in *Vectors & Matrices*, if $\mathsf{A}$ is non-singular we can write

$$\left(\mathsf{A}^{-1}\right)_{i,j} = \frac{1}{\det \mathsf{A}}\Delta_{j,i} \qquad i,j = 1,\ldots,n\,, \tag{5.2a}$$

where $\Delta_{i,j}$ is the cofactor of the $(i,j)^{\text{th}}$ element of the matrix $\mathsf{A}$. The required determinants can be evaluated by

$$\det \mathsf{A} = \sum_{i_1 i_2 \ldots i_n} \varepsilon_{i_1 i_2 \ldots i_n} A_{i_1,1} A_{i_2,2} \ldots A_{i_n,n}\,, \tag{5.2b}$$

or using the recursive definition of a determinant: hence, using these formulae, (5.1) can be solved explicitly by Cramer's rule. Unfortunately, the number of operations increases like $(n+1)!$ and thus, on a $10^{10}$ flop/sec. computer, the time required is

$$n = 10 \quad \Rightarrow \quad 10^{-5} \text{ sec}, \qquad n = 20 \quad \Rightarrow \quad 1\tfrac{3}{4} \text{ min}, \qquad n = 30 \quad \Rightarrow \quad 4 \times 10^4 \text{ years}.$$

This is impractical and so we will look at more efficient algorithms for both solving (5.1) and computing $\det \mathsf{A}$.

## 5.1 Triangular matrices

Problem 5.1 is easy to deal with (both theoretically and practically) when $\mathsf{A}$ is a *triangular* matrix.

If $\mathsf{L} \in \mathbb{R}^{n\times n}$ is a *lower* triangular matrix, i.e. $L_{i,j} = 0$ if $i < j$, then

$$\det \mathsf{L} = \prod_{i=1}^{n} L_{i,i}\,. \tag{5.3a}$$

Hence the singularity or non-singularity of $\mathsf{L}$ is immediately obvious. Moreover the problem $\mathsf{L}\boldsymbol{x} = \boldsymbol{b}$ can be solved in $\mathcal{O}(n^2)$ computional operations[8] by so-called *forward substitution*

$$x_i = \frac{1}{L_{i,i}}\left\{ b_i - \sum_{j=1}^{i-1} L_{i,j} x_j \right\}, \qquad i = 1,\ldots,n\,. \tag{5.3b}$$

[8] Where, as usual, we only bother to count multiplications/divisions.

(Here the standard convention for limits of sums is used.) For example

$$\begin{bmatrix} 1 & & \\ -2 & 1 & \\ 3 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ 5 \end{bmatrix} \begin{matrix} \downarrow \\ \downarrow \\ \downarrow \end{matrix} \quad \Rightarrow \quad \boldsymbol{x} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}.$$

This approach also gives us the columns of $\mathsf{L}^{-1}$ by solving

$$\mathsf{L}\boldsymbol{y}_j = \boldsymbol{e}_j \qquad j = 1, \ldots, n\,, \tag{5.4}$$

where $\boldsymbol{e}_j \in \mathbb{R}^n$ denotes the $j^{\text{th}}$ unit vector (i.e. the $j^{\text{th}}$ column of $\mathsf{I} \in \mathbb{R}^{n\times n}$). In addition, solving (5.4) by means of (5.3b) immediately shows that $\mathsf{L}^{-1}$ is also lower triangular.

Similarly, if $\mathsf{U} \in \mathbb{R}^{n\times n}$ is an *upper* triangular matrix, i.e. $U_{i,j} = 0$ if $i > j$, then

$$\det \mathsf{U} = \prod_{i=1}^{n} U_{i,i}\,. \tag{5.5a}$$

Hence the singularity or non-singularity of $\mathsf{U}$ is again immediately obvious. Moreover the problem $\mathsf{U}\boldsymbol{x} = \boldsymbol{b}$ can be solved in $\mathcal{O}\big(n^2\big)$ operations by so-called *back substitution*

$$x_i = \frac{1}{U_{i,i}} \left\{ b_i - \sum_{j=i+1}^{n} U_{i,j} x_j \right\}, \qquad i = n, \ldots, 1\,. \tag{5.5b}$$

For example

$$\begin{bmatrix} -3 & 2 & 3 \\ & 2 & 0 \\ & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} \begin{matrix} \uparrow \\ \uparrow \\ \uparrow \end{matrix} \quad \Rightarrow \quad \boldsymbol{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

This approach again gives us the columns of $\mathsf{U}^{-1}$ by solving

$$\mathsf{U}\boldsymbol{y}_j = \boldsymbol{e}_j \qquad j = 1, \ldots, n\,. \tag{5.6}$$

In addition, solving (5.6) by means of (5.5b) immediately shows that $\mathsf{U}^{-1}$ is also upper triangular.

We can combine the above two results in a clever idea: if we could factorize a general $\mathsf{A} \in \mathbb{R}^{n\times n}$ as

$$\mathsf{A} = \mathsf{L}\mathsf{U}\,,$$

where $\mathsf{L} \in \mathbb{R}^{n\times n}$ and $\mathsf{U} \in \mathbb{R}^{n\times n}$ are lower and upper triangular matrices respectively, then the solution of (5.1) just becomes $\mathsf{A}\boldsymbol{x} = \mathsf{L}\mathsf{U}\boldsymbol{x} = \mathsf{L}(\mathsf{U}\boldsymbol{x}) = \boldsymbol{b}$: i.e. we can split (5.1) into the two simple cases

$$\mathsf{L}\boldsymbol{y} = \boldsymbol{b} \quad \text{and} \quad \mathsf{U}\boldsymbol{x} = \boldsymbol{y}\,. \tag{5.7}$$

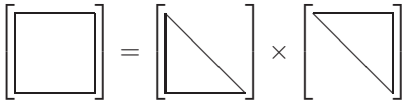
Both latter systems are triangular and can be solved in $\mathcal{O}\big(n^2\big)$ operations using (5.3b) and (5.5b) respectively (see also (5.12b)).

## 5.2 LU factorization and its generalization

In the light of the final paragraph of the previous section, we introduce the following definition.

**Definition 5.2** (LU *factorisation).* If $\mathsf{A} \in \mathbb{R}^{n\times n}$ then $\mathsf{A} = \mathsf{L}\mathsf{U}$ is an LU factorization of $\mathsf{A}$ if $\mathsf{U} \in \mathbb{R}^{n\times n}$ is an upper triangular matrix and $\mathsf{L} \in \mathbb{R}^{n\times n}$ is a *unit* lower triangular matrix.

Thus in picture form we have the following decomposition.



*Remarks*

(a) A *unit* triangular matrix has ones down the principal diagonal and hence is non-singular.

(b) It is only by tradition/convention that $\mathsf{L}$ is chosen unit lower triangular; it would be equally valid for $\mathsf{U}$ to be chosen unit upper triangular. In order for an $\mathsf{LU}$ factorisation to have a chance of being unique however, *some* normalisation for $\mathsf{L}$ and/or $\mathsf{U}$ must be imposed.

(c) We permit the possibility of $\mathsf{A}$ being singular in our definition: thus $\mathsf{L}$ is non-singular, but $\mathsf{A}$ and $\mathsf{U}$ are either both singular or both non-singular. However, we will be mainly concerned with the case of non-singular $\mathsf{A}$.

(d) A non-singular $\mathsf{A}$ may have *no* $\mathsf{LU}$ factorization, e.g.

$$\mathsf{A} \equiv \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2} \ :$$

if one exists however, it is unique and $\mathsf{U}$ is non-singular.

(e) A singular $\mathsf{A}$ may have an $\mathsf{LU}$ factorization, e.g. see Question 1 on Example Sheet 3, but in this case $\mathsf{U}$ must also be singular. We will return to this point subsequently (e.g. see Theorem 5.10 and the examples following it).

### 5.2.1 The construction of an LU factorization

Let $\{\boldsymbol{l}_j\}_{j=1}^n \subset \mathbb{R}^n$ denote the columns of $\mathsf{L}$ and let $\{\boldsymbol{u}_i^{\mathrm{T}}\}_{i=1}^n$ denote the rows of $\mathsf{U}$, where $\{\boldsymbol{u}_i\}_{i=1}^n \subset \mathbb{R}^n$. We follow a systematic plan for calculating $\{\boldsymbol{l}_1, \boldsymbol{u}_1\}, \{\boldsymbol{l}_2, \boldsymbol{u}_2\}, \ldots, \{\boldsymbol{l}_n, \boldsymbol{u}_n\}$ in turn. The conditions required to avoid breakdown of this algorithm will give us *sufficient* conditions for an $\mathsf{LU}$ factorisation to exist.

We may write the $\mathsf{A} = \mathsf{LU}$ equation in terms of the above rows and columns, i.e.

$$\mathsf{A} = \mathsf{LU} = \begin{bmatrix} \boldsymbol{l}_1 & \boldsymbol{l}_2 & \cdots & \boldsymbol{l}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1^{\mathrm{T}} \\ \boldsymbol{u}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{u}_n^{\mathrm{T}} \end{bmatrix} \tag{5.8a}$$

$$= \begin{bmatrix} * & & & \\ * & * & & \\ & & \ddots & \\ * & * & & * \\ * & * & & \end{bmatrix} \times \begin{bmatrix} * & * & * & * \\ & * & * & * \\ & & \ddots & \\ & & & * \end{bmatrix} \begin{matrix} \boldsymbol{u}_1^{\mathrm{T}} \\ \boldsymbol{u}_2^{\mathrm{T}} \\ \\ \boldsymbol{u}_n^{\mathrm{T}} \end{matrix} \quad = \sum_{k=1}^{n} \boldsymbol{l}_k \boldsymbol{u}_k^{\mathrm{T}} \, . \tag{5.8b}$$

$\boldsymbol{l}_1 \quad \boldsymbol{l}_2 \qquad \boldsymbol{l}_n$

Since the first $(k-1)$ elements of $\boldsymbol{l}_k$ and $\boldsymbol{u}_k$ are all zero for $k \geqslant 2$, each rank-one matrix $\boldsymbol{l}_k \boldsymbol{u}_k^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ has zeros in its first $(k-1)$ rows and columns. Pictorially this gives

$$\mathsf{A} = \underbrace{\begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}}_{\boldsymbol{l}_1 \boldsymbol{u}_1^{\mathrm{T}}} + \underbrace{\begin{bmatrix} & & & \\ & * & * & * \\ & * & * & * \\ & * & * & * \end{bmatrix}}_{\boldsymbol{l}_2 \boldsymbol{u}_2^{\mathrm{T}}} + \underbrace{\begin{bmatrix} & & & \\ & & * & * \\ & & * & * \\ & & & \end{bmatrix}}_{\boldsymbol{l}_3 \boldsymbol{u}_3^{\mathrm{T}}} + \cdots + \underbrace{\begin{bmatrix} & & & \\ & & & \\ & & & * \end{bmatrix}}_{\boldsymbol{l}_n \boldsymbol{u}_n^{\mathrm{T}}} . \tag{5.8c}$$

We see from (5.8c) that only the product $\boldsymbol{l}_1 \boldsymbol{u}_1^{\mathrm{T}}$ contributes to the first row and column of $\mathsf{A}$: the first row of this product being $L_{1,1}\boldsymbol{u}_1^{\mathrm{T}} = \boldsymbol{u}_1^{\mathrm{T}}$ [9] and the first column being $U_{1,1}\boldsymbol{l}_1$. [10] Hence, introducing the notation $\mathsf{A}^{(0)} \equiv \mathsf{A}$, we have

$$\boldsymbol{u}_1^{\mathrm{T}} = [\text{the first row of } \mathsf{A}^{(0)}], \qquad \boldsymbol{l}_1 = [\text{the first column of } \mathsf{A}^{(0)}]/A_{1,1}^{(0)}.$$

[9] Similarly, it follows that the $k^{\text{th}}$ row of $\boldsymbol{l}_k \boldsymbol{u}_k^{\mathrm{T}}$ is $L_{k,k}\boldsymbol{u}_k^{\mathrm{T}} = \boldsymbol{u}_k^{\mathrm{T}}$.
[10] Similarly, the $k^{\text{th}}$ column of $\boldsymbol{l}_k \boldsymbol{u}_k^{\mathrm{T}}$ is $U_{k,k}\boldsymbol{l}_k$.

After having obtained $\boldsymbol{l}_1$ and $\boldsymbol{u}_1$, we construct the matrix

$$\mathsf{A}^{(1)} \equiv \mathsf{A}^{(0)} - \boldsymbol{l}_1 \boldsymbol{u}_1^{\mathrm{T}} = \sum_{k=2}^{n} \boldsymbol{l}_k \boldsymbol{u}_k^{\mathrm{T}} \in \mathbb{R}^{n\times n}\,. \tag{5.9}$$

Thus from (5.8b) we see that, not only are the first row and column of $\mathsf{A}^{(1)}$ both zero, but also only the product $\boldsymbol{l}_2\boldsymbol{u}_2^{\mathrm{T}}$ contributes to the second row and column of $\mathsf{A}^{(1)}$. Since the second row of this product is $\boldsymbol{u}_2^{\mathrm{T}}$ and its second column is $U_{2,2}\boldsymbol{l}_2$, we obtain

$$\boldsymbol{u}_2^{\mathrm{T}} = [\text{the second row of } \mathsf{A}^{(1)}], \qquad \boldsymbol{l}_2 = [\text{the second column of } \mathsf{A}^{(1)}]/A^{(1)}_{2,2}.$$

Continuing this way we can formulate the entire algorithm: starting with $\mathsf{A}^{(0)} \equiv \mathsf{A}$ and performing the following calculations for $k = 1, \ldots, n$.

$$U_{k,j} = A^{(k-1)}_{k,j}, \qquad j = k, \ldots, n, \tag{5.10a}$$

$$L_{i,k} = \frac{A^{(k-1)}_{i,k}}{A^{(k-1)}_{k,k}}, \qquad i = k, \ldots, n, \tag{5.10b}$$

$$A^{(k)}_{i,j} = A^{(k-1)}_{i,j} - L_{i,k}U_{k,j}, \qquad i,j > k. \tag{5.10c}$$

(Note that, since $\mathsf{A}^{(n)}$ is zero, (5.10c) is not needed for $k = n$.)

*Remarks.*

(i) The above construction shows that the condition

$$A^{(k-1)}_{k,k} \neq 0\,, \quad k = 1, \ldots, n-1\,, \tag{5.11}$$

is a *sufficient* condition for an $\mathsf{LU}$ factorization to exist and be unique. $A^{(0)}_{1,1}$ is just $A_{1,1}$, but the other values are only derived during construction. We shall see in §5.3.1 how to obtain equivalent conditions in terms of the original matrix $\mathsf{A}$.

(ii) For the $k$-th step of the above algorithm, we see from (5.10b) and (5.10c) that $(n-k)$ divisions are required to determine the components of $\boldsymbol{l}_k$ and $(n-k)^2$ multiplications to construct $\mathsf{A}^{(k)}$. Hence the total number of operations required for a successful $\mathsf{LU}$ factorisation is

$$N_{LU} = \sum_{k=1}^{n-1} \left[(n-k)^2 + (n-k)\right] = \tfrac{1}{3}\,n^3 + \mathcal{O}(n^2)\,. \tag{5.12a}$$

Remember, from (5.3b) and (5.5b), that

$$N_F = N_B = \sum_{k=1}^{n} k \sim \tfrac{1}{2}\,n^2 \tag{5.12b}$$

operations are required to solve triangular systems of equations, i.e. with coefficient matrix $\mathsf{L}$ or $\mathsf{U}$.

(iii) Construction of an $\mathsf{A} = \mathsf{LU}$ factorisation is one of the best ways of obtaining $\det \mathsf{A}$: i.e. we use the formula

$$\det \mathsf{A} = \det \mathsf{L} \det \mathsf{U} = \left(\prod_{k=1}^{n} L_{k,k}\right)\left(\prod_{k=1}^{n} U_{k,k}\right) = \left(\prod_{k=1}^{n} U_{k,k}\right)\,. \tag{5.13}$$

(iv) We have seen in (5.4) and (5.6) that the columns of $\mathsf{L}^{-1}$ and $\mathsf{U}^{-1}$ may be obtained by forward and backward substitution respectively. If we have a factorisation $\mathsf{A} = \mathsf{LU}$, then this means $\mathsf{A}^{-1} = \mathsf{U}^{-1}\mathsf{L}^{-1}$. More directly (and efficiently), we can calculate each column of $\mathsf{A}^{-1}$ in turn by solving

$$\mathsf{A}\boldsymbol{x}_j = \boldsymbol{e}_j \qquad j = 1, \ldots, n\,. \tag{5.14}$$

Of course we use an $\mathsf{A}=\mathsf{LU}$ factorisation to solve (5.14), i.e.

$$\mathsf{L}\boldsymbol{y}_j=\boldsymbol{e}_j\,,\quad \mathsf{U}\boldsymbol{x}_j=\boldsymbol{y}_j\qquad j=1,\ldots,n\,.$$

Thus the construction of $\mathsf{A}^{-1}$ requires a single $\mathsf{A}=\mathsf{LU}$ factorisation followed by $n$ forward substitutions and $n$ backward substitutions.

*Example.*

$$\mathsf{A} = \begin{bmatrix} 2 & 3 & -5 \\ 4 & 8 & -3 \\ -6 & 1 & 4 \end{bmatrix} \to \boldsymbol{l}_1=\begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix},\quad \boldsymbol{u}_1^{\mathrm{T}}=[\,2 \quad 3 \quad -5\,],\quad \boldsymbol{l}_1\boldsymbol{u}_1^{\mathrm{T}}=\begin{bmatrix} 2 & 3 & -5 \\ 4 & 6 & -10 \\ -6 & -9 & 15 \end{bmatrix},$$

$$\mathsf{A}^{(1)} = \begin{bmatrix} & & \\ & 2 & 7 \\ & 10 & -11 \end{bmatrix} \to \boldsymbol{l}_2=\begin{bmatrix} 0 \\ 1 \\ 5 \end{bmatrix},\quad \boldsymbol{u}_2^{\mathrm{T}}=[\,0 \quad 2 \quad 7\,],\quad \boldsymbol{l}_2\boldsymbol{u}_2^{\mathrm{T}}=\begin{bmatrix} & & \\ & 2 & 7 \\ & 10 & 35 \end{bmatrix},$$

$$\mathsf{A}^{(2)} = \begin{bmatrix} & & \\ & & \\ & & -46 \end{bmatrix} \to \boldsymbol{l}_3=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},\quad \boldsymbol{u}_3^{\mathrm{T}}=[\,0 \quad 0 \quad -46\,],\quad \boldsymbol{l}_3\boldsymbol{u}_3^{\mathrm{T}}=\mathsf{A}^{(2)},$$

so that

$$\mathsf{A}=\begin{bmatrix} 2 & 3 & -5 \\ 4 & 8 & -3 \\ -6 & 1 & 4 \end{bmatrix}=\mathsf{LU},\qquad \mathsf{L}=\begin{bmatrix} 1 & & \\ 2 & 1 & \\ -3 & 5 & 1 \end{bmatrix},\qquad \mathsf{U}=\begin{bmatrix} 2 & 3 & -5 \\ & 2 & 7 \\ & & -46 \end{bmatrix}.$$

*Remark.* All elements in the first $k$ rows and columns of $\mathsf{A}^{(k)}$ are zero. Hence, we can use the storage of the original $\mathsf{A}$ to accumulate $\mathsf{L}$ and $\mathsf{U}$, and to store the elements of the matrices $\mathsf{A}^{(k)}$. Thus for our example:

$$\begin{bmatrix} 2 & 3 & -5 \\ 4 & 8 & -3 \\ -6 & 1 & 4 \end{bmatrix} \to \left[\begin{array}{ccc} 2 & 3 & -5 \\ \hline 2 & 2 & 7 \\ -3 & 10 & -11 \end{array}\right] \to \left[\begin{array}{ccc} 2 & 3 & -5 \\ \hline 2 & \multicolumn{1}{c|}{2} & 7 \\ -3 & 5 & -46 \end{array}\right].$$

*Algorithm.* From (5.10a), (5.10b) and (5.10c), the following pseudo-code computes the $\mathsf{LU}$ factorization by overwriting $\mathsf{A}$:

```
for k = 1:n-1
  for i = k+1:n                          % Calculate only non-unit elements of L
    A(i,k) = A(i,k)/A(k,k);              % L(i,k) = A(i,k)/A(k,k)
    for j = k+1:n                        % U(k,j) = A(k,j), so do nothing
      A(i,j) = A(i,j) - A(i,k)*A(k,j);   % A(i,j) = A(i,j) - L(i,k)*U(k,j)
    end
  end
end
```

### 5.2.2 Relation to Gaussian elimination

At the $k^{\text{th}}$ step of the $\mathsf{LU}$-algorithm, the operation $\mathsf{A}^{(k)}=\mathsf{A}^{(k-1)}-\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ has the property that the $i^{\text{th}}$ row of $\mathsf{A}^{(k)}$ is the $i^{\text{th}}$ row of $\mathsf{A}^{(k-1)}$ minus $L_{i,k}$ times $\boldsymbol{u}_k^{\mathrm{T}}$ (the $k^{\text{th}}$ row of $\mathsf{A}^{(k-1)}$), i.e.

$$[\text{the } i^{\text{th}} \text{ row of } \mathsf{A}^{(k)}] \;=\; [\text{the } i^{\text{th}} \text{ row of } \mathsf{A}^{(k-1)}] \;-\; L_{i,k}\times\, [\text{the } k^{\text{th}} \text{ row of } \mathsf{A}^{(k-1)}],$$

where the multipliers $L_{i,k}=A_{i,k}^{(k-1)}/A_{k,k}^{(k-1)}$ are chosen so that, at the outcome, the $k^{\text{th}}$ column of $\mathsf{A}^{(k)}$ is zero. This construction is analogous to Gaussian elimination for solving $\mathsf{A}\boldsymbol{x}=\boldsymbol{b}$.

*Example.*

$$\begin{bmatrix} 2 & 3 & -5 \\ 4 & 8 & -3 \\ -6 & 1 & 4 \end{bmatrix} \overset{\substack{2^{\text{nd}}-1^{\text{st}}\cdot\ 2 \\ 3^{\text{rd}}-1^{\text{st}}\cdot\ (-3)}}{\to} \left[\begin{array}{ccc} 2 & 3 & -5 \\ \hline 0 & 2 & 7 \\ 0 & 10 & -11 \end{array}\right] \overset{3^{\text{rd}}-2^{\text{nd}}\cdot\,5}{\to} \left[\begin{array}{ccc} 2 & 3 & -5 \\ \hline 0 & \multicolumn{1}{c|}{2} & 7 \\ 0 & 0 & -46 \end{array}\right].$$

If at each step we put the multipliers $L_{i,k}$ into the spare sub-diagonal part of $\mathsf{A}$, then we obtain exactly the same form of the $\mathsf{LU}$ factorization as above.

$$\begin{bmatrix} 2 & 3 & -5 \\ 4 & 8 & -3 \\ -6 & 1 & 4 \end{bmatrix} \overset{\substack{2^{\text{nd}}-1^{\text{st}}\cdot\ 2 \\ 3^{\text{rd}}-1^{\text{st}}\cdot\ (-3)}}{\rightarrow} \begin{bmatrix} 2 & 3 & -5 \\ \hline 2 & 2 & 7 \\ -3 & 10 & -11 \end{bmatrix} \overset{3^{\text{rd}}-2^{\text{nd}}\cdot\ 5}{\rightarrow} \begin{bmatrix} 2 & 3 & -5 \\ \hline 2 & 2 & 7 \\ -3 & 5 & -46 \end{bmatrix}.$$

*Remark.* An important difference between the $\mathsf{LU}$ approach and Gaussian elimination is that in $\mathsf{LU}$ we do not consider the right hand side $\boldsymbol{b}$ until the factorization is complete. This is useful, for example, when there are many right hand sides; in particular if not all the $\boldsymbol{b}$'s are known at the outset (e.g. as in *multi-grid methods*). In Gaussian elimination the solution for each new $\boldsymbol{b}$ would require $\mathcal{O}(n^3)$ computational operations: in contrast, with the $\mathsf{LU}$ algorithm, $\mathcal{O}(n^3)$ operations are only required for the single initial factorisation; but then the solution for each new $\boldsymbol{b}$ only requires $\mathcal{O}(n^2)$ operations (for the back- and forward substitutions).

