MATH2600: Numerical Analysis

School of Mathematics, University of Leeds

Lecturer: Jitse Niesen

Office: Room 9.22f, School of Mathematics

Phone: 0113 343 5870

E-mail: J.Niesen@leeds.ac.uk

Website: http://www.maths.leeds.ac.uk/~jitse/math2600.html

Lectures: Wednesday 09:00–10:00, Roger Stevens LT 17

Friday 12:00–13:00, Roger Stevens LT 25

Office Hours: Open door policy

Workshops:

Monday [Group 1] 14:00–15:00, Roger Stevens LT 10
Evy Kersalé (E.Kersale@leeds.ac.uk)

Monday [Group 2] 14:00–15:00, Roger Stevens LT 12
Alastair Rucklidge (A.M.Rucklidge@leeds.ac.uk)

Tuesday [Group 3] 13:00–14:00, Chemistry West Block LT E
Jitse Niesen (J.Niesen@leeds.ac.uk)

Course Outline:

- Introduction and mathematical preliminaries;
- Solving nonlinear equations;
- Interpolation;
- Numerical integration;
- Solving ordinary differential equations;
- Linear systems of equations.

Pre-requisites: Linear algebra, Taylor series, calculus, ODEs.

Assessment: 85% final examination and 15% coursework. (10 credits)

Textbooks:

- Recommended : R.L. Burden & J.D. Faires: *Numerical Analysis*, (Brooks/Cole 8$^{\text{th}}$ Edition 2005, or any other edition.)
- A. Ralston & P. Rabinowitz: *A First course in Numerical Analysis*, (Dover 2$^{\text{nd}}$ Edition 2001, or any other edition.)
- C.F. Gerald & P.O. Wheatley: *Applied Numerical Analysis*, (Addison-Wesley 7$^{\text{th}}$ Edition 2004, or any other edition.)

Lectures:

- You should read through and understand your notes before the next lecture ... otherwise you will get hopelessly lost.
- Please, do not hesitate to interrupt me whenever you have questions or if I am inaudible, illegible, unclear or just plain wrong. (I shall also stay at the front for a few minutes after lectures in order to answer questions.)
- If you feel that the module is too difficult, or that you are spending too much time on it, please come and talk to me.
- Please, do not wait until the end of term to give a feedback if you are unhappy with some aspects of the module.

Printed Notes:

- Detailed printed notes will be handed out for the module, so that you can listen to me and make the most of the lectures, rather than having to write down every sentence on paper. However, the notes provided should only be used as a supplement, not as an alternative to your personal notes.
- These printed notes are an adjunct to lectures and are not meant to be used independently.
- With a few exceptions, worked examples are deliberately omitted from the printed notes. The aim being that going through the examples at least once will help you to learn the material. (It might also help you to stay awake if you have to write something down from time to time.)
- Please email me (J.Niesen@leeds.ac.uk) corrections to the notes, examples sheets and model solutions.

Example Sheets & Homework:

- Five example sheets in total to be handed out every fortnight.
- Examples will help you to understand the material taught in the lectures and will give you practice on the types of questions that will be set in the examination. It is very important that you try them before the example classes.
- There will be only two, yet quite demanding, pieces of coursework. The **deadlines** are **1 November** and **6 December**. Your work will be marked and returned to you with a grade from 1–100.
- Model solutions will be distributed once the homework is handed in.

Contents

Chapter 1

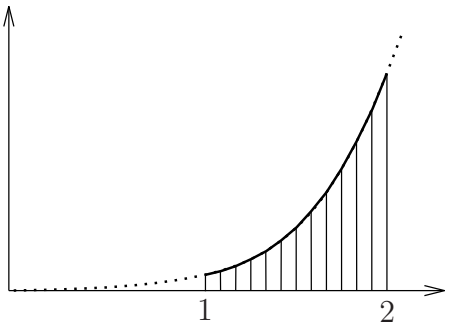
Introduction and mathematical preliminaries

Contents

1.1 Motivation

Most of the problems that you have encountered so far are unusual in that they can be solved analytically (e.g. integrals, differential equations). However, this is somewhat contrived as, in real-life, analytic solutions are rather rare, and therefore we must devise a way of approximating the solutions.

For example, while the integral $\int_1^2 \mathrm{e}^x \mathrm{d}x$ has a well known analytic solution, $\int_1^2 \mathrm{e}^{x^2} \mathrm{d}x$ can only be solved in terms of special functions and $\int_1^2 \mathrm{e}^{x^3} \mathrm{d}x$ has no analytic solution. These integrals however exist as the area under the curves $y = \exp(x^2)$ and $y = \exp(x^3)$; so, we can obtain a numerical approximation by estimating this area, e.g. by dividing it into strips and using the trapezium rule.



Numerical analysis is a part of mathematics concerned with (i) devising methods, called numerical algorithms, for obtaining numerical approximate solutions to mathematical problems and, importantly, (ii) being able to estimate the error involved. Traditionally, numerical algorithms are built upon the most simple arithmetic operations ($+, -, \times$ and $\div$).

Interestingly, digital computers can only do these very basic operations (although very sophisticated programmes like Maple exist). However, they are very fast and, hence, have led to major advances in applied mathematics in the last 60 years.

1.2 Finite-digit arithmetic

1.2.1 Computer arithmetic

Floating point numbers. Computers can store integers exactly but not real numbers in general. Instead, they approximate them as floating point numbers.

A *decimal* floating point (or machine number) is a number of the form

$$\pm 0.\underbrace{d_1\, d_2 \ldots d_k}_{m} \times 10^n, \quad 0 \le d_i \le 9, \quad d_1 \neq 0,$$

where the significand or mantissa m (i.e. the fractional part) and the exponent n are fixed-length integers. (m cannot start with a zero.)

In fact, computers use binary numbers (base 2) rather than decimal numbers (base 10) but the same principle applies (see appendix A).

Machine ε. Consider a simple computer where m is 3 digits long and n is one digit long. The smallest positive number this computer can store is 0.1×10^{-9} and the largest is 0.999×10^9.

Thus, the length of the exponent determines the range of numbers that can be stored. However, not all values in the range can be distinguished: numbers can only be recorded to a certain relative accuracy ε.

For example, on our simple computer, the next floating point number after $1 = 0.1 \times 10^1$ is $0.101 \times 10^1 = 1.01$. The quantity $\varepsilon_{\text{machine}} = 0.01$ (machine ε) is the worst relative uncertainty in the floating point representation of a number.

1.2.2 Chopping and rounding

There are two ways of terminating the mantissa of the k-digit decimal machine number approximating $0.d_1\, d_2 \ldots d_k\, d_{k+1}\, d_{k+2} \ldots \times 10^n$, $0 \le d_i \le 9$, $d_1 \neq 0$,

i. chopping: chop off the digits $d_{k+1}, d_{k+2}, \ldots$ to get $0.d_1\, d_2 \ldots d_k \times 10^n$.

ii. rounding: add $5 \times 10^{n-(k+1)}$ and chop off the $k+1, k+2, \ldots$ digits. (If $d_{k+1} \ge 5$ we add 1 to d_k before chopping.) Rounding is more accurate than chopping.

Example 1.1
The five-digit floating-point form of $\pi = 3.14159265359\ldots$ is 0.31415×10 using chopping and 0.31416×10 using rounding.

Similarly, the five-digit floating-point form of $2/3 = 0.6666666\ldots$ is 0.66666 using chopping and 0.66667 using rounding but that of $1/3 = 0.3333333\ldots$ is 0.33333 using either chopping or rounding.

1.3 Errors in numerical calculations

Much of numerical analysis is concerned with controlling the size of errors in calculations. These errors, quantified in two different ways, arise from two distinct sources.

1.3.1 Measure of the error

Let $p_\star$ be the result of a numerical calculation and p the exact answer (i.e. $p_\star$ is an approximation to p). We define two measures of the error,

i. absolute error: $E = |p - p_\star|$

ii. relative error: $E_r = |p - p_\star|/|p|$ (provided $p \neq 0$) which takes into consideration the size of the value.

Example 1.2
If $p = 2$ and $p_\star = 2.1$, the absolute error $E = 10^{-1}$; if $p = 2 \times 10^{-3}$ and $p_\star = 2.1 \times 10^{-3}$, $E = 10^{-4}$ is smaller; and if $p = 2 \times 10^3$ and $p_\star = 2.1 \times 10^3$, $E = 10^2$ is larger but in all three cases the relative error remains the same, $E_r = 5 \times 10^{-2}$.

1.3.2 Round-off errors

Caused by the imprecision of using finite-digit arithmetic in practical calculations (e.g. floating point numbers).

Example 1.3
The 4-digit representation of $x = \sqrt{2} = 1.4142136\ldots$ is $x_\star = 1.414 = 0.1414 \times 10$. Using 4-digit arithmetic, we can evaluate $x_\star^2 = 1.999 \neq 2$, due to round-off errors.

Significant digits. The number $p_\star$ approximates p to k significant digits if k if the largest non negative integer such that

$$\frac{|p - p_\star|}{|p|} \leq 5 \times 10^{-k}.$$

This is the bound on the relative error when using k-digit rounding arithmetic.

For example, the number $p_\star$ approximates $p = 100$ to 4 significant digits if

$$99.95 < p_\star < 100.05.$$

Magnification of the error. Computers store numbers to a relative accuracy ε. Thus, the true value of a floating point number $x_\star$ could be anywhere between $x_\star(1 - \varepsilon)$ and $x_\star(1 + \varepsilon)$. Now, if we add two numbers together, $x_\star + y_\star$, the true value lies in the interval

$$(x_\star + y_\star - \varepsilon(|x_\star| + |y_\star|), x_\star + y_\star + \varepsilon(|x_\star| + |y_\star|)) .$$

Thus, the absolute error is the sum of the errors in x and y, $E = \varepsilon(|x_\star| + |y_\star|)$ but the relative error of the answer is

$$E_r = \varepsilon \frac{|x_\star| + |y_\star|}{|x_\star + y_\star|}.$$

If $x_\star$ and $y_\star$ both have the same sign the relative accuracy remains equal to ε, but if the have opposite signs the relative error will be larger.

This magnification becomes particularly significant when two very close numbers are subtracted.

Example 1.4
Recall: the *exact* solutions of the quadratic equation $ax^2 + bx + c = 0$ are

$$x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}.$$

Using 4-digit rounding arithmetic, solve the quadratic equation $x^2 + 62x + 1 = 0$, with roots $x_1 \simeq -61.9838670$ and $x_2 \simeq -0.016133230$.
The discriminant $\sqrt{b^2 - 4ac} = \sqrt{3840} = 61.97$ is close to $b = 62$. Thus, $x_{1\star} = (-62 - 61.97)/2 = -124.0/2 = -62$, with a relative error $E_r = 2.6 \times 10^{-4}$, but $x_{2\star} = (-62 + 61.97)/2 = -0.0150$, with a much larger relative error $E_r = 7.0 \times 10^{-2}$.

Similarly, division by small numbers (or equivalently, multiplication by large numbers) magnifies the absolute error, leaving the relative error unchanged.

Round-off errors can be minimised by reducing the number of arithmetic operations, particularly those that magnify errors (e.g. using nested form of polynomials).

1.3.3 Truncation errors

Caused by the approximations in the computational algorithm itself. (An algorithm only gives an approximate solution to a mathematical problem, even if the arithmetic is exact.)

Example 1.5
Calculate the derivative of a function $f(x)$ at the point x_0.
Recall the definition of the derivative

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x_0) = \lim_{h\to 0} \frac{f(x_0 + h) - f(x_0)}{h}.$$

However, on a computer we cannot take $h \to 0$ (there exists a smallest positive floating point number), so h must take a finite value.
Using Taylor's theorem, $f(x_0 + h) = f(x_0) + hf'(x_0) + h^2/2\, f''(\xi)$, where $x_0 < \xi < x_0 + h$. Therefore,

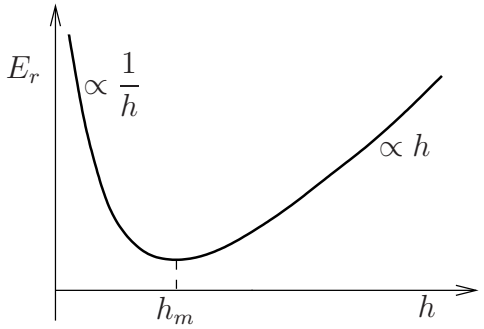
$$\frac{f(x_0 + h) - f(x_0)}{h} = \frac{hf'(x_0) + h^2/2\, f''(\xi)}{h} = f'(x_0) + \frac{h}{2} f''(\xi) \simeq f'(x_0) + \frac{h}{2} f''(x_0). \quad (1.1)$$

Consequently, using a finite value of h leads to a truncation error of size $\simeq h/2\,|f''(x_0)|$. (It is called a first order error since it is proportional to h.)

Clearly, small values of h make the truncation error smaller. However, because of round-off errors, $f(x_0)$ is only known to a relative accuracy of ε.
The absolute round-off error in $f(x_0 + h) - f(x_0)$ is $2\varepsilon|f(x_0)|$ and that in the derivative $(f(x_0 + h) - f(x_0))/h$ is $2\varepsilon/h\,|f(x_0)|$ (assuming h is exact). Note that this error increases with decreasing h.
The relative accuracy of the calculation of $f'(x_0)$ (i.e. the sum of the relative truncation and round-off errors) is

$$E_r = \frac{h}{2}\frac{|f''|}{|f'|} + \frac{2\varepsilon}{h}\frac{|f|}{|f'|},$$

which has a minimum, $\min(E_r) = 2\sqrt{\varepsilon}\sqrt{|ff''|}/|f'|$, for $h_m = 2\sqrt{\varepsilon}\sqrt{|f/f''|}$.



To improve the computation of the derivative we can use the central difference approximation (Taylor's theorem)

$$\frac{f(x_0 - h) - f(x_0 + h)}{2h} = f'(x_0) + \frac{h^2}{6} f'''(\xi), \quad x_0 - h < \xi < x_0 + h. \tag{1.2}$$

Its truncation error scales with h^2; therefore this second order method is more accurate than (1.1) for small values of h.

1.4 Goals for a good algorithm

1.4.1 Accuracy

All numerical solutions involve some error. An important goal for any algorithm is that the error should be small. However, it is equally important to have bounds on any error so that we have confidence in any answers. How accurate an answer needs to be depends upon the application. For example, an error of 0.1% may be more than enough in many cases but might be hopelessly inaccurate for landing a probe on a planet.

1.4.2 Efficiency

An efficient algorithm is one that minimises the amount of computation time, which usually means the number of arithmetic calculations. Whether this is important depends on the size of the calculation. If a calculation only takes a few seconds of minutes, it isn't worth spending a lot of time making it more efficient. However, a very large calculation can take weeks to run, so efficiency becomes important, especially if it needs to be run many times.

1.4.3 Stability and robustness

As we have seen, all calculations involve round-off errors. Although the error in each calculation is small, it is possible for the errors to multiply exponentially with each calculation so that the error becomes as large as the answer. When this happens, the methods is described as being unstable.

A robust algorithm is one that is stable for a wide range of problems.

Chapter 2

Solving nonlinear equations in one variable

Contents

In this section we shall be concerned with finding a root of the equation

$$f(x) = 0, \tag{2.1}$$

where f is a nonlinear function of x — if it is linear ($ax + b = 0$) we can solve trivially. We shall only be concerned with finding real roots of (2.1) though, of course, there is no guarantee that all, or indeed any, of the roots have to be real.

For example, we might wish to solve equations of the form

$$\begin{aligned} x - 2^{-x} &= 0, \\ \mathrm{e}^x - x^2 + 3x - 2 &= 0, \\ x\cos x + 2x^2 + 3x - 2 &= 0. \end{aligned}$$

None of these equations have solutions that can be written in a nice closed form, and so numerical approach is the only way.

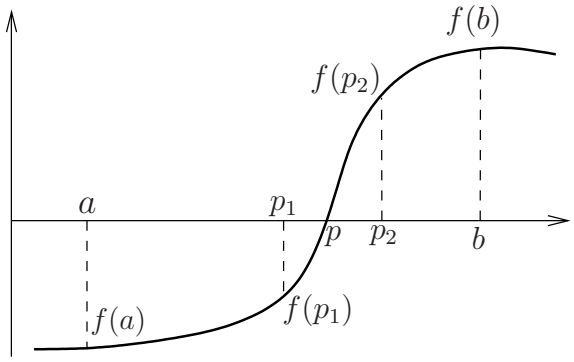
In this section, we shall describe iterative methods, i.e. methods that generate successive approximations to the exact solution, by applying a numerical algorithm to the previous approximation.

2.1 Bisection method

Suppose $f(x)$ is a continuous function on some interval $[a, b]$ with $f(a)$ and $f(b)$ of opposite sign[1]. Then, by the intermediate value theorem (IVT), $\exists p \in (a, b)$ with $f(p) = 0$. (For

[1]The product of two non-zero numbers $f(a)f(b) < 0$ if $f(a)$ and $f(b)$ are of opposite signs; $f(a)f(b) > 0$ if $f(a)$ has the same sign as $f(b)$.

simplicity, let's assume there exists only one root in (a, b).)



Bisection gives a simple and robust method for bracketing the root p. The algorithm works as follows:

Take the midpoint $p_1 = (a + b)/2$ as a first guess and calculate $f(p_1)$. If $f(p_1)$ and $f(b)$ are of opposite sign then the root lies in the interval $[p_1, b]$. (Conversely, if $f(p_1)$ has the opposite sign to $f(a)$ then the root lies in the interval $[a, p_1]$.)

This procedure can be repeated iteratively, e.g. taking a second guess $p_2 = (p_1 + b)/2$ from the example above.

At each iteration we halve the size of the interval $[a_n, b_n]$ that contains the root p and after n iterations of the procedure we have reduced the uncertainty in p down to,

$$E_n = |p - p_n| \leq \frac{b - a}{2^n}. \tag{2.2}$$

Example 2.1

Material covered in class. Please, see your lecture notes.

Advantages of the method :

i. provided the function is *continuous* on an interval $[a, b]$, with $f(a)f(b) < 0$, bisection is guaranteed to work (up to round-off error);

ii. the number of iterations needed to achieve a specific accuracy is known in advance.

Disadvantage of the method :

i. the method is slow to converge. (Reducing the error interval to 10^{-4} requires 16 iterations in the example 2.1);

ii. the errors in p_n and in $f(p_n)$ do not necessarily decrease between iterations. Note that, in the example 2.1, $p_2 = 1.5$ is closer to p (and $f(p_n)$ closer to 0) than the next 3 iterates. Also, no advantage is taken of intermediate good approximations.

2.2 Fixed point iteration

A fixed point of a function $g(x)$ is a value p such that $g(p) = p$.

Fixed point iteration procedure. Let g be a continuous function. If the sequence $p_n = g(p_{n-1})$ converges to p as $n \to \infty$ then $g(p) = p$. So, p is a stable fixed point of g.

Thus, provided p_0 is sufficiently close to p, we can use the simple iteration

$$p_n = g(p_{n-1}) \tag{2.3}$$

to find stable fixed points.

Root finding. A root finding problem, $f(x) = 0$, can be easily converted to a fixed point iteration by choosing, e.g., $g(x) = x - f(x)$, $g(x) = x + 3f(x)$, or $g(x) = \sqrt{x^2 + f(x)}$, etc.

It is important to note that there exists an infinite number of such functions g but the precise form of the g we choose will prove crucial for the convergence of the fixed point iteration.

