Mathematical Tripos: IB Numerical Analysis

Contents

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 `S.J.Cowley@damtp.cam.ac.uk`. If you want to read ahead:

- Arieh Iserles' handouts (an excellent summary in 32 pages instead of 84) 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 $\frac{2}{3}$ 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:
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.

Questionnaire Returns. Questionnaire results from previous years are available at

- `http://tinyurl.com/NA-IB-2011-Results`,
- `http://tinyurl.com/NA-IB-2012-Results`,
- `http://tinyurl.com/NA-IB-2013-Results`.

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 $O((n+1)!)$ operations by Cramer's rule, but only $O(n^3)$ 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

We start with *interpolation*. Suppose that we have a number of values at data points. *Curve fitting* is when we try to construct a function which closely fits those values. *Interpolation* is a specific case of curve fitting, in which the function must go exactly through the values at the data points.[2]

1.1 The interpolation problem

Let $[a, b]$ denote some real interval. Suppose that we are given $n+1$ distinct points, $x_0, x_1, \dots, x_n$, in $[a, b]$, together with real numbers $f_0, f_1, \dots, f_n$. We seek a function $p : \mathbb{R} \to \mathbb{R}$ such that

$$p(x_i) = f_i, \quad i = 0, 1, \dots, n.$$

Such a function is called an *interpolant*.

We denote by $\mathbb{P}_n[x]$ the *linear space* of all real polynomials of degree at most n, and we observe that each polynomial in $\mathbb{P}_n[x]$ is uniquely defined by its $n+1$ coefficients. Hence, such polynomials have $n+1$ degrees of freedom, while interpolation at $x_0, x_1, \dots, x_n$ constitutes $n+1$ conditions. This, intuitively, justifies seeking an interpolant $p \in \mathbb{P}_n[x]$.

Remark. It is not uncommon for the $f_0, f_1, \dots, f_n$ to be the values at $x_0, x_1, \dots, x_n$ of a real valued continuous function $f : [a, b] \to \mathbb{R}$ defined on $[a, b]$.

1.2 The Lagrange formula

Although, in principle, we may solve a linear problem with $n+1$ unknowns to determine a polynomial interpolant in $O(n^3)$ operations, this can be calculated in only $O(n^2)$ operations by using the explicit *Lagrange formula:*

$$p(x) = \sum_{k=0}^{n} f_k \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}, \qquad x \in \mathbb{R}. \tag{1.1a}$$

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

Proof. First, define the *Lagrange cardinal polynomials for the points* $x_0, x_1, \dots, x_n$:

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

Each ℓ_k is the product of n linear factors, hence $\ell_k \in \mathbb{P}_n[x]$ and, from (1.1a), $p \in \mathbb{P}_n[x]$. Further, it is straightforward to verify that $\ell_k(x_k) = 1$ and $\ell_k(x_j) = 0$ for $j \neq k$, i.e. $\ell_k(x_j) = \delta_{kj}$. Hence

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

and thus p is an interpolant.

In order to prove uniqueness, suppose that both $p \in \mathbb{P}_n[x]$ and $q \in \mathbb{P}_n[x]$ interpolate the same $n+1$ data. Then the polynomial $r = p - q$ is of degree n and vanishes at $n+1$ distinct points. But the only n^{th}-degree polynomial with $\geqslant n+1$ zeros is the zero polynomial. Therefore $p - q \equiv 0$ and the interpolating polynomial is unique. □

[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/`.

Remarks.

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

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

Then, in the expression (1.1b) for ℓ_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 (1.1a) and (1.2b) for the interpolating polynomials are often the appropriate forms to use when we wish to manipulate the interpolation polynomial as part of a larger mathematical expression. We will see an example 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* which has an *adaptive*, or *recurrent*, nature, i.e. if an extra data point is added, then the new interpolant, say p_{n+1}, can be constructed from the existing interpolant, p_n (rather than starting again from scratch).

1.3 The Newton interpolation formula

Our aim is to find an alternative representation of the interpolating polynomial. We again suppose that the f_i, $i = 0, 1, \ldots, n$, are given, and seek $p \in \mathbb{P}_n[x]$ such that $p(x_i) = f_i$, $i = 0, \ldots, n$. For $k = 0, 1, ..., n$, let $p_k \in \mathbb{P}_k$ be the polynomial interpolant to f on $x_0, ..., x_k$, then Newton's formula is analogous to the identity

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

In order to construct the formula we note that p_{k-1} and p_k interpolate the same values f_i for $i \leqslant k-1$, and hence their difference is a polynomial of degree k that vanishes at the k points $x_0, ..., x_{k-1}$. Thus

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

for some constant A_k, where we note that A_k is equal to the *leading* coefficient of p_k. It follows that $p = p_n$ can be built step by step as one constructs the sequence $(p_0, p_1, ...)$, with p_k obtained from p_{k-1} by the addition of the term on the right-hand side of (1.4), so that finally

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

What remains to be identified is an effective means to calculate the A_k.

1.3.1 Divided differences: a definition

Definition 1.2 ($C^s[a,b]$)**.** Let $[a,b]$ be a closed interval of $\mathbb{R}$. We denote by $C[a,b]$ the 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.

Definition 1.3 (Divided difference)**.** Given $f \in C[a,b]$ and $k+1$ distinct points $(x_i)_{i=0}^{k} \in [a,b]$, the *divided difference* $f[x_0, ..., x_k]$ is the leading coefficient of the polynomial $p_k \in \mathbb{P}_k$ which interpolates f at these points. We say that this divided difference is of *degree*, or *order*, k.

Remarks.

(i) By definition
$$A_k \equiv f[x_0, ..., x_k]. \tag{1.6}$$

(ii) A divided difference $f[x_0, \ldots, x_k]$ of order k is a real number that is derived from $(k+1)$ values of a function $f : \mathbb{R} \to \mathbb{R}$.

(iii) A divided difference $f[x_0, \ldots, x_k]$ is a symmetric function of the variables $[x_0, ..., x_k]$.

(iv) $f[x_0]$ is the coefficient of x^0 in the polynomial of degree 0 (i.e. a constant) that interpolates $f(x_0)$, hence
$$f[x_0] = f(x_0). \tag{1.7a}$$
More generally
$$f[\pounds] = f(\pounds)\,, \quad f[¥] = f(¥) \quad \text{and} \quad f[x_i] = f(x_i)\,. \tag{1.7b}$$

(v) If $f(x) = x^m$ and $k \geqslant m$, then
$$f[x_0, ..., x_k] = \delta_{km}. \tag{1.8}$$

1.3.2 The Newton formula for the interpolating polynomial

The following theorem is a consequence of the definition of a divided difference, i.e. (1.5) and (1.6).

Theorem 1.4 (Newton formula). *Given $n + 1$ distinct points $(x_i)_{i=0}^n$, let $p_n \in \mathbb{P}_n$ be the polynomial that interpolates f at these points. Then it may be written in the Newton form*

$$p_n(x) \;=\; f[x_0] + f[x_0, x_1]\,(x - x_0) + \cdots + f[x_0, x_1, \ldots, x_n]\,(x - x_0)(x - x_1) \cdots (x - x_{n-1})\,, \tag{1.9a}$$

or, more compactly (with a slight abuse of notation when $k = 0$)

$$p_n(x) = \sum_{k=0}^{n} f[x_0, ..., x_k] \prod_{i=0}^{k-1} (x - x_i). \tag{1.9b}$$

Remark. For this formula to be of any use, we need an expression for $f[x_0, ..., x_k]$. One such can be derived from the Lagrange formula (1.1a), or equivalently (1.2b), by identifying the leading coefficient of p. We conclude

$$f[x_0, ..., x_n] \;=\; \sum_{j=0}^{n} f(x_j) \prod_{\substack{i=0\\ i\neq j}}^{n} \frac{1}{x_j - x_i} \;=\; \sum_{j=0}^{n} \frac{f(x_j)}{\omega'(x_j)}, \quad \text{where} \quad \omega(x) = \prod_{i=0}^{n} (x - x_i). \tag{1.10}$$

However, like the Lagrange form itself, this expression takes $O(n^2)$ operations to evaluate. A better way to calculate divided differences is to again use an adaptive (or recurrent) approach.

1.3.3 Recurrence relations for divided differences

Theorem 1.5 (Recurrence relation). *Suppose that $x_0, x_1, \ldots, x_k$ are distinct, where $k \geqslant 1$, then*

$$f[x_0, ..., x_k] = \frac{f[x_1, ..., x_k] - f[x_0, ..., x_{k-1}]}{x_k - x_0} \tag{1.11}$$

Proof. Let $q_0, q_1 \in \mathbb{P}_{k-1}$ be the polynomials such that q_0 interpolates f on $(x_0, x_1, \ldots, x_{k-1})$
q_1 interpolates f on $(x_1, \ldots, x_{k-1}, x_k)$
and consider the polynomial

$$p(x) = \frac{x - x_0}{x_k - x_0}\, q_1(x) + \frac{x_k - x}{x_k - x_0}\, q_0(x)\,, \qquad p \in \mathbb{P}_k\,. \tag{1.12}$$

We note that

$$p(x_0) = f(x_0)\,, \quad p(x_k) = f(x_k)\,, \quad \text{and} \quad p(x_i) = f(x_i) \quad \text{for} \quad i = 1, \dots, k-1.$$

Hence p is the k-degree interpolating polynomial of $\{f(x_i) : i = 0, 1, \dots, k\}$. Moreover, from (1.12), the leading coefficient of p, i.e. $f[x_0, ..., x_k]$, is equal to the difference of those of q_1 and q_0, i.e. $f[x_1, ..., x_k]$ and $f[x_0, ..., x_{k-1}]$ respectively, divided by $x_k - x_0$; this is as required to prove (1.11). □

Method 1.6. Recalling from (1.7b) that $f[x_i] = f(x_i)$, the recursive formula (1.11) allows for rapid evaluation of the *divided difference table,* in the following manner:

$$\begin{tikzcd}
x_i & f[*] = f(*) & f[*,*] & f[*,*,*] & \dots & f[*,*,\dots,*] \\
x_0 \arrow[r] & f[x_0] \arrow[dr] & & & & \\
& & f[x_0,x_1] \arrow[dr] & & & \\
x_1 \arrow[r] & f[x_1] \arrow[ur] \arrow[dr] & & f[x_0,x_1,x_2] & \ddots & \\
& & f[x_1,x_2] \arrow[ur] \arrow[dr] & & & \\
x_2 \arrow[r] & f[x_2] \arrow[ur] \arrow[dr] & & f[x_1,x_2,x_3] & \vdots & \\
& & f[x_2,x_3] \arrow[ur] & \ddots & & \\
x_3 \arrow[r] & f[x_3] \arrow[ur] & \ddots & & & \\
\vdots & & & & \vdots & f[x_0,x_1,\dots,x_n] \\
x_{n-1} \arrow[r] & f[x_{n-1}] \arrow[dr] & \ddots & f[x_{n-2},x_{n-1},x_n] & \dots & \\
& & f[x_{n-1},x_n] \arrow[ur] & & & \\
x_n \arrow[r] & f[x_n] \arrow[ur] & & & &
\end{tikzcd}$$

Table 1: *The divided difference table*

Remarks

(i) The table can be evaluated in $\mathcal{O}(n^2)$ operations and the outcome is the numbers $\{f[x_0, \dots, x_k]\}_{k=0}^n$ at the head of the columns which can be used in the Newton form (1.9b).

(ii) While it is usual for the points $\{x_i : i = 0, 1, \dots, n\}$ 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) Suppose an extra interpolation point, say x_{n+1}, is added. Then in order to determine $f[x_0, \dots, x_{n+1}]$ only an extra diagonal of the divided difference table need be calculated in $\mathcal{O}(n)$ operations.

1/12
1/13
1/14

1.3.4 Horner's scheme

We note that the Newton interpolation formula (1.9b) requires only the top row of the divided differences in Table 1. Once these differences have been calculated, and stored together with the interpolation points x_i, the Newton formula has a potential speed advantage over the Lagrange formula if its evaluation at a given point x is calculated using *Horner's scheme.* Based on the observation that

$$c_2 x^2 + c_1 x + c_0 = (c_2 x + c_1)x + c_0\,,$$

write

$$\begin{aligned} p_n(x) \;=\; \Big\{ \dots \big\{ \{ f[x_0, \dots, x_n](x - x_{n-1}) + f[x_0, \dots, x_{n-1}] \}(x - x_{n-2}) \\ + f[x_0, \dots, x_{n-2}] \big\}(x - x_{n-3}) + \cdots \Big\}(x - x_0) + f[x_0]. \end{aligned}$$

The calculation is arranged as follows. Let σ be $f[x_0, x_1, \dots, x_n]$ initially. Then for $k = n-1, n-2, \dots, 0$, overwrite σ by the quantity

$$\sigma \;\leftarrow\; \sigma\,(x - x_k) + f[x_0, x_1, \dots, x_k]\,. \tag{1.13}$$

$p_n(x)$ is the final value of σ, and it is computed in only $\mathcal{O}(n)$ operations.

Remarks

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

(ii) An advantage of Newton's formula over Lagrange's formula is now evident. Suppose an extra interpolation point, say x_{n+1}, is added in order to improve accuracy. Then in order to calculate the extra coefficient in (1.9b), only an extra diagonal of the divided difference table need be calculated in $\mathcal{O}(n)$ operations, while to evaluate Newton's formula at a point 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 $p \in \mathbb{P}_3$ 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:

$$\begin{tikzcd}
0 & \mathbf{-3} \arrow[dr] & & & \\
 & & \frac{(-3)-(-3)}{1-0} = \mathbf{0} \arrow[dr] & & \\
1 & -3 \arrow[ur] \arrow[dr] & & \frac{2-0}{2-0} = \mathbf{1} \arrow[dr] & \\
 & & \frac{(-1)-(-3)}{2-1} = 2 \arrow[ur] \arrow[dr] & & \frac{4-1}{3-0} = \mathbf{1} \\
2 & -1 \arrow[ur] \arrow[dr] & & \frac{10-2}{3-1} = 4 \arrow[ur] & \\
 & & \frac{9-(-1)}{3-2} = 10 \arrow[ur] & & \\
3 & 9 \arrow[ur] & & &
\end{tikzcd}$$

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:

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

Exercise: Add a 5th 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

The following theorem shows that a divided difference can be regarded as a constant multiple of a derivative of degree n. However, first we need a lemma.

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

Proof. Rolle's theorem states that if a function $\phi \in C^1[a,b]$ vanishes at two distinct points in $[a,b]$ then its derivative ϕ' is zero at an intermediate point. So, we deduce that g' vanishes at least at $(n-1)+\ell$ distinct points. Next, applying Rolle to g', we conclude that g'' vanishes at $(n-2)+\ell$ points, and so on. □

Theorem 1.8. *Let $[\bar{a},\bar{b}]$ be the shortest interval that contains $x_0, x_1, \ldots, x_n$ and let $f \in C^n[\bar{a},\bar{b}]$. Then there exists $\xi \in [\bar{a},\bar{b}]$ such that*

$$f[x_0, x_1, \ldots, x_n] = \tfrac{1}{n!} f^{(n)}(\xi). \tag{1.14a}$$

Proof. Let $p \in \mathbb{P}_n[x]$ be the interpolating polynomial of $\{f(x_i) : i=0,1,\ldots,n\}$. Then the *error function* $(f-p)$ has at least $(n+1)$ zeros in $[\bar{a},\bar{b}]$. It follows from applying Lemma 1.7, that the n-th derivative $(f^{(n)}-p^{(n)})$, must vanish at some $\xi \in [\bar{a},\bar{b}]$, i.e.,

$$f^{(n)}(\xi) = p^{(n)}(\xi)$$

Moreover, since $p \in \mathbb{P}_n$, it follows that $p(x) = A_n x^n +$ (lower order terms) for some constant A_n, and hence (for any ξ)

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

The result (1.14a) follows. □

Application. A method of estimating a derivative, say $f^{(n)}(\xi)$ where ξ is now given, is to let the points $\{x_i : i=0,1,\ldots,n\}$ be suitably close to ξ, and to make the approximation

$$f^{(n)}(\xi) \approx n!\, f[x_0, x_1, \ldots, x_n]\,. \tag{1.14b}$$

However, a drawback is that if one achieves good accuracy in theory by picking closely spaced interpolation points, then 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. 1/05

1.6 Error bounds for polynomial interpolation

As noted earlier, it is important to understand the error of our approximations. Here we study the *interpolation error*

$$e_n(x) = f(x) - p_n(x), \quad p_n \in \mathbb{P}_n, \tag{1.15}$$

restricted to the class of differentiable functions $f \in C^k[a,b]$ that possess, say, k continuous derivatives on the interval $[a,b]$.

The following theorem shows the error in an interpolant to be 'like the next term' in the Newton form.

Theorem 1.9. *Let $p_n \in \mathbb{P}_n$ interpolate $f \in C[a,b]$ at $n+1$ distinct points $x_0, ..., x_n$. Then for any $x \notin (x_i)_{i=0}^n$*

$$f(x) - p_n(x) = f[x_0, ..., x_n, x]\,\omega(x) \tag{1.16}$$

Proof. Given $x_0, .., x_n$, let $\bar{x} = x_{n+1}$ be any other point. Then, from the recurrence relation (1.4) and the definition (1.6), the corresponding polynomials p_n and p_{n+1} are related by

$$p_{n+1}(x) = p_n(x) + f[x_0, ..., x_n, \bar{x}]\,\omega(x), \qquad \omega(x) = \prod_{i=0}^{n} (x - x_i)\,.$$

In particular, putting $x = \bar{x}$, and noticing that $p_{n+1}(\bar{x}) = f(\bar{x})$, we obtain, as in (1.16),

$$f(\bar{x}) = p_n(\bar{x}) + f[x_0, ..., x_n, \bar{x}]\,\omega(\bar{x})\,.$$ □

Remark. We cannot evaluate the right-hand side of (1.16) without knowing the number $f(x)$; however, as we now show, we can relate it to the $(n+1)^{\text{st}}$ derivative of f.

Theorem 1.10. *Given $f \in C^{n+1}[a,b]$, let $p \in \mathbb{P}_n[x]$ interpolate the values $f(x_i)$, $i = 0,1,\ldots,n$, where $x_0,\ldots,x_n \in [a,b]$ are distinct. Then for every $x \in [a,b]$ there exists $\xi \in [a,b]$ such that*

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (x - x_i). \tag{1.17}$$

Proof. From Theorem 1.9 and (1.16), and Theorem 1.8 and (1.14a),

$$\begin{aligned} f(x) - p(x) &= f[x_0, ..., x_n, x] \prod_{i=0}^{n} (x - x_i) \\ &= \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (x - x_i) \end{aligned}$$

where $\xi \in [a,b]$ since $x_0,\ldots,x_n,x \in [a,b]$. □

Alternative Proof. (*Unlectured, but useful for the Examples Sheet.*) The formula (1.17) 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 ϕ 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, ϕ 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 ϕ' has at least $n+1$ distinct zeros in $[a,b]$, that ϕ'' 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 $\phi^{(n+1)}(\xi) = 0$ for some $\xi \in [a,b]$. Hence

$$0 = \phi^{(n+1)}(\xi) = [f^{(n+1)}(\xi) - p^{(n+1)}(\xi)] \prod_{i=0}^{n} (x - x_i) - [f(x) - p(x)] \frac{\mathrm{d}^{n+1}}{\mathrm{d}t^{n+1}} \prod_{i=0}^{n} (\xi - x_i).$$

Since

$$p^{(n+1)} \equiv 0 \quad \text{and} \quad \frac{\mathrm{d}^{n+1}}{\mathrm{d}t^{n+1}} \prod_{i=0}^{n} (t - x_i) \equiv (n+1)!\,,$$

we obtain (1.17). □

Remark. The *equality* (1.17) with the value $f^{(n+1)}(\xi)$ for *some* ξ is of hardly any use. However, often one has a bound for $f^{(n+1)}$ in terms of some *norm*, e.g., the L_∞-norm (the *max*-norm)

$$\|g\|_\infty \equiv \|g\|_{L_\infty[a,b]} = \max_{t\in[a,b]} |g(t)|\,.$$

In such a case, it follows from (1.17), and the definition of the nodal polynomial (1.2a), that

$$|f(x) - p(x)| \leqslant \frac{1}{(n+1)!} |\omega(x)|\, \|f^{(n+1)}\|_\infty\,. \tag{1.18a}$$

If we want to find the maximal error over the interval, then maximizing over $x \in [a,b]$ we get yet one more error bound for polynomial interpolation

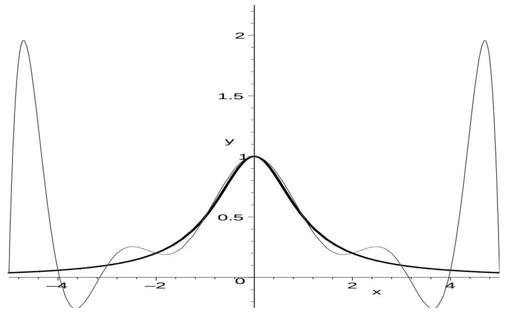
$$\|f - p_\Delta\|_\infty \leqslant \frac{1}{(n+1)!} \|\omega_\Delta\|_\infty\, \|f^{(n+1)}\|_\infty \tag{1.18b}$$

Here we put the lower index in ω_Δ in order to emphasize dependence of $\omega(x) = \prod_{i=0}^{n}(x - x_i)$ on the sequence of interpolating points $\Delta = (x_i)_{i=0}^{n}$. The choice of Δ can make a big difference!

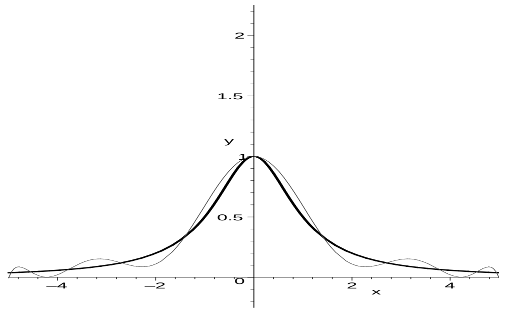
Runge's Example. We interpolate

$$f(x)=\frac{1}{1+x^2}, \quad x\in[-5,5], \tag{1.19}$$

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 $\{-5+j\}_{j=0}^{10}$.



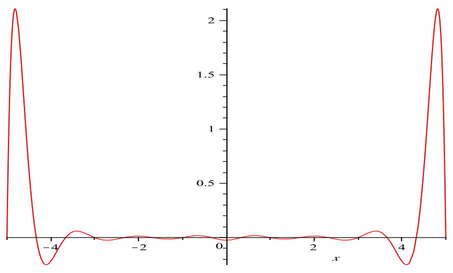
Interpolation at Chebyshev knots $\{-5\cos\frac{2j+1}{22}\pi\}_{j=0}^{10}$.

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 (1.17). 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 (1.19) at uniform knots: $n=15$.

x	$f(x)-p(x)$	$\prod_{i=0}^n(x-x_i)$
0.75	3.2×10^{-3}	-2.5×10^{6}
1.75	7.7×10^{-3}	-6.6×10^{6}
2.75	3.6×10^{-2}	-4.1×10^{7}
3.75	5.1×10^{-1}	-7.6×10^{8}
4.75	$4.0\times10^{+2}$	-7.3×10^{10}

Errors in interpolating (1.19) 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 $\max_{x\in[-5,5]}|\prod_{i=0}^n(x-x_i)|$.

Definition 1.11. The *Chebyshev*[3] polynomial of degree n on $[-1,1]$ is defined by

$$T_n(x)=\cos n\theta, \quad x=\cos\theta, \quad \theta\in[0,\pi]. \tag{1.20}$$

Properties of $T_n(x)$ *on* $[-1,1]$.

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

$$\|T_n\|_\infty=1, \quad T_n(X_k)=(-1)^k, \quad X_k=\cos\frac{\pi k}{n}, \quad k=0,1,\ldots,n; \tag{1.21a}$$

(ii) T_n has n distinct zeros:

$$T_n(x_k)=0, \quad x_k=\cos\frac{2k-1}{2n}\pi, \quad k=1,2,\ldots,n. \tag{1.21b}$$

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

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

$$T_0(x) \equiv 1, \quad T_1(x) = x, \tag{1.22a}$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geqslant 1. \tag{1.22b}$$

Proof. Expressions (1.22a) are straightforward, while the recurrence follows from the equality

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

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

Remark. It follows from (1.22a) and (1.22b) that T_n is an algebraic polynomial of degree n, with the leading coefficient 2^{n-1} (for $n \geqslant 1$).

Theorem 1.13. *On the interval $[-1,1]$, among all polynomials of degree n with leading coefficient equal to one, the Chebyshev polynomial γT_n deviates least from zero, i.e.,*

$$\inf_{(a_i)} \left\|x^n + a_{n-1}x^{n-1} + \cdots + a_0\right\|_\infty = \gamma\,\|T_n\|_\infty, \quad \gamma = \tfrac{1}{2^{n-1}}\,. \tag{1.23}$$

Proof. Suppose there is a polynomial $q_n(x) = x^n + \cdots + a_0$ such that $\|q\|_\infty < \gamma$, and set

$$r = \gamma\, T_n - q_n.$$

The leading coefficient of both q_n and γT_n is one, thus r is of degree at most $n-1$.

On the other hand, at the points $X_k = \cos\frac{\pi k}{n}$, the Chebyshev polynomial γT_n takes the values $\pm\gamma$ alternatively, while by assumption $|q_n(X_k)| < \gamma$. Hence r alternates in sign at these points, and therefore it has a zero in each of n intervals (X_k, X_{k+1}), i.e. at least n zeros in the interval $[-1,1]$, a contradiction to $r \in \mathbb{P}_{n-1}$. □

Corollary 1.14. *For $\Delta = (x_k)_{k=0}^n \subset [-1,1]$, let $\omega_\Delta(x) = \prod_{k=0}^n (x - x_k) \in \mathbb{P}_{n+1}$. Then, for all n, we have*

$$\inf_\Delta \|\omega_\Delta\|_\infty = \|\omega_{\Delta_*}\|_\infty = \frac{1}{2^n}\,, \tag{1.24a}$$

where

$$\Delta_* = (x_k^*)_{k=0}^n = \left(\cos\frac{2k+1}{2n+2}\pi\right)_{k=0}^n. \tag{1.24b}$$

Theorem 1.15. *For $f \in C^{n+1}[-1,1]$, the best choice of interpolating points is Δ_* as defined in (1.24b); hence from (1.18b)*

$$\|f - p_{\Delta_*}\|_\infty \leqslant \frac{1}{2^n}\frac{1}{(n+1)!}\,\|f^{(n+1)}\|_\infty\,. \tag{1.25}$$

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

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

2/11
2/12
2/13
2/14

2 Orthogonal Polynomials

2.1 Scalar products

Recall (e.g. from *Vectors & Matrices*) that the scalar product between two vectors, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, is given by

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

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

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

In general, a *scalar* (or *inner*) *product* is any function $\mathbb{V} \times \mathbb{V} \to \mathbb{R}$, where $\mathbb{V}$ is a linear vector space over the reals, subject to the following axioms:

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

We will consider linear vector spaces $\mathbb{V} = C^s[a,b]$ $(s \geqslant 0)$, i.e. linear vector spaces of all functions in $C[a,b]$ that possess s continuous derivatives (see Definition 1.2).

We wish to introduce scalar products for $\mathbb{V}$. In particular, suppose $w \in C[a,b]$ is a fixed *positive weight* function, then for all $f, g \in \mathbb{V}$ define

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

It is relatively straightforward to verify the axioms (2.2) for the scalar product, where, in order to prove the 'only if' part of the non-degeneracy condition $\langle f, f \rangle = 0$ iff $f = 0$, it is helpful to recall that w and f are continuous.

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 $\langle f, g \rangle = 0$.*

2.1.1 Degenerate inner products

If we consider *degenerate* inner products by dropping the non-degeneracy condition, a possible choice of a scalar product for $f, g \in \mathbb{V}$ is

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

where each w_j is a positive constant and each ξ_j is a fixed point of $[a,b]$. An important difference between (2.3) and (2.4) is that in the former case $\langle f, f \rangle = 0$ implies $f \equiv 0$, while in the latter case $\langle f, f \rangle = 0$ only implies $f(\xi_j) = 0$, $j = 1, 2, \ldots, m$.

2.2 Orthogonal polynomials – definition, existence, uniqueness

Let $\mathbb{V}$ be a space of functions from $[a,b]$ to $\mathbb{R}$ that includes all polynomials. Given a scalar product, say (2.3), we say that $p_n \in \mathbb{P}_n[x]$ is the n^{th} *orthogonal polynomial* if

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

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

Remark. Different inner products lead to different orthogonal polynomials.

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

Remark. We will normalise our orthogonal polynomials to be monic. However, with their usual definitions, some standard polynomials (e.g. the Chebyshev polynomials) are not necessarily scaled so as to be monic.

Theorem 2.3. *For a given scalar product, and for every $n \in \mathbb{Z}^+$, there exists a unique monic orthogonal polynomial of degree n. Moreover, any $p \in \mathbb{P}_n[x]$ can be expanded as a linear combination of $p_0, p_1, \ldots, p_n$, i.e. the $p_0, p_1, \ldots, p_n$ are a basis of $\mathbb{P}_n[x]$.*

Proof. First, we note that the unique monic polynomial of degree 0 is $p_0(x) \equiv 1$, and that every polynomial of degree 0 can be expressed as a linear multiple of p_0. We now proceed by induction on n.

Suppose that $p_0, p_1, \ldots, p_n$ have been already derived consistently with both assertions of the theorem. Let $q_{n+1}(x) \in \mathbb{P}_{n+1}[x]$ be a monic polynomial with degree of exactly $(n+1)$, e.g. $q_{n+1}(x) = x^{n+1}$. Guided by the *Gram–Schmidt algorithm,* we construct p_{n+1} by

$$p_{n+1}(x) = q_{n+1}(x) - \sum_{k=0}^{n} \frac{\langle q_{n+1}, p_k \rangle}{\langle p_k, p_k \rangle} \, p_k(x), \qquad x \in \mathbb{R}\,. \tag{2.6}$$

Then $p_{n+1} \in \mathbb{P}_{n+1}[x]$, and moreover p_{n+1} is monic (since all the terms in the sum are of degree $\leqslant n$).

To prove orthogonality, let $m \in \{0, 1, \ldots, n\}$; it follows from (2.6) and the induction hypothesis that

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

Hence, p_{n+1} is orthogonal to $p_0, \ldots, p_n$. Consequently, according to the second inductive assertion, it is orthogonal to all $p \in \mathbb{P}_n[x]$.

To prove uniqueness, we suppose the existence of two monic orthogonal polynomials $p_{n+1}, \tilde{p}_{n+1} \in \mathbb{P}_{n+1}[x]$. Let $r = p_{n+1} - \tilde{p}_{n+1}$. Since p_{n+1} and $\tilde{p}_{n+1}$ are both monic, $r \in \mathbb{P}_n[x]$, and hence $\langle p_{n+1}, r \rangle = \langle \tilde{p}_{n+1}, r \rangle = 0$. This implies

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

from which we deduce that $r \equiv 0$.

Finally, in order to prove that each $p \in \mathbb{P}_{n+1}[x]$ is a linear combination of $p_0, \ldots, p_{n+1}$, we note that we can always write it in the form $p = cp_{n+1} + q$, where c is the coefficient of x^{n+1} in p and where $q \in \mathbb{P}_n[x]$. According to the induction hypothesis, q can be expanded as a linear combination of $p_0, p_1, \ldots, p_n$, hence our assertion is true. □

2.3 The [quasi-linear] three-term recurrence relation

How to construct orthogonal polynomials? It turns out that the Gram–Schmidt algorithm (2.6) is not of much help, for it usually suffers from the same problem as the Gram–Schmidt algorithm in Euclidean spaces: loss of accuracy due to imprecisions in the calculation of scalar products. A smart choice of q_n can alleviate this problem. To this end we note that the only feature of the q_{n+1} that is necessary in the proof of the previous theorem is that the degree of the polynomial q_{n+1} is exactly $(n+1)$. Therefore the proof would remain valid if in (2.6) we make the choice

$$q_0(x) = 1 \quad \text{and} \quad q_{n+1}(x) = x\, p_n(x) \qquad n \geqslant 0\,, \quad x \in \mathbb{R}. \tag{2.7}$$

Note that the orthogonal polynomial $p_n \in \mathbb{P}_n$ is available before q_{n+1} is required.

The following theorem demonstrates that the functions (2.7) allow formula (2.6) to be simplified, provided that the scalar product has another property that is true in the case (2.3) (and also the case (2.4)); specifically, we require

$$\langle xf, g \rangle = \langle f, xg \rangle \quad \text{for} \quad x \in \mathbb{R}\,. \tag{2.8}$$

Theorem 2.4. *Monic orthogonal polynomials are given by the formula*

$$p_0(x) \equiv 1\,, \quad p_1(x) = (x-\alpha_0)p_0(x)\,, \tag{2.9a}$$
$$p_{n+1}(x) = (x-\alpha_n)p_n(x) - \beta_n p_{n-1}(x)\,, \qquad n=1,2,\ldots, \tag{2.9b}$$

where

$$\alpha_n = \frac{\langle p_n, xp_n\rangle}{\langle p_n, p_n\rangle}, \qquad \beta_n = \frac{\langle p_n, p_n\rangle}{\langle p_{n-1}, p_{n-1}\rangle} > 0. \tag{2.9c}$$

Proof. As before the unique monic polynomial of degree 0 is $p_0(x)\equiv 1$. Further, from (2.9a) and (2.9c)

$$p_1(x) = \left(x - \frac{\langle p_0, xp_0\rangle}{\langle p_0, p_0\rangle}\right) p_0(x)\,,$$

which of degree one and monic; moreover from (2.8)

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

and so p_1 is orthogonal to p_0.

Method 1. As suggested above, we define q_{n+1} by (2.7) and substitute into (2.6) to obtain for $n \geqslant 1$

$$\begin{aligned} p_{n+1} &= x\,p_n - \sum_{k=0}^{n} \frac{\langle x\,p_n, p_k\rangle}{\langle p_k, p_k\rangle}\,p_k \\ &= (x-\alpha_n)\,p_n - \frac{\langle p_n, x\,p_{n-1}\rangle}{\langle p_{n-1}, p_{n-1}\rangle}\,p_{n-1} - \sum_{k=0}^{n-2}\frac{\langle p_n, x\,p_k\rangle}{\langle p_k, p_k\rangle}\,p_k\,. \end{aligned}$$

Because of monicity, $x\,p_{n-1} = p_n + r$ where $r \in \mathbb{P}_{n-1}[x]$, and hence from the orthogonality property

$$\langle p_n, x\,p_{n-1}\rangle = \langle p_n, p_n\rangle\,.$$

Similarly, $\langle p_n, x\,p_k\rangle = 0$ for $k = 0, 1, \ldots, n-2$. The result (2.9b) follows. □

Method 2. Alternatively we need not use (2.6). We proceed by induction and assume that monic orthogonal polynomials $p_0, p_1, \ldots, p_n$ have been derived.

First note that p_{n+1} as defined by (2.9b) will be a monic polynomial of degree $n+1$. Further from the orthogonality of $p_0, p_1, \ldots, p_n$,

$$\langle p_{n+1}, p_\ell\rangle = \langle p_n, (x-\alpha_n)p_\ell\rangle - \beta_n\langle p_{n-1}, p_\ell\rangle = 0, \qquad \ell = 0, 1, \ldots, n-2.$$

As in *Method 1* we note that $xp_{n-1} = p_n + r$, where $r \in \mathbb{P}_{n-1}[x]$. Thus, from the definitions of α_n and β_n,

$$\begin{aligned} \langle p_{n+1}, p_{n-1}\rangle &= \langle p_n, (x-\alpha_n)p_{n-1}\rangle - \beta_n\langle p_{n-1}, p_{n-1}\rangle = \langle p_n, p_n\rangle - \beta_n\langle p_{n-1}, p_{n-1}\rangle = 0, \\ \langle p_{n+1}, p_n\rangle &= \langle p_n, (x-\alpha_n)p_n\rangle - \beta_n\langle p_{n-1}, p_n\rangle = \langle p_n, xp_n\rangle - \alpha_n\langle p_n, p_n\rangle = 0. \end{aligned}$$

Since any $p \in \mathbb{P}_n[x]$ can be expanded as a linear combination of $p_0, p_1, \ldots, p_n$, we conclude that p_{n+1} as defined by (2.9b) is orthogonal. □

2.4 Examples

As illustrated in Table 2, a number of standard orthogonal (non-monic) polynomials can be recovered by the appropriate choices of interval $[a, b] \subset \mathbb{R}$, and weight function w, in 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	Weight	Recurrence
Legendre	P_n	$[-1,1]$	$w(x)\equiv 1$	$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)$
Chebyshev	T_n	$[-1,1]$	$w(x)=\frac{1}{\sqrt{1-x^2}}$	$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$
Laguerre	L_n	$[0,\infty)$	$w(x)=\mathrm{e}^{-x}$	$(n+1)L_{n+1}(x) = (2n+1-x)L_n(x) - nL_{n-1}(x)$
Hermite	H_n	$(-\infty,\infty)$	$w(x)=\mathrm{e}^{-x^2}$	$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x)$

Table 2: *Some standard orthogonal polynomials*

2.4.1 Chebyshev polynomials

We have already defined the *Chebyshev* polynomial of degree n on $[-1,1]$ by (see (1.20))

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

where $T_0(x)\equiv 1$, $T_1(x)=x$, $T_2(x)=2x^2-1$, etc. We have also derived the recurrence relation (1.22b) for Chebyshev polynomials

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)\,, \quad n \geqslant 1, \tag{2.10b}$$

from the identity

$$\cos(n+1)\theta + \cos(n-1)\theta = 2\cos\theta\cos n\theta\,. \tag{2.10c}$$

We now show that this definition is consistent with the choice of scalar product, for $f,g\in C[-1,1]$,

$$\langle f,g\rangle = \int_{-1}^{1} f(x)g(x)\frac{\mathrm{d}x}{\sqrt{1-x^2}}\,, \quad w(x) = \frac{1}{\sqrt{1-x^2}}\,. \tag{2.11}$$

In particular, by the change of variable $x=\cos\theta$, we verify the orthogonality condition (2.5) for $n\neq m$

$$\begin{aligned}\langle T_n,T_m\rangle &= \int_{-1}^{1} T_n(x)T_m(x)\frac{\mathrm{d}x}{\sqrt{1-x^2}} = \int_0^\pi \cos n\theta\cos m\theta\,\mathrm{d}\theta \\ &= \tfrac12\int_0^\pi \{\cos(n+m)\theta + \cos(n-m)\theta\}\,\mathrm{d}\theta = 0\,.\end{aligned}$$

Remark. Recall that the Chebyshev polynomials are not monic, hence the inconsistency between (2.9b) and (2.10b). To obtain monic polynomials take

3/10
$$p_0 = 1\,, \quad p_n = \tfrac{1}{2^{n-1}}T_n(x) \quad \text{for} \quad n\geqslant 1.$$
3/11
3/12
3/13
3/14

2.5 Least-squares polynomial fitting

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 $f(x)$, $a\leqslant x\leqslant b$, or to function/data values $f(\xi_j)$, $j=1,2,\ldots,m>n+1$. Then it is often suitable to minimize the least-squares expression

$$\int_a^b w(x)[f(x)-p(x)]^2\,dx, \qquad \text{or} \qquad \sum_{j=1}^m w_j[f(\xi_j)-p(\xi_j)]^2, \tag{2.12}$$

respectively, where $w(x)>0$ for $x\in(a,b)$ and $w_1,w_2,\ldots,w_n>0$. Intuitively speaking, the polynomial p approximates f, and is an alternative to an interpolating polynomial; it is called a (*weighted*) *least-squares approximant.* The generality of the scalar product and the associated orthogonal polynomials are now highly useful, for in both the continuous and discrete cases we pick the scalar product (or degenerate scalar product) so that we require the least value of the distance $\|f-p\| = \langle f-p, f-p\rangle^{1/2}$.

