HMTHCS212 Numerical Methods
Block D, July 2022
University of Zimbabwe

Dr. Shelvean Kapita

Contents

1 Introduction

Scientific computing may be defined as the discipline concerned with the development and study of **numerical algorithms** for solving mathematical problems

arising from various disciplines such as science, finance, and engineering.

In most cases, the starting point is a **mathematical model** designed to understand an observed phenomenon in chemistry, biology, physics, engineering, economics or any other discipline. We will focus on mathematical models that are *continuous* or *piecewise continuous* that are difficult to solve using analytical methods. In order to solve such a problem on a computer, the continuous model is replaced by a discrete one. Functions are replaced by array values. Algorithms are then sought to solve the discrete approximations accurately and efficiently. **Numerical analysis** is the study of the theory behind these algorithms.

The most fundamental feature of a numerical algorithm is the presence of error. The result of a numerical computation is only approximate, and a major goal of numerical analysis is to make sure that the error is tolerably small.

Relative and Absolute Errors

There are two main types of measured error. Suppose x is a scalar quantity and x_c is its approximation.

- *absolute error*: $|x - x_c|$
- *relative error*: $\frac{|x-x_c|}{|x|}$ (assuming $x \neq 0$).

Example: Let 3.14 be an approximation of $\pi = 3.141592653589793....$ Find the absolute and relative errors of such an approximation to 6 significant figures.

Solution:

- absolute error $= |\pi - 3.14| = 0.00159265$
- relative error $= \frac{|\pi-3.14|}{|\pi|} = 0.000506957$
- percentage error = relative error $\times 100\% = 0.0506957\%$.

If the measured quantity $x \approx 1$, then there is not much difference between the absolute and relative errors. But when $|x| >> 1$ the relative error is more meaningful.

Types of Errors

x	x_c	abs. error	rel. error
1	0.99	0.01	0.01
1	1.01	0.01	0.01
-1.5	-1.2	0.3	0.2
100	99	1	0.01
100	99.99	0.01	0.0001

Table 1: Absolute and relative errors

1: Errors in the mathematical model: These are errors due to deliberate simplication of a mathematical model in the description of a phenomenon to make calculations tractable. For example planets are approximated by spheres, linearisation of a nonlinear phenomenon by neglecting higher order effects, or discarding relatively unimportant chemical reactions in complex chemical modelling.

2: Errors in the input data: These errors arise due to physical measurements which are never infinitely accurate, but are accurate up to some level of tolerance. The height of a chair measured using a tape is accurate up to a few millimetres, and the average radius of the earth can be estimated with an error up to a few kilometres, for example. At the level of numerical algorithms, there is really nothing one can do about modelling and input data errors mentioned above. Nevertheless, these errors should be taken into consideration when assessing the accuracy of the results of a numerical experiment to model a phenomenon.

3: Approximation errors: These errors arise when an approximate formula is used in place of the actual function to be evaluated.

- **Discretization errors** arise from discretization of continuous processes, e.g. replacing derivatives by difference formulas, integrals by Riemann sums, or approximating functions by interpolating polynomials.

  (i) $f'(x) = \boxed{\dfrac{f(x+h)-f(x)}{h}} + \frac{h}{2}f''(\xi)$, for $x < \xi < x+h$

  (ii) $\int_x^{x+h} f(s)\,ds = \boxed{\dfrac{h}{2}(f(x)+f(x+h))} - \frac{h^3}{12}f''(\xi)$, for $x < \xi < x+h$

  (iii) $\cos(x) = \boxed{1-\dfrac{x^2}{2}} + \frac{1}{6}x^3\sin(\xi)$, for x near 0, $0 < \xi < x$.

  The boxed quantity is the discrete approximation. The remaining term, containing (a generally unknown) ξ is the discretisation error.

- **Convergence errors:** Typically, nonlinear problems are solved using iterative methods. Generally, an iterative process would converge to the exact solution in infinitely many steps, but we stop the process after finitely many steps. For example, to approximate $\sqrt{2} = 1.41421356237\cdots$ by hand one may use the iterative formula

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{2}{x_n}\right).$$

This formula is known as the Babylonian method for finding square roots. It was known by the Babylonians (1500 BC), long before Newton generalized it. Starting with the guess $x_0 = 1$, we get

$$\begin{aligned} x_1 &= \frac{1}{2}\left(1 + \frac{2}{1}\right) = \frac{3}{2} = 1.5 \\ x_2 &= \frac{1}{2}\left(\frac{3}{2} + \frac{4}{3}\right) = \frac{17}{12} = 1.4166666\cdots \\ x_3 &= \frac{1}{2}\left(\frac{17}{12} + \frac{24}{17}\right) = \frac{577}{408} = 1.41421568627\cdots \end{aligned}$$

The convergence (absolute) errors are as follows:

$$\begin{aligned} |\sqrt{2} - x_1| &= 0.414213 \\ |\sqrt{2} - x_2| &= 0.00245310 \\ |\sqrt{2} - x_3| &= 0.000002139 \end{aligned}$$

Note that the iterates x_n converge rapidly to $\sqrt{2}$ when n increases.

Roundoff errors: These arise from the finite representation of real numbers on a computer. This finite representation of numbers affects both data and arithmetic operations. For example, the fraction

$$\frac{2}{3} = 0.66666666...$$

has a non-terminating decimal, which needs to be truncated at some point. In some cases, the number is rounded up, for example

$$\frac{2}{3} \approx 0.6666\cdots667$$

introducing an additional source of error. Even if a number is represented exactly on a computer, arithmetic operations such as addition, subtraction, multiplication and division may introduce roundoff errors.

Let us investigate the effects of discretisation and roundoff errors in the approximation of the derivative $f'(x)$ of the function $f(x) = e^x$ at $x = 0$. Since $f'(x) = e^x$, we know that $f'(0) = e^0 = 1$. Using the one-step forward difference formula,

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

with $x = 0$

$$f'(0) \approx \frac{e^h - 1}{h}.$$

The absolute error of the approximation, as we will show later, is in the region of $h/2$.

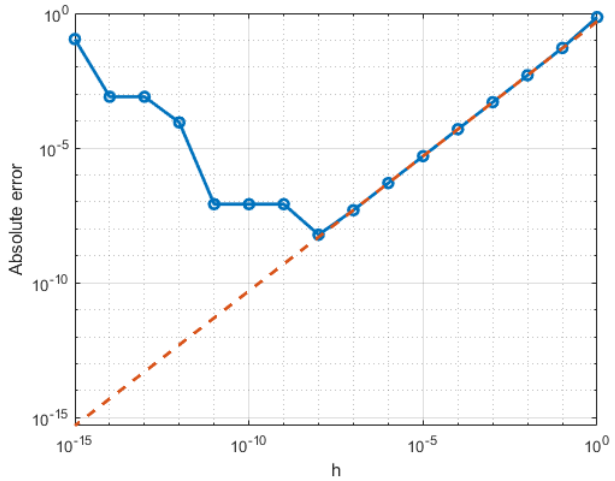


Figure 1: Discretization and roundoff errors. The solid curve interpolates the values of $|e^0 - (e^h - 1)/h|$. The dashed curve represents the discretization error without roundoff error. For very small h when $h < 10^{-8}$, roundoff error dominates the discretization error.

MATLAB code for generating Fig 1

```
>> df = @ (x,h) (exp(x+h)-exp(x))./h;
>> x = 0;
>> h = 10.^(0:-1:-15)';
>> abserror = abs(exp(x)-df(x,h));
>> discerror = h/2;
>> loglog(h,abserror,'o-',h,discerror,'--','Linewidth',2)
>> grid on; xlabel=('h'); ylabel=('Absolute error')
```

1.1 Loss of significance

A major source of roundoff error, known as cancellation, results from the subtraction of nearly equal numbers. For example, when $h < 10^{-8}$ in Fig 1, the quantity e^h is very close to 1, resulting in loss of significant figures when $e^h - 1$ is computed. To overcome this, we need to find ways to avoid subtracting nearly equal quantities. In this example, we will use a Taylor series expansion. Recall the Taylor series expansion. Suppose $f(x)$ has $k+1$ derivatives in an interval containing the points a and $x := a + h$. Then

$$f(x) = f(a) + hf'(a) + \frac{h^2}{2}f''(a) + \cdots + \frac{h^k}{k!}f^{(k)}(a) + \frac{h^{k+1}}{(k+1)!}f^{(k+1)}(\xi)$$

where $\xi \in (a, a+h)$.

Expanding e^h in Taylor series around $h = 0$:

$$e^h = 1 + \frac{h}{1!} + \frac{h^2}{2!} + \frac{h^3}{3!} + \cdots .$$

Subtracting 1, and dividing by h we see that

$$\frac{e^h - 1}{h} = 1 + \frac{h}{2} + \frac{h^2}{6} + \cdots$$

When h is small, e.g. $h = 10^{-8}$, then $h^2 = 10^{-16}$ is much smaller than h, so that terms h^2 and higher can be omitted from the sum, and the difference formula is accurately approximated by the linear term

$$\frac{e^h - 1}{h} \approx 1 + \frac{h}{2}.$$

Now observe that the approximation $1 + \frac{h}{2}$ avoids subtraction of nearly equal terms. The absolute error of the approximation of the derivative $f'(0)$ by the linear function $g(h) = 1 + h$ is then $|1 - (1 + h/2)| = h/2$ which is shown by the dotted line in Fig 1 to converge even for very small values of h.

Example 1: Find $\sqrt{x+1} - \sqrt{x}$ for $x = 10^{16}$.

In MATLAB, the above calculation returns 0. This is because the quantities $\sqrt{10^{16}+1}$ and $\sqrt{10^{16}}$ are almost equal, and their representations on the computer are the same, so their difference is calculated as zero. To avoid this error, let us use a trick from calculus called *conjugate multiplication*.

$$\begin{aligned}(\sqrt{x+1} - \sqrt{x}) \cdot \frac{\sqrt{x+1} + \sqrt{x}}{\sqrt{x+1} + \sqrt{x}} &= \frac{(x+1) - x}{\sqrt{x+1} + \sqrt{x}} \\ &= \frac{1}{\sqrt{x+1} + \sqrt{x}}\end{aligned}$$

Evaluating this expression at $x = 10^{16}$ in MATLAB gives

$$\frac{1}{\sqrt{10^{16}+1}+\sqrt{10^{16}}} = 5 \times 10^{-9},$$

a small, but nonzero number.

Example 2: Find both the roots of the quadratic equation $x^2 + 8^{12}x = 1$ correct to 5 significant figures.

It is well-known that the roots of the quadratic equation $ax^2 + bx + c = 0$ with $a \neq 0$ are given by the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Now setting $a = 1, b = 8^{12}$ and $c = -1$, we obtain the two roots

$$x_1 = \frac{-8^{12} - \sqrt{8^{24}+4}}{2}, \quad x_2 = \frac{-8^{12} + \sqrt{8^{24}+4}}{2}$$

Evaluating in MATLAB

$$x_1 = -6.8719 \times 10^{10}, \;\; x_2 = 0.0$$

Notice that to compute x_2 we subtract two nearly identical large numbers, resulting in catastrophic cancellation. To avoid this error, let us rewrite the formula in a new way. By conjugate multiplication,

$$\begin{aligned}
\frac{-b+\sqrt{b^2-4ac}}{2a} &= \frac{(-b+\sqrt{b^2-4ac})}{2a}\frac{(-b-\sqrt{b^2-4ac})}{(-b-\sqrt{b^2-4ac})} \\
&= \frac{b^2-(b^2-4ac)}{2a(-b-\sqrt{b^2-4ac})} \\
&= \frac{4ac}{2a(-b-\sqrt{b^2-4ac})} \\
&= \frac{2c}{-b-\sqrt{b^2-4ac}}.
\end{aligned}$$

Computing x_2 in MATLAB again using the new formula gives

$$x_2 = \frac{2}{-8^{12}-\sqrt{8^{24}+4}} \approx -1.4552 \times 10^{-11}$$

a small but nonzero number.

Exercise 1: Consider the function $f(x) = \ln(1+x)$. Suppose that the derivative $f'(x)$ at $x = 0$ is approximated by the forward difference formula $f'(0) \approx \frac{f(h)-f(0)}{h}$. For small values of h, significant roundoff error is observed. Use a Taylor series expansion about $h = 0$ to find a linear approximation of $\frac{f(h)-f(0)}{h}$. Explain why the linear approximation avoids the roundoff errors.

Exercise 2: Prove the following identities. Identify the values of x for which there is subtraction of nearly equal numbers. Which of the two formulas is more suitable for numerical computation? Explain why.

(a) $\ln(x - \sqrt{x^2 - 1}) = -\ln(x + \sqrt{x^2 - 1})$

(b) $\frac{1-\sec(x)}{\tan^2(x)} = \frac{-1}{1+\sec(x)}$

Exercise 3:

(a) Define $f(x) = x^2 - x\sqrt{x^2 + 9}$. Calculate $f(8^{12})$ correct to 5 significant figures.

(b) Find both the roots of $3x^2 - 9^{14}x + 100 = 0$ correct to 5 significant figures.

2 Roots of nonlinear equations

Equation solving is one of the most basic problems in scientific computing. This chapter introduces a number of methods to solve the equation

$$f(x) = 0,$$

where $x \in \mathbb{R}$ is a real variable. Any real number $r \in \mathbb{R}$ satisfying $f(r) = 0$ is called a *root* or a *zero* of the function $f(x)$. We will assume that f is a continuous function on an interval around the unknown solution. This condition is sufficient to guarantee convergence of the Bisection Method. Newton's method requires additional knowledge of the derivative $f'(x)$. In some situations, the derivative may be unavailable. In such a case, the Secant method which replaces the derivative by a forward difference formula may be used. In this chapter, we will study all three root-finding algorithms in terms of their rates of convergence and number of iterations to reach a particular error level.

Example 1: Find all solutions of the equation $ax + b = 0$, where $a \neq 0$.

This is a linear equation in one variable whose solution is the x-intercept, where the graph of the line crosses the x-axis. Algebraically, the solution is simply $x = -b/a$.

Example 2: Find all solutions of the equation $ax^2 + bx + c = 0$, $a \neq 0$.

This is a quadratic equation whose solutions in closed form are given by the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The number and kinds of solutions depend on the sign of the discriminant $b^2 - 4ac$

- if $b^2 - 4ac > 0$, the equation has two real roots
- if $b^2 - 4ac < 0$, the equation has no real roots (it has two complex roots)
- if $b^2 - 4ac = 0$, the equation has a repeated real root, $x = -b/2a$

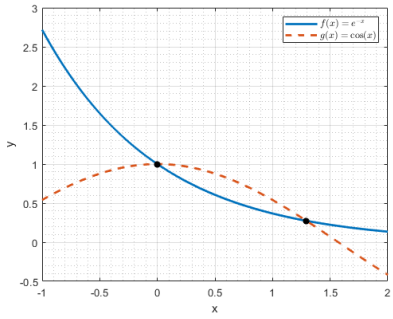
Determining the real roots of a polynomial of degree three or four using closed form formulas involving radicals is possible, though involved. For degrees five or higher, it is impossible to find a solution of a general polynomial involving radicals. This result is known as the *Abel-Ruffini* Theorem.

Example 3: Find all solutions of the equation $\sin(kx) = 0$, for some real number $k \neq 0$

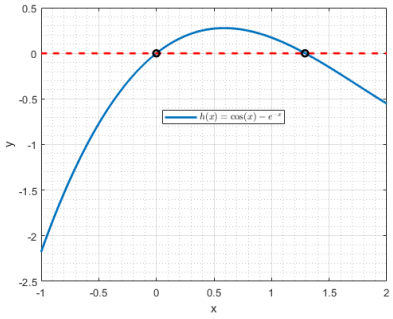
From the graph of sine, the roots are located at $kx = n\pi$ for any integer n. Therefore,

$$x = n\pi/k, \quad n \in \mathbb{Z}$$

generates all the solutions.



(a) Graph of $f(x) = e^{-x}$ and $g(x) = \cos(x)$.



(b) Graph of $h(x) = \cos(x) - e^{-x}$.

Figure 2: The roots of $h(x) = \cos(x) - e^{-x}$ are the points of intersection of the graphs of $g(x) = \cos(x)$ and $f(x) = e^{-x}$

Example 4: Find all roots of $h(x) = \cos(x) - e^{-x}$ in the interval $[-0.5, 1.5]$.

It is easy to see that $e^0 = \cos(0) = 1$ so that $r = 0$ is a solution of $h(x) = 0$. The graph in Figure 2 suggests there is another root in the interval $[1, 1.5]$. We find this root using the MATLAB function `fzero`

MATLAB fzero for finding roots of non-linear functions

```
>> f = @(x) cos(x)-exp(-x);  % function
>> x0 = 1;  % initial guess close to the presumed root
>> r = fzero(f, x0) % find root r
```

2.1 Bisection Method

In order to find a root, it is necessary to know that a root exists within a certain interval. This can be done by *bracketing* the root. Suppose $f(x)$ is a continuous function, and that on a certain interval $[a, b]$ the numbers $\{f(a), f(b)\}$ have *opposite* signs, then the graph of $f(x)$ must cross the x-axis somewhere at some r such that $a < r < b$. In other words if $f(a)f(b) < 0$ there exists an r in (a, b) with $f(r) = 0$. This is known as the Intermediate Value Theorem in analysis.

Theorem (Intermediate Value Theorem): Let f be a continuous function on $[a, b]$ satisfying $f(a)f(b) < 0$. Then f has a root r between a and b such that $a < r < b$ and $f(r) = 0$.

The example below shows how to use the Intermediate Value Theorem to find intervals containing roots.