Example 2.2

Material covered in class. Please, see your lecture notes.

Example 2.3

Material covered in class. Please, see your lecture notes.

Consider an iterate close to p, $p_n = p+\varepsilon$, say. Then, $p_{n+1} = g(p_n) = g(p+\varepsilon) = g(p)+\varepsilon g'(p)+O(\varepsilon^2) = p+\varepsilon g'(p)+O(\varepsilon^2)$ (using Taylor series). Thus $|p-p_n| = |\varepsilon|$ and $|p-p_{n+1}| \sim |\varepsilon||g'(p)|$, so if $|g'(p)| > 1$ then p_{n+1} is further away from p than p_n; there is no convergence.

Theorem 2.1 (Fixed point theorem)
If $g \in C[a,b]$ (i.e. is a continuous function on the interval $[a,b]$) and, $\forall x \in [a,b]$, $g(x) \in [a,b]$ then g has a fixed point in $[a,b]$ (existence).

If, in addition, $g'(x)$ exits on (a,b) and there exists a positive constant $K < 1$ such that, $\forall x \in (a,b)$, $|g'(x)| \leq K$ then

i. the fixed point is unique (uniqueness),

ii. for any $p_0 \in [a,b]$ the sequence $p_n = g(p_{n-1})$ converges to this unique fixed point $p \in [a,b]$ (stability).

Proof. The proof is in three parts. First, we prove that a fixed point exists. Second we prove that it is unique and third that the sequence must converge.

Existence. If $g(a) = a$ or $g(b) = b$ then a or b, respectively, is a fixed points. If not, then $g(a) > a$ and $g(b) < b$.

Define $h(x) = g(x) - x$. Then $h \in C[a,b]$ with $h(a) = g(a) - a > 0$ and $h(b) = g(b) - b < 0$. Therefore, the intermediate value theorem implies that there exists $p \in (a,b)$ such that $h(p) = 0$ i.e. $g(p) = p$, so p is a fixed point.

Uniqueness. Suppose that p and q are two distinct fixed points, i.e. $h(p) = h(q) = 0$, with $p < q$. Then by Rolle's theorem, $\exists c \in (p,q)$ such that $h'(c) = 0$.

But $h'(x) = g'(x) - 1 \leq K - 1 < 0$ since $g'(x) \leq K < 1$. This is a contradiction. Since the only assumption we have made is that $p \neq q$, then it follows that this assumption must be wrong. Thus, $p = q$ i.e. the fixed point is unique.

Stability. $|p_n - p| = |g(p_{n-1}) - g(p)|$. But the mean value theorem implies that $\exists \xi$ in the interval between p_{n-1} and p, subset of (a,b), such that $|g(p_{n-1}) - g(p)| = |g'(\xi)||(p_{n-1} - p)| \leq K|(p_{n-1} - p)|$.

Therefore, $|p_n - p| \leq K|(p_{n-1} - p)| \leq K^2|(p_{n-2} - p)| \ldots \leq K^n|(p_0 - p)|$, with $K < 1$. Hence, $|p_n - p| \leq K^n|(p_0 - p)| \to 0$ when $n \to \infty$ and the sequence (p_n) converges to the unique fixed point p.

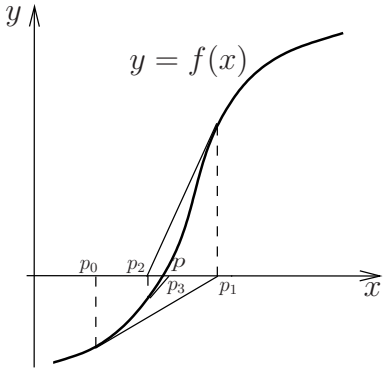
It is important to note that this theorem gives sufficient conditions for convergence, but they are not necessary.

In the example 2.2, when $g(x) = x - x^2/2 + 1$, $g'(x) = 1 - x$. So, $\forall x \in [1, 1.5]$, $|g'(x)| \leq 0.5$ and $g(x) \in [1.375, 1.5] \subset [1, 1.5]$ (subset). Hence, all the conditions for the fixed point theorem are satisfied and the iteration converges.

However, if $g(x) = x + 3x^2/2 - 3$ then $g'(x) = 1 + 3x > 1$ when $x > 0$. So, the fixed point $x = \sqrt{2}$ is unstable.

2.3 Newton-Raphson method

There are various ways of deriving this iterative procedure for solving nonlinear equations $f(x) = 0$ (see appendix C), but the graphical method is particularly instructive.



Suppose we have an initial guess p_o where f and f' are known. Then, p_1, the intersection of the x-axis with the tangent to f in p_0, is an improved estimate of the root.

$$f'(p_0) = \frac{f(p_0)}{p_0 - p_1} \Rightarrow p_1 = p_0 - \frac{f(p_0)}{f'(p_0)}.$$

Repeating this procedure leads to the general iterative scheme:

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \quad n \geq 1. \tag{2.4}$$

Obviously it is vital to have $f'(p_{n-1}) \neq 0$.

Relation with fixed point iteration. The Newton-Raphson method is a particular case of the fixed point iteration algorithm, with the iterative map $p_n = g(p_{n-1})$, where the mapping function $g(x) = x - f(x)/f'(x)$.

Example 2.4

Material covered in class. Please, see your lecture notes.

Advantages of Newton's method.

i. It is fast. (We shall quantify this later.)

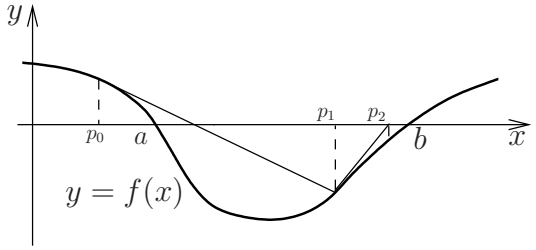
ii. It can be extended to multidimensional problem of finding, e.g. $f_1(x, y) = f_2(x, y) = 0$ for which bisection method cannot be used.

Disadvantages of Newton's method.

i. It requires evaluating the derivative, $f'(x)$, at each iteration.

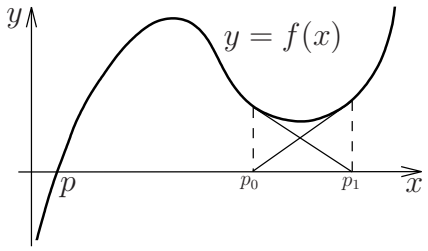
ii. It can fail if the initial guess is not sufficiently close to the solution, particularly if $f'(x) = 0$.

For instance, in the example 2.4 convergence is guaranteed. (Iterations converge to $+\sqrt{2}$ if $p_0 > 0$ and to $-\sqrt{2}$ if $p_0 < 0$.) However if f has the form



even though the initial guess p_0 is closest to the root $x = a$, (p_n) will converge to $x = b$.

Even more troublesome, a function of the form



has a root at $x = p$ but with the initial guess p_0 as shown, the iterations get caught in an endless loop, never converging.

2.4 Secant method

A solution to avoid the calculation of the derivative for Newton-Raphson method is to use the last two iterates to evaluate $f'(x)$.

By definition,

$$f'(p_{n-1}) = \lim_{x \to p_{n-1}} \frac{f(x) - f(p_{n-1})}{x - p_{n-1}},$$

but letting $x = p_{n-2}$ leads to the approximation

$$f'(p_{n-1}) \approx \frac{f(p_{n-2}) - f(p_{n-1})}{p_{n-2} - p_{n-1}}.$$

Rather than (2.4), the iteration scheme for the secant method is

$$p_n = p_{n-1} - f(p_{n-1}) \frac{p_{n-1} - p_{n-2}}{f(p_{n-1}) - f(p_{n-2})}. \tag{2.5}$$

Unlike Newton-Raphson, this method is a two-point method. (Iteration p_n uses both, p_{n-1} and p_{n-2}.) So, initially, two guesses are needed, p_0 and p_1.

Example 2.5

Material covered in class. Please, see your lecture notes.

Secant method is slightly slower than Newton's method, but it does not require the evaluation of a derivative. It does need two initial points but these do not have to straddle the root.

2.5 Rates of convergence

We mentioned, that the bisection method converges slowly, whilst, when the Newton-Raphson algorithm converges, it is fast. In this section we shall quantify the convergence rate of iteration schemes.

Suppose a numerical method producing a sequence of iterations (p_n) that converges to p. The method has order of convergence α if $|p_{n+1} - p| \sim K|p_n - p|^\alpha$ (i.e. $|p_{n+1} - p|$ is asymptotic to $K|p_n - p|^\alpha$), for some $K > 0$. Or equivalently if

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} = K.$$

When $\alpha = 1$ the convergence is linear, and when $\alpha = 2$ it is quadratic. If the error of an iterative algorithm is $O(\varepsilon)$ at iteration n then, for a linear methods, it remains $O(\varepsilon)$ at iteration $n + 1$, but for a quadratic method the error becomes $O(\varepsilon^2)$. Thus, higher order methods generally converge more rapidly than lower order ones.

2.5.1 Fixed point iteration

Let us consider a fixed point iteration $p_{n+1} = g(p_n)$ producing a sequence of iterations $(p_n) \to p$ as $n \to \infty$, such that $p = g(p)$ (since g is continuous).

Expand $g(p_n)$ as a Taylor series in powers of $(p_n - p)$. For some ξ_n in the interval between p and p_n,

$$g(p_n) = g(p) + (p_n - p)g'(\xi_n) \Rightarrow p_{n+1} = p + (p_n - p)g'(\xi_n).$$

Thus, $|p_{n+1} - p| = |g'(\xi_n)||p_n - p|$, and

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|} = |g'(p)| \Leftrightarrow |p_{n+1} - p| \sim |g'(p)||p_n - p|. \tag{2.6}$$

So, for $0 < |g'(p)| < 1$ fixed point iteration is linearly convergent.

2.5.2 Newton-Raphson

Equation (2.6) shows that the convergence of fixed point iterations is best when $g'(p)$ is as small as possible, and preferably zero. Newton-Raphson, which is a fixed point iteration method with the mapping function $g(x) = x - f(x)/f'(x)$, achieves this.

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

Thus, provided $f'(p) \neq 0$ (i.e. p is a simple root, i.e. of multiplicity one), $g'(p) = 0$ since, by definition, $f(p) = 0$.

Expand $g(p_n)$ as a Taylor series in powers of $(p_n - p)$, up to second order. For some ξ_n in the interval between p and p_n,

$$g(p_n) = g(p) + (p_n - p)g'(p) + \frac{(p_n - p)^2}{2}g''(\xi_n) \Rightarrow p_{n+1} = p + \frac{(p_n - p)^2}{2}g''(\xi_n),$$

since $p_{n+1} = g(p_n)$, $g(p) = p$ and $g'(p) = 0$. Thus,

$$|p_{n+1} - p| = \frac{|g''(\xi_n)|}{2}|p_n - p|^2$$

implies

$$\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|^2} = \frac{|g''(p)|}{2} \Leftrightarrow |p_{n+1}-p| \sim \frac{|g''(p)|}{2}|p_n-p|^2, \tag{2.7}$$

so the convergence of Newton's method is quadratic. (Note that $g''(p) = f''(p)/f'(p)$.)

However, in the case when $g'(p) \neq 0$ (i.e. if $f'(p) = 0$: p is a double root),

$$g(p_n) = g(p) + (p_n - p)g'(\xi_n) \Leftrightarrow p_{n+1} - p = (p_n - p)g'(\xi_n).$$

Thus,

$$\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|} = |g'(p)|,$$

So, if $g'(p) \neq 0$ the convergence of Newton's methods is not quadratic but linear (i.e. the same as the general form of fixed point iteration).

2.5.3 Other methods

Bisection. At each step, the size of the interval that contains the root is halved, so $\max(E_n) = \max(E_{n-1})/2$, but the error does not necessarily decrease monotonically. However, if we regard the upper bounds of the errors as an estimate of the error, then bisection is linearly convergent.

Secant. This method requires two steps and is harder to analyse. However, it can be shown in a strict mathematical sens (see appendix D) that $|p_{n+1} - p| = K|p_n - p||p_{n-1} - p|$, giving an order of convergence of $(1 + \sqrt{5})/2 \approx 1.62$ (golden ratio). The convergence is super-linear, i.e. better than linear but poorer than quadratic.

2.6 Evaluation of the error

Generally, the exact value of the root of a function (or the exact value of a fixed point) is unknown. So, it is impossible the evaluate the error of iterative methods, $E_n = |p - p_n|$, but we can find an upper bound on the error using the difference between two iterates.

A sequence (p_n) is called contractive if there exists a positive constant $K < 1$ such that

$$|p_{n+2} - p_{n+1}| \leq K|p_{n+1} - p_n|, \forall n \in \mathbb{N}.$$

Theorem 2.2
If (p_n) is contractive with constant K then,

i. (p_n) is convergent,

$$\lim_{n\to\infty} p_n = p. \tag{2.8}$$

ii. $|p - p_n|$ is bounded above,

$$|p - p_n| \leq \frac{K}{1-K}|p_n - p_{n-1}|. \tag{2.9}$$

Hence, when $K < 1/2$, $|p_n - p_{n-1}|$ provides an the upper bound on $E_n = |p - p_n|$, the error at iteration n.

Chapter 3

Interpolation

Contents

Suppose we have some table of discrete data, the population of a country at ten yearly intervals, say, and we need to evaluate the population at a date where no data are available. The estimation of the unknown data, using some or all of the available data, is a process known as interpolation.

Similarly, interpolation allows us to approximate a complicated or unknown function, f, with a simple function that agrees with f at a finite number of points, x_k.

i. If f is very computationally expensive to calculate, approximating f in some calculations by a simpler interpolating function reduces this cost. This was a common practice before calculators and computers became widely available.

ii. Exact calculations (e.g. derivation, quadrature) may become possible if the interpolating function is used instead of the complicated or unknown function f. This is a building block for advanced numerical methods (e.g. finite element and spectral-collocation methods).

In this section we shall consider the approximation of a function by polynomials or piecewise polynomial functions (splines) only.

3.1 Global polynomial interpolation

Fit a polynomial of degree $n-1$,

$$P(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0$$

through n known points $(x_k, f_k), k = 1, \ldots, n$. Hence, the interpolating polynomial satisfies $P(x_k) = f_k$. (The coordinates x_k are not necessarily equally spaced.)

3.1.1 Lagrange polynomials

A practical way to construct the polynomial $P(x)$ is to use the basis formed by Lagrange polynomials.

Let us define the Lagrange polynomials of degree $n-1$, $L_k(x)$ $(k = 1, \ldots, n)$, such that

$$L_k(x) = \begin{cases} 1 \text{ at } x = x_k, \\ 0 \text{ at } x = x_i \text{ for } i \neq k, \end{cases} \quad \text{i.e. } L_k(x_i) = \delta_{ik} \text{ (Kronecker delta).}$$

In other words, $L_k(x)$ is equal to 1 at the point x_k and 0 at all the other given points.

For $n = 2$, (i.e. $\{x_1, x_2\}$ given), the two Lagrange polynomials $L_1(x)$ and $L_2(x)$ satisfy

$$\begin{aligned} L_1(x_1) &= 1, \quad L_2(x_1) = 0, \\ L_1(x_2) &= 0, \quad L_2(x_2) = 1, \end{aligned}$$

from which we obtain the linear functions (i.e. polynomials of degree 1),

$$L_1(x) = \frac{x - x_2}{x_1 - x_2}, \quad L_2(x) = \frac{x - x_1}{x_2 - x_1}.$$

For $n > 2$, $L_1(x_1) = 1$ and $L_1(x_k) = 0$ for $k = 2, \ldots, n$. From above we see that

$$\frac{x - x_2}{x_1 - x_2} = \begin{cases} 1 \text{ at } x = x_1, \\ 0 \text{ at } x = x_2, \end{cases} \quad \text{while} \quad \frac{x - x_3}{x_1 - x_3} = \begin{cases} 1 \text{ at } x = x_1, \\ 0 \text{ at } x = x_3, \end{cases} \quad \text{etc.}$$

Thus, by multiplying $n-1$ such factors together, we obtain the Lagrange polynomial of degree $n-1$,

$$L_1(x) = \left(\frac{x - x_2}{x_1 - x_2}\right) \times \left(\frac{x - x_3}{x_1 - x_3}\right) \times \ldots \times \left(\frac{x - x_n}{x_1 - x_n}\right),$$

written in compact form as

$$L_1(x) = \prod_{i=2}^{n} \left(\frac{x - x_i}{x_1 - x_i}\right).$$

Similarly,

$$L_k(x) = \prod_{\substack{i=1 \\ i \neq k}}^{n} \left(\frac{x - x_i}{x_k - x_i}\right), \quad k = 1, \ldots, n. \tag{3.1}$$

The n Lagrange polynomials $L_k(x)$ of degree $n-1$ form a complete set of independent polynomials, so $\{L_1(x), L_2(x), \ldots, L_n(x)\}$ forms a basis of polynomials of degree $n-1$.

3.1.2 Lagrange form of the interpolating polynomial

Having obtained $L_k(x)$, we can write the Lagrange form of the polynomial interpolating the set of points $\{(x_k, f_k) \mid k = 1, \ldots, n\}$,

$$P(x) = \sum_{k=1}^{n} f_k L_k(x) = \sum_{k=1}^{n} f_k \prod_{\substack{i=1 \\ i \neq k}}^{n} \left(\frac{x - x_i}{x_k - x_i}\right). \tag{3.2}$$

(It is the Lagrange interpolating polynomial of degree $n-1$ satisfying $P(x_k) = f_k$.)

For $n = 2$,

$$\begin{aligned} P(x) &= f_1 L_1(x) + f_2 L_2(x), \\ &= f_1 \left(\frac{x - x_2}{x_1 - x_2} \right) + f_2 \left(\frac{x - x_1}{x_2 - x_1} \right) \end{aligned}$$

which rearranges to

$$\begin{aligned} P(x) &= \frac{f_1(x_2 - x_1) + (f_2 - f_1)(x - x_1)}{x_2 - x_1} \\ &= f_1 + \left(\frac{f_2 - f_1}{x_2 - x_1} \right) (x - x_1). \end{aligned} \tag{3.3}$$

This is the equation of a straight line between the two points (x_1, f_1) and (x_2, f_2) (linear interpolation).

Example 3.1

Material covered in class. Please, see your lecture notes.

3.2 Properties of global polynomial interpolation

3.2.1 Uniqueness

The interpolating polynomial is unique in that there is only one polynomial of degree at most $n - 1$, P, that satisfies $P(x_k) = f_k$ for n distinct points (x_k, f_k).

(This seems intuitively true, since we have n equations,

$$P(x_k) = \sum_{i=0}^{n-1} a_i x_k^i = f_k, \quad k = 1, \ldots, n,$$

with n unknowns, a_i, the coefficients of the polynomial P.)

Proof.

Material covered in class. Please, see your lecture notes.

3.2.2 The error term in global polynomial interpolation

In the section, interpolation is used to approximate a known function (not to interpolate tabulated data — in which case error cannot be evaluated).

If $f \in C^n[a, b]$ (i.e. f is a function n times differentiable in $[a, b]$ with continuous derivatives) then, $\forall x \in [a, b]$, there exists $\xi \in (a, b)$, that depends on x, such that

$$f(x) - P(x) = \frac{f^{(n)}(\xi)}{n!} \prod_{k=1}^{n} (x - x_k). \tag{3.4}$$

Proof. This result is trivially true for $x = x_i$ since both sides are zero.