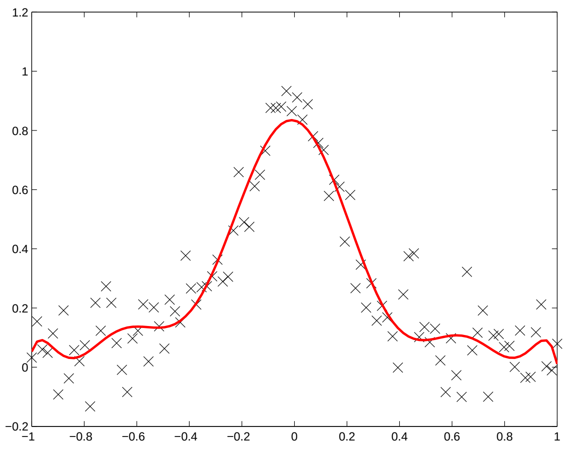


Figure 2.1: Least squares fit of a polynomial to discrete data.

Theorem 2.5. *Let $(p_k)_{k=0}^n$ be polynomials orthogonal with respect to a given inner product, where $p_k \in \mathbb{P}_k[x]$. Then the least squares approximant to any $f \in C[a,b]$ from $\mathbb{P}_n$ is given by the formula*

$$p \equiv p(f) = \sum_{k=0}^{n} c_k p_k, \quad c_k = \frac{\langle f, p_k\rangle}{\|p_k\|^2}, \tag{2.13a}$$

and the value of the least-squares approximation is

$$\|f-p\|^2 = \|f\|^2 - \sum_{k=0}^{n} \frac{\langle f, p_k\rangle^2}{\|p_k\|^2}. \tag{2.13b}$$

Proof. The orthogonal polynomials form a basis of $\mathbb{P}_n[x]$. Therefore for every $p \in \mathbb{P}_n$,

$$p = \sum_{k=0}^{n} c_k p_k \quad \text{for some} \quad c_0, c_1, \dots, c_n \in \mathbb{R}$$

Hence, from the properties of scalar products and the orthogonality conditions, $\langle p_j, p_k\rangle = 0$ for $j \neq k$,

$$\begin{aligned}
\langle f-p, f-p\rangle &= \langle f,f\rangle - 2\langle f,p\rangle + \langle p,p\rangle \\
&= \langle f,f\rangle - 2\left\langle f, \sum_{k=0}^{n} c_k p_k \right\rangle + \left\langle \sum_{j=0}^{n} c_j p_j, \sum_{k=0}^{n} c_k p_k \right\rangle \\
&= \langle f,f\rangle - 2\sum_{k=0}^{n} c_k \langle f, p_k\rangle + \sum_{j=0}^{n}\sum_{k=0}^{n} c_j c_k \langle p_j, p_k\rangle \\
&= \|f\|^2 - 2\sum_{k=0}^{n} c_k\langle f,p_k\rangle + \sum_{k=0}^{n} c_k^2 \|p_k\|^2 . \qquad (2.14)
\end{aligned}$$

The key point is that the orthogonality conditions remove the cross-terms from the right-hand side.

To derive optimal $c_0, c_1, \dots, c_n$ we seek to minimize the last expression, which we note is a quadratic function in each c_k. Since

$$\frac{\partial}{\partial c_j}\langle f-p, f-p\rangle = -2\langle p_j, f\rangle + 2c_j\langle p_j, p_j\rangle, \qquad j = 0, 1, \dots, n,$$

setting the gradient to zero yields

$$c_k = \frac{\langle f, p_k\rangle}{\langle p_k, p_k\rangle}, \quad \text{and hence} \quad p(x) = \sum_{k=0}^{n} \frac{\langle f, p_k\rangle}{\langle p_k, p_k\rangle} p_k(x). \tag{2.15}$$

Substituting the optimal c_k into (2.14) we obtain (2.13b). □

Remarks.

(i) We note that the c_k are the components of [sufficiently nice functions] f with respect to the [infinite] basis consisting of the p_n, $n = 0, 1, \ldots$.

(ii) We observe, using the expressions for p and c_k from (2.13a), that

$$\begin{aligned}\|p\|^2 \equiv \langle p, p\rangle &= \left\langle \sum_{j=0}^{n} c_j p_j, \sum_{k=0}^{n} c_k p_k \right\rangle = \sum_{j=0}^{n}\sum_{k=0}^{n} c_j c_k \langle p_j, p_k\rangle \\ &= \sum_{k=0}^{n} c_k^2 \langle p_k, p_k\rangle = \sum_{k=0}^{n} \frac{\langle f, p_k\rangle^2}{\|p_k\|^2}\,. \end{aligned} \tag{2.16a}$$

Thence from (2.13b) we obtain

$$\|f-p\|^2 + \|p\|^2 = \|f\|^2\,. \tag{2.16b}$$

This can be viewed as an analogy of the theorem of Pythagoras. Indeed, p and $f-p$ are orthogonal,[4] since it follows using the expressions for p and c_k from (2.13a) and the orthogonality of the p_k, that

$$\langle p, f-p\rangle = \left\langle \sum_{k=0}^{n} c_k p_k,\ f - \sum_{j=0}^{n} c_j p_j \right\rangle = \sum_{k=0}^{n} c_k \left(\langle p_k, f\rangle - c_k \langle p_k, p_k\rangle\right) = 0\,. \tag{2.16c}$$

2.5.1 Accuracy: how to choose n?

Suppose that we wish to choose n so that the 'error' $\langle f-p, f-p\rangle$ is less than a prescribed *tolerance* $\varepsilon > 0$, i.e. so that

$$0 \leqslant \langle f-p, f-p\rangle = \langle f, f\rangle - \langle p, p\rangle = \langle f, f\rangle - \sum_{k=0}^{n} \frac{\langle f, p_k\rangle^2}{\|p_k\|^2} < \varepsilon\,. \tag{2.17a}$$

We first note that the construction and evaluation of both p_k and $c_k = \langle p_k, f\rangle / \langle p_k, p_k\rangle$ 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 (2.15) or (2.17a). In particular all that is necessary to achieve the desired tolerance is to continue to add terms until (2.17a) is satisfied, i.e. until

$$\sigma_n \equiv \sum_{k=0}^{n} \frac{\langle f, p_k\rangle^2}{\langle p_k, p_k\rangle} > \langle f, f\rangle - \varepsilon \quad n = 0, 1, \ldots\,. \tag{2.17b}$$

We note that this condition can always be satisfied for the degenerate scalar product (2.4) if the data points $\{\xi_j : j = 1, 2, \ldots, m\}$ are distinct, because $\langle f-p, f-p\rangle$ is made zero by interpolation to $\{f(\xi_j) : j = 1, 2, \ldots, m\}$ if $n = m-1$. For the scalar product (2.3) when $[a, b]$ is a bounded interval and f is continuous, the following theorem ensures that n exists.

Theorem 2.6 (The Parseval identity)**.** *Let $[a, b]$ be finite. Then*

$$\sum_{k=0}^{\infty} \frac{\langle f, p_k\rangle^2}{\langle p_k, p_k\rangle} = \langle f, f\rangle. \tag{2.18}$$

Incomplete proof. From (2.17a) and (2.17b)

$$\langle f-p, f-p\rangle = \langle f, f\rangle - \sigma_n \geqslant 0\,.$$

The sequence $\{\sigma_n\}_{n=0}^{\infty}$ increases monotonically and $\sigma_n \leqslant \langle f, f\rangle$ implies that $\lim_{n\to\infty} \sigma_n$ exists. According to the *Weierstrass theorem* (no proof), any function in $C[a, b]$ can be approximated arbitrarily close by a polynomial, hence $\lim_{n\to\infty}\langle f-p, f-p\rangle = 0$. We deduce that $\sigma_n \to \langle f, f\rangle$ as $n \to \infty$, and that (2.18) is true. □

[4] Which in turn allows an alternative derivation of (2.16b) by use of

$$\langle f-p, f-p\rangle = \langle f, f-p\rangle - \langle p, f-p\rangle = \langle f, f\rangle - \langle f, p\rangle = \langle f, f\rangle - \langle p, p\rangle\,.$$

2.6 Least-squares fitting to discrete function values

Let $\boldsymbol{f}^{\mathrm{T}} = (f(x_1), f(x_2), \ldots, f(x_m))$ be m function values. Suppose that the points x_j are distinct, and that we seek a polynomial $p \in \mathbb{P}_n[x]$ with $n \leqslant (m-1)$ that minimizes $\langle f-p, f-p\rangle$ for the scalar product (2.4) with $w_j = 1$ $(j = 1, \ldots, m)$, i.e. we seek a polynomial that minimizes

$$\langle f-p, f-p\rangle = \sum_{j=1}^{m} (f(x_j) - p(x_j))^2 = \|\boldsymbol{f} - \boldsymbol{p}\|^2\,, \tag{2.19}$$

where $\boldsymbol{p}^{\mathrm{T}} = (p(x_1), p(x_2), \ldots, p(x_m))$.

(a) One possibility is to write $p = \sum_{k=0}^{n} \theta_k x^k$. Then for $j = 1, \ldots, m$,

$$p(x_j) = \sum_{k=0}^{n} \theta_k x_j^k = \sum_{k=0}^{n} A_{jk}\theta_k\,, \quad \text{where} \quad A_{jk} = x_j^k\,, \quad \text{i.e.} \quad \boldsymbol{p} = \mathsf{A}\boldsymbol{\theta}$$

As we shall see later, the problem of minimising (2.19) is then equivalent to finding a *least-squares solution* for $\boldsymbol{\theta}$ to the linear system

$$\sum_{k=0}^{n} x_j^k\,\theta_k = f(x_j) \quad \text{for} \quad j = 1, \ldots, m\,, \quad \text{i.e.} \quad \mathsf{A}\boldsymbol{\theta} = \boldsymbol{f}. \tag{2.20}$$

Remark: complexity. Using numerical linear algebra (in particular QR factorisation, as will be considered in the sequel), this requires $\mathcal{O}(mn^2)$ operations.

(b) An alternative is to construct orthogonal polynomials $p_0, p_1, \ldots, p_n$ w.r.t. the scalar product (2.4) with $w_j = 1$ $(j = 1, \ldots, m)$. For sufficiently small n it is possible to employ the three-term recurrence relations (2.9a)-(2.9c) to calculate $p_0, p_1, \ldots, p_n$. However, *inter alia*, the uniqueness part of the proof of Theorem (2.3), and the finiteness of the coefficients in the recurrence relations, depend on the scalar product being non-degenerate; unfortunately, as already noted, the scalar product (2.4) *is* degenerate (since $\langle f, f\rangle = 0$ if $f(\xi_j) = 0$ for $j = 1, \ldots, m$).

We make two observations.

(i) First that if q is a non-zero polynomial of degree $n \leqslant m-1$, then q has less than m zeros, and so $\langle q, q\rangle > 0$, i.e. if we restrict to $q(x) \in \mathbb{P}_n[x]$ with $n \leqslant m-1$ then the scalar product is non-degenerate.

(ii) Second, that this is not so if $n \geqslant m$, since if

$$q(x) = \prod_{k=1}^{m} (x - \xi_k) \ \in \mathbb{P}_m[x]\,. \tag{2.21}$$

then $\langle q, q\rangle = 0$.

The consequence is that only orthogonal polynomials $p_0, p_1, \ldots, p_{m-1}$ can be found for this scalar product. However, this is adequate for our purposes since we have assumed that $n \leqslant m-1$. The least-squares polynomial fit is then given by (cf. (2.13a))

$$p(x) = \sum_{k=0}^{n} \frac{\langle p_k, f\rangle}{\langle p_k, p_k\rangle}\, p_k(x)\,. \tag{2.22}$$

Remark: complexity. Since it requires $\mathcal{O}(m)$ operations to calculate each scalar product, it requires $\mathcal{O}(m)$ operations to calculate $p_{k+1}(x_j)$ from $p_k(x_j)$ and $p_{k-1}(x_j)$ for $j = 1, \ldots, m$. Hence the cost to find the n coefficients in (2.15) is $\mathcal{O}(mn)$ operations.

3 Approximation of Linear Functionals

3.1 Linear functionals

Definition 3.1. A *linear functional* is a linear mapping, $L : \mathbb{V} \to \mathbb{R}$, from a linear space of functions $\mathbb{V}$ to $\mathbb{R}$.

For example, if the space is $C[a,b]$, then

$$L(f) = \int_a^b f(x)w(x)\,dx \tag{3.1}$$

is a linear functional, because $L(f)\in\mathbb{R}$ for every f in $C[a,b]$ and because

$$L(\alpha f + \beta\, g) = \alpha\, L(f) + \beta\, L(g), \quad f, g\in C[a,b], \quad \alpha, \beta \in \mathbb{R}\,. \tag{3.2}$$

Other examples include

(i) $L(f) = f(\xi)$, where $f\in C[a,b]$, and ξ is any fixed point of $[a,b]$;

(ii) $L(f) = f'(\xi)$, where $f\in C^1[a,b]$, and ξ is any fixed point of $[a,b]$;

(iii) for $f\in C^1[a,b]$, $x \in [a,b]$, $(x+h) \in [a,b]$, and with a slight change in notation,

$$e_L(f) = f(x+h) - f(x) - \tfrac{1}{2}h\,\left[f'(x) + f'(x+h)\right]\,. \tag{3.3a}$$

We will treat the space $\mathbb{V} = C^{n+1}[a,b]$, and our goal is to find approximations of the form

$$L(f) \approx \sum_{i=0}^{N} a_i f(x_i), \qquad f \in C^{n+1}[a,b]\,. \tag{3.3b}$$

Remark. For the functionals (3.1) this is called *numerical integration*, and for

$$L(f) = f^{(k)}(\xi)\,, \quad \xi \in [a,b], \quad 0 \leqslant k \leqslant n+1\,, \tag{3.4}$$

this is called *numerical differentiation.*

Method 3.2. *Interpolating formulae.* A suggestive approximation method is to interpolate f by $p \in \mathbb{P}_n$, and then take $L(f) \approx L(p)$. This is an *interpolating formula* (of degree n), and in this case $N = n$.

We have already seen an interpolating formula, namely (1.1c), i.e. that of Lagrange for the functional $L(f) = f(\xi)$:

$$f(\xi) \approx p(\xi) = \sum_{i=0}^{n} \ell_i(\xi) f(x_i)\,, \tag{3.5a}$$

where, as before, we denote the Lagrange cardinal polynomials for the points $x_0, x_1, \ldots, x_n$ by $\ell_i(\xi)$. Because every $p \in \mathbb{P}_n[x]$ is its own interpolating polynomial, i.e. because $p(\xi) = \sum_{i=0}^{n} \ell_i(\xi)p(x_i)$, we can use this result and the linearity of L, to deduce that an interpolating formula has the form

$$L(f) \approx L(p) = \sum_{i=0}^{n} L(\ell_i)p(x_i) = \sum_{i=0}^{n} L(\ell_i)f(x_i). \tag{3.5b}$$

Method 3.3. *'Exact' formulae.* An alternative approximation method is to require the formula (3.3b) to be *exact on* $\mathbb{P}_n$, i.e. to require for any $p \in \mathbb{P}_n$ that

$$L(p) = \sum_{i=0}^{N} a_i p(x_i)\,. \tag{3.6}$$

In this case the number N of terms need not be as restricted:

- if $N > n$, then it is not a polynomial of degree n that substitutes the function;
- if $N < n$, then the formula is said to be of *high accuracy*;
- if $N = n$, then Method 3.2 and Method 3.3 are the same (see the following lemma).

Lemma 3.4. *The formula $L(f) \approx \sum_{i=0}^{n} a_i f(x_i)$ is interpolating $\quad\Leftrightarrow\quad$ it is exact for $f \in \mathbb{P}_n$.*

Proof. The interpolating formula (3.5b) is exact on $\mathbb{P}_n$ by definition. Conversely, if the formula in the lemma is exact on $\mathbb{P}_n$, take $f(x) = \ell_j(x) \in \mathbb{P}_n$ to obtain

$$L(\ell_j) = \sum_{i=0}^{n} a_i \ell_j(x_i) = a_j\,, \tag{3.7}$$

i.e. (3.5b). □

3.2 Gaussian quadrature

The term *quadrature* is used when an integral is approximated by a finite linear combination of function values. For instance, for $f \in C[a,b]$ a typical approximation, or *quadrature formula.*, is

$$I(f) = \int_a^b w(x) f(x)\,\mathrm{d}x \approx \sum_{k=1}^{\nu} b_k f(c_k)\,, \tag{3.8}$$

where w is a fixed positive weight function, $\nu = N+1$ is given, and the real multipliers b_k, $k = 1,\ldots,\nu$ (or *interpolatory weights*) and the points, $c_k \in [a,b]$, $k = 1,\ldots,\nu$ (or *nodes* or *knots*) are independent of the choice of $f \in C[a,b]$. 4/12

4/13
4/14
In order to obtain an approximation of high accuracy we adopt Method 3.3; in particular, based on the fact that Taylor series are polynomials, we seek an approximant that is exact for all $f \in \mathbb{P}_m[x]$, where m is as large as possible. We will demonstrate that $m = 2\nu - 1$ can be attained, and that that the required nodes $\{c_k : k=1,\ldots,\nu\}$ are the zeros of p_ν, where p_ν is the orthogonal monic polynomial of degree ν for the scalar product

$$\langle f, g\rangle = \int_a^b w(x) f(x) g(x)\,\mathrm{d}x\,, \quad f, g \in C[a,b]\,. \tag{3.9}$$

Claim 3.5. *Firstly, we claim that $m = 2\nu$ is impossible, i.e. that no quadrature formula with ν nodes is exact for all $q \in \mathbb{P}_m$ if $m \geqslant 2\nu$.*

Proof. We prove by contradiction. Let $c_1,\ldots,c_\nu$ be arbitrary nodes, and note that

$$q(x) = \prod_{k=1}^{\nu} (x - c_k)^2 \quad \in \quad \mathbb{P}_{2\nu}[x]\,.$$

But $\int_a^b w(x) q(x)\,\mathrm{d}x > 0$, while $\sum_{k=1}^{\nu} b_k q(c_k) = 0$ for any choice of weights $b_1,\ldots,b_\nu$. Hence the integral and the quadrature do not match. □

Next we obtain a result about the zeros of orthogonal monic polynomials $p_0, p_1, p_2, \ldots$.

Theorem 3.6. *Given $n \geqslant 1$, all the zeros of p_n are real, distinct and lie in the interval (a,b).*

Proof. Recall that $p_0 \equiv 1$. Thus, for $n \geqslant 1$ and by orthogonality,

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

We deduce that p_n changes sign at least once in (a,b). Denote by $m \geqslant 1$ the number of the sign changes of p_n in (a,b), and suppose that the points where a sign change occurs are given by $\xi_1, \xi_2, \ldots, \xi_m$. Let

$$q(x) = \prod_{j=1}^{m} (x - \xi_j)\,.$$

Next suppose that $m \leqslant n-1$. Then, since $q \in \mathbb{P}_m[x]$, it follows that $\langle q, p_n\rangle = 0$. On the other hand, it follows from our construction that $q(x)p_n(x)$ does not change sign throughout $[a,b]$ and vanishes at a finite number of points, hence

$$|\langle q, p_n\rangle| = \left|\int_a^b w(x) q(x) p_n(x)\,\mathrm{d}x\right| = \int_a^b w(x) |q(x) p_n(x)|\,\mathrm{d}x > 0\,.$$

This is a contradiction, so it follows that $m \geqslant n$; but a polynomial of degree n has at most n zeros, hence 4/10
$m = n$ and the proof is complete. □
4/11

3.2.1 Weights and knots

Theorem 3.7. *For pairwise-distinct nodes $c_1, c_2, \ldots, c_\nu \in [a,b]$, the quadrature formula* (3.8) *with the interpolatory weights*

$$b_k = \int_a^b w(x) \prod_{\substack{j=1 \\ j\neq k}}^{\nu} \frac{x-c_j}{c_k-c_j}\,\mathrm{d}x, \qquad k=1,2,\ldots,\nu\,, \tag{3.10}$$

is exact for all $f \in \mathbb{P}_{\nu-1}[x]$.

Proof. From (1.1b) the Lagrange cardinal polynomials for the points $c_1, \ldots, c_\nu$ are defined by

$$\ell_k(x) = \prod_{\substack{j=1 \\ j\neq k}}^{\nu} \frac{x-c_j}{c_k-c_j}, \qquad k=1,\ldots,\nu. \tag{3.11}$$

Then, because the interpolating formula is exact on $\mathbb{P}_{\nu-1}$ (see Lemma 3.4 and equation (3.5b)), the quadrature is exact for all $f \in \mathbb{P}_{\nu-1}[x]$ if, as required by (3.10).[5] $\square$

$$b_k = I(\ell_k) = \int_a^b w(x) \prod_{\substack{j=1 \\ j\neq k}}^{\nu} \frac{x-c_j}{c_k-c_j}\,\mathrm{d}x\,. \tag{3.12}$$

Example: Trapedoidal Rule. $w=1$, $c_1=a$, $c_2=b$, $b_1 = \int_a^b \frac{x-b}{a-b}\,\mathrm{d}x = \frac{1}{2}(b-a) = \int_a^b \frac{x-a}{b-a}\,\mathrm{d}x = b_2$.

Theorem 3.8. *If the $c_1, c_2, \ldots, c_\nu$ are the zeros of p_ν then* (3.8) *is exact for all $f \in \mathbb{P}_{2\nu-1}[x]$. Further, $b_k > 0$ for all k (i.e. all the weights are positive).*

Proof. Suppose that the $c_1, \ldots, c_\nu$ are the zeros of p_ν. Given $f \in \mathbb{P}_{2\nu-1}[x]$, we can represent it uniquely as

$$f(x) = q(x)p_\nu(x) + r(x)\,,$$

where $q, r \in \mathbb{P}_{\nu-1}[x]$. Thus, by orthogonality,

$$\begin{aligned}\int_a^b w(x)f(x)\,\mathrm{d}x &= \int_a^b w(x)[q(x)p_\nu(x)+r(x)]\,\mathrm{d}x = \langle q, p_\nu\rangle + \int_a^b w(x)r(x)\,\mathrm{d}x \\ &= \int_a^b w(x)r(x)\,\mathrm{d}x.\end{aligned}$$

On the other hand, the choice of quadrature knots gives

$$\sum_{k=1}^{\nu} b_k f(c_k) = \sum_{k=1}^{\nu} b_k[q(c_k)p_\nu(c_k) + r(c_k)] = \sum_{k=1}^{\nu} b_k r(c_k)\,.$$

Since $r \in \mathbb{P}_{\nu-1}[x]$ and from Theorem 3.7 the quadrature is exact for all polynomials in $\mathbb{P}_{\nu-1}[x]$, it follows that the integral and its approximation coincide.

Next, define the polynomials $\mathcal{L}_i(x) \in \mathbb{P}_{2\nu-2}[x]$ for $i=1,\ldots,\nu$ by

$$\mathcal{L}_i(x) = \prod_{\substack{j=1 \\ j\neq i}}^{\nu} \frac{(x-c_j)^2}{(c_i-c_j)^2}\,, \quad \text{where we note that} \quad \mathcal{L}_i(c_k) = \delta_{ik}\,. \tag{3.13a}$$

[5]Alternatively, recall that every $q \in \mathbb{P}_{\nu-1}[x]$ is its own interpolating polynomial. Hence by Lagrange's formula

$$q(x) = \sum_{k=1}^{\nu} q(c_k) \prod_{\substack{j=1 \\ j\neq k}}^{\nu} \frac{x-c_j}{c_k-c_j}.$$

The quadrature is exact for all $q \in \mathbb{P}_{\nu-1}[x]$ if

$$\sum_{k=1}^{\nu} b_k q(c_k) = \int_a^b w(x)q(x)\,\mathrm{d}x = \int_a^b w(x) \sum_{k=1}^{\nu} q(c_k) \prod_{\substack{j=1 \\ j\neq k}}^{\nu} \frac{x-c_j}{c_k-c_j}\,\mathrm{d}x = \sum_{k=1}^{\nu} \left(\int_a^b w(x) \prod_{\substack{j=1 \\ j\neq k}}^{\nu} \frac{x-c_j}{c_k-c_j}\,\mathrm{d}x \right) q(c_k)\,,$$

i.e. if the $b_1, b_2, \ldots, b_\nu$ are given by (3.10).

The $\mathcal{L}_i(x)$ are non-zero, continuous and positive. Further, the quadrature formula (3.8) is exact for them. Hence

$$0 < \int_a^b w(x)\mathcal{L}_i(x)dx = \sum_{k=1}^{\nu} b_k \mathcal{L}_i(c_k) = b_i. \tag{3.13b}$$

It follows that all the weights are positive. □

Definition 3.9. A quadrature with ν nodes that is exact on $\mathbb{P}_{2\nu-1}$ is called *Gaussian quadrature.*

3.2.2 Examples

(i) Let $[a,b]=[-1,1]$, $w(x)\equiv 1$. Then the underlying orthogonal polynomials are the *Legendre polynomials.* The first few polynomials are, with the customary non-monic normalization $P_\nu(1)=1$,

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

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

ν	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	$b_1=b_4=\frac{1}{2}+\frac{1}{6}\sqrt{\frac{5}{6}}$ $b_2=b_3=\frac{1}{2}-\frac{1}{6}\sqrt{\frac{5}{6}}$	$c_1=-c_4,\ c_4=\left(\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}\right)^{\frac{1}{2}}$ $c_2=-c_3,\ c_3=\left(\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}\right)^{\frac{1}{2}}$	$\mathbb{P}_7$

(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

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

(iii) There are many other, not necessarily Gaussian, schemes; e.g. for $w\equiv 1$, the rectangle rule on $[a,b]$,

$$I(f)\approx (b-a)f(a),$$

and others:

Rule	ν	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	$b_1=b_3=\frac{1}{6}(b-a)$ $b_2=\frac{2}{3}(b-a)$	$c_1=a,\ c_3=b$ $c_2=\frac{1}{2}(a+b)$	$\mathbb{P}_3$	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

Consider the interpolating formulae (3.5b) for numerical differentiation

$$L(f) = f^{(k)}(\xi) \approx \sum_{i=0}^{n} a_i f(x_i), \qquad a_i = L(\ell_i) = \ell_i^{(k)}(\xi)\,. \tag{3.14}$$

As previously noted, these are exact on the polynomials of degree n. The simplest formulae have $n = k$, in which case

$$f^{(k)}(\xi) \approx p^{(k)}(\xi)\,,$$

where p is the interpolating polynomial of degree k. Since $p^{(k)}(\xi)$ is $k!$ times the leading coefficient of p, i.e., $k!f[x_0, \ldots, x_k]$, we obtain the formulae (cf. (1.14a))

$$f^{(k)}(\xi) \approx k!f[x_0, \ldots, x_k]\,. \tag{3.15}$$

3.3.1 Examples (*Unlectured*)

$n = k$

Forward difference, 2-point, exact on linear fns:
$$f'(x) \approx f[x, x+h] = \frac{f(x+h) - f(x)}{h}, \tag{3.16a}$$

Central difference, 2-point, exact on quadratics:
$$f'(x) \approx f[x-h, x+h] = \frac{f(x+h) - f(x-h)}{2h}, \tag{3.16b}$$

2nd-order central difference, 3-point, exact on cubics:
$$f''(x) \approx 2f[x-h, x, x+h] = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{3.16c}$$

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

$$f'(0) \approx p_2'(0) = -\tfrac{3}{2}f(0) + 2f(1) - \tfrac{1}{2}f(2)\,. \tag{3.17}$$

Given the nodes (x_i), in our case $(0, 1, 2)$, we can find the corresponding coefficients (a_i) in two ways.

(i) First determine the fundamental Lagrange polynomials ℓ_i

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

then, from (3.14), set $a_i = L(\ell_i)$:

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

(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 (3.17) and $x_i = 0, 1, 2$

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

4 Expressing Errors in Terms of Derivatives

4.1 The error of an approximant

Given a linear functional L (e.g. a function, a derivative at a given point, an integral, etc.) and an approximation scheme $L(f) \approx \sum_{i=0}^{N} a_i f(x_i)$, we are interested in the error (which is also a linear functional)

$$e_L(f) = L(f) - \sum_{i=0}^{N} a_i f(x_i)\,. \tag{4.1a}$$

If L acts on $f \in C^{k+1}[a,b]$, then we seek an estimate in terms of $\|f^{(k+1)}\|_\infty$, i.e.,

$$|e_L(f)| \leqslant c_L\, \|f^{(k+1)}\|_\infty, \quad \forall f \in C^{k+1}[a,b]\,. \tag{4.1b}$$

where $c_L \in \mathbb{R}$. Since we know that $f^{(k+1)} \equiv 0$ for $f \in \mathbb{P}_k$, an estimate of the form (4.1b) can only exist if

$$e_L(f) = 0 \quad \forall f \in \mathbb{P}_k[x],$$

i.e., the approximation formula must be *exact on* $\mathbb{P}_k$ (e.g., it can be interpolating of degree k).

Example. Consider

$$L(f) = f(x+h) \approx f(x) + \tfrac{1}{2}h\,[f'(x+h) + f'(x)]\,. \tag{4.2a}$$

This approximation is exact if f is any quadratic polynomial. The error of the approximant is given by (cf. (3.3a))

$$e_L(f) = f(x+h) - f(x) - \tfrac{1}{2}h\;[f'(x) + f'(x+h)]\,, \tag{4.2b}$$

where $e_L(p) = 0$ for $p \in \mathbb{P}_2$.

Definition 4.1. If (4.1b) holds with some c_L and moreover, for any $\epsilon > 0$, there is an $f_\epsilon \in C^{k+1}[a,b]$ such that

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

then the constant c_L is called *least* or *sharp*.

4.2 Preliminaries

Since our aim is to obtain an expression for the error that depends on $f^{(k+1)}$, recall the Taylor formula in the form with an integral remainder term:

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + \cdots + \frac{(x-a)^k}{k!}f^{(k)}(a) + \frac{1}{k!}\int_a^x (x-\theta)^k f^{(k+1)}(\theta)\,\mathrm{d}\theta\,. \tag{4.3a}$$

This formula can be established by integration by parts. The range of integration can be made independent of x by introducing the notation for integer $k \geqslant 0$

$$(x-\theta)_+^k = \begin{cases} (x-\theta)^k & x \geqslant \theta \\ 0 & x < \theta \end{cases}\,. \tag{4.3b}$$

Remark. If H is the Heaviside/step function, then $(x-\theta)_+^0 = H(x-\theta)$.

We then have

$$f(x) = \sum_{r=0}^{k} \frac{1}{r!}(x-a)^r f^{(r)}(a) + \frac{1}{k!}\int_a^b (x-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta\,. \tag{4.3c}$$

Let λ be a linear functional on C^{n+1}, then

$$\lambda(f) = \sum_{r=0}^{k} \frac{f^{(r)}(a)}{r!}\lambda((x-a)^r) + \lambda\left(\frac{1}{k!}\int_a^b (x-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta\right)\,. \tag{4.4}$$

Suppose now that λ is a linear functional such that $\lambda(p) = 0$ for $p\in\mathbb{P}_k$. Then from (4.4) we have that for $f\in C^{k+1}[a,b]$ and $a\leqslant x\leqslant b$

$$\lambda(f) = \lambda\left(\frac{1}{k!}\int_a^b (x-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta\right). \tag{4.5}$$

If we identify λ with e_L then this is our hoped for expression for the error that depends on $f^{(k+1)}$.

Example. Since approximation (4.2a) is exact if $f \in \mathbb{P}_2$, we conclude for $f\in C^3[a,b]$ and $e_L(f)$ as defined by (4.2b), that

$$e_L(f) = e_L\left(\tfrac{1}{2}\int_a^b (x-\theta)_+^2 f'''(\theta)\,\mathrm{d}\theta\right).$$

4.2.1 Exchange of the order of L and integration

In what follows we will assume that the order of action of $\int$ and λ in (4.5) can be exchanged (see (4.8a)). Since, typically, λ involves summation, integration and differentiation, this assumption is not unduly restrictive. This is illustrated by the following examples, where we let

$$g(x) = \int_a^b (x-\theta)_+^k\, f^{(k+1)}(\theta)\,d\theta, \tag{4.6}$$

(i) First suppose for a function $f(x)$, that $\lambda(f) = \sum_{j=1}^m \alpha_j f(\xi_j)$, where $\alpha_j, \xi_j\in\mathbb{R}$. Then

$$\begin{aligned}\lambda(g) = \sum_{j=1}^m \alpha_j\left(\int_a^b (\xi_j-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta\right) &= \int_a^b \left(\sum_{j=1}^m \alpha_j(\xi_j-\theta)_+^k f^{(k+1)}(\theta)\right)\mathrm{d}\theta\\ &= \int_a^b \lambda\left((x-\theta)_+^k\right) f^{(k+1)}(\theta)\,\mathrm{d}\theta.\end{aligned}$$

(ii) Next suppose that $\lambda(f) = \int_a^b \beta(x)\, f(x)\,\mathrm{d}x$, where $\beta\in C[a,b]$. Then for $g\equiv g(x)$ given by (4.6)

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

(iii) Finally suppose that $\lambda(f) = \frac{\mathrm{d}^\ell f}{\mathrm{d}x^\ell}$, where $1\leqslant\ell\leqslant k-1$. Then for $g\equiv g(x)$ given by (4.6)

$$\begin{aligned}\lambda(g) = \frac{\mathrm{d}^\ell}{\mathrm{d}x^\ell}\left(\int_a^b (x-\theta)_+^k f^{(k+1)}(\theta)\,\mathrm{d}\theta\right) &= \int_a^b \frac{\mathrm{d}^\ell}{\mathrm{d}x^\ell}\left((x-\theta)_+^k f^{(k+1)}(\theta)\right)\mathrm{d}\theta\\ &= \int_a^b \lambda\left((x-\theta)_+^k\right) f^{(k+1)}(\theta)\,\mathrm{d}\theta.\end{aligned}$$

4.3 The Peano kernel theorem

Theorem 4.2 (Peano kernel theorem)**.** *Let λ be a linear functional from $C^{k+1}[a,b]$ to $\mathbb{R}$ such that $\lambda(f) = 0$ for all $f\in\mathbb{P}_k[x]$. Suppose also that λ applied to $\int_a^b (x-\theta)_+^k\, f^{(k+1)}(\theta)\,d\theta$ commutes with the integration sign, then $\lambda(f)$ is a linear functional of the derivative $f^{(k+1)}(\theta)$, $a\leqslant\theta\leqslant b$, of the form*

$$\lambda(f) = \frac{1}{k!}\int_a^b K(\theta) f^{(k+1)}(\theta)\,\mathrm{d}\theta, \tag{4.7a}$$

where for given $x\in[a,b]$

$$K(\theta) = \lambda((x-\theta)_+^k) \quad \textit{for} \quad a\leqslant\theta\leqslant b. \tag{4.7b}$$

5/13 *is the* Peano kernel function.

5/14 *Remark. $K(\theta)$ is independent of f.*

Proof. Since it is assumed that

$$\lambda\left(\int_a^b (x-\theta)_+^k\, f^{(k+1)}(\theta)\, d\theta\right) = \int_a^b \lambda\left((x-\theta)_+^k f^{(k+1)}(\theta)\right)\, d\theta\,, \tag{4.8a}$$

it follows from (4.5) and the linearity of λ that

$$\lambda(f) = \frac{1}{k!}\int_a^b \lambda\left((x-\theta)_+^k f^{(k+1)}(\theta)\right)\,\mathrm{d}\theta = \frac{1}{k!}\int_a^b \lambda\left((x-\theta)_+^k\right) f^{(k+1)}(\theta)\,\mathrm{d}\theta\,. \tag{4.8b}$$

5/12 (long) The formula (4.7a) follows from the definition of $K(\theta)$. □

4.3.1 Examples

(i) We have already seen that the functional (4.2b),

$$e_L(f) = f(\beta) - f(\alpha) - \tfrac{1}{2}(\beta-\alpha)\,\left[f'(\alpha)+f'(\beta)\right]\,, \tag{4.9}$$

where $\alpha = x$ and $\beta = x+h$, is the error of the approximant (4.2a). Since $e_L(p)=0$ for $p\in\mathbb{P}_2$, we may set $k=2$ in (4.7a). To evaluate the Peano kernel function K, we fix θ and let $g(x) = (x-\theta)_+^2$. It follows from the definition (4.3b) that

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

and thus that $g'(x) = 2(x-\theta)_+$. Hence

$$\begin{aligned} K(\theta) = e_L\left((x-\theta)_+^2\right) = e_L\left(g\right) &= g(\beta)-g(\alpha)-\tfrac{1}{2}(\beta-\alpha)\,\left(g'(\beta)+g'(\alpha)\right)\\ &= (\beta-\theta)_+^2-(\alpha-\theta)_+^2-\tfrac{1}{2}(\beta-\alpha)\,\left(2\,(\beta-\theta)_+ + 2\,(\alpha-\theta)_+\right)\\ &= \begin{cases} 0 & a\leqslant\theta\leqslant\alpha \quad\text{and}\quad \beta\leqslant\theta\leqslant b\\ (\alpha-\theta)\,(\beta-\theta) & \alpha\leqslant\theta\leqslant\beta \end{cases}\,. \end{aligned} \tag{4.10}$$

We conclude that

$$e_L(f) = \tfrac{1}{2}\int_\alpha^\beta (\alpha-\theta)\,(\beta-\theta)\,f'''(\theta)\,d\theta\,. \tag{4.11a}$$

Further, we note from integrating by parts that

$$e_L(f) = \tfrac{1}{2}\int_\alpha^\beta (\alpha+\beta-2\,\theta)\,f''(\theta)\,d\theta\,. \tag{4.11b}$$

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

(ii) *Unlectured.* Consider the approximation (3.17)

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

The error of this approximation is the linear functional

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

As determined earlier (or as may be verified by trying $f(x) = 1, x, x^2$ and then invoking linearity), $e_L(f) = 0$ for $f\in\mathbb{P}_2[x]$. Thus, for $f\in C^3[0,2]$ we have

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

The Peano kernel function K is given by, again with $g(x) = (x-\theta)_+^2$,

$$\begin{aligned} K(\theta) = e_L((x-\theta)_+^2) &= e_L(g) = g'(0) + \tfrac{3}{2}g(0) - 2g(1) + \tfrac{1}{2}g(2)\,. \\ &= 2(0-\theta)_+ + \tfrac{3}{2}(0-\theta)_+^2 - 2(1-\theta)_+^2 + \tfrac{1}{2}(2-\theta)_+^2 \\ &= \begin{cases} -2\theta + \frac{3}{2}\theta^2 + (2\theta - \frac{3}{2}\theta^2) \;\equiv\; 0 & \theta \leqslant 0 \\ -2(1-\theta)^2 + \frac{1}{2}(2-\theta)^2 \;=\; 2\theta - \frac{3}{2}\theta^2 & 0 \leqslant \theta \leqslant 1 \\ \frac{1}{2}(2-\theta)^2 & 1 \leqslant \theta \leqslant 2 \\ 0 & \theta \geqslant 2 \end{cases} . \end{aligned} \tag{4.12}$$

5/10 5/11 *Remark.* $K(\theta) = 0$ for $\theta \notin [0,2]$, since then e_L acts on a quadratic polynomial.