Example 1: Show that $f(x) = e^{-x}\cos(5x)$ has roots in the intervals $[0.2, 0.4]$, $[0.8, 1.0]$ and $[1.4, 1.6]$.

a	b	$f(a)$	$f(b)$	sign$(f(a)f(b))$
0.2	0.4	0.4424	-0.2790	–
0.8	1.0	-0.2937	0.1044	–
1.4	1.6	0.1859	-0.0294	–

Table 2: The function $f(x) = e^{-x}\cos(5x)$ has three roots in the intervals $[a, b]$

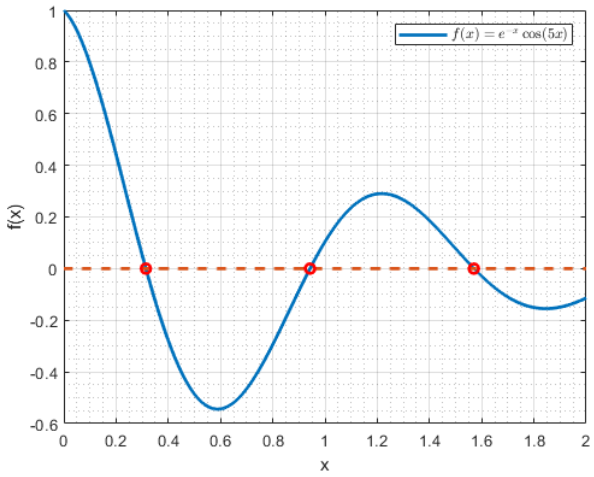


Figure 3: Graph of $f(x) = e^{-x}\cos(5x)$ depicting the roots.

To get better estimates of the roots, we need to “zoom" in at each root and take smaller and smaller intervals. This can be done by the *bisection algorithm.*

Bisection algorithm:

Given an initial interval $[a, b]$ such that $f(a)f(b) < 0$, tolerance `tol`
while $(b - a)/2 >$ `tol`
    $c = (b + a)/2$
    if $f(c) = 0$, **stop, end**
    if $f(a)f(c) < 0$
        $b = c$
    else
        $a = c$
    end
end

We now describe the bisection algorithm. Once an interval bracketing a root has been identified, the bisection algorithm is used to get a smaller and better interval to estimate the value of the root.

Inputs

- f a function whose roots are to be estimated
- a, b with $a < b$ for the interval $[a, b]$ bracketing a root.
- `tol`> 0 desirable tolerance for the width of the final interval.

Bisection process

- first estimate the root by the midpoint $c = (b + a)/2$
- if $f(c) = 0$, then we have found the root! The root is c
- otherwise $f(a)f(c) < 0$ or $f(c)f(b) < 0$
- if $f(a)f(c) < 0$, then the root is more to the left of the interval
- if the root is more to the left, change b to c, i.e. set $b = c$.
- otherwise if $f(c)f(b) < 0$, then the root is more to the right of the interval
- if the root is more to the right, change c to a, i.e. set $a = c$
- at this stage the interval is now half what we started with
- check now if width of the new interval $(b - a)/2 <$`tol`

- if the new width is bigger than `tol` repeat the steps above
- otherwise stop when $(b-a)/2 <$`tol`

Output

- estimated root c.

The above algorithm is robust and is guaranteed to converge to an approximate root as long as

- f is continuous
- the interval $[a, b]$ contains one root.
- $f(a)f(b) < 0$.

2.1.1 Convergence of bisection

How accurate and how fast is bisection? If $[a, b] := [a_0, b_0]$ is the starting interval, bisection produces a sequence of intervals $[a_0, b_0] \supset [a_1, b_1] \supset \cdots \supset [a_n, b_n]$. After n steps of the bisection algorithm, the interval $[a_n, b_n]$ at step n has size $(b-a)/2^n$. The best estimate of the root after n-steps is the midpoint of the interval $[a_n, b_n]$, i.e. $c_n = (b_n + a_n)/2 = (b-a)/2^{n+1}$. This shows that the solution error is

$$\text{bisection error} = |c_n - r| \leq \frac{(b-a)}{2^{n+1}}.$$

At what cost? The cost of bisection algorithm is the number of function evaluations required to arrive at c_n. First, there are the 2 evaluations $f(a)$ and $f(b)$. Then at each step, the evaluation of the midpoints c_k, $k = 1, \cdots, n$. So after n steps, there are $n + 2$ function evaluations.

$$\text{bisection function evaluations} = n + 2.$$

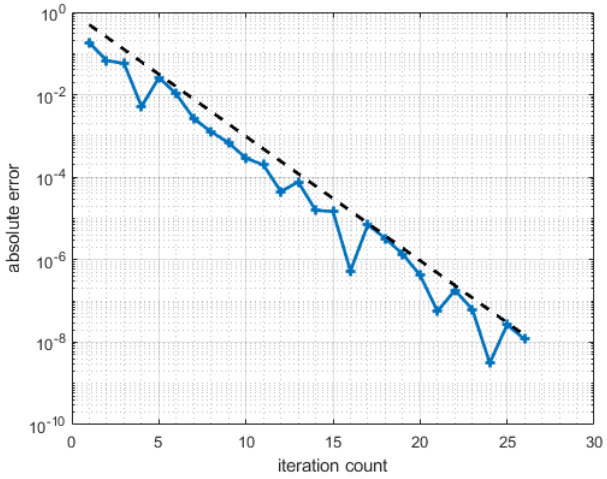


Figure 4: Convergence of bisection, $f(x) = x^3 + x - 1$ on the interval $[0, 1]$. Here tol $= 10^{-8}$ and $r = 0.682327803828019$. The dashed line is the maximum possible error at each iteration. The solid line is the absolute error.

MATLAB code for Bisection

```
% nmax is the maximum number of iterations allowed
% f = @(x) x.^3+x-1;
% a = 0; b = 1;
% tol = 1e-8;
% nmax = 100;
function [r,iter,err] = bisection(f,a,b,tol,nmax)
iter = 0;  % iteration count
err = zeros(1,nmax);
ex = fzero(f,b);   % exact solution
while ((b-a)/2>tol) && (iter < nmax)
    c = (b+a)/2;
    if f(a)*f(c)<0
        b = c;
    else
        a = c;
    end
    iter = iter + 1;
    err(iter) = abs(c - ex);
end
r = c;  % approximate root
err = err(1:iter);  % errors
```

n	c_n	$f(c_n)$	error
0	0.500000000000000	-0.375000000000000	0.182327803828019
1	0.750000000000000	0.171875000000000	0.067672196171981
2	0.625000000000000	-0.130859375000000	0.057327803828019
3	0.687500000000000	0.012451171875000	0.005172196171981
4	0.656250000000000	-0.061126708984375	0.026077803828019
⋮	⋮	⋮	⋮
23	0.682327866554260	1.50336848747656e-07	6.27262408681162e-08
24	0.682327806949615	7.48157225061163e-09	3.12159609272555e-09
25	0.682327777147293	-6.39460605578179e-08	2.66807262949698e-08
26	0.68232779204845	-2.82322445421812e-08	1.17795651011221e-08

Table 3: Convergence of bisection method for $f(x) = x^3 + x - 1$ on $[0, 1]$

Example: Use the Bisection method to find solutions accurate to within 10^{-2} for the function $x - 2^{-x} = 0$ on the interval $[0, 1]$.

A solution is accurate to within p decimal places if the error is less than 0.5×10^{-p}. In this case, the error is less than $0.5 \times 10^{-2} = 0.005$. First, let's determine how many steps are required by the bisection method. Recall the error formula

$$\frac{b-a}{2^{n+1}} < 0.5 \times 10^{-2}.$$

To solve for n with $a = 0, b = 1$, take natural log on both sides

$$\begin{aligned}
\ln\left(\frac{1-0}{2^{n+1}}\right) &< \ln(0.005) && (1)\\
\ln(1) - \ln(2^{n+1}) &< \ln(0.005) && (2)\\
-(n+1)\ln(2) &< \ln(0.005) && (3)\\
-(n+1) &< \ln(0.005)/\ln(2) \approx -7.64 && (4)\\
n &> 6.64 && (5)
\end{aligned}$$

The minimum number of iterations is $n = 7$.

2.2 Fixed-Point Iteration

A number p is called a **fixed point** of a function g if $g(p) = p$. In this section, we consider the problem of finding solutions to fixed point problems. Our aim is to solve equations of the form $f(x) = 0$, but in some instances, it may be advantageous to solve an equivalent fixed point problem instead.

n	a_n	b_n	c_n	sign$(f(a_n))$sign$(f(c_n))$	sign$(f(c_n))$sign$(f(b_n))$
0	0.0	1.0	0.5	+	-
1	0.5	1.0	0.75	-	+
2	0.5	0.75	0.625	+	-
3	0.625	0.75	0.6875	-	+
4	0.625	0.6875	0.65625	-	+
5	0.625	0.65625	0.640625	+	-
6	0.640625	0.65625	0.6484375	-	+
7	0.640625	0.6484375	0.64453125	-	+
8	0.640625	0.64453125	0.642573125		

Table 4: The solution of $x - 2^{-x} = 0$ is 0.64 to 2 decimal digits after 7 iterations.

- the equation $f(x) = 0$ is equivalent to $g(x) = x - f(x)$ or $g(x) = x + 4f(x)$ in the sense that if p is a root of f, then $f(p) = 0$. Thus $g(p) = p - f(p) = p$ or $g(p) = p + 4f(p) = p$, etc. So that p is also a fixed point of $g(x)$.
- if p is a fixed point of $g(x)$, then the function defined by $f(x) = x - g(x)$ has a root at p.

The following Theorem provides sufficient conditions for the existence and uniqueness of a fixed point.

Theorem: (Existence and Uniqueness of Fixed Points)

(i) If g is continuous on $[a, b]$ and $g(x) \in [a, b]$ for all $x \in [a, b]$, then g has at least one fixed point in $[a, b]$

(ii) If, in addition, $g'(x)$ exists on (a, b) and a positive constant $k < 1$ exists with

$$|g'(x) \leq k, \quad \text{for all } x \in (a, b),$$

then there is exactly one fixed point in $[a, b]$.

Exercise 1: Determine any fixed points of the function $g(x) = x^2 - 2$.

The fixed points of g are characterised by

$$g(x) = x^2 - 2 = x.$$

This equivalent to

$$x^2 - x - 2 = (x + 1)(x - 2) = 0.$$

So g has two fixed points $p = -1$ and $p = 2$. These are the points of intersection of the graph of g with the line $y = x$.

Exercise 2: Show that each of the following functions has a fixed point precisely when $f(p) = 0$ where $f(x) = x^4 + 2x^2 - x - 3$.

(a) $g_1(x) = (3 + x - 2x^2)^{1/4}$

$$x^4 + 2x^2 - x - 3 = 0$$
$$x^4 = 3 + x - 2x^2$$
$$\boxed{x = (3 + x - 2x^2)^{1/4}}$$

(b) $g_2(x) = \left(\frac{x+3-x^4}{2}\right)^{1/2}$

(c) $g_3(x) = \left(\frac{x+3}{x^2+2}\right)$

(d) $g_4(x) = \frac{3x^4+2x^2+3}{4x^3+4x+3}$
Hint: $g(x) = x - \alpha f(x)$ with $\alpha = 1/f'(x)$

Exercise 3: Find the fixed points of $g(x) = 2^{-x}$.

The fixed points of $g(x) = 2^{-x}$ are the solutions of the equation $x - 2^{-x} = 0$. These solutions cannot be determined in closed form. We can apply the **fixed point iteration** to find an approximate solution as follows

Fixed Point Iteration

Choose an initial guess p_0
$p_{i+1} = g(p_i)$ for $i = 0, 1, 2, 3, \cdots$

That is,

$$\begin{aligned} p_1 &= g(p_0) \\ p_2 &= g(p_1) = g(g(p_0))) \\ p_3 &= g(p_2) = g(g(g(p_0))) \\ &\vdots \end{aligned}$$

MATLAB code for Fixed-Point Iteration

```
% MATLAB code for Fixed Point Iteration
function [pc, iter] = fpi(g,p0,tol,nmax)
p = zeros(1,nmax);
p(1) = p0;
i = 1;
res = 1;
while (i < nmax)&& (res > tol)
    p(i+1)=g(p(i));
    res = abs(p(i+1)-p(i));
    i = i+1;
end
pc = p(i);
iter = i;
```

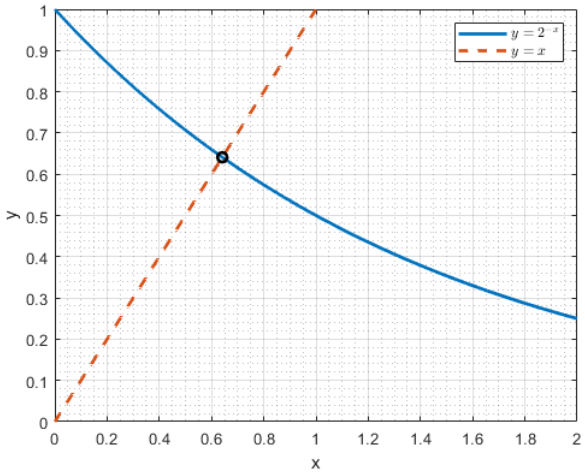


Figure 5: The fixed point of $g(x) = 2^{-x}$ is the point of intersection of the graph of $y = g(x)$ with the line $y = x$

If the sequence p_i converges to a number, p, and g is continuous, then fixed point iteration converges to a fixed point of g. To see this, note

$$g(p) = g\left(\lim_{i\to\infty} p_i\right) = \lim_{i\to\infty} g(p_i) = \lim_{i\to\infty} p_{i+1} = p$$

Exercise 4: Determine the first 8 iterates of the fixed point iteration applied to $g(x) = 2^{-x}$ starting with $x_0 = 1$.

n	x_i	$g(x_i)$	error
0	1.000000000000000	0.500000000000000	0.141185744504986
1	0.500000000000000	0.707106781186547	0.0659210366815615
2	0.707106781186547	0.612547326536066	0.0286384179689201
3	0.612547326536066	0.65404086004207	0.0128551155370835
4	0.654040866004207	0.635497845813374	0.00568789869161224
5	0.635497845813374	0.643718641722869	0.00253289721788308
6	0.643718641722869	0.64006102117724	0.00112472332774627
7	0.64006102117724	0.641685807042998	0.000500062538012269
8	0.641685807042998	0.640963537177963	0.000222207327022828

Table 5: Convergence of fixed point iteration for $g(x) = 2^{-x}$ with starting point $x_0 = 1.0$

The fixed point of g is also the root of $f(x) = x - 2^{-x}$ which is 0.64... from Table 4.

Exercise 5: Perform 6 iterations of the fixed point iteration on $g_1, \cdots, g_4$ from Exercise 2. Recall

$$g_1(x) = (3 + x - 2x^2)^{1/4}, \quad g_2(x) = \left(\frac{x + 3 - x^4}{2}\right)^{1/2}$$

$$g_3 = \frac{x+3}{x^2+2}, \quad g_4(x) = \frac{3x^4 + 2x^2 + 3}{4x^3 + 4x - 1}$$

n	$g_1(x)$	g_2	g_3	g_4
0	1.000000000000000	1.000000000000000	1.000000000000000	1.000000000000000
1	1.18920711500272	1.22474487139159	1.33333333333333	1.14285714285714
2	1.08005775266756	0.993666159077482	1.14705882352941	1.1244816900179
3	1.14967143058938	1.22856864527499	1.25071745369163	1.12412316394015
4	1.10782052951026	0.987506429150887	1.19258323697031	1.12412302970433
5	1.13393228450473	1.23218341831881	1.22509383726244	1.12412302970432
6	1.11800311771578	0.981585828612739	1.206874876498	1.12412302970432

Table 6: Convergence of fixed point iteration for different functions $g_1, \cdots, g_4$ corresponding to the roots of $f(x) = x^4 + 2x^2 - x - 3$. The function g_4 appears to give the best approximation to the solution.

2.2.1 Fixed-Point Theorem

How should we select a function so that fixed point iteration converges reliably and rapidly? The following Theorem provides clues.

Fixed-Point Theorem: Let g be a continuous function on $[a, b]$ satisfying $g(x) \in [a, b]$ for all $x \in [a, b]$. Suppose that g' exists on (a, b) and there exists a constant $0 < k < 1$ such that

$$|g'(x)| \leq k, \text{ for all } x \in (a, b).$$

Then for any number $p_0 \in [a, b]$ the sequence defined by

$$p_{n+1} = g(p_n), \quad n \geq 0$$

converges to the unique fixed point p in $[a, b]$.

Exercise 6: Show that $g(x) = 2^{-x}$ has a unique fixed point in $[0, 1]$.

Since $g'(x) = -2^{-x} \ln(2) < 0$ on $[0, 1]$ it follows that $g(x)$ is decreasing on $[0, 1]$. So

$$g(1) = 0.5 \leq g(x) \leq g(0) = 1.$$

Hence $g(x) \in [0, 1]$. Also

$$|g'(x)| = |2^{-x} \ln 2| \leq \ln 2 = 0.6931 < 1.$$

So $|g'(x)| < 1$ on $(0, 1)$. Then by the Fixed Point Theorem, for any number $p_0 \in [0, 1]$ the sequence

$$p_{n+1} = 2^{-p_n}, \quad n \geq 0$$

converges to a unique fixed point $p = 0.6411857445...$ in $[0, 1]$.

2.3 Newton's Method

Newton's Method (sometimes called Newton-Raphson) is one of the most powerful and well-known methods for solving a root-finding problem. Suppose we want to determine a root p such that $f(p) = 0$. To derive Newton's method, we make the following assumptions

- $f \in C^2[a, b]$, i.e. f is twice differentiable and $f''(x)$ is bounded for all $x \in (a, b)$
- there exists $p_0 \in [a, b]$ such that $|p - p_0|$ is small (p_0 is sufficiently close to the root p)

- and $f'(p_0) \neq 0$

Consider the Taylor polynomial of $f(x)$ expanded around p_0 and evaluated at $x = p$

$$f(p) = f(p_0) + (p - p_0)f'(p_0) + \frac{(p - p_0)^2}{2} f''(\xi)$$

where ξ lies between p and p_0.