For $x \neq x_i$, let us define

$$c(x) = \frac{f(x) - P(x)}{\prod_{k=1}^{n}(x - x_k)},$$

so that

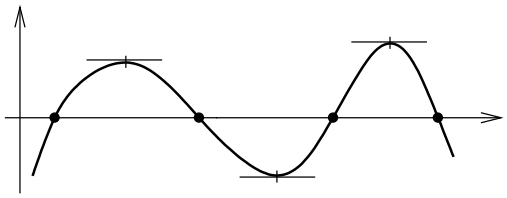
$$f(x) = P(x) + c(x)\prod_{k=1}^{n}(x - x_k).$$

Let us now define the function

$$w(t) = f(t) - P(t) - c(x)\prod_{k=1}^{n}(t - x_k),$$

which satisfies $w(t) = 0$ for $t = x$ and $t = x_1, \ldots, x_n$. So $w(t)$ has $n+1$ distinct roots in $[a, b]$.

By Rolle's theorem, there must be at least one root of $w(t)'$ between two successive roots of $w(t)$. So, $w'(t)$ has at least n distinct roots in (a, b).



Apply Rolle's theorem to derivatives of increasing order successively, to show that $w^{(n)}(t)$ must have at least one root in (a, b). Let us call this point ξ: $w^{(n)}(\xi) = 0$.

The n^{th} derivative of $w(t)$ is

$$w^{(n)}(t) = f^{(n)}(t) - P^{(n)}(t) - c(x)n!$$

but P is a polynomial of degree $n - 1$, so $P^{(n)} \equiv 0$. Let $t = \xi$,

$$f^{(n)}(\xi) - c(x)n! = 0 \quad \Rightarrow \quad c(x) = \frac{f^{(n)}(\xi)}{n!}.$$

So,

$$f(x) - P(x) = \frac{f^{(n)}(\xi)}{n!}\prod_{k=1}^{n}(x - x_k).$$

3.2.3 Accuracy of global polynomial interpolation

Using Equation (3.4) we can obtain bounds on the accuracy of Lagrange interpolation, provided we can bound the n^{th} derivative, $f^{(n)}$.

Since the error term involves $f^{(n)}$, Lagrange interpolation is exact for polynomial of degree n-1 at most.

In the example of $\sin(3x)$ (see appendix E.1), $|f^{(n)}| \leq 3^n$ while, e.g., for the five points $\{x_1 = 1, x_2 = 1.3, x_3 = 1.6, x_4 = 1.9, x_5 = 2.2\}$ and $x = 1.5$,

$$\left|\prod_{i=1}^{n}(x - x_i)\right| = 0.5 \times 0.2 \times 0.1 \times 0.4 \times 0.7 = \frac{7}{2500} = 2.8 \times 10^{-3}.$$

So, the error in the approximation to $f(1.5)$ is

$$|f(1.5) - P(1.5)| \leq \frac{3^5}{5!} \times \frac{7}{2500} = \frac{567}{10^5} = 5.67 \times 10^{-3}.$$

The actual error is $|f(1.5) - P(1.5)| \approx 2 \times 10^{-4}$.

However, the function $f(x) = 1/(1+x^2)$ cannot be approximated accurately by an interpolation polynomial, *using equidistant points* on the interval $[-5, 5]$ (see appendix E.2).

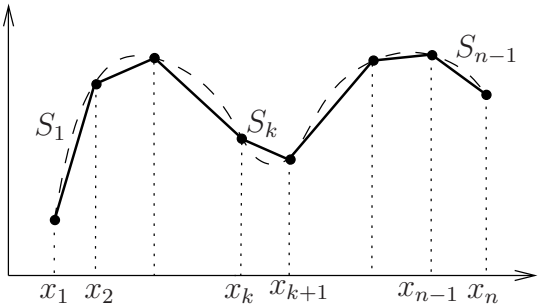
Although, f is differentiable, its derivatives grow with $n!$. ($|f^{(n)}(0)| = n!$, for n even.) So, $|f^{(n)}(\xi)/n!|$ does not converge to 0 while $\prod_{i=1}^{n}(x - x_i)$ increases (since interval width > 1).

3.3 Splines

Alternatively, to interpolate through n points (x_k, f_k) we can use piecewise polynomial functions called splines, i.e. $S(x) = \bigcup S_k(x)$ where the low degree polynomials $S_k(x)$ are defined on the intervals $[x_k, x_{k+1}]$, $k = 1, \ldots, n-1$.

3.3.1 Piecewise linear interpolation

The simplest spline is composed of linear functions of the form $S_k(x) = a_k x + b_k$, for $x \in [x_k, x_{k+1}]$ (i.e. a straight line between the two successive points x_k and x_{k+1}).



The coefficients a_k and b_k are determined by the conditions (i) $S_k(x_k) = f_k$ and (ii) $S_k(x_{k+1}) = f_{k+1}$, $k = 1, \ldots, n-1$. Thus,

$$S_k(x) = f_k + (x - x_k)\frac{f_{k+1} - f_k}{x_{k+1} - x_k}, \quad x \in [x_k, x_{k+1}], \quad k = 1, \ldots, n-1. \tag{3.5}$$

The interpolating function is continuous but it is not differentiable, i.e. not smooth, at the interior points ($S'_k(x_{k+1}) \neq S'_{k+1}(x_{k+1})$).

To retain the smoothness of Lagrange interpolation without producing large oscillations, higher order splines are needed.

3.3.2 Quadratic splines

Consider the spline composed of $n-1$ quadratic polynomials of the form $S_k(x) = a_k x^2 + b_k x + c_k$, for $x \in [x_k, x_{k+1}]$, $k = 1, \ldots, n-1$.

We need $3(n-1)$ equations to determine the $3(n-1)$ coefficients $\{a_k, b_k, c_k\}$. The conditions $S_k(x_k) = f_k$ and $S_k(x_{x+1}) = f_{k+1}$ provide $2(n-1)$ equations while the continuity of the derivative at the interior points, $S'_k(x_{k+1}) = S'_{k+1}(x_{k+1})$, $k = 1, \ldots, n-2$, provides $n-2$ extra equations.

In total we have $3(n-1)$ unknowns and $3(n-1)-1$ equations. So, only one degree of freedom is left, which is unsatisfactory if one needs to impose conditions on the derivatives of the interpolating function at *both* ends.

3.3.3 Cubic splines

In practical applications, cubic splines are preferred. Consider the spline composed of $n-1$ cubic polynomials of the form

$$S_k(x) = a_k(x-x_k)^3 + b_k(x-x_k)^2 + c_k(x-x_k) + d_k, \tag{3.6}$$

$$\text{so that, } S'_k(x) = 3a_k(x-x_k)^2 + 2b_k(x-x_k) + c_k, \tag{3.7}$$

$$S''_k(x) = 6a_k(x-x_k) + 2b_k. \tag{3.8}$$

for $x \in [x_k, x_{k+1}]$, $k = 1, \ldots, n-1$.

Properties. The interpolating function must pass through all the data points, be continuous, and have continuous derivative at the interior points. Furthermore, there are enough degrees of freedom to impose continuous curvature as well (continuity of second derivative).

$$S_k(x_k) = f_k, \quad k = 1, \ldots, n-1, \tag{3.9}$$

$$S_k(x_{k+1}) = f_{k+1}, \quad k = 1, \ldots, n-1, \tag{3.10}$$

$$S'_k(x_{k+1}) = S'_{k+1}(x_{k+1}), \quad k = 1, \ldots, n-2, \tag{3.11}$$

$$S''_k(x_{k+1}) = S''_{k+1}(x_{k+1}), \quad k = 1, \ldots, n-2. \tag{3.12}$$

We have $4(n-1)$ unknowns and $4(n-1)-2$ equations (i.e. conditions). Hence, we have the freedom to impose two extra conditions.

Let us define, $h_k = x_{k+1} - x_k$, the width of the k^{th} interval, and $C_k = S''_k(x_k)$ the curvature in x_k ($C_n = S''_{n-1}(x_n)$).

Next, we shall write all the equations in terms of h_k, f_k (given) and C_k (unknown) only.
From equation (3.8):

$$C_k = 2b_k \Leftrightarrow b_k = \frac{C_k}{2}, \quad k = 1, \ldots, n-1. \tag{3.13}$$

From equations (3.8) and (3.12):

$$6a_k h_k + C_k = C_{k+1} \Leftrightarrow a_k = \frac{1}{6}\frac{C_{k+1} - C_k}{h_k}, \quad k = 1, \ldots, n-1. \tag{3.14}$$

From equations (3.6) and (3.9):

$$d_k = f_k, \quad k = 1, \ldots, n-1. \tag{3.15}$$

From equations (3.6) and (3.10):

$$\begin{aligned} &a_k h_k^3 + b_k h_k^2 + c_k h_k + d_k = f_{k+1} \\ &\frac{1}{6}\frac{C_{k+1} - C_k}{h_k} h_k^3 + \frac{C_k}{2} h_k^2 + c_k h_k + f_k = f_{k+1} \\ &c_k = \frac{f_{k+1} - f_k}{h_k} - \frac{1}{6} h_k (C_{k+1} + 2C_k), \quad k = 1, \ldots, n-1. \end{aligned} \tag{3.16}$$

So far we have obtained expressions for the coefficients a_k, b_k, c_k and d_k in terms of h_k, C_k and f_k. All that remains is to match the slopes at the interior points.
From equations (3.7) and (3.11):

$$3a_k h_k^2 + 2b_k h_k + c_k = c_{k+1}, \quad k = 1, \ldots, n-2,$$

$$\frac{1}{2}\frac{C_{k+1} - C_k}{h_k} h_k^2 + C_k h_k + \frac{f_{k+1} - f_k}{h_k} - \frac{1}{6} h_k (C_{k+1} + 2C_k) = \frac{f_{k+2} - f_{k+1}}{h_{k+1}} - \frac{1}{6} h_{k+1}(C_{k+2} + 2C_{k+1}).$$

After some simple algebra, we obtain a system of $n-2$ equations to solve for C_k, $k = 1, \ldots, n-2$

$$h_k C_k + 2(h_k + h_{k+1}) C_{k+1} + h_{k+1} C_{k+2} = 6 \left(\frac{f_{k+2} - f_{k+1}}{h_{k+1}} - \frac{f_{k+1} - f_k}{h_k} \right). \tag{3.17}$$

So, the problem as been reduced from finding $4(n-1)$ coefficients a_k, b_k, c_k and d_k to finding n values of the curvature C_k.

We only have $n-2$ equations for C_k but we can obtain two additional equations by specifying end conditions on the curvature, i.e. conditions involving C_1 and C_n.

i. Natural or free boundaries: take $C_1 = C_n = 0$, i.e. the end cubic splines have no curvature at the end points.

ii. Clamped boundaries: fix the slopes at each end to specified values. (Note that, the slope at x_1, say, involves c_1 and thus relates to C_1 and C_2 through (3.16).)

iii. Periodic boundaries: take $C_1 = C_n$.

Equation (3.17) can be written in matrix form,

$$MC = F. \tag{3.18}$$

In the case of natural splines (i.e. for $C_1 = C_n = 0$), the matrix is non-singular and has a tri-diagonal structure. So, the solution for C_k is unique and can easily be obtained using rapid tri-diagonal matrix solvers.

$$M = \begin{pmatrix} 2(h_1+h_2) & h_2 & 0 & 0 & \ldots & 0 & 0 \\ h_2 & 2(h_2+h_3) & h_3 & 0 & \ldots & 0 & 0 \\ 0 & & & & & & 0 \\ 0 & \ldots & h_k & 2(h_k+h_{k+1}) & h_{k+1} & \ldots & 0 \\ 0 & & & & & & 0 \\ 0 & 0 & \ldots & 0 & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \\ 0 & 0 & \ldots & 0 & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix},$$

$$C = \begin{pmatrix} C_2 \\ C_3 \\ \vdots \\ C_{n-2} \\ C_{n-1} \end{pmatrix} \quad \text{and} \quad F = 6 \begin{pmatrix} \frac{f_3 - f_2}{h_2} - \frac{f_2 - f_1}{h_1} \\ \vdots \\ \frac{f_{k+2} - f_{k+1}}{h_{k+1}} - \frac{f_{k+1} - f_k}{h_k} \\ \vdots \\ \frac{f_n - f_{n-1}}{h_{n-1}} - \frac{f_{n-1} - f_{n-2}}{h_{n-2}} \end{pmatrix}.$$

So, equation (3.18) can be solved for C_k and the coefficients of the splines can be recovered from equations (3.13), (3.14), (3.15), (3.16).

Example 3.2

Material covered in class. Please, see your lecture notes.

Chapter 4

Numerical integration

Contents

4.1 Definite integrals

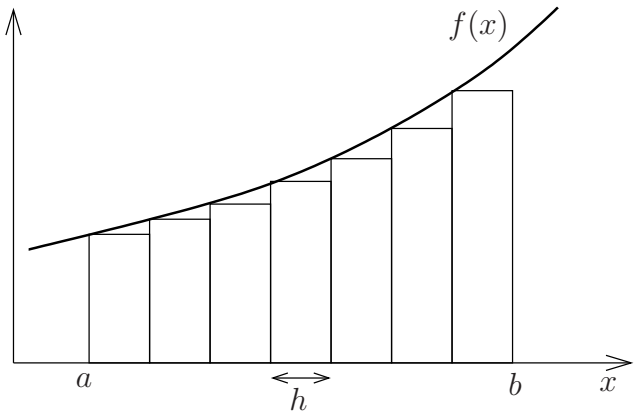
The definite integral of a function f on the interval $[a, b]$ is written as

$$\int_a^b f(x)\,\mathrm{d}x \tag{4.1}$$

and can be defined as the limit of the Riemann series

$$\int_a^b f(x)\,\mathrm{d}x = \lim_{h\to 0} \sum_{i=1}^{(b-a)/h} hf(a + (i-1)h),$$

i.e. as the area under the curve $(x, y = f(x))$.



Hence, numerical integration is often called quadrature (i.e. computation of an area under a curve).

The definite integral of a function exists even if there is no analytic antiderivative function $F(x)$ such that

$$\frac{\mathrm{d}}{\mathrm{d}x}F(x) = f(x).$$

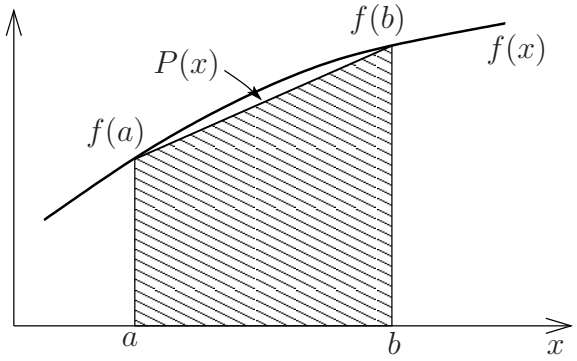
4.2 Closed Newton-Cotes formulae

The definite integral (4.1) can be approximated using the polynomial interpolating $f(x)$ through n equispaced points (x_k), leading to

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{k=1}^{n} w_k f(x_k) \quad \text{with} \quad x_k = a + (k-1)\frac{b-a}{n-1}.$$

4.2.1 Trapezium rule

This is the simplest numerical method for evaluating a definite integral. It calculates the area of the trapezium formed by approximating $f(x)$ using linear interpolation.



$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{2}\left(f(a)+f(b)\right). \tag{4.2}$$

Thus, the trapezium rule can be obtained by integrating the linear interpolation function

$$P(x) = f(a)\frac{x-b}{a-b} + f(b)\frac{x-a}{b-a} \quad \text{over the interval } [a,b].$$

(Check that $\displaystyle\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b P(x)\,\mathrm{d}x$ leads to (4.2).)

Measure of the error: From Equation (3.4), we know the error in the linear interpolation,

$$f(x) = P(x) + \frac{1}{2}f''(\xi(x))(x-a)(x-b), \quad \xi(x) \in [a,b].$$

Therefore,

$$\int_a^b f(x)\,\mathrm{d}x = \int_a^b P(x)\,\mathrm{d}x + \int_a^b \frac{1}{2}f''(\xi(x))(x-a)(x-b)\,\mathrm{d}x.$$

So, the error in the trapezium rule is equal to

$$E = \frac{1}{2}\int_a^b f''(\xi(x))(x-a)(x-b)\,\mathrm{d}x.$$

To evaluate this integral, we shall make use of the following theorem (generalisation of the Mean Value Theorem).

Theorem 4.1 (Weighted Mean Value for Integrals)
If $h \in C[a,b]$, if the integral of $g(x)$ exists on $[a,b]$, and if $g(x)$ does not change sign on $[a,b]$, then there exists $c \in [a,b]$ such that

$$\int_a^b h(x)g(x)\,\mathrm{d}x = h(c)\int_a^b g(x)\,\mathrm{d}x.$$

Hence,

$$\frac{1}{2}\int_a^b f''(\xi(x))(x-a)(x-b)\,\mathrm{d}x = \frac{f''(c)}{2}\int_a^b (x-a)(x-b)\,\mathrm{d}x$$

since $(x-a)(x-b) \leq 0$ on the interval $[a,b]$.

$$\begin{aligned} E &= \frac{1}{2}f''(c)\int_0^{b-a} u(u-(b-a))\,\mathrm{d}u \quad \text{with } u = x-a, \\ &= \frac{1}{2}f''(c)\left[\frac{u^3}{3} - \frac{u^2}{2}(b-a)\right]_0^{b-a} = -f''(c)\frac{(b-a)^3}{12}. \end{aligned}$$

Thus, the error in the trapezium method scales with the cube of the size on the interval. (Note that this method is exact for linear functions.)

We can generate other integration rules by using higher order Lagrange polynomials.

$$\begin{aligned} \int_a^b f(x)\,\mathrm{d}x \approx \int_a^b P(x)\,\mathrm{d}x &= \int_a^b \sum_{k=1}^n f(x_k)L_k(x)\,\mathrm{d}x \\ &= \sum_{k=1}^n f(x_k)\int_a^b L_k(x)\,\mathrm{d}x = \sum_{k=1}^n w_k f(x_k), \end{aligned}$$

where $w_k = \displaystyle\int_a^b L_k(x)\,\mathrm{d}x$ are called weights.

The error is given by $\displaystyle\int_a^b \frac{f^{(n)}(\xi(x))}{n!}\prod_{k=1}^n (x-x_k)\,\mathrm{d}x$.

This method provides a means of finding the weights and error terms. However, their evaluation becomes more and more complex with increasing n. (Theorem 4.1 cannot be used to evaluate the error since $\displaystyle\prod_{k=1}^n (x-x_k)$ changes sign for $n \geq 3$.)

4.2.2 Method of undetermined coefficients

Alternative way of evaluating Newton-Cotes integration formulae. To illustrate this method, let us derive the trapezium rule again.

Trapezium rule

We wish to find an approximation of the form

$$\int_a^b f(x)\,\mathrm{d}x \approx w_1 f(a) + w_2 f(b).$$

Since, we have two unknown parameters, w_1 and w_2, we can make this formulae exact for linear functions (i.e. functions are of the form $\alpha x + \beta$). So, the error term must be of the form $Kf''(\xi)$ for some $\xi \in (a,b)$.