4.3.2 Where does $K(\theta)$ vanish? (*Unlectured*)

We can extend the range of θ in the definition of $K(\theta) = \lambda((x-\theta)_+^k)$ in (4.7b) from $a \leqslant \theta \leqslant b$ to $\theta \in \mathbb{R}$. For instance, the example (4.10) retains the property that $K(\theta)=0$ for $\theta \leqslant a$ and $\theta \geqslant b$.

Indeed, suppose that the interval $[\alpha,\beta]$ is the shortest sub-interval of $[a,b]$ such that $\lambda(f)$ is independent of $\{f(x) : a \leqslant x < \alpha\}$ and $\{f(x) : \beta < x \leqslant b\}$. Then we can replace the definition of the Peano kernel function (4.7b) by the analogous formula

$$K(\theta) = \lambda((x-\theta)_+^k)\,, \quad \text{where} \quad \theta \in \mathbb{R} \quad \text{and} \quad x \in [\alpha,\beta]\,. \tag{4.13}$$

Hence, if $\theta \geqslant \beta$, λ is applied to the zero function, which gives $K(\theta)=0$ due to the linearity of λ. On the other hand if $\theta \leqslant \alpha$, then the '+' subscript can be removed from expression (4.13), so that $K(\theta)$ is the result of applying λ to the polynomial $(x-\theta)^k$, $x \in \mathbb{R}$. However, the statement of Peano kernel theorem includes the condition that $\lambda(f) = 0$, $f \in \mathbb{P}_k$, so we conclude that $K(\theta) = 0$ for $\theta \leqslant \alpha$. These remarks establish much of the following lemma.

Lemma 4.3. *Let the function $K(\theta)$ be defined by the linear functional λ through* (4.13)*. Then $K(\theta)$ is zero for $\theta \geqslant \beta$. Further, $K(\theta)$ is zero for all $\theta \leqslant \alpha$ if and only if λ has the property $\lambda(f)=0$ for $f \in \mathbb{P}_k$.*

Proof. Because of previous remarks, we have only to prove that $K(\theta)=0$ for $\theta \leqslant \alpha$, implies that $\lambda(f)=0$ for $f \in \mathbb{P}_k$. Now, due to the linearity of λ, we can write (4.13) when $\theta \leqslant \alpha$ in the form

$$\begin{aligned} K(\theta) &= \lambda\left(\sum_{j=0}^{k} \binom{k}{j} x^j (-\theta)^{k-j}\right) \\ &= \sum_{j=0}^{k} \binom{k}{j} (-\theta)^{k-j} \lambda\left(x^j\right)\,. \end{aligned} \tag{4.14}$$

Thus since $K(\theta)$ for $\theta \leqslant \alpha$ is a polynomial of degree at most k, if it vanishes identically, every coefficient of a power of θ must be zero. We conclude from (4.14) that $\lambda\left(x^j\right) = 0$ for $\alpha \leqslant x \leqslant \beta$ and $j=0,1,\ldots,k$. Since these powers of x provide a basis of $\mathbb{P}_k$, we have that $\lambda(f)=0$ for $f \in \mathbb{P}_k$. □

Remark. This lemma is sometimes useful when one applies the Peano kernel theorem. Indeed, if the function (4.13) fails to vanish for $\theta \leqslant \alpha$, then either the conditions of the theorem do not hold, or a mistake has occurred in the calculation of the kernel function K!

4.4 Estimate of the error $e_L(f)$ when $K(\theta)$ does not change sign

Theorem 4.4. *Suppose that K does not change sign in (a,b) and that $f \in C^{k+1}[a,b]$, then*

$$e_L(f) = \frac{1}{k!}\left(\int_a^b K(\theta)\,\mathrm{d}\theta\right) f^{(k+1)}(\xi) \quad \textit{for some} \quad \xi \in (a,b)\,. \tag{4.15}$$

Proof. Suppose that $K \geqslant 0$. Then

$$e_L(f) \leqslant \frac{1}{k!}\int_a^b K(\theta) \max_{x\in[a,b]} f^{(k+1)}(x)\,\mathrm{d}\theta = \frac{1}{k!}\left(\int_a^b K(\theta)\,\mathrm{d}\theta\right) \max_{x\in[a,b]} f^{(k+1)}(x).$$

Likewise

$$e_L(f) \geqslant \frac{1}{k!}\left(\int_a^b K(\theta)\,\mathrm{d}\theta\right) \min_{x\in[a,b]} f^{(k+1)}(x)\,.$$

Consequently

$$\min_{x\in[a,b]} f^{(k+1)}(x) \leqslant \frac{e_L(f)}{\frac{1}{k!}\int_a^b K(\theta)\,\mathrm{d}\theta} \leqslant \max_{x\in[a,b]} f^{(k+1)}(x)\,.$$

The required result follows from the intermediate value theorem. Similar analysis pertains to the case $K \leqslant 0$. □

Examples.

(i) In the case of (4.10), $K \leqslant 0$, and $\int_\alpha^\beta K(\theta)\,\mathrm{d}\theta = \int_\alpha^\beta (\alpha-\theta)(\beta-\theta)\,\mathrm{d}\theta = -\frac{1}{6}(\beta-\alpha)^3$. Hence

$$e_L(f) = -\tfrac{1}{12}(\beta-\alpha)^3 f'''(\xi) \quad \text{for some} \quad \xi \in (\alpha,\beta)\,. \tag{4.16a}$$

(ii) *Unlectured.* In the case of (4.12), $K\geqslant 0$ and

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

Consequently

$$e_L(f) = \tfrac{1}{2!} \times \tfrac{2}{3} f'''(\xi) = \tfrac{1}{3} f'''(\xi) \quad \text{for some} \quad \xi \in (0,2)\,. \tag{4.16b}$$

Comments.

(i) We note that $\frac{1}{k!}\int_a^b K(\theta)\,\mathrm{d}\theta$ is independent of f.

(ii) As an alternative to determining $\int_a^b K(\theta)\,\mathrm{d}\theta$ analytically, we can take advantage of the remark that, if equation (4.7a) is valid for all $f \in C^{k+1}[a,b]$, then it holds in the particular case $f(x)=x^{k+1}$. Since $f^{(k+1)}(\theta) = (k+1)!$, we deduce from (4.7a) that

$$\int_a^b K(\theta)\,d\theta = \frac{1}{k+1}\, e_L\left(x^{k+1}\right)\,. \tag{4.17}$$

Further, $e_L(p)=0$, $p\in\mathbb{P}_k$. This implies that (4.17) remains true if x^{k+1} is replaced by any monic polynomial of degree $k+1$, say p_{k+1}, for which the evaluation of $e_L(p_{k+1})$ is straightforward.

4.5 Bounds on the error $|e_L(f)|$

We can measure the 'size' of a function g in various manners. Popular choices include

1-norm:
$$\|g\|_1 = \int_a^b |g(x)|\,\mathrm{d}x\,; \tag{4.18a}$$

2-norm:
$$\|g\|_2 = \left\{\int_a^b [g(x)]^2\,\mathrm{d}x\right\}^{1/2}\,; \tag{4.18b}$$

∞-norm:
$$\|g\|_\infty = \max_{x\in[a,b]} |g(x)|\,. \tag{4.18c}$$

Using these definitions we can bound the size of the error in our approximation procedures.

(i) From (4.18a) and (4.18c) it follows that

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

Thence from (4.7a) we deduce that

$$|e_L(f)| \leqslant \frac{1}{k!}\,\|K\|_\infty \int_a^b |f^{(k+1)}(\theta)|\,\mathrm{d}\theta = \frac{1}{k!}\|K\|_\infty\,\|f^{(k+1)}\|_1\,, \tag{4.19a}$$

and similarly

$$|e_L(f)| \leqslant \frac{1}{k!}\int_a^b |K(\theta)|\,\mathrm{d}\theta\,\|f^{(k+1)}\|_\infty = \frac{1}{k!}\|K\|_1\,\|f^{(k+1)}\|_\infty\,. \tag{4.19b}$$

(ii) The *Cauchy–Schwarz inequality* states

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

It follows from (4.7a) that

$$|e_L(f)| \leqslant \frac{1}{k!}\|K\|_2\|f^{(k+1)}\|_2\,. \tag{4.20}$$

Remarks.

(i) The inequalities (4.19b), (4.19a) and (4.20) are valid whether or not $K(\theta)$ changes sign in $a\leqslant\theta\leqslant b$ (cf. (4.15)).

(ii) For the specific choice $f = x^{k+1}$, as above $f^{(k+1)} = (k+1)! = \|f^{(k+1)}\|_\infty$. Hence *for this choice*

$$e_L(f) = \frac{1}{k!}\int_a^b K(\theta)\,\mathrm{d}\theta\,\|f^{(k+1)}\|_\infty\,. \tag{4.21}$$

It follows that if K does not change sign, (4.19b) is *sharp*.

4.5.1 Examples (Unlectured)

(i) For example (4.10), we have from using (4.16a) that (cf. (4.19b))

$$|e_L(f)| \leqslant \tfrac{1}{12}|\beta-\alpha|^3\|f'''\|_\infty\,. \tag{4.22a}$$

(ii) For example (4.12), we have from using (4.16b) that (cf. (4.19b))

$$|e_L(f)| \leqslant \tfrac{1}{3}\|f'''\|_\infty\,. \tag{4.22b}$$

(iii) For Simpson's rule (see the table on page 20)

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

it follows, given that Simpson's rule is exact for quadratics (as well as cubics), that an expression for the error is

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

where, from (4.7b),

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

Now, $K(\theta)$ changes its sign at $\theta = 0$, so its L_1-norm has the value

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

so that from (4.19b)

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

5 Ordinary Differential Equations

5.1 Introduction

The aim of this section is to discuss some [elementary] methods that approximate the exact solution of the *ordinary differential equation (ODE)*

$$\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}), \qquad 0 \leqslant t \leqslant T, \tag{5.1}$$

where $\boldsymbol{y} \in \mathbb{R}^N$, $T \in \mathbb{R}$ and the function $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}^N$ is sufficiently 'smooth' or 'nice'. In principle, it is enough for $\boldsymbol{f}$ to be 'Lipschitz' (with respect to the second argument) to ensure that the solution exists and is unique.

Definition 5.1. A function $\boldsymbol{f}(t, \boldsymbol{y})$ is said to satisfy a *Lipschitz condition* of order α at $\boldsymbol{y} = \boldsymbol{x}$ if there exists $\delta > 0$ such that

$$\|\boldsymbol{f}(t, \boldsymbol{y}) - \boldsymbol{f}(t, \boldsymbol{x})\| \leqslant \lambda \|\boldsymbol{y} - \boldsymbol{x}\|^\alpha \quad \text{for the given } \boldsymbol{x} \text{ and for all } \|\boldsymbol{y} - \boldsymbol{x}\| < \delta, \tag{5.2a}$$

where $\lambda > 0$ and $\alpha > 0$ are independent of $\boldsymbol{y}$.

We will assume, at the minimum, that there exists $\lambda > 0$ such that

$$\|\boldsymbol{f}(t, \boldsymbol{y}) - \boldsymbol{f}(t, \boldsymbol{x})\| \leqslant \lambda \|\boldsymbol{y} - \boldsymbol{x}\| \quad \text{for } t \in [0, T], \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N. \tag{5.2b}$$

However, for simplicity, we will often further assume that $\boldsymbol{f}$ is analytic so that we are always able to expand locally into Taylor series.

In addition to equation (5.1) we also need an initial condition, and we shall assume that

$$\boldsymbol{y}(0) = \boldsymbol{y}_0 \,. \tag{5.3}$$

Hence, given we know $\boldsymbol{y}$ and its slope $\boldsymbol{y}'$ at $t = 0$, can we obtain an approximation to $\boldsymbol{y}(h)$, say $\boldsymbol{y}(t_1)$, where the *time step* $h > 0$ is small? If so, can we subsequently obtain approximations $\boldsymbol{y}_n \approx \boldsymbol{y}(t_n)$, $n = 2, \ldots$, where $t_n = nh$?

5.2 One-step methods

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

Definition 5.2 (*A one-step method*)**.** This is a map

$$\boldsymbol{y}_{n+1} = \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}_n), \tag{5.4}$$

i.e. an algorithm which allows $\boldsymbol{y}_{n+1}$ to depend only on t_n, $\boldsymbol{y}_n$, h and $\boldsymbol{f}$ (through the ODE (5.1)).

5.2.1 The Euler method

Given that we know $\boldsymbol{y}$ and its slope $\boldsymbol{y}'$ at $t = 0$ then, if we wish to approximate $\boldsymbol{y}$ at $t = h > 0$, the most obvious approach is to truncate the Taylor series

$$\boldsymbol{y}(h) = \boldsymbol{y}(0) + h\boldsymbol{y}'(0) + \tfrac{1}{2}h^2\boldsymbol{y}''(0) + \cdots \tag{5.5a}$$

at the $\mathcal{O}(h^2)$ term. We see from (5.1) that $\boldsymbol{y}'(0) = \boldsymbol{f}(t_0, \boldsymbol{y}_0)$, hence this procedure approximates $\boldsymbol{y}(h)$ by $\boldsymbol{y}_0 + h\boldsymbol{f}(t_0, \boldsymbol{y}_0)$, and we thus have the approximant

$$\boldsymbol{y}_1 = \boldsymbol{y}_0 + h\boldsymbol{f}(t_0, \boldsymbol{y}_0) \,. \tag{5.5b}$$

By the same token, we may advance from h to $2h$ by we treating $\boldsymbol{y}_1$ as the initial condition, and letting $\boldsymbol{y}_2 = \boldsymbol{y}_1 + h\boldsymbol{f}(t_1, \boldsymbol{y}_1)$. In general, we obtain the *Euler method*

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n), \qquad n = 0, 1, \ldots, \tag{5.6}$$

which on $[t_n, t_{n+1}]$ approximates $\boldsymbol{y}(t)$ with a straight line with slope $\boldsymbol{f}(t_n, \boldsymbol{y}_n)$.

Question. How good is the Euler method? Inasmuch as its derivation is intuitive, we need to underpin it with theory. An important question is:

Does the method (5.6) *converge to the exact solution as* $h \to 0$*?*

Definition 5.3 (*Convergence*)**.** Let $T > 0$ be given, and suppose that, for every $h > 0$, a method produces the solution sequence $\boldsymbol{y}_n = \boldsymbol{y}_n(h)$, $n = 0, 1, \ldots, \lfloor T/h \rfloor$. We say that the method *converges* if, as $h \to 0$ and $n_k(h)h \xrightarrow{k\to\infty} t$,

$$\boldsymbol{y}_{n_k} \to \boldsymbol{y}(t) \quad \text{uniformly for } t \in [0, T],$$

where $\boldsymbol{y}(t)$ is the exact solution of (5.1).

Theorem 5.4. *Let* $\boldsymbol{f}$ *be a Lipschitz function of order one with the Lipschitz constant* $\lambda \geqslant 0$ *for* $t \in [0, T]$*, i.e. suppose that* $\boldsymbol{f}$ *satisfies* (5.2b)*, then the Euler method* (5.6) *converges, i.e. for every* $t \in [0, T]$*,*

$$\lim_{\substack{h\to 0\\ nh\to t}} \boldsymbol{y}_n = \boldsymbol{y}(t). \tag{5.7a}$$

Further, let $\boldsymbol{e}_n = \boldsymbol{y}_n - \boldsymbol{y}(t_n)$ *be the error at step* $n \in \mathbb{Z}^+$*, where* $0 \leqslant n \leqslant T/h$*; then for some positive constant* $c \in \mathbb{R}$*,*

6/13
6/14

$$\|\boldsymbol{e}_n\| \leqslant ch\, \frac{e^{\lambda T} - 1}{\lambda}. \tag{5.7b}$$

Proof. The Taylor series for the exact solution $\boldsymbol{y}$ gives, by using the identity (5.1), $\boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y}(t))$,

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

where

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

By subtracting formula (5.6) from this equation, we find that the errors are related by the condition

$$\boldsymbol{e}_{n+1} = \boldsymbol{y}_{n+1} - \boldsymbol{y}(t_{n+1}) = [\boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n)] - [\boldsymbol{y}(t_n) + h\boldsymbol{f}(t_n, \boldsymbol{y}(t_n) + \boldsymbol{R}_n]\,.$$

The remainder term, $\boldsymbol{R}_n$, can be bounded *uniformly* (e.g., in the *Euclidean norm* or other underlying norm $\|\cdot\|$) *for all* $[0, T]$ by ch^2, for some positive constant $c \in \mathbb{R}$. For instance,

$$\|\boldsymbol{R}_n\|_\infty \leqslant \int_{t_n}^{t_{n+1}} (t_{n+1} - \theta)\|\boldsymbol{y}''(\theta)\|_\infty\, d\theta \leqslant \tfrac{1}{2}h^2\|\boldsymbol{y}''\|_\infty\,,$$

in which case $c = \frac{1}{2}\|\boldsymbol{y}''\|_\infty$. Thus, using the triangle inequality and the Lipschitz condition (5.2b)

$$\begin{aligned}
\|\boldsymbol{e}_{n+1}\| &\leqslant \|\boldsymbol{y}_n - \boldsymbol{y}(t_n)\| + h\|\boldsymbol{f}(t_n, \boldsymbol{y}_n) - \boldsymbol{f}(t_n, \boldsymbol{y}(t_n))\| + ch^2 \\
&\leqslant \|\boldsymbol{y}_n - \boldsymbol{y}(t_n)\| + h\lambda\|\boldsymbol{y}_n - \boldsymbol{y}(t_n)\| + ch^2 \\
&\leqslant (1 + h\lambda)\|\boldsymbol{e}_n\| + ch^2.
\end{aligned}$$

Consequently, by induction,

$$\|\boldsymbol{e}_{n+1}\| \leqslant (1 + h\lambda)^m\|\boldsymbol{e}_{n+1-m}\| + ch^2\sum_{j=0}^{m-1}(1 + h\lambda)^j, \qquad m = 0, 1, \ldots, n+1.$$

In particular, letting $m = n + 1$ and bearing in mind that $\boldsymbol{e}_0 = \boldsymbol{0}$, we have that

$$\|\boldsymbol{e}_{n+1}\| \leqslant ch^2\sum_{j=0}^{n}(1 + h\lambda)^j = ch^2\frac{(1 + h\lambda)^{n+1} - 1}{(1 + h\lambda) - 1} \leqslant \frac{ch}{\lambda}\left((1 + h\lambda)^{n+1} - 1\right),$$

or equivalently (by a re-labelling)

$$\|\boldsymbol{e}_n\| \leqslant \frac{ch}{\lambda}\left((1 + h\lambda)^n - 1\right). \tag{5.8}$$

Further, $0 < 1 + h\lambda \leqslant e^{h\lambda}$ since $h > 0$, and $nh \leqslant T$, hence $(1 + h\lambda)^n \leqslant e^{\lambda T}$. Thus

$$\|\boldsymbol{e}_n\| \leqslant ch\, \frac{e^{\lambda T} - 1}{\lambda} \xrightarrow{h\to 0} 0$$

6/10
6/11
6/12

uniformly for $0 \leqslant nh \leqslant T$, and the theorem is true. □

Remark. We have left the error bound in a form including the '-1' (although this is strictly unnecessary), in order to make it clear that as $\lambda \to 0$, the bound tends to chT, as it should.

Local truncation error. The *local truncation error* of a general numerical method

$$\boldsymbol{y}_{n+1} = \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}_0, \boldsymbol{y}_1, \dots, \boldsymbol{y}_n) \tag{5.9a}$$

for the solution of (5.1) is the error of the method relative to the true solution, i.e. the value $\boldsymbol{\eta}_{n+1}$ such that

$$\boldsymbol{\eta}_{n+1} = \boldsymbol{y}(t_{n+1}) - \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}(t_0), \boldsymbol{y}(t_1), \dots, \boldsymbol{y}(t_n))\,. \tag{5.9b}$$

Order. The *order* of the method is the largest integer $p \geqslant 0$ such that

$$\boldsymbol{\eta}_{n+1} = \boldsymbol{y}(t_{n+1}) - \boldsymbol{\varphi}_h(t_n, \boldsymbol{y}(t_0), \boldsymbol{y}(t_1), \dots, \boldsymbol{y}(t_n)) = O(h^{p+1}) \tag{5.10}$$

for all $h > 0$, $n \geqslant 0$ and for all sufficiently smooth functions $\boldsymbol{f}$ in (5.1).

Remarks. The order is one less than the power of h in the $\mathcal{O}(\,\cdot\,)$ term, so as to account for the fact that there are $\sim h^{-1}$ steps in $[0, T]$. The order shows how good the method is locally. Later we shall show that unless $p \geqslant 1$ the 'method' is an unsuitable approximation to (5.1): in particular, $p \geqslant 1$ is necessary for convergence (see Theorem (5.9)).

The order of Euler's method. For Euler's method, (5.6),

$$\boldsymbol{\varphi}_h(t, \boldsymbol{y}) = \boldsymbol{y} + h\boldsymbol{f}(t, \boldsymbol{y}).$$

Substituting the exact solution of (5.1), we obtain from the Taylor theorem

$$\begin{aligned}\boldsymbol{y}(t_{n+1}) - [\boldsymbol{y}(t_n) + h\boldsymbol{f}(t_n, \boldsymbol{y}(t_n))] &= [\boldsymbol{y}(t_n) + h\boldsymbol{y}'(t_n) + \tfrac{1}{2}h^2\boldsymbol{y}''(t_n) + \dots] - [\boldsymbol{y}(t_n) + h\boldsymbol{y}'(t_n)] \\ &= \mathcal{O}\big(h^2\big)\ . \end{aligned} \tag{5.11}$$

We deduce that Euler's method is of order 1.

5.2.2 Theta methods

Definition 5.5 (*Theta methods*)**.** One-step methods of the form

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h[\theta\boldsymbol{f}(t_n, \boldsymbol{y}_n) + (1-\theta)\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})], \qquad n = 0, 1, \dots, \tag{5.12}$$

where $\theta \in [0, 1]$ is a parameter, are known as *theta methods.*

Remarks.

(i) If $\theta = 1$, we recover Euler's method.

(ii) The choices $\theta = 0$ and $\theta = \frac{1}{2}$ are known respectively as

$$\text{Backward Euler:} \qquad \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}), \tag{5.13a}$$

$$\text{Trapezoidal rule:} \qquad \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \tfrac{1}{2}h[\boldsymbol{f}(t_n, \boldsymbol{y}_n) + \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})]. \tag{5.13b}$$

(iii) If $\theta \in [0, 1)$ then the *theta method* (5.12) is said to be *implicit*, because in order to find the unknown vector $\boldsymbol{y}_{n+1}$ we need to solve a [in general, nonlinear] algebraic system of N equations. The solution of nonlinear algebraic equations can be done by iteration. For example, for backward Euler, and letting $\boldsymbol{y}_{n+1}^{[0]} = \boldsymbol{y}_n$, we may use

Direct iteration $\qquad \boldsymbol{y}_{n+1}^{[j+1]} = \boldsymbol{y}_n + h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}^{[j]});$

Newton–Raphson: $\qquad \boldsymbol{y}_{n+1}^{[j+1]} = \boldsymbol{y}_{n+1}^{[j]} - \left[I - h\dfrac{\partial \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}^{[j]})}{\partial \boldsymbol{y}}\right]^{-1} [\boldsymbol{y}_{n+1}^{[j]} - \boldsymbol{y}_n - h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}^{[j]})];$

Modified Newton–Raphson: $\qquad \boldsymbol{y}_{n+1}^{[j+1]} = \boldsymbol{y}_{n+1}^{[j]} - \left[I - h\dfrac{\partial \boldsymbol{f}(t_n, \boldsymbol{y}_n)}{\partial \boldsymbol{y}}\right]^{-1} [\boldsymbol{y}_{n+1}^{[j]} - \boldsymbol{y}_n - h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}^{[j]})].$

We will return to this topic later.

The order of the theta method. It follows from (5.10), (5.12) and Taylor's theorem that

$$\begin{aligned}\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}(t_n) - h[\theta\boldsymbol{y}'(t_n) + (1-\theta)\boldsymbol{y}'(t_{n+1})] \\ =& [\boldsymbol{y}(t_n) + h\boldsymbol{y}'(t_n) + \tfrac{1}{2}h^2\boldsymbol{y}''(t_n) + \tfrac{1}{6}h^3\boldsymbol{y}'''(t_n)] - \boldsymbol{y}(t_n) - \theta h\boldsymbol{y}'(t_n) \\ &- (1-\theta)h[\boldsymbol{y}'(t_n) + h\boldsymbol{y}''(t_n) + \tfrac{1}{2}h^2\boldsymbol{y}'''(t_n)] + \mathcal{O}\big(h^4\big) \\ =& (\theta - \tfrac{1}{2})h^2\boldsymbol{y}''(t_n) + (\tfrac{1}{2}\theta - \tfrac{1}{3})h^3\boldsymbol{y}'''(t_n) + \mathcal{O}\big(h^4\big)\,. \end{aligned} \tag{5.13c}$$

Therefore the theta method is in general of order 1, except that the trapezoidal rule is of order 2.

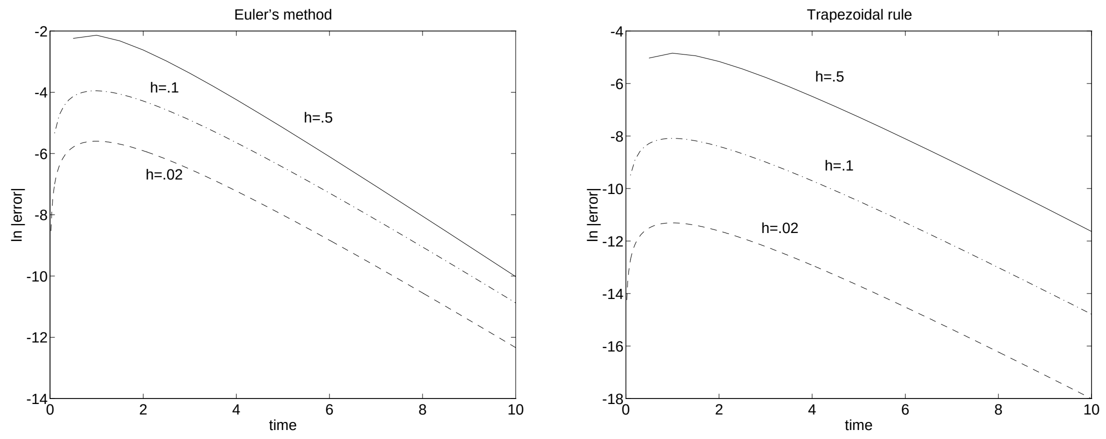


Figure 5.2: Error between the numerical solution and the exact solution of the equation $y' = -y$, $y(0) = 1$ for both Euler's method (first order) and the trapezoidal method (second order).

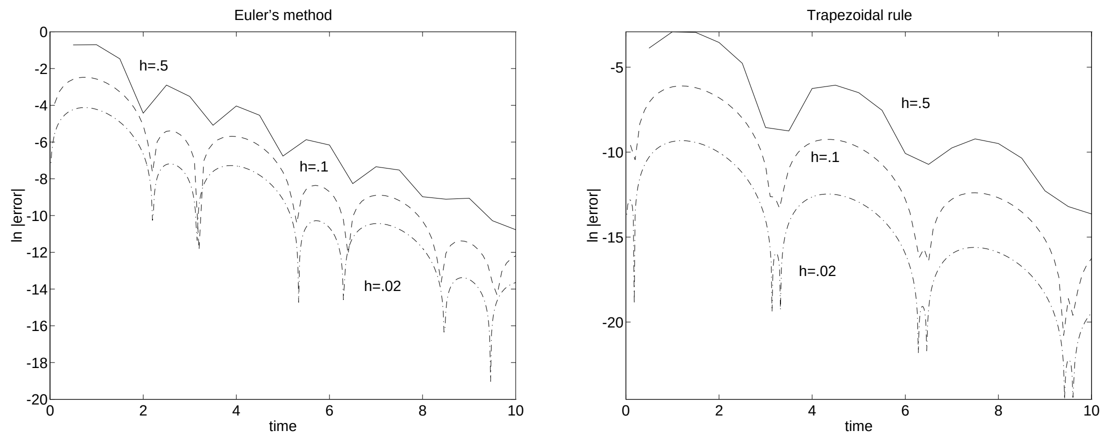


Figure 5.3: Error between the numerical solution and the exact solution of the equation $y' = -y + 2e^{-t}\cos 2t$, $y(0) = 0$ for both Euler's method (first order) and the trapezoidal method (second order).

5.3 Multistep methods

It is often useful to use past solution values to enhance the quality of approximation in the computation of a new value, e.g. the 2-step *Adams–Bashforth*[6] method

$$\boldsymbol{y}_{n+2} = \boldsymbol{y}_{n+1} + \tfrac{1}{2}h(3\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}) - \boldsymbol{f}(t_n, \boldsymbol{y}_n))\,. \tag{5.14}$$

Definition 5.6 (*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

$$\sum_{\ell=0}^{s} \rho_\ell \boldsymbol{y}_{n+\ell} = h \sum_{\ell=0}^{s} \sigma_\ell \boldsymbol{f}(t_{n+\ell}, \boldsymbol{y}_{n+\ell}), \qquad n = 0, 1, \ldots, \tag{5.15}$$

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

[6] Adams, as in Adams Road.

Remark. If $s \geqslant 2$, we need to obtain extra *starting values* $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{s-1}$ by a different time-stepping method.

5.3.1 The order of a multistep method

Theorem 5.7. *The multistep method* (5.15) *is of order* $p \geqslant 1$ *if and only if* [7]

$$\sum_{\ell=0}^{s} \rho_\ell = 0\,, \quad \textit{and} \quad \sum_{\ell=0}^{s} \rho_\ell \ell^k = k \sum_{\ell=0}^{s} \sigma_\ell \ell^{k-1} \quad \textit{for} \quad k = 1, \ldots, p. \tag{5.16}$$

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

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

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 (5.16) 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 (5.15) is of order p iff

$$\sum_{\ell=0}^{s} \rho_\ell\, Q(t_{n+\ell}) = h \sum_{\ell=0}^{s} \sigma_\ell Q'(t_{n+\ell})\,, \quad \forall\, Q \in \mathbb{P}_p\,. \tag{5.17}$$

In particular taking $Q(x) = x^k$ for $k = 0, \ldots, p$, $t_{n+\ell} = \ell$ and $h = 1$, we obtain (5.16).

(ii) If the desire is to have a order p method, then (5.16) might be viewed as $p+1$ equations for the $2s+1$ variables ρ_ℓ ($\ell = 0, \ldots, s-1$) and σ_ℓ ($\ell = 0, \ldots, s$). A key question is how to choose the ρ_ℓ and σ_ℓ (given that if $2s > p$ there is some wriggle room).

Example: the 2-step Adams–Bashforth method. The 2-step Adams–Bashforth method is (see (5.14))

$$\boldsymbol{y}_{n+2} - \boldsymbol{y}_{n+1} = h\left(\tfrac{3}{2}\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}) - \tfrac{1}{2}\boldsymbol{f}(t_n, \boldsymbol{y}_n)\right). \tag{5.18}$$

Hence $\rho_0 = 0$, $\rho_1 = -1$, $\rho_2 = 1$, $\sigma_0 = -\frac{1}{2}$, $\sigma_1 = \frac{3}{2}$, $\sigma_2 = 0$, and

$$\begin{aligned}\sum_{\ell=0}^{2} \rho_\ell &= 0 - 1 + 1 = 0\,, \\ \sum_{\ell=0}^{2} \rho_\ell \ell - \sum_{\ell=0}^{2} \sigma_\ell \ell^0 &= (0 - 1 + 2) - \left(-\tfrac{1}{2} + \tfrac{3}{2} + 0\right) = 0\,, \\ \sum_{\ell=0}^{2} \rho_\ell \ell^2 - 2\sum_{\ell=0}^{2} \sigma_\ell \ell &= \left(0 - 1 + 2^2\right) - 2\left(0 + \tfrac{3}{2} + 0\right) = 0\,, \\ \sum_{\ell=0}^{2} \rho_\ell \ell^3 - 3\sum_{\ell=0}^{2} \sigma_\ell \ell^2 &= \left(0 - 1 + 2^3\right) - 3\left(0 + \tfrac{3}{2} + 0\right) = \tfrac{5}{2} \neq 0\,.\end{aligned}$$

7/14 Hence the 2-step Adams–Bashforth method is of order 2.

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

Question. Is there an easier way to check order?

Answer. Arguably. First we define two polynomials of degree s for a Given a multistep method (5.15):

$$\rho(w) = \sum_{\ell=0}^{s} \rho_\ell w^\ell \quad \text{and} \quad \sigma(w) = \sum_{\ell=0}^{s} \sigma_\ell w^\ell \,. \tag{5.19}$$

Theorem 5.8. *The multistep method* (5.15) *is of order* $p \geqslant 1$ *iff*

$$\rho(\mathrm{e}^z) - z\sigma(\mathrm{e}^z) = \mathcal{O}\big(z^{p+1}\big) \quad \textit{as} \quad z \to 0. \tag{5.20}$$

Proof. Expanding again in Taylor series,

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

The theorem follows from (5.16). □

Remarks

(i) The reason that (5.20) is equivalent to (5.16) (and their proofs are almost identical) is because of the relation between Taylor series and the exponent e^{hD}, where D is the differentiation operator. Namely

$$\boldsymbol{f}(x+h) = (I + hD + \tfrac{1}{2}h^2 D^2 + \ldots)\boldsymbol{f}(x) = \mathrm{e}^{hD}\boldsymbol{f}(x) \,. \tag{5.21}$$

(ii) An equivalent statement of the theorem is that the multistep method (5.15) is of order $p \geqslant 1$ iff

$$\rho(w) - (\log w)\sigma(w) = \mathcal{O}\big(|w-1|^{p+1}\big) \,, \quad \text{as} \quad w \to 1. \tag{5.22}$$

Example: the 2-step Adams–Bashforth method. For the 2-step Adams–Bashforth method (5.18),

$$\rho(w) = w^2 - w \,, \quad \sigma(w) = \tfrac{3}{2}w - \tfrac{1}{2} \,, \tag{5.23a}$$

and hence

$$\begin{aligned}
\rho(\mathrm{e}^z) - z\sigma(\mathrm{e}^z) &= \big[1 + 2z + 2z^2 + \tfrac{4}{3}z^3\big] - \big[1 + z + \tfrac{1}{2}z^2 + \tfrac{1}{6}z^3\big] - \tfrac{3}{2}z\,\big[1 + z + \tfrac{1}{2}z^2\big] + \tfrac{1}{2}z + \mathcal{O}\big(z^4\big) \\
&= \tfrac{5}{12}z^3 + \mathcal{O}\big(z^4\big) \,.
\end{aligned} \tag{5.23b}$$

7/13 As before, we conclude that the 2-step Adams–Bashforth method is of order 2.

5.3.2 The convergence of multistep methods

Example: absence of convergence. Consider the 2-step method

$$\boldsymbol{y}_{n+2} + 4\boldsymbol{y}_{n+1} - 5\boldsymbol{y}_n = h(4\boldsymbol{f}_{n+1} + 2\boldsymbol{f}_n) \tag{5.24}$$

Now $\rho(w) = w^2 + 4w - 5$, $\sigma(w) = 4w + 2$, and it can be verified that the method is of order 3. Apply this method to the trivial ODE

$$y' = 0 \,, \quad y(0) = 1 \,. \tag{5.25}$$

Then a single step reads

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

The general solution of this recursion is

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

where c_1 and c_2 are determined by $y_0 = c_1 + c_2 = 1$ and the value of y_1.

Remark. Although (5.25) is a first-order ODE, (5.24) is a 2-step method and so we need two pieces of information to fix both c_1 and c_2, e.g. the value of y_1 in addition to the value of y_0.

If $y_1 \neq 1$, i.e. if there is a small error in the starting values, then $c_2 \neq 0$. This has an important consequence, for suppose that $h \to 0$ such that $nh \to t > 0$. Then $n \to \infty$, which implies that $|y_n| \to \infty$ if $c_2 \neq 0$, and hence *we cannot recover the exact solution* $y(t) \equiv 1$.

Remark. This can remain true in a calculation on a computer, even if we force $c_2 = 0$ by our choice of y_1, because of the presence of round-off errors.

We deduce that *the method* (5.24) *does not converge!* As a more general point, it is important to realise that many 'plausible' multistep methods may not be convergent and we need a theoretical tool to allow us to check for this feature.

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

7/10
7/11
7/12

Remark. Unless a method is convergent, *do not use it.*

Definition. We say that a polynomial, $\rho(w)$, obeys the *root condition* if all its zeros reside in $|w| \leqslant 1$ and all zeros of unit modulus are simple.

Theorem 5.9 (The Dahlquist equivalence theorem)**.** *The multistep method* (5.15) *is convergent iff it is of order $p \geqslant 1$ and the polynomial ρ obeys the root condition.*

Proof. See Part III. □

Definition. If ρ obeys the root condition, the multistep method (5.15) is sometimes said to be *zero-stable:* we will not use this terminology.

Examples. For the Adams–Bashforth method (5.18) we have $\rho(w) = (w-1)w$ and the root condition is obeyed. However, for (5.24) we obtain $\rho(w) = (w-1)(w+5)$, the root condition fails and it follows that there is no convergence.

5.3.3 Maximising order

Subject to convergence, it is frequently a good idea to maximise order. A useful procedure to generate multistep methods which are convergent and of high order is as follows.