### 5.2.3 Pivoting to avoid breakdown

We have seen in §5.2.1 that not all $\mathsf{A} \in \mathbb{R}^{n\times n}$ have an $\mathsf{LU}$ factorisation (even if $\mathsf{A}$ is non-singular), so in order to solve $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ efficiently we need to generalise our approach.

**Definition 5.3** (*Permutation matrix).* $\mathsf{P} \in \mathbb{R}^{n\times n}$ is a *permutation* matrix if its rows are a re-arrangement (a permutation!) of the rows of the identity matrix $\mathsf{I} \in \mathbb{R}^{n\times n}$.

(Hence $\boldsymbol{z} = \mathsf{P}\boldsymbol{y}$ just means that the elements of $\boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^n$ are related through this permutation. Similarly, $\mathsf{C} = \mathsf{PB}$ means that the rows of $\mathsf{B}, \mathsf{C} \in \mathbb{R}^{n\times n}$ are related through this permutation.) Our aim in the present section is to show that, for any $\mathsf{A} \in \mathbb{R}^{n\times n}$, there exists a permutation matrix $\mathsf{P} \in \mathbb{R}^{n\times n}$ such that $\mathsf{PA}$ has an $\mathsf{LU}$ factorisation; i.e. $\mathsf{PA} = \mathsf{LU}$. It is then easy to solve $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ because

$$\mathsf{A}\boldsymbol{x} = \boldsymbol{b} \quad \Rightarrow \quad \mathsf{PA}\boldsymbol{x} = \mathsf{P}\boldsymbol{b} \quad \Rightarrow \quad \begin{cases} \mathsf{L}\boldsymbol{y} = \mathsf{P}\boldsymbol{b} \\ \mathsf{U}\boldsymbol{x} = \boldsymbol{y} \end{cases}.$$

Of course there can be no uniqueness here: in general there will be many suitable permutation matrices.

We now work through the $\mathsf{A} = \mathsf{LU}$ algorithm in (5.10) again; showing how breakdown can be avoided by introducing appropriate permutation matrices.

1. For $k = 1$, we only have breakdown if $\mathsf{A}^{(0)}_{1,1} \equiv \mathsf{A}_{1,1} = 0$. In this case, we choose $p > 1$ and make use of the permutation matrix

$$\mathsf{P}_1 \equiv \mathsf{I} - (\boldsymbol{e}_1 - \boldsymbol{e}_p)(\boldsymbol{e}_1 - \boldsymbol{e}_p)^{\mathrm{T}} \in \mathbb{R}^{n\times n}.$$

(Thus $\mathsf{P}_1\boldsymbol{z}$ just interchanges components 1 and $p$ of $\boldsymbol{z} \in \mathbb{R}^n$, while leaving the other components unchanged: i.e. $\mathsf{P}_1$ performs a very simple permutation!) $p > 1$ is chosen so that step (5.10b) of the algorithm does not breakdown when applied to $\mathsf{P}_1\mathsf{A}^{(0)}$.

- $\mathsf{P}_1\mathsf{A}^{(0)}$ has a non-zero element in the $(1,1)$ position.
- Construct $\boldsymbol{l}_1 \in \mathbb{R}^n$ from $\mathsf{P}_1\mathsf{A}^{(0)}$ as in (5.10a).
- Construct $\boldsymbol{u}_1 \in \mathbb{R}^n$ from $\mathsf{P}_1\mathsf{A}^{(0)}$ as in (5.10b).
- Set

$$\mathsf{A}^{(1)} \equiv \mathsf{P}_1\mathsf{A}^{(0)} - \boldsymbol{l}_1\boldsymbol{u}_1^{\mathrm{T}} \tag{5.15}$$

as in (5.10c).

The only time we cannot find a suitable $p > 1$ is when the *whole* first column of $\mathsf{A}^{(0)} \equiv \mathsf{A}$ is zero. But in this case we can just choose $\mathsf{P}_1 \equiv \mathsf{I}$, $\boldsymbol{l}_1 \equiv \boldsymbol{e}_1$ and $\boldsymbol{u}_1^{\mathrm{T}} \equiv$ [the first row of $\mathsf{A}^{(0)}$]; before constructing $\mathsf{A}^{(1)}$ as in (5.15). (This is not the only way of ensuring that $\mathsf{A}^{(1)}$ has zero first row and column, but is certainly the simplest.) Note that in this case, both $\mathsf{A}$ and $\mathsf{U}$ are singular.

Of course, if $\mathsf{A}^{(0)}_{1,1} \neq 0$ then we do not have breakdown and we can just set $p = 1$, i.e. $\mathsf{P}_1 \equiv \mathsf{I}$, and proceed as in (5.10)

2. For $1 < k \leqslant n-1$, we only have breakdown if $\mathsf{A}^{(k-1)}_{k,k} = 0$, i.e.

$$\mathsf{A}^{(k-1)} = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & 0 & A^{(k-1)}_{k,k+1} & \cdots & A^{(k-1)}_{k,n} \\ \vdots & \ddots & 0 & A^{(k-1)}_{k+1,k} & A^{(k-1)}_{k+1,k+1} & \cdots & A^{(k-1)}_{k+1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & A^{(k-1)}_{n,k} & A^{(k-1)}_{n,k+1} & \cdots & A^{(k-1)}_{n,n} \end{pmatrix} . \tag{5.16}$$

In this case, we choose $p > k$ and make use of the permutation matrix

$$\mathsf{P}_k \equiv \mathsf{I} - (\boldsymbol{e}_k - \boldsymbol{e}_p)(\boldsymbol{e}_k - \boldsymbol{e}_p)^{\mathrm{T}} \in \mathbb{R}^{n\times n} ,$$

i.e.

$$\mathsf{P}_k = \begin{pmatrix} 1 & & & & & & & & 0 \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & 0 & & & 1 & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & 1 & & & 0 & & \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ 0 & & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow \text{ row } k \\ \\ \\ \\ \leftarrow \text{ row } p \\ \\ \\ \end{matrix} . \tag{5.17}$$

(Thus $\mathsf{P}_k\boldsymbol{z}$ just interchanges components $k$ and $p$ of $\boldsymbol{z} \in \mathbb{R}^n$, while leaving the other components unchanged.) $p > k$ is chosen so that step (5.10b) of the algorithm does not breakdown when applied to $\mathsf{P}_1\mathsf{A}^{(k-1)}$.

- $\mathsf{P}_k\mathsf{A}^{(k-1)}$ has a non-zero element in the $(k,k)$ position.
- Construct $\boldsymbol{l}_k \in \mathbb{R}^n$ from $\mathsf{P}_k\mathsf{A}^{(k-1)}$ as in (5.10a).
- Construct $\boldsymbol{u}_k \in \mathbb{R}^n$ from $\mathsf{P}_k\mathsf{A}^{(k-1)}$ as in (5.10b).
- Set
$$\mathsf{A}^{(k)} \equiv \mathsf{P}_k\mathsf{A}^{(k-1)} - \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}} \tag{5.18}$$
as in (5.10c).

The only time we cannot find a suitable $p > k$ is when the *whole* $k^{\text{th}}$ column of $\mathsf{A}^{(k-1)}$ is zero. But in this case we can just choose $\mathsf{P}_k \equiv \mathsf{I}$, $\boldsymbol{l}_k \equiv \boldsymbol{e}_k$ and $\boldsymbol{u}_k^{\mathrm{T}} \equiv$ [the $k^{\text{th}}$ row of $\mathsf{A}^{(k-1)}$]; before constructing $\mathsf{A}^{(k)}$ as in (5.18). (Again, this is not the only way of ensuring that $\mathsf{A}^{(k)}$ has zero $k^{\text{th}}$ row and column, but is certainly the simplest.) Note that in this case, both $\mathsf{A}^{(k)}$ and $\mathsf{U}$ are singular. (This can only happen when our original matrix $\mathsf{A}$ is singular!)

Of course, if $\mathsf{A}^{(k-1)}_{k,k} \neq 0$ then we do not have breakdown and we can just set $p = k$, i.e. $\mathsf{P}_k \equiv \mathsf{I}$, and proceed as in (5.10)

Having avoided any breakdown in our algorithm, by introducing possible interchanges $\mathsf{P}_k$ at each step $k = 1, \ldots, n-1$, we need to examine carefully what kind of $\mathsf{LU}$ factorisation has been constructed. Our fundamental recurrence formulae are

$$\mathsf{A}^{(k)} \equiv \mathsf{P}_k\mathsf{A}^{(k-1)} - \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}} \qquad k = 1, \ldots, n ; \tag{5.19}$$

where $\mathsf{A}^{(0)} \equiv \mathsf{A}$ and $\mathsf{A}^{(n)}$ is the zero matrix. (Also $\mathsf{P}_n \equiv \mathsf{I}$, because there is no possible interchange at step $k = n$.) By combining (5.19), we see that

$$\begin{aligned} \mathsf{A}^{(1)} &= \mathsf{P}_1\mathsf{A} - \boldsymbol{l}_1\boldsymbol{u}_1^{\mathrm{T}} \\ \mathsf{A}^{(2)} &= \mathsf{P}_2\mathsf{P}_1\mathsf{A} - (\mathsf{P}_2\boldsymbol{l}_1)\,\boldsymbol{u}_1^{\mathrm{T}} - \boldsymbol{l}_2\boldsymbol{u}_2^{\mathrm{T}} \\ \mathsf{A}^{(3)} &= \mathsf{P}_3\mathsf{P}_2\mathsf{P}_1\mathsf{A} - (\mathsf{P}_3\mathsf{P}_2\boldsymbol{l}_1)\,\boldsymbol{u}_1^{\mathrm{T}} - (\mathsf{P}_3\boldsymbol{l}_2)\,\boldsymbol{u}_2^{\mathrm{T}} - \boldsymbol{l}_3\boldsymbol{u}_3^{\mathrm{T}} \end{aligned}$$

and eventually

$$\mathsf{P}\mathsf{A} = \mathsf{L}\mathsf{U}\,, \tag{5.20}$$

where $\mathsf{P} \in \mathbb{R}^{n\times n}$ is the permutation matrix defined by

$$\mathsf{P} \equiv \mathsf{P}_{n-1}\ldots\mathsf{P}_2\mathsf{P}_1\,.$$

Also our triangular matrices in (5.20) are defined by

$$\mathsf{L} = \begin{bmatrix} \tilde{\boldsymbol{l}}_1 & \tilde{\boldsymbol{l}}_2 & \ldots & \tilde{\boldsymbol{l}}_n \end{bmatrix}, \qquad \text{and} \qquad \mathsf{U} = \begin{bmatrix} \boldsymbol{u}_1^{\mathrm{T}} \\ \boldsymbol{u}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{u}_n^{\mathrm{T}} \end{bmatrix}, \tag{5.21}$$

where

$$\tilde{\boldsymbol{l}}_n \equiv \boldsymbol{l}_n\,, \quad \tilde{\boldsymbol{l}}_{n-1} \equiv \boldsymbol{l}_{n-1} \qquad \text{and} \qquad \tilde{\boldsymbol{l}}_k \equiv \mathsf{P}_{n-1}\ldots\mathsf{P}_{k+1}\boldsymbol{l}_k \quad \text{for} \quad k = n-2,\ldots,1\,.$$

So we see that, although the interchanges do not explicitly appear in the rows of $\mathsf{U}$, they do affect the columns of $\mathsf{L}$. In practice, as described by the examples in §5.2.4 and §5.2.5, at each step $k$ the possible interchange matrix $\mathsf{P}_k$ is applied to the portion of $\mathsf{L}$ that has already been constructed, i.e. the first $k-1$ columns $\boldsymbol{l}_1,\ldots,\boldsymbol{l}_{k-1}$.

Note that we also need to record the product of the interchanges in the permutation matrix $\mathsf{P}$ of (5.20), so that $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ can be solved or $\det\mathsf{A}$ calculated.

### 5.2.4 Pivoting to maintain accuracy

In the previous section, we have shown how every $\mathsf{A} \in \mathbb{R}^{n\times n}$ (even singular matrices!) have an $\mathsf{LU}$ factorisation, *provided* that the rows of $\mathsf{A}$ are also allowed to be permuted: this is the essence of the key formula (5.20). For practical computation with inexact arithmetic however, this is not good enough. It is not sufficient to use pivoting just to avoid algorithm breakdown, it should also be used so that the accuracy of our results is as high as possible.

At each step $1 \leqslant k \leqslant n-1$, the key decision about any interchange is made by examining

$$\mathsf{A}^{(k-1)} \equiv \begin{pmatrix} 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & A_{k,k}^{(k-1)} & A_{k,k+1}^{(k-1)} & \cdots & A_{k,n}^{(k-1)} \\ \vdots & \ddots & 0 & A_{k+1,k}^{(k-1)} & A_{k+1,k+1}^{(k-1)} & \cdots & A_{k+1,n}^{(k-1)} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & A_{n,k}^{(k-1)} & A_{n,k+1}^{(k-1)} & \cdots & A_{n,n}^{(k-1)} \end{pmatrix}. \tag{5.22}$$

Instead of just interchanging rows of $\mathsf{A}^{(k-1)}$ when $A_{k,k}^{(k-1)} = 0$, we identify $k \leqslant p \leqslant n$ such that

$$\left|A_{p,k}^{(k-1)}\right| = \max_{k\leqslant i\leqslant n}\left|A_{i,k}^{(k-1)}\right| \tag{5.23}$$

and use this $p$ to define our interchange at step $k$. Hence we swap rows $p$ and $k$ of $\mathsf{A}^{(k-1)}$ using $\mathsf{P}_k$ as defined in (5.17) and then use the same algorithm as in the previous section: i.e. we construct $\boldsymbol{l}_k$ and $\boldsymbol{u}_k$ from $\mathsf{P}_k\mathsf{A}^{(k-1)}$ and then form $\mathsf{A}^{(k)}$ as in (5.18).

*Example.*

$$\begin{bmatrix} 2 & 1 & 1 \\ \mathbf{4} & 1 & 0 \\ -2 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \to \begin{bmatrix} 4 & 1 & 0 \\ 2 & 1 & 1 \\ -2 & 2 & 1 \end{bmatrix} \to \begin{bmatrix} 4 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \\ -\frac{1}{2} & \mathbf{\frac{5}{2}} & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} \to \begin{bmatrix} 4 & 1 & 0 \\ -\frac{1}{2} & \frac{5}{2} & 1 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \to \begin{bmatrix} 4 & 1 & 0 \\ -\frac{1}{2} & \frac{5}{2} & 1 \\ \frac{1}{2} & \frac{1}{5} & \frac{4}{5} \end{bmatrix},$$

$$\mathsf{PA} = \mathsf{LU} \quad \Rightarrow \quad \mathsf{A} = \mathsf{P}^{\mathrm{T}}\mathsf{LU}, \quad \mathsf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathsf{L} = \begin{bmatrix} 1 & & \\ -\frac{1}{2} & 1 & \\ \frac{1}{2} & \frac{1}{5} & 1 \end{bmatrix}, \quad \mathsf{U} = \begin{bmatrix} 4 & 1 & 0 \\ & \frac{5}{2} & 1 \\ & & \frac{4}{5} \end{bmatrix},$$

where $\mathsf{P}$ is the permutation matrix that sends *row 2 to row 1*, row 3 to row 2, and **row 1 to row 3**.

*Partial pivoting.* Interchanging according to (5.23) is known as *partial pivoting* and has the important advantage of forcing

$$|L_{i,j}| \leqslant 1 \qquad i,j = 1,\ldots,n\,.$$

It also enables theoretical bounds for

$$|U_{i,j}| \quad i,j = 1,\ldots,n \qquad \text{and} \qquad |A^{(k)}_{i,j}| \quad i,j = 1,\ldots,n\,; k = 1,\ldots,n-1$$

to be established and these control the errors introduced by inexact arithmetic. There is also another form of pivoting that replaces (5.23) with

$$\left|A^{(k-1)}_{p,q}\right| = \max_{\substack{k\leqslant i\leqslant n \\ k\leqslant j\leqslant n}} \left|A^{(k-1)}_{i,j}\right| \tag{5.24}$$

and uses both row interchanges ($k \leftrightarrow p$) and column interchanges ($k \leftrightarrow q$) to bring this largest-in-modulus component to the $(k,k)$ position. (Post-multiplication by a permutation matrix permutes columns!) This is known as *total pivoting* and has stronger theoretical properties than partial pivoting. In practice however, the extra computational effort required for total pivoting is not regarded as worthwhile and partial pivoting remains the standard choice.

### 5.2.5 Further examples (*Unlectured*)

LU *factorization.*

$$\begin{bmatrix} -3 & 2 & 3 & -1 \\ 6 & -2 & -6 & 0 \\ -9 & 4 & 10 & 3 \\ 12 & -4 & -13 & -5 \end{bmatrix} \begin{matrix} 2^{\text{nd}}-1^{\text{st}}\cdot(-2) \\ 3^{\text{rd}}-1^{\text{st}}\cdot\ 3 \\ 4^{\text{th}}-1^{\text{st}}\cdot(-4) \\ \to \end{matrix} \begin{bmatrix} -3 & 2 & 3 & -1 \\ -2 & 2 & 0 & -2 \\ 3 & -2 & 1 & 6 \\ -4 & 4 & -1 & -9 \end{bmatrix}$$

$$\begin{matrix} 3^{\text{rd}}-2^{\text{nd}}\cdot(-1) \\ 4^{\text{th}}-2^{\text{nd}}\cdot\ 2 \\ \to \end{matrix} \begin{bmatrix} -3 & 2 & 3 & -1 \\ -2 & 2 & 0 & -2 \\ 3 & -1 & 1 & 4 \\ -4 & 2 & -1 & -5 \end{bmatrix}$$

$$\begin{matrix} 4^{\text{th}}-3^{\text{rd}}\cdot(-1) \\ \to \end{matrix} \begin{bmatrix} -3 & 2 & 3 & -1 \\ -2 & 2 & 0 & -2 \\ 3 & -1 & 1 & 4 \\ -4 & 2 & -1 & -1 \end{bmatrix}$$

i.e.

$$\mathsf{A} = \mathsf{LU}, \quad \mathsf{L} = \begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ 3 & -1 & 1 & \\ -4 & 2 & -1 & 1 \end{bmatrix}, \quad \mathsf{U} = \begin{bmatrix} -3 & 2 & 3 & -1 \\ & 2 & 0 & -2 \\ & & 1 & 4 \\ & & & -1 \end{bmatrix}$$

*Forward and back substitution.* The solution to the system

$$\mathsf{A}\boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{b} = [3, -2, 2, 0]^{\mathrm{T}}$$

proceeds in two steps

$$\mathsf{A}\boldsymbol{x} = \mathsf{L}\underbrace{\mathsf{U}\boldsymbol{x}}_{\boldsymbol{y}} = \boldsymbol{b} \quad \Rightarrow \quad 1) \quad \mathsf{L}\boldsymbol{y} = \boldsymbol{b}, \quad 2) \quad \mathsf{U}\boldsymbol{x} = \boldsymbol{y}.$$

1) Forward substitution

$$\begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ 3 & -1 & 1 & \\ -4 & 2 & -1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ 2 \\ 0 \end{bmatrix} \begin{matrix} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{matrix} \quad \Rightarrow \quad \boldsymbol{y} = \begin{bmatrix} 3 \\ 4 \\ -3 \\ 1 \end{bmatrix},$$

2) Back substitution

$$\begin{bmatrix} -3 & 2 & 3 & -1 \\ & 2 & 0 & -2 \\ & & 1 & 4 \\ & & & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ -3 \\ 1 \end{bmatrix} \begin{matrix} \uparrow \\ \uparrow \\ \uparrow \\ \uparrow \end{matrix} \quad \Rightarrow \quad \boldsymbol{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix},$$

LU *factorization with pivoting.*

$$\begin{bmatrix} -3 & 2 & 3 & -1 \\ 6 & -2 & -6 & 0 \\ -9 & 4 & 10 & 3 \\ \mathbf{12} & -4 & -13 & -5 \end{bmatrix} \overset{\begin{bmatrix}4\\2\\3\\1\end{bmatrix}}{\rightarrow} \begin{bmatrix} 12 & -4 & -13 & -5 \\ 6 & -2 & -6 & 0 \\ -9 & 4 & 10 & 3 \\ -3 & 2 & 3 & -1 \end{bmatrix} \rightarrow \left[\begin{array}{cccc} 12 & -4 & -13 & -5 \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{5}{2} \\ -\frac{3}{4} & \mathbf{1} & \frac{1}{4} & -\frac{3}{4} \\ -\frac{1}{4} & 1 & -\frac{1}{4} & -\frac{9}{4} \end{array}\right]$$

$$\overset{\begin{bmatrix}4\\3\\2\\1\end{bmatrix}}{\rightarrow} \left[\begin{array}{cccc} 12 & -4 & -13 & -5 \\ -\frac{3}{4} & 1 & \frac{1}{4} & -\frac{3}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{5}{2} \\ -\frac{1}{4} & 1 & -\frac{1}{4} & -\frac{9}{4} \end{array}\right] \rightarrow \left[\begin{array}{cccc} 12 & -4 & -13 & -5 \\ -\frac{3}{4} & 1 & \frac{1}{4} & -\frac{3}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{5}{2} \\ -\frac{1}{4} & 1 & -\frac{1}{2} & -\frac{3}{2} \end{array}\right] \rightarrow \left[\begin{array}{cccc} 12 & -4 & -13 & -5 \\ -\frac{3}{4} & 1 & \frac{1}{4} & -\frac{3}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{5}{2} \\ -\frac{1}{4} & 1 & -1 & 1 \end{array}\right]$$

i.e.

$$\mathsf{A} = \mathsf{P}^{\mathrm{T}}\mathsf{L}\mathsf{U}, \quad \mathsf{P} = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & 1 & & \\ 1 & & & \end{bmatrix}, \quad \mathsf{L} = \begin{bmatrix} 1 & & & \\ -\frac{3}{4} & 1 & & \\ \frac{1}{2} & 0 & 1 & \\ -\frac{1}{4} & 1 & 1 & 1 \end{bmatrix}, \quad \mathsf{U} = \begin{bmatrix} 12 & -4 & -13 & -5 \\ & 1 & \frac{1}{4} & -\frac{3}{4} \\ & & \frac{1}{2} & \frac{5}{2} \\ & & & 1 \end{bmatrix}$$

## 5.3 LU factorization theory and application to structured A

In this section we first derive some existence and uniqueness results for $\mathsf{A} = \mathsf{L}\mathsf{U}$ factorisation and then apply these to matrices $\mathsf{A}$ with special form. We are looking at particular conditions that allow us, either theoretically or practically, to avoid pivoting.