Since p is a root of f, it follows that $f(p) = 0$. So that the equation above becomes

$$0 = f(p_0) + (p - p_0)f'(p_0) + \frac{(p - p_0)^2}{2} f''(\xi).$$

By assumption $|p - p_0|$ is small, so that $(p - p_0)^2/2$ is even smaller. Hence,

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

Solving for p, we get

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

Expanding p around p_1, we obtain p_2, etc. Continuing in this fashion, we generate a sequence $\{p_n\}_{n=0}^{\infty}$ by

$$p_{n+1} = p_n - \frac{f(p_n)}{f'(p_n)}, \quad \text{for} \quad n \geq 0,$$

provided that $f'(p_n) \neq 0$ for any $n \geq 0$.

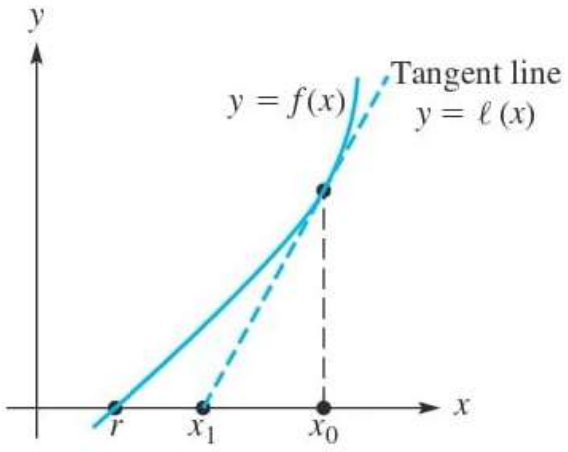


Figure 6: Graphical interpretation of Newton's method

n	x_n	$\lvert x_n - r \rvert$
0	1.000000000000000	0.358814255495014
1	0.628687207584368	0.012498536920618
2	0.641169034642714	0.000016709862272
3	0.641185744475211	0.000000000029775
4	0.641185744504986	0.000000000000000

Table 7: Convergence of Newton's method for the function $f(x) = x - 2^{-x}$.

MATLAB code for Newton's Method

```
function [r,iter] = newton(f,fp,p0,tol,nmax)
% f is the function
% fp is the derivative of f
% p0 is the initial iterate
% tol is the maximum tolerance, e.g. 1e-9
% nmax is the maximum number of iterations
res = 1; % residual
i = 1;
p = zeros(1,nmax);
p(1) = p0;
while (res>tol)&&(i <nmax)
    p(i+1) = p(i)-(f(p(i))/fp(p(i)));
    res = abs(p(i+1)-p(i));
    i = i+1;
end
r = p(1:i);
iter = i;
```

Exercise: Use Newton's method to find p_1, p_2 and p_3 for the equation $f(x) = x^3 + x - 1 = 0$ with a starting point $p_0 = 1$.

The derivative of f is $f'(x) = 3x^2 + 1$. Newton's method is

$$\begin{aligned} p_{n+1} &= p_n - \frac{f(p_n)}{f'(p_n)} \\ &= p_n - \frac{p_n^3 + p_n - 1}{3p_n^2 + 1} \\ &= \frac{3p_n^3 + p_n - (p_n^3 + p_n - 1)}{3p_n^2 + 1} \\ &= \frac{2p_n^3 + 1}{3p_n^2 + 1} \end{aligned}$$

Hence,

$$\begin{aligned} p_1 &= \frac{2(1^3) + 1}{3(1^2) + 1} = \frac{3}{4} = 0.75 \\ p_2 &= \frac{2(3/4)^3 + 1)}{3(3/4)^2 + 1} = \frac{118}{172} = 0.686046511627907 \\ p_3 &= \frac{2(118/172)^3 + 1}{3(118/172)^2 + 1} = 0.682339582597314. \end{aligned}$$

Exercise: Use Newton's method to approximate $\sqrt{2}$ to 5 decimal places.

The function $f(x) = x^2 - 2$ has roots at $\pm\sqrt{2}$. Applying Newton's method to f, we get

$$p_{n+1} = p_n - \frac{p_n^2 - 2}{2p_n} = \frac{2p_n^2 - (p_n^2 - 2)}{2p_n} = \frac{p_n^2 + 2}{2p_n} = \frac{1}{2}\left(p_n + \frac{2}{p_n}\right)$$

This is Heron's formula for approximating $\sqrt{2}$. Let $p_0 = 1$.

$$\begin{aligned} p_1 &= \frac{1}{2}\left(1 + \frac{2}{1}\right) = \frac{3}{2} = 1.5 \\ p_2 &= \frac{1}{2}\left(\frac{3}{2} + \frac{4}{3}\right) = \frac{17}{12} = 1.4166666\cdots \\ p_3 &= \frac{1}{2}\left(\frac{17}{12} + \frac{24}{17}\right) = \frac{577}{408} = 1.41421568627\cdots \\ p_4 = &= \frac{1}{2}\left(\frac{577}{408} + \frac{816}{577}\right) = 1.414213562374690 \end{aligned}$$

Hence $\sqrt{2} \approx 1.41421$ to 5 decimal places.

2.3.1 Convergence of Newton's Method

The examples above show that Newton's method converges remarkably quickly. For example, in Table 7, we start off with no correct digits at all, then successive iterations yield $2, 4, 9$ and 15 correct digits. We observe a doubling of the number of correct digits per iteration. In many cases, Newton's method converges to machine precision in 5 or 6 iterations. This doubling of the number of correct digits per iteration is known as **quadratic convergence**.

Formally, a sequence of real numbers $\{p_n\}$ **converges quadratically** to a number p if successive steps obey the inequality

$$|p - p_{n+1}| \leq C|p - p_n|^2$$

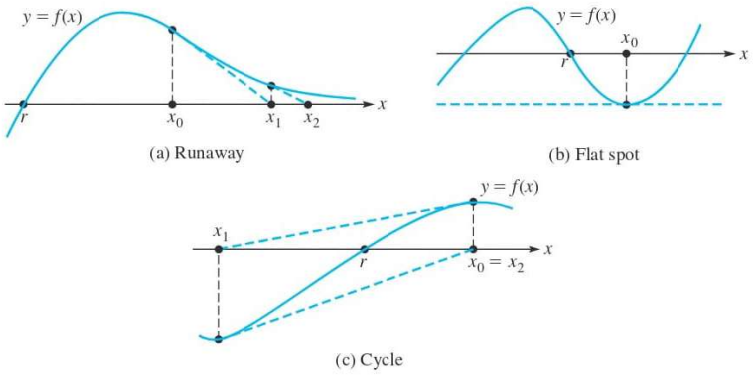


Figure 7: Failures of Newton's method can be avoided by a careful choice of initial iterate.

Theorem: (Newton's Method) Suppose f, f' and f'' are continuous in a neighbourhood of a root p of f, and that $f'(p) \neq 0$. Then there exists $\delta > 0$ such that if $|p - p_0| \leq \delta$ then all subsequent iterates p_n, satisfy $|p - p_n| \leq \delta$ for all $n \geq 1$ and converge quadratically to p, that is

$$|p - p_{n+1}| \leq C|p - p_n|^2$$

for some $C > 0$ depending on δ.

2.4 Secant Method

A major drawback of Newton's method is that the algorithm requires knowledge of the derivative of f. The **Secant method** replaces the derivative $f'(x)$ with a difference formula. Recall the difference approximation:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

Setting $x = p_n$, and $h = p_{n-1} - p_n$, we obtain the approximation

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

Replacing the above into Newton's formula yields the **secant method**

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

Note that p_{n+1} depends on two previous values p_n and p_{n-1}, so to start the secant method, we require p_0 and p_1.

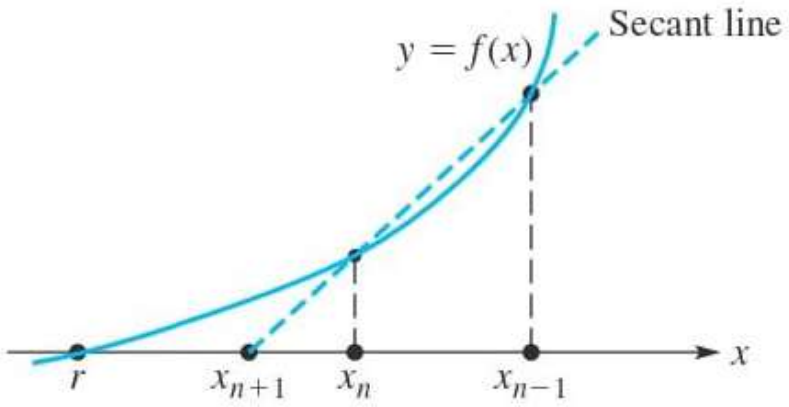


Figure 8: Interpretation of the secant method. $f'(x)$ is approximated by a secant line.

Convergence of the Secant method

It can be shown that the basic secant method converges with a rate $\alpha = \frac{1}{2}(1+\sqrt{5}) \approx 1.62$. That is, if r is a root, and $\{p_n\}$, $n \geq 1$ are the iterates generated by the secant method, then there exists $C > 0$ such that

$$|r - p_{n+1}| \leq C|r - p_n|^{\alpha}.$$

Since $\alpha > 1$, the convergence is said to be **superlinear**. In general, the speed of convergence of the secant method is between that of bisection and Newton's method.

MATLAB code for the Secant Method

```
function [r,iter] = secant(f,p0,p1,tol,nmax)
p = zeros(1,nmax);
p(1) = p0;
p(2) = p1;
i = 2;
res = 1;
while (res>tol)&&(i<nmax)
    p(i+1) = p(i)-f(p(i))*(p(i-1)-p(i))/(f(p(i-1))-f(p(i)));
    res = abs(p(i+1)-p(i));
    i = i+1;
end
r = p(1:i); % vector of iterates
iter = i; % number of iterations
```

n	p_n	$\lvert p_n - r\rvert$
0	1.000000000000000	0.317672196171982
1	0.500000000000000	0.182327803828018
2	0.646446609406726	0.005260864901740
3	0.641266292863391	0.000080548358405
4	0.641185699347306	0.000000045157680
5	0.641185744505374	0.000000000000388
6	0.641185744504986	0.000000000000000

Table 8: Convergence of the Secant method for $f(x) = x - 2^{-x}$ with $p_0 = 1$ and $p_1 = 0.5$. The secant method is in general faster than bisection but slower than Newton's method.

3 Solution of Linear Systems

In this chapter, we will consider methods for solving matrix systems of the form $A\mathbf{x} = \mathbf{b}$ where A is a matrix of size $n \times n$ and $\mathbf{b}$ is a vector of size n for an unknown vector $\mathbf{x}$ of size n. The numerical solution of linear systems, also called **numerical linear algebra**, is a central problem of numerical analysis. Many numerical problems, for example solving differential equations, ultimately lead to a solution of large linear systems. For example, discretizing a differential equation may lead to a linear system of size $10,000$ by $10,000$ or larger, and it is impossible to solve such a system by hand. Standard methods of solving linear equations that involve computing the determinant and cofactors become prohibitively expensive for large n. In this section, we will learn several algorithms for solving large linear systems reliably and inexpensively.

Example: Solve the linear system

$$\begin{aligned} x + 2y &= -1 \\ 2x + 3y &= 1 \end{aligned}$$

This is a system of simultaneous equations in x and y. One way to solve this system is, for example, to solve for x in the first equation

$$x = -1 - 2y$$

and then substitute into the second equation

$$\begin{aligned} 2(-1 - 2y) + 3y &= 1 \\ -2 - 4y + 3y &= 1 \\ -2 - y &= 1 \\ y &= -3. \end{aligned}$$

Now substituting y back into the first equation gives $x = -1 - 2(-3) = 5$. Hence the system has solution $x = 5, y = -3$.

If the number of variables is large, then solving linear equations this way becomes cumbersome. To overcome this problem, let us write the system in matrix-vector form.

$$\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

We will introduce the following notation:

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Then the matrix system can be written compactly as $A\mathbf{x} = \mathbf{b}$.

The solution of the matrix equation is then $\mathbf{x} = A^{-1}\mathbf{b}$ where A^{-1} is the inverse of the matrix A. This shows that the matrix equation can be solved by determining A^{-1} followed by a matrix multiplication by the vector $\mathbf{b}$. Recall from linear algebra that the inverse of a 2×2 matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is given by

$$A^{-1} = \frac{1}{\det(A)}\text{adj}(A) = \frac{1}{ad - bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

Here $ad - bc = \det(A)$ is the determinant of A, and $\begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \text{adj}(A)$ is the adjoint matrix of A.

Given $A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$ we have $\det(A) = 1 \cdot 3 - 2 \cdot 2 = -1$, and

$$A^{-1} = \frac{1}{-1}\begin{pmatrix} 3 & -2 \\ -2 & 1 \end{pmatrix} = \begin{pmatrix} -3 & 2 \\ 2 & -1 \end{pmatrix}.$$

Hence

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -3 & 2 \\ 2 & -1 \end{pmatrix}\begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ -3 \end{pmatrix}.$$

This method of solving linear systems by computing the determinant becomes expensive for large values of n. The formula for the determinant of an $n \times n$ matrix involves $n!$ terms. For example, if $n = 10$ then the number of terms required to be computed is $10! = 3,628,800$. Let us consider less expensive methods for solving linear systems.

Example: Solve the following linear system

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \\ 3 \end{pmatrix}$$

A is a diagonal matrix since the only non-zero terms are the diagonal ones. This system is particularly easy to solve. The solution is

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 2/3, \quad x_4 = 1/4, \quad x_5 = 3/5.$$

Example: Solve the linear system

$$\begin{aligned} 3x - 4y + 5z &= 2 \\ 3y - 4z &= -1 \\ 5z &= 5 \end{aligned}$$

From the last equation, we see that $z = 1$. Substituting into the second equation, $3y - 4(1) = -1$, we solve for y to get $y = 1$. Finally, we solve for x from the first equation $3x - 4(1) + 5(1) = 2$. That is $x = 1/3$. This solution process is called **back substitution** because the equations are solved from the bottom up.

In matrix-vector form, the system above is as follows:

$$\begin{pmatrix} 3 & -4 & 5 \\ 0 & 3 & -4 \\ 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 5 \end{pmatrix}$$

The matrix $A = \begin{pmatrix} 3 & -4 & 5 \\ 0 & 3 & -4 \\ 0 & 0 & 5 \end{pmatrix}$ is called **upper triangular** since the non-zero entries of A form a triangle on and above the diagonal. A matrix is **lower triangular** if all the non-zero entries are on and below the diagonal. The example above shows that solving an upper or lower triangular matrix is relatively easy.

3.1 Gaussian Elimination

Gaussian elimination is an efficient way of computing the solution of a system of n equations in n unknowns. Three operations can be applied to a linear system to yield an equivalent system without changing the solutions.

(i) interchange one equation for another

(ii) add or subtract a multiple of one equation from another

(iii) multiply an equation by a nonzero constant

These three operations can be applied to a linear system to yield an equivalent system that is **upper triangular** and hence easier to solve using back substitution.

Example: Apply Gaussian elimination to solve

$$\begin{aligned} 6x_1 - 2x_2 + 2x_3 + 4x_4 &= 16 \\ 12x_1 - 8x_2 + 6x_3 + 10x_4 &= 26 \\ 3x_1 - 13x_2 + 9x_3 + 3x_4 &= -19 \\ -6x_1 + 4x_2 + x_3 - 18x_4 &= -34 \end{aligned}$$

In the first step, multiples of the first equation are subtracted from the second, third, and forth equations to eliminate x_1 from these equations.