First put $z = 0$ in (5.20), or $w = 1$ in (5.22), to deduce from (5.16) that order $p \geqslant 1$ implies that

$$\rho(1) = 0\,. \tag{5.26}$$

Next choose an arbitrary s-degree polynomial ρ that obeys the root condition and is such that $\rho(1) = 0$. To maximize order, we let σ be the s-degree (alternatively, $(s-1)$-degree for explicit methods) polynomial arising from the truncation of the Taylor expansion about the point $w = 1$ of

$$\frac{\rho(w)}{\log w}\,.$$

For example, suppose $\sigma(w)$ is the s-degree polynomial for an *implicit method*, then

$$\sigma(w) = \frac{\rho(w)}{\log w} + \mathcal{O}\big(|w-1|^{s+1}\big)\,, \quad \text{which implies that} \quad \rho(\mathrm{e}^z) - z\sigma(\mathrm{e}^z) = \mathcal{O}\big(z^{s+2}\big)\,,$$

which then implies from (5.20) that the method has an order of at least $s + 1$.

5.3.4 Adams methods

The choice $\rho(w) = w^{s-1}(w-1)$ corresponds to *Adams methods.*

Adams–Bashforth methods. If $\sigma_s = 0$, then the method is explicit and of order s (e.g. (5.14) for $s = 2$).

Adams–Moulton methods. If $\sigma_s \neq 0$, then the method is implicit and of order $(s+1)$. For example, letting $s = 2$ and $\xi = w - 1$, we obtain the 3rd-order Adams–Moulton method by expanding

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

Therefore the 2-step, 3rd-order Adams–Moulton method is

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

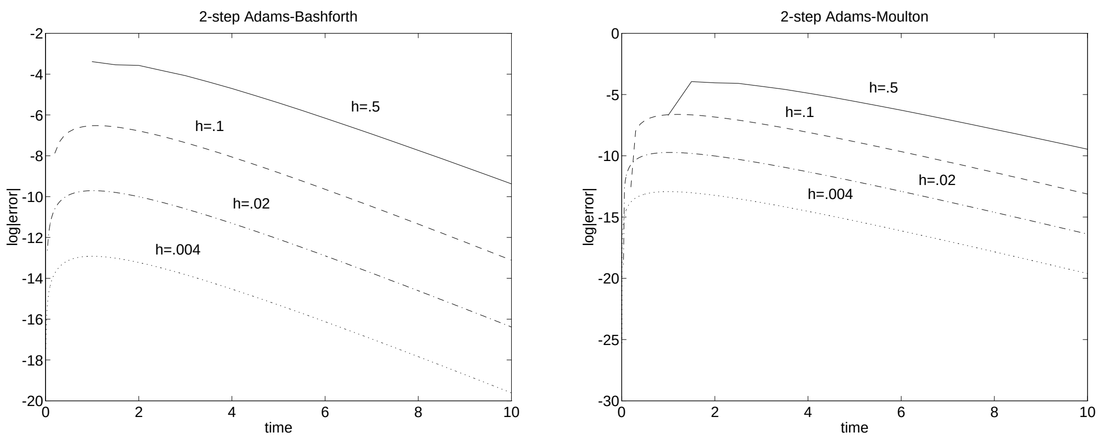


Figure 5.4: Error between the numerical solution and the exact solution of the equation $y' = -y$, $y(0) = 1$ for both the 2-step Adams–Bashforth method (second order) and the 2-step Adams–Moulton method (third order).

5.3.5 BDF methods

For reasons that will become clear later (see page 46), we wish to consider s-step, s-order methods such that $\sigma(w) = \sigma_s w^s$ for some $\sigma_s \in \mathbb{R} \setminus \{0\}$. Hence, from (5.15),

$$\sum_{\ell=0}^{s} \rho_\ell \boldsymbol{y}_{n+\ell} = h\sigma_s \boldsymbol{f}(t_{n+s}, \boldsymbol{y}_{n+s}), \qquad n = 0, 1, \ldots. \tag{5.27}$$

Such methods are called *backward differentiation formulae (BDF).*

Lemma 5.10. *The form of the s-step BDF method is*

$$\rho(w) = \sigma_s \sum_{\ell=1}^{s} \frac{1}{\ell} w^{s-\ell}(w-1)^\ell, \qquad \text{where} \qquad \sigma_s = \left(\sum_{\ell=1}^{s} \frac{1}{\ell}\right)^{-1}. \tag{5.28}$$

Proof. We need to solve for the ρ_ℓ in order to satisfy the s-order condition (5.22),

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

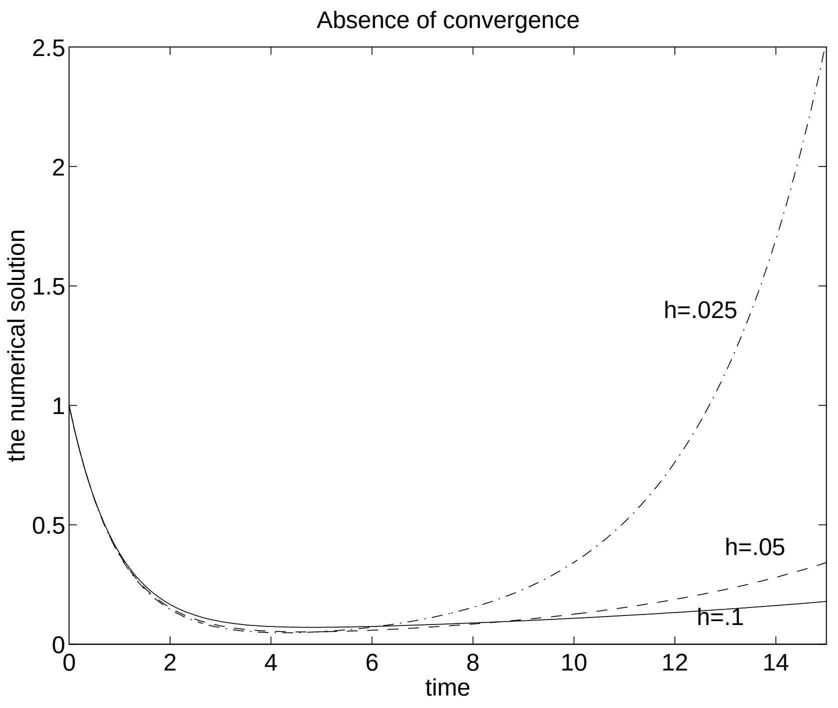


Figure 5.5: The numerical solution to the equation is $y' = -y$, $y(0) = 1$ using the 2-step method with $\rho(w) = w^2 - 2.01w + 1.01$, $\sigma(w) = 0.995w - 1.005$. Thus, ρ has a zero at 1.01, the method is not convergent, and the smaller h the worse the solution.

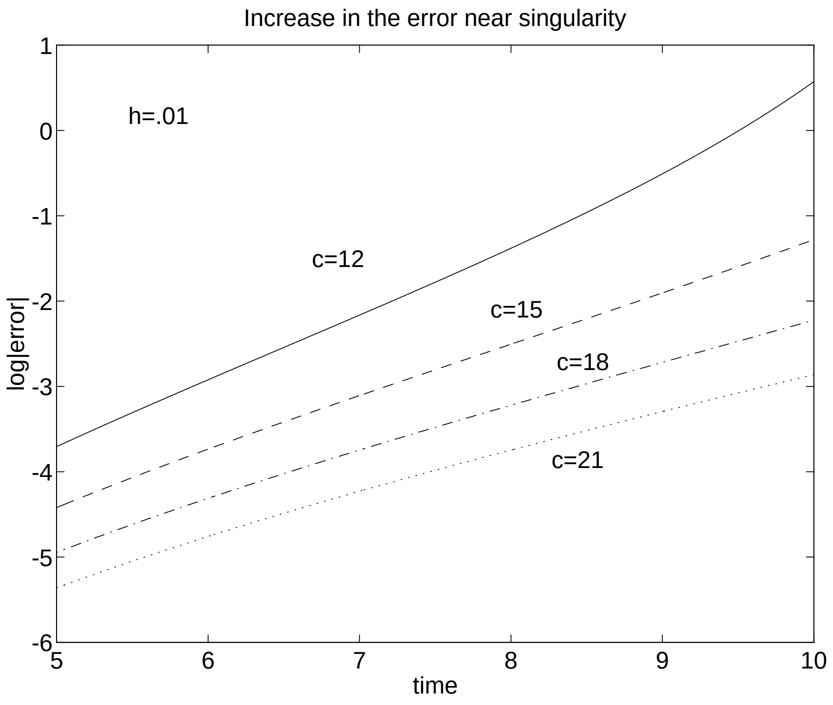


Figure 5.6: Plot showing that accuracy may deteriorate near a singularity. Plotted is the error to the solution of $y' = 2y/t + (y/t)^2$, $y(1) = 1/(c-1)$ using the 2-step Adams–Bashforth method, for various values of c. The exact solution is $y(t) = t^2/(c-t)$, with singularity at c. Accuracy worsens the nearer we are to the singularity.

To do this, rather than using Gaussian elimination or similar, we make the '*dirty trick*' observation that

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

Thence

$$\rho(w) = \sigma_s w^s \sum_{\ell=1}^{\infty}\frac{1}{\ell}\left(\frac{w-1}{w}\right)^{\ell} + \mathcal{O}\big(|w-1|^{s+1}\big) ,$$

and so

$$\sum_{\ell=0}^{s}\rho_\ell w^\ell = \sigma_s \sum_{\ell=1}^{s}\frac{1}{\ell}w^{s-\ell}(w-1)^{\ell} .$$

The result follows from collecting powers of w^s on the right, and then picking σ_s so that $\rho_s = 1$. □

Examples

(i) Let $s = 2$. Then substitution in (5.28) 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 ρ satisfies the root condition, and the 2-step BDF is

$$\boldsymbol{y}_{n+2} - \tfrac{4}{3}\boldsymbol{y}_{n+1} + \tfrac{1}{3}\boldsymbol{y}_n = \tfrac{2}{3}h\boldsymbol{f}(t_{n+2},\boldsymbol{y}_{n+2}). \tag{5.29a}$$

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

$$\boldsymbol{y}_{n+3} - \tfrac{18}{11}\boldsymbol{y}_{n+2} + \tfrac{9}{11}\boldsymbol{y}_{n+1} - \tfrac{2}{11}\boldsymbol{y}_n = \tfrac{6}{11}h\boldsymbol{f}(t_{n+3},\boldsymbol{y}_{n+3}). \tag{5.29b}$$

Convergence of BDF methods. We cannot take it for granted that BDF methods are convergent (i.e. that ρ satisfies the root condition). It is possible to prove that they are convergent iff $s \leqslant 6$. They *must not* be used outside this range!

5.4 Runge–Kutta methods

5.4.1 Quadrature formulae

We recall the quadrature formula (3.8) (with weight function $w \equiv 1$ and extra factors of h)

$$\int_0^h f(t)\mathrm{d}t \approx h\sum_{\ell=1}^{\nu} b_\ell f(c_\ell h) . \tag{5.30a}$$

If the weights b_ℓ are chosen in accordance with (3.10), i.e. if

$$hb_\ell = \int_0^h \prod_{\substack{j=1\\ j\neq\ell}}^{\nu}\frac{t-hc_j}{hc_\ell - hc_j}\,\mathrm{d}t, \qquad \ell = 1,2,\dots,\nu , \tag{5.30b}$$

this *quadrature formula* is exact for all polynomials of degree $\nu-1$. Further, provided that $\prod_{k=1}^{\nu}(t-hc_k)$ is orthogonal w.r.t. the weight function $w \equiv 1$, $0 \leqslant t \leqslant h$, the formula is exact for all polynomials of degree $2\nu - 1$.

Suppose that we wish to solve the 'ODE'

$$y' = f(t) , \quad y(0) = y_0 . \tag{5.31a}$$

The exact solution is

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t)\mathrm{d}t , \tag{5.31b}$$

and we can approximate it by quadrature. In general, we obtain the time-stepping scheme

$$y_{n+1} = y_n + h\sum_{\ell=1}^{\nu} b_\ell f(t_n + c_\ell h) \qquad n = 0, 1, \dots\,, \tag{5.31c}$$

8/10 where $h = t_{n+1} - t_n$ (and the points t_n need not be equi-spaced).

8/11
8/14
Formula (5.31c) holds for the special case when $f \equiv f(t)$. The natural question is whether we can generalize this to genuine ODEs of the form (5.1), i.e. when $\boldsymbol{f} \equiv \boldsymbol{f}(t, \boldsymbol{y})$. In this case we can formally conclude that

$$\boldsymbol{y}(t_{n+1}) = \boldsymbol{y}(t_n) + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(t, \boldsymbol{y}(t))\mathrm{d}t\,, \tag{5.32a}$$

and this can be 'approximated' by

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{\ell=1}^{\nu} b_\ell \boldsymbol{f}(t_n + c_\ell h, \boldsymbol{y}(t_n + c_\ell h)). \tag{5.32b}$$

except that, of course, the vectors $\boldsymbol{y}(t_n + c_\ell h)$ are unknown! *Runge–Kutta methods* are a means of implementing (5.32b) by replacing unknown values of $\boldsymbol{y}$ by suitable linear combinations. Specifically, the $\boldsymbol{y}(t_n + c_\ell h)$ can be approximated by another quadrature using approximate values of $\boldsymbol{y}(t_n{+}c_j h)$ obtained earlier:

$$\boldsymbol{y}(t_n{+}c_\ell h) = \boldsymbol{y}(t_n) + \int_{t_n}^{t_n+c_\ell h} \boldsymbol{f}(t, \boldsymbol{y}(t))\, dt \;\approx\; \boldsymbol{y}(t_n) + h\sum_{j=1}^{\ell-1} a_{\ell,j} \boldsymbol{f}(t_n{+}c_j h, \boldsymbol{y}(t_n{+}c_j h))\,, \tag{5.33a}$$

where, in order that the quadratures are exact on constants,

$$c_\ell = \sum_{j=1}^{\ell-1} a_{\ell,j}\,. \tag{5.33b}$$

Applying $\boldsymbol{f}$ to both sides of this formula we obtain

$$\boldsymbol{f}(t_n{+}c_\ell h, \boldsymbol{y}(t_n{+}c_\ell h) \approx \boldsymbol{f}\left(t_n{+}c_\ell h, \boldsymbol{y}(t_n) + h\sum_{j=1}^{\ell-1} a_{\ell,j} \boldsymbol{f}(t_n{+}c_j h, \boldsymbol{y}(t_n{+}c_j h))\right)\,.$$

Finally, letting $\boldsymbol{k}_\ell \approx \boldsymbol{f}(t_n{+}c_\ell h, \boldsymbol{y}(t_n{+}c_\ell h))$, we arrive at the general form of an *ν-stage explicit Runge–Kutta method* (RK):

$$\boldsymbol{k}_\ell = \boldsymbol{f}\left(t_n + c_\ell h, \boldsymbol{y}_n + h\sum_{j=1}^{\ell-1} a_{\ell,j}\boldsymbol{k}_j\right), \qquad \ell = 1\dots\nu, \tag{5.34a}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{\ell=1}^{\nu} b_\ell \boldsymbol{k}_\ell\,, \qquad \sum_{\ell=1}^{\nu} b_\ell = 1\,, \tag{5.34b}$$

where $\sum_{\ell=1}^{\nu} b_\ell = 1$ in order that the quadratures are exact on constants. Alternatively, written out with more details:

$$\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}(t_n, \boldsymbol{y}_n), & c_1 &= 0,\\
\boldsymbol{k}_2 &= \boldsymbol{f}(t_n + c_2 h, \boldsymbol{y}_n + h a_{2,1}\boldsymbol{k}_1), & c_2 &= a_{2,1},\\
\boldsymbol{k}_3 &= \boldsymbol{f}(t_n + c_3 h, \boldsymbol{y}_n + h(a_{3,1}\boldsymbol{k}_1 + a_{3,2}\boldsymbol{k}_2)), & c_3 &= a_{3,1} + a_{3,2},\\
&\vdots & &\vdots\\
\boldsymbol{k}_\nu &= \boldsymbol{f}\left(t_n + c_\nu h, \boldsymbol{y}_n + h\sum_{j=1}^{\nu-1} a_{\nu,j}\boldsymbol{k}_j\right), & c_\nu &= \sum_{j=1}^{\nu-1} a_{\nu,j},\\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\sum_{\ell=1}^{\nu} b_\ell \boldsymbol{k}_\ell\,, & \sum_{\ell=1}^{\nu} b_\ell &= 1\,.
\end{aligned}$$

8/12
8/13
Demos
The choice of the *RK coefficients* $a_{\ell,j}$ is motivated in the first instance by order considerations.

Notation. In order to make our choice of *RK coefficients* we will need to use the Taylor expansion of a vector function, so we need to be clear what we mean. We adopt the notation

$$\begin{aligned}(\boldsymbol{f}(t+h,\boldsymbol{y}+\boldsymbol{d}))_\ell &= f_\ell(t+h,\boldsymbol{y}+\boldsymbol{d})\\ &= f_\ell(t,\boldsymbol{y}) + h\frac{\partial f_\ell}{\partial t}(t,\boldsymbol{y}) + d_j\frac{\partial f_\ell}{\partial y_j}(t,\boldsymbol{y}) + \mathcal{O}\big(h^2,|\boldsymbol{d}|^2\big)\\ &= \left(\boldsymbol{f}(t,\boldsymbol{y}) + h\frac{\partial \boldsymbol{f}}{\partial t}(t,\boldsymbol{y}) + \boldsymbol{d}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t,\boldsymbol{y})\right)_\ell + \mathcal{O}\big(h^2,|\boldsymbol{d}|^2\big)\\ &= \left(\boldsymbol{f}(t,\boldsymbol{y}) + h\frac{\partial \boldsymbol{f}}{\partial t}(t,\boldsymbol{y}) + \boldsymbol{d}\cdot\boldsymbol{\nabla}\boldsymbol{f}(t,\boldsymbol{y})\right)_\ell + \mathcal{O}\big(h^2,|\boldsymbol{d}|^2\big)\ .\end{aligned}$$

5.4.2 2-stage explicit RK methods

Let us analyse the order of two-stage explicit methods where, with $\nu = 2$ above,

$$\boldsymbol{k}_1 = \boldsymbol{f}(t_n,\boldsymbol{y}_n), \tag{5.35a}$$

$$\boldsymbol{k}_2 = \boldsymbol{f}(t_n + c_2h, \boldsymbol{y}_n + c_2h\boldsymbol{k}_1), \tag{5.35b}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left(b_1\boldsymbol{k}_1 + b_2\boldsymbol{k}_2\right). \tag{5.35c}$$

Now Taylor-expand about $(t_n,\boldsymbol{y}_n)$ to obtain

$$\begin{aligned}\boldsymbol{k}_2 &= \boldsymbol{f}(t_n + c_2h, \boldsymbol{y}_n + c_2h\boldsymbol{f}(t_n,\boldsymbol{y}_n))\\ &= \boldsymbol{f}(t_n,\boldsymbol{y}_n) + c_2h\left(\frac{\partial \boldsymbol{f}}{\partial t}(t_n,\boldsymbol{y}_n) + \boldsymbol{f}(t_n,\boldsymbol{y}_n)\cdot\boldsymbol{\nabla}\boldsymbol{f}(t_n,\boldsymbol{y}_n)\right) + \mathcal{O}\big(h^2\big)\,.\end{aligned}$$

From (5.1) we have that $\boldsymbol{y}' = \boldsymbol{f}(t,\boldsymbol{y})$, hence

$$\boldsymbol{y}'' = \frac{\partial \boldsymbol{f}}{\partial t}(t,\boldsymbol{y}) + \boldsymbol{y}'\cdot\boldsymbol{\nabla}\boldsymbol{f}(t,\boldsymbol{y}) = \frac{\partial \boldsymbol{f}}{\partial t}(t,\boldsymbol{y}) + \boldsymbol{f}(t,\boldsymbol{y})\cdot\boldsymbol{\nabla}\boldsymbol{f}(t,\boldsymbol{y})\,.$$

Therefore, in terms of the exact solution $\boldsymbol{y}_n = \boldsymbol{y}(t_n)$, we obtain

$$\begin{aligned}\boldsymbol{k}_1 &= \boldsymbol{y}'(t_n)\,,\\ \boldsymbol{k}_2 &= \boldsymbol{y}'(t_n) + c_2h\boldsymbol{y}''(t_n) + \mathcal{O}\big(h^2\big)\ .\end{aligned}$$

Consequently, the *local* error is

$$\begin{aligned}\boldsymbol{y}(t_{n+1}) - \boldsymbol{\varphi}_h(t_n,\boldsymbol{y}(t_n)) =& \boldsymbol{y}(t_{n+1}) - \left(\boldsymbol{y}(t_n) + h\left(b_1\boldsymbol{k}_1 + b_2\boldsymbol{k}_2\right)\right)\\ =& \big(\boldsymbol{y}(t_n) + h\boldsymbol{y}'(t_n) + \tfrac{1}{2}h^2\boldsymbol{y}''(t_n) + \mathcal{O}\big(h^3\big)\big)\\ &- \big(\boldsymbol{y}(t_n) + (b_1+b_2)h\boldsymbol{y}'(t_n) + b_2c_2h^2\boldsymbol{y}''(t_n) + \mathcal{O}\big(h^3\big)\big)\\ =&(1-b_1-b_2)h\boldsymbol{y}'(t_n) + \big(\tfrac{1}{2} - b_2c_2\big)\,h^2\boldsymbol{y}''(t_n) + \mathcal{O}\big(h^3\big)\ .\end{aligned} \tag{5.37}$$

We deduce that the explicit RK method is of order 2 if $b_1 + b_2 = 1$ and $b_2c_2 = \frac{1}{2}$. A popular choice is $b_1 = 0$, $b_2 = 1$ and $c_2 = \frac{1}{2}$ (e.g. see Figure 5.7).

Remark. That no explicit 2-stage RK method can be of third order or greater can be demonstrated by applying it to $y' = \lambda y$.

5.4.3 General RK methods

A general ν-stage *Runge–Kutta method* takes the form

$$\boldsymbol{k}_\ell = \boldsymbol{f}\left(t_n + c_\ell h, \boldsymbol{y}_n + h\sum_{j=1}^{\nu} a_{\ell,j}\boldsymbol{k}_j\right) \quad \text{where} \quad \sum_{j=1}^{\nu} a_{\ell,j} = c_\ell, \qquad \ell = 1,2,\ldots,\nu, \tag{5.38a}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{\ell=1}^{\nu} b_\ell\boldsymbol{k}_\ell. \tag{5.38b}$$

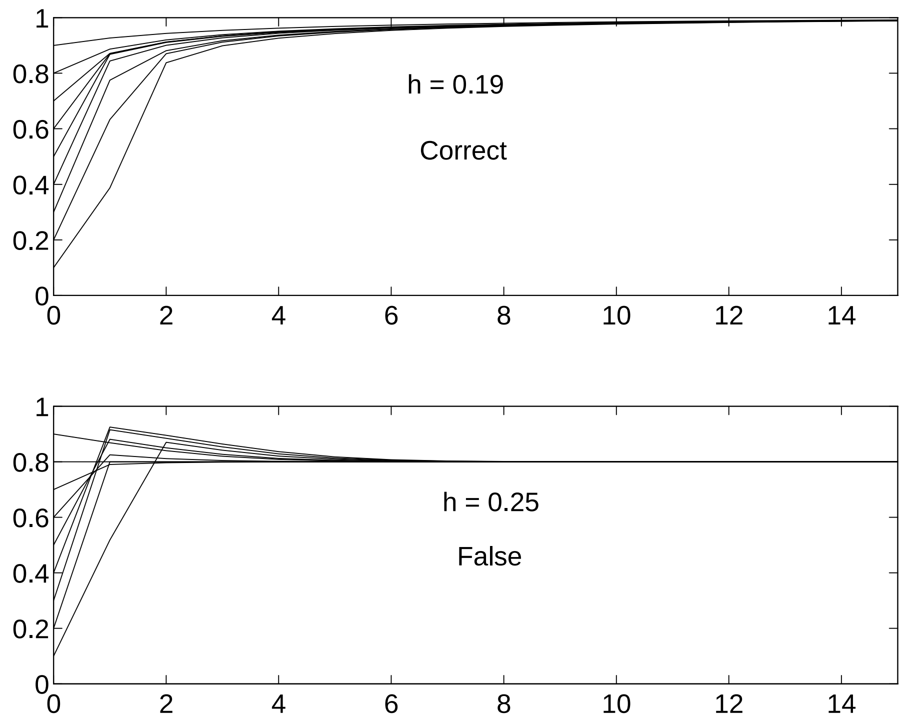


Figure 5.7: Solutions to the equation $y' = \alpha y(1-y)$ using the 2nd order RK method

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

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!

We denote it by the so-called Butcher's table

$$\begin{array}{c|c} \boldsymbol{c} & \mathsf{A} \\ \hline & \boldsymbol{b}^{\mathrm{T}} \end{array} \quad = \quad \begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,\nu} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,\nu} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_\nu & a_{\nu,1} & a_{\nu,2} & \cdots & a_{\nu,\nu} \\ \hline & b_1 & b_2 & \cdots & b_\nu \end{array} \tag{5.38c}$$

The *explicit* RK method corresponds to the case when $a_{\ell,j} = 0$ for $\ell \leqslant j$, i.e. when the matrix A is strictly lower triangular. Otherwise, an RK method is *implicit*, in particular *diagonal implicit* if A is lower triangular.

Definition. A method is said to be *consistent* if it has an order greater than 0.

Convergence. It can be shown that consistency is a necessary and sufficient condition for convergence of Runge–Kutta methods.

Example: a 2-stage implicit method. Consider the 2-stage method

$$\boldsymbol{k}_1 = \boldsymbol{f}\left(t_n, \boldsymbol{y}_n + \tfrac{1}{4}h(\boldsymbol{k}_1 - \boldsymbol{k}_2)\right), \tag{5.39a}$$
$$\boldsymbol{k}_2 = \boldsymbol{f}\left(t_n + \tfrac{2}{3}h, \boldsymbol{y}_n + \tfrac{1}{12}h(3\boldsymbol{k}_1 + 5\boldsymbol{k}_2)\right), \tag{5.39b}$$
$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \tfrac{1}{4}h(\boldsymbol{k}_1 + 3\boldsymbol{k}_2). \tag{5.39c}$$

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 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,

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

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

$$k_1 = f + \mathcal{O}(h^2) \quad \text{and} \quad k_2 = f + \tfrac{2}{3}hff_y + \mathcal{O}(h^2)\ .$$

Substituting again, we obtain

$$k_1 = f - \tfrac{1}{6}h^2 ff_y^2 + \mathcal{O}(h^3)\,,$$
$$k_2 = f + \tfrac{2}{3}hff_y + h^2\left(\tfrac{5}{18}ff_y^2 + \tfrac{2}{9}f^2 f_{yy}\right) + \mathcal{O}(h^3)$$

and hence

$$y_{n+1} = y + hf + \tfrac{1}{2}h^2 ff_y + \tfrac{1}{6}h^3(ff_y^2 + f^2 f_{yy}) + \mathcal{O}(h^4)\,.$$

But $y' = f$, and hence

$$y'' = ff_y \quad \text{and} \quad y''' = ff_y^2 + f^2 f_{yy}\,.$$

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 $y' = \lambda y$.

A better way. A better way of deriving the order of Runge-Kutta methods is based on graph-theoretic approaches.

5.4.4 Collocation (*Unlectured*)

Method 5.11 (Collocation)**.** Let p be a polynomial of degree s such that

$$p(t_n) = y_n, \qquad p'(t_n + c_i h) = f(t_n + c_i h, p(t_n + c_i h)), \quad i = 1, \ldots, s. \tag{5.40}$$

Then we let $y_{n+1} = p(t_{n+1})$ be the approximation.

Lemma 5.12. *Let ℓ_i be the fundamental Lagrange polynomials of degree $s-1$ for the knots c_i. Then the collocation method is identical to the RK method with parameters*

$$a_{ij} = \int_0^{c_i} \ell_j(t)\,dt, \qquad b_i = \int_0^1 \ell_j(t)\,dt\,. \tag{5.41}$$

Proof. Let $\tau_i = t_n + c_i h$ and let p satisfy (5.40). Then we have $p'(t) = \sum_{j=1}^s p'(\tau_j)\ell_j(\frac{t-t_n}{h})$ and integration yields

$$p(t) = y_n + h\sum p'(\tau_j)\int_0^{\frac{t-t_n}{h}} \ell_j(\tau)\,d\tau\,.$$

Set $k_i = p'(\tau_i) \overset{(5.40)}{=} f(\tau_i, p(\tau_i))$. Then $p(\tau_i) = y_n + h\sum_{j=1}^s a_{ij}k_j$, therefore

$$k_i = f(\tau_i, p(\tau_i)) = f(t_n + c_i h, y_n + h\sum_{j=1}^s a_{ij}k_j) \tag{5.42a}$$

$$y_{n+1} = p(t_{n+1}) = p(t_n) + h\sum b_i p'(\tau_i) = y_n + h\sum b_i k_i\,. \tag{5.42b}$$

□

Theorem 5.13 (No proof)**.** *Let $\omega(t) = \prod_{i=1}^s (t - c_i)$ be orthogonal on the interval $[0,1]$ to all polynomials of degree $r - 1 \leqslant s - 1$. Then the (highly implicit) RK method with parameters (5.41) is of order $s + r$.*

The highest order of an s-stage implicit RK method is $2s$, and corresponds to collocation at zeros of the Legendre polynomial (Gauss-Legendre RK method). Construction of explicit methods with high order is an art form.

6 Stiff Equations

6.1 Stiffness: the problem

Consider the linear system

$$\boldsymbol{y}' = \mathsf{A}\boldsymbol{y} \quad \text{with} \quad \mathsf{A} = \begin{pmatrix} -100 & 1 \\ 0 & -\frac{1}{10} \end{pmatrix}, \tag{6.1}$$

where we note that A is *diagonalisable*. The exact solution is a linear combination of e^{-100t} and $\mathrm{e}^{-t/10}$: the first decays very rapidly to zero, whereas the second decays gently. Suppose that we solve the ODE with the *forward Euler* method. As we shall see below the convergence requirement that $\lim_{n\to\infty} \boldsymbol{y}_n = \mathbf{0}$ (for fixed $h > 0$) then leads to a stringent restriction on the size of h.

With greater generality, consider the matrix differential equation

$$\boldsymbol{y}' = \mathsf{B}\boldsymbol{y}, \tag{6.2}$$

for a $M \times M$ constant *diagonalisable* matrix B.

Exact solution. Define

$$\mathrm{e}^{t\mathsf{B}} = \sum_{k=0}^{\infty} \frac{1}{k!} t^k \mathsf{B}^k . \tag{6.3a}$$

Then

$$\frac{d}{dt}\left(\mathrm{e}^{t\mathsf{B}}\right) = \sum_{k=1}^{\infty} \frac{1}{(k-1)!} t^{k-1} \mathsf{B}^k = \mathsf{B}\mathrm{e}^{t\mathsf{B}}, \tag{6.3b}$$

and hence the solution to (6.2) is

$$\boldsymbol{y} = \mathrm{e}^{t\mathsf{B}} \boldsymbol{y}_0 . \tag{6.3c}$$

Let the eigenvalues of B be $\lambda_1, \dots, \lambda_M$, and denote the corresponding linearly-independent eigenvectors by $\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_M$. Define $\mathsf{D} = \operatorname{diag}\boldsymbol{\lambda}$ and $\mathsf{V} = (\boldsymbol{v}_1 \ \boldsymbol{v}_2 \ \dots \ \boldsymbol{v}_M)$. Then $\mathsf{B} = \mathsf{V}\mathsf{D}\mathsf{V}^{-1}$ and

$$\mathrm{e}^{t\mathsf{B}} = \sum_{k=0}^{\infty} \frac{1}{k!} t^k (\mathsf{V}\mathsf{D}\mathsf{V}^{-1})^k = \mathsf{V}\left(\sum_{k=0}^{\infty} \frac{1}{k!} t^k \mathsf{D}^k\right)\mathsf{V}^{-1} = \mathsf{V}\mathrm{e}^{t\mathsf{D}}\mathsf{V}^{-1}, \tag{6.4a}$$

where

$$\mathrm{e}^{t\mathsf{D}} = \begin{pmatrix} \mathrm{e}^{t\lambda_1} & 0 & \dots & 0 \\ 0 & \mathrm{e}^{t\lambda_2} & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \mathrm{e}^{t\lambda_M} \end{pmatrix}. \tag{6.4b}$$

Further, if we assume that $\operatorname{Re}\lambda_\ell < 0$, $\ell = 1, \dots, M$, then we conclude that

$$\lim_{t\to\infty} \boldsymbol{y}(t) = \lim_{t\to\infty} \mathsf{V}\mathrm{e}^{t\mathsf{D}}\mathsf{V}^{-1}\boldsymbol{y}_0 = \mathbf{0}. \tag{6.5}$$

Numerical solution. Suppose that we use Euler's method to solve this equation, then

$$\boldsymbol{y}_{n+1} = (\mathsf{I} + h\mathsf{B})\boldsymbol{y}_n . \tag{6.6a}$$

Hence, by induction,

$$\begin{aligned} \boldsymbol{y}_n &= (\mathsf{I} + h\mathsf{B})^n \boldsymbol{y}_0 \\ &= \left(\mathsf{V}\mathsf{V}^{-1} + h\mathsf{V}\mathsf{D}\mathsf{V}^{-1}\right)^n \boldsymbol{y}_0 \\ &= \left(\mathsf{V}(\mathsf{I} + h\mathsf{D})\mathsf{V}^{-1}\right)^n \boldsymbol{y}_0 \\ &= \mathsf{V}(\mathsf{I} + h\mathsf{D})^n\mathsf{V}^{-1}\boldsymbol{y}_0, \end{aligned} \tag{6.6b}$$

where $(\mathsf{I}+h\mathsf{D})^n$ is a diagonal matrix with entries $(1+h\lambda_\ell)^n$, $\ell = 1, \dots, M$. Therefore $\lim_{n\to\infty} \boldsymbol{y}_n = \mathbf{0}$ for all initial values $\boldsymbol{y}_0$ iff

$$|1 + h\lambda_\ell| < 1, \quad \ell = 1, \dots, M. \tag{6.7}$$

For example (6.1) we thus require

$$|1 - \tfrac{1}{10}h| < 1 \quad \text{and} \quad |1 - 100h| < 1, \quad \text{i.e.} \quad h < \tfrac{1}{50}.$$

This restriction, necessary to recover the correct asymptotic behaviour, is not do with local accuracy. At large times, or equivalently for large n, the rapidly decaying e^{-100t} component is exceedingly small. The restriction is necessary to ensure that this component is modelled with sufficient accuracy that it does not grow to infect the numerical solution.

9/10
9/11
9/14

6.2 Linear stability

Definition: stiffness. We say that the ODE $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$ is *stiff* if (for some methods) we need to depress h to maintain *stability* well beyond requirements of accuracy. An important example of stiff systems occurs for linear equations when $\operatorname{Re}\lambda_\ell < 0$, $\ell = 1, 2, \ldots, M$, and the quotient $\max|\lambda_k|/\min|\lambda_k|$ is large: a ratio of 10^{20} is not unusual in real-life problems.

Remark. Stiff equations, mostly nonlinear, occur throughout applications, whenever we have two (or more) different timescales in the ODE. A typical example is the equations of *chemical kinetics,* where each timescale is determined by the speed of reaction between two compounds: such speeds can differ by many orders of magnitude.

Definition: linear stability domain. Suppose that a numerical method, applied to $y' = \lambda y$, $y(0) = 1$, with constant h, produces the solution sequence $\{y_n\}_{n\in\mathbb{Z}^+}$. We call the set

$$\mathcal{D} = \{h\lambda \in \mathbb{C} \,:\, \lim_{n\to\infty} y_n = 0\}$$

9/12 the *linear stability domain* of the method.

9/13 *Definition: A-stability.* The set of $\lambda \in \mathbb{C}$ for which the exact solution to $y' = \lambda y$ decays as $t \to \infty$ is the left half-plane $\mathbb{C}^- = \{z \in \mathbb{C} \,:\, \operatorname{Re} z < 0\}$. A good criterion for a 'stiff' method is that it recovers the correct asymptotics for all stable linear equations, i.e. that the numerical solution always decays to zero when the exact solution does. Such a method, i.e. one such that $\mathbb{C}^- \subseteq \mathcal{D}$, is said to be *A-stable.*

Example: Euler's method. We have already deduced, see (6.7), that for Euler's method $y_n \to 0$ iff $|1 + h\lambda| < 1$. Hence, as illustrated in figure (6.8),

$$\mathcal{D} = \{z \in \mathbb{C} \,:\, |1 + z| < 1\}.$$

We conclude that Euler's method is not A-stable.

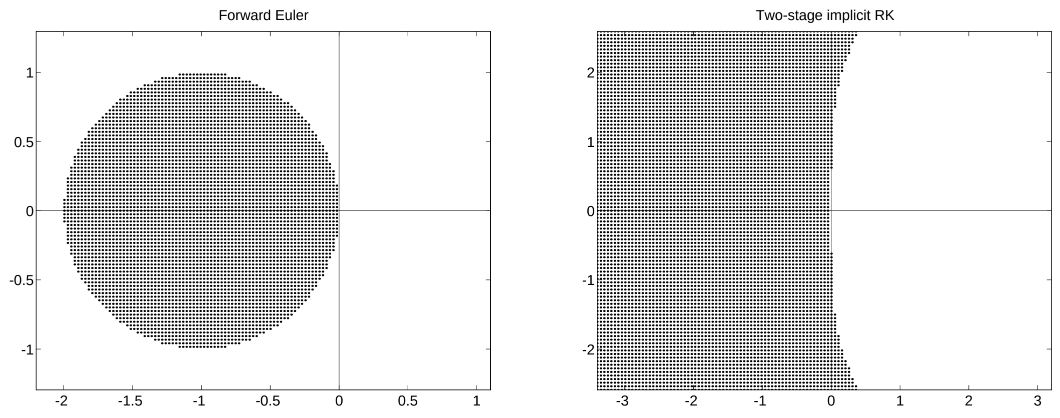


Figure 6.8: Linear stability domains: Euler's method and a 2-stage implicit RK. See also the *A-Stabilty* demonstration at http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php.

Example: Backward Euler method. Solving $y' = \lambda y$ with the backward Euler method we obtain

$$y_{n+1} = \frac{1}{1 - h\lambda}\, y_n\,,$$

and thus, by induction,

$$y_n = \left(\frac{1}{1-h\lambda}\right)^n y_0 \, .$$

Hence

$$\mathcal{D} = \{z \in \mathbb{C} \, : \, |1-z| > 1\} \, ,$$

i.e. the domain outside the unit circle centred on 1. We conclude that the backward Euler method is A-stable.