### 5.3.1 Existence and uniqueness of the LU factorization

We first relate the breakdown conditions (5.11) to explicit properties of the coefficient matrix $\mathsf{A}$.

**Definition 5.4** (*Leading principal submatrices).* The leading principal submatrices $\mathsf{A}_k \in \mathbb{R}^{k\times k}$ (for $k = 1, \ldots, n$) of $\mathsf{A} \in \mathbb{R}^{n\times n}$ are defined by

$$(A_k)_{i,j} = A_{i,j} \qquad i, j = 1, \ldots, k\,. \tag{5.25}$$

**Theorem 5.5.** *A sufficient condition for both the existence and uniqueness of an* $\mathsf{A} = \mathsf{LU}$ *factorization is that*

$$\det(\mathsf{A}_k) \neq 0 \qquad k = 1, \ldots, n-1\,.$$

*Proof. (Unlectured.)* We use induction on $n$, the size of the matrix. For $n = 1$, the result is clear.

Assume the result is true for $(n-1) \times (n-1)$ matrices and partition $\mathsf{A} \in \mathbb{R}^{n\times n}$ as

$$\mathsf{A} = \left[\begin{array}{c|c} \mathsf{A}_{n-1} & \boldsymbol{b} \\ \hline \boldsymbol{c}^{\mathrm{T}} & A_{n,n} \end{array}\right]. \tag{5.26a}$$

We require

$$\mathsf{A} = \mathsf{LU} \quad \text{with} \quad \mathsf{L} = \left[\begin{array}{c|c} \mathsf{L}_{n-1} & \boldsymbol{0} \\ \hline \boldsymbol{x}^{\mathrm{T}} & 1 \end{array}\right], \quad \mathsf{U} = \left[\begin{array}{c|c} \mathsf{U}_{n-1} & \boldsymbol{y} \\ \hline \boldsymbol{0}^{\mathrm{T}} & U_{n,n} \end{array}\right], \tag{5.26b}$$

where $\mathsf{L}_{n-1}, \mathsf{U}_{n-1} \in \mathbb{R}^{(n-1)\times(n-1)}$, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{n-1}$ and $U_{n,n}$ are to be determined. Multiplying out these block matrices, we see that we want to have

$$\mathsf{A} = \left[\begin{array}{c|c} \mathsf{A}_{n-1} & \boldsymbol{b} \\ \hline \boldsymbol{c}^{\mathrm{T}} & A_{n,n} \end{array}\right] = \left[\begin{array}{c|c} \mathsf{L}_{n-1}\mathsf{U}_{n-1} & \mathsf{L}_{n-1}\boldsymbol{y} \\ \hline \boldsymbol{x}^{\mathrm{T}}\mathsf{U}_{n-1} & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} + U_{n,n} \end{array}\right]. \tag{5.26c}$$

By virtue of the induction assumption $\mathsf{L}_{n-1}$ and $\mathsf{U}_{n-1}$ exist and are unique; further, $\mathsf{U}_{n-1}$ is non-singular since it is assumed that $\mathsf{A}_{n-1}$ is non-singular. Hence $\mathsf{L}_{n-1}\boldsymbol{y} = \boldsymbol{b}$ and $\boldsymbol{x}^{\mathrm{T}}\mathsf{U}_{n-1} = \boldsymbol{c}^{\mathrm{T}}$ can be uniquely solved to obtain

$$\boldsymbol{y} = \mathsf{L}_{n-1}^{-1}\boldsymbol{b}, \quad \boldsymbol{x}^{\mathrm{T}} = \boldsymbol{c}^{\mathrm{T}}\mathsf{U}_{n-1}^{-1}, \quad U_{n,n} = A_{n,n} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}. \tag{5.26d}$$

So, by construction, there exists a unique factorization $\mathsf{A} = \mathsf{LU}$. □

**Corollary 5.6.** *Our breakdown conditions in* (5.11) *are related to the leading principal submatrices of* $\mathsf{A}$ *by*

$$A_{1,1}^{(0)} = \det(\mathsf{A}_1) \qquad \textit{and} \qquad A_{k,k}^{(k-1)} = \frac{\det(\mathsf{A}_k)}{\det(\mathsf{A}_{k-1})} \quad k = 2, \ldots, n\,. \tag{5.27}$$

*Proof.* From (5.26b) and (5.26c)

$$\begin{aligned} \det(\mathsf{A}_n) = \det(\mathsf{A}) = \det(\mathsf{L})\det(\mathsf{U}) = \det(\mathsf{U}) = U_{n,n}\det(\mathsf{U}_{n-1})\,, \\ \det(\mathsf{A}_{n-1}) = \det(\mathsf{L}_{n-1})\det(\mathsf{U}_{n-1}) = \det(\mathsf{U}_{n-1})\,. \end{aligned}$$

So $\det(\mathsf{A}_n) = U_{n,n}\det(\mathsf{A}_{n-1})$, and thence by induction

$$U_{k,k} = \frac{\det(\mathsf{A}_k)}{\det(\mathsf{A}_{k-1})} \qquad k = 2, \ldots, n\,. \tag{5.28}$$

Further $U_{1,1} = A_{1,1} = \det(\mathsf{A}_1)$ and, from (5.10a) with $j = k$, we have that $U_{k,k} = A_{k,k}^{(k-1)}$ for $k = 1, \ldots, n$. □

Note that $\mathsf{A}$ being non-singular ($\det(\mathsf{A}_n) \neq 0$!) is *not* required in Theorem 5.5 and, since $\mathsf{U} = \mathsf{L}^{-1}\mathsf{A}$, $\mathsf{A}$ and $\mathsf{U}$ are either both singular or both non-singular. If we do include this condition, then a useful stronger result is obtained.

**Theorem 5.7.** *If* $\det(\mathsf{A}_k) \neq 0$ *for* $k = 1, \ldots, n$ *then* $\mathsf{A} \in \mathbb{R}^{n\times n}$ *has a unique factorization of the form*

$$\mathsf{A} = \mathsf{LDU}\,; \tag{5.29}$$

*where* $\mathsf{L} \in \mathbb{R}^{n\times n}$ *is unit lower triangular,* $\mathsf{U} \in \mathbb{R}^{n\times n}$ *is unit upper triangular and* $\mathsf{D} \in \mathbb{R}^{n\times n}$ *is a non-singular diagonal matrix.*

*Proof.* From Theorem 5.5 we have $\mathsf{A} = \mathsf{L}\widehat{\mathsf{U}}$, where $\widehat{\mathsf{U}}$ is non-singular and upper triangular; so we just write $\widehat{\mathsf{U}} = \mathsf{D}\mathsf{U}$, where $\mathsf{U}$ is unit upper triangular. □

In pictorial form, this last result is

$$\mathsf{A} = \begin{bmatrix} \boldsymbol{l}_1 & \boldsymbol{l}_2 & \cdots \boldsymbol{l}_n \end{bmatrix} \begin{bmatrix} D_{1,1} & 0 & \cdots & 0 \\ 0 & D_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & D_{n,n} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1^{\mathrm{T}} \\ \boldsymbol{u}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{u}_n^{\mathrm{T}} \end{bmatrix} = \sum_{k=1}^{n} D_{k,k} \boldsymbol{l}_k \boldsymbol{u}_k^{\mathrm{T}} , \tag{5.30}$$

where $\boldsymbol{l}_k$ is the $k^{\text{th}}$ column of $\mathsf{L}$ and $\boldsymbol{u}_k^{\mathrm{T}}$ is the $k^{\text{th}}$ row of $\mathsf{U}$.

We end this section with some results of mainly theoretical interest.

**Theorem 5.8.** *If* $\mathsf{A}$ *is non-singular and an* $\mathsf{LU}$ *factorization of* $\mathsf{A}$ *exists then*

- $\mathsf{A}_k$ *is non-singular for* $k = 1, \ldots, n-1$,
- *this* $\mathsf{LU}$ *factorization is unique.*

*Proof.* The first part follows from (5.28), i.e.

$$\det(\mathsf{A}) = \prod_{i=1}^{n} U_{i,i} \neq 0 \qquad \Rightarrow \qquad \det(\mathsf{A}_k) = \prod_{i=1}^{k} U_{i,i} \neq 0 \,.$$

The second part then follows from Theorem 5.5. □

It is also possible to prove the second part of the last theorem directly.

**Theorem 5.9** (Uniqueness)**.** *If* $\mathsf{A}$ *is non-singular, it is impossible for more than one* $\mathsf{LU}$ *factorization to exist, i.e.*

$$\mathsf{A} = \mathsf{L}_1\mathsf{U}_1 = \mathsf{L}_2\mathsf{U}_2 \qquad \textit{implies} \qquad \textit{both} \quad \mathsf{L}_1 = \mathsf{L}_2 \quad \textit{and} \quad \mathsf{U}_1 = \mathsf{U}_2.$$

*Proof.* If $\mathsf{A}$ is non-singular, then both $\mathsf{U}_1$ and $\mathsf{U}_2$ are non-singular, and hence the equality $\mathsf{L}_1\mathsf{U}_1 = \mathsf{L}_2\mathsf{U}_2$ implies $\mathsf{L}_2^{-1}\mathsf{L}_1 = \mathsf{U}_2\mathsf{U}_1^{-1} = \mathsf{V}$. The product of lower/upper triangular matrices is lower/upper triangular and, as already noted in (5.4) and (5.6), the inverse of a lower/upper triangular matrix is lower/upper triangular. Consequently, $\mathsf{V}$ is simultaneously lower and upper triangular, hence it is diagonal. Since $\mathsf{L}_2^{-1}\mathsf{L}_1$ has unit diagonal, we obtain $\mathsf{V} = \mathsf{I}$. □

Finally, we just state the converse of Theorem 5.5.

**Theorem 5.10** (Unproved)**.** *If at least one of* $\{\mathsf{A}_k\}_{k=1}^{n-1}$ *is singular, then* either *no* $\mathsf{LU}$ *factorization exists* or *the* $\mathsf{LU}$ *factorization is not unique.*

*Examples.*

(i) There is no $\mathsf{LU}$ factorization for

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

or, indeed, for any non-singular $n \times n$ matrix $\mathsf{A}$ such that $\det(\mathsf{A}_k) = 0$ for some $k = 1, \ldots, n-1$.

(ii) *Some* singular matrices may be $\mathsf{LU}$ factorized in many ways, e.g.

$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & \frac{1}{2} \end{bmatrix}.$$

### 5.3.2 Symmetric matrices

If $\mathsf{A} \in \mathbb{R}^{n\times n}$ is symmetric, then one can try to construct different kinds of "symmetric" $\mathsf{LU}$ factorisation.

**Theorem 5.11.** *If $\mathsf{A} \in \mathbb{R}^{n\times n}$ is symmetric and non-singular, with*

$$\det(\mathsf{A}_k) \neq 0 \qquad k = 1, \ldots, n-1\,;$$

*then a unique factorization of the form*

$$\mathsf{A} = \mathsf{L}\mathsf{D}\mathsf{L}^{\mathrm{T}}. \tag{5.31}$$

*exists, with $\mathsf{L} \in \mathbb{R}^{n\times n}$ unit lower triangular and $\mathsf{D} \in \mathbb{R}^{n\times n}$ a non-singular diagonal matrix.*

*Proof.* From Theorem 5.7 we have

$$\mathsf{L}\mathsf{D}\mathsf{U} = \mathsf{A} = \mathsf{A}^{\mathrm{T}} = \mathsf{U}^{\mathrm{T}}\mathsf{D}\mathsf{L}^{\mathrm{T}}\ :$$

hence, by uniqueness, $\mathsf{U} = \mathsf{L}^{\mathrm{T}}$. □

*Remark.* Clearly this form of $\mathsf{LU}$ factorisation is a suitable exploitation of symmetry and requires roughly half the storage of conventional $\mathsf{LU}$. Specifically, to compute this factorization, we let $\mathsf{A}^{(0)} = \mathsf{A}$ and for $k = 1, 2, \ldots, n$ let $\boldsymbol{l}_k$ be the multiple of the $k^{\text{th}}$ column of $\mathsf{A}^{(k-1)}$ such that $L_{k,k} = 1$. We then set

$$D_{k,k} = A^{(k-1)}_{k,k} \quad \text{and form} \quad \mathsf{A}^{(k)} = \mathsf{A}^{(k-1)} - D_{k,k}\boldsymbol{l}_k\boldsymbol{l}_k^{\mathrm{T}}. \tag{5.32}$$

*Example.* Let $\mathsf{A} = \mathsf{A}^{(0)} = \begin{bmatrix} 2 & 4 \\ 4 & 11 \end{bmatrix}$. Hence $\boldsymbol{l}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $D_{1,1} = 2$ and

$$\mathsf{A}^{(1)} = \mathsf{A}^{(0)} - D_{1,1}\boldsymbol{l}_1\boldsymbol{l}_1^{\mathrm{T}} = \begin{bmatrix} 2 & 4 \\ 4 & 11 \end{bmatrix} - 2\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 3 \end{bmatrix}.$$

We deduce that $\boldsymbol{l}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $D_{2,2} = 3$ and

$$\mathsf{A} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}.$$

*Remark.* Even with symmetric matrices, some form of pivoting is generally necessary, both to avoid breakdown and to maintain accuracy when using inexact arithmetic. Clearly, permuting the rows of $\mathsf{A}$ will destroy symmetry unless we simultaneously permute the corresponding columns: i.e.

$$\mathsf{A} \quad \to \quad \mathsf{P}\mathsf{A}\mathsf{P}^{\mathrm{T}},$$

where $\mathsf{P} \in \mathbb{R}^{n\times n}$ is a permutation matrix. One would *like* to prove that, for any symmetric $\mathsf{A} \in \mathbb{R}^{n\times n}$, a symmetric factorisation of the form

$$\mathsf{P}\mathsf{A}\mathsf{P}^{\mathrm{T}} = \mathsf{L}\mathsf{D}\mathsf{L}^{\mathrm{T}},$$

where $\mathsf{L} \in \mathbb{R}^{n\times n}$ is unit lower triangular and $\mathsf{D} \in \mathbb{R}^{n\times n}$ is diagonal, exists. This however is not true, even if $\mathsf{A}$ is restricted to be nonsingular. Fortunately, the next best result *is* true: for any symmetric $\mathsf{A} \in \mathbb{R}^{n\times n}$, a symmetric factorisation of the form

$$\mathsf{P}\mathsf{A}\mathsf{P}^{\mathrm{T}} = \mathsf{L}\mathsf{T}\mathsf{L}^{\mathrm{T}},$$

where $\mathsf{L} \in \mathbb{R}^{n\times n}$ is unit lower triangular and $\mathsf{T} \in \mathbb{R}^{n\times n}$ is both symmetric and tridiagonal, exists. The permutation matrix $\mathsf{P}$ can be chosen so that the factorisation is safe to use with inexact arithmetic.

### 5.3.3 Positive definite matrices

This is another commonly occurring type of matrix.

**Definition 5.12** (*Positive definite matrices).* $\mathsf{A} \in \mathbb{R}^{n\times n}$ is positive definite if

$$\boldsymbol{x}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} > 0 \qquad \forall \boldsymbol{x} \neq \mathbf{0} \in \mathbb{R}^n\,.$$

It is easy to show that all positive definite matrices are non-singular.

**Theorem 5.13.** *If* $\mathsf{A} \in \mathbb{R}^{n\times n}$ *is positive definite then it is non-singular.*

*Proof.* If $\mathsf{A}$ is singular then $\exists \boldsymbol{z} \neq \mathbf{0} \in \mathbb{R}^n$ such that $\mathsf{A}\boldsymbol{z} = \mathbf{0}$: but then $\boldsymbol{z}^{\mathrm{T}}\mathsf{A}\boldsymbol{z} = 0$, which is impossible if $\mathsf{A}$ is positive definite. □

The same kind of argument shows that every positive definite matrix has a unique $\mathsf{LU}$ factorisation.

**Theorem 5.14.** *If* $\mathsf{A} \in \mathbb{R}^{n\times n}$ *is positive definite then*

$$\det(\mathsf{A}_k) \neq 0 \qquad k = 1, \ldots, n-1\,.$$

*Proof.* By Theorem 5.13, it is sufficient to show that $\mathsf{A}_k \in \mathbb{R}^{k\times k}$ is positive definite for $k = 1, \ldots, n-1$.

For any $\boldsymbol{y} \neq \mathbf{0} \in \mathbb{R}^k$, construct $\boldsymbol{x} \neq \mathbf{0} \in \mathbb{R}^n$ by adding $n-k$ zero components to $\boldsymbol{y}$. This means that

$$\boldsymbol{y}^{\mathrm{T}}\mathsf{A}_k\boldsymbol{y} = \boldsymbol{x}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} > 0$$

and so $\mathsf{A}_k \in \mathbb{R}^{k\times k}$ is positive definite for $k = 1, \ldots, n-1$. □

**Corollary 5.15.** *If* $\mathsf{A} \in \mathbb{R}^{n\times n}$ *is positive definite then Theorem 5.5 shows that it has a unique* $\mathsf{A} = \mathsf{LU}$ *factorization.*

*Remark.* In practice, it is *not* true that pivoting is unnecessary for positive definite coefficient matrices $\mathsf{A}$. In general, with inexact arithmetic, the accuracy of solutions for $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ (using $\mathsf{A} = \mathsf{LU}$ factorisation because $\mathsf{A}$ is positive definite) can be completely unsatisfactory.

### 5.3.4 Symmetric positive definite matrices

If $\mathsf{A} \in \mathbb{R}^{n\times n}$ is both symmetric and positive definite, pivoting is no longer required either theoretically or practically.

**Theorem 5.16.** *A symmetric* $\mathsf{A} \in \mathbb{R}^{n\times n}$ *is positive definite if and only if it has a factorization of the form*

$$\mathsf{A} = \mathsf{L}\mathsf{D}\mathsf{L}^{\mathrm{T}}, \tag{5.33}$$

*where* $\mathsf{L} \in \mathbb{R}^{n\times n}$ *is unit lower triangular and* $\mathsf{D} \in \mathbb{R}^{n\times n}$ *is diagonal with* $D_{k,k} > 0 \quad k = 1, \ldots, n$.

*Proof.* If such a factorisation exists then, for any $\boldsymbol{x} \in \mathbb{R}^n$, we have

$$\boldsymbol{x}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} = \boldsymbol{x}^{\mathrm{T}}\mathsf{L}\mathsf{D}\mathsf{L}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{y}^{\mathrm{T}}\mathsf{D}\boldsymbol{y} = \sum_{k=1}^{n} D_{k,k}y_k^2\,,$$

where $\boldsymbol{y} = \mathsf{L}^{\mathrm{T}}\boldsymbol{x}$. If $\boldsymbol{x} \neq \mathbf{0}$ then we must also have $\boldsymbol{y} \neq \mathbf{0}$ (since $\mathsf{L}$ is non-singular), so $\sum_{k=1}^{n} D_{k,k}y_k^2 > 0$ and hence also $\boldsymbol{x}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} > 0$. Thus $\mathsf{A}$ must be positive definite.

Conversely, if $\mathsf{A}$ is positive definite then we know from Theorem 5.11 (making use of Theorem 5.13 and Theorem 5.14) that it has a unique factorisation of the form

$$\mathsf{A} = \mathsf{L}\mathsf{D}\mathsf{L}^{\mathrm{T}}\,,$$

where $\mathsf{L} \in \mathbb{R}^{n\times n}$ is unit lower triangular and $\mathsf{D} \in \mathbb{R}^{n\times n}$ is both diagonal and non-singular. In addition, after defining $\{\boldsymbol{y}_k\}_{k=1}^n \subset \mathbb{R}^n$ by

$$\mathsf{L}^{\mathrm{T}}\boldsymbol{y}_k = \boldsymbol{e}_k \qquad k = 1, \ldots, n\,,$$

we see that $\boldsymbol{y}_k \neq \boldsymbol{0}$ for $k = 1, \ldots, n$ and that

$$D_{k,k} = \boldsymbol{e}_k^{\mathrm{T}}\mathsf{D}\boldsymbol{e}_k = \boldsymbol{y}_k^{\mathrm{T}}\mathsf{L}\mathsf{D}\mathsf{L}^{\mathrm{T}}\boldsymbol{y}_k = \boldsymbol{y}_k^{\mathrm{T}}\mathsf{A}\boldsymbol{y}_k > 0 \qquad k = 1, \ldots, n\,.$$

□

*Remark.* Hence it is possible to find out if a symmetric $\mathsf{A} \in \mathbb{R}^{n\times n}$ is positive definite by simply trying to construct an $\mathsf{A} = \mathsf{L}\mathsf{D}\mathsf{L}^{\mathrm{T}}$ factorization and then checking that $D_{k,k} > 0$ for $k = 1, \ldots, n$.

*Example.* The following $3 \times 3$ matrix is shown to be positive definite.

$$\begin{bmatrix} 2 & 6 & -2 \\ 6 & 21 & 0 \\ -2 & 0 & 16 \end{bmatrix} \to \left[\begin{array}{ccc} 2 & 6 & -2 \\ \hline 3 & 3 & 6 \\ -1 & 6 & 14 \end{array}\right] \to \left[\begin{array}{ccc} 2 & 6 & -2 \\ \hline 3 & 3 & 6 \\ -1 & 2 & 2 \end{array}\right] \Rightarrow L = \begin{bmatrix} 1 & & \\ 3 & 1 & \\ -1 & 2 & 1 \end{bmatrix},\ D = \begin{bmatrix} 2 & & \\ & 3 & \\ & & 2 \end{bmatrix}.$$

*Cholesky factorization.* This is a famous alternative (but equivalent!) factorisation for matrices $\mathsf{A} \in \mathbb{R}^{n\times n}$ which are both symmetric and positive definite. If $\mathsf{D} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with positive diagonal elements then we can define $\mathsf{D}^{1/2} \in \mathbb{R}^{n\times n}$ as the diagonal matrix whose diagonal elements are $(D_{k,k})^{1/2}$ for $k = 1, \ldots, n$. (The positive square root is used here!) Consequently $\mathsf{D}^{1/2}$ has positive diagonal elements. Then, using Theorem 5.16, we can write

$$\mathsf{A} = \mathsf{L}\mathsf{D}\mathsf{L}^{\mathrm{T}} = \mathsf{L}\mathsf{D}^{1/2}\mathsf{D}^{1/2}\mathsf{L}^{\mathrm{T}} = \mathsf{L}\mathsf{D}^{1/2}\left(\mathsf{L}\mathsf{D}^{1/2}\right)^{\mathrm{T}} = \mathsf{G}\mathsf{G}^{\mathrm{T}}\,, \tag{5.34}$$

where $\mathsf{G} \equiv \mathsf{L}\mathsf{D}^{1/2} \in \mathbb{R}^{n\times n}$ is a lower triangular matrix with positive diagonal elements. This is the unique *Cholesky factorization* for a symmetric, positive definite $\mathsf{A}$.

*Remark.* We emphasise that both (5.33) and (5.34) are completely safe even with inexact arithmetic. Pivoting is *not* necessary when $\mathsf{A}$ is both symmetric and positive definite.

### 5.3.5 Sparse matrices

In this section, we look at a few of the properties of the $\mathsf{A} = \mathsf{L}\mathsf{U}$ factorisation for a different type of matrix $\mathsf{A}$. (For simplicity, we shall always assume that $\mathsf{A}$ is non-singular and that the factorisation exists and is unique.)