Hence, we seek a quadrature rule of the form

$$\int_a^b f(x)\,\mathrm{d}x = w_1 f(a) + w_2 f(b) + K f''(\xi), \quad \xi \in (a,b).$$

To find w_1 and w_2 we need to ensure that the formula is exact for all linear functions. However, using the principle of superposition, it is sufficient to make this work for any two independent linear polynomials (i.e. polynomials of degree one at most).
E.g. choose the two independent polynomials $f \equiv 1$ and $f \equiv x$.

$$\text{For } f \equiv 1 : \int_a^b \mathrm{d}x = w_1 + w_2 = b - a;$$
$$\text{for } f \equiv x : \int_a^b x\,\mathrm{d}x = a w_1 + b w_2 = \frac{b^2 - a^2}{2}.$$

Solving these simultaneous equations gives

$$w_1 = w_2 = \frac{b-a}{2}.$$

To find the value of K, we can use any quadratic function so that $f''(\xi)$ is a non-zero constant, *independent* of the unknown ξ. E.g. take $f \equiv x^2$,

$$\begin{aligned}
\int_a^b x^2\,\mathrm{d}x = \frac{b^3 - a^3}{3} &= \frac{b-a}{2}\left(a^2 + b^2\right) + 2K \quad (f'' = 2),\\
\Rightarrow 2K &= \frac{b^3 - a^3}{3} - \frac{b-a}{2}\left(a^2 + b^2\right),\\
&= (b-a)\left[\frac{1}{3}\left(a^2 + ab + b^2\right) - \frac{1}{2}\left(a^2 + b^2\right)\right],\\
&= (b-a)\left[-\frac{1}{6}\left(a^2 - 2ab + b^2\right)\right] = -\frac{(b-a)^3}{6},\\
\Rightarrow K &= -\frac{(b-a)^3}{12}.
\end{aligned}$$

Hence,

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2}\left(f(a) + f(b)\right) - \frac{(b-a)^3}{12} f''(\xi), \quad \xi \in (a,b), \tag{4.3}$$

in agreement with § 4.2.1 (but easier to derive).

Simpson's rule

Three points integration rule derived using the method of undetermined coefficients.
Suppose that we add a quadrature point at the middle of the interval $[a,b]$,

$$\int_a^b f(x)\,\mathrm{d}x \approx w_1 f(a) + w_2 f\left(\frac{a+b}{2}\right) + w_3 f(b).$$

To avoid algebra, substitute $x = \dfrac{a+b}{2} + u$ and define $h = \dfrac{b-a}{2}$ so that the integral becomes

$$\int_{-h}^{h} F(u)\,\mathrm{d}u \approx w_1 F(-h) + w_2 F(0) + w_3 F(h), \quad \text{with } F(u) = f(x).$$

Since we have three unknowns, we can make this formula exact for all quadratic functions; so, let us use, e.g., $F \equiv 1$, $F \equiv u$ and $F \equiv u^2$.

$$F \equiv 1 \Rightarrow \int_{-h}^{h} \mathrm{d}u = 2h = w_1 + w_2 + w_3;$$

$$F \equiv u \Rightarrow \int_{-h}^{h} u\,\mathrm{d}u = 0 = -hw_1 + hw_3 \Rightarrow w_1 = w_3;$$

$$F \equiv u^2 \Rightarrow \int_{-h}^{h} u^2\,\mathrm{d}u = \frac{2}{3}h^3 = h^2 w_1 + h^2 w_3 \Rightarrow w_1 = w_3 = \frac{h}{3};$$

$$w_2 = 2h - w_1 - w_3 = 2h - \frac{2}{3}h = \frac{4}{3}h.$$

Hence we obtain the approximation

$$\int_{-h}^{h} F(u)\,\mathrm{d}u \approx \frac{h}{3}F(-h) + \frac{4h}{3}F(0) + \frac{h}{3}F(h),$$

which translates into

$$\int_{a}^{b} f(x)\,\mathrm{d}x \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$

This is called Simpson's rule.

As this formula is exact for all quadratic functions, we expect the error to be of the form $KF'''(\xi)$.

However, if we examine $F \equiv u^3$, the integral $\displaystyle\int_{-h}^{h} u^3\,\mathrm{d}u = 0$ and Simpson's rule gives

$$\frac{h}{3}\left[(-h)^3 + 4 \times 0 + h^3\right] = 0 \Rightarrow K \equiv 0.$$

Therefore Simpson's rule is exact for cubic functions as well. Consequently, we try an error term of the form $KF^{(\mathrm{iv})}(\xi)$, $\xi \in (-h, h)$.

To find the value of K use, e.g., $F(u) \equiv x^4$ (with $F^{(\mathrm{iv})}(\xi) = 4!$, independent of ξ).

$$\int_{-h}^{h} u^4 \mathrm{d}u = \frac{2h^5}{5} = \frac{h}{3}\left[(-h)^4 + h^4\right] + 4!K \Rightarrow 4!K = 2h^5\left[\frac{1}{5} - \frac{1}{3}\right] = -\frac{4h^5}{15},$$

$$\Rightarrow K = -\frac{h^5}{90}.$$

Simpson's rule is fifth-order accurate: the error varies as the fifth power of the width of the interval.

Simpson's rule is given by

$$\int_{a}^{b} f(x)\,\mathrm{d}x = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] - \frac{(b-a)^5}{2880}f^{(\mathrm{iv})}(\xi), \quad \xi \in (a,b). \tag{4.4}$$

Example 4.1

Material covered in class. Please, see your lecture notes.

Closed Newton-Cotes formula of higher order can be derived using more equispaced intermediate points ($n = 2$: trapezium; $n = 3$: Simpson).

As observed for Simpson's rule, the error for n odd is of order $(b-a)^{n+2}$ rather than $(b-a)^{n+1}$, due to symmetry.

4.3 Open Newton-Cotes formulae

Closed Newton-Cotes formulae such as (4.3) and (4.4) include the value of the function at the endpoints.

By contrast, *open Newton-Cotes* formulae are based on the interior points only.

Midpoint rule. For example, consider the open Newton-Cotes formula

$$\int_a^b f(x)\,\mathrm{d}x \approx w_1 f\left(\frac{a+b}{2}\right).$$

Again, we can substitute $u = x - \dfrac{a+b}{2}$ and $h = \dfrac{b-a}{2}$;

$$\int_{-h}^{h} F(u)\,\mathrm{d}u \approx w_1 F(0).$$

This formula must hold for constant functions. So, taking, e.g. $F \equiv 1 \Rightarrow 2h = w_1$. However, since for $F(u) \equiv u$, $\displaystyle\int_{-h}^{h} F(u)\,\mathrm{d}u = 0$, the quadrature rule is exact for linear functions as well. So, we look for an error of the form $KF''(\xi)$, $\xi \in (-h, h)$;

$$\int_{-h}^{h} F(u)\,\mathrm{d}u = 2hF(0) + KF''(\xi), \quad \xi \in (-h, h).$$

Substitute $F(u) = u^2$,

$$\frac{2}{3}h^3 = 0 + 2K \Rightarrow K = \frac{h^3}{3}.$$

$$\int_a^b f(x)\,\mathrm{d}x = (b-a) f\left(\frac{a+b}{2}\right) + \frac{(b-a)^3}{24} f''(\xi), \quad \xi \in (a, b). \tag{4.5}$$

Same order as trapezium and coefficient of the error smaller (halved).

Example 4.2

Material covered in class. Please, see your lecture notes.

4.4 Gauss quadrature

There is no necessity to use equispaced points. By choosing the quadrature points x_k appropriately we can derive n-points methods of order $2n+1$ (i.e. error varies as $(b-a)^{2n+1}$), exact for polynomials of degree $(2n-1)$.

These are called *Gauss formulae* and can give stunning accuracy. However, for $n \geq 2$, the values of x_k (roots of Legendre polynomials) and the weights w_k are non rational.

4.5 Composite methods

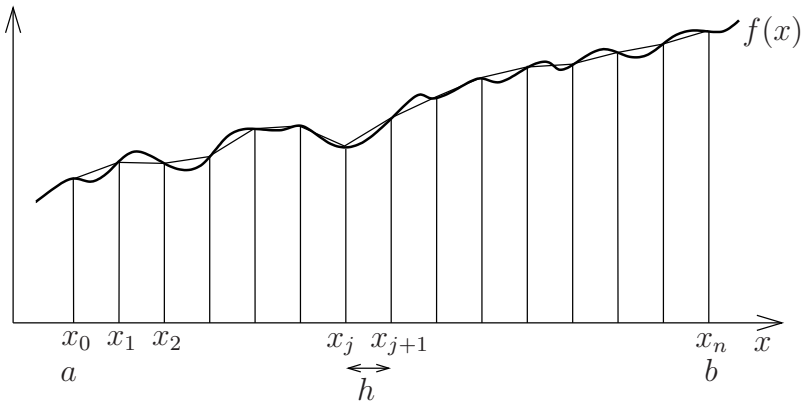
Disadvantage of higher order methods: Since higher order methods are based on Lagrange interpolation, they also suffer the same weaknesses.

While very accurate for functions with well-behaved higher order derivatives, they becomes inaccurate for functions with unbounded higher derivatives.

Therefore, it is often safer to use low order methods, such as trapezium and Simpson's rules. While these quadrature rules are not very accurate on large intervals (error varying as $(b-a)^3$ and as $(b-a)^5$), accurate approximations can be obtained by breaking the interval $[a,b]$ into n subintervals of equal width h.

4.5.1 Composite trapezium rule

Split the interval $[a,b]$ into n intervals of width $h = \dfrac{b-a}{n}$, i.e. consider $n+1$ equispaced points $x_j = a + jh$, $j = 0, \ldots, n$. The subinterval j is $[x_j, x_{j+1}]$, $j = 0, \ldots, n-1$.



$$\int_a^b f(x)\,\mathrm{d}x = \sum_{j=0}^{n-1} \int_{x_j}^{x_{j+1}} f(x)\,\mathrm{d}x = \sum_{j=0}^{n-1} \left\{ \frac{h}{2}\left[f(x_j) + f(x_{j+1})\right] - \frac{h^3}{12} f''(\xi_j) \right\} \quad \xi_j \in (x_j, x_{j+1}),$$

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x = {} & \frac{h}{2}[f(a) + f(x_1)] + \frac{h}{2}[f(x_1) + f(x_2)] + \ldots \\ & + \frac{h}{2}[f(x_{n-2}) + f(x_{n-1})] + \frac{h}{2}[f(x_{n-1}) + f(b)] \\ & - \frac{h^3}{12}\left[f''(\xi_0) + f''(\xi_1) + \ldots + f''(\xi_{n-1})\right].\end{aligned}$$

Error: since

$$\min_{[a,b]}(f''(x)) \le \min_j(f''(\xi_j)) \le K = \frac{1}{n}\sum_{j=0}^{n-1} f''(\xi_j) \le \max_j(f''(\xi_j)) \le \max_{[a,b]}(f''(x)),$$

and f'' is continuous, there exits $\xi \in (a,b)$ with $f''(\xi) = K$ (analogous to the intermediate value theorem). Hence, the total error is of the form

$$E = -\frac{nh^3}{12} f''(\xi) = -\frac{b-a}{12} h^2 f''(\xi) \quad \text{since } b - a = nh.$$

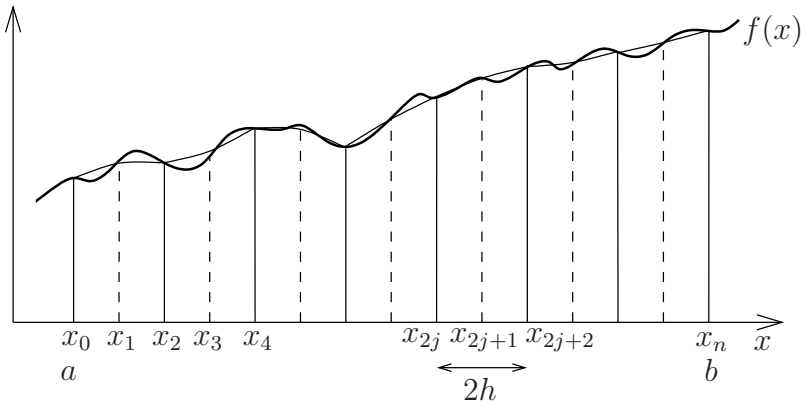
We can rewrite the composite trapezium rule as

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] - \frac{b-a}{12} h^2 f''(\xi), \quad \xi \in (a,b). \tag{4.6}$$

Composite trapezium rule has error of order h^2, compared to h^3 for each individual interval.

4.5.2 Composite Simpson's rule

Consider $n/2$ intervals $[x_{2j}, x_{2j+2}]$, $j = 0, \ldots, n/2-1$, of width $2h = \dfrac{b-a}{n/2}$ with equispaced quadrature points $x_j = a + jh$, $j = 0, \ldots, n$.



$$\int_a^b f(x)\,\mathrm{d}x = \sum_{j=0}^{n/2-1} \int_{x_{2j}}^{x_{2j+2}} f(x)\,\mathrm{d}x,$$

$$= \sum_{j=0}^{n/2-1} \left\{ \frac{h}{3}\left[f(x_{2j}) + 4f(x_{2j+1}) + f(x_{2j+2})\right] - \frac{h^5}{90} f^{(\mathrm{iv})}(\xi_j) \right\} \quad \xi_j \in [x_{2j}, x_{2j+2}],$$

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{3}[f(a) + 4f(x_1) + f(x_2)] + \frac{h}{3}[f(x_2) + 4f(x_3) + f(x_4)] + \ldots$$
$$+ \frac{h}{3}[f(x_{n-2}) + 4f(x_{n-1}) + f(b)] - \frac{h^5}{90}\left[f^{(\mathrm{iv})}(\xi_0) + f^{(\mathrm{iv})}(\xi_1) + \ldots + f^{(\mathrm{iv})}(\xi_{n/2-1})\right].$$

As for the composite trapezium rule, there exists $\xi \in (a, b)$ such that

$$\frac{n}{2} f^{(\mathrm{iv})}(\xi) = \sum_{j=0}^{n/2-1} f^{(\mathrm{iv})}(\xi_j).$$

Using also $h = \dfrac{b-a}{n}$, we can rewrite the composite Simpson's rule as

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{3}\left[f(a) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=0}^{n/2-1} f(x_{2j+1}) + f(b)\right]$$
$$- \frac{b-a}{180} h^4 f^{(\mathrm{iv})}(\xi), \quad \xi \in (a, b). \quad (4.7)$$

So, the composite Simpson's rule is 4^{th} order accurate.

Chapter 5

Ordinary differential equations

Contents

In this section we shall use the following notations: $Y(t)$ is the exact solution of an ODE and y_n is an approximation to the solution at time $t = t_n = nh$. We may also use $f_n \equiv f(t_n, y_n)$ for the right-hand side of the ODE.

5.1 Initial value problem

In this section, we shall consider numerical solution of first order IVPs (initial value problems) of the form

$$\frac{\mathrm{d}Y}{\mathrm{d}t} = f(t, Y) \quad \text{for } t_0 < t < t_{\max} \text{ with initial condition } Y(t_0) = \alpha. \tag{5.1}$$

Provided that $f \in C[t_0, t_{\max}]$ and that $\left|\frac{\partial f}{\partial Y}\right|$ is bounded for all Y, this problem has a unique solution.

The methods that we shall discuss can be easily extended to systems of 1^{st} order ODEs of the form

$$\begin{cases} \frac{\mathrm{d}Y_1}{\mathrm{d}t} &= f_1(t, Y_1, \ldots, Y_n), \quad Y_1 = \alpha_1 \text{ at } t = t_0, \\ &\vdots \\ \frac{\mathrm{d}Y_n}{\mathrm{d}t} &= f_n(t, Y_1, \ldots, Y_n), \quad Y_n = \alpha_n \text{ at } t = t_0. \end{cases}$$

Note that higher order IVPs can always be reduced to 1^{st} order systems. E.g.

$$\frac{\mathrm{d}^2Y}{\mathrm{d}t^2} + q(t)\frac{\mathrm{d}Y}{\mathrm{d}t} + r(t)Y = s(t)$$

can be rewritten as the system of 1st order ODEs

$$\begin{cases} \dfrac{\mathrm{d}Y}{\mathrm{d}t} = Z, \\ \dfrac{\mathrm{d}Z}{\mathrm{d}t} = -q(t)Z - r(t)Y + s(t). \end{cases}$$

Integral form of the solution: Integrate both sides of IVP (5.1) on $[t_0, t]$

$$\int_{t_0}^{t} \frac{\mathrm{d}Y}{\mathrm{d}\tau}\,\mathrm{d}\tau = \int_{Y(t_0)}^{Y(t)} \mathrm{d}Y = \int_{t_0}^{t} f(\tau, Y(\tau))\,\mathrm{d}\tau,$$
$$\Rightarrow Y(t) = Y(t_0) + \int_{t_0}^{t} f(\tau, Y(\tau))\,\mathrm{d}\tau = \alpha + \int_{t_0}^{t} f(\tau, Y(\tau))\,\mathrm{d}\tau.$$

Similarly, integrating on $[t, t+h]$:

$$Y(t+h) = Y(t) + \int_{t}^{t+h} f(\tau, Y(\tau))\,\mathrm{d}\tau. \tag{5.2}$$

5.2 Forward Euler's method

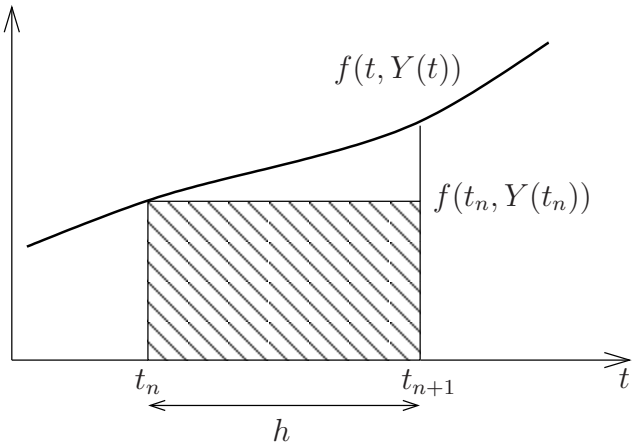
The simplest scheme for solving (5.1) is Euler's method.

5.2.1 Approximation

Discrete t: $t_n = t_0 + nh$ where h is the step-size.

$$\text{Equation (5.2)} \Rightarrow Y(t_{n+1}) = Y(t_n) + \int_{t_n}^{t_{n+1}} f(t, Y(t))\,\mathrm{d}t.$$

The Euler method consists in approximating $f(t, Y(t))$ by the constant $f(t_n, Y(t_n))$ in the integral (i.e. on the interval $[t_n, t_{n+1}]$).



So,

$$Y(t_{n+1}) \approx Y(t_n) + hf(t_n, Y(t_n)). \tag{5.3}$$

Hence, the Euler scheme

$$y_{n+1} = y_n + hf(t_n, y_n). \tag{5.4}$$

It provides an approximation to $Y(t_{n+1})$, using the approximation at the previous step.

Thus, the differential equation is replaced by a *difference equation*.