Example: trapezoidal rule. Solving $y' = \lambda y$ with the *trapezoidal rule,* we obtain

$$y_{n+1} = \frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda} \, y_n \, ,$$

and thus, by induction,

$$y_n = \left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)^n y_0 \, .$$

Therefore

$$\begin{aligned} z \in \mathcal{D} &\Leftrightarrow \left|\frac{1+\frac{1}{2}z}{1-\frac{1}{2}z}\right| < 1 \\ &\Leftrightarrow \left(1+\tfrac{1}{2}z\right)\left(1+\tfrac{1}{2}\bar{z}\right) < \left(1-\tfrac{1}{2}z\right)\left(1-\tfrac{1}{2}\bar{z}\right) \\ &\Leftrightarrow z + \bar{z} < 0 \\ &\Leftrightarrow \operatorname{Re} z < 0 \end{aligned}$$

We deduce that $\mathcal{D} = \mathbb{C}^-$. Hence, the trapezoidal rule is A-stable, and it is not necessary to reduce the step-size prevent instabilities.

Remark. Note that A-stability does not mean that *any* step size will do! We need to choose h small enough to ensure the right accuracy, but we do not want to have to depress it much further to prevent instability.

Multistep methods. The determination of $\mathcal{D}$ for multistep methods is considerably more difficult. Further, their A-stability is, more often than not, disappointing. This is because, according to the *second Dahlquist barrier*, no multistep method of order $p \geqslant 3$ may be A-stable. Note that the $p = 2$ barrier for A-stability is attained by the trapezoidal rule (which can be viewed as the 1-step Adams-Moulton method).

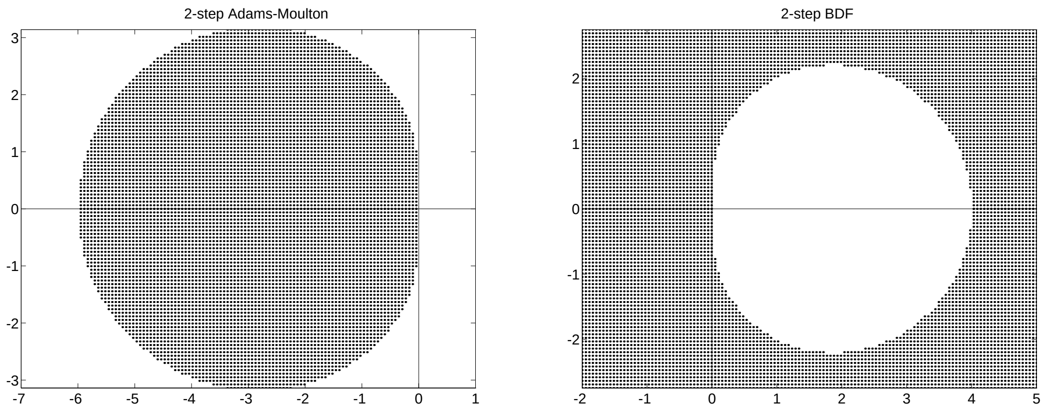


Figure 6.9: Linear stability domains: 2-step Adams-Moulton and 2-step BDF. See also the *A-Stabilty* demonstration at http://www.maths.cam.ac.uk/undergrad/course/na/ib/partib.php.

6.3 Overcoming the Dahlquist barrier

The Dahlquist barrier implies that, in our quest for higher-order methods with good stability properties, we need to pursue one of the following strategies:

- either relax the definition of A-stability,
- or consider other methods in place of multistep.

The two courses of action will be considered next.

Stiffness and BDF methods. Inasmuch as no multistep method of order $p \geqslant 3$ may be A-stable, the stability properties of convergent BDF methods are 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*[8] systems, the eigenvalues are not just in $\mathbb{C}^-$ but also well away from $\mathrm{i}\mathbb{R}$. Hence schemes that are stable well away from $\mathrm{i}\mathbb{R}$ can be competitive. This is the case for all BDF methods of order $p \leqslant 6$ (i.e. all convergent BDF methods), since they share the feature that the linear stability domain $\mathcal{D}$ includes a wedge about $(-\infty, 0)$: such methods are said to be A_0*-stable.*

Stiffness and Runge–Kutta. Unlike multistep methods, implicit high-order RK may be A-stable. For example, recall the 3rd-order method (5.39a)-(5.39c):

$$\boldsymbol{k}_1 = \boldsymbol{f}\left(t_n, \boldsymbol{y}_n + \tfrac{1}{4}h(\boldsymbol{k}_1 - \boldsymbol{k}_2)\right), \tag{6.8a}$$

$$\boldsymbol{k}_2 = \boldsymbol{f}\left(t_n + \tfrac{2}{3}h, \boldsymbol{y}_n + \tfrac{1}{12}h(3\boldsymbol{k}_1 + 5\boldsymbol{k}_2)\right), \tag{6.8b}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \tfrac{1}{4}h(\boldsymbol{k}_1 + 3\boldsymbol{k}_2). \tag{6.8c}$$

Applying this scheme to $y' = \lambda y$, we have that

$$hk_1 = h\lambda\left(y_n + \tfrac{1}{4}hk_1 - \tfrac{1}{4}hk_2\right),$$
$$hk_2 = h\lambda\left(y_n + \tfrac{1}{4}hk_1 + \tfrac{5}{12}hk_2\right).$$

This is a linear system for hk_1 and hk_2, whose solution is

$$\begin{bmatrix} hk_1 \\ hk_2 \end{bmatrix} = \begin{bmatrix} 1-\frac{1}{4}h\lambda & \frac{1}{4}h\lambda \\ -\frac{1}{4}h\lambda & 1-\frac{5}{12}h\lambda \end{bmatrix}^{-1} \begin{bmatrix} h\lambda y_n \\ h\lambda y_n \end{bmatrix} = \frac{h\lambda y_n}{1-\frac{2}{3}h\lambda+\frac{1}{6}(h\lambda)^2} \begin{bmatrix} 1-\frac{2}{3}h\lambda \\ 1 \end{bmatrix}.$$

Therefore

$$y_{n+1} = y_n + \tfrac{1}{4}hk_1 + \tfrac{3}{4}hk_2 = \frac{1+\frac{1}{3}h\lambda}{1-\frac{2}{3}h\lambda+\frac{1}{6}h^2\lambda^2}y_n.$$

Let

$$r(z) = \frac{1+\frac{1}{3}z}{1-\frac{2}{3}z+\frac{1}{6}z^2},$$

then $y_{n+1} = r(h\lambda)y_n$, and hence, by induction,

$$y_n = [r(h\lambda)]^n y_0\,.$$

We deduce that

$$\mathcal{D} = \{z \in \mathbb{C} : |r(z)| < 1\}.$$

We wish to prove that $|r(z)| < 1$ for every $z \in \mathbb{C}^-$, since this is equivalent to A-stability. This can be done by a technique that can be applied to other RK methods. According to the *maximum modulus principle* from *Complex Analysis*[9], if g is analytic in the closed complex domain $\mathcal{V}$ then $|g|$ attains its maximum on $\partial\mathcal{V}$. We let $g = r$. This is a rational function, hence its only singularities are the

[8] The analysis of *nonlinear* stiff equations is difficult and well outside the scope of this course.
[9] If you are taking *Complex Methods* then this is your chance to learn the statement of the *maximum modulus principle*.

poles $2 \pm \mathrm{i}\sqrt{2}$, which we note lie in the right-half plane, i.e. outside of $\mathbb{C}^-$. Hence g is analytic in $\mathcal{V} = \operatorname{cl}\mathbb{C}^- = \{z \in \mathbb{C} : \operatorname{Re} z \leqslant 0\}$. Therefore it attains its maximum in $\mathcal{V}$ on $\partial\mathcal{V} = \mathrm{i}\mathbb{R}$ and

$$\text{A-stability} \quad \Leftrightarrow \quad |r(z)| < 1, \quad z \in \mathbb{C}^- \quad \Leftrightarrow \quad |r(\mathrm{i}t)| \leqslant 1, \quad t \in \mathbb{R}.$$

In turn,

$$|r(\mathrm{i}t)|^2 \leqslant 1 \quad \Leftrightarrow \quad |1 - \tfrac{2}{3}\mathrm{i}t - \tfrac{1}{6}t^2|^2 - |1 + \tfrac{1}{3}\mathrm{i}t|^2 \geqslant 0.$$

But $|1 - \frac{2}{3}\mathrm{i}t - \frac{1}{6}t^2|^2 - |1 + \frac{1}{3}\mathrm{i}t|^2 = \frac{1}{36}t^4 \geqslant 0$, and it follows that the method is A-stable.

Example: the 2-stage Gauss–Legendre method. It is possible to prove that the 2-stage *Gauss–Legendre method*

$$\boldsymbol{k}_1 = \boldsymbol{f}(t_n + (\tfrac{1}{2} - \tfrac{\sqrt{3}}{6})h, \boldsymbol{y}_n + \tfrac{1}{4}h\boldsymbol{k}_1 + (\tfrac{1}{4} - \tfrac{\sqrt{3}}{6})h\boldsymbol{k}_2), \tag{6.9a}$$

$$\boldsymbol{k}_2 = \boldsymbol{f}(t_n + (\tfrac{1}{2} + \tfrac{\sqrt{3}}{6})h, \boldsymbol{y}_n + (\tfrac{1}{4} + \tfrac{\sqrt{3}}{6})h\boldsymbol{k}_1 + \tfrac{1}{4}h\boldsymbol{k}_2), \tag{6.9b}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \tfrac{1}{2}h(\boldsymbol{k}_1 + \boldsymbol{k}_2), \tag{6.9c}$$

is of order 4, e.g. by expansion for $y' = f(y)$ (note: expansions become messy for $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$). Applying it to $y' = \lambda y$ we obtain, with $z = h\lambda$,

$$hk_1 = z\Big(y_n + \tfrac{1}{4}hk_1 + \big(\tfrac{1}{4} - \tfrac{\sqrt{3}}{6}\big)hk_2\Big), \tag{6.10a}$$

$$hk_2 = z\Big(y_n + \big(\tfrac{1}{4} + \tfrac{\sqrt{3}}{6}\big)hk_1 + \tfrac{1}{4}hk_2\Big). \tag{6.10b}$$

This is a linear system with unknowns hk_1 and hk_2 whose solution is

$$\begin{aligned} \begin{bmatrix} hk_1 \\ hk_2 \end{bmatrix} &= \begin{bmatrix} 1 - \frac{1}{4}z & -(\frac{1}{4} - \frac{\sqrt{3}}{6})z \\ -(\frac{1}{4} + \frac{\sqrt{3}}{6})z & 1 - \frac{1}{4}z \end{bmatrix}^{-1} \begin{bmatrix} zy_n \\ zy_n \end{bmatrix} \\ &= \frac{1}{\det(\mathsf{A})} \begin{bmatrix} 1 - \frac{1}{4}z & (\frac{1}{4} + \frac{\sqrt{3}}{6})z \\ (\frac{1}{4} - \frac{\sqrt{3}}{6})z & 1 - \frac{1}{4}z \end{bmatrix} \begin{bmatrix} zy_n \\ zy_n \end{bmatrix}. \end{aligned}$$

We need

$$hk_1 + hk_2 = \tfrac{2z}{\det(\mathsf{A})}\, y_n = \tfrac{2z}{1 - \frac{1}{2}z + \frac{1}{12}z^2}\, y_n\,,$$

Hence

$$y_{n+1} = y_n + \tfrac{1}{2}(hk_1 + hk_2) = \frac{1 + \frac{1}{2}z + \frac{1}{12}z^2}{1 - \frac{1}{2}z + \frac{1}{12}z^2}\, y_n\,, \tag{6.11a}$$

and so by induction

$$y_n = [r(z)]^n y_0\,, \quad \text{where} \quad r(z) = \frac{1 + \frac{1}{2}z + \frac{1}{12}z^2}{1 - \frac{1}{2}z + \frac{1}{12}z^2}. \tag{6.11b}$$

$r(z)$ is a rational function, and its only singularities are the poles $3 \pm i\sqrt{3}$ which lie in the right half-lane; hence r is analytic in $\mathcal{V} = \operatorname{cl}\mathbb{C}^- = \{z \in \mathbb{C} : \operatorname{Re} z \leqslant 0\}$. Therefore it attains its maximum in $\mathcal{V}$ on $\partial\mathcal{V} = \mathrm{i}\mathbb{R}$ and

$$\text{A-stability} \quad \Leftrightarrow \quad |r(z)| < 1, \quad z \in \mathbb{C}^- \quad \Leftrightarrow \quad |r(it)| \leqslant 1, \quad t \in \mathbb{R}.$$

But $|r(it)| = 1$, and thus the Gauss–Legendre method is A-stable. Moreover, since $r(z) = \frac{1}{r(-z)}$ we deduce that $\mathcal{D} = \mathbb{C}^-$.

Theorem 6.1 (No proof)**.** *No explicit Runge-Kutta method may be A-stable (or even A_0-stable).*

7 Implementation of ODE Methods

The step size h is not some preordained quantity: it is a parameter of the method (in reality, many parameters, since we may vary it from step to step). However, the basic input of a well-written computer package for ODEs is not the step size but the *error tolerance:* the level of precision, as required by the user. The choice of $h > 0$ is an important tool at our disposal to keep a local estimate of the error beneath the required tolerance in the solution interval. In other words, we need not only a *time-stepping algorithm,* but also mechanisms for *error control* and rules for choosing the step size. 10/10

7.1 Error constants

Suppose that we wish to monitor the error of the trapezoidal rule

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \tfrac{1}{2}h[\boldsymbol{f}(\boldsymbol{y}_n) + \boldsymbol{f}(\boldsymbol{y}_{n+1})]\,, \tag{7.1a}$$

for which we already know that the order is 2. If we substitute the true solution, $y_n = y(t_n)$ into (7.1a) we deduce that (see also (5.13c) with $\theta = \frac{1}{2}$)

$$\boldsymbol{y}(t_{n+1}) - \{\boldsymbol{y}(t_n) + \tfrac{1}{2}h[\boldsymbol{y}'(t_n) + \boldsymbol{y}'(t_{n+1})]\} = c_{\mathrm{TR}}h^3\boldsymbol{y}'''(t_n) + \mathcal{O}\big(h^4\big)\,, \tag{7.1b}$$

where

$$c_{\mathrm{TR}} = -\tfrac{1}{12}\,. \tag{7.1c}$$

To estimate the error in a single step we *assume* that $\boldsymbol{y}_n = \boldsymbol{y}(t_n)$ and use (7.1a) and (7.1b) to deduce that

$$\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}^{\mathrm{TR}}_{n+1} = c_{\mathrm{TR}}h^3\boldsymbol{y}'''(t_n) + \mathcal{O}\big(h^4\big)\ . \tag{7.1d}$$

Therefore, the error in each step is increased roughly by $c_{\mathrm{TR}}h^3\boldsymbol{y}'''(t_n)$ (although this error estimate does not help much because the value $\boldsymbol{y}'''(t_n)$ is unknown).

Definition. The number c_{TR} is called the *error constant* of the trapezoidal rule.

Each multistep method (but not RK!) has its own error constant. For example, the 2nd order 2-step Adams–Bashforth method, (5.18),

$$\boldsymbol{y}_{n+1} - \boldsymbol{y}_n = \tfrac{1}{2}h\left[3\boldsymbol{f}(t_n,\boldsymbol{y}_n) - \boldsymbol{f}(t_{n-1},\boldsymbol{y}_{n-1})\right], \tag{7.2a}$$

has the error constant $c_{\mathrm{AB}} = \frac{5}{12}$ (see (5.23b)), i.e.

$$\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}^{\mathrm{AB}}_{n+1} = c_{\mathrm{AB}}h^3\boldsymbol{y}'''(t_n) + \mathcal{O}(h^4)\,. \tag{7.2b}$$

10/11
10/12
10/13
10/14
Demos

7.2 The Milne device

The idea behind the *Milne device* is to use two multistep methods of the same order, one explicit and the second implicit (e.g., (7.2a) and (7.1a), respectively), to estimate the local error of the implicit method. For example, *locally,*

$$\begin{aligned}\boldsymbol{y}^{\mathrm{AB}}_{n+1} &\approx \boldsymbol{y}(t_{n+1}) - c_{\mathrm{AB}}h^3\boldsymbol{y}'''(t_n) = \boldsymbol{y}(t_{n+1}) - \tfrac{5}{12}h^3\boldsymbol{y}'''(t_n),\\ \boldsymbol{y}^{\mathrm{TR}}_{n+1} &\approx \boldsymbol{y}(t_{n+1}) - c_{\mathrm{TR}}h^3\boldsymbol{y}'''(t_n) = \boldsymbol{y}(t_{n+1}) + \tfrac{1}{12}h^3\boldsymbol{y}'''(t_n).\end{aligned}$$

Subtracting, we obtain the estimate

$$h^3\boldsymbol{y}'''(t_n) \approx -\frac{1}{c_{\mathrm{AB}} - c_{\mathrm{TR}}}\left(\boldsymbol{y}^{\mathrm{AB}}_{n+1} - \boldsymbol{y}^{\mathrm{TR}}_{n+1}\right) = -2(\boldsymbol{y}^{\mathrm{AB}}_{n+1} - \boldsymbol{y}^{\mathrm{TR}}_{n+1}), \tag{7.3a}$$

therefore

$$\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}^{\mathrm{TR}}_{n+1} \approx -\frac{c_{\mathrm{TR}}}{c_{\mathrm{AB}} - c_{\mathrm{TR}}}(\boldsymbol{y}^{\mathrm{AB}}_{n+1} - \boldsymbol{y}^{\mathrm{TR}}_{n+1}) = \tfrac{1}{6}(\boldsymbol{y}^{\mathrm{AB}}_{n+1} - \boldsymbol{y}^{\mathrm{TR}}_{n+1})\,. \tag{7.3b}$$

We use the right hand side as an estimate of the local error.

Remark. TR is a far better method than AB: it is A-stable, hence its *global* behaviour is superior. We employ AB *solely* to estimate the local error. This adds very little to the overall cost of TR, since AB is an explicit method.

7.3 Implementation of the Milne device (and Predictor-Corrector methods)

To implement the *Milne device,* we work with a *pair* of multistep methods of the same order, one explicit *(predictor)* and the other implicit *(corrector),* e.g. the third-order Adams–Bashforth and Adams–Moulton methods respectively:

$$\begin{aligned}
\text{Predictor :}\quad & \boldsymbol{y}^{\mathrm{P}}_{n+2} = \boldsymbol{y}_{n+1} + h[\tfrac{5}{12}\boldsymbol{f}(t_{n-1},\boldsymbol{y}_{n-1}) - \tfrac{4}{3}\boldsymbol{f}(t_n,\boldsymbol{y}_n) + \tfrac{23}{12}\boldsymbol{f}(t_{n+1},\boldsymbol{y}_{n+1})], & (7.4a)\\
\text{Corrector :}\quad & \boldsymbol{y}^{\mathrm{C}}_{n+2} = \boldsymbol{y}_{n+1} + h[-\tfrac{1}{12}\boldsymbol{f}(t_n,\boldsymbol{y}_n) + \tfrac{2}{3}\boldsymbol{f}(t_{n+1},\boldsymbol{y}_{n+1}) + \tfrac{5}{12}\boldsymbol{f}(t_{n+2},\boldsymbol{y}_{n+2})]\,. & (7.4b)
\end{aligned}$$

The predictor is employed not just to estimate the error of the corrector, but also to provide *a good initial guess in the solution of the implicit corrector equations.* Typically, for nonstiff equations, we iterate correction equations at most twice, while stiff equations require *iteration to convergence*, otherwise the typically superior stability 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) $\alpha\,\varepsilon_{\mathrm{TOL}} \leqslant \|\,\text{error}\,\| \leqslant \varepsilon_{\mathrm{TOL}}$, say with $\alpha = \frac{1}{10}$: accept the step, continue to t_{n+2} with the same step size;

(b) $\|\,\text{error}\,\| < \alpha\,\varepsilon_{\mathrm{TOL}}$: accept the step and increase the step length;

(c) $\|\,\text{error}\,\| > \varepsilon_{\mathrm{TOL}}$: reject the step, recommence integration from t_n with smaller h.

In the case of (b) and (c), amending step size can be done with polynomial interpolation, although this means that we need to store past values in excess of what is necessary for simple implementation of both multistep methods.

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.

7.4 Embedded Runge–Kutta methods

The situation is more complicated with RK, since no single error constant determines local growth of the error. The approach of *embedded RK* requires, again, two (typically explicit) methods: an RK method of ν stages and order p, say, and another method, of $\nu+\ell$ stages, $\ell \geqslant 1$, and order $p+1$, such that *the first ν stages of both methods are identical.* The latter condition ensures that the cost of implementing the higher-order method is marginal, once we have computed the lower-order approximation. For example, consider (and verify)

$$\left.\begin{aligned}
&\left.\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}(t_n,\boldsymbol{y}_n),\\
\boldsymbol{k}_2 &= \boldsymbol{f}\big(t_n+\tfrac12 h, \boldsymbol{y}_n + \tfrac12 h\boldsymbol{k}_1\big),\\
\boldsymbol{y}^{[1]}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{k}_2;
\end{aligned}\right\}\ \text{order 2, local error } \mathcal{O}\big(h^3\big)\\
&\begin{aligned}
\boldsymbol{k}_3 &= \boldsymbol{f}(t_n+h, \boldsymbol{y}_n - h\boldsymbol{k}_1 + 2h\boldsymbol{k}_2),\\
\boldsymbol{y}^{[2]}_{n+1} &= \boldsymbol{y}_n + \tfrac16 h(\boldsymbol{k}_1 + 4\boldsymbol{k}_2 + \boldsymbol{k}_3)\,.
\end{aligned}
\end{aligned}\right\}\ \text{order 3, local error } \mathcal{O}\big(h^4\big)$$

We then estimate the error $\boldsymbol{y}^{[1]}_{n+1} - \boldsymbol{y}(t_{n+1})$ by

$$\boldsymbol{y}^{[1]}_{n+1} - \boldsymbol{y}(t_{n+1}) \approx \boldsymbol{y}^{[1]}_{n+1} - \boldsymbol{y}^{[2]}_{n+1}\,. \tag{7.5}$$

Remark. While it might look paradoxical, at least at first glance, the only purpose of the higher-order method is to provide error control for the lower-order one.

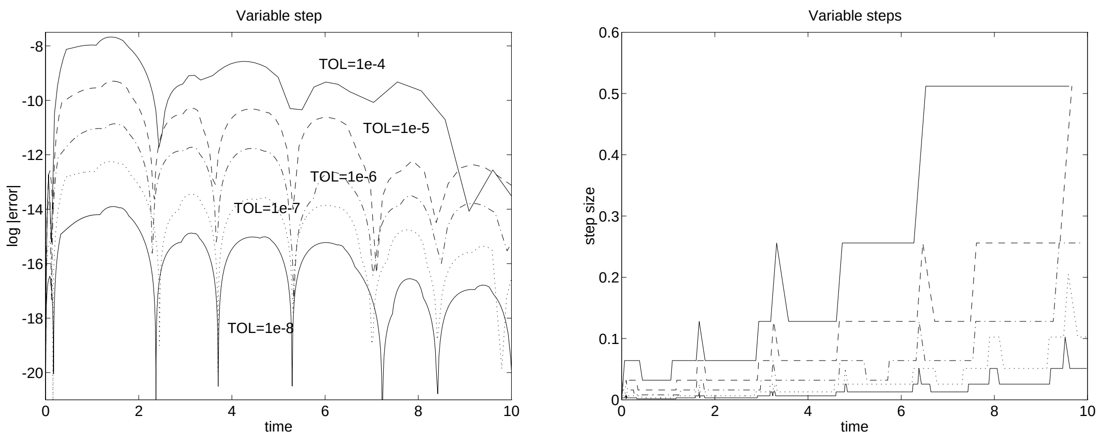


Figure 7.10: Adjustments of the step size in the solution of the equation $y' = -y + 2e^{-t}\cos 2t$, $y(0) = 0$.

7.5 The Zadunaisky device

Suppose that the ODE $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$, $\boldsymbol{y}(0) = \boldsymbol{y}_0$, is solved by an arbitrary numerical method of order p and that we have stored (not necessarily equidistant) past solution values $\boldsymbol{y}_n, \boldsymbol{y}_{n-1}, \ldots, \boldsymbol{y}_{n-p}$. We form an interpolating p^{th} degree polynomial (with vector coefficients) $\boldsymbol{d}$ such that $\boldsymbol{d}(t_{n-i}) = \boldsymbol{y}_{n-i}$, $i = 0, 1, \ldots, p$, and consider the differential equation

$$\boldsymbol{z}' = \boldsymbol{f}(t, \boldsymbol{z}) + \boldsymbol{d}'(t) - \boldsymbol{f}(t, \boldsymbol{d}), \qquad \boldsymbol{z}(t_n) = \boldsymbol{y}_n. \tag{7.6}$$

There are two important observations with regard to (7.6):

(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, (7.6) is a small perturbation of the original ODE.

(ii) The exact solution of (7.6) is $\boldsymbol{z}(t) = \boldsymbol{d}(t)$.

So, having produced $\boldsymbol{y}_{n+1}$ with our numerical method, we proceed to evaluate $\boldsymbol{z}_{n+1}$ as well, *using exactly the same method and implementation details.* 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 $\boldsymbol{y}_{n+1}$.

7.6 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.

7.7 Solving nonlinear algebraic systems

We have already observed that the implementation of an implicit ODE method, whether multistep or RK, requires the solution of (in general, nonlinear) algebraic equations in each step. For example, for an s-step method, we need to solve in each step the algebraic system

$$\boldsymbol{y}_{n+s} = \sigma_s h \boldsymbol{f}(t_{n+s}, \boldsymbol{y}_{n+s}) + \boldsymbol{v}, \tag{7.7}$$

where the vector $\boldsymbol{v}$ can be formed from past (hence known) solution values and their derivatives. The easiest approach is *functional iteration*

$$\boldsymbol{y}_{n+s}^{[j+1]} = \sigma_s h \boldsymbol{f}(t_{n+s}, \boldsymbol{y}_{n+s}^{[j]}) + \boldsymbol{v}, \qquad j = 0, 1, \ldots, \tag{7.8}$$

where $\boldsymbol{y}_{n+s}^{[0]}$ is typically provided by the predictor scheme, i.e. in the notation of (7.4a)

$$\boldsymbol{y}_{n+s}^{\mathrm{C}\,[0]} = \boldsymbol{y}_{n+s}^{\mathrm{P}}.$$

This is effective for *nonstiff* equations but fails for *stiff ODEs*, since the convergence of this iterative scheme requires a similar restriction on h as that we strive to avoid by choosing an implicit method in the first place! 11/10

If the ODE is stiff, we might prefer a *Newton–Raphson* method. Suppose that we wish to solve

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{y})\,. \tag{7.9}$$

Let $\boldsymbol{y}^{[j]}$ be the j^{th} approximation to the solution. We linearise (7.9) locally about $\boldsymbol{y} = \boldsymbol{y}^{[j]} + \boldsymbol{d}$ to obtain

$$\boldsymbol{y}^{[j]} + \boldsymbol{d} = \boldsymbol{g}(\boldsymbol{y}^{[j]} + \boldsymbol{d}) \approx \boldsymbol{g}(\boldsymbol{y}^{[j]}) + \boldsymbol{d} \cdot \boldsymbol{\nabla} \boldsymbol{g}(\boldsymbol{y}^{[j]})\,, \tag{7.10a}$$

or in suffix notation, after a little rearrangement,

$$\left(\delta_{ik} - \frac{\partial g_i}{\partial y_k}(\boldsymbol{y}^{[j]})\right) d_k \approx g_i(\boldsymbol{y}^{[j]}) - y_i^{[j]}\,. \tag{7.10b}$$

Define the *Jacobian matrix* to be

$$A_{ik}^{[j]} = \left(\delta_{ik} - \frac{\partial g_i}{\partial y_k}(\boldsymbol{y}^{[j]})\right)\,, \tag{7.11a}$$

or in matrix notation

$$\mathsf{A}^{[j]} = \left(\mathsf{I} - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}(\boldsymbol{y}^{[j]})\right)\,. \tag{7.11b}$$

Then from (7.10b)

$$\boldsymbol{d} = (\mathsf{A}^{[j]})^{-1} \left(\boldsymbol{g}(\boldsymbol{y}^{[j]}) - \boldsymbol{y}^{[j]}\right)\,. \tag{7.11c}$$

We now choose the $(j+1)^{\text{th}}$ approximation of the solution to be

$$\boldsymbol{y}^{[j+1]} = \boldsymbol{y}^{[j]} + \boldsymbol{d}\,. \tag{7.12a}$$

Applying the above method to (7.7) we obtain

$$\boldsymbol{y}_{n+s}^{[j+1]} = \boldsymbol{y}_{n+s}^{[j]} + \left[\mathsf{I} - \sigma_s h \frac{\partial \boldsymbol{f}(t_{n+s}, \boldsymbol{y}_{n+s}^{[j]})}{\partial \boldsymbol{y}}\right]^{-1} [\sigma_s h \boldsymbol{f}(t_{n+s}, \boldsymbol{y}_{n+s}^{[j]}) + \boldsymbol{v} - \boldsymbol{y}_{n+s}^{[j]}]. \tag{7.12b}$$

The snag with this approach is that repeatedly evaluating and inverting (e.g. by LU-factorization as in §8.2) the Jacobian matrix in every iteration is *very* expensive. A remedy is to implement the *modified Newton–Raphson method* and to use $\mathsf{A}^{[0]}$, rather than $\mathsf{A}^{[j]}$, for every iteration for a given step; (7.12b) then becomes

$$\boldsymbol{y}_{n+s}^{[j+1]} = \boldsymbol{y}_{n+s}^{[j]} + \left[\mathsf{I} - \sigma_s h \frac{\partial \boldsymbol{f}(t_{n+s}, \boldsymbol{y}_{n+s}^{[0]})}{\partial \boldsymbol{y}}\right]^{-1} [\sigma_s h \boldsymbol{f}(t_{n+s}, \boldsymbol{y}_{n+s}^{[j]}) + \boldsymbol{v} - \boldsymbol{y}_{n+s}^{[j]}]. \tag{7.12c}$$

11/11 Thus, the Jacobian need be evaluated only *once* a step.

Remarks

(i) The only role the Jacobian matrix plays in (7.12c) is to ensure convergence: its precise value makes no difference to the ultimate value of $\lim_{j\to\infty} \boldsymbol{y}_{n+s}^{[j]}$. Therefore we might replace it with a finite-difference approximation, and/or evaluate it once every several steps, and/or ...

(ii) Implementation of (7.12c) requires repeated solution of linear algebraic systems *with the same matrix.* In §8.2 we study LU factorization of matrices, and there we shall see that this remark can lead to substantial savings.

(iii) For stiff equations it is much cheaper to solve nonlinear algebraic equations with (7.12c) than using a minute step size with a 'bad' (e.g., explicit multistep or explicit RK) method.

7.8 *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 equations of 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.

11/12
11/13
11/14

8 Numerical Linear Algebra

8.1 Introduction

Problem 8.1. For a real $n\times n$ matrix A, and vectors $\boldsymbol{x}$ and $\boldsymbol{b}$, one of the most basic problems of numerical analysis is: solve

$$\mathsf{A}\boldsymbol{x} = \boldsymbol{b}. \tag{8.1}$$

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 inverse of the square matrix A can be expressed as

$$(\mathsf{A}^{-1})_{i,j} = \frac{1}{\det \mathsf{A}}\Delta_{j,i}\,, \tag{8.2a}$$

where $\Delta_{i,j}$ is the cofactor of the ij^{th} element of the matrix A. The required determinants can be evaluated by, say,

$$\det \mathsf{A} = \sum_{i_1 i_2 \dots i_n} \varepsilon_{i_1 i_2 \dots i_n} A_{i_1,1} A_{i_2,2} \dots A_{i_n,n}\,, \tag{8.2b}$$

or using the recursive definition of a determinant. Using these formulae (8.1) can be solved explicitly by Cramer's rule. Unfortunately, the number of operations increases like $(n+1)!$. Thus, on a 10^{10} flop/sec. computer

$$n=10 \quad\Rightarrow\quad 10^{-5}\text{ sec}, \qquad n=20 \quad\Rightarrow\quad 1\tfrac{3}{4}\text{ min}, \qquad n=30 \quad\Rightarrow\quad 4\times 10^4\text{ years}.$$

Thus we have to look for more practical methods.

8.1.1 Triangular matrices

An *upper triangular* $n\times n$ square matrix U has $U_{i,j}=0$ if $i>j$. Hence

$$\det \mathsf{U} = \prod_{i=1}^{n} U_{i,i}. \tag{8.3a}$$

Also, we can solve $\mathsf{U}\boldsymbol{x}=\boldsymbol{y}$ in $\mathcal{O}(n^2)$ computational operations[10] by so-called *back substitution*

$$x_n = \frac{y_n}{U_{n,n}}, \qquad x_i = \frac{1}{U_{i,i}}\left(y_i - \sum_{j=i+1}^{n} U_{i,j}x_j\right), \quad i=n-1,\dots,1. \tag{8.3b}$$

For instance

$$\begin{bmatrix} -3 & 2 & 3 \\ & 2 & 0 \\ & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} \begin{matrix} \uparrow \\ \uparrow \\ \uparrow \end{matrix} \quad\Rightarrow\quad \boldsymbol{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Similarly, if L is a *lower triangular* $n\times n$ square matrix (such that $L_{i,j}=0$ if $i<j$) then

$$\det \mathsf{L} = \prod_{i=1}^{n} L_{i,i}, \tag{8.4a}$$

and $\mathsf{L}\boldsymbol{y}=\boldsymbol{b}$ can be solved in $\mathcal{O}(n^2)$ operations by *forward substitution*

$$y_1 = \frac{b_1}{L_{1,1}}, \qquad y_i = \frac{1}{L_{i,i}}\left(b_i - \sum_{j=1}^{i-1} L_{i,j}y_j\right), \quad i=2,\dots,n-1. \tag{8.4b}$$

[10] Where, as usual, we only bother to count multiplications/divisions.

For instance

$$\begin{bmatrix} 1 & & \\ -2 & 1 & \\ 3 & -1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ 5 \end{bmatrix} \begin{matrix} \downarrow \\ \downarrow \\ \downarrow \end{matrix} \quad \Rightarrow \quad \boldsymbol{y} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}.$$

Hence, if we could manage to factorize A as

$$\mathsf{A} = \mathsf{LU},$$

where L and U are lower and upper triangular matrices respectively, then the solution of the system $\mathsf{A}\boldsymbol{x} = \mathsf{LU}\boldsymbol{x} = \mathsf{L}(\mathsf{U}\boldsymbol{x}) = \boldsymbol{b}$ could be split into the two cases

$$\mathsf{L}\boldsymbol{y} = \boldsymbol{b}, \quad \mathsf{U}\boldsymbol{x} = \boldsymbol{y}. \tag{8.5}$$

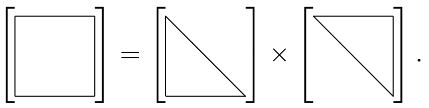
Both latter systems are triangular and can be solved in $\mathcal{O}(n^2)$ operations using (8.4b) and (8.3b) respectively (see also (8.12b)).

8.2 LU factorization and its generalizations

Definition 8.2. By the LU factorization of a $n \times n$ matrix A we understand a representation of A as a product

$$\mathsf{A} = \mathsf{LU},$$

where L is a *unit* lower triangular matrix (i.e. a [non-singular] lower triangular matrix with ones down the diagonal) and U is an upper triangular matrix. Therefore the factorization takes the form



Remarks

(a) The convention is that L is taken to be unit lower triangular; an equally possible convention is that U is taken to be unit upper triangular.

(b) We allow for the possibility that A is singular in our definition. However, we will be mainly concerned with the case when A is non-singular.

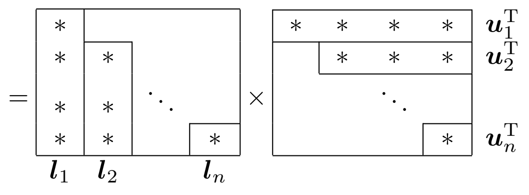
(c) A non-singular A may not have a LU factorization, while a singular A may have a LU factorization (see Question 1 on Example Sheet 3 for an example). We will return to this point subsequently (e.g. see Theorem 8.8 and the following examples).

8.2.1 The calculation of LU factorization

We present a method of the LU factorization that is based on a direct approach. So we assume that the factorization exists.

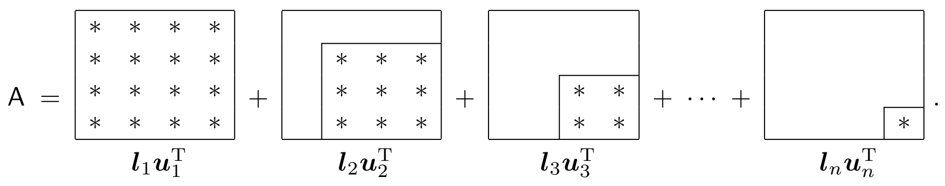
We denote the *columns* of L by $\boldsymbol{l}_1, \boldsymbol{l}_2, \dots, \boldsymbol{l}_n$ and the *rows* of U by $\boldsymbol{u}_1^{\mathrm{T}}, \boldsymbol{u}_2^{\mathrm{T}}, \dots, \boldsymbol{u}_n^{\mathrm{T}}$. Then

$$\mathsf{A} = \mathsf{LU} = [\ \boldsymbol{l}_1 \quad \boldsymbol{l}_2 \quad \cdots \quad \boldsymbol{l}_n\ ] \begin{bmatrix} \boldsymbol{u}_1^{\mathrm{T}} \\ \boldsymbol{u}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{u}_n^{\mathrm{T}} \end{bmatrix} \tag{8.6a}$$


$=$
$\boldsymbol{l}_1$
$\boldsymbol{l}_2$
$\boldsymbol{l}_n$
$\times$
$\boldsymbol{u}_1^{\mathrm{T}}$
$\boldsymbol{u}_2^{\mathrm{T}}$
$\boldsymbol{u}_n^{\mathrm{T}}$

$$= \sum_{k=1}^{n} \boldsymbol{l}_k \boldsymbol{u}_k^{\mathrm{T}}. \tag{8.6b}$$

Since the first $(k-1)$ components of $\boldsymbol{l}_k$ and $\boldsymbol{u}_k$ are all zero for $k \geqslant 2$, each rank-one matrix $\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ has zeros in its first $(k-1)$ rows and columns. Therefore


A =
+
+
+ ⋯ +
.
$\boldsymbol{l}_1\boldsymbol{u}_1^{\mathrm{T}}$
$\boldsymbol{l}_2\boldsymbol{u}_2^{\mathrm{T}}$
$\boldsymbol{l}_3\boldsymbol{u}_3^{\mathrm{T}}$
$\boldsymbol{l}_n\boldsymbol{u}_n^{\mathrm{T}}$
(8.6c)

We begin our calculation by extracting $\boldsymbol{l}_1$ and $\boldsymbol{u}_1^{\mathrm{T}}$ from A, and then proceed similarly to extract $\boldsymbol{l}_2$ and $\boldsymbol{u}_2^{\mathrm{T}}$, etc. First we note from (8.6c) that the first row of the matrix $\boldsymbol{l}_1\boldsymbol{u}_1^{\mathrm{T}}$ is $L_{1,1}\boldsymbol{u}_1^{\mathrm{T}} = \boldsymbol{u}_1^{\mathrm{T}}$, since the diagonal elements of L equal one.[11] Hence $\boldsymbol{u}_1^{\mathrm{T}}$ coincides with the first row of A, while the first column of $\boldsymbol{l}_1\boldsymbol{u}_1^{\mathrm{T}}$, which is $U_{1,1}\boldsymbol{l}_1 = A_{1,1}\boldsymbol{l}_1$,[12] coincides with the first column of A. Hence, with $\mathsf{A}^{(0)} = \mathsf{A}$,

$$\boldsymbol{u}_1^{\mathrm{T}} = [\text{the first row of } \mathsf{A}^{(0)}], \qquad \boldsymbol{l}_1 = [\text{the first column of } \mathsf{A}^{(0)}]/A_{1,1}^{(0)}.$$

Next, having found $\boldsymbol{l}_1$ and $\boldsymbol{u}_1$, we form the matrix

$$\mathsf{A}^{(1)} = \mathsf{A}^{(0)} - \boldsymbol{l}_1\boldsymbol{u}_1^{\mathrm{T}} = \sum_{k=2}^{n} \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}. \tag{8.7}$$

By construction, the first row and column of $\mathsf{A}^{(1)}$ are zero. It follows that $\boldsymbol{u}_2^{\mathrm{T}}$ is the second row of $\mathsf{A}^{(1)}$, while $\boldsymbol{l}_2$ is its second column, scaled so that $L_{2,2} = 1$; hence we obtain

$$\boldsymbol{u}_2^{\mathrm{T}} = [\text{the second row of } \mathsf{A}^{(1)}], \qquad \boldsymbol{l}_2 = [\text{the second column of } \mathsf{A}^{(1)}]/A_{2,2}^{(1)}.$$

Continuing this way we can formulate an algorithm.