**Definition 5.17.** A matrix $\mathsf{A} \in \mathbb{R}^{n\times n}$ is called a *sparse* matrix if nearly all the elements of $\mathsf{A}$ are zero.

The simplest examples of sparse matrices are band matrices.

**Definition 5.18.** A matrix $\mathsf{A} \in \mathbb{R}^{n\times n}$ is called a *band* matrix if there exists an integer $r \ll n$ such that

$$A_{i,j} = 0 \quad \text{for all} \quad |i - j| > r.$$

In other words, all the nonzero elements of $\mathsf{A}$ reside in a narrow band of width $2r + 1$ about the principal diagonal. ($r$ subdiagonals below the principal diagonal and $r$ superdiagonals above the principal diagonal.)

$$r = 1 \begin{bmatrix} * & * & & & \\ * & * & * & & \\ & * & * & * & \\ & & * & * & * \\ & & & * & * \end{bmatrix}, \quad r = 2 \begin{bmatrix} * & * & * & & \\ * & * & * & * & \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \end{bmatrix}, \quad r = 3 \begin{bmatrix} * & * & * & * & \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \end{bmatrix}.$$

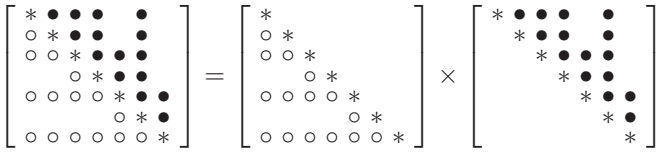
($r = 1$ is commonly called a tridiagonal matrix; similarly, $r = 2$ is a pentadiagonal matrix.)

It is often required to solve *very* large systems $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ ($n = 10^9$ is a relatively modest example) where $\mathsf{A}$ is sparse. The efficient solution of such systems should exploit the sparsity. In particular, we wish the matrices $\mathsf{L}$ and $\mathsf{U}$ to inherit as much as possible of the sparsity of $\mathsf{A}$, so that the cost of performing the forward and backward substitutions with $\mathsf{L}$ and $\mathsf{U}$ is comparable with the cost of forming the product $\mathsf{A}\boldsymbol{x}$: i.e. the cost of computation should be determined by the number of nonzero entries, rather than by $n$. To this end the following theorem is useful.

**Theorem 5.19.** *Let* $\mathsf{A} = \mathsf{LU}$ *be the LU factorization (without pivoting!) of a sparse matrix. Then*

(i) *all leading zeros in the rows of* $\mathsf{A}$ *to the left of the principal diagonal are inherited by* $\mathsf{L}$,

(ii) *all leading zeros in the columns of* $\mathsf{A}$ *above the principal diagonal are inherited by* $\mathsf{U}$.



*Here* $\circ$ *refers to non-zeros to the left of the principal diagonal in* $\mathsf{A}$ *and* $\mathsf{L}$*, while* $\bullet$ *refers to non-zeros above the principal diagonal in* $\mathsf{A}$ *and* $\mathsf{U}$.

*Proof.* From our conditions on $\mathsf{A}$ at the beginning of this section, we know that $U_{k,k} \neq 0$ for $k = 1, \ldots, n$. If $A_{i,1} = A_{i,2} = \cdots = 0$ are the leading zeros in the $i$-th row, then we obtain

$$\begin{aligned} 0 = A_{i,1} &= L_{i,1}U_{1,1} && \Rightarrow \quad L_{i,1} = 0, \\ 0 = A_{i,2} &= L_{i,1}U_{1,2} + L_{i,2}U_{2,2} && \Rightarrow \quad L_{i,2} = 0, \\ 0 = A_{i,3} &= L_{i,1}U_{1,3} + L_{i,2}U_{2,3} + L_{i,3}U_{3,3} && \Rightarrow \quad L_{i,3} = 0, \quad \text{and so on.} \end{aligned}$$

Similarly for the leading zeros in the $j$-th column. Since $L_{k,k} = 1$ for $k = 1, \ldots, n$, it follows that

$$\begin{aligned} 0 = A_{1,j} &= L_{1,1}U_{1,j} && \Rightarrow \quad U_{1,j} = 0, \\ 0 = A_{2,j} &= L_{2,1}U_{1,j} + L_{2,2}U_{2,j} && \Rightarrow \quad U_{2,j} = 0, \\ 0 = A_{3,j} &= L_{3,1}U_{1,j} + L_{3,2}U_{2,j} + L_{3,3}U_{3,j} && \Rightarrow \quad U_{3,j} = 0, \quad \text{and so on.} \end{aligned}$$

□

**Corollary 5.20.** *If* $\mathsf{A}$ *is a band matrix and* $\mathsf{A} = \mathsf{LU}$*, then* $L_{i,j} = U_{i,j} = 0$ *for all* $|i - j| > r$*. Hence the sparsity structure is inherited by the factorization and both* $\mathsf{L}$ *and* $\mathsf{U}$ *are band matrices with the same band width as* $\mathsf{A}$.

*Cost.* In general, the expense of calculating an $\mathsf{LU}$ factorization of an $n \times n$ *dense* matrix $\mathsf{A}$ is $\mathcal{O}(n^3)$ operations and the expense of solving $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$, provided that the factorization is known, is $\mathcal{O}(n^2)$. However, in the case of a banded $\mathsf{A}$, we need just

(i) $\mathcal{O}(r^2 n)$ operations to factorize, and

(ii) $\mathcal{O}(rn)$ operations to solve a linear system (after factorization).

**Method 5.21.** Theorem 5.19 suggests that, for the factorization of a sparse (but not band!) matrix $A$, one might try to reorder its rows and columns beforehand so that many of the zero elements become leading zeros in rows and columns. (Thus we are now using interchanges to reduce the fill-in for $\mathsf{L}$ and $\mathsf{U}$, rather than to prevent breakdown of the factorisation.)

*Example 1.* The $\mathsf{LU}$ factorization of

$$\mathsf{A} = \begin{bmatrix} 5 & 1 & 1 & 1 & 1 \\ 1 & 1 & & & \\ 1 & & 1 & & \\ 1 & & & 1 & \\ 1 & & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ \frac{1}{5} & 1 & & & \\ \frac{1}{5} & -\frac{1}{4} & 1 & & \\ \frac{1}{5} & -\frac{1}{4} & -\frac{1}{3} & 1 & \\ \frac{1}{5} & -\frac{1}{4} & -\frac{1}{3} & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 5 & 1 & 1 & 1 & 1 \\ & \frac{4}{5} & -\frac{1}{5} & -\frac{1}{5} & -\frac{1}{5} \\ & & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ & & & \frac{2}{3} & -\frac{1}{3} \\ & & & & \frac{1}{2} \end{bmatrix}$$

has significant fill-in. However, exchanging the first and the last rows and columns yields

$$\mathsf{PAP}^{\mathrm{T}} = \begin{bmatrix} 1 & & & & 1 \\ & 1 & & & 1 \\ & & 1 & & 1 \\ & & & 1 & 1 \\ 1 & 1 & 1 & 1 & 5 \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & 1 \\ & 1 & & & 1 \\ & & 1 & & 1 \\ & & & 1 & 1 \\ & & & & 1 \end{bmatrix}.$$

Thus we conclude that if the non-zeros of $\mathsf{A}$ occur only on the diagonal, in one row and in one column; then the full row and column should be placed at the bottom and on the right of $\mathsf{A}$, respectively.

*Example 2.* The $\mathsf{LU}$ factorisation of

$$\begin{bmatrix} -3 & 1 & 1 & 2 & 0 \\ 1 & -3 & 0 & 0 & 1 \\ 1 & 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 3 & 0 \\ 0 & 1 & 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 & 0 & 0 \\ -\frac{1}{3} & -\frac{1}{8} & 1 & 0 & 0 \\ -\frac{2}{3} & -\frac{1}{4} & \frac{6}{19} & 1 & 0 \\ 0 & -\frac{3}{8} & \frac{1}{19} & \frac{4}{81} & 1 \end{bmatrix} \begin{bmatrix} -3 & 1 & 1 & 2 & 0 \\ 0 & -\frac{8}{3} & \frac{1}{3} & \frac{2}{3} & 1 \\ 0 & 0 & \frac{19}{8} & \frac{3}{4} & \frac{1}{8} \\ 0 & 0 & 0 & \frac{81}{19} & \frac{4}{19} \\ 0 & 0 & 0 & 0 & \frac{272}{81} \end{bmatrix},$$

has significant fill-in. However, reordering (symmetrically) rows and columns $1 \leftrightarrow 3$, $2 \leftrightarrow 4$ and $4 \leftrightarrow 5$ yields

$$\begin{bmatrix} 2 & 0 & 1 & 0 & 0 \\ 0 & 3 & 2 & 0 & 0 \\ 1 & 2 & -3 & 0 & 1 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 1 & 1 & -3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{2}{3} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{6}{29} & \frac{1}{3} & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 1 & 0 & 0 \\ 0 & 3 & 2 & 0 & 0 \\ 0 & 0 & -\frac{29}{6} & 0 & 1 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & -\frac{272}{87} \end{bmatrix}.$$

*General sparse matrices.* These feature in a wide range of applications, e.g. the solution of partial differential equations, and there exists a wealth of methods for their solution. One approach is efficient factorization, that minimizes *fill in.* Yet another is to use iterative methods (see the Part II *Numerical Analysis* course). There also exists a substantial body of other, highly effective methods, e.g. Fast Fourier Transforms, preconditioned conjugate gradients and multi-grid techniques (again see the Part II *Numerical Analysis* course), fast multi-pole techniques and much more.

*Sparsity and graph theory.* An exceedingly powerful (and beautiful) methodology of ordering pivots to minimize fill-in of sparse matrices uses graph theory and, like many other cool applications of mathematics in numerical analysis, is alas not in the schedules `:-(`

# 6 Linear Least Squares and the QR factorisation

In this section we wish to "solve" the *over-determined rectangular* system

$$\mathsf{A}\boldsymbol{x} = \boldsymbol{b}\,, \tag{6.1}$$

where our given data is $\mathsf{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ with $m > n$. (Our approach can also deal with the case $m = n$, even when $\mathsf{A}$ is singular! When $\mathsf{A}$ is non-singular, however, the algorithms in §5 are more efficient.) Consequently we are looking for a solution $\boldsymbol{x} \in \mathbb{R}^n$; but because we have more equations then unknowns ($m > n$!), we need to make clear exactly what is meant by a solution of (6.1).

Throughout this section, we regard (6.1) as a *least squares* problem and define $\boldsymbol{x}^* \in \mathbb{R}^n$ as a solution of (6.1) if

$$\|\mathsf{A}\boldsymbol{x}^* - \boldsymbol{b}\|^2 = \min_{\boldsymbol{x}\in\mathbb{R}^n} \|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|^2\,,$$

where

$$\|\boldsymbol{z}\|^2 \equiv \langle \boldsymbol{z}, \boldsymbol{z}\rangle \equiv \sum_{i=1}^{m} z_i^2$$

is the Euclidean vector norm and associated scalar product on $\mathbb{R}^m$.

*Remark.* Problems of this form occur frequently when we collect $m$ observations $(x_i, y_i)$, which are typically prone to measurement error, and wish to fit them to an $n$-variable linear model, typically with $m \gg n$. In statistics, this is called *linear regression.*

For instance, suppose that we have $m$ measurements of $y \approx F(x)$, and that we wish to model $F$ with a linear combination of $\{\phi_j(x)\}_{j=1}^n$, i.e.

$$F(x) \equiv c_1\phi_1(x) + c_2\phi_2(x) + \cdots + c_n\phi_n(x) \qquad \text{and} \qquad F(x_i) \approx y_i \quad \text{for} \quad i = 1...m\,.$$

(Such a problem might occur if we were trying to match some planet observations to an ellipse.) Hence we want to determine $\boldsymbol{c} \in \mathbb{R}^n$ such that the $F(x_i)$ 'best' fit the $y_i$, i.e.

$$\mathsf{A}\boldsymbol{c} \equiv \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \vdots & & \vdots \\ \phi_1(x_n) & \cdots & \phi_n(x_n) \\ \vdots & & \vdots \\ \phi_1(x_m) & \cdots & \phi_n(x_m) \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} \equiv \begin{bmatrix} F(x_1) \\ \vdots \\ F(x_n) \\ \vdots \\ F(x_m) \end{bmatrix} \approx \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ \vdots \\ y_m \end{bmatrix} \equiv \boldsymbol{y}\,.$$

There are many ways of doing this; we will determine the $\boldsymbol{c} \in \mathbb{R}^n$ that minimizes the sum of squares of the deviation, i.e. we minimize

$$\sum_{i=1}^{m} \left(F(x_i) - y_i\right)^2 \equiv \|\mathsf{A}\boldsymbol{c} - \boldsymbol{y}\|^2.$$

This leads to a *linear* system of equations for the determination of the unknown $\boldsymbol{c} \in \mathbb{R}^n$.

## 6.1 The normal equations

We can immediately characterise the solutions of (6.1) in terms of a square system of equations.

**Theorem 6.1.** *$\boldsymbol{x}^* \in \mathbb{R}^n$ is a solution of (6.1) if and only if*

$$\mathsf{A}^{\mathrm{T}}\left(\mathsf{A}\boldsymbol{x}^* - \boldsymbol{b}\right) = \boldsymbol{0}\,. \tag{6.2}$$

*Proof.* If $\boldsymbol{x}^* \in \mathbb{R}^n$ is a solution of (6.1) then it must minimize

$$f(\boldsymbol{x}) \equiv \|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 = \langle \mathsf{A}\boldsymbol{x} - \boldsymbol{b}, \mathsf{A}\boldsymbol{x} - \boldsymbol{b} \rangle = \boldsymbol{x}^{\mathrm{T}}\mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} - 2\boldsymbol{x}^{\mathrm{T}}\mathsf{A}^{\mathrm{T}}\boldsymbol{b} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{b}\,.$$

But the gradient of $f$ is

$$\nabla f(\boldsymbol{x}) \equiv 2\mathsf{A}^{\mathrm{T}}\left(\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\right)$$

and we know that $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ is a necessary condition for $\boldsymbol{x}^* \in \mathbb{R}^n$ to be a minimum of $f$.

Conversely, if $\boldsymbol{x}^* \in \mathbb{R}^n$ satisfies (6.2) then we can write

$$\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 = \|\mathsf{A}(\boldsymbol{x}^* + \boldsymbol{y}) - \boldsymbol{b}\|^2 \qquad \forall \boldsymbol{x} \in \mathbb{R}^n\,, \tag{6.3}$$

where $\boldsymbol{y} \equiv \boldsymbol{x} - \boldsymbol{x}^*$. Consequently

$$\begin{aligned} \|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 &= \|\mathsf{A}\boldsymbol{y} + (\mathsf{A}\boldsymbol{x}^* - \boldsymbol{b})\|^2 \\ &= \langle \mathsf{A}\boldsymbol{y}, \mathsf{A}\boldsymbol{y} \rangle + 2\boldsymbol{y}^{\mathrm{T}}\mathsf{A}^{\mathrm{T}}\left(\mathsf{A}\boldsymbol{x}^* - \boldsymbol{b}\right) + \langle \mathsf{A}\boldsymbol{x}^* - \boldsymbol{b}, \mathsf{A}\boldsymbol{x}^* - \boldsymbol{b} \rangle \\ &= \|\mathsf{A}\boldsymbol{y}\|^2 + \|\mathsf{A}\boldsymbol{x}^* - \boldsymbol{b}\|^2 \\ &\geqslant \|\mathsf{A}\boldsymbol{x}^* - \boldsymbol{b}\|^2 \end{aligned}$$

shows that $\boldsymbol{x}^*$ is a solution of (6.1). □

Remembering that $\mathsf{A} \in \mathbb{R}^{m\times n}$ is said to have *full rank* if its columns form a linearly independent set of vectors in $\mathbb{R}^m$, we have the following useful uniqueness result.

**Corollary 6.2.** *If $\mathsf{A} \in \mathbb{R}^{m\times n}$ has full rank then* (6.1) *has a unique solution.*

*Proof.* If $\mathsf{A}$ has full rank then

$$\boldsymbol{y} \neq \boldsymbol{0} \in \mathbb{R}^n \Rightarrow \mathsf{A}\boldsymbol{y} \neq \boldsymbol{0} \in \mathbb{R}^m\,.$$

Hence the symmetric matrix $\mathsf{A}^{\mathrm{T}}\mathsf{A} \in \mathbb{R}^{n\times n}$ is positive definite since

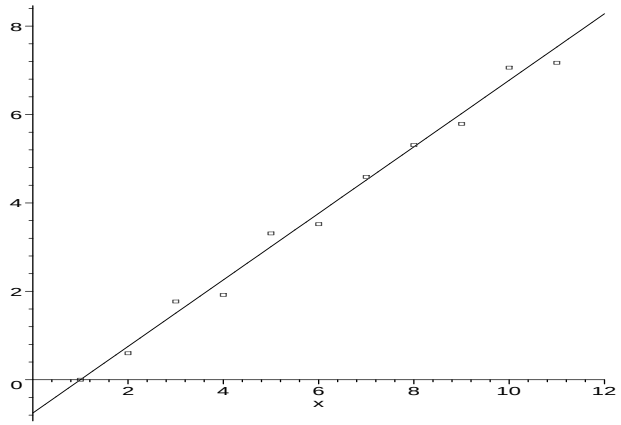
$$\boldsymbol{x}^{\mathrm{T}}\mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} = \|\mathsf{A}\boldsymbol{x}\|^2 \qquad \forall \boldsymbol{x} \in \mathbb{R}^n\,.$$

Thus Theorem 5.13 tells us that the square system $\mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} = \mathsf{A}^{\mathrm{T}}\boldsymbol{b}$ has a unique solution $\boldsymbol{x}^* \in \mathbb{R}^n$ and so, by Theorem 6.1, (6.1) has a unique solution. □

Together Theorem 6.1 and Corollary 6.2 show us that one way of solving the least squares problem (6.1) is to solve the square linear system

$$\mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} = \mathsf{A}^{\mathrm{T}}\boldsymbol{b} \qquad \mathsf{A}^{\mathrm{T}}\mathsf{A} \in \mathbb{R}^{n\times n}, \quad \mathsf{A}^{\mathrm{T}}\boldsymbol{b} \in \mathbb{R}^n\,. \tag{6.4}$$

These are called the *normal equations* and $\mathsf{A}^{\mathrm{T}}\mathsf{A}$ is called the *Gram* matrix.




| $x_i$ | $y_i$ |
|---|---|
| 1 | 0.00 |
| 2 | 0.60 |
| 3 | 1.77 |
| 4 | 1.92 |
| 5 | 3.31 |
| 6 | 3.52 |
| 7 | 4.59 |
| 8 | 5.31 |
| 9 | 5.79 |
| 10 | 7.06 |
| 11 | 7.17 |

**Figure 6.11:** Least squares straight line data fitting.

*Example 1.* The least squares approximation, by a straight line, to the data plotted in Figure 6.11

$$F(x) = c_1 + c_2 x \qquad (\phi_1(x) = 1, \quad \phi_2(x) = x),$$

results in the following normal equations and solution:

$$\mathsf{A} = \big(\phi_j(x_i)\big) = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ \vdots & \vdots \\ 1 & 11 \end{bmatrix}, \quad \underbrace{\begin{bmatrix} 11 & 66 \\ 66 & 506 \end{bmatrix}}_{\mathsf{A}^{\mathrm{T}}\mathsf{A}} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 41.04 \\ 328.05 \end{bmatrix}}_{\mathsf{A}^{\mathrm{T}}\boldsymbol{y}} \quad \Rightarrow \quad \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -0.7314 \\ 0.7437 \end{bmatrix}.$$

*Example 2.* Using the normal equations, find the least squares approximation to the data

| $x_i$ | $y_i$ |
|---|---|
| $-1$ | 2 |
| 0 | 1 |
| 1 | 0 |

by a function $F = c_1\phi_1 + c_2\phi_2$, where $\phi_1(x) = x$ and $\phi_2(x) = 1 + x - x^2$.

We solve the system of normal equations:

$$\mathsf{A} = \big(\phi_j(x_i)\big) = \begin{bmatrix} -1 & -1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \qquad \underbrace{\begin{bmatrix} 2 & 2 \\ 2 & 3 \end{bmatrix}}_{\mathsf{A}^{\mathrm{T}}\mathsf{A}} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \underbrace{\begin{bmatrix} -2 \\ -1 \end{bmatrix}}_{\mathsf{A}^{\mathrm{T}}\boldsymbol{y}} \quad \Rightarrow \quad \boldsymbol{c} = \begin{bmatrix} -2 \\ 1 \end{bmatrix}.$$

The residual is

$$\mathsf{A}\boldsymbol{c} - \boldsymbol{y} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ -1 \end{bmatrix} \quad \Rightarrow \quad \|\mathsf{A}\boldsymbol{c} - \boldsymbol{y}\| = \sqrt{2}\,.$$

We may also derive the normal equations directly by a geometrical argument. If we denote the columns of $\mathsf{A}$ by $\{\boldsymbol{a_j}\}_{j=1}^n \subset \mathbb{R}^m$ then the least squares problem can be written

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \Big\| \boldsymbol{b} - \sum_{j=1}^n x_j \boldsymbol{a}_j \Big\|.$$

The value of this minimum is the Euclidean distance between the given vector $\boldsymbol{b} \in \mathbb{R}^m$ and the subspace $\mathcal{C} \subset \mathbb{R}^m$ that is the column-space of $\mathsf{A}$. (This subspace has dimension $n$ if and only if $\mathsf{A}$ has full rank.) Consequently the minimum is attained by $\boldsymbol{x}^* \in \mathbb{R}^n$ when

$$\boldsymbol{b} - \sum_{j=1}^n x_j^* \boldsymbol{a}_j \equiv \boldsymbol{b} - \mathsf{A}\boldsymbol{x}^*$$

is orthogonal to all the vectors in $\mathcal{C}$, i.e.

$$\mathsf{A}^{\mathrm{T}} \left(\boldsymbol{b} - \mathsf{A}\boldsymbol{x}^*\right) = \boldsymbol{0}\,.$$

Continuing this geometrical argument, it is easy to show directly that the normal equations (6.4) are always consistent. We use the orthogonal decomposition

$$\mathbb{R}^m = \mathcal{C} \oplus \mathcal{C}^{\perp}\,,$$

where $\mathcal{C}^{\perp}$ denotes the subspace of $\mathbb{R}^m$ which is orthogonal to $\mathcal{C}$ with respect to the Euclidean scalar product, to uniquely write

$$\boldsymbol{b} = \boldsymbol{b}_1 + \boldsymbol{b}_2 \qquad \boldsymbol{b}_1 \in \mathcal{C}, \boldsymbol{b}_2 \in \mathcal{C}^{\perp}\,.$$

Since $\boldsymbol{x} \in \mathbb{R}^n \;\Rightarrow\; \mathsf{A}\boldsymbol{x} \in \mathcal{C}$, all solutions of our least squares problem (6.1) satisfy $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}_1$ and

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

(If $\mathsf{A}$ has full rank, there is a unique solution: otherwise there are an infinite number of solutions for $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}_1$.) Consequently

$$\mathsf{A}\boldsymbol{x} = \boldsymbol{b}_1 \Rightarrow \boldsymbol{b} - \mathsf{A}\boldsymbol{x} = \boldsymbol{b}_2 \Rightarrow \mathsf{A}^{\mathrm{T}}(\boldsymbol{b} - \mathsf{A}\boldsymbol{x}) = \boldsymbol{0}$$

shows that the normal equations are consistent.