Example 5.1

$$\frac{\mathrm{d}Y}{\mathrm{d}t} = 1 - Y \quad \text{with } Y(0) = 0.$$

$$\begin{aligned} &\underline{\text{Analytic solution}}\text{: } Y(t) = 1 - \mathrm{e}^{-t}. \\ &\underline{\text{Euler's method}}\text{: } \quad y_{n+1} = y_n + h(1 - y_n), \quad y_0 = 0. \end{aligned} \tag{5.5}$$

The solution of the homogeneous difference equation $y_{n+1} = (1-h)y_n$ is $y_n = (1-h)^n y_0$. In addition, $y_{n+1} = y_n = 1$ is a particular solution of (5.5). So, the general solution of (5.5) is $y_n = 1 + A(1-h)^n$.
Use the initial condition to determine A: $y_0 = 1 + A = 0 \Rightarrow A = -1$. Hence, Euler's method gives the approximate solution

$$y_n = 1 - (1-h)^n.$$

Stability: The error in the solution is

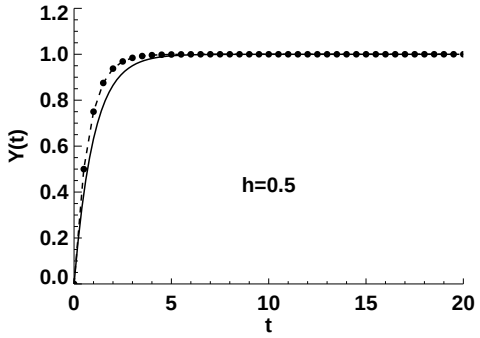
$$\begin{aligned} |Y(t_n) - y_n| &= \left|1 - \mathrm{e}^{-t_n} - (1 - (1-h)^n)\right|, \\ &= \left|(1-h)^n - \mathrm{e}^{-t_n}\right|. \end{aligned}$$

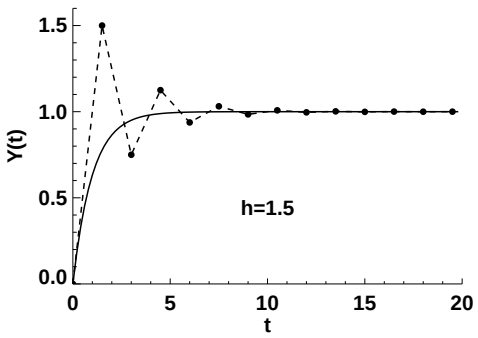
Provided $h < 2$, $(1-h)^n \to 0$ as $n \to \infty$; and since $\mathrm{e}^{-t_n} = \mathrm{e}^{-nh} \to 0$ as $n \to \infty$, the error $|Y(t_n) - y_n| \to 0$ as $n \to \infty$,

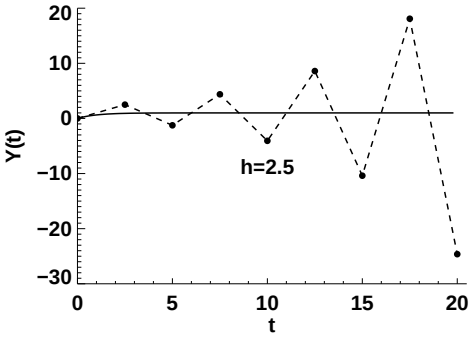
$$\Rightarrow \lim_{n\to\infty} y_n = \lim_{n\to\infty} Y(t_n) = \lim_{n\to\infty} (1 - \mathrm{e}^{-t_n}) = 1.$$

The Euler solution will tend to 1, the exact solution, as $n \to \infty$.

However, for $h > 2$, $(1-h)^n$ diverges implying that y_n diverges as $n \to \infty$.

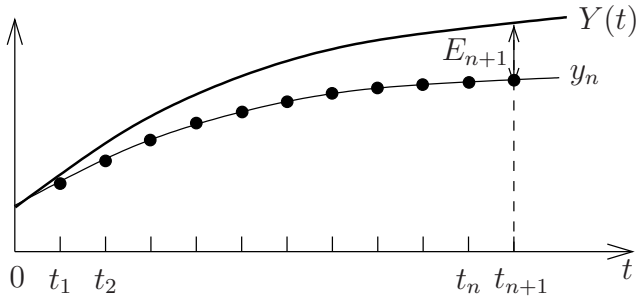






For this problem, the Euler scheme is stable for $h < 2$, but how accurate is it?

5.2.2 Error analysis



All numerical methods for solving IVPs use the results from previous steps in calculating the approximation to the solution for next steps.

Consequently the global error at t_{n+1} includes both the error made at step $(n+1)$ — *local truncation error* — and the error in y_n propagating from the previous steps.

Local truncation error

The local truncation error measures the error generated by replacing

$$Y(t_{n+1}) = Y(t_n) + \int_{t_n}^{t_{n+1}} f(t, Y(t))\,\mathrm{d}t$$

with

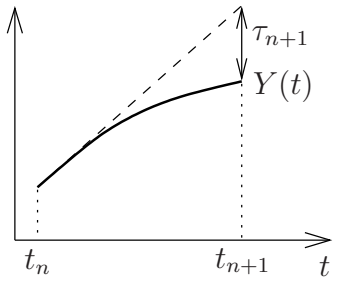
$$Y(t_{n+1}) \approx Y(t_n) + hf(t_n, Y(t_n)).$$

From Taylor's series we have

$$\begin{aligned} Y(t_{n+1}) &= Y(t_n) + hY'(t_n) + \frac{h^2}{2}Y''(\xi_{n+1}), \quad \text{for some } \xi_{n+1} \in [t_n, t_{n+1}], \\ &= Y(t_n) + hf(t_n, Y(t_n)) + \frac{h^2}{2}Y''(\xi_{n+1}). \end{aligned} \tag{5.6}$$

This shows that when approximating $Y(t_{n+1})$ using forward Euler, the local truncation error is of the form

$$\tau_{n+1} = \frac{h^2}{2}Y''(\xi_{n+1}), \quad \text{for some } \xi_{n+1} \in (t_n, t_{n+1}). \tag{5.7}$$



(Alternative derivation of the local truncation error:

$$\int_t^{t+h} g(\tau)\,\mathrm{d}\tau = hg(t) + Kg'(\xi) \quad \text{(exact for constant functions)}.$$

Let $g \equiv \tau \Rightarrow (t+h)^2/2 - t^2/2 = ht + K \Rightarrow K = h^2/2$.)

Global error

The global error includes both the local truncation error and the flow transported error.

Subtract Euler's scheme (5.4) from the Taylor expansion (5.6),

$$Y(t_{n+1}) - y_{n+1} = Y(t_n) - y_n + h\left[f(t_n, Y(t_n)) - f(t_n, y_n)\right] + \tau_{n+1}.$$

Since f is a continuous and differentiable function, the mean value theorem implies that there exists Z_n between y_n and $Y(t_n)$ such that

$$f(t_n, Y(t_n)) - f(t_n, y_n) = \left[Y(t_n) - y_n\right] \frac{\partial}{\partial Z} f(t_n, Z_n).$$

Hence,

$$\begin{aligned} Y(t_{n+1}) - y_{n+1} &= Y(t_n) - y_n + h\left[Y(t_n) - y_n\right] \frac{\partial}{\partial Z} f(t_n, Z_n) + \tau_{n+1}, \\ \Rightarrow E_{n+1} &= E_n \left[1 + h \frac{\partial}{\partial Z} f(t_n, Z_n)\right] + \tau_{n+1}, \end{aligned} \tag{5.8}$$

where $E_n = Y(t_n) - y_n$ is the global error at the n^{th} step.

Example 5.2
Consider the IVP

$$\frac{\mathrm{d}Y}{\mathrm{d}t} = 1 - Y \quad \text{with } Y(0) = 0.$$

Analytic solution: $Y(t) = 1 - \mathrm{e}^{-t} \Rightarrow Y''(t) = -\mathrm{e}^{-t}$ and $\dfrac{\partial}{\partial Y} f(t, Y) = \dfrac{\mathrm{d}f(Y)}{\mathrm{d}Y} = -1$. Hence, from Equation (5.8), $E_n = E_{n-1}(1 - h) - \dfrac{h^2}{2}\mathrm{e}^{-\xi_n}$ with $\xi_n \in (t_{n-1}, t_n)$.

We want to bound E_n:

$$|E_n| \leq |E_{n-1}||1 - h| + \frac{h^2}{2}\mathrm{e}^{-\xi_n} \leq |E_{n-1}||1 - h| + \frac{h^2}{2} \quad (\text{since } \xi_n > 0).$$

Let $q = |1 - h|$;

$$|E_n| \leq |E_{n-1}|q + \frac{h^2}{2} \leq \left(|E_{n-2}|q + \frac{h^2}{2}\right) q + \frac{h^2}{2} \leq \left[\left(|E_{n-3}|q + \frac{h^2}{2}\right) q + \frac{h^2}{2}\right] q + \frac{h^2}{2},$$

so that

$$|E_n| \leq q^n |E_0| + \frac{h^2}{2}\left(1 + q + q^2 + \ldots + q^{n-1}\right).$$

However, $E_0 = Y(0) - y_0 = 0$, so $|E_n| \leq \dfrac{h^2}{2} \displaystyle\sum_{j=0}^{n-1} q^j$ $(*)$.

We look for bounds on error when Euler's scheme is stable, i.e. when $h < 2 \Rightarrow q = |h - 1| < 1$,

$$\Rightarrow |E_n| \leq \frac{h^2}{2} \sum_{j=0}^{n-1} 1 = n\frac{h^2}{2} = \frac{h}{2} t_n \quad \text{since } n = \frac{t_n}{h}.$$

This result shows that the global error is proportional to the step-size h. Thus, Euler's method is described as being first order accurate.

Note: from $(*)$, $|E_n| \leq \dfrac{h^2}{2}\dfrac{1 - q^n}{1 - q} \leq \dfrac{h^2}{2}\dfrac{1}{1 - q}$ if $q < 1$ (stable case). If $h < 1$ then $q = 1 - h$ and $|E_n| \leq \dfrac{h^2}{2}\dfrac{1}{h} = \dfrac{h}{2}$ $\forall n$.

While in principle more accuracy can be achieved by using smaller step-size, the accuracy is limited by round-off errors. So, Euler's method rarely used in practice.

Note: the local error is due to truncation and round-off,

$$\tau_n = \frac{h^2}{2} Y''(\xi_n) + \varepsilon Y(t_n).$$

So, in the example 5.2, $|E_n| \leq n\left(\frac{h^2}{2} + \varepsilon\right) = t_n\left(\frac{h}{2} + \frac{\varepsilon}{h}\right)$ which reaches its minimum value $\min\left[t_n\left(\frac{h}{2} + \frac{\varepsilon}{h}\right)\right] = \sqrt{2\varepsilon}$ at $h = \sqrt{2\varepsilon}$.

5.3 Runge-Kutta methods

5.3.1 Modified Euler method

Integral form of the solution to the initial value problem:

$$Y(t+h) = Y(t) + \int_t^{t+h} f(\tau, Y(\tau))\,\mathrm{d}\tau.$$

The local truncation error can be reduced by using a more accurate quadrature method for the integral than Euler's scheme.

E.g., the trapezium rule

$$\int_t^{t+h} f(\tau, Y(\tau))\,\mathrm{d}\tau = \frac{h}{2}\left[f(t, Y(t)) + f(t+h, Y(t+h))\right] - \frac{h^3}{12} Y'''(\xi)$$

gives an order h^3 local truncation error. However, it cannot be used directly as we do not know $Y(t+h)$. (This is what we are trying to calculate.)

The solution is to use the forward Euler method to estimate $Y(t+h)$ as $Y(t) + hf(t, Y(t))$ in the trapezium rule. So that,

$$Y(t+h) \approx Y(t) + \frac{h}{2}\left[f(t, Y(t)) + f(t+h, Y(t) + hf(t, Y(t)))\right],$$

leading to the new scheme

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_n + h, y_n + hf(t_n, y_n))\right],$$

or equivalently

$$\left.\begin{aligned} K_1 &= hf(t_n, y_n) \\ K_2 &= hf(t_n + h, y_n + K_1) \\ y_{n+1} &= y_n + \frac{1}{2}(K_1 + K_2) \end{aligned}\right\} \quad \text{modified Euler method.} \tag{5.9}$$

Local truncation error

Let us check the local truncation error of this scheme, using the Taylor series expansion

$$Y(t_n + h) = Y(t_n) + hY'(t_n) + \frac{h^2}{2}Y''(t_n) + \frac{h^3}{6}Y'''(\xi), \quad \xi \in (t_n, t_n + h).$$

Substituting $Y'(t_n) = f(t_n, Y(t_n))$ and

$$\begin{aligned} Y''(t_n) &= \frac{\mathrm{d}}{\mathrm{d}t}f(t, Y(t))\bigg|_{t_n} = \frac{\partial}{\partial t}f(t_n, Y(t_n)) + \frac{\mathrm{d}}{\mathrm{d}t}Y(t_n)\frac{\partial}{\partial Y}f(t_n, Y(t_n)) \quad \text{(chain rule)}, \\ &= \frac{\partial}{\partial t}f(t_n, Y(t_n)) + f(t_n, Y(t_n))\frac{\partial}{\partial Y}f(t_n, Y(t_n)), \end{aligned}$$

$$\begin{aligned} Y(t_n + h) = Y(t_n) &+ hf(t_n, Y(t_n)) \\ &+ \frac{h^2}{2}\left[\frac{\partial}{\partial t}f(t_n, Y(t_n)) + f(t_n, Y(t_n))\frac{\partial}{\partial Y}f(t_n, Y(t_n))\right] + O\left(h^3\right). \end{aligned} \quad (5.10)$$

Modified Euler: Let $K_1 = hf(t_n, Y(t_n))$ and $K_2 = hf(t_n + h, Y(t_n) + K_1)$.

Using Taylor series for two variables,

$$K_2 = h\left[f(t_n, Y(t_n)) + h\frac{\partial}{\partial t}f(t_n, Y(t_n)) + K_1\frac{\partial}{\partial Y}f(t_n, Y(t_n)) + O\left(h^2, K_1^2\right)\right],$$

$$\Rightarrow K_2 = hf(t_n, Y(t_n)) + h^2\left[\frac{\partial}{\partial t}f(t_n, Y(t_n)) + f(t_n, Y(t_n))\frac{\partial}{\partial Y}f(t_n, Y(t_n))\right] + O\left(h^3\right),$$

(Note $K_1 = O(h)$.)

$$\begin{aligned} \Rightarrow \frac{1}{2}(K_1 + K_2) = hf(t_n, Y(t_n)) &\\ + \frac{h^2}{2}\left[\frac{\partial}{\partial t}f(t_n, Y(t_n)) + f(t_n, Y(t_n))\frac{\partial}{\partial Y}f(t_n, Y(t_n))\right] &+ O\left(h^3\right), \end{aligned}$$

$$\text{Eq. (5.10)} \Rightarrow Y(t_n + h) = Y(t_n) + \frac{1}{2}(K_1 + K_2) + \underbrace{O\left(h^3\right)}_{\text{local truncation error}}.$$

Hence, the local truncation error is indeed $\tau_n = O\left(h^3\right)$, so that the modified Euler method is second order accurate. (A method is conventionally called p^{th} order if the local truncation error is of order $p + 1$.)

5.3.2 Second order Runge-Kutta scheme

Modified Euler is an example of 2^{nd} order R-K method.

The general 2^{nd} order R-K scheme takes the form

$$\left.\begin{aligned} K_1 &= hf(t_n, y_n) \\ K_2 &= hf(t_n + \alpha h, y_n + \beta K_1) \\ y_{n+1} &= y_n + a_1K_1 + a_2K_2 \end{aligned}\right\}. \quad (5.11)$$

Repeating the earlier analysis, we see that

$$K_2 = hf(t_n, Y(t_n)) + h^2\left[\alpha\frac{\partial}{\partial t}f(t_n, Y(t_n)) + \beta f(t_n, Y(t_n))\frac{\partial}{\partial Y}f(t_n, Y(t_n))\right] + O(h^3).$$

So,

$$Y(t_{n+1}) = Y(t_n) + a_1K_1 + a_2K_2 \Rightarrow Y(t_{n+1}) = Y(t_n) + h(a_1+a_2)f(t_n, Y(t_n)) \\ + a_2h^2\left[\alpha\frac{\partial}{\partial t}f(t_n, Y(t_n)) + \beta f(t_n, Y(t_n))\frac{\partial}{\partial Y}f(t_n, Y(t_n))\right] + O(h^3).$$

Comparing this expression with (5.10) (Taylor series) one gets

$$\left.\begin{aligned} a_1 + a_2 &= 1 \\ \alpha a_2 = \beta a_2 &= \frac{1}{2} \end{aligned}\right\}. \tag{5.12}$$

Since we have 3 equations and 4 unknowns, there are infinitely many solutions.

The most popular are,

- Modified Euler: $a_1 = a_2 = \frac{1}{2}; \alpha = \beta = 1;$
- Midpoint method: $a_1 = 0; a_2 = 1; \alpha = \beta = \frac{1}{2},$

$$\left.\begin{aligned} & K_1 = hf(t_n, y_n) \\ \Rightarrow\ & K_2 = hf(t_n + h/2, y_n + K_1/2) \\ & y_{n+1} = y_n + K_2 \end{aligned}\right\};$$

- Heun's method: $a_1 = 1/4; a_2 = 3/4; \alpha = \beta = \frac{2}{3},$

$$\left.\begin{aligned} & K_1 = hf(t_n, y_n) \\ \Rightarrow\ & K_2 = hf(t_n + 2h/3, y_n + 2K_1/3) \\ & y_{n+1} = y_n + \frac{1}{4}(K_1 + 3K_2) \end{aligned}\right\}.$$

5.3.3 Higher order Runge-Kutta methods

Schemes of the form (5.11) can be extended to higher order methods. The most widely used R-K scheme is the 4^{th} order scheme RK4 based on Simpson's rule.

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4), \\ \text{where } K_1 &= hf(t_n, y_n), \\ K_2 &= hf\left(t_n + \frac{h}{2}, y_n + \frac{K_1}{2}\right), \\ K_3 &= hf\left(t_n + \frac{h}{2}, y_n + \frac{K_2}{2}\right), \\ K_4 &= hf(t_n + h, y_n + K_3). \end{aligned} \tag{5.13}$$

This scheme has local truncation error of order h^5, which can be checked in the same way as the second order scheme, but involves rather messy algebra.

5.4 Multistep methods

The Euler and R-K methods are described as being one-step methods in that they only use the values at t_n to compute y_{n+1}.

However (as with Lagrange interpolation), we can get higher order estimates for y_{n+1} if we make use of more of our previous solutions: $y_{n-1}, y_{n-2}, \ldots$. (Clearly such schemes cannot be used for the first few steps.)

Such schemes are known as multistep methods.

5.4.1 Adams-Bashforth methods

Returning to the integral form of the solution,

$$Y(t_{n+1}) = Y(t_n) + \int_{t_n}^{t_{n+1}} f(t, Y(t))\,\mathrm{d}t.$$

Let us seek a quadrature rule based on the values of f at t_n and t_{n-1},

$$\int_{t_n}^{t_n+h} g(t)\,\mathrm{d}t = w_1 g(t_n - h) + w_2 g(t_n) + E.$$

Using the method of undetermined coefficients, we can find w_1 and w_2 such that the quadrature rule holds for all linear functions;

- take $g(t) = 1 \Rightarrow h = w_1 + w_2$,
- take $g(t) = t - t_n \Rightarrow \dfrac{h^2}{2} = -hw_1 \Rightarrow w_1 = -\dfrac{h}{2}$ and $w_2 = \dfrac{3h}{2}$.
- Error: $E = Kg''(\xi)$, so let $g(t) \equiv (t - t_n)^2$,

$$\Rightarrow \frac{h^3}{3} = w_1 h^2 + 2K \Rightarrow 2K = \frac{h^3}{3} + \frac{h^3}{2} = \frac{5}{6}h^3 \Rightarrow K = \frac{5}{12}h^3.$$

Hence,

$$\int_{t_n}^{t_n+h} g(t)\,\mathrm{d}t = -\frac{h}{2}g(t_n - h) + \frac{3h}{2}g(t_n) + \frac{5}{12}h^3\, g''(\xi).$$

This gives us the two-step Adams-Bashforth method

$$y_{n+1} = y_n + \frac{h}{2}[3f(t_n, y_n) - f(t_{n-1}, y_{n-1})], \tag{5.14}$$

with local truncation error $\dfrac{5}{12}h^3\, Y'''(\xi)$. This is a *second order method.*

Higher order A-B methods can be derived using $y_n, y_{n-1}, y_{n-2}, \ldots$. A method using N steps has a local truncation error of order h^{n+1}.