The LU *algorithm.* Set $\mathsf{A}^{(0)} = \mathsf{A}$. For all $k = 1, 2, \ldots, n$ set $\boldsymbol{u}_k^{\mathrm{T}}$ to be the kth row of $\mathsf{A}^{(k-1)}$ and $\boldsymbol{l}_k$ to the kth column of $\mathsf{A}^{(k-1)}$, scaled so that $L_{k,k} = 1$. Further, calculate $\mathsf{A}^{(k)} = \mathsf{A}^{(k-1)} - \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ before incrementing k.

Hence we perform the calculations by the formulae for $k = 1, 2, \ldots, n$:

$$U_{k,j} = A_{k,j}^{(k-1)}, \qquad j = k, \ldots, n, \tag{8.8a}$$

$$L_{k,k} = 1, \quad L_{i,k} = \frac{A_{i,k}^{(k-1)}}{A_{k,k}^{(k-1)}}, \qquad i = k+1, \ldots, n, \tag{8.8b}$$

$$A_{i,j}^{(k)} = A_{i,j}^{(k-1)} - L_{i,k}U_{k,j}, \qquad i, j > k. \tag{8.8c}$$

Remarks.

(i) This construction shows that the condition

$$A_{k,k}^{(k-1)} \neq 0, \quad k = 1, \ldots, n-1, \tag{8.9}$$

is a *sufficient* condition for an LU factorization to exist and be unique. $A_{1,1}^{(0)}$ is just $A_{1,1}$, but the other values are only derived during construction. We shall see in §8.2.7 how to obtain equivalent conditions in terms of the original matrix A.

(ii) We note that $\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ stays the same if we replace

$$\boldsymbol{l}_k \to \alpha\boldsymbol{l}_k, \quad \boldsymbol{u}_k \to \alpha^{-1}\boldsymbol{u}_k, \quad \text{where} \quad \alpha \neq 0\,.$$

It is for this reason, subject to (8.9), that we may assume w.l.o.g. that all diagonal elements of L equal one.

[11] Similarly, it follows that the kth row of $\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ is $L_{k,k}\boldsymbol{u}_k^{\mathrm{T}} = \boldsymbol{u}_k^{\mathrm{T}}$.
[12] Similarly, the kth column of $\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ is $U_{k,k}\boldsymbol{l}_k$.

Example.

$$\mathsf{A} \;=\; \begin{bmatrix} 2 & 3 & -5 \\ 4 & 8 & -3 \\ -6 & 1 & 4 \end{bmatrix} \to \boldsymbol{l}_1 = \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix}, \quad \boldsymbol{u}_1^{\mathrm{T}} = [\,2 \quad 3 \quad -5\,], \;\; \boldsymbol{l}_1\boldsymbol{u}_1^{\mathrm{T}} = \begin{bmatrix} 2 & 3 & -5 \\ 4 & 6 & -10 \\ -6 & -9 & 15 \end{bmatrix},$$

$$\mathsf{A}^{(1)} = \begin{bmatrix} & & \\ & 2 & 7 \\ & 10 & -11 \end{bmatrix} \to \boldsymbol{l}_2 = \begin{bmatrix} 0 \\ 1 \\ 5 \end{bmatrix}, \quad \boldsymbol{u}_2^{\mathrm{T}} = [\,0 \quad 2 \quad 7\,], \;\; \boldsymbol{l}_2\boldsymbol{u}_2^{\mathrm{T}} = \begin{bmatrix} & & \\ & 2 & 7 \\ & 10 & 35 \end{bmatrix},$$

$$\mathsf{A}^{(2)} = \begin{bmatrix} & & \\ & & \\ & & -46 \end{bmatrix} \to \boldsymbol{l}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{u}_3^{\mathrm{T}} = [\,0 \quad 0 \quad -46\,], \;\; \boldsymbol{l}_3\boldsymbol{u}_3^{\mathrm{T}} = \mathsf{A}^{(2)},$$

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 A to accumulate L and U, and to store the elements of the matrices $\mathsf{A}^{(k)}$. Hence for our example:

$$\begin{bmatrix} 2 & 3 & -5 \\ 4 & 8 & -3 \\ -6 & 1 & 4 \end{bmatrix} \to \left[\begin{array}{c|cc} 2 & 3 & -5 \\ \cline{1-1} 2 & 2 & 7 \\ -3 & 10 & -11 \end{array}\right] \to \left[\begin{array}{cc|c} 2 & 3 & -5 \\ \cline{1-1} 2 & \multicolumn{1}{|c}{2} & 7 \\ \cline{2-2} -3 & 5 & -46 \end{array}\right].$$

Algorithm. From (8.8a), (8.8b) and (8.8c), the following pseudo-code computes the LU factorization by overwriting 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
```

8.2.2 Applications

The LU decomposition has many applications.

Testing non-singularity. $\mathsf{A} = \mathsf{LU}$ is non-singular iff all the diagonal elements of U are nonzero.

Calculation of the determinant.

$$\det \mathsf{A} = \det \mathsf{L} \det \mathsf{U} = \left(\prod_{k=1}^{n} L_{k,k}\right)\left(\prod_{k=1}^{n} U_{k,k}\right) = \left(\prod_{k=1}^{n} U_{k,k}\right). \tag{8.10}$$

Solution of linear systems. See (8.5), especially when there are multiple right hand sides.

Finding the inverse of A. The columns, say $\boldsymbol{x}_j$, of the inverse of A, are the solutions of

$$\mathsf{A}\boldsymbol{x}_j = \boldsymbol{e}_j\,. \tag{8.11a}$$

Similarly, the inverses of L and U can be obtained by solving $\mathsf{L}\boldsymbol{y}_j = \boldsymbol{e}_j$ and $\mathsf{U}\boldsymbol{z}_j = \boldsymbol{e}_j$. This construction demonstrates that the inverse of a lower/upper triangular matrix is lower/upper triangular. Thence, for a non-singular matrix,

$$\mathsf{A}^{-1} = \mathsf{U}^{-1}\mathsf{L}^{-1}. \tag{8.11b}$$

8.2.3 Cost

In calculating cost it is only multiplications and divisions that matter. For the k-th step of the LU algorithm we see from (8.8b) and (8.8c) that we perform $(n-k)$ divisions to determine the components of $\boldsymbol{l}_k$ and $(n-k)^2$ multiplications in finding $\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$. Hence

$$N_{LU}=\sum_{k=1}^{n-1}\left[(n-k)^2+(n-k)\right]=\tfrac{1}{3}\,n^3+\mathcal{O}(n^2). \tag{8.12a}$$

Solving $\mathsf{L}\boldsymbol{y}=\mathbf{b}$ by forward substitution we use k multiplications/divisions to determine y_k, and similarly for back substitution, thus

$$N_F=N_B=\sum_{k=1}^{n}k\sim\tfrac{1}{2}\,n^2. \tag{8.12b}$$

8.2.4 Relation to Gaussian elimination

At the kth step of the 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^{th} row of A^k is the i^{th} row of $\mathsf{A}^{(k-1)}$ minus $L_{i,k}$ times $\boldsymbol{u}_k^{\mathrm{T}}$ (the k^{th} row of $\mathsf{A}^{(k-1)}$), i.e.

$$[\text{the } i\text{th row of } \mathsf{A}^{(k)}] \;=\; [\text{the } i\text{th row of } \mathsf{A}^{(k-1)}] \;-\; L_{i,k}\times\; [\text{the } k\text{th row of } \mathsf{A}^{(k-1)}],$$

where the multipliers $L_{i,k}=A_{i,k}^{(k-1)}/A_{k,k}^{(k-1)}$ are chosen so that, at the outcome, the kth column of $\mathsf{A}^{(k)}$ is zero. This construction is analogous to Gaussian elimination for solving $\mathsf{A}\boldsymbol{x}=\boldsymbol{b}$.

Example.

$$\begin{bmatrix} 2 & 3 & -5\\ 4 & 8 & -3\\ -6 & 1 & 4\end{bmatrix} \xrightarrow{\substack{2^{\text{nd}}-1^{\text{st}}\cdot\ 2\\ 3^{\text{rd}}-1^{\text{st}}\cdot\ (-3)}} \left[\begin{array}{c|cc} 2 & 3 & -5\\ \hline 0 & 2 & 7\\ 0 & 10 & -11\end{array}\right] \xrightarrow{3^{\text{rd}}-2^{\text{nd}}\cdot\ 5} \left[\begin{array}{cc|c} 2 & 3 & -5\\ \hline 0 & 2 & 7\\ \cline{2-2} 0 & 0 & -46\end{array}\right].$$

If at each step we put the multipliers $L_{i,k}$ into the spare sub-diagonal part of A, then we obtain exactly the same form of the LU factorization as above.

$$\begin{bmatrix} 2 & 3 & -5\\ 4 & 8 & -3\\ -6 & 1 & 4\end{bmatrix} \xrightarrow{\substack{2^{\text{nd}}-1^{\text{st}}\cdot\ 2\\ 3^{\text{rd}}-1^{\text{st}}\cdot\ (-3)}} \left[\begin{array}{c|cc} 2 & 3 & -5\\ \hline 2 & 2 & 7\\ -3 & 10 & -11\end{array}\right] \xrightarrow{3^{\text{rd}}-2^{\text{nd}}\cdot\ 5} \left[\begin{array}{cc|c} 2 & 3 & -5\\ \hline 2 & 2 & 7\\ \cline{2-2} -3 & 5 & -46\end{array}\right].$$

Remark. An important difference between LU algorithm and Gaussian elimination is that in LU we do not consider the right hand side $\boldsymbol{b}$ until the factorization is complete. For instance, this is useful 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, whereas with LU factorisation $\mathcal{O}(n^3)$ operations are required for the initial factorisation, but then the solution for each new $\boldsymbol{b}$ only requires $\mathcal{O}(n^2)$ operations (for the back- and forward substitutions).

12/10

8.2.5 Pivoting

Column pivoting: description. Naive LU factorization fails when, for example, $A_{k,k}^{(k-1)}=0$. A remedy is to exchange rows of A by picking a suitable *pivotal equation.* This technique is called *column pivoting* (or *partial pivoting*), and is equivalent to picking a suitable equation for eliminating the unknown in Gaussian elimination. Specifically, column pivoting means that, having obtained $\mathsf{A}^{(k-1)}$, we exchange two rows of $\mathsf{A}^{(k-1)}$ so that the element of largest magnitude in the k^{th} column is in the 'pivotal position' (k,k). In other words,

$$\left|A_{k,k}^{(k-1)}\right|=\max_{i=k\ldots n}\left|A_{i,k}^{(k-1)}\right|.$$

The exchange of rows, say k and p, can be regarded as the pre-multiplication of the relevant matrix $\mathsf{A}^{(k-1)}$ by a permutation matrix P_k,

$$\mathsf{P}_k = \begin{array}{c} \begin{pmatrix} 1 & & & & & & & & & 0 \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & 0 & & & 1 & & & \\ & & & & 1 & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & 1 & & & \\ & & & 1 & & & 0 & & & \\ & & & & & & & 1 & & \\ & & & & & & & & \ddots & \\ 0 & & & & & & & & & 1 \end{pmatrix} \end{array} \begin{array}{l} \\ \\ \\ \leftarrow \text{ row } k \\ \\ \\ \\ \leftarrow \text{ row } p \\ \\ \\ \\ \end{array} . \tag{8.13}$$

Column pivoting: algorithm Suppose $A^{(k-1)}_{k,k} = 0$, or is close to zero, i.e.

$$\mathsf{A}^{(k-1)} = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & 0 & A^{(k-1)}_{k,k+1} & \cdots & A^{(k-1)}_{k,n} \\ \vdots & \ddots & 0 & A^{(k-1)}_{k+1,k} & A^{(k-1)}_{k+1,k+1} & \cdots & A^{(k-1)}_{k+1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & A^{(k-1)}_{n,k} & A^{(k-1)}_{n,k+1} & \cdots & A^{(k-1)}_{n,n} \end{pmatrix} . \tag{8.14a}$$

First identify p such that

$$\left| A^{(k-1)}_{p,k} \right| = \max_i \left| A^{(k-1)}_{i,k} \right| , \tag{8.14b}$$

then swap rows p and k of $\mathsf{A}^{(k-1)}$ using P_k as defined in (8.13), then construct $\boldsymbol{l}_k$ and $\boldsymbol{u}_k$ from $\mathsf{P}_k\mathsf{A}^{(k-1)}$ using the same algorithm as before, and then form

$$\mathsf{A}^{(k)} = \mathsf{P}_k\mathsf{A}^{(k-1)} - \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}} . \tag{8.15a}$$

By recursion

$$\begin{aligned} \mathsf{A}^{(k)} &= \mathsf{P}_k(\mathsf{P}_{k-1}\mathsf{A}^{(k-2)} - \boldsymbol{l}_{k-1}\boldsymbol{u}_{k-1}^{\mathrm{T}}) - \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}} \\ &= \mathsf{P}_k\mathsf{P}_{k-1}\mathsf{A}^{(k-2)} - \mathsf{P}_k\boldsymbol{l}_{k-1}\boldsymbol{u}_{k-1}^{\mathrm{T}} - \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}} \\ &= \dots . \end{aligned}$$

Hence, on defining for convenience $\mathsf{P}_n = \mathsf{I}$ and $\mathsf{P} = \mathsf{P}_n\mathsf{P}_{n-1}\dots\mathsf{P}_1$, we find that

$$\mathsf{PA} \equiv \mathsf{P}_n\mathsf{P}_{n-1}\dots\mathsf{P}_1\mathsf{A} = \sum_{k=1}^{n-1} \mathsf{P}_n\dots\mathsf{P}_{k+1}\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}} \; + \; \boldsymbol{l}_n\boldsymbol{u}_n^{\mathrm{T}} = \mathsf{LU} , \tag{8.15b}$$

where

$$\mathsf{L} = [\ \mathsf{P}_n\dots\mathsf{P}_2\boldsymbol{l}_1 \quad \dots \quad \mathsf{P}_n\dots\mathsf{P}_{k+1}\boldsymbol{l}_k \quad \dots \quad \boldsymbol{l}_n\ ] , \quad \text{and} \quad \mathsf{U} = \begin{bmatrix} \boldsymbol{u}_1^{\mathrm{T}} \\ \boldsymbol{u}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{u}_n^{\mathrm{T}} \end{bmatrix} . \tag{8.15c}$$

Remarks

(i) At each stage the permutation of rows is applied to the portion of L that has been formed already (i.e. the first $k-1$ columns of L), and then only to the bottom $n-k+1$ components of these vectors.

(ii) We need to record the permutation of rows to solve for the right hand side and/or to compute the determinant.

Column pivoting: the zero-column case. Column pivoting copes with zeros at the pivot position, except when the whole k^{th} column of $\mathsf{A}^{(k-1)}$ is zero. This can only happen if A is singular, and in that case we let $\boldsymbol{l}_k$ be the k^{th} unit vector, while taking $\boldsymbol{u}_k^{\mathrm{T}}$ as the k-th row of $\mathsf{A}^{(k-1)}$ as before. This choice preserves the condition that the matrix $\boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$ has the same k-th row and column as $\mathsf{A}^{(k-1)}$. Thus

$$\mathsf{A}^{(k)} = \mathsf{A}^{(k-1)} - \boldsymbol{l}_k\boldsymbol{u}_k^{\mathrm{T}}$$

still has zeros in its k-th row and column as required.

Column pivoting: remark. An important advantage of *column pivoting* is that every element of L has magnitude at most one, i.e. $|L_{i,j}| \leqslant 1$ for all $i,j = 1,\dots,n$. This avoids not just division by zero but also tends to reduce the chance of very large numbers occurring during the factorization, a
12/11 phenomenon that might lead to accumulation of *round-off error* and to *ill conditioning*.

Example.

$$\begin{bmatrix} 2 & 1 & 1 \\ \mathbf{4} & 1 & 0 \\ -2 & 2 & 1 \end{bmatrix} \overset{\left[\begin{smallmatrix} 2\\ 1\\ \mathbf{3}\end{smallmatrix}\right]}{\rightarrow} \begin{bmatrix} 4 & 1 & 0 \\ 2 & 1 & 1 \\ -2 & 2 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \\ -\frac{1}{2} & \mathbf{\frac{5}{2}} & 1 \end{bmatrix} \overset{\left[\begin{smallmatrix} 2\\ 3\\ \mathbf{1}\end{smallmatrix}\right]}{\rightarrow} \begin{bmatrix} 4 & 1 & 0 \\ -\frac{1}{2} & \frac{5}{2} & 1 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 1 & 0 \\ -\frac{1}{2} & \frac{5}{2} & 1 \\ \frac{1}{2} & \frac{1}{5} & \frac{4}{5} \end{bmatrix},$$

$$\mathsf{PA} = \mathsf{LU} \quad\Rightarrow\quad \mathsf{A} = \mathsf{P}^{\mathrm{T}}\mathsf{LU}, \quad \mathsf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathsf{L} = \begin{bmatrix} 1 & & \\ -\frac{1}{2} & 1 & \\ \frac{1}{2} & \frac{1}{5} & 1 \end{bmatrix}, \quad \mathsf{U} = \begin{bmatrix} 4 & 1 & 0 \\ & \frac{5}{2} & 1 \\ & & \frac{4}{5} \end{bmatrix},$$

where P is the permutation matrix that sends *row 2 to row 1*, `row 3 to row 2`, and **row 1 to row 3**.

Row pivoting. In row pivoting one exchanges columns of $\mathsf{A}^{(k-1)}$, rather than rows (sic!), so that $(\mathsf{A}^{(k-1)})_{k,k}$ becomes the element of largest modulus in the k-th row of $\mathsf{A}^{(k-1)}$.

Total pivoting. Total pivoting corresponds to the exchange of both rows and columns, so that the modulus
12/12 of the pivotal element $(\mathsf{A}^{(k-1)})_{k,k}$ is maximized.
12/13
12/14

8.2.6 Further examples (*Unlectured*)

LU *factorization.*

$$\begin{bmatrix} -3 & 2 & 3 & -1 \\ 6 & -2 & -6 & 0 \\ -9 & 4 & 10 & 3 \\ 12 & -4 & -13 & -5 \end{bmatrix} \xrightarrow[4^{\text{th}}-1^{\text{st}}\cdot(-4)]{\substack{2^{\text{nd}}-1^{\text{st}}\cdot(-2)\\ 3^{\text{rd}}-1^{\text{st}}\cdot\ \ 3}} \begin{bmatrix} -3 & 2 & 3 & -1 \\ -2 & 2 & 0 & -2 \\ 3 & -2 & 1 & 6 \\ -4 & 4 & -1 & -9 \end{bmatrix}$$

$$\xrightarrow{\substack{3^{\text{rd}}-2^{\text{nd}}\cdot(-1)\\ 4^{\text{th}}-2^{\text{nd}}\cdot\ \ 2}} \begin{bmatrix} -3 & 2 & 3 & -1 \\ -2 & 2 & 0 & -2 \\ 3 & -1 & 1 & 4 \\ -4 & 2 & -1 & -5 \end{bmatrix}$$

$$\xrightarrow{4^{\text{th}}-3^{\text{rd}}\cdot(-1)} \begin{bmatrix} -3 & 2 & 3 & -1 \\ -2 & 2 & 0 & -2 \\ 3 & -1 & 1 & 4 \\ -4 & 2 & -1 & -1 \end{bmatrix}$$

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.*

$$\begin{bmatrix} -3 & 2 & 3 & -1 \\ 6 & -2 & -6 & 0 \\ -9 & 4 & 10 & 3 \\ \mathbf{12} & -4 & -13 & -5 \end{bmatrix} \xrightarrow{\begin{bmatrix} 4 \\ 2 \\ 3 \\ 1 \end{bmatrix}} \begin{bmatrix} 12 & -4 & -13 & -5 \\ 6 & -2 & -6 & 0 \\ -9 & 4 & 10 & 3 \\ -3 & 2 & 3 & -1 \end{bmatrix} \to \left[\begin{array}{c|ccc} 12 & -4 & -13 & -5 \\ \hline \frac{1}{2} & 0 & \frac{1}{2} & \frac{5}{2} \\ -\frac{3}{4} & \mathbf{1} & \frac{1}{4} & -\frac{3}{4} \\ -\frac{1}{4} & 1 & -\frac{1}{4} & -\frac{9}{4} \end{array}\right]$$

$$\xrightarrow{\begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}} \left[\begin{array}{c|ccc} 12 & -4 & -13 & -5 \\ \hline -\frac{3}{4} & 1 & \frac{1}{4} & -\frac{3}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{5}{2} \\ -\frac{1}{4} & 1 & -\frac{1}{4} & -\frac{9}{4} \end{array}\right] \to \left[\begin{array}{cc|cc} 12 & -4 & -13 & -5 \\ -\frac{3}{4} & 1 & \frac{1}{4} & -\frac{3}{4} \\ \hline \frac{1}{2} & 0 & \frac{1}{2} & \frac{5}{2} \\ -\frac{1}{4} & 1 & -\frac{1}{2} & -\frac{3}{2} \end{array}\right] \to \left[\begin{array}{ccc|c} 12 & -4 & -13 & -5 \\ -\frac{3}{4} & 1 & \frac{1}{4} & -\frac{3}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{5}{2} \\ \hline -\frac{1}{4} & 1 & -1 & 1 \end{array}\right]$$

i.e.

$$\mathsf{A} = \mathsf{P}^{\mathrm{T}}\mathsf{LU}, \quad \mathsf{P} = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & 1 & & \\ 1 & & & \end{bmatrix}, \quad \mathsf{L} = \begin{bmatrix} 1 & & & \\ -\frac{3}{4} & 1 & & \\ \frac{1}{2} & 0 & 1 & \\ -\frac{1}{4} & 1 & 1 & 1 \end{bmatrix}, \quad \mathsf{U} = \begin{bmatrix} 12 & -4 & -13 & -5 \\ & 1 & \frac{1}{4} & -\frac{3}{4} \\ & & \frac{1}{2} & \frac{5}{2} \\ & & & 1 \end{bmatrix}$$

8.2.7 Existence and uniqueness of the LU factorization

For a $n \times n$ square matrix A, define its leading $k \times k$ submatrices A_k, for $k = 1, \ldots, n$, by

$$(\mathsf{A}_k)_{i,j} = A_{i,j}\,, \quad \text{for } i, j = 1, \ldots, k. \tag{8.16}$$

Theorem 8.3. *A sufficient condition for a* LU *factorization of a matrix* A *to exist and be unique is that the* A_k*, for* $k = 1, \ldots, n-1$*, are non-singular.*

Proof. (Unlectured.) We use induction. For $n = 1$ the result is clear.

Assume the result for $(n-1) \times (n-1)$ matrices. Partition the $n \times n$ matrix A as

$$\mathsf{A} = \left[\begin{array}{c|c} \mathsf{A}_{n-1} & \boldsymbol{b} \\ \hline \boldsymbol{c}^{\mathrm{T}} & A_{n,n} \end{array}\right] . \tag{8.17a}$$

We require

$$\mathsf{A} = \mathsf{LU} \quad \text{with} \quad \mathsf{L} = \left[\begin{array}{c|c} \mathsf{L}_{n-1} & \boldsymbol{0} \\ \hline \boldsymbol{x}^{\mathrm{T}} & 1 \end{array}\right], \quad \mathsf{U} = \left[\begin{array}{c|c} \mathsf{U}_{n-1} & \boldsymbol{y} \\ \hline \boldsymbol{0}^{\mathrm{T}} & U_{n,n} \end{array}\right], \tag{8.17b}$$

where L_{n-1}, U_{n-1}, $\boldsymbol{x}^{\mathrm{T}}$, $\boldsymbol{y}$ and $U_{n,n}$ are to be determined. Multiplying out these block matrices we see that we want to have

$$\mathsf{A} = \left[\begin{array}{c|c} \mathsf{A}_{n-1} & \boldsymbol{b} \\ \hline \boldsymbol{c}^{\mathrm{T}} & A_{n,n} \end{array}\right] = \left[\begin{array}{c|c} \mathsf{L}_{n-1}\mathsf{U}_{n-1} & \mathsf{L}_{n-1}\boldsymbol{y} \\ \hline \boldsymbol{x}^{\mathrm{T}}\mathsf{U}_{n-1} & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} + U_{n,n} \end{array}\right] . \tag{8.17c}$$

By virtue of the induction assumption L_{n-1} and U_{n-1} exist; further both are non-singular since it is assumed that 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 solved to obtain

$$\boldsymbol{y} = \mathsf{L}_{n-1}^{-1}\boldsymbol{b}, \quad \boldsymbol{x}^{\mathrm{T}} = \boldsymbol{c}^{\mathrm{T}}\mathsf{U}_{n-1}^{-1}, \quad U_{n,n} = A_{n,n} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}. \tag{8.17d}$$

So, by construction, there exists a unique factorization $\mathsf{A} = \mathsf{LU}$ with $L_{ii} = 1$. □

Corollary 8.4.

$$A_{1,1}^{(0)} = \det(\mathsf{A}_1) , \tag{8.18a}$$

$$A_{k,k}^{(k-1)} = \frac{\det(\mathsf{A}_k)}{\det(\mathsf{A}_{k-1})} . \tag{8.18b}$$

Proof. From (8.17b) and (8.17c)

$$\det(\mathsf{A}_n) = \det(\mathsf{A}) = \det(\mathsf{L})\det(\mathsf{U}) = \det(\mathsf{U}) = U_{n,n}\det(\mathsf{U}_{n-1}) ,$$
$$\det(\mathsf{A}_{n-1}) = \det(\mathsf{L}_{n-1})\det(\mathsf{U}_{n-1}) = \det(\mathsf{U}_{n-1}) .$$

So $\det(\mathsf{A}_n) = U_{n,n}\det(\mathsf{A}_{n-1})$, and thence by induction

$$U_{k,k} = \frac{\det(\mathsf{A}_k)}{\det(\mathsf{A}_{k-1})} . \tag{8.19}$$

Further $U_{1,1} = A_{1,1} = \det(\mathsf{A}_1)$, and from (8.8a) with $j = k$, we have that $U_{k,k} = A_{k,k}^{(k-1)}$. □

Theorem 8.5. *If A is non-singular and an LU factorization of A exists, then A_k is non-singular for $k = 1, \ldots, n-1$.*

Proof. From (8.19), or otherwise,

$$\det(\mathsf{A}) = \prod_{i=1}^{n} U_{i,i} \neq 0 , \quad \text{and hence} \quad \det(\mathsf{A}_k) = \prod_{i=1}^{k} U_{i,i} \neq 0 .$$ □

Corollary 8.6. *Combining Theorem 8.5 and Theorem 8.3 we see that if A is non-singular and an LU factorization exists, then this LU factorization is unique.*

It is also possible to prove uniqueness directly.

Theorem 8.7 (Uniqueness)**.** *If A is non-singular, it is impossible for more than one LU factorization to exist, i.e.*

$$\mathsf{A} = \mathsf{L}_1\mathsf{U}_1 = \mathsf{L}_2\mathsf{U}_2 \quad \textit{implies} \quad \mathsf{L}_1 = \mathsf{L}_2, \quad \mathsf{U}_1 = \mathsf{U}_2.$$

Proof. If A is non-singular, then both U_1 and 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, the inverse of a lower/upper triangular matrix is lower/upper triangular. Consequently, V is simultaneously lower and upper triangular, hence it is diagonal. Since $\mathsf{L}_2^{-1}\mathsf{L}_1$ has unit diagonal, we obtain $\mathsf{V} = \mathsf{I}$. □

Theorem 8.8 (Unproved)**.** *If it is* not *the case that the* A_k, $k = 1, \dots, n-1$ *are all non-singular, then* either *no* LU *factorization exists,* or *the* LU *factorization is not unique.*

Examples.

(i) There is no LU factorization for

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

or, indeed, for any non-singular $n \times n$ matrix A such that $\det(\mathsf{A}_k) = 0$ for some $k = 1, \dots, n-1$.

(ii) *Some* singular matrices may be LU factorized in many ways, e.g.

$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 0 & \frac{1}{2} \end{bmatrix}.$$

Remark. Every non-singular matrix A admits an LU factorization with pivoting, i.e. for every non-singular matrix A there exists a permutation matrix P such that $\mathsf{PA} = \mathsf{LU}$.

Corollary 8.9. *A [non-singular]* $n \times n$ *matrix* A *such that* $\det(\mathsf{A}_k) \neq 0$ *for* $k = 1, \dots, n$ *has a unique factorization*

$$\mathsf{A} = \mathsf{LDU}', \tag{8.20}$$

where both L *and* U' *have unit diagonals, and* D *is a non-singular diagonal matrix, i.e. we can express* A *as*

$$\mathsf{A} = \begin{bmatrix} \boldsymbol{l}_1 & \boldsymbol{l}_2 & \cdots \boldsymbol{l}_n \end{bmatrix} \begin{bmatrix} D_{1,1} & 0 & \cdots & 0 \\ 0 & D_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & D_{n,n} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}'^{\mathrm{T}}_1 \\ \boldsymbol{u}'^{\mathrm{T}}_2 \\ \vdots \\ \boldsymbol{u}'^{\mathrm{T}}_n \end{bmatrix} = \sum_{k=1}^{n} D_{k,k}\boldsymbol{l}_k\boldsymbol{u}'^{\mathrm{T}}_k \tag{8.21}$$

where, as before, $\boldsymbol{l}_k$ *is the* k^{th} *column of* L *and* $\boldsymbol{u}'_k$ *is the* k^{th} *row of* U'.

Proof.

$$D_{i,i} = U_{i,i} \neq 0\,, \quad U'_{i,j} = U_{i,j}/U_{i,i}\,.$$ □

8.3 Factorization of structured matrices

8.3.1 Symmetric matrices

Let A be a real $n \times n$ symmetric matrix (i.e., $A_{k,\ell} = A_{\ell,k}$). If A has an LU factorization then we can take advantage of symmetry to express A in the form of a product $\mathsf{LDL}^{\mathrm{T}}$, where L is $n \times n$ unit lower triangular and D is a diagonal matrix.

Corollary 8.10. *Let* A *be a real* $n \times n$ *symmetric matrix . If* $\mathsf{A}_k \neq 0$ *for* $k = 1, \dots, n$ *then there is a unique factorization*

$$\mathsf{A} = \mathsf{LDL}^{\mathrm{T}}. \tag{8.22}$$

Proof. $\mathsf{A} = \mathsf{LDU}'$ from (8.20), and by symmetry,

$$\mathsf{LDU}' = \mathsf{A} = \mathsf{A}^{\mathrm{T}} = \mathsf{U}'^{\mathrm{T}}\mathsf{DL}^{\mathrm{T}}.$$

Since the LDU' factorization is unique, $\mathsf{U}' = \mathsf{L}^{\mathrm{T}}$. □

Remark. Clearly $\mathsf{U} = \mathsf{DL}^{\mathrm{T}}$. However an advantage of this form is that it lends itself better to exploitation of symmetry and requires roughly half the storage of conventional LU. Specifically, to compute this factorization, we let $\mathsf{A}^{(0)} = \mathsf{A}$ and for $k = 1, 2, \dots, n$ let $\boldsymbol{l}_k$ be the multiple of the kth column of $\mathsf{A}^{(k-1)}$ such that $L_{k,k} = 1$. We then set

$$D_{k,k} = A_{k,k}^{(k-1)} \quad \text{and form} \quad \mathsf{A}^{(k)} = \mathsf{A}^{(k-1)} - D_{k,k}\boldsymbol{l}_k\boldsymbol{l}_k^{\mathrm{T}}. \tag{8.23}$$

Note, however, that pivoting can/will destroy the symmetry.

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

$$\mathsf{A} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}.$$

8.3.2 Positive definite matrices

Definition. A matrix A is *positive definite* if $\boldsymbol{x}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \mathbf{0}$.

Theorem 8.11. *If A is a real $n \times n$ positive definite matrix, then it is non-singular.*

Proof. We prove by contradiction. Assume that A is singular; then there exists $\boldsymbol{z} \neq 0$ such that

$$\mathsf{A}\boldsymbol{z} = \mathbf{0}.$$

Hence

$$\boldsymbol{z}^{\mathrm{T}}\mathsf{A}\boldsymbol{z} = \mathbf{0},$$

and so A cannot be positive definite. □

Theorem 8.12. *If A is positive definite matrix, then the A_k are non-singular for $k = 1, \dots, n-1$.*

Proof. Let $\boldsymbol{y} \in \mathbb{R}^k \setminus \{\mathbf{0}\}$, $k = 1, \dots, n-1$. Choose $\boldsymbol{x} \in \mathbb{R}^n$ so that the first k components equal $\boldsymbol{y}$ and the bottom $n - k$ elements are all zero. Then

$$\boldsymbol{y}^{\mathrm{T}}\mathsf{A}_k\boldsymbol{y} = \boldsymbol{x}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} > 0.$$

Hence the A_k are all positive definite and so, from Theorem 8.11, non-singular. □

Corollary 8.13. *If A is a real $n \times n$ positive definite matrix then, from Theorem 8.3, a unique LU factorization exists.*

8.3.3 Symmetric positive definite matrices

Theorem 8.14. *Let A be a real $n \times n$ symmetric matrix. It is positive definite if and only if it has an $\mathsf{LDL}^{\mathrm{T}}$ factorization in which the diagonal elements of D are all positive.*

Proof. Suppose that $\mathsf{A} = \mathsf{LDL}^{\mathrm{T}}$, with the diagonal elements of D all positive, and let $\boldsymbol{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$. Since L is non-singular, $\boldsymbol{y} = \mathsf{L}^{\mathrm{T}}\boldsymbol{x} \neq \mathbf{0}$. Then

$$\boldsymbol{x}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} = \boldsymbol{y}^{\mathrm{T}}\mathsf{D}\boldsymbol{y} = \sum_{k=1}^{n} D_{k,k}y_k^2 > 0,$$

hence A is positive definite.

Conversely, suppose that A is positive definite. Then from Corollary 8.13 the A_k are non-singular and a unique LU factorization exists. Next, from Corollary 8.10 the factorization can be written in the form $\mathsf{A} = \mathsf{LDL}^{\mathrm{T}}$. Finally, let $\boldsymbol{y}_k$ be defined such that

$$\mathsf{L}^{\mathrm{T}}\boldsymbol{y}_k = \boldsymbol{e}_k\,, \quad \text{i.e.} \quad \boldsymbol{y}_k = (\mathsf{L}^{\mathrm{T}})^{-1}\boldsymbol{e}_k \neq \mathbf{0}\,;$$

then

$$D_{k,k} = \boldsymbol{e}_k^{\mathrm{T}}\mathsf{D}\boldsymbol{e}_k = \boldsymbol{y}_k^{\mathrm{T}}\mathsf{LDL}^{\mathrm{T}}\boldsymbol{y}_k = \boldsymbol{y}_k^{\mathrm{T}}\mathsf{A}\boldsymbol{y}_k > 0\,. \qquad \square$$

Corollary 8.15. *It is possible to check if a symmetric matrix is positive definite by trying to form its* $\mathsf{LDL}^{\mathrm{T}}$ *factorization.*

Example. The matrix below is positive definite.