In the full rank case, using the normal equations (6.4) to solve least squares problems is popular in many applications: i.e. one only has to construct the Cholesky decomposition (5.34) for the symmetric positive definite Gram matrix $\mathsf{A}^{\mathrm{T}}\mathsf{A}$. Unfortunately, there are often practical disadvantages in forming $\mathsf{A}^{\mathrm{T}}\mathsf{A}$.

(i) $\mathsf{A}$ may have useful sparsity properties, which are lost when forming $\mathsf{A}^{\mathrm{T}}\mathsf{A}$.

(ii) With inexact arithmetic, $\mathsf{A}$ can have full rank but $\mathsf{A}^{\mathrm{T}}\mathsf{A}$ may be singular. For instance, suppose that our computer works to the IEEE arithmetic standard ($\approx 15$ significant digits) and let

$$\mathsf{A} = \begin{bmatrix} 10^8 & -10^8 \\ 1 & 1 \end{bmatrix} \qquad \Longrightarrow \qquad \mathsf{A}^{\mathrm{T}}\mathsf{A} = \begin{bmatrix} 10^{16}+1 & -10^{16}+1 \\ -10^{16}+1 & 10^{16}+1 \end{bmatrix} \approx 10^{16} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Suppose $\boldsymbol{b} = [0, 2]^{\mathrm{T}}$, then the solution of $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ is $[1, 1]^{\mathrm{T}}$, as can be shown by Gaussian elimination; however, our computer 'believes' that $\mathsf{A}^{\mathrm{T}}\mathsf{A}$ is singular!

(iii) Even if things are not quite this bad, $\mathsf{A}^{\mathrm{T}}\mathsf{A}$ may be much closer to singularity than $\mathsf{A}$ is to being rank deficient. The "squaring" process is inherently dangerous and $\mathsf{A}^{\mathrm{T}}\mathsf{A}$ can be a much more ill-conditioned matrix than $\mathsf{A}$.

We shall see that there are often much better algorithms for solving least squares problems than the normal equations approach. These superior algorithms are based on the systematic use of orthogonal matrices and are described in the rest of this section.

## 6.2 Orthogonal matrices

Orthogonal matrices are the "nicest" type of non-singular matrix.

**Definition 6.3** (*Orthogonal matrix).* $\mathsf{Q} \in \mathbb{R}^{n\times n}$ is *orthogonal* if

$$\mathsf{Q}^{\mathrm{T}}\mathsf{Q} = \mathsf{I}\,.$$

Thus an orthogonal $\mathsf{Q}$ must be non-singular and its inverse is its transpose. (Hence $\mathsf{Q}^{\mathrm{T}}$ must also be an orthogonal matrix.) Equivalent definitions are:-

a) the columns of $\mathsf{Q}$ form an orthonormal set in $\mathbb{R}^n$;

b) the rows of $\mathsf{Q}$ form an orthonormal set in $\mathbb{R}^n$.

(Of course here we mean orthonormality with respect to the Euclidean scalar product.)

The key property of orthogonal matrices in connection with least squares problems is that they leave the Euclidean vector norm *invariant*.

**Proposition 6.4.** *If $\mathsf{Q} \in \mathbb{R}^{n\times n}$ is orthogonal then*

$$\|\mathsf{Q}\boldsymbol{x}\| = \|\boldsymbol{x}\| \quad \forall \boldsymbol{x} \in \mathbb{R}^n\,.$$

*Proof.*

$$\|\mathsf{Q}\boldsymbol{x}\|^2 = \boldsymbol{x}^{\mathrm{T}}\mathsf{Q}^{\mathrm{T}}\mathsf{Q}\boldsymbol{x} = \|\boldsymbol{x}\|^2\,.$$

□

A similar argument shows that the product of orthogonal matrices is orthogonal.

**Proposition 6.5.** *If* $\mathsf{P}, \mathsf{Q} \in \mathbb{R}^{n\times n}$ *are orthogonal then so is* $\mathsf{PQ}$.

*Proof.*

$$(\mathsf{PQ})^{\mathrm{T}}(\mathsf{PQ}) = \mathsf{Q}^{\mathrm{T}}\mathsf{P}^{\mathrm{T}}\mathsf{PQ} = \mathsf{I}\,.$$

□

It is also easy to see that the determinant of an orthogonal matrix must be $\pm 1$, i.e.

$$1 = \det(\mathsf{I}) = \det(\mathsf{Q}\mathsf{Q}^{\mathrm{T}}) = \det(\mathsf{Q})\det(\mathsf{Q}^{\mathrm{T}}) = (\det(\mathsf{Q}))^2\,.$$

Both of these are possible, e.g. $\mathsf{Q} = \pm\mathsf{I} \in \mathbb{R}^{3\times 3}$.

Finally, we use an orthogonality argument to show how an orthonormal set of vectors can be extended. (This result will be useful later to construct orthogonal matrices.)

**Lemma 6.6.** *If* $\{\boldsymbol{q}_j\}_{j=1}^n \subset \mathbb{R}^m$ *is orthonormal, with* $n < m$, *then* $\exists \boldsymbol{q}_{n+1} \in \mathbb{R}^m$ *such that* $\{\boldsymbol{q}_j\}_{j=1}^{n+1}$ *is orthonormal.*

*Proof.* We describe a method for constructing such a $\boldsymbol{q}_{n+1}$. Let $\mathsf{Q} \in \mathbb{R}^{m\times n}$ have columns $\{\boldsymbol{q}_j\}_{j=1}^n$: so that from

$$\sum_{k=1}^{m}\left(\sum_{j=1}^{n}\mathsf{Q}_{k,j}^2\right) \equiv \sum_{j=1}^{n}\|\boldsymbol{q}_j\|^2 = n < m\,,$$

it follows that $\exists i \in \{1,2,\ldots,m\}$ such that $\sum_{j=1}^n \mathsf{Q}_{i,j}^2 < 1$. If we define

$$\boldsymbol{w} \equiv \boldsymbol{e}_i - \sum_{j=1}^{n}\langle \boldsymbol{q}_j, \boldsymbol{e}_i\rangle \boldsymbol{q}_j$$

then (by construction) $\boldsymbol{w}$ is orthogonal to $\{\boldsymbol{q}_j\}_{j=1}^n$, i.e.

$$\langle \boldsymbol{q}_\ell, \boldsymbol{w}\rangle = \langle \boldsymbol{q}_\ell, \boldsymbol{e}_i\rangle - \sum_{j=1}^{n}\langle \boldsymbol{q}_j, \boldsymbol{e}_i\rangle\langle \boldsymbol{q}_\ell, \boldsymbol{q}_j\rangle = 0 \qquad \ell = 1,\ldots,n\,.$$

Furthermore, since $\mathsf{Q}_{i,j} = \langle \boldsymbol{q}_j, \boldsymbol{e}_i\rangle$, we have

$$\|\boldsymbol{w}\|^2 = \langle \boldsymbol{e}_i, \boldsymbol{e}_i\rangle - 2\sum_{j=1}^{n}\langle \boldsymbol{q}_j, \boldsymbol{e}_i\rangle\langle \boldsymbol{e}_i, \boldsymbol{q}_j\rangle + \sum_{j=1}^{n}\langle \boldsymbol{q}_j, \boldsymbol{e}_i\rangle\sum_{k=1}^{n}\langle \boldsymbol{q}_k, \boldsymbol{e}_i\rangle\langle \boldsymbol{q}_j, \boldsymbol{q}_k\rangle = 1 - \sum_{j=1}^{n}\mathsf{Q}_{i,j}^2 > 0\,.$$

Hence we can define $\boldsymbol{q}_{n+1} \equiv \boldsymbol{w}/\|\boldsymbol{w}\|$. □

## 6.3 The QR factorization

We immediately see the practical importance of orthogonal matrices with respect to the least squares problem (6.1), because Proposition 6.4 tells us that

$$\min_{\boldsymbol{x}\in\mathbb{R}^n}\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\| = \min_{\boldsymbol{x}\in\mathbb{R}^n}\|\mathsf{Q}\mathsf{A}\boldsymbol{x} - \mathsf{Q}\boldsymbol{b}\| \tag{6.5}$$

for all orthogonal $\mathsf{Q} \in \mathbb{R}^{m\times m}$. Hence we would like to find an orthogonal $\mathsf{Q} \in \mathbb{R}^{m\times m}$ so that the least squares problem on the right in (6.5) is as "simple" as possible and we know (by analogy with §5) that this means forcing $\mathsf{QA}$ to be some kind of triangular matrix.

**Definition 6.7.** $\mathsf{A} = \mathsf{QR}$ is a $\mathsf{QR}$ factorization of $\mathsf{A} \in \mathbb{R}^{m\times n}$ if $\mathsf{Q} \in \mathbb{R}^{m\times m}$ is orthogonal and $\mathsf{R} \in \mathbb{R}^{m\times n}$ is "upper triangular", i.e.

$$\underbrace{\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ A_{n,1} & \cdots & A_{n,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ A_{m,1} & \cdots & A_{m,n} \end{bmatrix}}_{n} = \underbrace{\begin{bmatrix} Q_{1,1} & \cdots & Q_{1,n} & \cdots & Q_{1,m} \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ Q_{n,1} & & Q_{n,n} & & Q_{n,m} \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ Q_{m,1} & \cdots & Q_{m,n} & \cdots & Q_{m,m} \end{bmatrix}}_{m>n} \underbrace{\left.\begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ & R_{2,2} & & \vdots \\ & & \ddots & \vdots \\ & & & R_{n,n} \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}\right\}}_{n} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix} m>n\,. \tag{6.6}$$

(Thus $R_{i,j} = 0$ for $i > j$.)

*Remarks.*

a) Every $\mathsf{A} \in \mathbb{R}^{m\times n}$ has a (non-unique) $\mathsf{QR}$ factorization and we shall prove this by construction. Note that non-uniqueness is immediately obvious by changing sign in both a column of $\mathsf{Q}$ and the corresponding row of $\mathsf{R}$.

b) If we denote the columns of $\mathsf{A}$ and $\mathsf{Q}$ by $\{\boldsymbol{a}_j\}_{j=1}^n$ and $\{\boldsymbol{q}_j\}_{j=1}^m$ respectively, then (6.6) becomes

$$\begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_m \end{bmatrix} \begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ 0 & R_{2,2} & & \vdots \\ \vdots & \ddots & \ddots & \\ & & 0 & R_{n,n} \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix},$$

and

$$\boldsymbol{a}_j = \sum_{i=1}^{j} R_{i,j}\boldsymbol{q}_i, \qquad j = 1, 2, \ldots, n\,. \tag{6.7}$$

In other words, the $j^{\text{th}}$ column of $\mathsf{A}$ is expressed as a linear combination of the first $j$ columns of $\mathsf{Q}$. (Remember that the columns of $\mathsf{Q}$ form an orthonormal set in $\mathbb{R}^m$.)

c) We see from (6.6) that the last $m - n$ rows of $\mathsf{R}$ and last $m - n$ columns of $\mathsf{Q}$ are redundant, so that we have $\mathsf{A} = \widetilde{\mathsf{Q}}\widetilde{\mathsf{R}}$ where $\widetilde{\mathsf{Q}} \in \mathbb{R}^{m\times n}$ and $\widetilde{\mathsf{R}} \in \mathbb{R}^{n\times n}$. This is called a *skinny* $\mathsf{QR}$ factorisation and written

$$m>n \left\{ \underbrace{\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ A_{n,1} & \cdots & A_{n,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ A_{m,1} & \cdots & A_{m,n} \end{bmatrix}}_{n} = \underbrace{\begin{bmatrix} Q_{1,1} & \cdots & Q_{1,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ Q_{n,1} & \cdots & Q_{n,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ Q_{m,1} & \cdots & Q_{m,n} \end{bmatrix}}_{n<m} \underbrace{\left.\begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ & R_{2,2} & & \vdots \\ & & \ddots & \vdots \\ & & & R_{n,n} \end{bmatrix}\right\}}_{n} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix} n\,. \tag{6.8}$$

d) Having constructed a $\mathsf{QR}$ factorisation of $\mathsf{A}$, the least squares problem (6.1) becomes

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\| = \min_{\boldsymbol{x}\in\mathbb{R}^n} \|\mathsf{R}\boldsymbol{x} - \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\|\,. \tag{6.9}$$

In §6.5 we shall see how to solve least squares problems with an upper triangular coefficient matrix. In this section we shall also see that a skinny $\mathsf{QR}$ factorisation is good enough to solve a least squares problem.

## 6.4 Constructing a QR factorization

We shall describe three standard algorithms for constructing an $\mathsf{A} = \mathsf{QR}$ factorization.

(i) The Gram–Schmidt process (as in *Vectors & Matrices*) is used to orthogonalise the columns of $\mathsf{A} \in \mathbb{R}^{m\times n}$.

(ii) Simple rotation matrices $\Omega \in \mathbb{R}^{m\times m}$ (called *Givens*, *Jacobi* or plane rotations) are used to gradually transform $\mathsf{A} \in \mathbb{R}^{m\times n}$ element-by-element into "upper triangular" form, i.e.

$$\Omega_k \dots \Omega_2 \Omega_1 \mathsf{A} = \mathsf{R} \qquad \text{where} \qquad k \leqslant mn - n(n+1)/2\,.$$

Each rotation matrix is orthogonal and therefore Proposition 6.5 tells us that

$$\mathsf{Q} \equiv \Omega_1^{\mathrm{T}} \Omega_2^{\mathrm{T}} \cdots \Omega_k^{\mathrm{T}} \quad \Rightarrow \quad \mathsf{A} = \mathsf{QR}\,.$$

(iii) Simple reflection matrices $\mathsf{H} \in \mathbb{R}^{m\times m}$ (called *Householder* reflections) are used to gradually transform $\mathsf{A} \in \mathbb{R}^{m\times n}$ column-by-column into "upper triangular" form, i.e.

$$\mathsf{H}_k \dots \mathsf{H}_2 \mathsf{H}_1 \mathsf{A} = \mathsf{R} \qquad \text{where} \qquad k \leqslant n\,.$$

Each reflection matrix is symmetric and orthogonal and therefore Proposition 6.5 again tells us that

$$\mathsf{Q} \equiv \mathsf{H}_1 \mathsf{H}_2 \cdots \mathsf{H}_k \quad \Rightarrow \quad \mathsf{A} = \mathsf{QR}\,.$$

### 6.4.1 The Gram–Schmidt orthogonalisation process

First recall, from *Vectors & Matrices*, that given a finite, linearly independent set of vectors $\{\boldsymbol{w}_1, \dots, \boldsymbol{w}_r\}$ the Gram–Schmidt process is a method of generating an orthogonal set $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_r\}$ which spans the same subspace. This is done, at stage $k$, by projecting $\boldsymbol{w}_k$ orthogonally onto the subspace generated by $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_{k-1}\}$, and then the vector $\boldsymbol{v}_k$ is defined to be the difference between $\boldsymbol{w}_k$ and this projection.

We will use the Gram–Schmidt process to construct a skinny $\mathsf{QR}$ factorisation (6.8) of $\mathsf{A} \in \mathbb{R}^{m\times n}$. (For simplicity, we will omit the tildes from $\widetilde{\mathsf{Q}}$ and $\widetilde{\mathsf{R}}$!) Thus, given the $n$ columns of $\mathsf{A}$

$$\mathsf{A} \equiv \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_n \end{bmatrix}$$

in $\mathbb{R}^m$, we wish to construct $\mathsf{Q} \in \mathbb{R}^{m\times n}$ (whose columns form an orthonormal set in $\mathbb{R}^m$) and $\mathsf{R} \in \mathbb{R}^{n\times n}$ so that

$$\boldsymbol{a}_j = \sum_{i=1}^{j} R_{i,j} \boldsymbol{q}_i \qquad j = 1, 2, \dots, n\,. \tag{6.10}$$

Each column of $\mathsf{Q}$ and $\mathsf{R}$ will be obtained from the corresponding column of $\mathsf{A}$.

For simplicity, we will first describe the Gram–Schmidt algorithm when $\mathsf{A}$ has full rank. This is the most important case, when Corollary 6.2 shows that the least squares problem (6.1) has a unique solution.

**Algorithm 6.8.** *Since* $\mathsf{A}$ *has full rank, its* $1^{\text{st}}$ *column* $\boldsymbol{a}_1 \neq 0$. *Set* $\boldsymbol{q}_1 \equiv \boldsymbol{a}_1/\|\boldsymbol{a}_1\|$ *and* $R_{1,1} \equiv \|\boldsymbol{a}_1\|$.

*For columns* $2 \leqslant j \leqslant n$, *we set*

$$R_{i,j} \equiv \langle \boldsymbol{q}_i, \boldsymbol{a}_j \rangle \qquad i = 1, \dots, j-1$$

*and compute*

$$\boldsymbol{d}_j = \boldsymbol{a}_j - \sum_{i=1}^{j-1} R_{i,j} \boldsymbol{q}_i\,.$$

*Since* $\mathsf{A}$ *has full rank, we must have* $\boldsymbol{d}_j \neq \boldsymbol{0}$: *otherwise the set* $\{\boldsymbol{a}_k\}_{k=1}^{j}$ *would be linearly dependent. Set* $\boldsymbol{q}_j \equiv \boldsymbol{d}_j/\|\boldsymbol{d}_j\|$ *and* $R_{j,j} \equiv \|\boldsymbol{d}_j\|$.

When using this algorithm to solve (6.10) for $\mathsf{Q}$ and $\mathsf{R}$, we see that the only lack of uniqueness in the solution is the choice of sign for each column of $\mathsf{Q}$ and corresponding row of $\mathsf{R}$. Thus we have the following uniqueness result.

**Theorem 6.9.** *If $\mathsf{A} \in \mathbb{R}^{m\times n}$ has full rank, then its skinny $\mathsf{QR}$ factorisation (6.8) is unique provided we impose the restrictions*

$$R_{i,i} > 0 \qquad i = 1, \ldots, n\,.$$

When $\mathsf{A} \in \mathbb{R}^{m\times n}$ has full rank, we can also link its unique skinny $\mathsf{QR}$ factorisation with the unique Cholesky factorisation of the Gram matrix $\mathsf{A}^{\mathrm{T}}\mathsf{A} \in \mathbb{R}^{n\times n}$. (Remember that Corollary 6.2 shows us that $\mathsf{A}^{\mathrm{T}}\mathsf{A}$ is symmetric positive definite and (5.34) has $\mathsf{A}^{\mathrm{T}}\mathsf{A} = \mathsf{G}\mathsf{G}^{\mathrm{T}}$, where $\mathsf{G} \in \mathbb{R}^{n\times n}$ is a lower triangular matrix with positive diagonal elements.) Thus $\mathsf{A}\mathsf{G}^{-\mathrm{T}} \in \mathbb{R}^{m\times n}$ satisfies

$$\left(\mathsf{A}\mathsf{G}^{-\mathrm{T}}\right)^{\mathrm{T}} \mathsf{A}\mathsf{G}^{-\mathrm{T}} = \mathsf{G}^{-1}\mathsf{A}^{\mathrm{T}}\mathsf{A}\mathsf{G}^{-\mathrm{T}} = \mathsf{I} \in \mathbb{R}^{n\times n}$$

and so the columns of $\mathsf{A}\mathsf{G}^{-\mathrm{T}}$ form an orthonormal set in $\mathbb{R}^m$. Hence $\mathsf{A} = \left(\mathsf{A}\mathsf{G}^{-\mathrm{T}}\right)\mathsf{G}^{\mathrm{T}}$ is our unique skinny $\mathsf{QR}$ factorisation, with $\mathsf{A}\mathsf{G}^{-\mathrm{T}}$ playing the role of $\mathsf{Q}$ and $\mathsf{G}^{\mathrm{T}}$ playing the role of $\mathsf{R}$.

*Example.* Let us find the $\mathsf{QR}$ factorization by Gram–Schmidt of

$$\mathsf{A} \equiv \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix}. \tag{6.11}$$

(Square non-singular matrices have full rank and therefore the unique skinny $\mathsf{QR}$ factorisation is also the unique $\mathsf{QR}$ factorisation.) From above

$$\begin{aligned}
&R_{1,1} = \|\boldsymbol{a}_1\| \quad = 3, \quad \boldsymbol{q}_1 = \boldsymbol{a}_1/R_{1,1} = \tfrac{1}{3}\begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix};\\
&R_{1,2} = \langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle = 3, \qquad\qquad \boldsymbol{d}_2 = \boldsymbol{a}_2 - R_{1,2}\boldsymbol{q}_1 = \begin{bmatrix} 4 \\ -1 \\ 1 \end{bmatrix} - 3\cdot\tfrac{1}{3}\begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ -1 \end{bmatrix},\\
&R_{2,2} = \|\boldsymbol{d}_2\| \quad = 3, \quad \boldsymbol{q}_2 = \boldsymbol{d}_2/R_{2,2} = \tfrac{1}{3}\begin{bmatrix} 2 \\ -2 \\ -1 \end{bmatrix};\\
&R_{1,3} = \langle \boldsymbol{q}_1, \boldsymbol{a}_3\rangle = 3, \quad R_{2,3} = \langle \boldsymbol{q}_2, \boldsymbol{a}_3\rangle = 3, \qquad \boldsymbol{d}_3 = \boldsymbol{a}_3 - R_{1,3}\boldsymbol{q}_1 - R_{2,3}\boldsymbol{q}_2\\
&\qquad\qquad\qquad\qquad\qquad\qquad = \begin{bmatrix} 5 \\ 1 \\ -1 \end{bmatrix} - 3\cdot\tfrac{1}{3}\begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - 3\cdot\tfrac{1}{3}\begin{bmatrix} 2 \\ -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix},\\
&R_{3,3} = \|\boldsymbol{d}_3\| \quad = 3, \quad \boldsymbol{q}_3 = \boldsymbol{d}_3/R_{3,3} = \tfrac{1}{3}\begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}.
\end{aligned}$$

So,

$$\mathsf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \underbrace{\frac{1}{3}\begin{bmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\begin{bmatrix} 3 & 3 & 3 \\ & 3 & 3 \\ & & 3 \end{bmatrix}}_{\mathsf{R}}$$

This example reminds us that, if $\mathsf{A}$ is square non-singular, we can solve $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ by calculating the $\mathsf{QR}$ factorization of $\mathsf{A}$ and then proceeding in two steps: i.e.

$$\mathsf{A}\boldsymbol{x} = \mathsf{Q}\underbrace{\mathsf{R}\boldsymbol{x}}_{\boldsymbol{y}} = \boldsymbol{b}\,. \tag{6.12}$$

Remembering $\mathsf{Q}^{-1} = \mathsf{Q}^{\mathrm{T}}$, we first solve $\mathsf{Q}\boldsymbol{y} = \boldsymbol{b}$ in $\mathcal{O}(n^2)$ operations to obtain $\boldsymbol{y} = \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}$, and then solve the triangular system $\mathsf{R}\boldsymbol{x} = \boldsymbol{y}$ in $\mathcal{O}(n^2)$ operations by back-substitution. The work of calculating the $\mathsf{QR}$ factorization makes this method more expensive than the $\mathsf{LU}$ procedure for solving $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$; so the $\mathsf{LU}$ algorithm is more efficient for square non-singular systems.