(1) Multiply the first equation by 2

$$2 \cdot [6x_1 - 2x_2 + 2x_3 + 4x_4 = 16] = 12x_1 - 4x_2 + 4x_3 + 8x_4 = 32$$

and subtract the result from the second equation

$$-4x_2 + 2x_3 + 2x_4 = -6.$$

Multiply the first equation by $\frac{1}{2}$

$$\frac{1}{2} \cdot [6x_1 - 2x_2 + 2x_3 + 4x_4 = 16] = 3x_1 - x_2 + x_3 + 2x_4 = 8$$

and substract the result from the third equation

$$-12x_2 + 8x_3 + x_4 = -27.$$

Finally, add the first equation to the fourth

$$2x_2 + 3x_3 - 14x_4 = -18.$$

The resulting linear system is

$$\begin{aligned} 6x_1 - 2x_2 + 2x_3 + 4x_4 &= 16 \\ -4x_2 + 2x_3 + 2x_4 &= -6 \\ -12x_2 + 8x_3 + x_4 &= -27 \\ 2x_2 + 3x_3 - 14x_4 &= -18 \end{aligned}$$

(2) To eliminate x_2 from equations 3 and 4, multiply the second equation above by 3

$$3 \cdot [-4x_2 + 2x_3 + 2x_4 = -6] = -12x_2 + 6x_3 + 6x_4 = -18$$

and subtract the result from the third equation

$$2x_3 - 5x_4 = -9.$$

Multiply the last equation by 2

$$4x_2 + 6x_3 - 28x_4 = -36$$

and add the result to the second equation

$$8x_3 - 26x_4 = -42.$$

The resultant system is as follows:

$$\begin{aligned} 6x_1 - 2x_2 + 2x_3 + 4x_4 &= 16 \\ -4x_2 + 2x_3 + 2x_4 &= -6 \\ 2x_3 - 5x_4 &= -9 \\ 8x_3 - 26x_4 &= -42 \end{aligned}$$

(3) To eliminate x_3 from the last equation, multiply the third equation by 4

$$4 \cdot [2x_3 - 5x_4 = -9] = 8x_3 - 20x_4 = -36$$

and subtract from the last equation to get

$$-6x_3 = -6$$

. The resulant system is

$$\begin{aligned} 6x_1 - 2x_2 + 2x_3 + 4x_4 &= 16 \\ -4x_2 + 2x_3 + 2x_4 &= -6 \\ 2x_3 - 5x_4 &= -9 \\ -6x_3 &= -6 \end{aligned}$$

(4) This completes the first step of **forward elimination**. The next step is **backward substitution**. Starting from the last equation

$$x_4 = \frac{-6}{-6} = 1.$$

Taking $x_4 = 1$, solve for x_3 from the third equation

$$2x_3 - 5 = -9,$$

hence

$$x_3 = \frac{-9+5}{2} = -2.$$

Taking $x_3 = -2$ and $x_4 = 1$, solve for x_2 from the second equation

$$-4x_2 + 2 \cdot -2 + 2 \cdot 1 = -6,$$

so that

$$x_2 = 1.$$

Finally, substituting $x_2 = 1, x_3 = -2$ and $x_4 = 1$ into the first equation

$$6x_1 - 2 \cdot 1 + 2 \cdot -2 + 4 \cdot 1 = 16.$$

Solving for x_1 yields

$$x_1 = 3.$$

The solution is

$$x_1 = 3, x_2 = 1, x_3 = -2, \text{and } x_4 = 1.$$

Check in MATLAB.

```
>> A = [6,-2,2,4; 12, -8, 6, 10; 3,-13,9,3; -6,4,1,-18];
>> b = [16; 26; -19; -34];
>> x = A\b % solve by backslash
x =
     3
     1
     -2
     1
```

The same elimination can work without carrying variables by using the tableau form:

$$\left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 12 & -8 & 6 & 10 & 26 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right) \xrightarrow{R2-2R1} \left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & -16 \\ 0 & -4 & 2 & 2 & -6 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right) \xrightarrow{R3-\frac{1}{2}R1} \left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & -12 & 8 & 1 & -27 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & -12 & 8 & 1 & -27 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right) \xrightarrow{R4+R1} \left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & -12 & 8 & 1 & -27 \\ 0 & 2 & 3 & -14 & -18 \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & -12 & 8 & 1 & -27 \\ 0 & 2 & 3 & -14 & -18 \end{array}\right) \xrightarrow{R3-3R2} \left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 2 & 3 & -14 & -18 \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 2 & 3 & -14 & -18 \end{array}\right) \xrightarrow{R4+\frac{1}{2}R2} \left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 0 & 4 & -13 & -21 \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 0 & 4 & -13 & -21 \end{array}\right) \xrightarrow{R4-2R3} \left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 0 & -4 & 2 & 2 & -6 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 0 & 0 & -3 & -3 \end{array}\right)$$

Returning to the equations

$$\begin{aligned} 6x_1 - 2x_2 + 2x_3 + 4x_4 &= 16 \\ -4x_2 + 2x_3 + 2x_4 &= -6 \\ 2x_3 - 5x_4 &= -9 \\ -3x_4 &= -3 \end{aligned}$$

By back substitution,

$$x_4 = \frac{-3}{-3} = 1$$

$$2x_3 - 5 = -9 \Longrightarrow x_3 = -2$$

$$-4x_2 + 2 \cdot -2 + 2 \cdot 1 = -6 \Longrightarrow x_2 = 1$$

$$6x_1 - 2 \cdot 1 + 2 \cdot -2 + 4 \cdot 1 = 16 \Longrightarrow x_1 = 3.$$

That is

$$x_1 = 3, \;\; x_2 = 1, \;\; x_3 = -2, \;\; x_4 = 1.$$

3.2 LU Factorization

The LU factorization is a matrix representation of Gaussian elimination.

$$A = LU$$

where U is the upper triangular matrix obtained from the Gaussian elimination process, and L is a lower triangular matrix.

Example: Find the LU factorisation of the matrix $A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$. The elimination steps proceed as follows

$$\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \xrightarrow{R2-2R1} \begin{pmatrix} 1 & 2 \\ 0 & -1 \end{pmatrix} = U$$

Define L to be the lower triangular matrix with 1's on the diagonal, and the multiplier 2 in the $(2,1)$ location

$$L = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

Let us check if $A = LU$

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

Example: Find the LU factorization of

$$A = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix}$$

The upper triangular matrix obtained by Gaussian elimination is

$$U = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{pmatrix}$$

The lower triangular matrix L is obtained by placing 1's on the diagonal and the multipliers in the lower triangle

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

3.2.1 LU factorization and solution of linear systems

We can use the LU factorization to solve a linear system $A\mathbf{x} = \mathbf{b}$ using the following steps

(i) Solve $L\mathbf{y} = \mathbf{b}$ for $\mathbf{y}$

(ii) Use $\mathbf{y}$ from step (i) to solve $U\mathbf{x} = \mathbf{y}$ for $\mathbf{x}$.

This follows from $A\mathbf{x} = \mathbf{b} \implies LU\mathbf{x} = \mathbf{b} \implies L(U\mathbf{x}) = \mathbf{b}$. Hence $L\mathbf{y} = \mathbf{b}$ if $U\mathbf{x} = \mathbf{y}$.

Exercise: Use an LU factorization to solve the linear system

$$\begin{aligned} x_1 + 2x_2 &= -1 \\ 2x_1 + 3x_2 &= 1 \end{aligned}$$

In matrix form $A\mathbf{x} = \mathbf{b}$, with $\mathbf{x} = (x_1, x_2)$

$$\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

The LU factorization of A is

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

(i) Solve $L\mathbf{y} = \mathbf{b}$ for $\mathbf{y} := (y_1, y_2)$

$$\begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

We get $y_1 = -1$ and $2 \cdot -1 + y_2 = 1 \longrightarrow y_2 = 3$. So $\mathbf{y} = \begin{pmatrix} -1 \\ 3 \end{pmatrix}$

(ii) Solve for $\mathbf{x}$ in $U\mathbf{x} = \mathbf{y}$

$$\begin{pmatrix} 1 & 2 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -1 \\ 3 \end{pmatrix}$$

The solutions are $x_2 = -3$ and $x_1 + 2 \cdot -3 = -1$ or $x_1 = 5$. That is $\mathbf{x} = \begin{pmatrix} 5 \\ -3 \end{pmatrix}$

Exercise: Use an LU factorization to solve the linear system

$$\begin{aligned} 6x_1 - 2x_2 + 2x_3 + 4x_4 &= 16 \\ 12x_1 - 8x_2 + 6x_3 + 10x_4 &= 26 \\ 3x_1 - 13x_2 + 9x_3 + 3x_4 &= -19 \\ -6x_1 + 4x_2 + x_3 - 18x_4 &= -34 \end{aligned}$$

Recall that the LU factorization of the above system is

$$A = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{pmatrix} \begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{pmatrix}$$

(i) Solve $L\mathbf{y} = \mathbf{b}$ for $\mathbf{y} = (y_1, y_2, y_3, y_4)$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 16 \\ 26 \\ -19 \\ -34 \end{pmatrix}$$

$$y_1 = 16,\ 2 \cdot 16 + y_2 = 26 \Longrightarrow y_2 = -6,\ \frac{1}{2} \cdot 16 + 3 \cdot -6 + y_3 - 19 \Longrightarrow y_3 = -9$$

$$-16 - \left(\frac{1}{2} \cdot -6\right) + 2 \cdot -9 + y_4 = -34 \Longrightarrow y_4 = -3.$$

That is

$$\mathbf{y} = (16, -6, 9, -3)$$

(ii) Solve for $\mathbf{x}$ in $U\mathbf{x} = \mathbf{y}$

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 16 \\ -6 \\ -9 \\ -3 \end{pmatrix}$$

Solving for $\mathbf{x} = (x_1, x_2, x_3, x_4)$

$$x_1 = 3,\ \ x_2 = 1,\ \ x_3 = -2,\ \ x_4 = 1.$$

MATLAB code for Naive Gaussian Elimination

```
function x = gauss_elim(A,b)
% naive Gaussian elimination
% A is an n by n matrix
% b is an n vector
% solves Ax = b
n = size(A,1);
for j=1:n-1
    if abs(A(j,j))<eps
        error('zero pivot encountered');
    end
    for i = j+1:n
        mult = A(i,j)/A(j,j);
        for k = j:n
            A(i,k) = A(i,k)-mult*A(j,k);
        end
        b(i) = b(i)-mult*b(j);
    end
end
% back substitution
x = zeros(n,1);
for i = n:-1:1
    for j=i+1:n
        b(i)=b(i)-A(i,j)*x(j);
    end
    x(i)=b(i)/A(i,i);
end
```

Failures of Naive Gaussian Elimination

The process of Gaussian elimination cannot be applied unchanged to all nonsingular matrices.

(i) Consider the system

$$\begin{aligned} 0x_1 + x_2 &= 1 \\ x_1 + x_2 &= 2 \end{aligned}$$

The system matrix is $A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$. Gaussian elimination applied to A breaks down immediately since $a_{11} = 0$. A remedy for this is to interchange

the rows, while remembering to interchange the values of the right hand side vector.

$$\begin{aligned} x_1 + x_2 &= 2 \\ 0x_1 + x_2 &= 1 \end{aligned}$$

Then

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

can be solved by Gaussian elimination.

(ii) Let ε be a very small number (e.g. $\varepsilon = 10^{-20}$). Let us apply Gaussian elimination to the system

$$\begin{aligned} \varepsilon x_1 + x_2 &= 1 \\ x_1 + x_2 &= 2 \end{aligned}$$

Multiplying the first equation by $1/\varepsilon$ and subtracting from the second, we obtain

$$\begin{aligned} \varepsilon x_1 + x_2 &= 1 \\ (1 - \varepsilon^{-1})x_2 &= 2 - \varepsilon^{-1} \end{aligned}$$

In the back substitution phase,

$$x_2 = \frac{2 - \varepsilon^{-1}}{1 - \varepsilon^{-1}} \approx 1, \quad x_1 = \varepsilon^{-1}(1 - x_2) \approx 0.$$

Since $\varepsilon^{-1} = 10^{20}$ is very large, both $2 - \varepsilon^{-1}$ and $1 - \varepsilon^{-1}$ will be represented by the same number $-\varepsilon^{-1}$. For very small values of ε, the computer will calculate

$$x_2 \approx 1, \quad x_1 \approx 0.$$

The actual solution of the system is

$$x_2 = 2 - x_1 \Longrightarrow \varepsilon x_1 + (2 - x_1) = 1 \Longrightarrow x_1 = \frac{-1}{\varepsilon - 1} \approx 1$$

and

$$x_2 \approx 1.$$

Now let us interchange the rows and apply Gaussian elimination

$$\begin{aligned} x_1 + x_2 &= 2 \\ \varepsilon x_1 + x_2 &= 1 \end{aligned}$$

Multiplying the first equation by ε and subtracting from the second,

$$\begin{aligned} x_1 + x_2 &= 2 \\ (1-\varepsilon)x_2 &= 1 - 2\varepsilon \end{aligned}$$

Using back substitution,

$$x_2 = \frac{1-2\varepsilon}{1-\varepsilon} \approx 1, \quad x_1 = 2 - x_2 \approx 1.$$

This is a good approximation of the exact solution.

3.2.2 Gaussian Elimination with Partial Pivoting

The first step is to compare numbers in the first column before carrying out the elimination step. The largest entry in the first column is located and its row is swapped with the first row. The same check is applied for every choice of pivot during the elimination. This is called Gaussian elimination with **partial pivoting**.

Exercise: Apply Gaussian elimination with partial pivoting to solve

$$\begin{aligned} 6x_1 - 2x_2 + 2x_3 + 4x_4 &= 16 \\ 12x_1 - 8x_2 + 6x_3 + 10x_4 &= 26 \\ 3x_1 - 13x_2 + 9x_3 + 3x_4 &= -19 \\ -6x_1 + 4x_2 + x_3 - 18x_4 &= -34 \end{aligned}$$

Let us write the system in tableau form

$$\left(\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 16 \\ 12 & -8 & 6 & 10 & 26 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right)$$

The first column has coefficients $6, 12, 3, -6$. The second column has the largest number in absolute value, hence interchange rows 1 and 2, and carry out elimination

$$\left(\begin{array}{cccc|c} 12 & -8 & 6 & 10 & 26 \\ 6 & -2 & 2 & 4 & 16 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right) \xrightarrow{R2-\frac{1}{2}R1} \left(\begin{array}{cccc|c} 12 & -8 & 6 & 10 & 26 \\ 0 & 2 & -1 & -1 & 3 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 12 & -8 & 6 & 10 & 26 \\ 0 & 2 & -1 & -1 & 3 \\ 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \end{array}\right) \xrightarrow{R3-\frac{1}{4}R1} \left(\begin{array}{cccc|c} 12 & -8 & 6 & 10 & 26 \\ 0 & 2 & -1 & -1 & 3 \\ 0 & -11 & \frac{15}{2} & \frac{1}{2} & -\frac{51}{2} \\ -6 & 4 & 1 & -18 & -34 \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 12 & -8 & 6 & 10 & 26 \\ 0 & 2 & -1 & -1 & 3 \\ 0 & -11 & \frac{15}{2} & \frac{1}{2} & -\frac{51}{2} \\ -6 & 4 & 1 & -18 & -34 \end{array}\right) \xrightarrow{R4+\frac{1}{2}R1} \left(\begin{array}{cccc|c} 12 & -8 & 6 & 10 & 26 \\ 0 & 2 & -1 & -1 & 3 \\ 0 & -11 & \frac{15}{2} & \frac{1}{2} & -\frac{51}{2} \\ 0 & 0 & 4 & -13 & -21 \end{array}\right)$$

Now the numbers in the second column on the diagonal and below are $\{2, -11, 0\}$. The largest in absolute value is -11, so interchange the second row with the third row.

$$\left(\begin{array}{cccc|c} 12 & -8 & 6 & 10 & 26 \\ 0 & -11 & \frac{15}{2} & \frac{1}{2} & -\frac{51}{2} \\ 0 & 2 & -1 & -1 & 3 \\ 0 & 0 & 4 & -13 & -21 \end{array}\right) \xrightarrow{R3+\frac{11}{2}R2} \left(\begin{array}{cccc|c} 12 & -8 & 6 & 10 & 26 \\ 0 & -11 & \frac{15}{2} & \frac{1}{2} & -\frac{51}{2} \\ 0 & 0 & \frac{4}{11} & -\frac{10}{11} & -\frac{18}{11} \\ 0 & 0 & 4 & -13 & -21 \end{array}\right)$$

The numbers in the third column on the diagonal and below are $\{\frac{4}{11}, 4\}$. The largest one is 4, so swap the third and fourth rows.

$$\left(\begin{array}{cccc|c} 12 & -8 & 6 & 10 & 26 \\ 0 & -11 & \frac{15}{2} & \frac{1}{2} & -\frac{51}{2} \\ 0 & 0 & 4 & -13 & -21 \\ 0 & 0 & \frac{4}{11} & -\frac{10}{11} & -\frac{18}{11} \end{array}\right) \xrightarrow{R4-\frac{1}{11}R3} \left(\begin{array}{cccc|c} 12 & -8 & 6 & 10 & 26 \\ 0 & -11 & \frac{15}{2} & \frac{1}{2} & -\frac{51}{2} \\ 0 & 0 & 4 & -13 & -21 \\ 0 & 0 & 0 & \frac{3}{11} & \frac{3}{11} \end{array}\right)$$

Using backwards substitution,

$$\frac{3}{11}x_4 = \frac{3}{11} \Longrightarrow x_4 = 1$$

$$4x_3 - 13 \cdot 1 = -21 \Longrightarrow 4x_3 = -8 \Longrightarrow x_3 = -2$$
$$-11x_2 + \left(\frac{15}{2} \cdot -2\right) + \frac{1}{2} \cdot 1 = -\frac{51}{2} \Longrightarrow x_2 = 1$$
$$12x_1 - 8 \cdot 1 + 6 \cdot -2 + 10 \cdot 1 = 26 \Longrightarrow 12x_1 = 36 \Longrightarrow x_1 = 3.$$

4 Iterative Methods for Solving Linear Systems

Gaussian elimination is an example of a so called **direct method** for the solution of a linear system. In theory, direct methods give an exact solution of a non-singular matrix system in a finite number of steps. In practice, due to round-off errors, the result is only approximate. Direct methods are contrasted with **iterative methods** which can be applied to solving a linear system of equations. Like fixed-point iteration, an iterative method for solving linear systems begins with an approximate guess which is refined at each iteration, converging to the solution vector.

4.1 Jacobi Iteration

The Jacobi method is a form of fixed-point iteration for a system of equations. The first step is to rewrite the equation to solve the ith equation for the ith unkown. Then iterate starting with an initial guess.