$$\begin{bmatrix} 2 & 6 & -2 \\ 6 & 21 & 0 \\ -2 & 0 & 16 \end{bmatrix} \rightarrow \left[\begin{array}{r|rr} 2 & 6 & -2 \\ \hline 3 & 3 & 6 \\ -1 & 6 & 14 \end{array}\right] \rightarrow \left[\begin{array}{r|r|r} 2 & 6 & -2 \\ \hline 3 & 3 & 6 \\ \cline{2-2} -1 & 2 & 2 \end{array}\right] \Rightarrow L = \begin{bmatrix} 1 & & \\ 3 & 1 & \\ -1 & 2 & 1 \end{bmatrix}, \; D = \begin{bmatrix} 2 & & \\ & 3 & \\ & & 2 \end{bmatrix}.$$

Cholesky factorization. Define $\mathsf{D}^{1/2}$ as the diagonal matrix whose (k,k) element is $D_{k,k}^{1/2}$, hence

$$\mathsf{D}^{1/2}\mathsf{D}^{1/2} = \mathsf{D}.$$

Then, A being symmetric positive definite, we can write

$$\mathsf{A} = (\mathsf{LD}^{1/2})(\mathsf{D}^{1/2}\mathsf{L}^{\mathrm{T}}) = (\mathsf{LD}^{1/2})(\mathsf{LD}^{1/2})^{\mathrm{T}}.$$

In other words, by defining the lower triangular matrix G by $\mathsf{G} = \mathsf{LD}^{1/2}$, we obtain the *Cholesky factorization*

$$\mathsf{A} = \mathsf{GG}^{\mathrm{T}}. \tag{8.24}$$

8.3.4 Sparse matrices

Definition 8.16. A matrix $\mathsf{A} \in \mathbb{R}^{n\times n}$ is called a *sparse* matrix if nearly all elements of A are zero. Examples are band matrices and block band matrices

Definition 8.17. A matrix A is called a *band matrix* if there exists an integer $r < n-1$ such that

$$A_{i,j} = 0 \quad \text{for all} \quad |i-j| > r.$$

In other words, all nonzero elements of A reside in a band of width $2r+1$ along the main diagonal.

$$r=1 \begin{bmatrix} * & * & & & & & \\ * & * & * & & & & \\ & * & * & * & & & \\ & & * & * & * & & \\ & & & * & * & * & \\ & & & & * & * & * \\ & & & & & * & * \end{bmatrix}, \quad r=2 \begin{bmatrix} * & * & * & & & & \\ * & * & * & * & & & \\ * & * & * & * & * & & \\ & * & * & * & * & * & \\ & & * & * & * & * & * \\ & & & * & * & * & * \\ & & & & * & * & * \end{bmatrix}, \quad r=3 \begin{bmatrix} * & * & * & * & & & \\ * & * & * & * & * & & \\ * & * & * & * & * & * & \\ * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & & * & * & * & * & * \\ & & & * & * & * & * \end{bmatrix}.$$

It is often required to solve *very* large systems $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ ($n = 10^9$ is a relatively modest example) where A is sparse. The efficient solution of such systems should exploit the sparsity. In particular, we wish the matrices L and U to inherit as much as possible of the sparsity of A (so that the cost of computing $\mathsf{U}\boldsymbol{x}$, say, is comparable with that of $\mathsf{A}\boldsymbol{x}$); the cost of computation should be determined by the number of nonzero entries, rather than by n. The only tool at our disposal at the moment is the freedom to exchange rows and columns to minimise *fill-in.* To this end the following theorem is useful.

Theorem 8.18. *Let* $\mathsf{A} = \mathsf{LU}$ *be the LU factorization (without pivoting) of a sparse matrix. Then*

(i) *all leading zeros in the rows of* A *to the left of diagonal are inherited by* L,

(ii) *all leading zeros in the columns of* A *above the diagonal are inherited by* U.

$$\begin{bmatrix} \ast & \bullet & \bullet & \bullet & & \bullet & \\ \circ & \ast & \bullet & \bullet & & \bullet & \\ \circ & \circ & \ast & \bullet & \bullet & \bullet & \\ & & \circ & \ast & \bullet & \bullet & \\ \circ & \circ & \circ & \circ & \ast & \bullet & \bullet \\ & & & & \circ & \ast & \bullet \\ \circ & \circ & \circ & \circ & \circ & \circ & \ast \end{bmatrix} = \begin{bmatrix} \ast & & & & & & \\ \circ & \ast & & & & & \\ \circ & \circ & \ast & & & & \\ & & \circ & \ast & & & \\ \circ & \circ & \circ & \circ & \ast & & \\ & & & & \circ & \ast & \\ \circ & \circ & \circ & \circ & \circ & \circ & \ast \end{bmatrix} \times \begin{bmatrix} \ast & \bullet & \bullet & \bullet & & \bullet & \\ & \ast & \bullet & \bullet & & \bullet & \\ & & \ast & \bullet & \bullet & \bullet & \\ & & & \ast & \bullet & \bullet & \\ & & & & \ast & \bullet & \bullet \\ & & & & & \ast & \bullet \\ & & & & & & \ast \end{bmatrix}$$

Proof. Let $A_{i,1} = A_{i,2} = \cdots = 0$ be the leading zeros in the i-th row. Then, if $U_{k,k} \neq 0$, we obtain

$$\begin{aligned} 0 = A_{i,1} &= L_{i,1}U_{1,1} && \Rightarrow \quad L_{i,1} = 0, \\ 0 = A_{i,2} &= L_{i,1}U_{1,2} + L_{i,2}U_{2,2} && \Rightarrow \quad L_{i,2} = 0, \\ 0 = A_{i,3} &= L_{i,1}U_{1,3} + L_{i,2}U_{2,3} + L_{i,3}U_{3,3} && \Rightarrow \quad L_{i,3} = 0, \quad \text{and so on.} \end{aligned}$$

If $U_{k,k} = 0$ for some k, we can choose $L_{i,k} = 0$. Similarly for the leading zeros in the j-th column. Since $L_{k,k} = 1$, it follows that

$$\begin{aligned} 0 = A_{1,j} &= L_{1,1}U_{1,j} && \Rightarrow \quad U_{1,j} = 0, \\ 0 = A_{2,j} &= L_{2,1}U_{1,j} + L_{2,2}U_{2,j} && \Rightarrow \quad U_{2,j} = 0, \\ 0 = A_{3,j} &= L_{3,1}U_{1,j} + L_{3,2}U_{2,j} + L_{3,3}U_{3,j} && \Rightarrow \quad U_{3,j} = 0, \quad \text{and so on.} \end{aligned}$$ □

13/13

Corollary 8.19. *If* 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* L *and* U *are band matrices with the same band width as* A.

Cost. In general, the expense of calculating an LU factorization of an $n \times n$ *dense* matrix 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 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 8.20. Theorem 8.18 suggests that for a factorization of a sparse but not nicely structured matrix A one might try to reorder its rows and columns by a preliminary calculation so that many of the zero elements become leading zero elements in rows and columns. This will reduce the fill-in in L and U.

Example 1. The 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

$$\mathsf{PAP} = \begin{bmatrix} 1 & & & & 1 \\ & 1 & & & 1 \\ & & 1 & & 1 \\ & & & 1 & 1 \\ 1 & 1 & 1 & 1 & 5 \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & 1 \\ & 1 & & & 1 \\ & & 1 & & 1 \\ & & & 1 & 1 \\ & & & & 1 \end{bmatrix}.$$

Example 2. If the non-zeros of 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 A, respectively. 13/10

Example 3. The 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

$$\begin{bmatrix} 2 & 0 & 1 & 0 & 0 \\ 0 & 3 & 2 & 0 & 0 \\ 1 & 2 & -3 & 0 & 1 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 1 & 1 & -3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{2}{3} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{6}{29} & \frac{1}{3} & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 1 & 0 & 0 \\ 0 & 3 & 2 & 0 & 0 \\ 0 & 0 & -\frac{29}{6} & 0 & 1 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & -\frac{272}{87} \end{bmatrix}.$$

13/12
13/14

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 `:-(`

13/11

8.4 QR factorization of matrices

8.4.1 Inner product spaces

We first recall some facts about inner product spaces.

- The axioms of an inner product on a linear vector space over the reals, say $\mathbb{V}$, were given in (2.2).
- A vector space with an inner product is called an *inner product* space.
- For $\boldsymbol{u} \in \mathbb{V}$, the function $\|\boldsymbol{u}\| = \langle \boldsymbol{u}, \boldsymbol{u}\rangle^{1/2}$ is the *norm* of $\boldsymbol{u}$ (induced by the given inner product), and we have the Cauchy–Schwarz inequality

$$\langle \boldsymbol{u}, \boldsymbol{v}\rangle \leqslant \|\boldsymbol{u}\|\,\|\boldsymbol{v}\| \quad \forall\, \boldsymbol{u}, \boldsymbol{v} \in \mathbb{V}. \tag{8.25a}$$

- The vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{V}$ are said to be *orthogonal* if $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = 0$.
- A set of vectors $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m \in \mathbb{V}$ is said to be *orthonormal* if

$$\langle \boldsymbol{q}_i, \boldsymbol{q}_j\rangle = \delta_{ij} = \begin{cases} 1, & i = j; \\ 0, & i \neq j\,. \end{cases} \tag{8.25b}$$

Example. For $\mathbb{V} = \mathbb{R}^n$, we define the so-called Euclidean inner product for all $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ by

$$\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \langle \boldsymbol{v}, \boldsymbol{u}\rangle = \sum_{j=1}^{n} u_j v_j = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{v} = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{u}\,, \tag{8.26a}$$

in which case the *norm* (a.k.a. the *Euclidean length*) of $\boldsymbol{u} \in \mathbb{R}^n$ is

$$\|\boldsymbol{u}\| = \langle \boldsymbol{u}, \boldsymbol{u}\rangle^{1/2} = \left(\sum_{j=1}^{n} u_j^2\right)^{1/2} \geqslant 0. \tag{8.26b}$$

8.4.2 Properties of orthogonal matrices

A $n \times n$ real matrix Q is *orthogonal* if all its columns are orthonormal. Since $(\mathsf{Q}^{\mathrm{T}}\mathsf{Q})_{i,j} = \langle \boldsymbol{q}_i, \boldsymbol{q}_j \rangle$, it follows that

$$\mathsf{Q}^{\mathrm{T}}\mathsf{Q} = \mathsf{I} \tag{8.27a}$$

and hence

$$\mathsf{Q}^{-1} = \mathsf{Q}^{\mathrm{T}} \quad \text{and} \quad \mathsf{Q}\mathsf{Q}^{\mathrm{T}} = \mathsf{Q}\mathsf{Q}^{-1} = \mathsf{I}. \tag{8.27b}$$

It also follows that the rows of an orthogonal matrix are orthonormal, and that Q^{T} is an orthogonal matrix. It further follows from

$$1 = \det \mathsf{I} = \det(\mathsf{Q}\mathsf{Q}^{\mathrm{T}}) = \det \mathsf{Q} \det \mathsf{Q}^{\mathrm{T}} = (\det \mathsf{Q})^2,$$

that $\det \mathsf{Q} = \pm 1$, and hence that an orthogonal matrix is non-singular.

Proposition 8.21. *If* P, Q *are orthogonal then so is* $\mathsf{P}\mathsf{Q}$.

Proof. Since $\mathsf{P}^{\mathrm{T}}\mathsf{P} = \mathsf{Q}^{\mathrm{T}}\mathsf{Q} = \mathsf{I}$, we have

$$(\mathsf{P}\mathsf{Q})^{\mathrm{T}}(\mathsf{P}\mathsf{Q}) = (\mathsf{Q}^{\mathrm{T}}\mathsf{P}^{\mathrm{T}})(\mathsf{P}\mathsf{Q}) = \mathsf{Q}^{\mathrm{T}}(\mathsf{P}^{\mathrm{T}}\mathsf{P})\mathsf{Q} = \mathsf{Q}^{\mathrm{T}}\mathsf{Q} = \mathsf{I}.$$

Hence $\mathsf{P}\mathsf{Q}$ is orthogonal. □

Proposition 8.22. *Let* $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m \in \mathbb{R}^n$ *be orthonormal. Then* $m \leqslant n$.

Proof. We argue by contradiction. Suppose that $m \geqslant (n+1)$, and let Q be the orthogonal matrix whose columns are $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n$. Since Q is non-singular and $\boldsymbol{q}_m \neq \mathbf{0}$, there exists a nonzero solution, $\boldsymbol{a} \neq 0$, to the linear system

$$\mathsf{Q}\boldsymbol{a} = \boldsymbol{q}_m,$$

i.e. there exists $\boldsymbol{a} \neq 0$ such that

$$\sum_{j=1}^{n} a_j \boldsymbol{q}_j = \boldsymbol{q}_m \,.$$

Hence

$$0 = \langle \boldsymbol{q}_i, \boldsymbol{q}_m \rangle = \left\langle \boldsymbol{q}_i, \sum_{j=1}^{n} a_j \boldsymbol{q}_j \right\rangle = \sum_{j=1}^{n} a_j \langle \boldsymbol{q}_i, \boldsymbol{q}_j \rangle = a_i, \qquad i = 1, 2, \ldots, n,$$

and we have the contradictory result that $\boldsymbol{a} = \mathbf{0}$. We deduce that $m \leqslant n$. □

Lemma 8.23. *Let* $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m \in \mathbb{R}^n$ *be orthonormal and* $m \leqslant n-1$. *Then there exists* $\boldsymbol{q}_{m+1} \in \mathbb{R}^n$ *such that* $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{m+1}$ *are orthonormal.*

Proof. We construct $\boldsymbol{q}_{m+1}$. Let Q be the $n \times m$ matrix whose columns are $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m$. Since

$$\sum_{k=1}^{n} \left(\sum_{j=1}^{m} \mathsf{Q}_{k,j}^2 \right) = \sum_{j=1}^{m} \|\boldsymbol{q}_j\|^2 = m < n,$$

it follows that $\exists\ i \in \{1, 2, \ldots, n\}$ such that $\sum_{j=1}^{m} \mathsf{Q}_{i,j}^2 < 1$. We let

$$\boldsymbol{w} = \boldsymbol{e}_i - \sum_{j=1}^{m} \langle \boldsymbol{q}_j, \boldsymbol{e}_i \rangle \boldsymbol{q}_j,$$

then for $\ell = 1, 2, \ldots, m$

$$\langle \boldsymbol{q}_\ell, \boldsymbol{w} \rangle = \langle \boldsymbol{q}_\ell, \boldsymbol{e}_i \rangle - \sum_{j=1}^{m} \langle \boldsymbol{q}_j, \boldsymbol{e}_i \rangle \langle \boldsymbol{q}_\ell, \boldsymbol{q}_j \rangle = 0.$$

Hence, by design, $\boldsymbol{w}$ is orthogonal to $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m$. Further, since $\mathsf{Q}_{i,j} = \langle \boldsymbol{q}_j, \boldsymbol{e}_i \rangle$, we have

$$\|\boldsymbol{w}\|^2 = \langle \boldsymbol{w}, \boldsymbol{w} \rangle = \langle \boldsymbol{e}_i, \boldsymbol{e}_i \rangle - 2 \sum_{j=1}^{m} \langle \boldsymbol{q}_j, \boldsymbol{e}_i \rangle \langle \boldsymbol{e}_i, \boldsymbol{q}_j \rangle + \sum_{j=1}^{m} \langle \boldsymbol{q}_j, \boldsymbol{e}_i \rangle \sum_{k=1}^{m} \langle \boldsymbol{q}_k, \boldsymbol{e}_i \rangle \langle \boldsymbol{q}_j, \boldsymbol{q}_k \rangle = 1 - \sum_{j=1}^{m} \mathsf{Q}_{i,j}^2 > 0.$$

Thus we define $\boldsymbol{q}_{m+1} = \boldsymbol{w}/\|\boldsymbol{w}\|$. □

8.4.3 The QR factorization

Definition 8.24. The QR *factorization* of an $m \times n$ matrix A is the representation

$$\mathsf{A} = \mathsf{QR}, \tag{8.28a}$$

i.e.

$$\underbrace{\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ A_{n,1} & \cdots & A_{n,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ A_{m,1} & \cdots & A_{m,n} \end{bmatrix}}_{n} = \underbrace{\begin{bmatrix} Q_{1,1} & \cdots & Q_{1,n} & \cdots & Q_{1,m} \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ Q_{n,1} & & Q_{n,n} & & Q_{n,m} \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ Q_{m,1} & \cdots & Q_{m,n} & \cdots & Q_{m,m} \end{bmatrix}}_{m\geqslant n} \underbrace{\begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ & R_{2,2} & & \vdots \\ & & \ddots & \vdots \\ & & & R_{n,n} \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}}_{n} \left.\vphantom{\begin{bmatrix} 1\\1\\1\\1\\1\\1\\1 \end{bmatrix}}\right\} m\geqslant n\,, \tag{8.28b}$$

where Q is an $m \times m$ *orthogonal* matrix and R is an $m \times n$ *upper triangular* matrix (i.e. $R_{i,j} = 0$ for $i > j$).

Remarks.

(i) We shall see that every matrix has a (non-unique) QR factorization.

(ii) For clarity of presentation, and unless stated otherwise, we will assume that $m \geqslant n$.

Interpretation of the QR *factorization.* Let $m \geqslant n$ and denote the columns of A and Q by $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$ and $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m$ respectively. Since

$$\begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_m \end{bmatrix} \begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ 0 & R_{2,2} & & \vdots \\ \vdots & \ddots & \ddots & \\ & & 0 & R_{n,n} \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix},$$

where the bottom rows of zeros are absent if $m = n$, we have that

$$\boldsymbol{a}_j = \sum_{i=1}^{m} R_{i,j}\boldsymbol{q}_i = \sum_{i=1}^{j} R_{i,j}\boldsymbol{q}_i, \quad j = 1, 2, \ldots, n. \tag{8.29}$$

In other words, Q has the property that the j^{th} column of A can be expressed as a linear combination of the first j columns of Q.

Definition 8.25. If $m > n$, then because of the bottom zero elements of R, the columns $\boldsymbol{q}_{n+1}, \ldots, \boldsymbol{q}_m$ are not essential for the representation, and hence we can write

$$m\geqslant n \left\{\vphantom{\begin{bmatrix} 1\\1\\1\\1\\1\\1 \end{bmatrix}}\right. \underbrace{\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ A_{n,1} & \cdots & A_{n,n} \\ \vdots & & \vdots \\ A_{m,1} & \cdots & A_{m,n} \end{bmatrix}}_{n} = \underbrace{\begin{bmatrix} Q_{1,1} & \cdots & Q_{1,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ Q_{n,1} & \cdots & Q_{n,n} \\ \vdots & & \vdots \\ Q_{m,1} & \cdots & Q_{m,n} \end{bmatrix}}_{n\leqslant m} \underbrace{\begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ & R_{2,2} & & \vdots \\ & & \ddots & \vdots \\ & & & R_{n,n} \end{bmatrix}}_{n} \left.\vphantom{\begin{bmatrix} 1\\1\\1\\1 \end{bmatrix}}\right\} n\,. \tag{8.30}$$

The latter formula is called the *skinny* QR factorization.

Definition 8.26. We say that a matrix, say R, is in a *standard form* if it has the property that the number of leading zeros in each row increases *strictly monotonically* until all the rows of R are zero.

Remarks

(i) A matrix in a standard form is allowed entire rows of zeros, but only at the bottom.

(ii) If R_{i,j_i} is the first nonzero entry in the i^{th} row, the j_is form a strictly monotone sequence.

Theorem 8.27. *Every matrix* A *has a* QR *factorization. If* A *is of full rank (i.e.,* $\mathsf{A}\boldsymbol{x} \neq \mathbf{0}$ *if* $\boldsymbol{x} \neq \mathbf{0}$*), then there exists a unique skinny factorization* $\mathsf{A} = \mathsf{QR}$ *with* R *having a positive main diagonal.*

Proof. Given a matrix A we shall see that there are three algorithms by which a QR factorization can be constructed, namely:

(i) Gram–Schmidt orthogonalization (for the 'skinny' version);

(ii) Givens rotations;

(iii) Householder reflections.

Let A be of full rank, then the $n \times n$ *Gram* matrix, $\mathsf{A}^{\mathrm{T}}\mathsf{A}$, is symmetric positive definite since

$$\boldsymbol{x}^{\mathrm{T}}\mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} > 0 \quad \text{for any } \boldsymbol{x}\neq 0, \text{ since } \boldsymbol{y} = \mathsf{A}\boldsymbol{x} \neq \mathbf{0}.$$

Hence, from Theorem 8.14 and (8.24), there is a unique Cholesky factorization $\mathsf{A}^{\mathrm{T}}\mathsf{A} = \mathsf{G}\mathsf{G}^{\mathrm{T}}$ with G having a positive main diagonal. On the other hand, if $\mathsf{A} = \mathsf{QR}$, then $\mathsf{A}^{\mathrm{T}}\mathsf{A} = \mathsf{R}^{\mathrm{T}}\mathsf{Q}^{\mathrm{T}}\mathsf{QR}$, so an upper triangular R with $R_{i,i} > 0$ coincides with G^{T}, and $\mathsf{Q} = \mathsf{A}\mathsf{R}^{-1}$. □

8.4.4 Applications of QR factorization

Application: solution of linear systems when $m = n$. If A is square non-singular, we can solve $\mathsf{A}\boldsymbol{x} = b$ by calculating the QR factorization of A and then proceeding in two steps

$$\mathsf{A}\boldsymbol{x} = \mathsf{Q}\,\underbrace{\mathsf{R}\boldsymbol{x}}_{\boldsymbol{y}} = \boldsymbol{b}\,. \tag{8.31}$$

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.

Remark. The work of calculating the QR factorization makes this method more expensive than the LU procedure for solving $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$.

Application: least-squares solution of overdetermined equations. If A is $m \times n$ and $m > n$, then usually there is no solution of

$$\mathsf{A}\boldsymbol{x} = \boldsymbol{b}.$$

In this case, one sometimes requires the vector $\boldsymbol{x}$ that minimizes the norm $\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|$. In §9 we will see that the QR factorization is suitable for determining this $\boldsymbol{x}$.

Finding eigenvalues and eigenvectors. See next year.

14/10

8.4.5 The Gram–Schmidt process

First recall, from *Vectors & Matrices*, that given a finite, linearly independent set of vectors $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_r\}$ the Gram–Schmidt process is a method of generating an orthogonal set $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r\}$. This is done, at stage k, by projecting $\boldsymbol{w}_k$ orthogonally onto the subspace generated by $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k-1}\}$, and then the vector $\boldsymbol{v}_k$ is defined to be the difference between $\boldsymbol{w}_k$ and this projection.

Given an $m \times n$ non-zero matrix A with the columns $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n \in \mathbb{R}^m$, we want to use this process to construct the columns of the orthogonal matrix Q and the upper triangular matrix R such that $\mathsf{A} = \mathsf{QR}$, i.e. from (8.29) such that

$$\sum_{i=1}^{j} R_{i,j}\boldsymbol{q}_i = \boldsymbol{a}_j, \quad j = 1, 2, \ldots, n, \qquad \text{where} \qquad \mathsf{A} = [\ \boldsymbol{a}_1 \quad \boldsymbol{a}_2 \quad \cdots \quad \boldsymbol{a}_n\ ]. \tag{8.32}$$

In order to understand how the method works, let us start by supposing that $\boldsymbol{a}_1 \neq \boldsymbol{0}$. Then we derive $\boldsymbol{q}_1$ and $R_{1,1}$ from the equation (8.32) with $j = 1$: $R_{1,1}\boldsymbol{q}_1 = \boldsymbol{a}_1$. Since we require $\|\boldsymbol{q}_1\| = 1$, we let $\boldsymbol{q}_1 = \boldsymbol{a}_1/\|\boldsymbol{a}_1\|$, $R_{1,1} = \|\boldsymbol{a}_1\|$.

Next we form the vector $\boldsymbol{b} = \boldsymbol{a}_2 - \langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle \boldsymbol{q}_1$. By construction $\boldsymbol{b}$ is orthogonal to $\boldsymbol{q}_1$ since

$$\langle \boldsymbol{q}_1, \boldsymbol{b}\rangle = \langle \boldsymbol{q}_1, \boldsymbol{a}_2 - \langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle \boldsymbol{q}_1\rangle = \langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle - \langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle\langle \boldsymbol{q}_1, \boldsymbol{q}_1\rangle = 0.$$

If $\boldsymbol{b} \neq \boldsymbol{0}$, we set $\boldsymbol{q}_2 = \boldsymbol{b}/\|\boldsymbol{b}\|$; $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are then orthonormal. Moreover,

$$\langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle \boldsymbol{q}_1 + \|\boldsymbol{b}\|\boldsymbol{q}_2 = \langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle \boldsymbol{q}_1 + \boldsymbol{b} = \boldsymbol{a}_2,$$

hence, to obey (8.32) for $j = 2$, i.e. to obey $R_{1,2}\boldsymbol{q}_1 + R_{2,2}\boldsymbol{q}_2 = \boldsymbol{a}_2$, we let $R_{1,2} = \langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle$, $R_{2,2} = \|\boldsymbol{b}\|$.

If $\boldsymbol{b} = \boldsymbol{0}$, then $\langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle \boldsymbol{q}_1 = \boldsymbol{a}_2$, and thus $\boldsymbol{a}_2$ is in the vector space spanned by $\boldsymbol{q}_1$. Hence there is no need to add another column to Q in order to span the first 2 columns of A. From comparison with (8.32), we therefore set $R_{1,2} = \langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle$ and $R_{2,2} = \|\boldsymbol{b}\| = 0$.

The Gram–Schmidt algorithm. The above idea can be extended to all columns of A. The only slight difficulty being keeping track of which columns Q have been formed.

Step 1. Set $j = 0$, $k = 0$ where we use j to keep track of the number of columns of A and R that have been already considered, and k to keep track of the number of columns of Q that have been formed ($k \leqslant j$).

Step 2. Increase j by 1.

- If $k = 0$ set $\boldsymbol{b} = \boldsymbol{a}_j$.
- If $k \geqslant 1$ set $\boldsymbol{b} = \boldsymbol{a}_j - \sum_{i=1}^{k}\langle \boldsymbol{q}_i, \boldsymbol{a}_j\rangle \boldsymbol{q}_i$ and $R_{i,j} = \langle \boldsymbol{q}_i, \boldsymbol{a}_j\rangle$, $i = 1, 2, \ldots, k$. Note that by construction

$$\boldsymbol{b} \quad \text{is orthogonal to} \quad \boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_k\,, \quad \text{and} \quad \sum_{i=1}^{k} R_{i,j}\boldsymbol{q}_i + \boldsymbol{b} = \boldsymbol{a}_j\,. \tag{8.33a}$$

Step 3.

- If $\boldsymbol{b} \neq \boldsymbol{0}$, set $\boldsymbol{q}_{k+1} = \boldsymbol{b}/\|\boldsymbol{b}\|$, $R_{k+1,j} = \|\boldsymbol{b}\|$ and $R_{i,j} = 0$ for $i \geqslant k + 2$. With these definitions, the first $k + 1$ columns of Q are orthonormal. Further, R is upper triangular since $j \geqslant k + 1$, and from (8.33a)

$$\sum_{i=1}^{j} R_{i,j}\boldsymbol{q}_i = \sum_{i=1}^{k+1} R_{i,j}\boldsymbol{q}_i = \sum_{i=1}^{k} R_{i,j}\boldsymbol{q}_i + \|\boldsymbol{b}\|\frac{\boldsymbol{b}}{\|\boldsymbol{b}\|} = \boldsymbol{a}_j\,. \tag{8.33b}$$

  Increase k by 1.

- If $\boldsymbol{b} = \boldsymbol{0}$, then from (8.33a), $\boldsymbol{a}_j$ is in the vector space spanned by $\boldsymbol{q}_i$, $i = 1, \ldots, k$. Set $R_{i,j} = 0$ for $i \geqslant k + 1$. R is upper triangular since $j \geqslant k + 1$, and from (8.33a)

$$\sum_{i=1}^{j} R_{i,j}\boldsymbol{q}_i = \sum_{i=1}^{k} R_{i,j}\boldsymbol{q}_i = \boldsymbol{a}_j\,. \tag{8.33c}$$

Step 4. Terminate if $j = n$, otherwise go to *Step 2*.

14/11

- From Proposition (8.22) we have that, since the columns of Q are orthonormal, there are at most m of them, i.e. the final value of k cannot exceed m.
- If the final k is less then m, then Lemma (8.23) demonstrates that we can add columns so that Q becomes $m \times m$ and orthogonal.
- Note also that if $n > m$, there must be stages in the algorithm when $\boldsymbol{b} = \boldsymbol{0}$.

Example. Let us find the QR factorization by Gram–Schmidt of

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

From above

$$\begin{aligned}
&R_{1,1} = \|\boldsymbol{a}_1\| = 3, \quad \boldsymbol{q}_1 = \boldsymbol{a}_1/R_{1,1} = \tfrac{1}{3}\begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix};\\
&R_{1,2} = \langle \boldsymbol{q}_1, \boldsymbol{a}_2\rangle = 3, \qquad \boldsymbol{b}_2 = \boldsymbol{a}_2 - R_{1,2}\boldsymbol{q}_1 = \begin{bmatrix} 4 \\ -1 \\ 1 \end{bmatrix} - 3\cdot\tfrac{1}{3}\begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ -1 \end{bmatrix},\\
&R_{2,2} = \|\boldsymbol{b}_2\| = 3, \quad \boldsymbol{q}_2 = \boldsymbol{b}_2/R_{2,2} = \tfrac{1}{3}\begin{bmatrix} 2 \\ -2 \\ -1 \end{bmatrix};\\
&R_{1,3} = \langle \boldsymbol{q}_1, \boldsymbol{a}_3\rangle = 3, \quad R_{2,3} = \langle \boldsymbol{q}_2, \boldsymbol{a}_3\rangle = 3, \qquad \boldsymbol{b}_3 = \boldsymbol{a}_3 - R_{1,3}\boldsymbol{q}_1 - R_{2,3}\boldsymbol{q}_2\\
&\qquad = \begin{bmatrix} 5 \\ 1 \\ -1 \end{bmatrix} - 3\cdot\tfrac{1}{3}\begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - 3\cdot\tfrac{1}{3}\begin{bmatrix} 2 \\ -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix},\\
&R_{3,3} = \|\boldsymbol{b}_3\| = 3, \quad \boldsymbol{q}_3 = \boldsymbol{b}_3/R_{3,3} = \tfrac{1}{3}\begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}.
\end{aligned}$$

So,

$$\mathsf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \underbrace{\frac{1}{3}\begin{bmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\begin{bmatrix} 3 & 3 & 3 \\ & 3 & 3 \\ & & 3 \end{bmatrix}}_{\mathsf{R}}$$

Modified Gram–Schmidt. The disadvantage of the Gram–Schmidt algorithm is that its numerical accuracy is poor if much cancellation occurs when the vector $\boldsymbol{b}$ is formed. This is likely when using finite precision arithmetic to calculate the inner product. The result is that the new column of Q can be a multiple of a vector that is composed mainly of rounding errors, causing the off-diagonal elements of $\mathsf{Q}^{\mathrm{T}}\mathsf{Q}$ to be very different from zero. Indeed, errors can accumulate very fast with the result that even for moderate values of m problems are likely. The Gram–Schmidt algorithm is said to be *sensitive.*[13] However, the Gram–Schmidt process can be stabilized by a small modification, and this version is sometimes referred to as modified Gram–Schmidt or MGS.

If we assume throughout that $\boldsymbol{b} \neq \boldsymbol{0}$, then the classic Gram–Schmidt (CGS) procedure as described above can be written in MATLAB as (where we note that at the j^{th} stage the j^{th} columns of both Q and R are determined):