We want to extend Algorithm 6.8 to the rank-deficient case, but then $\mathsf{A} \in \mathbb{R}^{m\times n}$ can have rank $p$ with $1 \leqslant p \leqslant n$. ($p = 0$ corresponds to $\mathsf{A}$ being zero and $p = n$ corresponds to $\mathsf{A}$ having full rank.) Although we will not usually know $p$ beforehand, it is clear that our skinny $\mathsf{QR}$ can then be further simplified with $\mathsf{Q} \in \mathbb{R}^{m\times p}$ and $\mathbb{R} \in \mathbb{R}^{p\times n}$, i.e.

$$m>n \left\{ \underbrace{\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & & \vdots \\ A_{n,1} & \cdots & A_{n,n} \\ \vdots & & \vdots \\ A_{m,1} & \cdots & A_{m,n} \end{bmatrix}}_{n} = \underbrace{\begin{bmatrix} Q_{1,1} & \cdots & Q_{1,p} \\ \vdots & & \vdots \\ Q_{n,1} & \cdots & Q_{n,p} \\ \vdots & & \vdots \\ Q_{m,1} & \cdots & Q_{m,p} \end{bmatrix}}_{p\leqslant n} \underbrace{\begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,p} & \cdots & R_{1,n} \\ & R_{2,2} & & \vdots & & \vdots \\ & & \ddots & \vdots & & \vdots \\ & & & R_{p,p} & \cdots & R_{p,n} \end{bmatrix}}_{n} \right\} p \tag{6.13}$$

where the columns of $\mathsf{Q}$ again form an orthonormal set in $\mathbb{R}^m$ and $\mathsf{R}$ has rank $p$ and so is full rank. We can now extend our algorithm to the rank-deficient case, but we need to calculate $p$.

**Algorithm 6.10.** *Set $j = 0$, $k = 0$; where we use $j$ to keep track of the number of columns of $\mathsf{A}$ and $\mathsf{R}$ that have been already considered, and $k$ to keep track of the number of columns of $\mathsf{Q}$ that have been formed ($k \leqslant j$). On termination of the algorithm, we will have $k = p$, the rank of $\mathsf{A}$.*

*Step 1. Increase $j$ by 1.*

- *If $k = 0$, set $\boldsymbol{d}_j = \boldsymbol{a}_j$.*
- *If $k \geqslant 1$, set*
$$R_{i,j} \equiv \langle \boldsymbol{q}_i, \boldsymbol{a}_j \rangle \quad \text{for} \quad i = 1, 2, \ldots, k$$
*and compute*
$$\boldsymbol{d}_j = \boldsymbol{a}_j - \sum_{i=1}^{k} R_{i,j}\boldsymbol{q}_i \,.$$

*Step 2.*

- *If $\boldsymbol{d}_j \neq \boldsymbol{0}$, set $\boldsymbol{q}_{k+1} \equiv \boldsymbol{d}_j/\|\boldsymbol{d}_j\|$, $R_{k+1,j} \equiv \|\boldsymbol{d}_j\|$ and (if $j \geqslant k+2$) put $R_{i,j} = 0$ for $j \geqslant i \geqslant k+2$. Finally, increase $k$ by 1.*
- *If $\boldsymbol{d}_j = \boldsymbol{0}$, put $R_{i,j} = 0$ for $j \geqslant i \geqslant k+1$.*

*Step 3. Terminate if $j = n$, otherwise go to Step 1.*

There is no necessity for the diagonal elements $\{R_{i,i}\}_{i=1}^{p}$ in (6.13) to be non-zero: in fact, if $\boldsymbol{a}_1 = \boldsymbol{0}$ then $R_{1,1} = 0$; or, if $\boldsymbol{a}_1 \neq \boldsymbol{0}$ and $\boldsymbol{a}_2$ is a multiple of $\boldsymbol{a}_1$, then $R_{2,2} = 0$. However the matrix $\mathsf{R} \in \mathbb{R}^{p\times n}$ constructed by Algorithm 6.10 is in "standard form".

**Definition 6.11.** We say that a matrix, say $\mathsf{R}$, is in a *standard form* if it has the property that the number of leading zeros in each row increases strictly monotonically: i.e. if $R_{i,j_i}$ is the first nonzero entry in the $i^{\text{th}}$ row, then $\{j_1, \ldots, j_p\}$ is a strictly monotonically increasing sequence. Completely zero rows of $\mathsf{R}$ are also allowed, but they must all be at the bottom of $\mathsf{R}$.

As constructed in the above algorithm, the first non-zero element of each row of $\mathsf{R}$ also has the property that it is greater than zero. This gives us conditions that make the skinny $\mathsf{QR}$ factorisation unique.

**Theorem 6.12.** *If $\mathsf{A} \in \mathbb{R}^{m\times n}$ has rank $p$, with $1 \leqslant p \leqslant n$, then the skinny $\mathsf{QR}$ factorisation (6.13) is unique if*

- *$\mathsf{R}$ is in standard form;*
- *the first non-zero element in each row of $\mathsf{R}$ is greater than zero.*

Finally note that a skinny QR factorisation can always be converted into a QR factorisation satisfying Definition 6.7. We only have to

- append extra zero rows to $\mathsf{R}$ until $\mathsf{R} \in \mathbb{R}^{n\times n}$;
- append extra columns to $\mathsf{Q}$, using Lemma 6.6, until $\mathsf{Q} \in \mathbb{R}^{m\times m}$ is orthogonal.

*Example.* Let us find the skinny QR factorization by Gram–Schmidt of

$$\mathsf{A} \equiv \begin{bmatrix} 2 & 4 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{6.14}$$

| | | | | | |
|---|---|---|---|---|---|
| $k=0$ | $j=1$ | | $\boldsymbol{d}_1 = [2,1,1,0]^{\mathrm{T}}$ | $\boldsymbol{q}_1 \equiv [2,1,1,0]^{\mathrm{T}}/\sqrt{6}$ | $R_{1,1} \equiv \sqrt{6}$ |
| $k=1$ | $j=2$ | $R_{1,2} \equiv 2\sqrt{6}$ | $\boldsymbol{d}_2 = \mathbf{0}$ | | $R_{2,2} \equiv 0$ |
| $k=1$ | $j=3$ | $R_{1,3} \equiv 0$ | $\boldsymbol{d}_3 = [0,0,0,1]^{\mathrm{T}}$ | $\boldsymbol{q}_2 \equiv [0,0,0,1]^{\mathrm{T}}$ | $R_{2,3} \equiv 1$ |

Hence $\mathsf{A}$ has rank $p = 2$ and

$$\mathsf{A} \equiv \begin{bmatrix} 2 & 4 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \underbrace{\frac{1}{\sqrt{6}} \begin{bmatrix} 2 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & \sqrt{6} \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\sqrt{6} \begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 1/\sqrt{6} \end{bmatrix}}_{\mathsf{R}}$$

*Modified Gram–Schmidt.* We should mention that the precise form of the Gram–Schmidt algorithm (as described in this section) is not used in practice because the numerical results can be very poor with inexact arithmetic. In particular, the columns of the computed $\mathsf{Q}$ matrix can fail to be anywhere near orthogonal! We can give a simple example of this behaviour by exhibiting the MATLAB code for Algorithm 6.8, i.e.

```
r=zeros(n);
for j = 1:n
  d=a(:,j);
  for i=1:j-1
    r(i,j)=a(:,j)'*q(:,i);
    d=d-r(i,j)*q(:,i);
  end
  r(j,j)=norm(d);
  q(:,j)=d/r(j,j);
end
```

If this code is applied to the matrix

$$\mathsf{A} \equiv \begin{bmatrix} 1 & 1 & 1 \\ \epsilon & & \\ & \epsilon & \\ & & \epsilon \end{bmatrix} \qquad \epsilon \ll 1 \tag{6.15}$$

and our finite precision arithmetic is that all higher powers of $\epsilon$ are neglected during the calculations, then the final computed QR factorisation is

$$\mathsf{A} \approx \begin{bmatrix} 1 & 0 & 0 \\ \epsilon & -1/\sqrt{2} & -1/\sqrt{2} \\ 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ & \sqrt{2}\epsilon & 0 \\ & & \sqrt{2}\epsilon \end{bmatrix}$$

Thus $\boldsymbol{q}_2^{\mathrm{T}}\boldsymbol{q}_3 = \frac{1}{2}$!

Fortunately the algorithm can be "saved" in finite precision arithmetic by simply swapping the order of the `for` loops. This is the modified Gram–Schmidt (MGS) algorithm and its MATLAB code becomes

```
q=a; r=zeros(n);
for i=1:n
  r(i,i)=norm(q(:,i));
  q(:,i)=q(:,i)/r(i,i);
  for j=i+1:n
    r(i,j)=q(:,j)'*q(:,i);
    q(:,j)=q(:,j)-r(i,j)*q(:,i);
  end
end
```

The numerical results, when this code is applied to $\mathsf{A}$ in (6.15) using our finite precision arithmetic, are now

$$\mathsf{A} \approx \begin{bmatrix} 1 & 0 & 0 \\ \epsilon & -1/\sqrt{2} & -1/\sqrt{6} \\ 0 & 1/\sqrt{2} & -1/\sqrt{6} \\ 0 & 0 & 2/\sqrt{6} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ & \sqrt{2}\epsilon & \epsilon/\sqrt{2} \\ & & \sqrt{3}\epsilon/\sqrt{2} \end{bmatrix},$$

which is much better.

### 6.4.2 Givens rotation matrices

In $\mathbb{R}^2$, a clockwise rotation of $\theta \in [-\pi, \pi]$ is performed by the orthogonal matrix

$$\Omega_\theta \equiv \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \quad \Rightarrow \quad \Omega_\theta \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha\cos\theta + \beta\sin\theta \\ -\alpha\sin\theta + \beta\cos\theta \end{bmatrix}.$$

Note that, as proved in Proposition 6.4, the Euclidean vector norm is invariant. If $[\alpha, \beta]$ is non-zero then, by choosing

$$\cos\theta \equiv \frac{\alpha}{\sqrt{\alpha^2+\beta^2}} \qquad \text{and} \qquad \sin\theta \equiv \frac{\beta}{\sqrt{\alpha^2+\beta^2}},$$

the righthand side becomes $[\sqrt{\alpha^2+\beta^2}, 0]^{\mathrm{T}}$; similarly, by choosing

$$\cos\theta \equiv \frac{\beta}{\sqrt{\alpha^2+\beta^2}} \qquad \text{and} \qquad \sin\theta \equiv \frac{-\alpha}{\sqrt{\alpha^2+\beta^2}},$$

the righthand side becomes $[0, \sqrt{\alpha^2+\beta^2}]^{\mathrm{T}}$. (We do not need to calculate $\theta$ itself, the values of $\cos\theta$ and $\sin\theta$ are sufficient.)

In $\mathbb{R}^m$, with $m > 2$, Givens rotation matrices are defined by 3 parameters ($1 \leqslant p < q \leqslant m$ and $\theta \in [-\pi, \pi]$), i.e.

$$\Omega_\theta^{[p,q]} \equiv \begin{bmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & \cos\theta & & & \sin\theta & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & -\sin\theta & & & \cos\theta & & \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ & & & & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ p \\ \\ \\ \\ q \\ \\ \\ \end{matrix}$$

$$\begin{matrix} & & p & & & & q & & \end{matrix}$$

$(p, q)$ determines the plane of rotation and $\theta$ is the rotation in that plane, e.g.

$$m = 4 \quad \Longrightarrow \quad \Omega_\theta^{[1,2]} \equiv \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \Omega_\theta^{[2,4]} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & 0 & \sin\theta \\ 0 & 0 & 1 & 0 \\ 0 & -\sin\theta & 0 & \cos\theta \end{bmatrix}.$$

*Remarks*

(i) For $\boldsymbol{y} \in \mathbb{R}^m$, $\|\Omega_\theta^{[p,q]}\boldsymbol{y}\| = \|\boldsymbol{y}\|$. $\Omega_\theta^{[p,q]}\boldsymbol{y}$ only alters components $p$ and $q$ of $\boldsymbol{y}$: the new components being linear combinations of the old components.

(ii) If $\mathsf{B} \in \mathbb{R}^{m\times n}$ has columns $\{\boldsymbol{b}_j\}_{j=1}^n$, then $\|\Omega_\theta^{[p,q]}\boldsymbol{b}_j\| = \|\boldsymbol{b}_j\|$ for $j = 1, \ldots, n$. $\Omega_\theta^{[p,q]}\mathsf{B}$ only alters rows $p$ and $q$ of $\mathsf{B}$: the new rows being linear combinations of the old rows.

(iii) Given $\boldsymbol{z} \in \mathbb{R}^m$, with $z_p^2 + z_q^2 > 0$, we can either choose $\theta$ so that $\left[\Omega_\theta^{[p,q]}\boldsymbol{z}\right]_p = 0$, i.e.

$$\cos\theta \equiv \frac{z_q}{\sqrt{z_p^2 + z_q^2}} \qquad \text{and} \qquad \sin\theta \equiv \frac{-z_p}{\sqrt{z_p^2 + z_q^2}},$$

or choose $\theta$ so that $\left[\Omega_\theta^{[p,q]}\boldsymbol{z}\right]_q = 0$, i.e.

$$\cos\theta \equiv \frac{z_p}{\sqrt{z_p^2 + z_q^2}} \qquad \text{and} \qquad \sin\theta \equiv \frac{z_q}{\sqrt{z_p^2 + z_q^2}}.$$

We now explain how a full rank $\mathsf{A} \in \mathbb{R}^{m\times n}$ can be transformed into "upper triangular" form by applying $s \equiv mn - n(n+1)/2$ Givens rotations: i.e.

$$\mathsf{Q}_s \ldots \mathsf{Q}_1 \mathsf{A} = \mathsf{R},$$

where each element of $\{\mathsf{Q}_k\}_{k=1}^s$ is a Givens rotation. $\mathsf{R} \in \mathbb{R}^{m\times n}$ is an "upper triangular" matrix with the last $m - n$ rows zero, as in (6.6): thus $s$ components of $\mathsf{A}$ have been forced to be zero. The basic idea becomes clear with a small example.

*An illustration.* If $\mathsf{A} \in \mathbb{R}^{4\times 3}$, so that $s = 6$, we can force zeros underneath the main diagonal as follows.

(i) If $A_{2,1} \neq 0$, choose $\theta_1$ so that $(\Omega_{\theta_1}^{[1,2]}\mathsf{A})_{2,1} = 0$, i.e.

$$\mathsf{A}^{(1)} \equiv \Omega_{\theta_1}^{[1,2]}\mathsf{A} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}.$$

(ii) If $A_{3,1}^{(1)} \neq 0$, choose $\theta_2$ so that $(\Omega_{\theta_2}^{[1,3]}\mathsf{A}^{(1)})_{3,1} = 0$. Multiplication by $\Omega_{\theta_2}^{[1,3]}$ doesn't alter the second row, hence $(\Omega_{\theta_2}^{[1,3]}\mathsf{A}^{(1)})_{2,1}$ remains zero and

$$\mathsf{A}^{(2)} \equiv \Omega_{\theta_2}^{[1,3]}\mathsf{A}^{(1)} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ \times & \times & \times \end{bmatrix}.$$

(iii) If $A_{4,1}^{(2)} \neq 0$, choose $\theta_3$ so that $(\Omega_{\theta_3}^{[1,4]}\mathsf{A}^{(2)})_{4,1} = 0$. Multiplication by $\Omega_{\theta_3}^{[1,4]}$ doesn't alter the second and third rows, hence $(\Omega_{\theta_3}^{[1,4]}\mathsf{A}^{(2)})_{2,1}$ and $(\Omega_{\theta_3}^{[1,4]}\mathsf{A}^{(2)})_{3,1}$ remain zero and

$$\mathsf{A}^{(3)} \equiv \Omega_{\theta_3}^{[1,4]}\mathsf{A}^{(2)} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}.$$

(iv) If $A^{(3)}_{3,2} \neq 0$, choose $\theta_4$ so that $(\Omega^{[2,3]}_{\theta_4}\mathsf{A}^{(3)})_{3,2} = 0$. Multiplication by $\Omega^{[2,3]}_{\theta_4}$ doesn't alter the fourth row, hence $(\Omega^{[2,3]}_{\theta_4}\mathsf{A}^{(3)})_{4,1}$ remains zero. Since both the second and third rows of $\mathsf{A}^{(3)}$ have a leading zero, their linear combination preserves these zeros; hence

$$(\Omega^{[2,3]}_{\theta_4}\mathsf{A}^{(3)})_{2,1} = (\Omega^{[2,3]}_{\theta_4}\mathsf{A}^{(3)})_{3,1} = 0$$

and

$$\mathsf{A}^{(4)} \equiv \Omega^{[2,3]}_{\theta_4}\mathsf{A}^{(3)} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & \times & \times \end{bmatrix}.$$

(v) If $A^{(4)}_{4,2} \neq 0$, choose $\theta_5$ so that $(\Omega^{[2,4]}_{\theta_5}\mathsf{A}^{(4)})_{4,2} = 0$. Multiplication by $\Omega^{[2,4]}_{\theta_5}$ doesn't alter the third row, hence

$$(\Omega^{[2,4]}_{\theta_5}\mathsf{A}^{(4)})_{3,1} = (\Omega^{[2,4]}_{\theta_5}\mathsf{A}^{(4)})_{3,2} = 0\,.$$

Since both the second and fourth rows of $\mathsf{A}^{(4)}$ have a leading zero, their linear combination preserves these zeros; hence

$$(\Omega^{[2,4]}_{\theta_5}\mathsf{A}^{(4)})_{2,1} = (\Omega^{[2,4]}_{\theta_5}\mathsf{A}^{(4)})_{4,1} = 0$$

and

$$\mathsf{A}^{(5)} \equiv \Omega^{[2,4]}_{\theta_5}\mathsf{A}^{(4)} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{bmatrix}.$$

(vi) Finally, if $A^{(5)}_{4,3} \neq 0$, choose $\theta_6$ so that $(\Omega^{[3,4]}_{\theta_6}\mathsf{A}^{(5)})_{4,3} = 0$. Multiplication by $\Omega^{[3,4]}_{\theta_6}$ doesn't alter the second row, hence

$$(\Omega^{[3,4]}_{\theta_6}\mathsf{A}^{(5)})_{2,1} = 0\,.$$

Since both the third and fourth rows of $\mathsf{A}^{(5)}$ have two leading zeros, their linear combination preserves these zeros; hence

$$(\Omega^{[3,4]}_{\theta_6}\mathsf{A}^{(5)})_{3,1} = (\Omega^{[3,4]}_{\theta_6}\mathsf{A}^{(5)})_{4,1} = 0 \qquad \text{and} \qquad (\Omega^{[3,4]}_{\theta_6}\mathsf{A}^{(5)})_{3,2} = (\Omega^{[3,4]}_{\theta_6}\mathsf{A}^{(5)})_{4,2} = 0\,.$$

Thus we end up with

$$\mathsf{A}^{(6)} \equiv \Omega^{[3,4]}_{\theta_6}\mathsf{A}^{(5)} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \end{bmatrix}.$$

So we have $\mathsf{A} = \mathsf{QR}$ with $\mathsf{R} = \mathsf{A}^{(6)}$ and

$$\mathsf{Q} \equiv \left(\Omega^{[3,4]}_{\theta_6}\Omega^{[2,4]}_{\theta_5}\Omega^{[2,3]}_{\theta_4}\Omega^{[1,4]}_{\theta_3}\Omega^{[1,3]}_{\theta_2}\Omega^{[1,2]}_{\theta_1}\right)^{\mathrm{T}}.$$

Another informative picture is to emphasize which components have been changed at each step: i.e.

$$\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \overset{\Omega^{[1,2]}}{\to} \begin{bmatrix} \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet \\ * & * & * \\ * & * & * \end{bmatrix} \overset{\Omega^{[1,3]}}{\to} \begin{bmatrix} \bullet & \bullet & \bullet \\ 0 & * & * \\ 0 & \bullet & \bullet \\ * & * & * \end{bmatrix} \overset{\Omega^{[1,4]}}{\to} \begin{bmatrix} \bullet & \bullet & \bullet \\ 0 & * & * \\ 0 & * & * \\ 0 & \bullet & \bullet \end{bmatrix} \overset{\Omega^{[2,3]}}{\to} \begin{bmatrix} * & * & * \\ 0 & \bullet & \bullet \\ 0 & 0 & \bullet \\ 0 & * & * \end{bmatrix} \overset{\Omega^{[2,4]}}{\to} \begin{bmatrix} * & * & * \\ 0 & \bullet & \bullet \\ 0 & 0 & * \\ 0 & 0 & \bullet \end{bmatrix} \overset{\Omega^{[3,4]}}{\to} \begin{bmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & \bullet \\ 0 & 0 & 0 \end{bmatrix},$$

where the $\bullet$-elements have been changed by the current rotation while the $*$-elements have not.

There are alternative strategies for systematically introducing the $s$ zeros into $\mathsf{A}$: e.g. one can start at the bottom of each column and use rotations of the form $\Omega^{[m-1,m]}, \Omega^{[m-2,m-1]}, \ldots$ to gradually create zeros into that column. The key point is that previously created zeros must not be destroyed. (Of course, it could turn out that less than $s$ Givens rotations are required: e.g. $\mathsf{A}$ may already contain some zeros.)

*Example.* We repeat the simple example (6.11) in §6.4.1 by using Givens rotations to find the QR factorization of

$$\mathsf{A} \equiv \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix}.$$

$$\Omega^{[1,2]}\mathsf{A} = \underbrace{\begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & 0 \\ -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\Omega^{[1,2]}} \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} \sqrt{5} & \frac{7}{\sqrt{5}} & \frac{11}{\sqrt{5}} \\ 0 & -\frac{6}{\sqrt{5}} & -\frac{3}{\sqrt{5}} \\ 2 & 1 & -1 \end{bmatrix},$$

$$\Omega^{[1,3]}(\Omega^{[1,2]}\mathsf{A}) = \underbrace{\begin{bmatrix} \frac{\sqrt{5}}{3} & 0 & \frac{2}{3} \\ 0 & 1 & 0 \\ -\frac{2}{3} & 0 & \frac{\sqrt{5}}{3} \end{bmatrix}}_{\Omega^{[1,3]}} \begin{bmatrix} \sqrt{5} & \frac{7}{\sqrt{5}} & \frac{11}{\sqrt{5}} \\ 0 & -\frac{6}{\sqrt{5}} & -\frac{3}{\sqrt{5}} \\ 2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & -\frac{6}{\sqrt{5}} & -\frac{3}{\sqrt{5}} \\ 0 & -\frac{3}{\sqrt{5}} & -\frac{9}{\sqrt{5}} \end{bmatrix},$$

$$\Omega^{[2,3]}(\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A}) = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \\ 0 & \frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} \end{bmatrix}}_{\Omega^{[2,3]}} \begin{bmatrix} 3 & 3 & 3 \\ 0 & -\frac{6}{\sqrt{5}} & -\frac{3}{\sqrt{5}} \\ 0 & -\frac{3}{\sqrt{5}} & -\frac{9}{\sqrt{5}} \end{bmatrix} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 0 & 3 \end{bmatrix}.$$

Finally,

$$\mathsf{Q}^{\mathrm{T}} = \Omega^{[2,3]}\Omega^{[1,3]}\Omega^{[1,2]} = \frac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 2 & -2 & -1 \\ 1 & 2 & -2 \end{bmatrix},$$

$$\mathsf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \underbrace{\frac{1}{3}\begin{bmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\begin{bmatrix} 3 & 3 & 3 \\ & 3 & 3 \\ & & 3 \end{bmatrix}}_{\mathsf{R}}.$$

Note that our Givens rotations are designed to produce positive diagonal elements in R, so that our computed QR factorisation is exactly the same as before. (If we had ended up with $R_{3,3} < 0$, a final multiplication by the orthogonal matrix $\mathsf{H}_{\boldsymbol{e}_3}$ would have been necessary. See Definition 6.15.)