Advantages (over R-K): A-B methods only require one function evaluation per step (compared with 4 evaluations for RK4).

5.4.2 Milne's methods

Family of methods that extend the interval of integration so that the quadrature points lie *within* the range of integration.

The two-step Milne method is obtained as follow: in the expression

$$Y(t_{n+1}) = Y(t_{n-1}) + \int_{t_{n-1}}^{t_{n+1}} f(t, Y(t))\,\mathrm{d}t,$$

evaluate the integral using the *open Newton-Cotes* method:

$$\int_{t_n-h}^{t_n+h} f(t, Y(t))\,\mathrm{d}t = 2hf(t_n, Y(t_n)) + \frac{h^3}{3}Y'''(\xi),$$

giving the scheme

$$y_{n+1} = y_{n-1} + 2hf(t_n, y_n). \tag{5.15}$$

This is sometimes called the "leap-frog" method. Again, the local truncation error is of order h^3 (i.e. second order method).

In the same way, we may obtain the scheme

$$y_{n+1} = y_{n-3} + \frac{4h}{3}\left[2f(t_n, y_n) - f(t_{n-1}, y_{n-1}) + 2f(t_{n-2}, y_{n-2})\right], \tag{5.16}$$

with l.t.e. $O(h^5)$.

Milne's method (5.15) involves $f(t_n, y_n)$ only. So, it seems more efficient than the A-B scheme (5.14), which involves both $f(t_n, y_n)$ and $f(t_{n-1}, y_{n-1})$ for the same accuracy. However, there is a catch!

Stability

In certain cases, Milne's method can suffer from an instability in which the error cannot be eliminated by letting $h \to 0$.

Example 5.3
Using the two-step Milne method (5.15), solve

$$\frac{\mathrm{d}Y}{\mathrm{d}t} = 1 - Y, Y(0) = 0.$$

$$\Rightarrow y_{n+1} = y_{n-1} + 2h(1 - y_n) \Rightarrow y_{n+1} + 2hy_n - y_{n-1} = 2h.$$

To solve this difference equation,

i. consider the homogeneous problem $z_{n+1} + 2hz_n - z_{n-1} = 0$ and seek solution of the form $z_n = \lambda^n$,

$$\lambda^{n-1}\left[\lambda^2 + 2h\lambda - 1\right] = 0 \Rightarrow \lambda = -h \pm \sqrt{h^2+1}.$$

Thus the complementary function is $z_n = \alpha\left(-h + \sqrt{h^2+1}\right)^n + \beta\left(-h - \sqrt{h^2+1}\right)^n$;

ii. we observe that $y_n = 1$ is a particular solution of the difference equation:

$$1 + 2h \times 1 - 1 = 2h.$$

The general solution is

$$y_n = 1 + \alpha\left(-h + \sqrt{h^2+1}\right)^n + \beta\left(-h - \sqrt{h^2+1}\right)^n.$$

Since $h > 0$, $h + \sqrt{h^2+1} > 1$. We see that if $\beta \neq 0$, y_n diverges as $n \to \infty$.
As $h \to 0$, $\sqrt{h^2+1} = 1 + O(h^2)$ and

$$-h + \sqrt{h^2+1} \sim 1 - h \sim \mathrm{e}^{-h},$$
$$-h - \sqrt{h^2+1} \sim -(1+h) \sim -\mathrm{e}^{h}.$$

So, using $t_n = nh$,

$$y_n \sim 1 + \alpha \mathrm{e}^{-t_n} + \beta(-1)^n\, \mathrm{e}^{t_n},$$

$\Rightarrow$ The error grows exponentially!

Milne's method introduces a parasitic term $\beta(-1)^n\ \mathrm{e}^{t_n}$. Even if $\beta = 0$ initially, round-off errors will give $\beta \neq 0$ after a few steps.

Thus, Milne's method is unstable for all step sizes for this problem.

5.5 Implicit and predictor-corrector methods

5.5.1 Implicit methods

All the methods that we have discussed so far have been explicit in that the calculation of y_{n+1} requires information from *previous* steps only.

Implicit methods are derived using y_{n+1} as an additional interpolation point in the integral

$$\int_{t_n}^{t_n+h} f(t, Y(t))\,\mathrm{d}t.$$

E.g. the Adams-Moulton three-step implicit method is derived from

$$Y(t_{n+1}) = Y(t_n) + \int_{t_n}^{t_n+h} f(t, Y(t))\,\mathrm{d}t,$$

where the integral is approximated using the quadrature rule

$$\int_{t_n}^{t_n+h} g(t)\,\mathrm{d}t = \frac{h}{24}\left[g(t_{n-2}) - 5g(t_{n-1}) + 19g(t_n) + 9g(t_{n+1})\right] - \frac{19}{720}h^5\, g^{(\mathrm{iv})}(\xi),$$

giving the fourth order method

$$y_{n+1} = y_n + \frac{h}{24}\left[9f(t_{n+1}, y_{n+1}) + 19f(t_n, y_n) - 5f(t_{n-1}, y_{n-1}) + f(t_{n-2}, y_{n-2})\right]. \qquad (5.17)$$

Finding y_{n+1} requires now to solve a non-linear equation, so there appears to be little benefit in such a scheme.

However, the local truncation error of (5.17) is $-\frac{19}{720}h^5 Y^{(\mathrm{v})}(\xi)$, whereas the explicit 4-step Adams-Bashforth method,

$$y_{n+1} = y_n + \frac{h}{24}\left[55f(t_n, y_n) - 59f(t_{n-1}, y_{n-1}) + 37f(t_{n-2}, y_{n-2}) - 9f(t_{n-3}, y_{n-3})\right], \qquad (5.18)$$

has a local truncation error $-\frac{251}{720}h^5 Y^{(\mathrm{v})}(\xi)$.

Hence, although the two schemes are of the same order, the implicit scheme is over 13 times more accurate.

5.5.2 Predictor-corrector methods

Implicit schemes are often used together with an explicit scheme in a predictor-corrector method.

An explicit scheme is used to "predict" y_{n+1} and this predicted value y_{n+1}^{P} is then used to evaluate $f(t_{n+1}, y_{n+1})$ in an implicit scheme.

E.g., predictor step,

$$y_{n+1}^{\mathrm{P}} = y_n + \frac{h}{24}\left[55f(t_n, y_n) - 59f(t_{n-1}, y_{n-1}) + 37f(t_{n-2}, y_{n-2}) - 9f(t_{n-3}, y_{n-3})\right],$$

using 4-step A-B explicit scheme, then corrector step,

$$y_{n+1} = y_n + \frac{h}{24}\left[9f(t_{n+1}, y_{n+1}^{\mathrm{P}}) + 19f(t_n, y_n) - 5f(t_{n-1}, y_{n-1}) + f(t_{n-2}, y_{n-2})\right].$$

At each step, a predictor-corrector scheme requires two function evaluations rather than just one for a multistep method. However, the increased accuracy means that larger step size can be used to achieve the same accuracy.

In addition, $\left|y_{n+1}^{\mathrm{P}} - y_n\right|$ provides an estimation of the local truncation error.

Chapter 6

Linear systems of equations

Contents

Many applications (e.g. solution of PDEs) involve the solution of large sets of simultaneous linear equations.

This is an ideal task for a computer since it only involves lots of additions and multiplications.

6.1 Solution of simultaneous equations

6.1.1 Gaussian elimination with back-substitution

Example 6.1

$$\begin{array}{lrcrcl} (E_1) & 3x_1 & + & 5x_2 & = & 9, \\ (E_2) & 6x_1 & + & 7x_2 & = & 4. \end{array}$$

i. Add multiple of (E_1) to (E_2), such that coefficient of x_1 is zero.

$$\begin{array}{lrcrcl} (E_1) & \rightarrow(E_1) \quad 3x_1 & + & 5x_2 & = & 9, \\ (E_2 - 2E_1) & \rightarrow(E_2) \quad\quad & - & 3x_2 & = & -14. \end{array}$$

(E_2) gives $x_2 = 14/3$.

ii. Back-substitute in (E_1): $3x_1 = 9 - 5x_2 = 9 - 70/3 = -43/3 \Rightarrow x_1 = -43/9$.

Example 6.2

$$\begin{array}{lrcrcrcl} (E_1) & x_1 & - & x_2 & + & 3x_3 & = & 13, \\ (E_2) & 4x_1 & - & 2x_2 & + & x_3 & = & 15, \\ (E_3) & -3x_1 & - & x_2 & + & 4x_3 & = & 8. \end{array}$$

$$\begin{array}{lrcrcrcl} (E_1) & \rightarrow(E_1) \quad x_1 & - & x_2 & + & 3x_3 & = & 13, \\ (E_2 - 4E_1) & \rightarrow(E_2) \quad\quad & & 2x_2 & - & 11x_3 & = & -37, \\ (E_3 + 3E_1) & \rightarrow(E_3) \quad\quad & & -4x_2 & + & 13x_3 & = & 47. \end{array}$$

$$\begin{array}{lrcrcrcl} (E_1) & \rightarrow(E_1) \quad x_1 & - & x_2 & + & 3x_3 & = & 13, \\ (E_2) & \rightarrow(E_2) \quad\quad & & 2x_2 & - & 11x_3 & = & -37, \\ (E_3 + 2E_2) & \rightarrow(E_3) \quad\quad & & & & -9x_3 & = & -27. \end{array}$$

$(E_3) \Rightarrow x_3 = 27/9 = 3$; back-substitute in (E_2),
$(E_2) \Rightarrow 2x_2 = -37 + 11x_3 = -37 + 33 = -4 \Rightarrow x_2 = -2$; back-substitute in (E_1),
$(E_1) \Rightarrow x_1 = 13 + x_2 - 3x_3 = 13 - 2 - 9 = 2$.

$$\{x_1 = 2; x_2 = -2; x_3 = 3\}$$

6.1.2 Matrix factorisation — LU decomposition

Matrix form of a general linear system of equations:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\ \vdots &= \vdots, \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n. \end{cases} \Leftrightarrow A\mathbf{x} = \mathbf{b} \tag{6.1}$$

with $A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}$, $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$.

Do not calculate $\mathbf{x} = A^{-1}\mathbf{b}$: inefficient and unstable.

We could write the method of Gaussian elimination in terms of matrices.

Example 6.3
(See example 6.1.)

$$\begin{pmatrix} 3 & 5 \\ 6 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 9 \\ 4 \end{pmatrix}$$

Now replace row(2) by row(2) - 2 × row(1):

$$\overset{M}{\begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix}} \overset{A}{\begin{pmatrix} 3 & 5 \\ 6 & 7 \end{pmatrix}} \overset{\mathbf{x}}{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}} = \overset{M}{\begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix}} \overset{\mathbf{b}}{\begin{pmatrix} 9 \\ 4 \end{pmatrix}}$$

$$\overset{U}{\begin{pmatrix} 3 & 5 \\ 0 & -3 \end{pmatrix}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 9 \\ -14 \end{pmatrix}.$$

The matrix $M = \begin{pmatrix} 1 & 0 \\ m & 1 \end{pmatrix}$ performs the row operation of adding m × row(1) to row(2), and is called Gaussian transformation.

This can be easily generalised to n-dimension systems (6.1).

The matrix $M^{(1)} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ m_{21} & 1 & 0 & \ldots & 0 \\ m_{31} & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & & \ddots & \\ m_{n1} & 0 & 0 & \ldots & 1 \end{pmatrix}$ where $m_{i1} = -\dfrac{a_{i1}}{a_{11}}$ performs the operations $(\mathrm{R}_i + m_{i1} \times \mathrm{R}_1 \to \mathrm{R}_i)$, $i = 2, \ldots, n$.

Similarly the matrix

$$M^{(j)} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \dots & \ddots & \dots & 0 & \dots & 0 \\ 0 & \dots & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & \dots & m_{j+1,j} & 1 & \dots & 0 \\ \vdots & & & \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & m_{nj} & 0 & \dots & 1 \end{pmatrix} \quad \text{where} \quad m_{ij} = -\frac{a_{ij}}{a_{jj}} \tag{6.2}$$

performs the operations $(\mathrm{R}_i + m_{ij} \times \mathrm{R}_j \to \mathrm{R}_i)$, $i = j+1, \dots, n$.

Example 6.4
(See example 6.2.)

$$\begin{cases} r_1 \to r_1 \\ r_2 - 4r_1 \to r_2 \\ r_3 + 3r_1 \to r_3 \end{cases} \Rightarrow \overset{M^{(1)}}{\begin{pmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}} \overset{A}{\begin{pmatrix} 1 & -1 & 3 \\ 4 & -2 & 1 \\ -3 & -1 & 4 \end{pmatrix}} \overset{\mathbf{x}}{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}} = \overset{M^{(1)}}{\begin{pmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}} \overset{\mathbf{b}}{\begin{pmatrix} 13 \\ 15 \\ 8 \end{pmatrix}}$$

$$\begin{cases} r_1 \to r_1 \\ r_2 \to r_2 \\ r_3 + 2r_2 \to r_3 \end{cases} \Rightarrow \overset{M^{(2)}}{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}} \overset{M^{(1)}A}{\begin{pmatrix} 1 & -1 & 3 \\ 0 & 2 & -11 \\ 0 & -4 & 13 \end{pmatrix}} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \overset{M^{(2)}}{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}} \overset{M^{(1)}\mathbf{b}}{\begin{pmatrix} 13 \\ -37 \\ 47 \end{pmatrix}}$$

$$\overset{U = M^{(2)}M^{(1)}A}{\begin{pmatrix} 1 & -1 & 3 \\ 0 & 2 & -11 \\ 0 & 0 & -9 \end{pmatrix}} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \overset{M^{(2)}M^{(1)}\mathbf{b}}{\begin{pmatrix} 13 \\ -37 \\ -27 \end{pmatrix}}.$$

So far, we have made use of $(n-1)$ Gaussian transformations $M^{(j)}$ to transform the matrix A into

$$U = M^{(n-1)} M^{(n-2)} \dots M^{(2)} M^{(1)} A, \tag{6.3}$$

where U is an *upper triangular* matrix, i.e. $u_{ij} = 0, i > j$ (zero below the diagonal).

Recall that $M^{(j)}$ (see equation (6.2)) generates the row operations $(\mathrm{R}_i + m_{ij} \times \mathrm{R}_j \to \mathrm{R}_i)$, $i = j+1, \dots, n$.
So, the row operations $(\mathrm{R}_i - m_{ij} \times \mathrm{R}_j \to \mathrm{R}_i)$, $i = j+1, \dots, n$ reverse the effects of $M^{(j)}$. Hence, $L^{(j)} = {M^{(j)}}^{-1}$, the inverse of $M^{(j)}$ is

$$L^{(j)} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \dots & \ddots & \dots & 0 & \dots & 0 \\ 0 & \dots & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & \dots & -m_{j+1,j} & 1 & \dots & 0 \\ \vdots & & & \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & -m_{nj} & 0 & \dots & 1 \end{pmatrix} = {M^{(j)}}^{-1} \quad \text{where} \quad -m_{ij} = \frac{a_{ij}}{a_{jj}} \tag{6.4}$$

Example 6.5

- $n = 2$ (example 6.3):

$$M = \begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix} \Rightarrow L = M^{-1} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}.$$

- $n = 3$ (example 6.4):

$$M^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix} \Rightarrow L^{(1)} = {M^{(1)}}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix},$$

$$M^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix} \Rightarrow L^{(2)} = {M^{(2)}}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix}.$$

Thus, equation (6.3) can be transformed into

$$A = L^{(1)} L^{(2)} \dots L^{(n-1)} U, \tag{6.5}$$

and it is easy to verify that

$$L^{(1)} L^{(2)} \dots L^{(n-1)} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ -m_{21} & 1 & 0 & \dots & 0 \\ -m_{31} & -m_{32} & 1 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ -m_{n1} & -m_{n2} & \dots & -m_{n,n-1} & 1 \end{pmatrix} \equiv L. \tag{6.6}$$

L is a *lower triangular* matrix where $L_{ij} = 0, i < j$ and $L_{ij} = 1, i = j$.

Example 6.6
From example 6.5

$$L^{(1)} L^{(2)} = L = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ -3 & -2 & 1 \end{pmatrix}.$$

Matrix factorisation: The square matrix A can be factored into $LU = A$ where L is a lower triangular matrix (i.e. $L_{ij} = 0$ for $j > i$ and $L_{ii} = 1$) and U is an upper triangular matrix (i.e. $U_{ij} = 0$ for $j < i$).
The decomposition of A into a product LU is not unique. The construction with 1's on the diagonal of L is called *Doolittle's method*.