```
r=zeros(n);
b=a(:,1);
r(1,1)=norm(b);
q(:,1)=b/norm(b);
for j = 2:n
  b=a(:,j);
  for i=1:j-1
    r(i,j)=a(:,j)'*q(:,i);
    b=b-r(i,j)*q(:,i);
  end
  r(j,j)=norm(b);
  q(:,j)=b/norm(b);
end
```

[13] Problems are said to be *well-conditioned* or *ill-conditioned*, while algorithms are said to be *good/stable/insensitive* or *bad/unstable/sensitive*.

The key point in order to stabilise the scheme is to ensure that the elements of R are constructed from partially orthogonalised vectors, and *not* from the original columns of A. The constructed vectors are then orthogonalized against any errors introduced in the computation so far. In MGS this is done by, *de facto*, swapping the order of the `for` loops, so that at the i^{th} stage the i^{th} column of Q and the i^{th} row of R are determined.

```
q=a;
r=zeros(n);
for i=1:n-1
  r(i,i)=norm(q(:,i));
  q(:,i)=q(:,i)/norm(q(:,i));
  for j=i+1:n
    r(i,j)=q(:,j)'*q(:,i);
    q(:,j)=q(:,j)-r(i,j)*q(:,i);
  end
end
r(n,n)=norm(q(:,n));
q(:,n)=q(:,n)/norm(q(:,n));
```

14/13
14/14

Remark. There are alternatives to MGS, e.g. it is possible to show that orthogonality conditions are preserved well when a new orthogonal matrix is generated by computing the product of two given orthogonal matrices. Therefore algorithms that express Q as a product of simple orthogonal matrices are highly useful.

14/12

8.4.6 Orthogonal transformations

Our aim is to express Q as a product of simple orthogonal matrices. To this end, given a real, $m \times n$ matrix A, we consider $\mathsf{A} = \mathsf{QR}$ in the form $\mathsf{Q}^{\mathrm{T}}\mathsf{A} = \mathsf{R}$. We then seek a sequence of simple $m \times m$ orthogonal matrices $\Omega_1, \Omega_2, \dots, \Omega_k$ such that

$$\mathsf{A}_j = \Omega_j \mathsf{A}_{j-1}\,, \quad j = 1, 2, \dots, k \tag{8.34a}$$

where $\mathsf{A}_0 = \mathsf{A}$, A_k is upper triangular and the Ω_j are chosen so that the matrix A_j has *more* zero elements below the main diagonal than A_{j-1} (this condition ensures that the value of k is finite). Then, as required,

$$\mathsf{R} = \mathsf{A}_k = \Omega_k \cdots \Omega_2 \Omega_1 \mathsf{A} = \mathsf{Q}^{\mathrm{T}}\mathsf{A}\,, \tag{8.34b}$$

where

$$\mathsf{Q} = (\Omega_k \Omega_{k-1} \cdots \Omega_1)^{-1} = \Omega_1^{\mathrm{T}} \Omega_2^{\mathrm{T}} \cdots \Omega_k^{\mathrm{T}}\,, \tag{8.34c}$$

is orthogonal since the product of orthogonal matrices is orthogonal, as is the transpose of an orthogonal matrix.

We recall from *Vectors & Matrices* that in Euclidean space an orthogonal matrix Q represents a *rotation* or a *reflection* according as $\det \mathsf{Q} = 1$ or $\det \mathsf{Q} = -1$. We will describe two 'geometric' methods, based on a sequence of elementary rotations and reflections, that ensure that all zeros of A_{j-1} that are in 'suitable' positions are inherited by A_j, where the meaning of 'suitable' will become clear. We begin with the 'Givens algorithm'.

8.4.7 Givens rotations

We say that an $m \times m$ orthogonal matrix Ω_j is a *Givens rotation* if it coincides with the unit matrix, except for four elements, and $\det \Omega_j = 1$. Specifically, we use the notation $\Omega^{[p,q]}$, where $1 \leqslant p < q \leqslant m$ for a matrix such that

$$\Omega^{[p,q]}_{p,p} = \Omega^{[p,q]}_{q,q} = \cos\theta, \qquad \Omega^{[p,q]}_{p,q} = \sin\theta, \qquad \Omega^{[p,q]}_{q,p} = -\sin\theta$$

for some $\theta \in [-\pi, \pi]$. The remaining elements of $\Omega^{[p,q]}$ are those of a unit matrix. For example,

$$m = 4 \quad \Longrightarrow \quad \Omega^{[1,2]} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \Omega^{[2,4]} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & 0 & \sin\theta \\ 0 & 0 & 1 & 0 \\ 0 & -\sin\theta & 0 & \cos\theta \end{bmatrix}.$$

Geometrically, the mapping $\Omega^{[p,q]}$ rotates vectors in the two-dimensional plane spanned by $\boldsymbol{e}_p$ and $\boldsymbol{e}_q$ (clockwise by the angle θ), hence it is orthogonal. In mechanics this is called an *Euler rotation*.

Remark. It is sometimes helpful to express the interesting part of $\Omega^{(p,q)}$ in the form

$$\begin{pmatrix} \Omega^{(p,q)}_{p,p} & \Omega^{(p,q)}_{p,q} \\ \Omega^{(p,q)}_{q,p} & \Omega^{(p,q)}_{q,q} \end{pmatrix} = \frac{1}{\sqrt{\alpha^2+\beta^2}} \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix}, \tag{8.35}$$

where α and β are any real numbers that are not both zero.

Theorem 8.28. *Let* A *be an* $m \times n$ *matrix. Then, for every* $1 \leqslant p < q \leqslant m$*, there exists* $\theta \in [-\pi, \pi]$ *such that*

(i) $(\Omega^{[p,q]}\mathsf{A})_{i,j} = 0$*, where* $i \in \{p,q\}$*, and* $1 \leqslant j \leqslant n$*;*

(ii) *all the rows of* $\Omega^{[p,q]}\mathsf{A}$*, except for the* pth *and the* qth*, are the same as the corresponding rows of* A*;*

(iii) *the* pth *and the* qth *rows are linear combinations of the 'old'* pth *and* qth *rows.*

Proof. First suppose that $i = q$, and recall that

$$(\Omega^{[p,q]}\mathsf{A})_{r,s} = \sum_{t=1}^{m} \Omega^{[p,q]}_{r,t} A_{t,s} \quad \text{for} \quad r = 1,\ldots,m \quad \text{and} \quad s = 1,\ldots,n\,.$$

$r \neq p,q$. When $r \neq p,q$, then $\Omega^{[p,q]}_{r,t} = \delta_{rt}$, and hence $(\Omega^{[p,q]}\mathsf{A})_{r,s} = A_{r,s}$ as required.

$r = p$. The pth row is a linear combinations of the 'old' pth and qth rows since

$$(\Omega^{[p,q]}\mathsf{A})_{p,s} = (\cos\theta)A_{p,s} + (\sin\theta)A_{q,s}, \quad s = 1,2,\ldots,n.$$

$r = q$. Similarly, the qth row is a linear combinations of the 'old' pth and qth rows since

$$(\Omega^{[p,q]}\mathsf{A})_{q,s} = -(\sin\theta)A_{p,s} + (\cos\theta)A_{q,s}, \quad s = 1,2,\ldots,n.$$

Hence, if $A_{p,j} = A_{q,j} = 0$ then $(\Omega^{[p,q]}A)_{q,j} = 0$ for any θ, otherwise let

$$\cos\theta = A_{p,j}\Big/\sqrt{A^2_{p,j} + A^2_{q,j}}, \qquad \sin\theta = A_{q,j}\Big/\sqrt{A^2_{p,j} + A^2_{q,j}}.$$

so that

$$(\Omega^{[p,q]}\mathsf{A})_{q,j} = -\frac{A_{q,j}}{\left(A^2_{p,j} + A^2_{q,j}\right)^{1/2}}\,A_{p,j} + \frac{A_{p,j}}{\left(A^2_{p,j} + A^2_{q,j}\right)^{1/2}}\,A_{q,j} = 0\,.$$

If instead $i = p$, we let $\cos\theta = A_{q,j}\Big/\sqrt{A^2_{p,j} + A^2_{q,j}}$ and $\sin\theta = -A_{p,j}\Big/\sqrt{A^2_{p,j} + A^2_{q,j}}$. □

An example. Suppose that A is 3×3. We can force zeros underneath the main diagonal as follows.

(i) First pick $\Omega^{[1,2]}$ so that $(\Omega^{[1,2]}\mathsf{A})_{2,1} = 0$, i.e. so that

$$\Omega^{[1,2]}\mathsf{A} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ \times & \times & \times \end{bmatrix}.$$

(ii) Next pick $\Omega^{[1,3]}$ so that $(\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A})_{3,1} = 0$. Multiplication by $\Omega^{[1,3]}$ doesn't alter the second row, hence $(\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A})_{2,1}$ remains zero, and hence

$$\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}.$$

(iii) Finally, pick $\Omega^{[2,3]}$ so that $(\Omega^{[2,3]}\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A})_{3,2} = 0$. Since both second and third row of $\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A}$ have a leading zero, their linear combination preserves these zeros, hence

$$(\Omega^{[2,3]}\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A})_{2,1} = (\Omega^{[2,3]}\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A})_{3,1} = 0.$$

It follows that $\Omega^{[2,3]}\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A}$ is upper triangular. Therefore

$$\mathsf{R} = \Omega^{[2,3]}\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}, \quad \text{and} \quad \mathsf{Q} = (\Omega^{[2,3]}\Omega^{[1,3]}\Omega^{[1,2]})^{\mathrm{T}}.$$

8.4.8 The Givens algorithm for calculating the QR factorisation of A

Theorem 8.29. *For any matrix* A*, there exist Givens matrices* $\Omega^{[p,q]}$ *such that*

$$\mathsf{R} = \left(\Omega^{[m-1,m]}\right)\cdots\left(\Omega^{[2,m]}\cdots\Omega^{[2,3]}\right)\left(\Omega^{[1,m]}\cdots\Omega^{[1,3]}\Omega^{[1,2]}\right)\mathsf{A}$$

is an upper triangular matrix.

Proof. In pictures:

$$\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \xrightarrow{\Omega^{[1,2]}} \begin{bmatrix} \bullet&\bullet&\bullet \\ 0&\bullet&\bullet \\ *&*&* \\ *&*&* \end{bmatrix} \xrightarrow{\Omega^{[1,3]}} \begin{bmatrix} \bullet&\bullet&\bullet \\ 0&*&* \\ 0&\bullet&\bullet \\ *&*&* \end{bmatrix} \xrightarrow{\Omega^{[1,4]}} \begin{bmatrix} \bullet&\bullet&\bullet\\0&*&*\\0&*&*\\0&\bullet&\bullet\end{bmatrix} \xrightarrow{\Omega^{[2,3]}} \begin{bmatrix} *&*&*\\0&\bullet&\bullet\\0&0&\bullet\\0&*&*\end{bmatrix} \xrightarrow{\Omega^{[2,4]}} \begin{bmatrix}*&*&*\\0&\bullet&\bullet\\0&0&*\\0&0&\bullet\end{bmatrix} \xrightarrow{\Omega^{[3,4]}} \begin{bmatrix}*&*&*\\0&*&*\\0&0&\bullet\\0&0&0\end{bmatrix},$$

where the $\bullet$-elements have changed through a single rotation while the $*$-elements have not.

Alternatively (in words, and introducing some flexibility in the choice of the $\Omega^{[p,q]}$), given an $m\times n$ matrix A, let l_i be the number of leading zeros in the i^{th} row of A, $i = 1, 2, \ldots, m$.

Step 1. Stop if the (integer) sequence $\{l_1, l_2, \ldots, l_m\}$ increases monotonically, the increase being strictly monotone for $l_i \leqslant n$.

Step 2. Pick any two integers $1 \leqslant p < q \leqslant m$ such that either $l_p > l_q$ or $l_p = l_q < n$; note that $A_{q,l_q+1} \neq 0$ from the definition of l_q.

Step 3. Replace A by $\Omega^{[p,q]}\mathsf{A}$, using the Givens rotation that annihilates the $(q, l_q + 1)$ element. Update the values of l_p and l_q and go to *Step 1.*

The final matrix A is upper triangular and also has the property that the number of leading zeros in each row increases *strictly monotonically* until all the rows of A are zero. The matrix is thus in *standard form*, and is the required matrix R. □

The choice of p and q. It is convenient to make the *Step 2* choices of p and q in the following way. Let $r-1$ be the number of rows of R that are complete already, r being set to 1 initially. If $r\geqslant 2$, then the sequence $\{l_i : i=1,2,\ldots,r-1\}$ increases strictly monotonically.

Let $\mathcal{L}_r = \min\{l_i : i=r, r+1, \ldots, m\}$. The Givens rotations that determine the r-th row of R have $p=r$ and $q \in \{i : l_i = \mathcal{L}_r\}$ in *Steps 2 & 3*, until *Step 3* provides $l_i > \mathcal{L}_r$ for every i in $[r+1, m]$. Then r is increased by one. This choice of p and q ensures that $\mathcal{L}_r > l_{r-1}$ while constructing the r-th row of R.

Remark. This sequence of Givens rotations is not the only one that leads to a QR factorization, e.g. a sequence consisting only of rotations $\Omega^{[q-1,q]}$ will also do the job.

The cost.

(i) 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 R is $\mathcal{O}(mn^2)$.

(ii) If the matrix Q is required in an explicit form, say, for solving the system with many right-hand sides, set $\Omega = \mathsf{I}$ and, for each successive rotation, replace Ω by $\Omega^{[p,q]}\Omega$. The final Ω is the product of all the rotations, in correct order, and we let $\mathsf{Q} = \Omega^{\mathrm{T}}$. The extra cost is $\mathcal{O}(m^2 n)$.

(iii) If only one vector $\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}$ is required (e.g. in the case of the solution of a single linear system), we multiply the vector by successive rotations, the cost being $\mathcal{O}(mn)$.

(iv) For $m = n$, each rotation requires four times more multiplications as the corresponding Gaussian elimination, hence the total cost is $\frac{4}{3}n^3 + \mathcal{O}(n^2)$, four times as expensive. However, the QR factorization is generally more stable than the LU one.

Example. Find the QR factorization by Givens rotations of

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

From Theorem 8.28 (with initially $p = 1$, $q = 2$, $i = 2$, $j = 1$, $\cos\theta = \frac{2}{\sqrt{5}}$ and $\sin\theta = \frac{1}{\sqrt{5}}$),

$$\Omega^{[1,2]}\mathsf{A} = \underbrace{\begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & 0 \\ -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\Omega^{[1,2]}} \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} \sqrt{5} & \frac{7}{\sqrt{5}} & \frac{11}{\sqrt{5}} \\ 0 & -\frac{6}{\sqrt{5}} & -\frac{3}{\sqrt{5}} \\ 2 & 1 & -1 \end{bmatrix},$$

$$\Omega^{[1,3]}(\Omega^{[1,2]}\mathsf{A}) = \underbrace{\begin{bmatrix} \frac{\sqrt{5}}{3} & 0 & \frac{2}{3} \\ 0 & 1 & 0 \\ -\frac{2}{3} & 0 & \frac{\sqrt{5}}{3} \end{bmatrix}}_{\Omega^{[1,3]}} \begin{bmatrix} \sqrt{5} & \frac{7}{\sqrt{5}} & \frac{11}{\sqrt{5}} \\ 0 & -\frac{6}{\sqrt{5}} & -\frac{3}{\sqrt{5}} \\ 2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & -\frac{6}{\sqrt{5}} & -\frac{3}{\sqrt{5}} \\ 0 & -\frac{3}{\sqrt{5}} & -\frac{9}{\sqrt{5}} \end{bmatrix},$$

$$\Omega^{[2,3]}(\Omega^{[1,3]}\Omega^{[1,2]}\mathsf{A}) = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \\ 0 & \frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} \end{bmatrix}}_{\Omega^{[2,3]}} \begin{bmatrix} 3 & 3 & 3 \\ 0 & -\frac{6}{\sqrt{5}} & -\frac{3}{\sqrt{5}} \\ 0 & -\frac{3}{\sqrt{5}} & -\frac{9}{\sqrt{5}} \end{bmatrix} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 0 & 3 \end{bmatrix}.$$

Finally,

$$\mathsf{Q}^{\mathrm{T}} = \Omega^{[2,3]}\Omega^{[1,3]}\Omega^{[1,2]} = \frac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 2 & -2 & -1 \\ 1 & 2 & -2 \end{bmatrix},$$

$$\mathsf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \underbrace{\frac{1}{3}\begin{bmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\begin{bmatrix} 3 & 3 & 3 \\ & 3 & 3 \\ & & 3 \end{bmatrix}}_{\mathsf{R}}.$$

15/10

8.4.9 Householder transformations

A *Householder transformation* is another simple orthogonal matrix. *Householder transformations* offer an alternative to Given rotations in the calculation of a QR factorization.

Definition 8.30. If $\boldsymbol{u} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$, the $m \times m$ matrix

$$\mathsf{H} = \mathsf{H}_{\boldsymbol{u}} = \mathsf{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2} \tag{8.36a}$$

is called a *Householder transformation* (or *Householder matrix*, or *Householder reflection*, or *Householder 'rotation'*).

Remarks

(i) Each such matrix is symmetric and orthogonal, since

$$\left(\mathsf{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2}\right)^{\mathrm{T}} \left(\mathsf{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2}\right) = \left(\mathsf{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2}\right)^2 = \mathsf{I} - 4\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2} + 4\frac{\boldsymbol{u}(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u})\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^4} = \mathsf{I}.$$

(ii) Further, since
$$\mathsf{H}\boldsymbol{u} = -\boldsymbol{u}, \quad \text{and} \quad \mathsf{H}\boldsymbol{v} = \boldsymbol{v} \quad \text{if} \quad \boldsymbol{u}^{\mathrm{T}}\boldsymbol{v} = 0,$$
this transformation reflects any vector $\boldsymbol{x} \in \mathbb{R}^m$ in the $(m-1)$-dimensional hyperplane orthogonal to $\boldsymbol{u}$.

(iii) For $\lambda \in \mathbb{R} \setminus \{0\}$
$$\mathsf{H}_{\lambda\boldsymbol{u}} = \mathsf{H}_{\boldsymbol{u}}\,, \tag{8.36b}$$
since a vector orthogonal to $\boldsymbol{u}$ is also orthogonal to $\lambda\boldsymbol{u}$. 15/11

Lemma 8.31. *For any two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^m$ of equal length, the Householder transformation $\mathsf{H}_{\boldsymbol{u}}$ with $\boldsymbol{u} = \boldsymbol{a} - \boldsymbol{b}$ reflects $\boldsymbol{a}$ onto $\boldsymbol{b}$, i.e.*
$$\boldsymbol{u} = \boldsymbol{a} - \boldsymbol{b}, \quad \|\boldsymbol{a}\| = \|\boldsymbol{b}\| \quad \Rightarrow \quad \mathsf{H}_{\boldsymbol{u}}\boldsymbol{a} = \boldsymbol{b}. \tag{8.37a}$$
In particular, for any $\boldsymbol{a} \in \mathbb{R}^m$, the choice $\boldsymbol{b} = \gamma\boldsymbol{e}_1$ implies
$$\boldsymbol{u} = \boldsymbol{a} - \gamma\boldsymbol{e}_1, \quad \gamma = \pm\|\boldsymbol{a}\| \quad \Rightarrow \quad \mathsf{H}_{\boldsymbol{u}}\boldsymbol{a} = \gamma\boldsymbol{e}_1. \tag{8.37b}$$

15/12
15/14

Proof. Either draw a picture, or expand $\mathsf{H}_{\boldsymbol{u}}\boldsymbol{a}$ using the fact that
$$\begin{aligned}\|\boldsymbol{a}-\boldsymbol{b}\|^2 &= \boldsymbol{a}^{\mathrm{T}}\boldsymbol{a} - \boldsymbol{a}^{\mathrm{T}}\boldsymbol{b} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{a} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{b} \\ &= 2\boldsymbol{a}^{\mathrm{T}}(\boldsymbol{a}-\boldsymbol{b})\,.\end{aligned}$$
□

Example. To reflect
$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \overset{\mathsf{H}_{\boldsymbol{u}}}{\to} \begin{bmatrix} \gamma \\ 0 \\ 0 \end{bmatrix} = \gamma\boldsymbol{e}_1$$
with some $\boldsymbol{u}$ and γ, we should set $\gamma = \|\boldsymbol{a}\|$ or $\gamma = -\|\boldsymbol{a}\|$ to have equal lengths of vectors $\boldsymbol{a}$ and $\gamma\boldsymbol{e}_1$, and take
$$\boldsymbol{u} = \begin{bmatrix} a_1 - \|\boldsymbol{a}\| \\ a_2 \\ a_3 \end{bmatrix} \quad \text{or} \quad \boldsymbol{u} = \begin{bmatrix} a_1 + \|\boldsymbol{a}\| \\ a_2 \\ a_3 \end{bmatrix},$$
respectively. For example, we can reflect
$$\boldsymbol{a} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} \to \begin{bmatrix} \gamma \\ 0 \\ 0 \end{bmatrix} = \gamma\boldsymbol{e}_1$$
either with
$$\gamma = 3 \quad \Rightarrow \quad \boldsymbol{u} = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix} \quad \Rightarrow \quad \mathsf{H}_{\boldsymbol{u}}\boldsymbol{a} = \frac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix},$$
or with
$$\gamma = -3 \quad \Rightarrow \quad \boldsymbol{u} = \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix} \quad \Rightarrow \quad \mathsf{H}_{\boldsymbol{u}}a = \frac{1}{15}\begin{bmatrix} -10 & -5 & -10 \\ -5 & 14 & -2 \\ -10 & -2 & 11 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix}.$$

Choosing the sign of γ. For hand calculations it is generally best to choose $\operatorname{sgn}\gamma = \operatorname{sgn} a_1$ as this choice leads to smaller numbers. However for numerical calculations the opposite choice generally provides better stability (since when $\|\boldsymbol{u}\| \ll 1$ numerical difficulties can occur as a result of division by a tiny number). 15/13

Example. To reflect

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \overset{H_{\boldsymbol{u}}}{\to} \begin{bmatrix} a_1 \\ \gamma \\ 0 \\ 0 \end{bmatrix} = \boldsymbol{b},$$

we set $\gamma = \pm\sqrt{a_2^2 + a_3^2 + a_4^2}$ (to equalize the lengths), and take

$$\boldsymbol{u} = \boldsymbol{a} - \boldsymbol{b} = \begin{bmatrix} 0 \\ a_2 - \gamma \\ a_3 \\ a_4 \end{bmatrix}.$$

Theorem 8.32. *For any matrix* $\mathsf{A} \in \mathbb{R}^{m\times n}$*, there exist Householder matrices* H_k *such that*

$$\mathsf{R} = \mathsf{H}_{n-1}\cdots\mathsf{H}_2\mathsf{H}_1\mathsf{A}$$

is an upper triangular matrix.

Proof. Our goal is to multiply the $m \times n$ matrix A by a sequence of Householder transformations so that each product 'peels off' the requisite nonzero elements in a succession of whole columns. To start with, we seek a transformation that transforms the first nonzero column of A to a multiple of $\boldsymbol{e}_1$.

(i) We take $\mathsf{H}_1 = \mathsf{H}_{\boldsymbol{u}_1}$ with the vector $\boldsymbol{u}_1 = \boldsymbol{a}_1 - \gamma_1\boldsymbol{e}_1$, where $\boldsymbol{a}_1$ is the first column of A. Then $\mathsf{H}_1\mathsf{A}$ has $\gamma_1\boldsymbol{e}_1$ as its first column. Explicitly,

$$\boldsymbol{a}_1 = \begin{bmatrix} A_{1,1} \\ A_{2,1} \\ A_{3,1} \\ A_{4,1} \end{bmatrix}, \quad \boldsymbol{u}_1 = \begin{bmatrix} A_{1,1} - \gamma_1 \\ A_{2,1} \\ A_{3,1} \\ A_{4,1} \end{bmatrix}, \quad \gamma_1 = \pm\|\boldsymbol{a}_1\| \quad \Rightarrow \quad \mathsf{A}^{(1)} = \mathsf{H}_1\mathsf{A} = \begin{bmatrix} \gamma_1 & A^{(1)}_{1,2} & * & * \\ 0 & A^{(1)}_{2,2} & * & * \\ 0 & A^{(1)}_{3,2} & * & * \\ 0 & A^{(1)}_{4,2} & * & * \end{bmatrix}.$$

(ii) We take $\mathsf{H}_2 = \mathsf{H}_{u_2}$ with the vector $\boldsymbol{u}_2$ formed from the bottom three components of the second column, $\boldsymbol{a}_2^{(1)}$, of $\mathsf{A}^{(1)}$:

$$\boldsymbol{u}_2 = \begin{bmatrix} 0 \\ A^{(1)}_{2,2} - \gamma_2 \\ A^{(1)}_{3,2} \\ A^{(1)}_{4,2} \end{bmatrix}, \quad \gamma_2 = \pm\left(\sum_{i=2}^{4}\left(A^{(1)}_{i,2}\right)^2\right)^{\frac{1}{2}} \quad \Rightarrow \quad \mathsf{A}^{(2)} = \mathsf{H}_2\mathsf{A}^{(1)} = \begin{bmatrix} \gamma_1 & A^{(1)}_{1,2} & A^{(1)}_{1,3} & * \\ 0 & \gamma_2 & A^{(2)}_{2,3} & * \\ 0 & 0 & A^{(2)}_{3,3} & * \\ 0 & 0 & A^{(2)}_{4,3} & * \end{bmatrix}.$$

(iii) We take $\mathsf{H}_3 = \mathsf{H}_{u_3}$ with the vector $\boldsymbol{u}_3$ formed from the bottom two components of the third column, $\boldsymbol{a}_3^{(2)}$, of $\mathsf{A}^{(2)}$:

$$\boldsymbol{u}_3 = \begin{bmatrix} 0 \\ 0 \\ A^{(2)}_{3,3} - \gamma_3 \\ A^{(2)}_{4,3} \end{bmatrix}, \quad \gamma_3 = \pm\left(\sum_{i=3}^{4}\left(A^{(2)}_{i,3}\right)^2\right)^{\frac{1}{2}} \quad \Rightarrow \quad \mathsf{A}^{(3)} = \mathsf{H}_3\mathsf{A}^{(2)} = \begin{bmatrix} \gamma_1 & A^{(1)}_{1,2} & A^{(1)}_{1,3} & A^{(1)}_{1,4} \\ 0 & \gamma_2 & A^{(2)}_{2,3} & A^{(2)}_{2,4} \\ 0 & 0 & \gamma_3 & A^{(3)}_{3,4} \\ 0 & 0 & 0 & A^{(3)}_{4,4} \end{bmatrix}, \text{ etc.}$$

At the k-th step

(a) by construction the bottom $m - k$ components of the k-th column of $\mathsf{A}^{(k)} = \mathsf{H}_k\mathsf{A}^{(k-1)}$ will vanish (by Lemma 8.31);

(b) the first $k - 1$ columns of $\mathsf{A}^{(k-1)}$ remain invariant under reflection H_k (because they are orthogonal to $\boldsymbol{u}_k$), as well as the first $k - 1$ rows;

(c) therefore, the first k columns of $\mathsf{A}^{(k)}$ have an upper triangular form.

The end result is an upper triangular matrix R in standard form. □

Remark. In pictures:

$$\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \xrightarrow{\mathsf{H}_1} \begin{bmatrix} \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \end{bmatrix} \xrightarrow{\mathsf{H}_2} \begin{bmatrix} * & * & * \\ 0 & \bullet & \bullet \\ 0 & 0 & \bullet \\ 0 & 0 & \bullet \end{bmatrix} \xrightarrow{\mathsf{H}_3} \begin{bmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & \bullet \\ 0 & 0 & 0 \end{bmatrix}$$

The $\bullet$-elements have changed through a single reflection while the $*$-elements have remained the same.

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 Q. If the matrix Q is required in an explicit form, set $\Omega = \mathsf{I}$ initially and, for each successive transformation, replace Ω by

$$\left(\mathsf{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{\|\boldsymbol{u}\|^2}\right)\Omega = \Omega - \frac{2}{\|\boldsymbol{u}\|^2}\boldsymbol{u}(\boldsymbol{u}^{\mathrm{T}}\Omega),$$

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 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 Q, then we set initially $\boldsymbol{c} = \boldsymbol{b}$ 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 QR factorization by Householder reflections of

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

First, we do this [the long way] by calculating the H_k:

$$\mathsf{A} = \left[\begin{array}{|c|cc} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{array}\right],$$

$$\boldsymbol{u}_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}, \quad \mathsf{H}_1 = \tfrac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix}, \quad \mathsf{A}^{(1)} = \mathsf{H}_1\mathsf{A} = \left[\begin{array}{c|c|c} 3 & 3 & 3 \\ 0 & 0 & 3 \\ 0 & 3 & 3 \end{array}\right],$$

$$\boldsymbol{u}_2 = \begin{bmatrix} 0 \\ -3 \\ 3 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}, \quad \mathsf{H}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathsf{R} = \mathsf{H}_2\mathsf{A}^{(1)} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 0 & 3 \end{bmatrix}.$$

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 H_k:

$$\boldsymbol{u}_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}, \quad 2\frac{\boldsymbol{u}_1^{\mathrm{T}}\mathsf{A}}{\|\boldsymbol{u}_1\|^2} = \begin{bmatrix} 1 & -1 & -2 \end{bmatrix}, \quad \mathsf{A}^{(1)} = \mathsf{A} - 2\frac{\boldsymbol{u}_1(\boldsymbol{u}_1^{\mathrm{T}}\mathsf{A})}{\|\boldsymbol{u}_1\|^2} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 3 & 3 \end{bmatrix},$$

$$\boldsymbol{u}_2 = \begin{bmatrix} 0 \\ -3 \\ 3 \end{bmatrix}, \quad 2\frac{\boldsymbol{u}_2^{\mathrm{T}}\mathsf{A}^{(1)}}{\|\boldsymbol{u}_2\|^2} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \quad \mathsf{R} = \mathsf{A}^{(1)} - 2\frac{\boldsymbol{u}_2(\boldsymbol{u}_2^{\mathrm{T}}\mathsf{A}^{(1)})}{\|\boldsymbol{u}_2\|^2} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 0 & 3 \end{bmatrix}.$$

If we require Q, start with $\Omega = \mathsf{I}$, then

$$\boldsymbol{u}_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}, \quad 2\frac{\boldsymbol{u}_1^{\mathrm{T}}\Omega}{\|\boldsymbol{u}_1\|^2} = \tfrac{1}{3}\begin{bmatrix} -1 & 1 & 2 \end{bmatrix}, \quad \Omega^{(1)} = \Omega - 2\frac{\boldsymbol{u}_1(\boldsymbol{u}_1^{\mathrm{T}}\Omega)}{\|\boldsymbol{u}_1\|^2} = \tfrac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix},$$

$$\boldsymbol{u}_2 = \begin{bmatrix} 0 \\ -3 \\ 3 \end{bmatrix}, \quad 2\frac{\boldsymbol{u}_2^{\mathrm{T}}\Omega^{(1)}}{\|\boldsymbol{u}_2\|^2} = \tfrac{1}{9}\begin{bmatrix} 1 & -4 & 1 \end{bmatrix}, \quad \mathsf{Q}^{\mathrm{T}} = \Omega^{(1)} - 2\frac{\boldsymbol{u}_2(\boldsymbol{u}_2^{\mathrm{T}}\Omega^{(1)})}{\|\boldsymbol{u}_2\|^2} = \tfrac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 2 & -2 & -1 \\ 1 & 2 & -2 \end{bmatrix}.$$

As before

$$\mathsf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \underbrace{\tfrac{1}{3}\begin{bmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{bmatrix}}_{\mathsf{Q}} \cdot \underbrace{\begin{bmatrix} 3 & 3 & 3 \\ & 3 & 3 \\ & & 3 \end{bmatrix}}_{\mathsf{R}}.$$

Givens or Householder? If A is dense, it is in general more convenient to use Householder transformations. Givens rotations come into their own, however, when A has many leading zeros in its rows. In an extreme case, if an $n \times n$ matrix 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.

9 Linear Least Squares

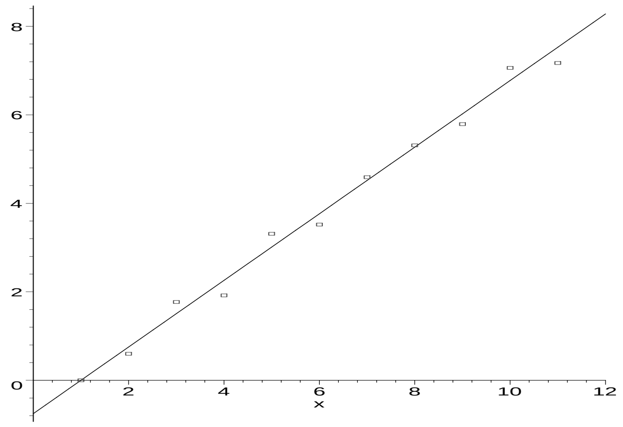
9.1 Statement of the problem

Suppose that an $m \times n$ matrix A and a vector $\boldsymbol{b} \in \mathbb{R}^m$ are given. If $m < n$ the equation $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ usually has an infinity of solutions, while if $m > n$ it in general has no solution. When there are more equations than unknowns, the system $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ is called *overdetermined.* In general, an overdetermined system has no solution, but we would like to have $\mathsf{A}\boldsymbol{x}$ and $\boldsymbol{b}$ close in a sense. Choosing the Euclidean distance $\|\boldsymbol{z}\| = (\sum_{i=1}^m z_i^2)^{1/2}$ as a measure of closeness, we obtain the following problem.

Problem 9.1 (Least squares in $\mathbb{R}^m$)**.** Given $\mathsf{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, find

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 \,, \tag{9.1}$$

i.e., find (the argument) $\boldsymbol{x}^* \in \mathbb{R}^n$ which minimizes (the functional) $\|\mathsf{A}\boldsymbol{x}-\boldsymbol{b}\|^2$ — the *least-squares problem.*



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 9.11: Least squares straight line data fitting.

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 exploit them to form 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 $F(x)$, and that we wish to model F with a linear combination of n functions $\phi_j(x)$, i.e.

$$F(x) = c_1\phi_1(x) + c_2\phi_2(x) + \cdots + c_n\phi_n(x)\,, \quad \text{and} \quad F(x_i) \approx y_i, \quad i = 1...m.$$

Such a problem might occur if we were trying to match some planet observations to an ellipse. Hence we want to determine $\boldsymbol{c}$ such that the $F(x_i)$ 'best' fit the y_i, i.e.

$$\mathsf{A}\boldsymbol{c} = \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \vdots & & \vdots \\ \phi_1(x_n) & \cdots & \phi_n(x_n) \\ \vdots & & \vdots \\ \phi_1(x_m) & \cdots & \phi_n(x_m) \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} F(x_1) \\ \vdots \\ F(x_n) \\ \vdots \\ F(x_m) \end{bmatrix} \approx \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ \vdots \\ y_m \end{bmatrix} = \boldsymbol{y}.$$

There are many ways of doing this; we will determine the $\boldsymbol{c}$ that minimizes the sum of squares of the deviation, i.e. we minimize

$$\sum_{i=1}^{m} (F(x_i) - y_i)^2 = \|\mathsf{A}\boldsymbol{c} - \boldsymbol{y}\|^2.$$

This leads to a *linear* system of equations for the determination of the unknown $\boldsymbol{c}$.

Theorem 9.2. *$\boldsymbol{x} \in \mathbb{R}^n$ is a solution of the least-squares problem* (9.1) *iff* $\mathsf{A}^{\mathrm{T}}(\mathsf{A}\boldsymbol{x} - \boldsymbol{b}) = \mathbf{0}$.

Proof. If $\boldsymbol{x}$ is a solution then it minimizes

$$f(\boldsymbol{x}) = \|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 = \langle \mathsf{A}\boldsymbol{x} - \boldsymbol{b}, \mathsf{A}\boldsymbol{x} - \boldsymbol{b}\rangle = \boldsymbol{x}^{\mathrm{T}}\mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} - 2\boldsymbol{x}^{\mathrm{T}}\mathsf{A}^{\mathrm{T}}\boldsymbol{b} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{b}.$$

At a minimum the gradient $\nabla f = \left[\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n}\right]^{\mathrm{T}}$ vanishes, i.e. $\nabla f(\boldsymbol{x}) = \mathbf{0}$. But

$$\nabla f(\boldsymbol{x}) = 2\mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} - 2\mathsf{A}^{\mathrm{T}}\boldsymbol{b} = 2\mathsf{A}^{\mathrm{T}}(\mathsf{A}\boldsymbol{x} - \boldsymbol{b}), \quad \text{and hence} \quad \mathsf{A}^{\mathrm{T}}(\mathsf{A}\boldsymbol{x} - \boldsymbol{b}) = \mathbf{0}.$$

Conversely, suppose that $\mathsf{A}^{\mathrm{T}}(\mathsf{A}\boldsymbol{x} - \boldsymbol{b}) = \mathbf{0}$ and let $\boldsymbol{u} \in \mathbb{R}^n$. Hence, letting $\boldsymbol{y} = \boldsymbol{u} - \boldsymbol{x}$,

$$\begin{aligned} \|\mathsf{A}\boldsymbol{u} - \boldsymbol{b}\|^2 &= \langle \mathsf{A}\boldsymbol{y} + (\mathsf{A}\boldsymbol{x} - \boldsymbol{b}), \mathsf{A}\boldsymbol{y} + (\mathsf{A}\boldsymbol{x} - \boldsymbol{b})\rangle \\ &= \langle \mathsf{A}\boldsymbol{y}, \mathsf{A}\boldsymbol{y}\rangle + 2\boldsymbol{y}^{\mathrm{T}}\mathsf{A}^{\mathrm{T}}(\mathsf{A}\boldsymbol{x} - \boldsymbol{b}) + \langle \mathsf{A}\boldsymbol{x} - \boldsymbol{b}, \mathsf{A}\boldsymbol{x} - \boldsymbol{b}\rangle \\ &= \|\mathsf{A}\boldsymbol{y}\|^2 + \|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \geqslant \|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 . \end{aligned}$$

It follows that $\boldsymbol{x}$ is indeed optimal (i.e. a minimizer). Moreover, if A is of full rank and $\boldsymbol{y} = \boldsymbol{u} - \boldsymbol{x} \not\equiv 0$, then the last inequality is strict, hence $\boldsymbol{x}$ is unique. □

Corollary 9.3. *The optimality of $\boldsymbol{x}$ $\Leftrightarrow$ the vector $\mathsf{A}\boldsymbol{x} - \boldsymbol{b}$ is orthogonal to all columns of A.*

Remark. This corollary has a geometrical visualization. If we denote the columns of A as $\boldsymbol{a_j}$, $j = 1, \ldots, n$, then the least squares problem is the problem of finding the value

$$\min_{\boldsymbol{x}} \left\| \sum_{j=1}^{n} x_j \boldsymbol{a}_j - \boldsymbol{b} \right\| ,$$

which is the minimum of the Euclidean distance between a given vector $\boldsymbol{b}$ and vectors in the plane $\mathcal{B} = \mathrm{span}\,(\boldsymbol{a}_j)$. Geometrically this minimum is attained when

$$\sum_{j=1}^{n} x_j \boldsymbol{a}_j = \mathsf{A}\boldsymbol{x}$$

is the foot of the perpendicular from $\boldsymbol{b}$ onto the plane $\mathcal{B}$, i.e. when $\boldsymbol{b} - \sum x_j \boldsymbol{a}_j = \boldsymbol{b} - \mathsf{A}\boldsymbol{x}$ is orthogonal to all vectors $\boldsymbol{a}_j$ (i.e. the columns of A).

9.2 Normal equations

One way of finding optimal $\boldsymbol{x}$ is by solving the $n \times n$ linear system

$$\mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} = \mathsf{A}^{\mathrm{T}}\boldsymbol{b}. \tag{9.2}$$

This is the method of *normal equations*; $\mathsf{A}^{\mathrm{T}}\mathsf{A}$ is the *Gram* matrix, while $\boldsymbol{b}$ is the *normal* solution.

Example. The least squares approximation, by a straight line, to the data plotted in Figure 9.11

$$F(x) = c_1 + c_2 x \qquad (\phi_1(x) = 1, \quad \phi_2(x) = x),$$

results in the following normal equations and solution:

$$\mathsf{A} = \big(\phi_j(x_i)\big) = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ \vdots & \vdots \\ 1 & 11 \end{bmatrix}, \quad \underbrace{\begin{bmatrix} 11 & 66 \\ 66 & 506 \end{bmatrix}}_{\mathsf{A}^{\mathrm{T}}\mathsf{A}} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 41.04 \\ 328.05 \end{bmatrix}}_{\mathsf{A}^{\mathrm{T}}\boldsymbol{y}} \quad \Rightarrow \quad \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -0.7314 \\ 0.7437 \end{bmatrix}.$$

Remark. A normal equation approach to the solution of the linear least squares problem is popular in many applications; however, there are three disadvantages.

(i) Firstly, $\mathsf{A}^{\mathrm{T}}\mathsf{A}$ might be singular.

(ii) Secondly a sparse A might be replaced by a dense $\mathsf{A}^{\mathrm{T}}\mathsf{A}$.

(iii) Finally, forming $\mathsf{A}^{\mathrm{T}}\mathsf{A}$ might lead to loss of accuracy. For instance, suppose that our computer works to the IEEE arithmetic standard (≈ 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!

9.3 The solution of the least-squares problem using QR factorisation.

An alternative to solution by normal equations is provided by QR factorisation. First, some revision.

Lemma 9.4. *Let $\boldsymbol{v} \in \mathbb{R}^m$, and let Ω be an $m\times m$ orthogonal matrix. Then $\|\Omega\boldsymbol{v}\| = \|\boldsymbol{v}\|$, i.e. the Euclidean length of a vector is unchanged if the vector is pre-multiplied by any orthogonal matrix.*

Proof.

$$\|\Omega\boldsymbol{v}\|^2 = (\Omega\boldsymbol{v})^{\mathrm{T}}(\Omega\boldsymbol{v}) = \boldsymbol{v}^{\mathrm{T}}\Omega^{\mathrm{T}}\Omega\boldsymbol{v} = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{v} = \|\boldsymbol{v}\|^2 . \qquad \square$$

Corollary 9.5. *Let A be any $m \times n$ matrix and let $\boldsymbol{b} \in \mathbb{R}^m$. The vector $\boldsymbol{x} \in \mathbb{R}^n$ minimises $\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|$ iff it minimises $\|\Omega\mathsf{A}\boldsymbol{x} - \Omega\boldsymbol{b}\|$ for an arbitrary $m \times m$ orthogonal matrix Ω.*

Proof.

$$\|\Omega\mathsf{A}\boldsymbol{x} - \Omega\boldsymbol{b}\|^2 = \|\Omega(\mathsf{A}\boldsymbol{x} - \boldsymbol{b})\|^2 = \|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|^2. \qquad \square$$

Suppose that $\mathsf{A} = \mathsf{QR}$ is a QR factorization of A with R in a *standard form.* Because of the corollary, and letting $\Omega = \mathsf{Q}^{\mathrm{T}}$, minimizing $\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|$ for $\boldsymbol{x} \in \mathbb{R}^n$, is equivalent to minimizing

$$\|\mathsf{Q}^{\mathrm{T}}\mathsf{A}\boldsymbol{x} - \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\| = \|\mathsf{R}\boldsymbol{x} - \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\|. \tag{9.3}$$

Next we note that because R is in standard form, its nonzero rows, $\{\boldsymbol{r}_i^{\mathrm{T}} : i=1,2,\ldots,\ell\}$ say, are linearly independent. Therefore

(i) there exists $\boldsymbol{x}$ satisfying $\boldsymbol{r}_i^{\mathrm{T}}\boldsymbol{x} = (\mathsf{Q}^{\mathrm{T}}\boldsymbol{b})_i$, $i=1,2,\ldots,\ell$,

(ii) $\boldsymbol{x}$ can be computed by back-substitution using the upper triangularity of R,

(iii) $\boldsymbol{x}$ is unique if and only if $\ell = n$.

To demonstrate that such an $\boldsymbol{x}$ is a least-squares solution, we note that because the last $(m–\ell)$ components of $\mathsf{R}\boldsymbol{x}$ are zero for every $\boldsymbol{x} \in \mathbb{R}^n$,

$$\|\mathsf{R}\boldsymbol{x} - \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\|^2 = \sum_{i=1}^{\ell}(\mathsf{R}\boldsymbol{x} - \mathsf{Q}^{\mathrm{T}}\boldsymbol{b})_i^2 + \sum_{i=\ell+1}^{m}(\mathsf{Q}^{\mathrm{T}}\boldsymbol{b})_i^2 \geqslant \sum_{i=\ell+1}^{m}(\mathsf{Q}^{\mathrm{T}}\boldsymbol{b})_i^2 , \tag{9.4}$$

with equality iff $\boldsymbol{r}_i^{\mathrm{T}}\boldsymbol{x} = (\mathsf{Q}^{\mathrm{T}}\boldsymbol{b})_i$, $i=1,2,\ldots,\ell$. Hence the back-substitution provides an $\boldsymbol{x}$ that minimizes $\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|$ as required.

Remarks

(a) Often $m \gg n$, in which case many rows of R consist of zeros.

(b) Note that we do not require Q explicitly, we need only to evaluate $\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}$.

9.3.1 Examples

(i) Find $\boldsymbol{x} \in \mathbb{R}^3$ that minimizes $\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|$, where

$$\mathsf{A} = \tfrac{1}{2}\begin{bmatrix} 1 & 3 & 6 \\ 1 & 1 & 2 \\ 1 & 3 & 4 \\ 1 & 1 & 0 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

To this end we need to solve the system

$$\mathsf{A}^{\mathrm{T}}(\mathsf{A}\boldsymbol{x} - \boldsymbol{b}) = 0\,,$$

or equivalently

$$\mathsf{R}^{\mathrm{T}}\mathsf{Q}^{\mathrm{T}}(\mathsf{Q}\mathsf{R}\boldsymbol{x} - \boldsymbol{b}) = \mathsf{R}^{\mathrm{T}}(\mathsf{R}\boldsymbol{x} - \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}) = 0\,,$$

which can be achieved by first solving

$$\mathsf{R}^{\mathrm{T}}\boldsymbol{y} = 0\,, \quad \text{and then} \quad \mathsf{R}\boldsymbol{x} = \mathsf{Q}^{\mathrm{T}}\boldsymbol{b} + \boldsymbol{y}\,.$$

The QR-factorization of A is

$$\mathsf{A} = \underbrace{\tfrac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}}_{\mathsf{Q}} \times \underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathsf{R}}.$$

Hence

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}}_{\mathsf{R}^{\mathrm{T}}} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = 0\,, \quad \text{with solution} \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \lambda \end{bmatrix}$$

where $\lambda \in \mathbb{R}$. Next

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathsf{R}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \underbrace{\tfrac{1}{2}\begin{bmatrix} 3 \\ 1 \\ 1 \\ -1 \end{bmatrix}}_{\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \lambda \end{bmatrix},$$

which has solution

$$\lambda = \tfrac{1}{2}\,, \quad \boldsymbol{x} = \tfrac{1}{2}\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}.$$

The error is the norm of the vector formed by the bottom $(m - n)$ components of the right-hand side $\mathsf{Q}^{\mathrm{T}}\boldsymbol{b}$:

$$\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\| = \|\mathsf{R}\boldsymbol{x} - \mathsf{Q}^{\mathrm{T}}\boldsymbol{b}\| = \|\boldsymbol{y}\| = \tfrac{1}{2}\,.$$

(ii) Using the normal solution, 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 error is

$$\mathsf{A}\boldsymbol{c} - \boldsymbol{y} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ -1 \end{bmatrix} \quad \Rightarrow \quad \|\mathsf{A}\boldsymbol{c} - \boldsymbol{y}\| = \sqrt{2}\,.$$

16/10
16/11
16/12
16/13
16/14

10 Questionnaire Results

Questionnaire results are available at `http://tinyurl.com/NA-IB-2014-Results`.