We will now state clearly our Givens rotation algorithm for transforming a full rank $\mathsf{A} \in \mathbb{R}^{m\times n}$ into "upper triangular" form.

**Algorithm 6.13.** *For column* 1*, since* A *has full rank,* $\sum_{i=1}^{m} A_{i,1}^2 \neq 0$*. Use Givens rotations* $\{\Omega^{[1,i]}\}_{i=2}^{m}$ *to create zeros in elements* $\{(i,1)\}_{i=2}^{m}$ *of* A *respectively and leave* $A_{1,1} > 0$*.*

*For columns* $2 \leqslant j \leqslant n$*, since the transformed* A *still has full rank,* $\sum_{i=j}^{m} A_{i,j}^2 \neq 0$*. Use Givens rotations* $\{\Omega^{[j,i]}\}_{i=j+1}^{m}$ *to create zeros in elements* $\{(i,j)\}_{i=j+1}^{m}$ *of* A *respectively and leave* $A_{j,j} > 0$*. (This will not destroy any zeros created in the previous columns of* A*.)*

After completion of this algorithm, A has been overwritten by

$$\mathsf{R} = \left(\Omega^{[n,m]}\cdots\Omega^{[n,n+1]}\right)\cdots\left(\Omega^{[2,m]}\cdots\Omega^{[2,3]}\right)\left(\Omega^{[1,m]}\cdots\Omega^{[1,3]}\Omega^{[1,2]}\right)\mathsf{A} \in \mathbb{R}^{m\times n}$$

and so we have a QR factorisation of the form (6.6) with

$$\mathsf{Q} \equiv \left(\Omega^{[1,m]}\cdots\Omega^{[1,2]}\right)^{\mathrm{T}}\left(\Omega^{[2,m]}\cdots\Omega^{[2,3]}\right)^{\mathrm{T}}\cdots\left(\Omega^{[n,m]}\cdots\Omega^{[n,n+1]}\right)^{\mathrm{T}} \in \mathbb{R}^{m\times m}. \tag{6.16}$$

Since the diagonal elements of $\{R_{i,i}\}_{i=1}^{n}$ are all positive, if we neglect the last $m-n$ columns of Q and the last $m-n$ zero rows of R then we end up with exactly the unique skinny QR factorisation produced by Algorithm 6.8.

*Remarks*

(a) Since at least one more zero is introduced into the matrix with each Givens rotation, there are less than $mn$ rotations. Since each rotation replaces two rows (of length $n$) by their linear combinations, the total cost of computing $\mathsf{R}$ is $\mathcal{O}(mn^2)$ operations.

(b) The matrix $\mathsf{Q}$ may be required explicitly: say, for solving many least squares problems with the same $\mathsf{A}$ but different $\boldsymbol{b}$. It can be calculated from (6.16) in $\mathcal{O}(m^2 n)$ operations.

(c) If only one vector $\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}$ is required (e.g. in the case of the solution of a single least squares problem), it is more efficient to multiply $\boldsymbol{b}$ by successive rotations: the cost being $\mathcal{O}(mn)$ operations.

(d) For $m = n$, each rotation requires four times more multiplications compared with the corresponding Gaussian elimination: hence the total cost is $\frac{4}{3}n^3 + \mathcal{O}(n^2)$ operations, four times as expensive. However, the $\mathsf{QR}$ factorization is generally more reliable than the $\mathsf{LU}$ one.

We will now generalise the above algorithm so that it can deal with the rank-deficient case and produce a final $\mathsf{R} \in \mathbb{R}^{m\times n}$ in standard form. (Just like the extension of the Gram–Schmidt process from Algorithm 6.8 to Algorithm 6.10, we need to calculate the rank of $\mathsf{A}$.)

**Algorithm 6.14.** *Set $j = 0$, $k = 0$; where we use $j$ to keep track of the number of columns of $\mathsf{A}$ and $\mathsf{R}$ that have been considered, and $k$ to count the number of linearly independent columns of $\mathsf{A}$ ($k \leqslant j$).*

*Step 1. Increase $j$ by 1.*

- *If $\{A_{i,j}\}_{i=k+1}^{m}$ are all zero, go to Step 2.*
- *Otherwise, use Givens rotations $\{\Omega^{[k+1,i]}\}_{i=k+2}^{m}$ to create zeros in elements $\{(i,j)\}_{i=k+2}^{m}$ of $\mathsf{A}$ respectively. (This will not destroy any zeros created in earlier columns of $\mathsf{A}$.) Increase $k$ by 1.*

*Step 2. Terminate if $j = n$, otherwise go to Step 1.*

If the final value of $k$ produced by this algorithm is $p$, then $\mathsf{A} \in \mathbb{R}^{m\times n}$ has been transformed into an "upper triangular"

$$\mathsf{R} = \left(\Omega^{[p,m]} \cdots \Omega^{[p,n+1]}\right) \cdots \left(\Omega^{[2,m]} \cdots \Omega^{[2,3]}\right) \left(\Omega^{[1,m]} \cdots \Omega^{[1,3]}\Omega^{[1,2]}\right) \mathsf{A} \in \mathbb{R}^{m\times n}$$

whose last $m - p$ rows are zero. Thus we have a $\mathsf{QR}$ factorisation of the form (6.6) with

$$\mathsf{Q} \equiv \left(\Omega^{[1,m]} \cdots \Omega^{[1,2]}\right)^{\mathrm{T}} \left(\Omega^{[2,m]} \cdots \Omega^{[2,3]}\right)^{\mathrm{T}} \cdots \left(\Omega^{[p,m]} \cdots \Omega^{[p,p+1]}\right)^{\mathrm{T}} \in \mathbb{R}^{m\times m}\,.$$

Since the leading non-zero element in each row of $\mathsf{R}$ is positive, if we neglect the last $m - p$ columns of $\mathsf{Q}$ and the last $m - p$ zero rows of $\mathsf{R}$ then we end up with exactly the unique skinny $\mathsf{QR}$ factorisation produced by Algorithm 6.10.

*Example.* Let us find the $\mathsf{QR}$ factorization of

$$\mathsf{A} \equiv \begin{bmatrix} 2 & 4 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

as in (6.14) of §6.4.1, using Jacobi rotations.

$k=0$ $j=1$ $p=1 \quad q=2 \quad \cos\theta_1 = \dfrac{2}{\sqrt{5}} \quad \sin\theta_1 = \dfrac{1}{\sqrt{5}}$

$$\mathsf{A} \to \Omega_{\theta_1}^{[1,2]}\mathsf{A} = \begin{bmatrix} \sqrt{5} & 2\sqrt{5} & 0 \\ 0 & 0 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$p=1 \quad q=3 \quad \cos\theta_2 = \dfrac{\sqrt{5}}{\sqrt{6}} \quad \sin\theta_2 = \dfrac{1}{\sqrt{6}}$

$$\mathsf{A} \to \Omega_{\theta_2}^{[1,3]}\mathsf{A} = \begin{bmatrix} \sqrt{6} & 2\sqrt{6} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$k=1$ $j=2$ No change to $\mathsf{A}$

$k=1$ $j=3$ $p=2 \quad q=4 \quad \cos\theta_3 = 0 \quad \sin\theta_3 = 1$

$$\mathsf{A} \to \Omega_{\theta_3}^{[2,4]}\mathsf{A} = \begin{bmatrix} \sqrt{6} & 2\sqrt{6} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Hence $\mathsf{A}$ has rank $p=2$ and, since

$$\begin{aligned} \mathsf{Q} &\equiv \left[\Omega_{\theta_1}^{[1,2]}\right]^{\mathrm{T}} \left[\Omega_{\theta_2}^{[1,3]}\right]^{\mathrm{T}} \left[\Omega_{\theta_3}^{[2,4]}\right]^{\mathrm{T}} \\ &= \frac{1}{\sqrt{5}}\begin{bmatrix} 2 & -1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & \sqrt{5} & 0 \\ 0 & 0 & 0 & \sqrt{5} \end{bmatrix} \frac{1}{\sqrt{6}}\begin{bmatrix} \sqrt{5} & 0 & -1 & 0 \\ 0 & \sqrt{6} & 0 & 0 \\ 1 & 0 & \sqrt{5} & 0 \\ 0 & 0 & 0 & \sqrt{6} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 2/\sqrt{6} & 0 & -2/\sqrt{30} & 1/\sqrt{5} \\ 1/\sqrt{6} & 0 & -1/\sqrt{30} & -2/\sqrt{5} \\ 1/\sqrt{6} & 0 & 5/\sqrt{30} & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \end{aligned}$$

we have

$$\mathsf{A} \equiv \begin{bmatrix} 2 & 4 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 2/\sqrt{6} & 0 & -2/\sqrt{30} & 1/\sqrt{5} \\ 1/\sqrt{6} & 0 & -1/\sqrt{30} & -2/\sqrt{5} \\ 1/\sqrt{6} & 0 & 5/\sqrt{30} & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\begin{bmatrix} \sqrt{6} & 2\sqrt{6} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathsf{R}}$$

### 6.4.3 Householder reflection matrices

This is another example of a class of simple orthogonal matrices. They are an alternative to Givens rotations for constructing the $\mathsf{QR}$ factorization in (6.6).

**Definition 6.15.** For $\boldsymbol{u} \neq \mathbf{0} \in \mathbb{R}^m$,

$$\mathsf{H}_{\boldsymbol{u}} \equiv \mathsf{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2} \in \mathbb{R}^{m\times m} \tag{6.17}$$

is called a *Householder reflection.*

*Remarks*

(i) Each Householder reflection is clearly a symmetric matrix and

$$\left(\mathsf{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2}\right)^2 = \mathsf{I} - 4\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2} + 4\frac{\boldsymbol{u}(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u})\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^4} = \mathsf{I}$$

immediately shows that they are orthogonal.

(ii) The reason they are called reflections becomes clear if we uniquely decompose an arbitrary $\boldsymbol{x} \in \mathbb{R}^m$ into

$$\boldsymbol{x} = \alpha\boldsymbol{u} + \boldsymbol{w} \qquad \text{where} \quad \alpha \equiv \boldsymbol{u}^{\mathrm{T}}\boldsymbol{x}/\|\boldsymbol{u}\|^2 \quad \text{and} \quad \boldsymbol{u}^{\mathrm{T}}\boldsymbol{w} = 0 \ :$$

i.e. this is an orthogonal decomposition with $\alpha\boldsymbol{u}$ parallel to $\boldsymbol{u}$ and $\boldsymbol{w}$ perpendicular to $\boldsymbol{u}$. Hence

$$\mathsf{H}_{\boldsymbol{u}}\boldsymbol{u} = -\boldsymbol{u} \quad \text{and} \qquad \mathsf{H}_{\boldsymbol{u}}\boldsymbol{v} = \boldsymbol{v} \quad \text{if} \quad \boldsymbol{u}^{\mathrm{T}}\boldsymbol{v} = 0$$

means that

$$\mathsf{H}_{\boldsymbol{u}}\boldsymbol{x} = -\alpha\boldsymbol{u} + \boldsymbol{w} \ :$$

i.e. our matrix reflects any $\boldsymbol{x} \in \mathbb{R}^m$ in the $(m-1)$-dimensional hyperplane orthogonal to $\boldsymbol{u}$.

(iii) For any $\lambda \neq 0 \in \mathbb{R}$, $\mathsf{H}_{\lambda\boldsymbol{u}} = \mathsf{H}_{\boldsymbol{u}}$.

(iv) We avoid explicitly forming $\mathsf{H}_{\boldsymbol{u}}$ if possible, because it is usually more efficient to *operate* with it. Thus for $\boldsymbol{z} \in \mathbb{R}^m$,

$$\mathsf{H}_{\boldsymbol{u}}\boldsymbol{z} = \boldsymbol{z} - 2\frac{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{z}}{\|\boldsymbol{u}\|^2}\boldsymbol{u}$$

can be constructed in $\mathcal{O}(m)$ operations: in general, a matrix-vector multiplication requires $\mathcal{O}(m^2)$ operations.

Our aim is to show how any $\mathsf{A} \in \mathbb{R}^{m\times n}$ can be transformed into "upper triangular" form by applying $n$ suitable Householder reflections: i.e.

$$\mathsf{R} = \mathsf{H}_n \cdots \mathsf{H}_2\mathsf{H}_1\mathsf{A} \tag{6.18}$$

will produce the $\mathsf{QR}$ factorisation (6.6) with

$$\mathsf{Q} \equiv \mathsf{H}_1\mathsf{H}_2 \cdots \mathsf{H}_n \,. \tag{6.19}$$

Each $\mathsf{H}_k$ will be designed to introduce zeros into column $k$ of $\mathsf{A}$, while leaving unchanged the zeros already created in earlier columns.

The following lemma explains how we can create zeros using reflections.

**Lemma 6.16.** *If $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^m$, with $\boldsymbol{c} \neq \boldsymbol{d}$ but $\|\boldsymbol{c}\| = \|\boldsymbol{d}\|$, then*

$$\boldsymbol{u} \equiv \boldsymbol{c} - \boldsymbol{d} \quad \Rightarrow \quad \mathsf{H}_{\boldsymbol{u}}\boldsymbol{c} = \boldsymbol{d} \,. \tag{6.20}$$

*Proof.*

$$\mathsf{H}_{\boldsymbol{u}}\boldsymbol{c} = \boldsymbol{c} - \frac{2(\|\boldsymbol{c}\|^2 - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{d})}{\|\boldsymbol{c}\|^2 - 2\boldsymbol{c}^{\mathrm{T}}\boldsymbol{d} + \|\boldsymbol{d}\|^2}(\boldsymbol{c} - \boldsymbol{d}) = \boldsymbol{d}$$

□

We can immediately use Lemma 6.16 to construct $\mathsf{H}_1$ in (6.18): i.e. if $\boldsymbol{a} \in \mathbb{R}^m$ is the first column of $\mathsf{A}$ then, unless $\boldsymbol{a}$ is a non-negative multiple of $\boldsymbol{e}_1$, we can choose

$$\boldsymbol{u}_1 \equiv \boldsymbol{a} - \gamma\boldsymbol{e}_1 \quad \Rightarrow \quad \mathsf{H}_{\boldsymbol{u}_1}\boldsymbol{a} = \gamma\boldsymbol{e}_1 \,, \tag{6.21}$$

where $\gamma \equiv \|\boldsymbol{a}\|$. Thus $\mathsf{H}_1 \equiv \mathsf{H}_{\boldsymbol{u}_1}$ and $\mathsf{H}_1\mathsf{A}$ has the required zeros in column 1, i.e.

$$\mathsf{H}_1\mathsf{A} = \begin{bmatrix} * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & * & * & \cdots & * \end{bmatrix} .$$

(With inexact arithmetic, and when $a_1 > 0$, it is preferable to use the equivalent formula $\left(-\sum_{i=2}^m a_i^2\right)/(a_1+\gamma)$ for the $1^{\text{st}}$ component of $\boldsymbol{u}_1$ in (6.21). This avoids possible loss of precision due to cancellation.)

To introduce zeros into later columns of $\mathsf{A}$, we need to make sure that our chosen reflections will not destroy previously created zeros.

**Lemma 6.17.** *If the first $k-1$ components of $\boldsymbol{u} \neq \boldsymbol{0} \in \mathbb{R}^m$ are zero then*

- *$\forall \boldsymbol{x} \in \mathbb{R}^m$, forming $\mathsf{H}_{\boldsymbol{u}}\boldsymbol{x}$ does not alter the first $k-1$ components of $\boldsymbol{x}$;*
- *if the last $m-k+1$ components of $\boldsymbol{y} \in \mathbb{R}^m$ are zero, so that $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{y} = 0$, then $\mathsf{H}_{\boldsymbol{u}}\boldsymbol{y} = \boldsymbol{y}$.*

We can now use Lemma 6.17 to generalise Lemma 6.16.

**Lemma 6.18.** *Let $1 \leqslant k \leqslant m$. If $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^m$, with*

$$\begin{bmatrix} c_k \\ \vdots \\ c_m \end{bmatrix} \neq \begin{bmatrix} d_k \\ \vdots \\ d_m \end{bmatrix} \qquad \textit{but} \quad \sum_{i=k}^m c_i^2 = \sum_{i=k}^m d_i^2 \,,$$

*then*

$$\boldsymbol{u} \equiv [0, \ldots, 0, c_k - d_k, \ldots, c_m - d_m]^{\mathrm{T}} \quad \Rightarrow \quad \mathsf{H}_{\boldsymbol{u}}\boldsymbol{c} = [c_1, \ldots, c_{k-1}, d_k, \ldots, d_m]^{\mathrm{T}} \,. \tag{6.22}$$

We can immediately use Lemma 6.18 to construct $\mathsf{H}_2$ in (6.18): i.e. if $\boldsymbol{a} \in \mathbb{R}^m$ is the second column of $\mathsf{H}_1\mathsf{A}$ then, unless $[0, a_2, \ldots, a_m]^{\mathrm{T}}$ is a non-negative multiple of $\boldsymbol{e}_2$, we can choose

$$\boldsymbol{u}_2 \equiv [0, a_2 - \gamma, a_3, \ldots, a_m]^{\mathrm{T}} \quad \Rightarrow \quad \mathsf{H}_{\boldsymbol{u}_2}\boldsymbol{a} = [a_1, \gamma, 0, \ldots, 0]^{\mathrm{T}} \,, \tag{6.23}$$

where $\gamma \equiv \left(\sum_{i=2}^m a_i^2\right)^{1/2}$. Thus $\mathsf{H}_2 \equiv \mathsf{H}_{\boldsymbol{u}_2}$ and $\mathsf{H}_2\mathsf{H}_1\mathsf{A}$ has the required zeros in columns 1 and 2, i.e.

$$\mathsf{H}_2\mathsf{H}_1\mathsf{A} = \begin{bmatrix} * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & 0 & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & * & \cdots & * \end{bmatrix} ,$$

and multiplying by $\mathsf{H}_2$ did not alter the first row and column of $\mathsf{H}_1\mathsf{A}$. (With inexact arithmetic, and when $a_2 > 0$, it is preferable to use the equivalent formula $\left(-\sum_{i=3}^m a_i^2\right)/(a_2+\gamma)$ for the $2^{\text{nd}}$ component of $\boldsymbol{u}_2$ in (6.23). This avoids possible loss of precision due to cancellation.)

Finally we describe the general case, i.e. $\mathsf{H}_{k-1}\ldots\mathsf{H}_1\mathsf{A}$ has already been constructed with appropriate zeros in columns 1 to $k-1$ and $\boldsymbol{a} \in \mathbb{R}^m$ be the $k^{\text{th}}$ column. We want to use Lemma 6.18 to construct $\mathsf{H}_k$ in (6.18): thus, unless $[0, \ldots, 0, a_k, \ldots, a_m]^{\mathrm{T}}$ is a non-negative multiple of $\boldsymbol{e}_k$, we can choose

$$\boldsymbol{u}_k \equiv [0, \ldots, 0, a_k - \gamma, a_{k+1}, \ldots, a_m]^{\mathrm{T}} \quad \Rightarrow \quad \mathsf{H}_{\boldsymbol{u}_k}\boldsymbol{a} = [a_1, \ldots, a_{k-1}, \gamma, 0, \ldots, 0]^{\mathrm{T}} \,, \tag{6.24}$$

where $\gamma \equiv \left(\sum_{i=k}^m a_i^2\right)^{1/2}$. Hence $\mathsf{H}_k \equiv \mathsf{H}_{\boldsymbol{u}_k}$ and $\mathsf{H}_k\ldots\mathsf{H}_1\mathsf{A}$ now has the required zeros in columns 1 to $k$. Multiplying by $\mathsf{H}_k$ did not alter the first $k-1$ rows and columns of $\mathsf{H}_{k-1}\ldots\mathsf{H}_1\mathsf{A}$. (With inexact arithmetic, and when $a_k > 0$, it is preferable to use the equivalent formula $\left(-\sum_{i=k+1}^m a_i^2\right)/(a_k+\gamma)$ for the $k^{\text{th}}$ component of $\boldsymbol{u}_k$ in (6.24). This again avoids possible loss of precision due to cancellation.)

The following picture (for $\mathsf{A} \in \mathbb{R}^{4\times 3}$) emphasises which components are altered.

$$\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \xrightarrow{\mathsf{H}_1} \begin{bmatrix} \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \end{bmatrix} \xrightarrow{\mathsf{H}_2} \begin{bmatrix} * & * & * \\ 0 & \bullet & \bullet \\ 0 & 0 & \bullet \\ 0 & 0 & \bullet \end{bmatrix} \xrightarrow{\mathsf{H}_3} \begin{bmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & \bullet \\ 0 & 0 & 0 \end{bmatrix}$$

The $\bullet$-elements have changed through the current reflection, while the $*$-elements have remained the same.

We will now state clearly our Householder reflection algorithm for transforming a full rank $\mathsf{A} \in \mathbb{R}^{m\times n}$ into "upper triangular" form.

**Algorithm 6.19.** *For column* 1*, since* $\mathsf{A}$ *has full rank,* $\sum_{i=1}^{m} A_{i,1}^2 \neq 0$*. Unless* $A_{1,1} > 0$ *and* $\sum_{i=2}^{m} A_{i,1}^2 = 0$*, we can construct the Householder reflection* $\mathsf{H}_1$ *so that* $\mathsf{A}^{(1)} \equiv \mathsf{H}_1\mathsf{A}$ *satisfies* $A_{1,1}^{(1)} > 0$ *and* $\{A_{i,1}^{(1)}\}_{i=2}^{m}$ *all zero. Otherwise we simply take* $\mathsf{H}_1 \equiv \mathsf{I}$*.*

*For columns* $2 \leqslant j \leqslant n$*, if* $\mathsf{A}^{(j-1)} \equiv \mathsf{H}_{j-1} \dots \mathsf{H}_1\mathsf{A}$*,* $\mathsf{A}^{(j-1)}$ *must still have full rank and so* $\sum_{i=j}^{m} \left(A_{i,j}^{(j-1)}\right)^2 \neq 0$*. Unless* $A_{j,j}^{(j-1)} > 0$ *and* $\sum_{i=j+1}^{m} \left(A_{i,j}^{(j-1)}\right)^2 = 0$*, we can construct the Householder reflection* $\mathsf{H}_j$ *so that* $\mathsf{A}^{(j)} \equiv \mathsf{H}_j\mathsf{A}^{(j-1)}$ *satisfies* $A_{j,j}^{(j)} > 0$ *and* $\{A_{i,j}^{(j)}\}_{i=j+1}^{m}$ *all zero. (This will not destroy any zeros created in earlier columns.) Otherwise we simply take* $\mathsf{H}_j \equiv \mathsf{I}$*.*

After completion of this algorithm, $\mathsf{A}$ has been overwritten by

$$\mathsf{R} = \mathsf{H}_n \cdots \mathsf{H}_2\mathsf{H}_1\mathsf{A}$$

and so we have a $\mathsf{QR}$ factorisation of the form (6.6) with

$$\mathsf{Q} \equiv \mathsf{H}_1 \cdots \mathsf{H}_n \in \mathbb{R}^{m\times m}\,. \tag{6.25}$$

Since the diagonal elements of $\{R_{i,i}\}_{i=1}^{n}$ are all positive, if we neglect the last $m-n$ columns of $\mathsf{Q}$ and the last $m-n$ zero rows of $\mathsf{R}$ then we end up with exactly the unique skinny $\mathsf{QR}$ factorisation produced by Algorithms 6.8 and 6.13.