Example 6.7

$$\begin{pmatrix} 3 & 5 \\ 6 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 3 & 5 \\ 0 & -3 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & -1 & 3 \\ 4 & -2 & 1 \\ -3 & -1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ -3 & -2 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 & 3 \\ 0 & 2 & -11 \\ 0 & 0 & -9 \end{pmatrix}.$$

6.1.3 Solution of linear systems using LU decomposition

We can make use of the LU factorisation of A to solve the linear system

$$A\mathbf{x} = LU\mathbf{x} = \mathbf{b}$$

in a two-step triangular process.
Let $\mathbf{y} = U\mathbf{x}$, so that $L\mathbf{y} = \mathbf{b}$. The solution is obtained by solving recursively the two systems

$$L\mathbf{y} = \mathbf{b} \quad \text{and} \quad U\mathbf{x} = \mathbf{y}.$$

Example 6.8

i.

$$\begin{pmatrix} 3 & 5 \\ 6 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 9 \\ 4 \end{pmatrix} \Leftrightarrow \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 3 & 5 \\ 0 & -3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 9 \\ 4 \end{pmatrix}.$$

$$\text{First solve} \quad \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 9 \\ 4 \end{pmatrix} \Rightarrow \begin{cases} y_1 = 9, \\ y_2 = 4 - 2y_1 = -14. \end{cases}$$

$$\text{Then solve} \quad \begin{pmatrix} 3 & 5 \\ 0 & -3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 9 \\ -14 \end{pmatrix}$$

$$\Rightarrow \begin{cases} x_2 = \dfrac{14}{3}, \\ 3x_1 = 9 - 5x_2 = 9 - \dfrac{70}{3} = -\dfrac{43}{3} \Rightarrow x_1 = -\dfrac{43}{9}. \end{cases}$$

$$\text{Solution:} \quad \left\{ x_1 = -\frac{43}{9}; x_2 = \frac{14}{3} \right\}$$

ii.

$$\begin{pmatrix} 1 & -1 & 3 \\ 4 & -2 & 1 \\ -3 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 13 \\ 15 \\ 8 \end{pmatrix} \Leftrightarrow \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ -3 & -2 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 & 3 \\ 0 & 2 & -11 \\ 0 & 0 & -9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 13 \\ 15 \\ 8 \end{pmatrix}.$$

Let $\mathbf{y} = U\mathbf{x}$ and solve $L\mathbf{y} = \mathbf{b}$,

$$\begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ -3 & -2 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 13 \\ 15 \\ 8 \end{pmatrix},$$

$$\Rightarrow \begin{cases} y_1 = 13, \\ y_2 = 15 - 4 \times 13 = -37, \\ y_3 = 8 - (-3) \times 13 - (-2) \times (-37), \\ \quad = 8 + 39 - 74 = -27. \end{cases}$$

Then solve $U\mathbf{x} = \mathbf{y}$

$$\begin{pmatrix} 1 & -1 & 3 \\ 0 & 2 & -11 \\ 0 & 0 & -9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 13 \\ -37 \\ -27 \end{pmatrix},$$

$$\Rightarrow \begin{cases} x_3 = 3, \\ 2x_2 = -37 - (-11) \times 3 = -4 \Rightarrow x_2 = -2, \\ x_1 = 13 - (-1) \times (-2) - 3 \times 3 = 2. \end{cases}$$

$$\text{Solution:} \quad \{x_1 = 2; x_2 = -2; x_3 = 3\}.$$

Example 6.9

Solve the system

$$\begin{pmatrix} 3 & 2 & 1 \\ 1 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 10 \\ 13 \\ 6 \end{pmatrix}$$

i. Find the LU factorisation of the matrix.

$$\left\{\begin{array}{rl} r_1 & \to r_1 \\ r_2 - r_1/3 & \to r_2 \\ r_3 - r_1/3 & \to r_3 \end{array}\right. \Rightarrow \begin{pmatrix} 3 & 2 & 1 \\ 1 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 1/3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 3 & 2 & 1 \\ 0 & 7/3 & 5/3 \\ 0 & 1/3 & 2/3 \end{pmatrix}$$

$$\left\{\begin{array}{rl} r_1 & \to r_1 \\ r_2 & \to r_2 \\ r_3 - r_2/7 & \to r_3 \end{array}\right. \Rightarrow \begin{pmatrix} 3 & 2 & 1 \\ 1 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 1/3 & 1/7 & 1 \end{pmatrix} \begin{pmatrix} 3 & 2 & 1 \\ 0 & 7/3 & 5/3 \\ 0 & 0 & 3/7 \end{pmatrix}$$

(Note $2/3 - 5/21 = (14-5)/21 = 9/21 = 3/7$.) So,

$$\overset{A}{\begin{pmatrix} 3 & 2 & 1 \\ 1 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix}} = \overset{L}{\begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 1/3 & 1/7 & 1 \end{pmatrix}} \overset{U}{\begin{pmatrix} 3 & 2 & 1 \\ 0 & 7/3 & 5/3 \\ 0 & 0 & 3/7 \end{pmatrix}}.$$

ii. Solve the linear system by back-substitution and forward-substitution.

$$\begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 1/3 & 1/7 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 10 \\ 13 \\ 6 \end{pmatrix} \Rightarrow \left\{\begin{array}{l} u = 10, \\ v = 13 - \dfrac{10}{3} = \dfrac{29}{3}, \\ w = 6 - \dfrac{10}{3} - \dfrac{29}{21} = \dfrac{9}{7}. \end{array}\right.$$

$$\begin{pmatrix} 3 & 2 & 1 \\ 0 & 7/3 & 5/3 \\ 0 & 0 & 3/7 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 10 \\ 29/3 \\ 9/7 \end{pmatrix} \Rightarrow \left\{\begin{array}{l} z = 3, \\ \dfrac{7}{3}y = \dfrac{29}{3} - 3 \times \dfrac{5}{3} = \dfrac{14}{3} \Rightarrow y = 2, \\ 3x = 10 - 2 \times 2 - 3 = 3 \Rightarrow x = 1. \end{array}\right.$$

Solution: $\{x = 1; y = 2; z = 3\}$.

6.1.4 Compact variants of Gaussian elimination

Rather than constructing L and U by using Gaussian elimination steps, it is possible to solve directly for these matrices.

Illustration of the direct computation of L and U, considering $n = 3$:
Write $A = LU$ as

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix},$$

$$= \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} & l_{21}u_{13} + u_{23} \\ l_{31}u_{11} & l_{31}u_{12} + l_{32}u_{22} & l_{31}u_{13} + l_{32}u_{23} + u_{33} \end{pmatrix}.$$

First row: $u_{11} = a_{11}$, $u_{12} = a_{12}$, $u_{13} = a_{13}$;

Second row: $l_{21}u_{11} = a_{21} \Rightarrow l_{21} = \dfrac{a_{21}}{u_{11}}$, then $u_{22} = a_{22} - l_{21}u_{12}$ and $u_{23} = a_{23} - l_{21}u_{13}$;

Third row: $l_{31}u_{11} = a_{31} \Rightarrow l_{31} = \dfrac{a_{31}}{u_{11}}$ and $l_{31}u_{12} + l_{32}u_{22} = a_{32} \Rightarrow l_{32} = \dfrac{a_{32} - l_{31}u_{12}}{u_{22}}$ then $u_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23}$.

Example 6.10

i.

$$\begin{pmatrix} 3 & 5 \\ 6 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ l_{21} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix} = \begin{pmatrix} u_{11} & u_{12} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} \end{pmatrix},$$

$\Rightarrow u_{11} = 3$, $u_{12} = 5$, $l_{21} = \dfrac{6}{3} = 2$ and $u_{22} = 7 - 2 \times 5 = -3$.

So, $L = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$ and $U = \begin{pmatrix} 3 & 5 \\ 0 & -3 \end{pmatrix}$.

ii.

$$\begin{pmatrix} 1 & -1 & 3 \\ 4 & -2 & 1 \\ -3 & -1 & 4 \end{pmatrix} = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} & l_{21}u_{13} + u_{23} \\ l_{31}u_{11} & l_{31}u_{12} + l_{32}u_{22} & l_{31}u_{13} + l_{32}u_{23} + u_{33} \end{pmatrix}$$

First row: $u_{11} = 1$, $u_{12} = -1$ and $u_{13} = 3$;

Second row: $l_{21} = \dfrac{4}{1} = 4$, $u_{22} = -2 - 4 \times (-1) = 2$, $u_{23} = 1 - 4 \times 3 = -11$;

Third row: $l_{31} = -\dfrac{3}{1} = -3$, $l_{32} = \dfrac{-1 - (-3) \times (-1)}{2} = -2$,
$u_{33} = 4 - (-3) \times 3 - (-2) \times (-11) = -9$.

So, $L = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ -3 & -2 & 1 \end{pmatrix}$ and $U = \begin{pmatrix} 1 & -1 & 3 \\ 0 & 2 & -11 \\ 0 & 0 & -9 \end{pmatrix}$.

6.1.5 The computational cost

It is crucial to minimise the number of multiplications and divisions when solving large linear systems. (The computational time taken for multiplications and divisions is comparable and much greater than that for additions and subtractions.)

The Gaussian elimination procedure requires $O(n^3)$ operations whereas the back-substitution step requires only $O(n^2)$ operations.

Once the LU-factorisation of A has been obtained ($O(n^3)$ operations), solving $A\mathbf{x} = \mathbf{b}$ with different $\mathbf{b}$ requires $O(n^2)$ operations only.

6.1.6 Calculation of the determinant

Since the determinant of a product of matrices $\det(LU) = \det(L)\det(U)$, we can calculate $\det(A) = \det(L)\det(U)$.

For the diagonal matrices L and U,

$$\det(L) = \prod_{i=1}^{n} L_{ii} = 1 \qquad \text{and} \qquad \det(U) = \prod_{i=1}^{n} U_{ii}.$$

So, $\det(A) = \det(L)\det(U) = \det(U) = \prod_{i=1}^{n} U_{ii}$ (if $L_{ii} = 1, i = 1, \ldots, n$).

Example 6.11

$$\begin{vmatrix} 1 & -1 & 3 \\ 4 & -2 & 1 \\ -3 & -1 & 4 \end{vmatrix} = \begin{vmatrix} 1 & -1 & 3 \\ 0 & 2 & -11 \\ 0 & 0 & -9 \end{vmatrix} = 1 \times 2 \times (-9) = -18.$$

6.2 Pivoting

6.2.1 Failure of Gaussian elimination

Gaussian elimination will fail if the matrix is singular, i.e. if $\det(A) = 0$.

However, LU-factorisation will also fail if at any stage in the row elimination procedure $\left(r_i - \frac{a_{ij}}{a_{jj}} r_j \to r_i\right)$, the *pivot element* $a_{jj} = 0$.

E.g. $A = \begin{pmatrix} \mathbf{0} & 1 \\ 1 & 1 \end{pmatrix}$ is non-singular ($\det(A) = -1$) but LU-decomposition will fail (a_{21} cannot be removed since $a_{11} = 0$).

Similarly,

$$\begin{cases} r_1 & \to r_1 \\ r_2 - r_1 & \to r_2 \\ r_3 - 2r_1 & \to r_3 \end{cases} \Rightarrow \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 4 \\ 2 & 3 & 5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & \mathbf{0} & 1 \\ 0 & -1 & -1 \end{pmatrix}.$$

Again a_{32} cannot be removed since $a_{22} = 0$; so, LU-factorisation is impossible.

6.2.2 Large rounding errors

Even non-zero pivot elements a_{jj} can cause trouble if they are *small* compared to the other elements in the row.

E.g., solving

$$\begin{pmatrix} \mathbf{0.001} & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1.001 \\ 2 \end{pmatrix} \Leftrightarrow \begin{pmatrix} 1 & 0 \\ \mathbf{1000} & 1 \end{pmatrix} \begin{pmatrix} 0.001 & 1 \\ 0 & \mathbf{999} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1.001 \\ 2 \end{pmatrix}.$$

Large factors appear, giving rise to large rounding errors. Thus, for very large matrices (e.g. $10,000^2$), the error becomes large enough to swamp the solution. The method is unstable.

6.2.3 Permutation matrix

How to avoid zero pivot elements a_{jj}?

The ordering of the equations in a linear system is arbitrary and we could write them in a different order.

E.g., solving $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ instead of $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

We use a permutation matrix P, so that $M = PA$ can be factored, i.e. $PA = M = LU$.

E.g., $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ swaps (r_1) and (r_2): this process is called *pivoting*.

$$PA\mathbf{x} = P\mathbf{b} \Leftrightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Similarly, we could deal with the problem of small diagonal elements leading to large rounding errors by swapping rows, keeping all the terms small. Again this is called pivoting.

$$\overset{P}{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}} \overset{A}{\begin{pmatrix} 0.001 & 1 \\ 1 & 1 \end{pmatrix}} = \overset{M}{\begin{pmatrix} 1 & 1 \\ 0.001 & 1 \end{pmatrix}} = \overset{L}{\begin{pmatrix} 1 & 0 \\ 0.001 & 1 \end{pmatrix}} \overset{U}{\begin{pmatrix} 1 & 1 \\ 0 & 0.999 \end{pmatrix}}$$

Example 6.12
LU-factorisation.

$$\begin{cases} r_1 \quad \to r_1 \\ r_2 + r_1 \to r_2 \\ r_3 - r_1 \to r_3 \end{cases} \Rightarrow A = \begin{pmatrix} 2 & -1 & 1 \\ -2 & 1 & 0 \\ 2 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & -1 & 1 \\ 0 & \mathbf{0} & 1 \\ 0 & 3 & 2 \end{pmatrix}.$$

A cannot be factored into the product LU since the pivot element $a_{22} = 0$. But we can permute (r_2) and (r_3) using the permutation matrix

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

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 2 & -1 & 1 \\ -2 & 1 & 0 \\ 2 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 3 \\ -2 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & -1 & 1 \\ 0 & 3 & 2 \\ 0 & 0 & 1 \end{pmatrix}.$$

6.2.4 Practical applications

Partial pivoting

Pivoting techniques for Gaussian elimination in row j: if the pivot element $a_{jj} = 0$, then interchange row j with row i, where i is the smaller integer greater then j with $a_{ij} \neq 0$.

In practise we want the largest number in a column to be the pivot element before proceeding to the Gaussian elimination.

For the column j, find the row i below row j, for which $|a_{ij}|$ is the largest and swap the rows i and j, so that

$$|a_{jj}| \geq |a_{ij}|, \quad i > j.$$

Thus, $LU = PA$ where P permutes rows.

With partial pivoting we obtain a matrix L such that $|L_{ij}| \leq 1$, which is almost always sufficient to ensure that the LU-decomposition is stable.

Full pivoting

We can interchange the columns as well as the rows to make the pivot elements a_{jj} as large as possible (i.e. to make the element of L as small as possible).
Column 1: find the position of the element of A with the largest magnitude and swap rows and columns to bring it to position $(1, 1)$, so that $|a_{11}| \geq |a_{ij}|$, $\forall i, j$.
Column j (i.e. after $j-1$ Gaussian transformations): find the largest element in the remaining submatrix

$$\begin{pmatrix} a_{11} & a_{12} & \dots & \dots & a_{1n} \\ 0 & \ddots & \dots & \dots & \dots \\ 0 & \dots & a_{jj} & \dots & a_{jn} \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & a_{nj} & \dots & a_{nn} \end{pmatrix}$$

and bring it to position (j, j), so that $|a_{jj}| \geq |a_{ik}|$, $\forall i > j, \forall k > j$. So that, $LU = PAQ$ where P permutes rows and Q permutes columns.

Although full pivoting is in principle better than partial pivoting, it is not easy to implement in programs.

In general you *must* do pivoting to ensure stability. However, we normally only use partial pivoting since it is usually sufficient to ensure stability.

Pivoting is not needed only when the matrix is diagonally dominant, i.e.

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad i = 1, \ldots, n.$$

This is often true for matrices from partial differential equations.

Appendix A

Example of computer arithmetic

A.1 A very primitive computer

Let us consider a computer that uses 20-bit floating point numbers of the form

$$(-1)^s \times 0.m \times 2^{n-p},$$

with a 1-bit sign indicator, s, a 7-bit exponent, n, and a 12-bit mantissa, m, stored as binary numbers. The most significant bit of the mantissa must be 1. p is a bias subtracted from n to represent both positive and negative exponents.

Sign. $s = 0$ for positive numbers and $s = 1$ for negative numbers.

Exponent. The maximum value of the 7-bit exponent is $n = 1111111$ (binary) i.e. $n = 2^6 + \ldots + 2^0 = 2^7 - 1 = 127$ (decimal).
The length of the exponent controls the range of numbers that can be represented. To ensure however that numbers with small magnitude can be represented as accurately as numbers with large amplitude, we subtract the bias $p = 2^6 - 1 = 63$ from the exponent n. Thus, the effective range of the exponent is not $n \in \{0, \ldots, 127\}$ but $q = n - p \in \{-63, \ldots, 64\}$.

Mantissa. The minimum value of $0.m = 0.1$ (binary) i.e. $0.m = 2^{-1} = 0.5$ (decimal), and its maximum value is $0.m = 0.111111111111$ (binary) i.e. $0.m = 2^{-1} + 2^{-2} + \ldots + 2^{-12} = 1 - 2^{-12} \lessapprox 1$. Thus, $0.5 \leq 0.m < 1$.

Overflow and underflow. The absolute value of the largest floating point number that can be stored in the computer is $L = (1 - 2^{-12}) \times 2^{64} \approx 1.8 \times 10^{19}$. Computations involving larger numbers, e.g. L^2, produce an overflow error.

Similarly, the absolute value of the smallest floating point stored in the computer is $S = 0.5 \times 2^{-63} = 2^{-64} \approx 5.4 \times 10^{-20}$. Computations involving smaller numbers, e.g. S^2, produce an underflow error.

Example of machine number. Consider the number represented by

sign	exponent	mantissa
0	1001001	110100010011

that is $+0.110100010011 \times 2^{1001001-0111111} = +0.110100010011 \times 2^{0001010}$.

The sign indicator is 0, i.e. the number is positive.

The exponent is $n = 1001001$ (binary) i.e. $n = 2^6 + 2^3 + 2^0 = 64 + 8 + 1 = 73$ (decimal). Thus, the effective exponent $q = 73 - 63 = 10$ i.e. $q = 1010$ (binary).
The mantissa gives $0.m = 0.110100010011$ (binary) i.e. $0.m = 2^{-1} + 2^{-2} + 2^{-4} + 2^{-8} + 2^{-11} + 2^{-12} = 3347/4096$ (decimal).
So, the machine number represents $+3347/409 \times 2^{10} = 836.75$.

The next floating point number that we can store in this machine is $0|1001001|110100010100$, where the sign and the exponent remain unchanged and we simply add 1 to the least significant bit of the mantissa. The new number is $3348/4096 \times 2^{10} = 837$. So, our primitive computer would be unable to store exactly any number between 836.75 and 837, leading to a relative uncertainty equal to $0.25/836.75 \approx 3 \times 10^{-4}$

Machine ε. At worst, the relative uncertainty in the value of floating point numbers that this primitive computer can store is equal to

$$\varepsilon_{\text{machine}} = \frac{(2^{-1} + 2^{-12}) \times 2 - 2^{-1} \times 2}{2^{-1} \times 2} = 2^{-11} \approx 5 \times 10^{-4}$$

.
and is called machine ε. ($\varepsilon_{\text{machine}}$ is the smallest number that, when added to the floating point representation of 1, gives a number greater than 1.)

Rounding and chopping. Suppose that we perform a calculation to which the answer is $0.1101000100111101 \times 2^{1010} = 53565/65536 \times 2^{10} \approx 836.95$. There are two ways to approximate this:

i. the most accurate is rounding to the nearest floating point number, $0.110100010100 \times 2^{1010} = 3348/4096 \times 2^{10} = 837$;

ii. however, many computers simply chop-off the expression at the bit length of the mantissa and ignore the extra digits, giving an answer of $0.110100010011 \times 2^{1010} = 3347/409 \times 2^{10} = 836.75$.

The relative error incurred by either approximation is at worst $\varepsilon_{\text{machine}}$, and such errors are called round-off errors.

A.2 Real computers (IEEE Standard 754)

Normalised numbers. Computers usually assume implicitly that the most significant bit of the mantissa is 1 and hence increase the size of the mantissa. Thus, the normalised representation of floating point numbers is

$$(-1)^s \times 1.m \times 2^{n-p}.$$

Single precision. Numbers stored using 4 bytes, i.e. 32 bits, with a 1-bit sign indicator, a 8-bit exponent, $0 < n < 255$, and a 23-bit mantissa are called single precision floating point. (Here, $p = 2^7 - 1 = 127$.)

The absolute value of the largest and smallest single precision floating point numbers are, $L = (2 - 2^{-23}) \times 2^{127} \approx 3.4 \times 10^{38}$ and $S = 2^{-126} \approx 1.2 \times 10^{-38}$, respectively.

The machine ε, $\varepsilon_{\text{machine}} = 2^{-23} \approx 1.9 \times 10^{-7}$. Hence calculations are correct to 7 decimal places (i.e. numbers have 7 significant digits).

Note that, $n = 0$ and $n = 255$ are used to represent special values (zero, "Not a Number", infinity).