Example 4.1: Apply four steps of the Jacobi iteration to approximate the solution of the linear system

$$\begin{aligned} 3x + y - z &= 4 \\ 2x + 4y + z &= 1 \\ -x + 2y + 5z &= 1 \end{aligned}$$

Let's start by solving the first equation for x, the second equation for y, and the last equation for z.

$$
\begin{aligned}
x &= \frac{4+z-y}{3} \\
y &= \frac{1-z-2x}{4} \\
z &= \frac{1+x-2y}{5}
\end{aligned}
$$

Let us use the initial guess $\mathbf{x}_0 = (x_0, y_0, z_0) = (0, 0, 0)$

$$
\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}
$$

$$
\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{bmatrix} \frac{4+z_0-y_0}{3} \\ \frac{1-z_0-2x_0}{4} \\ \frac{1+x_0-2y_0}{5} \end{bmatrix} = \begin{bmatrix} \frac{4+0-0}{3} \\ \frac{1-0-2\cdot 0}{4} \\ \frac{1+0-2\cdot 0}{5} \end{bmatrix} = \begin{pmatrix} \frac{4}{3} \\ \frac{1}{4} \\ \frac{1}{5} \end{pmatrix} \approx \begin{pmatrix} 1.3333 \\ 0.2500 \\ 0.200 \end{pmatrix}
$$

$$
\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{bmatrix} \frac{4+z_1-y_1}{3} \\ \frac{1-z_1-2x_1}{4} \\ \frac{1+x_1-2y_1}{5} \end{bmatrix} = \begin{bmatrix} \frac{4+1/5-1/4}{3} \\ \frac{1-1/5-2\cdot 4/3}{4} \\ \frac{1+4/3-2\cdot 1/4}{5} \end{bmatrix} = \begin{pmatrix} \frac{79}{60} \\ -\frac{7}{15} \\ \frac{11}{30} \end{pmatrix} \approx \begin{pmatrix} 1.31667 \\ -0.4667 \\ 0.3667 \end{pmatrix}
$$

$$
\begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix} = \begin{bmatrix} \frac{4+z_2-y_2}{3} \\ \frac{1-z_2-2x_2}{4} \\ \frac{1+x_2-2y_2}{5} \end{bmatrix} = \begin{bmatrix} \frac{4+11/30+7/15}{3} \\ \frac{1-11/30-2\cdot 79/60}{4} \\ \frac{1+79/60+2\cdot 7/15}{5} \end{bmatrix} = \begin{pmatrix} \frac{29}{18} \\ -\frac{1}{2} \\ \frac{13}{20} \end{pmatrix} \approx \begin{pmatrix} 1.6111 \\ -0.5000 \\ 0.6500 \end{pmatrix}
$$

$$
\begin{pmatrix} x_4 \\ y_4 \\ z_4 \end{pmatrix} = \begin{bmatrix} \frac{4+z_3-y_3}{3} \\ \frac{1-z_3-2x_3}{4} \\ \frac{1+x_3-2y_3}{5} \end{bmatrix} = \begin{bmatrix} \frac{4+13/20+1/2}{3} \\ \frac{1-13/20-2\cdot 29/18}{4} \\ \frac{1+29/18+2\cdot 1/2}{5} \end{bmatrix} = \begin{pmatrix} \frac{103}{60} \\ -\frac{517}{720} \\ \frac{13}{18} \end{pmatrix} \approx \begin{pmatrix} 1.71667 \\ -0.7180555 \\ 0.72222 \end{pmatrix}
$$

The iteration converges to the vector $\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$ for large n.

MATLAB code for Jacobi Iteration

```
function [x,iter,err] = jacobi(A,b,tol,nmax)
% Jacobi iteration
% A is strictly diagonally dominant
n = size(A,1); % size of the matrix
D = diag(diag(A)); % matrix of diagonal
invD = diag(1./diag(A)); % inverse of diagonal matrix
```

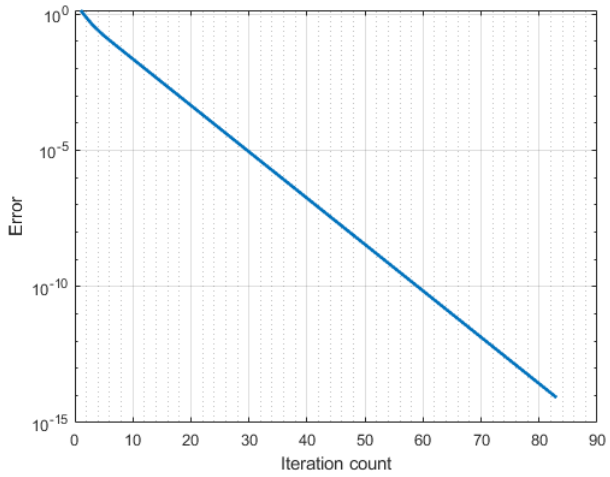


Figure 9: Semilog plot of $|\mathbf{x}_{n+1} - \mathbf{x}_n|$ vs number of iterations for the Jacobi iteration for Example 4.1.

```
Q = A-D; % this is L+U
i = 1; % iteration count
res = 1; % residual
x_old = zeros(n,1);
err = zeros(nmax,1); % error vector
while (res>tol) && (i<nmax)
    x = invD*(b-Q*x_old); % Jacobi iteration
    res = norm(x-x_old); % residual
    err(i)=res;
    i = i+1;
    x_old = x;
end
iter = i;
err = err(1:i);
```

Example 4.2: Apply the Jacobi iteration to approximate the solution of the following system:

$$\begin{aligned} 2x + 4y + z &= 1 \\ 3x + y - z &= 4 \\ -x + 2y + 5z &= 1 \end{aligned}$$

Note that this is the same system as in the previous example with the first 2

equations swapped. Solving for x, y and z

$$\begin{aligned} x &= \frac{1-z-4y}{2} \\ y &= 4+z-3x \\ z &= \frac{1+x-2y}{5} \end{aligned}$$

Applying the Jacobi iteration with initial vector $(x_0, y_0, z_0) = (0, 0, 0)$, we get the first few iterates

$$\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 4.0 \\ 0.2 \end{pmatrix}, \quad \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} -7.6 \\ 2.7 \\ -1.3 \end{pmatrix}, \quad \begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix} = \begin{pmatrix} -4.25 \\ 25.5 \\ -2.4 \end{pmatrix}$$

show that the sequence diverges. What conditions should be satisfied for the Jacobi iteration to work?

Definition: A matrix $A = (a_{ij})$ is said to be **strictly diagonally dominant** if for each $1 \leq i \leq n$, $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$. In other words, the magnitude of the diagonal entry is larger than the sum of the magnitudes of all the other entry in each row.

For example, the matrix $A = \begin{pmatrix} 3 & 1 & -1 \\ 2 & 4 & 1 \\ -1 & 2 & 5 \end{pmatrix}$ is strictly diagonally dominant since

$$|3| > |1| + |-1|, \quad |4| > |2| + |1|, \quad |5| > |-1| + |2|$$

while the matrix $B = \begin{pmatrix} 2 & 4 & 1 \\ 3 & 1 & -1 \\ -1 & 2 & 5 \end{pmatrix}$ is not strictly diagonally dominant since $|2| \not> |4| + |1|$.

Theorem: If the $n \times n$ matrix A is strictly diagonally dominant, then A is non-singular, and for every vector $\mathbf{b}$ and for every starting guess $\mathbf{x}_0$, the Jacobi iteration applied to $A\mathbf{x} = \mathbf{b}$ converges to the unique solution.

Let us write the Jacobi method in matrix notation. Let D be the diagonal of A, L the lower triangle (entries below the diagonal) of A, and U the upper triangle of A (entries above the diagonal). The system of equations $A\mathbf{x} = \mathbf{b}$ can be written

in fixed-point form

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ (D+L+U)\mathbf{x} &= \mathbf{b} \\ D\mathbf{x} &= \mathbf{b} - (L+U)\mathbf{x} \\ \mathbf{x} &= D^{-1}\left(\mathbf{b} - (L+U)\mathbf{x}\right) \end{aligned}$$

Since D is diagonal, D^{-1} is the diagonal matrix of the reciprocals of the entries of D. The Jacobi iteration is the fixed point iteration defined as follows:

$\mathbf{x}_0 =$ initial vector
$\mathbf{x}_{n+1} = D^{-1}\left(\mathbf{b} - (L+U)\mathbf{x}_n\right)$ for $n = 0, 1, 2, 3, \ldots$

Example:

$$\begin{pmatrix} 3 & 1 & -1 \\ 2 & 4 & 1 \\ -1 & 2 & 5 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ 1 \end{pmatrix}$$

$$D = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 5 \end{pmatrix}, \; L = \begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ -1 & 2 & 0 \end{pmatrix}, \; U = \begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

The Jacobi iteration with $\mathbf{x}_n = \begin{pmatrix} u_n \\ v_n \\ w_n \end{pmatrix}$ is

$$\begin{pmatrix} u_{n+1} \\ v_{n+1} \\ w_{n+1} \end{pmatrix} = \begin{pmatrix} 1/3 & 0 & 0 \\ 0 & 1/4 & 0 \\ 0 & 0 & 1/5 \end{pmatrix} \left(\begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & -1 \\ 2 & 0 & 1 \\ -1 & 2 & 0 \end{bmatrix} \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix} \right) = \begin{bmatrix} (4 - v_n + w_n)/3 \\ (1 - 2u_n - w_n)/4 \\ (1 + u_n - 2v_n)/5 \end{bmatrix}$$

which agrees with the previous version.

4.2 Gauss-Seidel Iteration

The Gauss-Seidel method is closely related to the Jacobi method. The only difference between them is that in the Gauss-Seidel method, the most recently updated values are used in the iteration.

Exercise: Use the Gauss-Seidel method to approximate the solution of the linear

system

$$\begin{aligned} 2x + 4y + z &= 1 \\ 3x + y - z &= 4 \\ -x + 2y + 5z &= 1 \end{aligned}$$

As shown previously, the matrix associated with the system above is not strictly diagonally dominant. This can be fixed by swapping the first two equations, or equivalently, swapping the first two rows of the resulting matrix. Then the matrix system is:

$$\begin{pmatrix} 3 & 1 & -1 \\ 2 & 4 & 1 \\ -1 & 2 & 5 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ 1 \end{pmatrix}.$$

The matrix is now strictly diagonally dominant. Let $(x_0, y_0, z_0) = (0, 0, 0)$ Then

$$\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{bmatrix} \frac{4+z_0-y_0}{3} \\ \frac{1-z_0-2x_1}{4} \\ \frac{1+x_1-2y_1}{5} \end{bmatrix} = \begin{bmatrix} 4/3 \\ \frac{1-2\cdot 4/3}{4} \\ \frac{1+4/3+2\cdot 5/12}{5} \end{bmatrix} = \begin{pmatrix} 4/3 \\ -5/12 \\ 19/30 \end{pmatrix} \approx \begin{pmatrix} 1.3333 \\ -0.41667 \\ 0.63333 \end{pmatrix}$$

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{bmatrix} \frac{4+z_1-y_1}{3} \\ \frac{1-z_1-2x_2}{4} \\ \frac{1+x_2-2y_2}{5} \end{bmatrix} = \begin{bmatrix} (4 + 19/30 + 5/12)/3 \\ (1 - 19/30 - 101/30)/4 \\ (1 + 101/60 + 3/2)/5 \end{bmatrix} = \begin{pmatrix} 101/60 \\ -3/4 \\ 251/300 \end{pmatrix} \approx \begin{pmatrix} 1.68333 \\ -0.7500 \\ 0.836667 \end{pmatrix}$$

$$\begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix} = \begin{bmatrix} \frac{4+z_2-y_2}{3} \\ \frac{1-z_2-2x_3}{4} \\ \frac{1+x_3-2y_3}{5} \end{bmatrix} = \begin{bmatrix} (4 + 251/300 + 3/4)/3 \\ (1 - 251/300 - 838/225)/4 \\ (1 + 1613/450 + 289/180)/5 \end{bmatrix} = \begin{pmatrix} 419/225 \\ -3197/3600 \\ 2783/3000 \end{pmatrix} \approx \begin{pmatrix} 1.86222 \\ -0.88056 \\ 0.927667 \end{pmatrix}$$

The Gauss-Seidel iteration converges to the vector $\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$

Gauss-Seidel can be written in matrix form and identified as a fixed-point iteration. Using similar notation to the Jacobi iteration,

$$A\mathbf{x} = (L + D + U)\mathbf{x} = \mathbf{b}$$

$$(L + D)\mathbf{x} = -U\mathbf{x} + \mathbf{b}$$

$$(L+D)\mathbf{x}_{n+1} = -U\mathbf{x}_n + \mathbf{b}$$

The use of newly computed entries is accommodated by L.

Gauss-Seidel iteration:
$\mathbf{x}_0$ = initial vector
$\mathbf{x}_{n+1} = D^{-1}(\mathbf{b} - U\mathbf{x}_n - L\mathbf{x}_{n+1})$ for $n = 0, 1, 2, 3, \cdots$

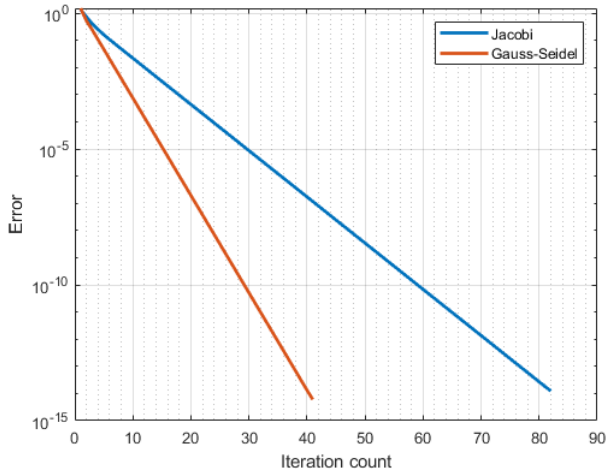


Figure 10: Convergence of Jacobi and Gauss-Seidel iterations on Example 5.1. Gauss-Seidel converges takes about 40 iterations to reach an error of 10^{-10}. Jacobi iteration takes 80 iterations to achieve the same accuracy.

MATLAB code for Gauss-Seidel

```
 function [x,iter,err] = gauss_seidel(A,b,tol,nmax)
% Gauss-Seidel
% A must be strictly diagonally dominant
% nmax maximum number of iterations
% tol tolerance
i = 1; res = 1; n = size(A,1);
x_old = zeros(n,1); x = x_old;
invD = 1./diag(A);
err = zeros(nmax,1);
U = triu(A,1);
L = tril(A,-1);
while (res>tol)&&(i<nmax)
    for j=1:n
        x(j)= invD(j)*(b(j)-U(j,:)*x_old-L(j,1:j-1)*x(1:j-1));
```

```
    end
    res = norm(x-x_old); % residual error
    x_old = x;
    err(i) = res;
    i = i+1; % increment count
end
iter = i;
err = err(1:i);
```

5 Polynomial Interpolation

This chapter introduces **polynomial interpolation** as a way to represent data. Suppose data points (x_i, y_i) are taken from a function $y = f(x)$, for example x may represent the temperature and y the reaction rate. Polynomial interpolation aims at finding a polynomial $p(x)$ that approximates f on a suitable domain. The reason why polynomials are used is that they are the most fundamental functions for digital computers, and computers have fast methods for floating point addition and multiplication, which are the only operations required in evaluating a polynomial. For this reason, complicated functions (such as $\sin(x)$, $\cosh(x)$, $\exp(-x^2)$) can be approximated by interpolating polynomials in order to make them computable.

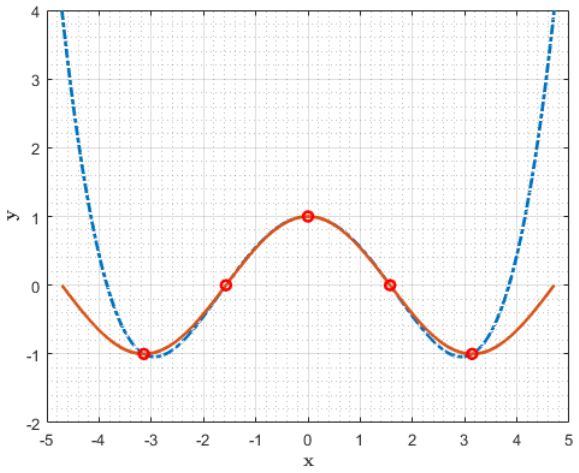


Figure 11: The dashed curve is a polynomial of degree 4 that interpolates the solid curve $y = \cos(x)$ on the interval $[-\pi, \pi]$ at the points $(-\pi, -1)$, $(-\pi/2, 0)$, $(0, 1)$, $(\pi/2, 0)$ and $(\pi, -1)$.

A polynomial $y = p(x)$ **interpolates** the data $(x_0, y_0), \cdots, (x_n, y_n)$ if $y_i = p(x_i)$ for each $0 \leq i \leq n$.

We now introduce methods for determining the interpolating polynomial $p(x)$ given a set of data $(x_0, y_0), \cdots, (x_n, y_n)$.

5.1 Monomial Interpolation

Suppose that $p(x)$ is a polynomial of degree less than or equal to n. Then the standard way of writing $p(x)$ is as follows

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n.$$

The set $\{1, x, x^2, \cdots, x^n\}$ of **monomials** forms a basis for the set of all polynomials of degree not greater than n. Each polynomial of degree $\leq n$ is uniquely determined by its set of $n+1$ coefficients $\{a_0, a_1, a_2, \cdots, a_n\}$. Thus, to find $p(x)$, it is sufficient to determine its coefficients.

Exercise 5.1: Find an interpolating polynomial of degree 2 that interpolates the points $(0, 1), \quad (2, 2), \quad (3, 4)$.