*Cost.* Note that for large $m$ we do ***not*** execute explicit matrix multiplication (an $\mathcal{O}(m^2n)$ operation). Instead, to calculate

$$\left(\mathsf{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2}\right)\mathsf{A} = \mathsf{A} - 2\frac{\boldsymbol{u}(\boldsymbol{u}^{\mathrm{T}}\mathsf{A})}{\|\boldsymbol{u}\|^2}\,,$$

first evaluate $\boldsymbol{w}^{\mathrm{T}} = \boldsymbol{u}^{\mathrm{T}}\mathsf{A}$, and then form $\mathsf{A} - \frac{2}{\|\boldsymbol{u}\|^2}\boldsymbol{u}\boldsymbol{w}^{\mathrm{T}}$ (both $\mathcal{O}(mn)$ operations).

*Calculation of* $\mathsf{Q}$*.* If the matrix $\mathsf{Q}$ is required in an explicit form, set $\mathsf{Q} \equiv \mathsf{I}$ initially and, for each successive transformation, replace $\mathsf{Q}$ by

$$\mathsf{Q}\left(\mathsf{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2}\right) = \mathsf{Q} - \frac{2}{\|\boldsymbol{u}\|^2}(\mathsf{Q}\boldsymbol{u})\boldsymbol{u}^{\mathrm{T}},$$

remembering ***not*** to perform explicit matrix multiplication. As in the case of Givens rotations, by the end of the computation, $\Omega = \mathsf{Q}^{\mathrm{T}}$. However, if we are, say, solving a single linear system $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ and require just the vector $\boldsymbol{c} = \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}$ rather than the matrix $\mathsf{Q}$, then we set initially $\boldsymbol{c} \equiv \boldsymbol{b}$ and in each stage replace $\boldsymbol{c}$ by

$$\left(\mathsf{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2}\right)\boldsymbol{c} = \boldsymbol{c} - 2\frac{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{c}}{\|\boldsymbol{u}\|^2}\boldsymbol{u}.$$

*Example.* Calculate the $\mathsf{QR}$ factorization by Householder reflections of (6.11) in §6.4.1, i.e.

$$\mathsf{A} \equiv \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix}$$

as also used in §6.4.2.

First, we do this [the long way] by calculating the $\mathsf{H}_k$:

$$\mathsf{A} = \begin{bmatrix} \boxed{2} & 4 & 5 \\ \boxed{1} & -1 & 1 \\ \boxed{2} & 1 & -1 \end{bmatrix},$$

$$\boldsymbol{u}_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}, \qquad \mathsf{H}_1 = \tfrac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix}, \qquad \mathsf{A}^{(1)} = \mathsf{H}_1\mathsf{A} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & \boxed{0} & 3 \\ 0 & \boxed{3} & 3 \end{bmatrix},$$

$$\boldsymbol{u}_2 = \begin{bmatrix} 0 \\ -3 \\ 3 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}, \qquad \mathsf{H}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad \mathsf{R} = \mathsf{H}_2\mathsf{A}^{(1)} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 0 & 3 \end{bmatrix}.$$

Finally,

$$\mathsf{Q} = (\mathsf{H}_2\mathsf{H}_1)^{\mathrm{T}} = \left( \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tfrac{1}{3} \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix} \right)^{\mathrm{T}} = \tfrac{1}{3} \begin{bmatrix} 2 & 1 & 2 \\ 2 & -2 & -1 \\ 1 & 2 & -2 \end{bmatrix}^{\mathrm{T}} = \tfrac{1}{3} \begin{bmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{bmatrix},$$

so that

$$\mathsf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \underbrace{\tfrac{1}{3} \begin{bmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\begin{bmatrix} 3 & 3 & 3 \\ & 3 & 3 \\ & & 3 \end{bmatrix}}_{\mathsf{R}}.$$

Second, we avoid the explicit calculation of the $\mathsf{H}_k$:

$$\boldsymbol{u}_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}, \quad 2\frac{\boldsymbol{u}_1^{\mathrm{T}}\mathsf{A}}{\|\boldsymbol{u}_1\|^2} = \begin{bmatrix} 1 & -1 & -2 \end{bmatrix}, \qquad \mathsf{A}^{(1)} = \mathsf{A} - 2\frac{\boldsymbol{u}_1(\boldsymbol{u}_1^{\mathrm{T}}\mathsf{A})}{\|\boldsymbol{u}_1\|^2} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 3 & 3 \end{bmatrix},$$

$$\boldsymbol{u}_2 = \begin{bmatrix} 0 \\ -3 \\ 3 \end{bmatrix}, \quad 2\frac{\boldsymbol{u}_2^{\mathrm{T}}\mathsf{A}^{(1)}}{\|\boldsymbol{u}_2\|^2} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \quad \mathsf{R} = \mathsf{A}^{(1)} - 2\frac{\boldsymbol{u}_2(\boldsymbol{u}_2^{\mathrm{T}}\mathsf{A}^{(1)})}{\|\boldsymbol{u}_2\|^2} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 0 & 3 \end{bmatrix}.$$

If we require $\mathsf{Q}$, start with $\Omega = \mathsf{I}$, then

$$\boldsymbol{u}_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}, \quad 2\frac{\boldsymbol{u}_1^{\mathrm{T}}\Omega}{\|\boldsymbol{u}_1\|^2} = \tfrac{1}{3}\begin{bmatrix} -1 & 1 & 2 \end{bmatrix}, \qquad \Omega^{(1)} = \Omega - 2\frac{\boldsymbol{u}_1(\boldsymbol{u}_1^{\mathrm{T}}\Omega)}{\|\boldsymbol{u}_1\|^2} = \tfrac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix},$$

$$\boldsymbol{u}_2 = \begin{bmatrix} 0 \\ -3 \\ 3 \end{bmatrix}, \quad 2\frac{\boldsymbol{u}_2^{\mathrm{T}}\Omega^{(1)}}{\|\boldsymbol{u}_2\|^2} = \tfrac{1}{9}\begin{bmatrix} 1 & -4 & 1 \end{bmatrix}, \quad \mathsf{Q}^{\mathrm{T}} = \Omega^{(1)} - 2\frac{\boldsymbol{u}_2(\boldsymbol{u}_2^{\mathrm{T}}\Omega^{(1)})}{\|\boldsymbol{u}_2\|^2} = \tfrac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 2 & -2 & -1 \\ 1 & 2 & -2 \end{bmatrix}.$$

As before

$$\mathsf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \underbrace{\tfrac{1}{3} \begin{bmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\begin{bmatrix} 3 & 3 & 3 \\ & 3 & 3 \\ & & 3 \end{bmatrix}}_{\mathsf{R}}.$$

We will now generalise Algorithm 6.19 so that it can deal with the rank-deficient case and produce a final $\mathsf{R} \in \mathbb{R}^{m\times n}$ in standard form. (Just like the extension of the Gram–Schmidt process from Algorithm 6.8 to Algorithm 6.10 and the extension of Algorithm 6.13 to Algorithm 6.14, we need to calculate the rank of $\mathsf{A}$.) Unlike the description in Algorithm 6.19, we will now overwrite $\mathsf{A}$ explicitly.

**Algorithm 6.20.** *Set $j = 0$, $k = 0$; where we use $j$ to keep track of the number of columns of $\mathsf{A}$ and $\mathsf{R}$ that have been considered, and $k$ to count the number of linearly independent columns of $\mathsf{A}$ ($k \leqslant j$).*

*Step 1. Increase $j$ by 1.*

- *If $\{A_{i,j}\}_{i=k+1}^{m}$ are all zero, go to Step 2.*
- *If $A_{k+1,j} > 0$ and $\{A_{i,j}\}_{i=k+2}^{m}$ are all zero, we don't need to apply a reflection: simply regard $\mathsf{H}_{k+1} \equiv \mathsf{I}$, increase $k$ by 1 and go to Step 2.*
- *Otherwise, construct and apply the Householder reflection $\mathsf{H}_{k+1}$ so that we obtain $A_{k+1,j} > 0$ and $\{A_{i,j}\}_{i=k+2}^{m}$ all zero. (This will not destroy any zeros created in earlier columns of $\mathsf{A}$.) Increase $k$ by 1 and go to Step 2.*

*Step 2. Terminate if $j = n$, otherwise go to Step 1.*

If the final value of $k$ produced by this algorithm is $p$, then $\mathsf{A} \in \mathbb{R}^{m\times n}$ has been transformed into an "upper triangular"

$$\mathsf{R} = \mathsf{H}_p \cdots \mathsf{H}_2\mathsf{H}_1\mathsf{A} \in \mathbb{R}^{m\times n}$$

whose last $m-p$ rows are zero. Thus we have a $\mathsf{QR}$ factorisation of the form (6.6) with

$$\mathsf{Q} \equiv \mathsf{H}_1 \cdots \mathsf{H}_p \in \mathbb{R}^{m\times m}\,.$$

Since the leading non-zero element in each row of $\mathsf{R}$ is positive, if we neglect the last $m-p$ columns of $\mathsf{Q}$ and the last $m-p$ zero rows of $\mathsf{R}$ then we end up with exactly the unique skinny $\mathsf{QR}$ factorisation produced by Algorithms 6.10 and 6.14.

*Example.* Let us use Householder reflections to find the $\mathsf{QR}$ factorization of (6.14) in §6.4.1, i.e.

$$\mathsf{A} \equiv \begin{bmatrix} 2 & 4 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

as also used in §6.4.2.

$$\begin{array}{lll} k=0 & j=1 & \boldsymbol{u} = [2-\sqrt{6},1,1,0]^{\mathrm{T}} \quad \mathsf{H}_1 \equiv \mathsf{H}_{\boldsymbol{u}} \\ & & \mathsf{A} \to \mathsf{H}_1\mathsf{A} = \begin{bmatrix} \sqrt{6} & 2\sqrt{6} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ k=1 & j=2 & \mathsf{H}_2 \equiv \mathsf{I} \\ k=1 & j=3 & \boldsymbol{u} = [0,-1,0,1]^{\mathrm{T}} \quad \mathsf{H}_3 \equiv \mathsf{H}_{\boldsymbol{u}} \\ & & \mathsf{A} \to \mathsf{H}_3\mathsf{A} = \begin{bmatrix} \sqrt{6} & 2\sqrt{6} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

Hence $\mathsf{A}$ has rank $p=2$ and, since

$$\mathsf{H}_1\mathsf{H}_2\mathsf{H}_3 = \frac{1}{\sqrt{6}} \begin{bmatrix} 2 & 0 & 1 & 1 \\ 1 & 0 & -\sqrt{6}/2-1 & \sqrt{6}/2-1 \\ 1 & 0 & \sqrt{6}/2-1 & -\sqrt{6}/2-1 \\ 0 & \sqrt{6} & 0 & 0 \end{bmatrix},$$

we have

$$\mathsf{A} \equiv \begin{bmatrix} 2 & 4 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \underbrace{\frac{1}{\sqrt{6}} \begin{bmatrix} 2 & 0 & 1 & 1 \\ 1 & 0 & -\sqrt{6}/2-1 & \sqrt{6}/2-1 \\ 1 & 0 & \sqrt{6}/2-1 & -\sqrt{6}/2-1 \\ 0 & \sqrt{6} & 0 & 0 \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\begin{bmatrix} \sqrt{6} & 2\sqrt{6} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathsf{R}}$$

*Givens or Householder?* If $\mathsf{A}$ is dense, it is in general more convenient to use Householder transformations. Givens rotations come into their own, however, when $\mathsf{A}$ has many leading zeros in its rows. In an extreme case, if an $n \times n$ matrix $\mathsf{A}$ consists of zeros underneath the first sub-diagonal, they can be 'rotated away' in $(n-1)$ Givens rotations, at the cost of just $\mathcal{O}(n^2)$ operations.

## 6.5 Solving least squares problems with the QR factorisation

Suppose that $\mathsf{A} \in \mathbb{R}^{m\times n}$ has full rank and that we have a QR factorisation of $\mathsf{A}$ as in (6.6). As stated in (6.9), this means that our least squares problem (6.1) can be simplified to

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\mathsf{A}\boldsymbol{x}-\boldsymbol{b}\| = \min_{\boldsymbol{x}\in\mathbb{R}^n} \|\mathsf{R}\boldsymbol{x}-\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\|\,.$$

Since the last $m-n$ rows of $\mathsf{R}$ are zero, it is easy to solve this simplified minimisation problem: i.e. if $\widetilde{\mathsf{R}} \in \mathbb{R}^{n\times n}$ contains the first $n$ rows of $\mathsf{R}$ and $\widetilde{\mathsf{Q}} \in \mathbb{R}^{m\times n}$ contains the first $n$ columns of $\mathsf{Q}$, then the unique solution of our least squares problem is that $\boldsymbol{x}^* \in \mathbb{R}^n$ which satisfies the $n\times n$ non-singular upper triangular system

$$\widetilde{\mathsf{R}}\boldsymbol{x}^* = \widetilde{\mathsf{Q}}^{\mathrm{T}}\boldsymbol{b}\,. \tag{6.26}$$

(This can of course be obtained by back substitution!) The size of the *residual* for our least squares problem is just the Euclidean norm of the neglected last $m-n$ components: i.e.

$$\|\mathsf{R}\boldsymbol{x}^*-\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\| = \left(\sum_{i=n+1}^{m}\left[\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\right]_i^2\right)^{1/2}\,.$$

Note that $\mathsf{A} = \widetilde{\mathsf{Q}}\widetilde{\mathsf{R}}$ is exactly the skinny QR factorisation of $\mathsf{A}$ described in (6.8), so this skinny factorisation is sufficient to solve our least squares problem. (Although the norm of the residual would have to be calculated in a different way!)

Now suppose that $\mathsf{A} \in \mathbb{R}^{m\times n}$ has rank $p<n$, so that our QR factorisation of $\mathsf{A}$ as in (6.6) becomes

$$m>n\left\{\underbrace{\begin{bmatrix} A_{1,1} & \cdots & A_{1,n}\\ \vdots & & \vdots\\ \vdots & & \vdots\\ A_{n,1} & \cdots & A_{n,n}\\ \vdots & & \vdots\\ \vdots & & \vdots\\ A_{m,1} & \cdots & A_{m,n}\end{bmatrix}}_{n} = \underbrace{\begin{bmatrix} Q_{1,1} & \cdots & Q_{1,m}\\ \vdots & & \vdots\\ \vdots & & \vdots\\ Q_{n,1} & \cdots & Q_{n,m}\\ \vdots & & \vdots\\ \vdots & & \vdots\\ Q_{m,1} & \cdots & Q_{m,m}\end{bmatrix}}_{m} \underbrace{\begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,p} & \cdots & R_{1,n}\\ & R_{2,2} & & \vdots & & \vdots\\ & & \ddots & \vdots & & \vdots\\ & & & R_{p,p} & \cdots & R_{p,n}\\ 0 & \cdots & \cdots & 0 & \cdots & 0\\ \vdots & & & \vdots & & \vdots\\ 0 & \cdots & \cdots & 0 & \cdots & 0\end{bmatrix}}_{n}\right\} m \tag{6.27}$$

where $\mathsf{R}$ is in standard form as described in Definition 6.11. As stated in (6.9), this means that our least squares problem again simplifies to

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\mathsf{A}\boldsymbol{x}-\boldsymbol{b}\| = \min_{\boldsymbol{x}\in\mathbb{R}^n} \|\mathsf{R}\boldsymbol{x}-\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\|\,.$$

Since the last $m-p$ rows of $\mathsf{R}$ are zero, this simplified minimisation problem again reduces to

$$\widetilde{\mathsf{R}}\boldsymbol{x} = \widetilde{\mathsf{Q}}^{\mathrm{T}}\boldsymbol{b}\,, \tag{6.28}$$

where $\widetilde{\mathsf{R}} \in \mathbb{R}^{p\times n}$ contains the first $p$ rows of $\mathsf{R}$ and $\widetilde{\mathsf{Q}} \in \mathbb{R}^{m\times p}$ contains the first $p$ columns of $\mathsf{Q}$. Now, however, (6.28) is an under-determined system with an infinite number of solutions depending on $n-p$ free parameters. Nevertheless, since $\mathsf{R}$ is in standard form, we can easily describe these solutions.

Let $\{j_1,\ldots,j_p\}$ be the column numbers for the leading non-zero elements of rows $1\to p$ of $\mathsf{R}$: So $\{j_i\}_{i=1}^p$ is a subsequence of $\{1,\ldots,n\}$. For any $\boldsymbol{x}\in\mathbb{R}^n$, we can write

$$\widetilde{\mathsf{R}}\boldsymbol{x} = \widetilde{\mathsf{R}}_1\boldsymbol{x}_1 + \widetilde{\mathsf{R}}_2\boldsymbol{x}_2 \qquad \boldsymbol{x}_1\in\mathbb{R}^p, \boldsymbol{x}_2\in\mathbb{R}^{n-p}, \widetilde{\mathsf{R}}_1\in\mathbb{R}^{p\times p}, \widetilde{\mathsf{R}}_2\in\mathbb{R}^{p\times(n-p)}\,,$$

where $\boldsymbol{x}_1$ contains components $\{j_1,\ldots,j_p\}$ of $\boldsymbol{x}$ and $\boldsymbol{x}_2$ contains the other $n-p$ components. Hence (6.28) becomes

$$\widetilde{\mathsf{R}}_1\boldsymbol{x}_1 = \widetilde{\mathsf{Q}}^{\mathrm{T}}\boldsymbol{b} - \widetilde{\mathsf{R}}_2\boldsymbol{x}_2\ :$$

where

- the components of $\boldsymbol{x}_2$ are $n-p$ free parameters;

- the components of $\boldsymbol{x}_1$ can be uniquely solved for, because $\widetilde{\mathsf{R}}_1 \in \mathbb{R}^{p\times p}$ is a non-singular upper triangular matrix.

If we denote any of these solutions by $\boldsymbol{x}^* \in \mathbb{R}^n$, then the residual for our least squares problem is exactly the same: i.e.

$$\|\mathsf{R}\boldsymbol{x}^* - \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\| = \left(\sum_{i=p+1}^{m} \left[\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\right]_i^2\right)^{1/2} .$$

Note that $\mathsf{A} = \widetilde{\mathsf{Q}}\widetilde{\mathsf{R}}$ is precisely the skinny QR factorisation of $\mathsf{A}$ described in (6.13), so this skinny factorisation is sufficient to solve our rank deficient least squares problem.

### 6.5.1 Examples

(i) Find $\boldsymbol{x} \in \mathbb{R}^3$ that minimizes $\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|$, where

$$\mathsf{A} = \tfrac{1}{2}\begin{bmatrix} 1 & 3 & 6 \\ 1 & 1 & 2 \\ 1 & 3 & 4 \\ 1 & 1 & 0 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

The QR-factorization of $\mathsf{A}$ is

$$\mathsf{A} = \underbrace{\tfrac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}}_{\mathsf{Q}} \times \underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathsf{R}},$$

so $\mathsf{A}$ has full rank and the least squares problem has a unique solution.

a) A simple approach (suitable for exact arithmetic) is to use the QR factorisation to simplify the normal equations: i.e.

$$\mathsf{A}^{\mathrm{T}}(\mathsf{A}\boldsymbol{x} - \boldsymbol{b}) = \boldsymbol{0} \quad \Rightarrow \quad \mathsf{R}^{\mathrm{T}}(\mathsf{R}\boldsymbol{x} - \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}) = \boldsymbol{0}\,.$$

We can then decompose the simplified equation into

$$\mathsf{R}^{\mathrm{T}}\boldsymbol{y} = \boldsymbol{0} \quad \text{for } \boldsymbol{y} \in \mathbb{R}^4 \qquad \text{and} \qquad \mathsf{R}\boldsymbol{x} = \mathsf{Q}^{\mathrm{T}}\boldsymbol{b} + \boldsymbol{y} \quad \text{for } \boldsymbol{x} \in \mathbb{R}^3\,.$$

From the former, we obtain

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}}_{\mathsf{R}^{\mathrm{T}}} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = 0\,, \quad \text{with solution} \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \lambda \end{bmatrix}$$

where $\lambda \in \mathbb{R}$. From the latter, we obtain

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathsf{R}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \underbrace{\tfrac{1}{2}\begin{bmatrix} 3 \\ 1 \\ 1 \\ -1 \end{bmatrix}}_{\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \lambda \end{bmatrix},$$

which has solution

$$\lambda = \tfrac{1}{2}\,, \quad \boldsymbol{x}^* = \tfrac{1}{2}\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}.$$

The size of the residual is the norm of the vector formed by the bottom $(m-n)$ components of the right-hand side $\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}$:

$$\|\mathsf{A}\boldsymbol{x}^* - \boldsymbol{b}\| = \|\mathsf{R}\boldsymbol{x}^* - \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\| = \|\boldsymbol{y}\| = \tfrac{1}{2}\,.$$

b) From (6.26), we use

$$\widetilde{\mathsf{R}}\boldsymbol{x} = \widetilde{\mathsf{Q}}^{\mathrm{T}}\boldsymbol{b} \quad \Rightarrow \quad \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \tfrac{1}{2} \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{x}^* = \tfrac{1}{2} \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} .$$

(ii) Find $\boldsymbol{x} \in \mathbb{R}^3$ that minimizes $\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|$, where

$$\mathsf{A} \equiv \begin{bmatrix} 2 & 4 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{b} \equiv \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} .$$

Slightly different $\mathsf{QR}$ factorizations of $\mathsf{A}$ have been computed in §6.4.1, §6.4.2 and §6.4.3; but these all produce the same skinny factorisation

$$\mathsf{A} = \underbrace{\frac{1}{\sqrt{6}} \begin{bmatrix} 2 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & \sqrt{6} \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\sqrt{6} \begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 1/\sqrt{6} \end{bmatrix}}_{\mathsf{R}} .$$

So $\mathsf{A}$ has rank $p = 2$ and there is a 1-parameter family of solutions obtained from

$$\underbrace{\begin{bmatrix} \sqrt{6} & 2\sqrt{6} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathsf{R}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 4/\sqrt{6} \\ 1 \end{bmatrix}}_{\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}} .$$

Hence we write

$$\begin{bmatrix} \sqrt{6} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4/\sqrt{6} \\ 1 \end{bmatrix} - x_2 \begin{bmatrix} 2\sqrt{6} \\ 0 \end{bmatrix} ,$$

where $x_2$ is regarded as a free parameter, and arrive at

$$\boldsymbol{x} = \begin{bmatrix} \frac{2}{3} - 2\alpha \\ \alpha \\ 1 \end{bmatrix} \qquad \text{for arbitrary } \alpha \in \mathbb{R}.$$

The residual for any of these solutions is

$$\mathsf{A}\boldsymbol{x} - \boldsymbol{b} = \frac{1}{3} \begin{bmatrix} 1 \\ -1 \\ -1 \\ 0 \end{bmatrix}$$

and so $\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\| = 1/\sqrt{3}$.