Double precision. Numbers stored using 8 bytes, i.e. 64 bits, with a 1-bit sign indicator, a 11-bit exponent, $0 < n < 2047$, and a 52-bit mantissa, are called double precision floating point. (Here, $p = 2^{10} - 1 = 1023$.)

The absolute value of largest and smallest double precision floating point numbers are, $L = (2 - 2^{-52}) \times 2^{1023} \approx 0.18 \times 10^{309}$ and $S = 2^{-1022} \approx 0.2 \times 10^{-307}$, respectively.

The machine ε, $\varepsilon_{\text{machine}} = 2^{-52} \approx 2.2 \times 10^{-16}$. Hence calculations are correct to 16 decimal places (i.e. numbers have 16 significant digits).

Again, $n = 0$ and $n = 2047$ are used to represent special values.

Appendix B

Useful theorems from analysis

Theorem B.1 (Taylor's Theorem)
If $f \in C^n[a,b]$ and $f^{(n+1)}$ exists on (a,b), then $f(x)$ can be written as

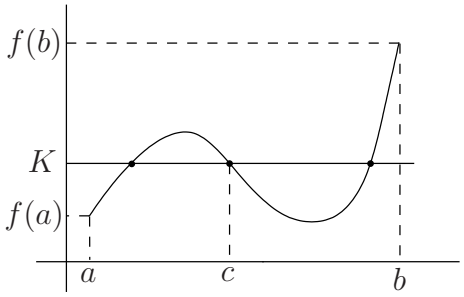
$$f(x) = f(x_0) + \frac{f^{(1)}(x_0)}{1!}(x-x_0) + \frac{f^{(2)}(x_0)}{2!}(x-x_0)^2 + \frac{f^{(3)}(x_0)}{3!}(x-x_0)^3 + \ldots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n + R_n(x),$$

for all $x \in [a,b]$, where the remainder term (Lagrange form) is

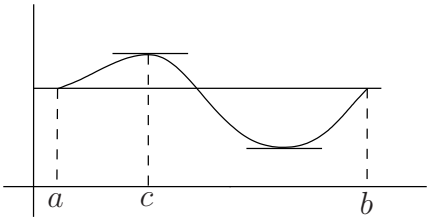
$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1},$$

with ξ some number between x_0 and x.

Theorem B.2 (Intermediate Value Theorem)
If $f \in C[a,b]$ (i.e. f is a continuous function on $[a,b]$) then, $\forall K$ such that $f(a) \leq K \leq f(b)$, $\exists c \in [a,b]$ with $f(c) = K$.

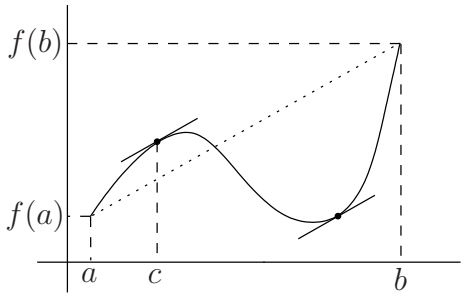


Theorem B.3 (Rolle's Theorem)
Suppose $f \in C[a,b]$ and f is differentiable on (a,b).
If $f(a) = f(b)$ then $\exists c \in [a,b]$ such that $f'(c) = 0$.



Theorem B.4 (Mean Value Theorem)
If $f \in C[a,b]$ and f is differentiable on (a,b) then $\exists c \in (a,b)$ such that

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$



Theorem B.5 (Generalised Rolle's theorem)
If $f \in C[a,b]$ and f is n times differentiable on (a,b), then if $f(x)$ is zero at $n+1$ distinct points in $[a,b]$, $\exists c \in (a,b)$ with $f^{(n+1)}(c) = 0$.

Theorem B.6 (Weighted Mean Value Theorem for Integrals)
Suppose f is a continuous function on $[a,b]$ and g is a differentiable function on $[a,b]$ that does not change sign on $[a,b]$. Then $\exists c \in [a,b]$ with

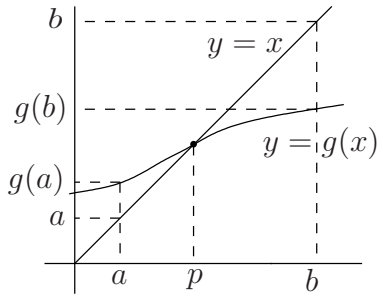
$$\int_a^b f(x)g(x)\mathrm{d}x = f(c)\int_a^b g(x)\mathrm{d}x.$$

Theorem B.7 (Fixed Point Theorem)
If $g \in C[a,b]$ (i.e. is a continuous function on the interval $[a,b]$) and, $\forall x \in [a,b]$, $g(x) \in [a,b]$ then g has a fixed point in $[a,b]$ (existence).

If, in addition, $g'(x)$ exits on (a,b) and there exists a positive constant $K < 1$ such that, $\forall x \in (a,b)$, $|g'(x)| \leq K$ then

i. the fixed point is unique (uniqueness),

ii. for any $p_0 \in [a,b]$ the sequence $\{p_n = g(p_{n-1})\}$ converges to this unique fixed point $p \in [a,b]$ (stability).



It is important to note that this theorem gives sufficient conditions for convergence, but they are not necessary.

Definition We define

$$f(x) = O[g(x)], \quad x \to x_0,$$

and say "$f(x)$ is — at most — of order $g(x)$ as $x \to x_0$" or "$f(x)$ is 'O' of $g(x)$ as $x \to x_0$", if $f(x)/g(x)$ is bounded for x near x_0; that is, if there exits $M \in \mathbb{R}_+^\star$ and $\delta \in \mathbb{R}_+^\star$ such that $|f(x)| \leq M|g(x)|$ for $|x - x_o| < \delta$.

Appendix C

Analytic derivation of the Newton-Raphson method

Let p be a root of the function $f \in C^2[a,b]$ (i.e. $f(p) = 0$), and p_0 be an approximation to p. If p_0 is sufficiently close to p, the expansion of $f(p)$ as a Taylor series in powers of $(p - p_0)$ is

$$f(p) = f(p_0) + (p - p_0)f'(p_0) + O((p - p_0)^2).$$

To find an approximate to p, we keep the linear term only

$$f(p) = 0 \approx f(p_0) + (p - p_0)f'(p_0).$$

Thus,

$$p - p_0 \approx -\frac{f(p_0)}{f'(p_0)} \Rightarrow p \approx p_1 = p_0 - \frac{f(p_0)}{f'(p_0)}.$$

Starting from the point p_0, the amount of offset needed to get closer to the root $p \approx p_1 = p_0 + h$ is $h = -f(p_0)/f'(p_0)$.

Newton-Raphson uses this algorithm iteratively to generate the sequence $\{p_n\}$, where

$$p_{n+1} = p_n - \frac{f(p_n)}{f'(p_n)}.$$

Clearly, for all iterates $f'(p_n) \neq 0$ is required.

Appendix D

Order of convergence of the secant method

The iteration scheme for solving $f(x) = 0$ using the secant method is

$$p_{n+1} = p_n - (p_n - p_{n-1})\frac{f(p_n)}{f(p_n) - f(p_{n-1})}.$$

Expanding in Taylor series around the root $x = p$ for p_n and p_{n-1} gives,

$$f(p_{n-1}) = f(p) + \varepsilon_{n-1}f'(p) + \frac{\varepsilon_{n-1}^2}{2}f''(p) + O(\varepsilon_{n-1}^3),$$
$$f(p_n) = f(p) + \varepsilon_n f'(p) + \frac{\varepsilon_n^2}{2}f''(p) + O(\varepsilon_n^3),$$

where $\varepsilon_n = p_n - p$. So,

$$\begin{aligned}
\varepsilon_{n+1} &= \varepsilon_n - (\varepsilon_n - \varepsilon_{n-1})\frac{\varepsilon_n f'(p) + \varepsilon_n^2/2\ f''(p) + O(\varepsilon_n^3)}{(\varepsilon_n - \varepsilon_{n-1})f'(p) + (\varepsilon_n^2 - \varepsilon_{n-1}^2)/2\ f''(p) + O(\varepsilon_{n-1}^3)}, \\
&= \varepsilon_n - \frac{\varepsilon_n f'(p) + \varepsilon_n^2/2\ f''(p) + O(\varepsilon_n^3)}{f'(p) + (\varepsilon_n + \varepsilon_{n-1})/2\ f''(p) + O(\varepsilon_{n-1}^3)}, \\
&= \varepsilon_n - \frac{\varepsilon_n + \varepsilon_n^2 f''(p)/2f'(p) + O(\varepsilon_n^3)}{1 + (\varepsilon_n + \varepsilon_{n-1})f''(p)/2f'(p) + O(\varepsilon_{n-1}^3)}, \\
&= \varepsilon_n - (\varepsilon_n + \varepsilon_n^2\frac{f''(p)}{2f'(p)} + O(\varepsilon_n^3))(1 - (\varepsilon_n + \varepsilon_{n-1})\frac{f''(p)}{2f'(p)} + O(\varepsilon_{n-1}^3)), \\
&= \varepsilon_n - \varepsilon_n - \varepsilon_n^2\frac{f''(p)}{2f'(p)} + \varepsilon_n(\varepsilon_n + \varepsilon_{n-1})\frac{f''(p)}{2f'(p)} + O(\varepsilon_n^3), \\
&= \varepsilon_n\varepsilon_{n-1}\frac{f''(p)}{2f'(p)} + O(\varepsilon_n^3).
\end{aligned}$$

The error at iteration $n+1$, $|\varepsilon_{n+1}|$, depends on $|\varepsilon_n|$ and $|\varepsilon_{n-1}|$, so we cannot directly use our definition of the order convergence. However, $\varepsilon_{n+1} \propto \varepsilon_n\varepsilon_{n-1}$ suggests a power law relationship of the form

$$\varepsilon_n = \varepsilon_{n-1}^{\alpha}\left(\frac{f''(p)}{2f'(p)}\right)^{\beta}.$$

Thus, we can substitute

$$\varepsilon_{n-1} = \varepsilon_n^{1/\alpha}\left(\frac{f''(p)}{2f'(p)}\right)^{-\beta/\alpha}$$

in the expression for ε_{n+1},

$$\varepsilon_{n+1} = \varepsilon_n \varepsilon_n^{1/\alpha} \left(\frac{f''(p)}{2f'(p)} \right)^{-\beta/\alpha} \frac{f''(p)}{2f'(p)} = \varepsilon_n^{(1+\alpha)/\alpha} \left(\frac{f''(p)}{2f'(p)} \right)^{1-\beta/\alpha},$$

which we equate with the power law relationship,

$$\alpha = \frac{1+\alpha}{\alpha} = \frac{1+\sqrt{5}}{2} \text{ and } \beta = 1 - \frac{\beta}{\alpha} = \frac{1}{\alpha}.$$

Thus,

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} = \lim_{n\to\infty} \frac{|\varepsilon_{n+1}|}{|\varepsilon_n|^\alpha} = \left| \frac{f''(p)}{2f'(p)} \right|^{1/\alpha}$$

shows that the secant method has order of convergence $\alpha \approx 1.62$.

Appendix E

More examples of Lagrange interpolation

E.1 Lagrange polynomials

We wish to find the polynomial interpolating the points

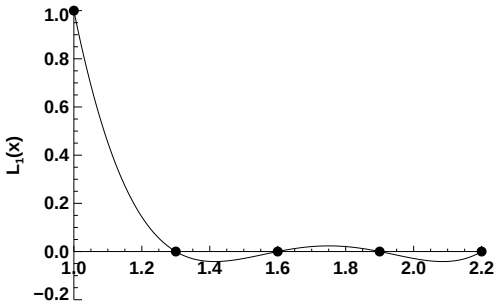
x	1	1.3	1.6	1.9	2.2
$f(x)$	0.1411	-0.6878	-0.9962	-0.5507	0.3115

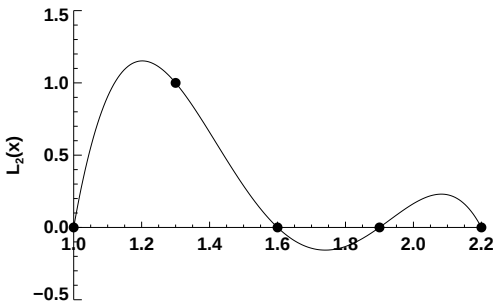
where $f(x) = \sin(3x)$, and estimate $f(1.5)$.

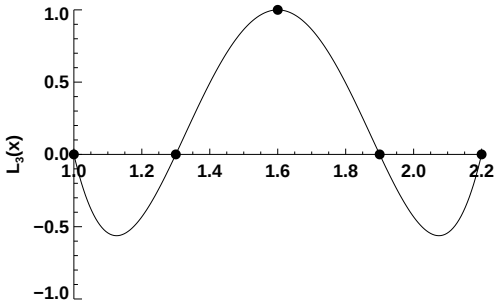
First, we find Lagrange polynomials $L_k(x)$, $k = 1 \dots 5$,

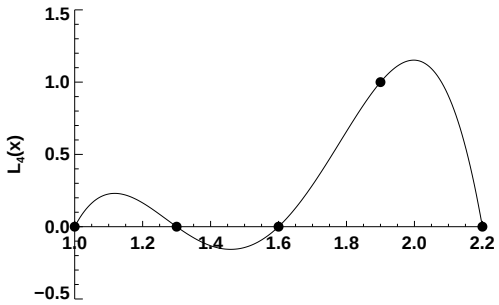
$$L_1(x) = \frac{(x-1.3)(x-1.6)(x-1.9)(x-2.2)}{(1-1.3)(1-1.6)(1-1.9)(1-2.2)}, \quad L_2(x) = \frac{(x-1)(x-1.6)(x-1.9)(x-2.2)}{(1.3-1)(1.3-1.6)(1.3-1.9)(1.3-2.2)}$$
$$L_3(x) = \frac{(x-1)(x-1.3)(x-1.9)(x-2.2)}{(1.6-1)(1.6-1.3)(1.6-1.9)(1.6-2.2)}, \quad L_4(x) = \frac{(x-1)(x-1.3)(x-1.6)(x-2.2)}{(1.9-1)(1.9-1.3)(1.9-1.6)(1.9-2.2)}$$
$$L_5(x) = \frac{(x-1)(x-1.3)(x-1.6)(x-1.9)}{(2.2-1)(2.2-1.3)(2.2-1.6)(2.2-1.9))}$$

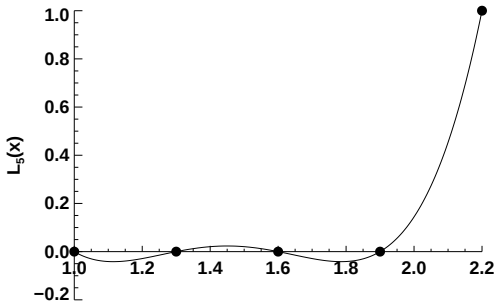
with the following graphs,





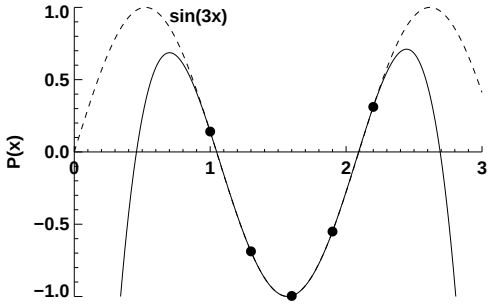






Clearly, $L_k(x_i) = \delta_{ik}$. Next, the polynomial approximation can be assembled,

$$P(x) = 0.1411 \times L_1(x) - 0.6878 \times L_2(x) - 0.9962 \times L_3(x) - 0.5507 \times L_4(x) + 0.3115 \times L_5(x).$$

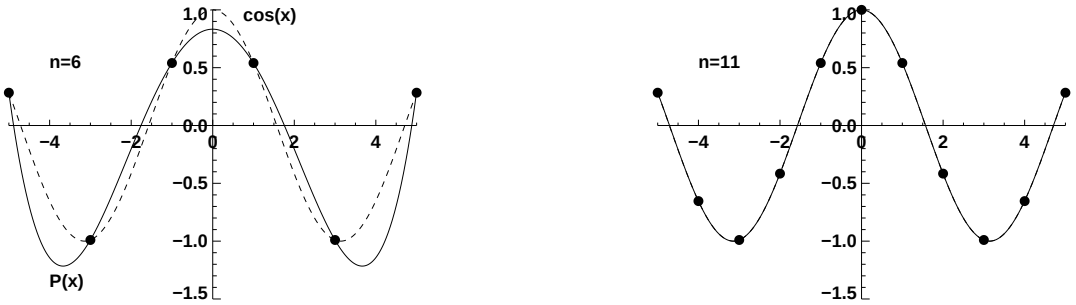


The interpolating polynomial approximates accurately the function $f(x) = \sin(3x)$ in the interval $[1, 2.2]$, with five points only.

So, $P(1.5) \approx -0.9773$ is an approximate to $f(1.5) = \sin(4.5) \approx -0.9775$ accurate within $E \approx 2 \times 10^{-4}$.

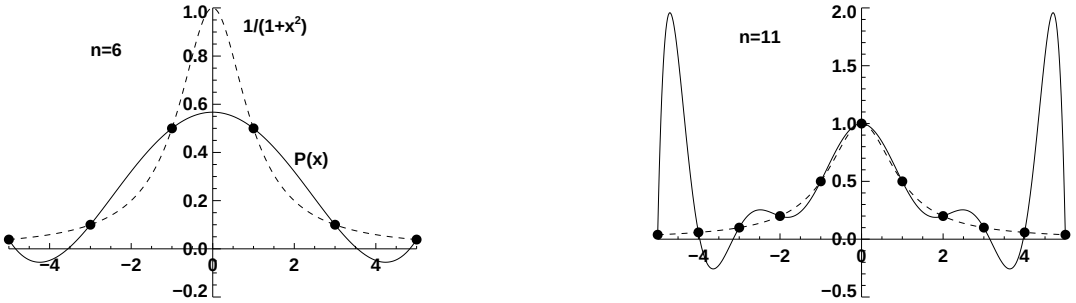
E.2 Convergence of "Lagrange" interpolation

First, consider, $P(x)$, the polynomial interpolating $f(x) = \cos(x)$ through a set of equidistant points in the interval $[-5, 5]$.

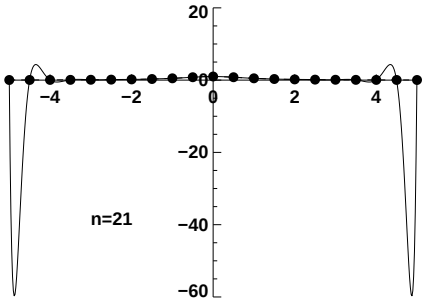


Clearly, increasing the number of equidistant points from $n = 6$ (left panel) to $n = 11$ (right panel) significantly improves the approximation of $f(x)$ by the polynomial P. In the right panel, the 10^{th} order interpolating polynomial (solid line) matches perfectly with the function $\cos(x)$.

However, Lagrange interpolation is not always accurate. For instance, consider the polynomial interpolating the Lorentz function, $f(x) = 1/(1 + x^2)$, through a set of equidistant points in the interval $[-5, 5]$.



Increasing the number of equidistant points from $n = 6$ (left panel) to $n = 11$ (right panel) improves the polynomial interpolation in the central part of $f(x)$, but large oscillations are present in the flat region.



If the number of equidistant interpolation points is increased further, these oscillations get even larger. The interpolating polynomial of degree $n - 1$, $P(x)$, does not converge to the function $1/(1 + x^2)$ as $n \to \infty$.