A polynomial of degree 2 is written as

$$p(x) = a_0 + a_1 x + a_2 x^2.$$

Since $p(x)$ interpolates the data at the given points, we have

$$p(0) = 1, \quad p(2) = 2, \quad p(3) = 4.$$

After substitution at the x-values, the following system of equation is obtained.

$$\begin{aligned} a_0 + a_1 \cdot 0 + a_2 \cdot 0^2 &= 1 \\ a_0 + a_1 \cdot 2 + a_2 \cdot 2^2 &= 2 \\ a_0 + a_1 \cdot 3 + a_2 \cdot 3^2 &= 4 \end{aligned}$$

In matrix notation,

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 0 & 0 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$$

By Gaussian elimination, the system above has solutions

$$a_0 = 1, a_1 = -\frac{1}{2}, a_2 = \frac{1}{2}.$$

Therefore, the interpolating polynomial is

$$\boxed{p(x) = 1 - \frac{1}{2}x + \frac{1}{2}x^2}$$

Check in MATLAB:

```
>> x = [0,2,3];
>> y = [1,2,4];
>> d = length(x)-1; % polynomial degree
>> p = polyfit(x,y,d); % find polynomial interpolant
>> format rat % output solution as fractions
>> p
p =
      1/2        -1/2         1
```

The polynomial coefficients in MATLAB are arranged from the highest degree to the lowest degree. In this case the polynomial is

```
p = 1/2*x.^2 - 1/2*x + 1.
```

In general, the determination of a polynomial $p(x) = a_0 + a_1x + \cdots + a_nx^n$ interpolating the points $(x_0, y_0), \cdots, (x_n, y_n)$ leads to the solution of a linear system of equations

$$\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & x_2^3 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \cdots \\ y_n \end{pmatrix}$$

The matrix in the system is known as the **Vandermonde matrix**. The determinant of the Vandermonde matrix can be expressed as

$$\det(V) = \prod_{0 \le i < j \le n} (x_j - x_i).$$

If the x_i's are distinct, all the factors in the determinant are nonzero. Hence $\det(V) \neq 0$ so that V is invertible and hence the system can be solved. In general, solving a system of equations is expensive when n is large. Hence monomial interpolation is not used in general.

5.2 Lagrange Interpolation

Suppose we wish to interpolate the $n+1$ points $(x_0, y_1), \cdots, (x_n, y_n)$ by a polynomial $p(x)$ of degree less than or equal to n. To do this, we introduce a special class of polynomials known as cardinal polynomials in interpolation theory denoted by $\ell_0, \ell_1, \cdots, \ell_n$ with the property that

$$\ell_i(x_j) = \delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j. \end{cases}$$

If these polynomials are determined, any continuous function f can be approximated in Lagrange form of the interpolating polynomial

$$p_n(x) = \ell_0(x)f(x_0) + \ell_1(x)f(x_1) + \cdots + \ell_n(x)f(x_n) = \sum_{i=0}^{n} \ell_i(x)f(x_i).$$

Let us evaluate $p_n(x)$ at x_j

$$p_n(x_j) = \sum_{i=0}^{n} \ell_i(x_j)f(x_i) = \sum_{i=0}^{n} \delta_{ij}f(x_i) = f(x_j)$$

since all terms $\ell_i(x_j)$ are zero except for the term $\ell_j(x_j) = 1$. This shows that $p_n(x)$ interpolates the function f at the points $x_0, \cdots, x_n$. It remains to determine a formula for the polynomials $\ell_j(x)$ $j = 0, \cdots, n$.

$$\ell_i(x) = \prod_{\substack{j \neq i \\ j=0}} \left(\frac{x - x_j}{x_i - x_j} \right), \text{ for all } 0 \leq i \leq n.$$

The formula for $\ell_i(x)$ is a product of n linear factors. The denominators $(x_i - x_j)$ are just numbers, and the numerators $(x - x_j)$ are each linear polynomials so that $p_n(x)$ is a polynomial of degree n.

Example 5.2: Find the Lagrange form of the interpolating polynomial for the data The cardinal polynomials are

x	1	2	3	4
y	2	1	6	47

$$\ell_0(x) = \left(\frac{x - x_1}{x_0 - x_1} \right) \left(\frac{x - x_2}{x_0 - x_2} \right) \left(\frac{x - x_3}{x_0 - x_3} \right) = \left(\frac{x - 2}{1 - 2} \right) \left(\frac{x - 3}{1 - 3} \right) \left(\frac{x - 4}{1 - 4} \right)$$

$$\ell_0(x) = -\frac{1}{6}(x-2)(x-3)(x-4)$$

$$\ell_1(x) = \left(\frac{x-x_0}{x_1-x_0}\right)\left(\frac{x-x_2}{x_1-x_2}\right)\left(\frac{x-x_3}{x_1-x_3}\right) = \left(\frac{x-1}{2-1}\right)\left(\frac{x-3}{2-3}\right)\left(\frac{x-4}{2-4}\right)$$

$$\ell_1(x) = \frac{1}{2}(x-1)(x-3)(x-4)$$

$$\ell_2(x) = \left(\frac{x-x_0}{x_2-x_0}\right)\left(\frac{x-x_1}{x_2-x_1}\right)\left(\frac{x-x_3}{x_2-x_3}\right) = \left(\frac{x-1}{3-1}\right)\left(\frac{x-2}{3-2}\right)\left(\frac{x-4}{3-4}\right)$$

$$\ell_2(x) = -\frac{1}{2}(x-1)(x-2)(x-4)$$

$$\ell_3(x) = \left(\frac{x-x_0}{x_3-x_0}\right)\left(\frac{x-x_1}{x_3-x_1}\right)\left(\frac{x-x_2}{x_3-x_2}\right) = \left(\frac{x-1}{4-1}\right)\left(\frac{x-2}{4-2}\right)\left(\frac{x-3}{4-3}\right)$$

$$\ell_3(x) = \frac{1}{6}(x-1)(x-2)(x-3)$$

The interpolating polynomial

$$p_3(x) = \ell_0(x)f(x_0) + \ell_1(x)f(x_1) + \ell_2(x)f(x_2) + \ell_3(x)f(x_3)$$

is then

$$\begin{aligned} p_3(x) \;=\; & -\frac{2}{6}(x-2)(x-3)(x-4) + \frac{1}{2}(x-1)(x-3)(x-4) \\ & -\frac{6}{2}(x-1)(x-2)(x-4) + \frac{47}{6}(x-1)(x-2)(x-3) \end{aligned}$$

In expanded form

$$\boxed{p_3(x) = 5x^3 - 27x^2 + 45x - 21}$$

Check:

$$p(1) = 5\cdot 1^3 - 27\cdot 1^2 + 45\cdot 1 - 21 = 2. \quad \checkmark$$

$$p(2) = 5\cdot 2^3 - 27\cdot 2^2 + 45\cdot 2 - 21 = 1. \quad \checkmark$$

$$p(3) = 5\cdot 3^3 - 27\cdot 3^2 + 45\cdot 3 - 21 = 6. \quad \checkmark$$

$$p(4) = 5 \cdot 4^3 - 27 \cdot 4^2 + 45 \cdot 4 - 21 = 47. \quad \checkmark$$

Let us check if this polynomial is the same as obtained using monomial interpolation. The Vandermonde system is

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 6 \\ 47 \end{pmatrix}$$

Using Gaussian elimination, we get that

$$a_0 = -21, a_1 = 45, a_2 = -27, a_3 = 5.$$

The polynomial is

$$p(x) = 5x^3 - 27x^2 + 45x - 21$$

which is the same as that obtained by Lagrange interpolation. This example supports the conclusion that the interpolating polynomial is unique.

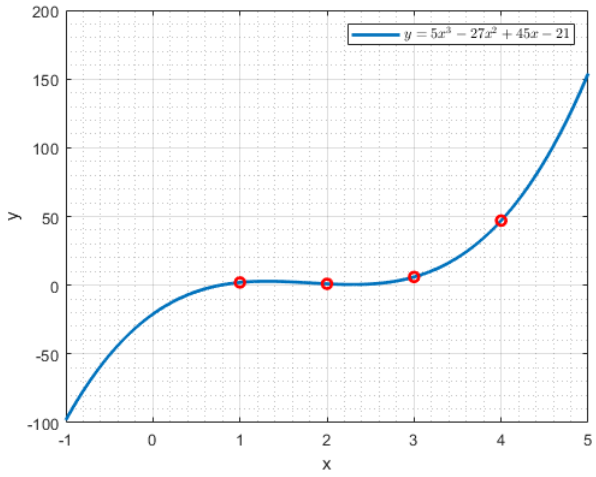


Figure 12: The polynomial $y = 5x^3 - 27x^2 + 45x - 21$ interpolates the data $(1, 2)$, $(2, 1)$, $(3, 6)$ and $(4, 47)$.

Theorem: (Existence and Uniqueness of Polynomial Interpolation)

Let $x_0, x_1, \cdots, x_n$ be distinct. Then for arbitrary values $y_0, y_1, \cdots, y_n$ there exists a unique polynomial $p(x)$ of degree at most n such that $p(x_i) = y_i$ for $0 \leq i \leq n$.

5.3 Newton form of the Interpolating Polynomial

The Lagrange form of the interpolating polynomial is rarely used in practice because there are other more efficient ways of forming the interpolating polynomial. Consider data obtained from evaluating a function $y = f(x)$ at points $x_0, \cdots, x_n$. The **Newton form of the interpolating polynomial** is of the form

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}).$$

The Newton's form can be written in a nested form that is particularly efficient for evaluation. For example, the Newton polynomial of degree 3

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2)$$

can be evaluated in nested form as

$$p(x) = a_0 + (x - x_0)\left\{a_1 + (x - x_1)\left[a_2 + a_3(x - x_2)\right]\right\}$$

requiring fewer arithmetic operations than a direct evaluation.

Example 5.3: Find the Newton form of the interpolating polynomial for the data The Newton form of the polynomial is of the form

x	1	2	3	4
y	2	1	6	47

$$p(x) = a_0 + a_1(x - 1) + a_2(x - 1)(x - 2) + a_3(x - 1)(x - 2)(x - 3)$$

We need to determine the coefficients $a_0, \cdots, a_3$. Note that

$$p(1) = a_0 = 2$$

since all the other terms vanish at $x = 1$. Next, evaluating at $x = 2$ we get

$$p(2) = a_0 + a_1(2 - 1) = 2 + a_1(2 - 1) = 2 + a_1 = 1 \Longrightarrow a_1 = -1.$$

Evaluating the polynomial at $x = 3$, we get

$$\begin{aligned} p(3) &= a_0 + a_1(3 - 1) + a_2(3 - 1)(3 - 2) \\ &= 2 + (-1) \cdot (3 - 1) + a_2(3 - 1)(3 - 2) \\ &= 2a_2 \\ &= 6 \end{aligned}$$

Hence $a_2 = 3$ Finally, evaluating at $x = 4$,

$$\begin{aligned} p(4) &= a_0 + a_1(4-1) + a_2(4-1)(4-2) + a_3(4-1)(4-2)(4-3) \\ &= 2 + (-1)\cdot 3 + 3\cdot 3\cdot 2 + a_3 \cdot 3\cdot 2\cdot 1 \\ &= 17 + 6a_3 \\ &= 47. \end{aligned}$$

So

$$6a_3 = 30 \Longrightarrow a_3 = 5.$$

Hence

$$p(x) = 2 - (x-1) + 3(x-1)(x-2) + 5(x-1)(x-2)(x-3).$$

In expanded form,

$$\boxed{p(x) = 5x^3 - 27x^2 + 45x - 21}.$$

There is a particularly efficient way of computing the coefficients $a_0, \cdots, a_3$ that uses the so-called **divided difference table**. For this, let us introduce some notation. Assuming the data is obtained from sampling a function $f(x)$, our goal is to interpolate the points $(x_0, f(x_0)), \cdots, (x_n, f(x_n))$. Listing the points in a table

$$\begin{array}{c|c} x_0 & f(x_0) \\ x_1 & f(x_1) \\ \vdots & \vdots \\ x_n & f(x_n) \end{array}$$

We define the divided differences as follows

$$\begin{aligned} f[x_k] &= f(x_k) \\ f[x_k, x_{k+1}] &= \frac{f[x_{k+1}] - f[x_k]}{x_{k+1} - x_k} \\ f[x_k, x_{k+1}, x_{k+2}] &= \frac{f[x_{k+1}, x_{k+2}] - f[x_k, x_{k+1}]}{x_{k+2} - x_k} \\ f[x_k, x_{k+1}, x_{k+2}, x_{k+3}] &= \frac{f[x_{k+1}, x_{k+2}, x_{k+3}] - f[x_k, x_{k+1}, x_{k+2}]}{x_{k+3} - x_k} \end{aligned}$$

and so on. The Newton's divided difference formula is then

$$\begin{aligned} p(x) = {} & f[x_0] + f[x_0, x_1](x - x_0) \\ & + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ & + f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2) \\ & + \cdots \\ & + f[x_0, \cdots, x_n](x - x_0)\cdots(x - x_{n-1}) \end{aligned}$$

The divided differences can be arranged in a convenient table

x_0	$f[x_0]$				
		$f[x_0, x_1]$			
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$		
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$	
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$		$f[x_0, x_1, x_2, x_3, x_4]$
		$f[x_2, x_3]$		$f[x_1, x_2, x_3, x_4]$	
x_3	$f[x_3]$		$f[x_2, x_3, x_4]$		
		$f[x_3, x_4]$			
x_4	$f[x_4]$				

Table 9: Divided differences table with 5 data points.

Example 5.4: Use Newton's divided differences table to find the interpolating polynomial passing through the points $(1, 2), \quad (2, 1), \quad (3, 6), \quad (4, 47)$

Applying the divided differences leads to the following table

1	$\boxed{2}$			
		$\boxed{-1}$		
2	1		$\boxed{3}$	
		5		$\boxed{5}$
3	6		18	
		41		
4	47			

Table 10: Divided differences table for the data in Example 5.4

The interpolating polynomial is then

$$p(x) = 2 - (x - 1) + 3(x - 1)(x - 2) + 5(x - 1)(x - 2)(x - 3)$$

or

$$\boxed{p(x) = 5x^3 - 27x^2 + 45x - 21}$$

MATLAB code for Newton's Divided Differences

```
function c = newtondd(x,y)
% Newton divided differences
% computes coefficients of interpolating poly
```

```
% [x,y] data points
n = length(x);
v = zeros(n); % initialize table
c = zeros(n,1); % initialize output coeffs
for j=1:n
    v(j,1)=y(j); % first column
end
for i=2:n % column i
    for j=1:n+1-i % fill column top to bottom
        v(j,i)=(v(j+1,i-1)-v(j,i-1))/(x(j+i-1)-x(j));
    end
end
for i=1:n
    c(i)=v(1,i);
end
```

5.4 Polynomial evaluation by nested multiplication

Suppose we want to evaluate the polynomial

$$p(x) = 3x^4 + x^3 - 4x^2 + 2x - 1$$

at $x = \frac{1}{2}$. The direct way is as follows

$$p(1/2) = 3 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} - 4 \cdot \frac{1}{2} \cdot \frac{1}{2} + 2 \cdot \frac{1}{2} - 1.$$

There are 9 multiplications and 4 additions (or subtractions), that is 13 operations. An efficient way to evaluate the polynomial is the so-called Horner's scheme or nested multiplication.

$$p(1/2) = -1 + \frac{1}{2}\left(2 + \frac{1}{2}\left(-4 + \frac{1}{2}\left(1 + \frac{1}{2}(3)\right)\right)\right) = \frac{3}{4}$$

There are now only 4 multiplications and 4 additions, or 8 operations.

Exercise: Write the following polynomial in nested form

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2)$$

In nested form, the Newton polynomial can be written as follows

$$p(x) = a_0 + (x - x_0)\left\{a_1 + (x - x_1)\left[a_2 + (x - x_2)(a_3)\right]\right\}$$

MATLAB code for Horner's nested polynomial evaluation

```
function y = nest(c,x,b)
% Horner's scheme
% evaluate a polynomial using nested multiplication
% c : vector of coefficients in Newton form
% x : x values at which to evaluate
% b : base points
% output y: values of p(x)
n = length(b);
d = n-1; % degree of polynomial
y = c(n);
for i=d:-1:1
    y = y.*(x-b(i))+c(i);
end
```

5.5 Errors in Polynomial Interpolation

In this section, we compute an error bound for polynomial interpolation.

Theorem: (Interpolation Error) Suppose $x_0, \cdots, x_n$ are distinct numbers in $[a, b]$ and $f \in C^{n+1}[a, b]$. Then for each $x \in [a, b]$ there exists a number $\xi \in (a, b)$ with

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0) \cdots (x - x_n)$$

where $p(x)$ is the interpolating polynomial.

Example: Suppose $f(x) = 1/x$ is approximated by a polynomial on $[2, 4]$ using the nodes $x = 2,\ 2.75$, and $x = 4$. Determine the error form for this polynomial, and the maximum error when the polynomial is used to approximate $f(x)$ for $x \in [2, 4]$.

Solution:

Let $p(x)$ be the interpolating polynomial. Then $p(x)$ is a polynomial of degree 2, and the error form is

$$f(x) - p(x) = \frac{f^{(3)}(\xi)}{3!}(x - 2)(x - 2.75)(x - 4)$$

where $\xi \in (2, 4)$

$$f(x) = x^{-1}, \quad f'(x) = -x^{-2}, \quad f''(x) = 2x^{-3}, \quad f^{(3)}(x) = -6x^{-4}.$$

Therefore

$$\frac{f^{(3)}(\xi)}{3!}(x-2)(x-2.75)(x-4) = -\frac{1}{\xi^4}(x-2)(x-2.75)(x-4).$$

The maximum of $\frac{1}{\xi^4}$ on $[2,4]$ is $\frac{1}{2^4} = \frac{1}{16}$. We now determine the maximum of

$$\phi(x) = (x-2)(x-2.75)(x-4)$$

on this interval. We first determine the critical points of $\phi(x)$.

$$\begin{aligned}
\phi(x) &= (x-2)(x-2.75)(x-4) \\
&= (x-2)(x-11/4)(x-4) \\
&= \frac{1}{4}(x-2)(4x-11)(x-4) \\
&= \frac{1}{4}(4x^2-19x+22)(x-4) \\
&= \frac{1}{4}(4x^3-35x^2+98x-88)
\end{aligned}$$

Hence

$$\phi'(x) = \frac{1}{4}(12x^2-70x+98) = 3x^2 - \frac{35}{2}x + \frac{49}{2} = \frac{1}{2}(3x-7)(2x-7).$$

The critical points occur at $x = \frac{7}{3} \approx 2.333$ and $x = \frac{7}{2} = 3.5$, both within the interval $[2,4]$. The values are $\phi(7/3) = 0.2315$ and $\phi(7/2) = 0.5625$. Therefore

$$\max_{x\in[2,4]} \phi(x) = 0.5625$$

and so the maximum error is

$$\frac{f^{(3)}(\xi)}{3!}|(x-2)(x-2.75)(x-4)| \leq \frac{1}{16\cdot 6}\cdot 0.5625 \approx 0.0059.$$

```
>> xx = [2,2.75,4]; % interpolating nodes
>> yy = 1./xx; % y-values at interpolating nodes
>> c = newtondd(xx, yy); % Newton polynomial coefficients
>> x = linspace(1,5,500); % x-values for evaluation
>> y = nest(c, x, xx); % y-values by Horner's scheme
>> plot(xx,yy,'ko',x,y,'-.',x,1./x,'b','Linewidth',2)
>> grid on
>> grid minor
```

MATLAB code for generating Fig 13.

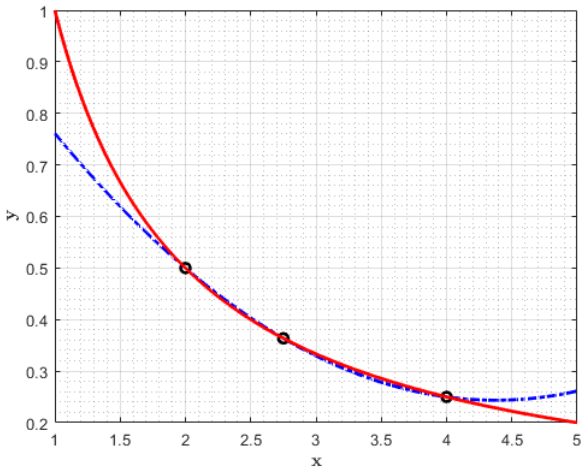


Figure 13: Interpolating polynomial (dashed curve) $p(x) = \frac{1}{22}x^2 - \frac{35}{88}x + \frac{49}{44}$ of the function $f(x) = 1/x$ (solid curve) on the interval $[2, 4]$ with nodes at $x = 2, 2.75$ and 4.

6 Numerical Differentiation

The main problem in computational calculus is how to compute the derivative or integral of a function. There are two ways one can do this:

- **Symbolic Computing:** one can compute the derivative or integral of a function in closed form using software such as Mathematica, Maple, Python (Sympy) and symbolic MATLAB (syms). It is quicker to compute the 5th derivative of the function $f(x) = \sin^2\left(2x^{\arctan(x)}\cosh(5x)\right)$ by symbolic computing methods where the calculus rules are carried out by a computer.
- **Numerical Computing:** A function may be specified as a tabulated list of numbers $(x_0, y_0), \cdots, (x_n, y_n)$ which may represent time/temperature or height/weight measurements. The functional relationship between the variables is usually unknown, so that using the rules of calculus to determine the derivative or integral is impossible. In this case, numerical differentiation and integration are used.

6.1 Finite Difference Formulas

Recall the limit definition of the derivative of a function f at the point x:

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}$$

provided that the limit exists. This leads to a useful formula for approximating the derivative of a function at x. By Taylor's formula:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(\xi)$$

where ξ is a number between x and $x+h$.

Two-point forward-difference formula

$$\boxed{f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(\xi)}.$$

When h is small,

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

is a good approximation of the derivative. The term $-\frac{h}{2}f''(\xi)$ is the approximation error that depends on a generally unknown ξ. Since the error is

proportional to h, the rate of convergence (without round-off error) is first-order. The two-point forward-difference formula is a first-order method of approximating $f'(x)$. In general if the error is proportional to h^n, then the method is nth order. We write $\mathcal{O}(h^n)$ to denote an nth order method.

Exercise 6.1: Use the two-point forward difference formula to approximate $f'(1)$ and find the approximation error, where $f(x) = \ln(x)$ for (a) $h = 0.1$, (b) $h = 0.01$ (c) $h = 0.001$.

Solution: The two-point forward difference formula is

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

(a)

$$f'(1) \approx \frac{\ln(1.0 + 0.1) - \ln(1.0)}{0.1} = 0.953101798043249$$

(b)

$$f'(1) \approx \frac{\ln(1.0 + 0.01) - \ln(1.0)}{0.01} = 0.995033085316809$$

(c)

$$f'(1) \approx \frac{\ln(1.0 + 0.001) - \ln(1.0)}{0.001} = 0.999500333083423$$

Since $f'(x) = 1/x$, then $f'(1) = 1$. The approximation errors are easily computed.

h	approx. error
0.1	0.0468982019567507
0.01	0.00496691468319077
0.001	0.000499666916576769

Table 11: Convergence of the two-point forward difference formula on $f(x) = \ln(x)$ for several values of h

Three-point centered difference formula

An $\mathcal{O}(h^2)$ can be derived for $f'(x)$ by a more advanced strategy. Suppose $f(x)$ is three times continuously differentiable. Then

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f^{(3)}(\xi_1)$$

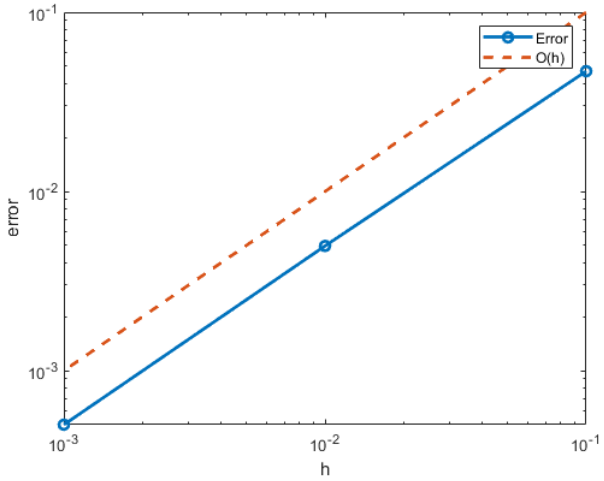


Figure 14: Loglog plot of the convergence of the two-point forward difference for approximating the derivative of $f(x) = \ln(x)$ at $x = 1$ showing first-order convergence.

and

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f^{(3)}(\xi_2)$$

where

$$x - h < \xi_2 < x < \xi_1 < x + h.$$

Subtracting the two equations gives

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{h^3}{6}\left[f^{(3)}(\xi_1) + f^{(3)}(\xi_2)\right]$$

Solving for $f'(x)$ gives

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{12}\left[f^{(3)}(\xi_1) + f^{(3)}(\xi_2)\right]$$

By the Generalized Intermediate Value Theorem, there exists ξ such that $x - h < \xi < x + h$ with,

$$\boxed{f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6}f^{(3)}(\xi)}$$

Generalized Intermediate Value Thorem: Let f be a continuous function on $[a, b]$, $\xi_1, \cdots, \xi_n \in [a, b]$ and $c_1, \cdots, c_n > 0$ positive numbers. Then there exists

a number $\xi \in (a, b)$ such that

$$f(\xi) = \frac{c_1 f(\xi_1) + \cdots + c_n f(\xi_n)}{c_1 + \cdots + c_n}.$$

In deriving the three-point centered difference formula, choose

$$c_1 = c_2 = 1$$

and $a = x - h, b = x + h$ in the Generalized Intermediate Value Theorem

Exercise 6.2: Use the three-point centered difference formula to approximate the derivative of $f(x) = \ln(x)$ at $x = 1$ and find the approximation error. using (a) $h = 0.1$, (b) $h = 0.01$ and (c) $h = 0.001$.

Solution: The two-point forward difference formula is

$$f'(x) \approx \frac{f(x + h) - f(x - h)}{2h}$$

(a)

$$f'(1) \approx \frac{\ln(1.0 + 0.1) - \ln(1.0 - 0.1)}{0.2} = 1.00335347731076$$

(b)

$$f'(1) \approx \frac{\ln(1.0 + 0.01) - \ln(1.0 - 0.01)}{0.02} = 1.00003333533348$$

(c)

$$f'(1) \approx \frac{\ln(1.0 + 0.001) - \ln(1.0 - 0.001)}{0.002} = 1.00000033333348$$

h	approx. error
0.1	0.00335347731075597
0.01	3.3335333477158e-05
0.001	3.33333478819142e-07

Table 12: Convergence of the three-point centered difference formula for $f(x) = \ln(x)$ for several values of h

Three-point centered differences for second derivative

$$\boxed{f''(x) = \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} - \frac{h^2}{12} f^{(4)}(\xi).}$$

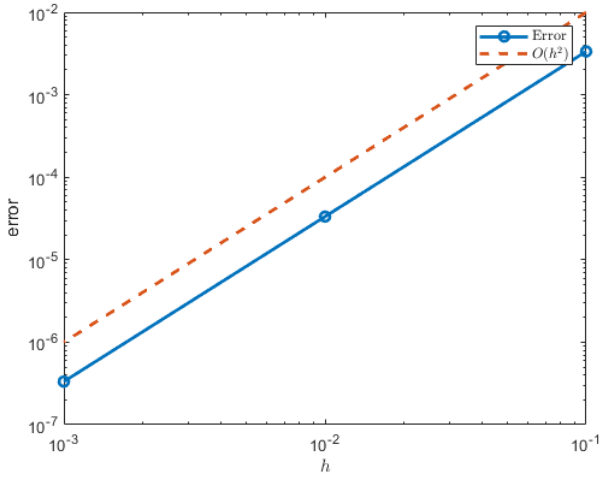


Figure 15: Loglog plot of the convergence of the three-point centered difference for approximating the derivative of $f(x) = \ln(x)$ at $x = 1$ showing 2nd-order convergence.

Proof: By Taylor's series

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f^{(3)}(x) + \frac{h^4}{24}f^{(4)}(\xi_1)$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f^{(3)}(x) + \frac{h^4}{24}f^{(4)}(\xi_2).$$

So that

$$\begin{aligned} f(x+h) - 2f(x) + f(x-h) &= [f(x) - 2f(x) + f(x)] + f'(x)[h-h] \\ &\quad + \frac{h^2}{2}\cdot 2f''(x) + \frac{h^3}{6}\left[f^{(3)}(x) - f^{(3)}(x)\right] \\ &\quad + \frac{h^4}{24}\left[f^{(4)}(\xi_1) + f^{(4)}(\xi_2)\right] \\ &= h^2 f''(x) + \frac{h^4}{24}\left[f^{(4)}(\xi_1) + f^{(4)}(\xi_2)\right] \end{aligned}$$

Dividing by h^2, and applying the Generalized Intermediate Value Theorem, we obtain the three-point centered difference formula for the second derivative.

Exercise 6.3: Find the values of $\alpha, \beta, \gamma, \delta,$ and p such that

$$f'(x) \approx \frac{\alpha f(x-h) + \beta f(x) + \gamma f(x+2h)}{\delta h} + \mathcal{O}(h^p)$$

is the best possible approximation for $f'(x)$.

Solution:

By Taylor series

$$\alpha f(x-h) = \alpha\left[f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f^{(3)}(\xi_1)\right]$$

$$\beta f(x) = \beta f(x)$$

$$\gamma f(x+2h) = \gamma\left[f(x) + 2hf'(x) + \frac{(2h)^2}{2}f''(x) + \frac{(2h)^3}{6}f^{(3)}(\xi_2)\right]$$

for some $\xi_1 \in (x-h, x)$ and $\xi_2 \in (x, x+2h)$. We require that

$$\alpha + \beta + \gamma = 0,\ \ 2\gamma - \alpha = 1,\ \ 2\gamma + \frac{\alpha}{2} = 0.$$

These are 3 equations in three unknowns α, β and γ.

From the second equation $\alpha = 2\gamma - 1$. By the third equation

$$2\gamma + (2\gamma - 1)/2 = 0.$$

Multiplying by 2,

$$4\gamma + 2\gamma - 1 = 0$$

or

$$\gamma = \frac{1}{6}.$$

So

$$\alpha = \frac{2}{6} - 1 = -\frac{4}{6}$$

and

$$\beta = -\alpha - \gamma = \frac{4}{6} - \frac{1}{6} = \frac{3}{6}.$$

Hence $\delta = 6$ and

$$f'(x) = \frac{-4f(x-h) + 3f(x) + f(x+2h)}{6h} - \frac{h^2}{6}\left[\frac{4}{6}f^{(3)}(\xi_1) + \frac{8}{6}f^{(3)}(\xi_2)\right]$$

$$\boxed{f'(x) = \frac{-4f(x-h) + 3f(x) + f(x+2h)}{6h} - \frac{h^2}{3}f^{(3)}(\xi)}$$

where $\xi \in (x-h, x+2h)$ and we have used the Generalized Intermediate Value Theorem $f(\xi) = (w_1 f(\xi_1) + w_2 f(\xi_2))/(w_1 + w_2)$ with $w_1 = \frac{4}{6}$ and $w_2 = \frac{8}{6}$.

7 Numerical Integration

This section is concerned with the approximation of definite integrals. We will discuss methods for **numerical integration,** or **quadrature** based on interpolation.

7.1 Trapezoidal rule

Suppose that $f(x)$ is a function with continuous second derivative defined on an interval $[x_0, x_1]$. Consider the degree 1 polynomial interpolating f at $(x_0, f(x_0))$ and $(x_1, f(x_1))$. Using the Lagrange formulation, we find that

$$f(x) = f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0} + \frac{(x - x_0)(x - x_1)}{2!}f''(\xi) = p(x) + E(x)$$

where $E(x)$ is the error term and $p(x)$ is the linear interpolant. Integrating both sides on $[x_0, x_1]$, we get

$$\int_{x_0}^{x_1} f(x)\, dx = \int_{x_0}^{x_1} p(x)\, dx + \int_{x_0}^{x_1} E(x)\, dx.$$

Computing the first integral gives,

$$\int_{x_0}^{x_1} p(x)\, dx = f(x_0)\int_{x_0}^{x_1} \frac{x - x_1}{x_0 - x_1}\, dx + f(x_1)\int_{x_0}^{x_1} \frac{x - x_0}{x_1 - x_0}\, dx.$$

Setting $h = x_1 - x_0$, and making a substitution $u = x_1 - x$, we get $du = -dx$ and

$$\int_{x_0}^{x_1} \frac{x - x_1}{x_0 - x_1}\, dx = \int_h^0 \frac{-u}{-h}\,(-du) = \frac{1}{h}\int_0^h u\, du = \frac{h}{2}.$$

Using $x - x_0 = (x_1 - x_0) - u = h - u$

$$\int_{x_0}^{x_1} \frac{x - x_0}{x_1 - x_0}\, dx = \frac{1}{h}\int_0^h (u - h)\, du = \frac{h}{2}$$

Hence,

$$\int_{x_0}^{x_1} p(x)\, dx = \frac{h}{2}\left[f(x_0) + f(x_1)\right]$$

calculates the area of the trapezoid between x_0 and x_1. Let us compute the integral of the error term

$$\begin{aligned}
\int_{x_0}^{x_1} E(x)\,dx &= \frac{1}{2!}\int_{x_0}^{x_1}(x-x_0)(x-x_1)f''(\xi)\,dx \\
&= \frac{f''(\xi)}{2}\int_{x_0}^{x_1}(x-x_0)(x-x_1)\,dx \\
&= \frac{f''(\xi)}{2}\int_0^h u(u-h)\,du \\
&= \frac{f''(\xi)}{2}\left[\frac{h^3}{3}-\frac{h^3}{2}\right] \\
&= -\frac{h^3}{12}f''(\xi).
\end{aligned}$$

Trapezoidal rule:

$$\boxed{\int_{x_0}^{x_1} f(x)\,dx = \frac{h}{2}\left[f(x_0)+f(x_1)\right] - \frac{h^3}{12}f''(\xi)}$$

where $h = x_1 - x_0$ and $\xi \in (x_0, x_1)$.

Composite Trapezoidal Rule:

The trapezoidal rule above is for one interval $[x_0, x_1]$. Let us extend this formula for multiple intervals defined on the nodes $a = x_0 < x_1 < x_2 < \cdots < x_n = b$. We are going to assume, further, that the width of interval k, $h_{k+1} := x_{k+1} - x_k = h$ for every $k = 0, 1, \cdots, n$, so that the sub-division is uniform.

$$\begin{aligned}
\int_a^b f(x)\,dx &= \sum_{k=1}^{n}\int_{x_k}^{x_{k+1}} f(x)\,dx \\
&= \frac{h}{2}\left[f(x_0)+f(x_1)+f(x_1)+f(x_2)+\cdots+f(x_n)\right] - \frac{h^3}{12}\sum_{k=1}^{n} f''(\xi_k) \\
&= \frac{h}{2}\left[f(x_0)+2\sum_{k=1}^{n-1} f(x_k)+f(x_n)\right] - \frac{h^3}{12}\frac{f''(\xi)}{n}
\end{aligned}$$

Since the sub-division is uniform, then $h = \frac{b-a}{n}$, and using $a = x_0$, $b = x_n$ and $x_k = a + kh$, it follows that

$$\boxed{\int_a^b f(x)\,dx = \frac{h}{2}\left[f(a)+2\sum_{k=1}^{n-1} f(a+kh)+f(b)\right] - \frac{(b-a)h^2}{12}f''(\xi)}$$

Exercise 7.1: Use the trapezoidal rule with $n = 4$ sub-intervals to approximate the definite integral

$$\int_1^2 \ln(x)\, dx.$$

Solution: $h = (b-a)/n = (2-1)/4 = \frac{1}{4}$. The nodes are $a = x_0 = 1, \quad x_1 = 1.25, \quad x_2 = 1.5, \quad x_3 = 1.75, \quad b = x_4 = 2.$

$$\begin{aligned} \int_1^2 \ln(x)\, dx &\approx \frac{0.25}{2}\left[\ln(1) + 2\ln(1.25) + 2\ln(1.5) + 2\ln(1.75) + \ln(2)\right] \\ &= 0.383699509 \end{aligned}$$

To check the accuracy, let us compute the exact solution

$$\int_1^2 \ln(x)\, dx = [x\ln(x) - x]_1^2 = (2\ln(2)-2)-(\ln(1)-1) = 2\ln(2)-1 = 0.386294361$$

MATLAB code for the Trapezoidal Rule

```
 function q = trapezoid(f,a,b,n)
% trapezoidal rule
% f is the function to be integrated
% n number of sub-divisions
% a - left end-point
% b - right end-point
h = (b-a)/n;
q = 0.5*h*(f(a)+f(b)); % end-points
for k=1:n-1 % middle-points
    xk = a+k*h;
    q = q + h*f(xk);
end
```

Degree of precision: The degree of precision of a numerical integration rule is the maximum possible degree of a polynomial such that the rule applied to the polynomial is exact, that is, incurs no discretization errors.

Exercise 7.2: Find the degree of precision of the trapezoidal rule.

Solution: We test the trapezoidal rule on the monomials $\{1, x, x^2, x^3, \cdots\}$ until the rule no longer applies. Consider $\int_0^1 f(x)\, dx$, and set $h = 1 - 0 = 1$, $f(x) = 1, x, x^2, x^3, \cdots$

$$\int_0^1 1\, dx = 1, \quad \frac{1}{2}[1+1] = 1\checkmark$$

$$\int_0^1 x\,dx = \frac{1}{2}, \quad \frac{1}{2}[0+1] = \frac{1}{2}\checkmark$$
$$\int_0^1 x^2\,dx = \frac{1}{3}, \quad \frac{1}{2}[0+1] = \frac{1}{2}, \mathsf{X}$$

The degree of precision of the trapezoidal rule is 1.

7.2 Simpson's rule

Let us approximate the function $f(x)$ by a quadratic function. For this, we need at least 3 points. Suppose x_0, x_1 and x_2 are given. The Lagrange interpolant of $f(x)$ at the points $(x_0, f(x_0))$, $(x_1, f(x_1))$ and $(x_2, f(x_2))$ is

$$p(x) = f(x_0)\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + f(x_2)\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}.$$

If f is three-times continuously differentiable, then

$$f(x) = p(x) + \frac{f^{(3)}(\xi)}{3!}(x-x_0)(x-x_1)(x-x_2).$$

Then

$$\int_{x_0}^{x_2} f(x)\,dx = \int_{x_0}^{x_2} p(x)\,dx + \int_{x_0}^{x_2} E(x)\,dx$$

where

$$E(x) = \frac{f^{(3)}(\xi)}{3!}(x-x_0)(x-x_1)(x-x_2)$$

is the error term.

Let us assume that $h = x_1 - x_0 = x_2 - x_1$, $u = x_2 - x$. Then $du = -dx$ and $x = x_2 - u$, $x - x_1 = h - u$, $x - x_0 = 2h - u$, $x - x_2 = -u$.

$$\begin{aligned}
\int_{x_0}^{x_2} p(x)\,dx &= \frac{f(x_0)}{2h^2}\int_{x_0}^{x_2}(x-x_1)(x-x_2)\,dx - \frac{f(x_1)}{h^2}\int_{x_0}^{x_2}(x-x_0)(x-x_2)\,dx \\
&\quad + \frac{f(x_2)}{2h^2}\int_{x_0}^{x_2}(x-x_0)(x-x_1)\,dx \\
&= \frac{f(x_0)}{2h^2}\int_0^{2h} u(u-h)\,du + \frac{f(x_1)}{h^2}\int_0^{2h} u(2h-u)\,dx + \\
&\quad + \frac{f(x_2)}{h^2}\int_0^{2h}(2h-u)(h-u)\,du \\
&= \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right]
\end{aligned}$$

The error term can be computed as

$$\int_{x_0}^{x_2} E(x)\,dx = -\frac{h^5}{90} f^{(4)}(\xi)$$

where $h = x_2 - x_1 = x_1 - x_0$ and $\xi \in (x_0, x_2)$.

Simpson's Rule

$$\boxed{\int_{x_0}^{x_2} f(x)\,dx = \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right] - \frac{h^5}{90} f^{(4)}(\xi)}$$

Composite Simpson's Rule

The simple Simpson's rule requires two sub-intervals, so that composite Simpson's rule requires an even number of sub-divisions. Let $n = 2m$, where m is an integer greater than or equal to one. Let $h = (b-a)/n$, and $x_k = a + kh$ for $k = 0, 1, 2, \cdots, n$, $a = x_0$ and $b = x_n$. Then,

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(a) + f(b) + 4\sum_{\ell=1}^{m} f(x_{2\ell-1}) + 2\sum_{\ell=1}^{m-1} f(x_{2\ell})\right] - \sum_{\ell=1}^{m-1} \frac{h^5}{90} f^{(4)}(\xi_\ell)$$

The error term can be expressed as, by the Generalized Intermediate Value Theorem,

$$\frac{h^5}{90}\sum_{\ell=1}^{m-1} f^{(4)}(\xi_\ell) = \frac{h^5}{90} m f^{(4)}(\xi)$$

Since $h = (b-a)/n$ and $n = 2m$, then $h \cdot m = (b-a)/2$. The error term is $-(b-a)h^4 f^{(4)}(\xi)/180$ assuming that the fourth derivative is continuous, and that $x_0 < \xi < x_2$

$$\boxed{\int_a^b f(x)\,dx = \frac{h}{3}\left[f(a) + f(b) + 4\sum_{\ell=1}^{m} f(x_{2\ell-1}) + 2\sum_{\ell=1}^{m-1} f(x_{2\ell})\right] - \frac{(b-a)h^4}{180} f^{(4)}(\xi)}$$

Exercise 7.3: Use Simpson's rule with $n = 4$ sub-intervals to approximate the value of

$$\int_1^2 \ln(x)\,dx.$$

Use the error formula to approximate the maximum possible error.

Solution: The width of each sub-interval is $h = (b-a)/n = (2-1)/4 = 0.25$.

$$\begin{aligned}\int_1^2 \ln(x)\,dx &\approx \frac{0.25}{3}\left[\ln(1) + \ln(2) + 4(\ln(1.25) + \ln(1.75)) + 2\ln(1.5)\right] \\ &= 0.386259562814567\end{aligned}$$

The exact value is $\int_1^2 \ln(x)\,dx = 2\ln(2) - 1 \approx 0.386294361119891$ and the exact error is 0.0000347983. The maximum error is

$$(b-a)h^4|f^{(4)}(\xi)|/180$$

for some $1 \le \xi \le 2$.

$$f'(x) = 1/x, \quad f''(x) = -x^{-2}, \quad f^{(3)}(x) = 2x^{-3}, \quad f^{(4)}(x) = -6x^{-4}$$

The maximum value of $|f^{(4)}(x)|$ on the interval $[1,2]$ is $\frac{6}{1^4} = 6$. Hence the maximum error is

$$\frac{(2-1)\cdot(0.25)^4}{180}\cdot 6 = \frac{(0.25)^4}{30} = 0.00013021$$

Exercise 7.4: Determine the minimum number of sub-divisions (and hence the width h of each sub-interval) so that the maximum error in using Simpson's method to approximate

$$\int_0^\pi \sin(x)\,dx$$

is less that 10^{-6}.

Solution: We require

$$(b-a)h^4|f^{(4)}(\xi)|/180 < 10^{-6}$$

where $f(x) = \sin(x)$. Since $f^{(4)}(x) = \sin(x)$, it follows that $|f^{(4)}(\xi)| \le 1$ for any ξ. Hence,

$$\pi h^4/180 < 10^{-6}$$

or

$$h < \left(180\cdot 10^{-6}/\pi\right)^{1/4} = 0.087$$

The number of sub-divisions n is then $n = (b-a)/h = \pi/0.087 = 36.1$. Since n must be an even integer, we choose $n = 38$ sub-divisions.

In practice, the value of n is an over-estimate since $n = 34$ is sufficient to bring the error to less than 10^{-6}. The value computed from the error formula is a good ball-park figure.

MATLAB code for Simpson's Rule

```
function q = simpson(f,a,b,n)
assert(mod(n,2)==0,'n must be even!')
% throw error if n is odd
h = (b-a)/n;
hs = h/3;
q = hs*(f(a)+f(b)); % first and last values
for k = 1:n-1
    if mod(k,2)==1 % odd indices
        q = q + 4*hs*f(a+k*h);
    else
        q = q + 2*hs*f(a+k*h); % even indices
    end
end
```

Exercise 7.5: Show that the degree of precision of Simpson's rule is $n = 3$.

7.3 Gaussian quadrature

The degree of precision of a quadrature method is the highest degree of a polynomial that can be calculated by the quadrature method without error. For example, the degree of precision of the Trapezoidal rule is $n = 1$, and the degree of precision of Simpson's rule is $n = 3$. In general, if a quadrature rule is developed from a degree n interpolatory polynomial using evenly spaced points, then the degree of precision is n if the polynomial degree is odd, and $n + 1$ if the polynomial degree is even. More accurate methods (Newton-Cotes methods) can be developed using polynomials of higher degree $n = 3, 4, \cdots$ and evenly spaced points. The number of function evlauations for Newton-Cotes methods is $n + 1$, e.g. $n = 2$ function evaluations for the trapezoidal rule and $n = 3$ for Simpson's rule.

Are there methods that can achieve a higher degree of precision with fewer function evaluations than Newton-Cotes methods? It turns out that this is true, if the nodes are chosen unevenly. **Gaussian quadrature** turns out to have degree of precision $2n + 1$ using only $n + 1$ function evaluations.

Gaussian Quadrature:

$$\int_{-1}^{1} f(x)\, dx \approx w_1 f(x_1) + \cdots w_n f(x_n)$$

where the w_i are the **weights** and x_i are the **nodes**. The weights and nodes for Gaussian quadrature are tabulated below for $n = 2, 3$ and 4.

n	nodes x_i	weights w_i
2	$-\sqrt{1/3}$	1
	$\sqrt{1/3}$	1
3	$-\sqrt{3/5}$	$5/9$
	0	$8/9$
	$\sqrt{3/5}$	$5/9$
4	$-\sqrt{\frac{15+2\sqrt{30}}{35}}$	$\frac{90-5\sqrt{30}}{180}$
	$-\sqrt{\frac{15-2\sqrt{30}}{35}}$	$\frac{90+5\sqrt{30}}{180}$
	$\sqrt{\frac{15-2\sqrt{30}}{35}}$	$\frac{90+5\sqrt{30}}{180}$
	$\sqrt{\frac{15+2\sqrt{30}}{35}}$	$\frac{90-5\sqrt{30}}{180}$

Table 13: Gaussian quadrature nodes and weights

Exercise 7.6: Approximate the integral

$$\int_{-1}^{1} e^{-\frac{x^2}{2}}\, dx$$

using Gaussian quadrature.

Solution: The correct answer to 14 digits is 1.7112487837843. Let $f(x) = e^{-x^2/2}$ be the integrand

$n = 2$:

$$\begin{aligned}\int_{-1}^{1} f(x)\, dx &\approx w_1 f(x_1) + w_2 f(x_2) \\ &= 1 \cdot f(-\sqrt{1/3}) + 1 \cdot f(\sqrt{1/3}) \\ &\approx 1.69296344978123\end{aligned}$$

$n = 3$:

$$\begin{aligned}\int_{-1}^{1} f(x)\, dx &\approx w_1 f(x1) + w_2 f(x_2) + w_3 f(x_3) \\ &= \frac{5}{9} f(-\sqrt{3/5}) + \frac{8}{9} f(0) + \frac{5}{9} f(\sqrt{3/5}) \\ &\approx 1.71202024520191\end{aligned}$$

$n = 4$:

$$\begin{aligned}\int_{-1}^{1} f(x)\,dx &\approx w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3) + w_4 f(x_4) \\ &\approx 1.71122450459949\end{aligned}$$

Gaussian Quadrature on $[a, b]$

To approximate an integral on $[a, b]$ the problem needs to be translated to $[-1, 1]$ using the substitution

$$t = \frac{2x - a - b}{b - a}.$$

Then

$$x = \frac{(b-a)t + a + b}{2}$$

and

$$dx = \frac{b-a}{2}dt.$$

$$t = -1 \Longrightarrow x = a, \quad t = 1 \Longrightarrow x = b.$$

Hence

$$\boxed{\int_a^b f(x)\,dx = \int_{-1}^{1} f\left(\frac{(b-a)t + a + b}{2}\right)\frac{b-a}{2}dt}$$

Exercise 7.7: Use Gaussian quadrature with $n = 3$ to approximate

$$\int_1^2 \ln(x)\,dx.$$

$$\int_1^2 \ln(x)\,dx = \int_{-1}^{1} \ln\left(\frac{t+3}{2}\right)\frac{1}{2}dt$$

Using $f(t) = \frac{1}{2}\ln((t+3)/2)$, we get 0.386300421584011. The exact solution is $2\ln(2) - 1 \approx 0.386294361119891$

Exercise 7.8: Use Gaussian quadrature with $n = 2, 3$ and 4 to compute

$$\int_0^1 \frac{\sin(x)}{x}\,dx.$$

Solution: Note that the function $f(x) = \sin(x)/x$ has a removable singularity at $x = 0$, and can be made into a continuous function by defining $f(0) = 1$. The solution to 14 digits is 0.9460830703671831.

By symmetry,

$$\int_0^1 \frac{\sin(x)}{x}\,dx = \frac{1}{2}\int_{-1}^1 \frac{\sin(x)}{x}\,dx.$$

Set

$$f(x) = \frac{\sin(x)}{2x}.$$

(i) $n = 2$:

$$\int_{-1}^1 f(x)\,dx \approx 1 \cdot \frac{\sin(-\sqrt{1/3})}{-2\sqrt{1/3}} + 1 \cdot \frac{\sin(\sqrt{1/3})}{2\sqrt{1/3}} = 0.9453630556704172$$

(ii) $n = 3$:

$$\int_{-1}^1 f(x)\,dx \approx \frac{5}{9}\cdot\frac{\sin(-\sqrt{3/5})}{-2\sqrt{3/5}} + \frac{8}{9}\cdot\frac{1}{2} + \frac{5}{9}\cdot\frac{\sin(\sqrt{3/5})}{2\sqrt{3/5}} = 0.9460874989218995$$

(iii) $n = 4$:

$$\int_{-1}^1 f(x) = w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3) + w_4 f(x_4) = 0.9460830546901068$$

MATLAB codes for Gaussian Quadrature

```
function q = gaussquad(f,a,b,n,m)
% Gaussian quadrature with m nodes per sub-interval
% n: number of subintervals
h = (b-a)/n;
[x,w]=gauleg(m); % nodes and weights
k = 0:n;
nds = a+k*h; % nodes
q = 0;
for j=1:n
    aj = nds(j);
    bj = nds(j+1);
    xj = 0.5*(bj-aj)*x+0.5*(bj+aj);
    wj = 0.5*(bj-aj)*w;
    q = q + wj'*f(xj);
end
```

```
function [x,w]=gauleg(n)
% input n: order of the Gaussian quadrature
% outputs: x nodes, w weights
% defined on [-1,1]
x=zeros(n,1);
w=zeros(n,1);
  m=(n+1)/2;
  xm=0.0;
  xl=1.0;
  for i=1:m
    z=cos(pi*(i-0.25)/(n+0.5));
    while 1
      p1=1.0;
      p2=0.0;
      for j=1:n
        p3=p2;
        p2=p1;
        p1=((2.0*j-1.0)*z*p2-(j-1.0)*p3)/j;
      end
      pp=n*(z*p1-p2)/(z*z-1.0);
      z1=z;
      z=z1-p1/pp;
      if (abs(z-z1)<eps), break, end
    end
    x(i)=xm-xl*z;
    x(n+1-i)=xm+xl*z;
    w(i)=2.0*xl/((1.0-z*z)*pp*pp);
    w(n+1-i)=w(i);
  end
```

8 Ordinary Differential Equations

The solution of ordinary and partial differential equations occupies a major part of activities within scientific computing. This is because many processes in science, engineering, and finance are modeled by differential equations, and except for a small number of simple cases, cannot be solved by analytic methods. Numerical algorithms for differential equations have thus become an integral part of scientific problem solving.

In this section, we will learn how to solve ordinary differential equations of the form

$$y'(t) = f(t, y(t)).$$

These are first-order differential equations, where the rate of change of a quantity y depends on the value of the quantity and time.

8.1 Initial Value Problems

Let us begin our study of ordinary differential equations with an illustration. Consider the **logistic equation** modeling the rate of change of a population

$$y' = cy(1 - y)$$

where y' is the rate of change of the population with respect to time, c the growth rate. The rate of change y' is proportional to the product of the current population and the remaining capacity $1 - y$. The growth rate of the population is small both when the population y is small and when the population is near capacity y close to 1.

The IVP has infinitely many solutions. By specifying an initial value, we can restrict the solutions to a single solution. An **initial value problem** (IVP) of a first order differential equation is the equation together with an initial condition on a specified interval $a \leq t \leq b$:

$$\begin{cases} y' = f(t, y) \\ y(a) = y_a \\ t \text{ in } [a, b] \end{cases}$$

The logistic differential equation has a solution that can be written in terms of elementary functions.

$$y(t) = 1 - \frac{1}{1 + \frac{y_0}{1-y_0} e^{ct}}.$$

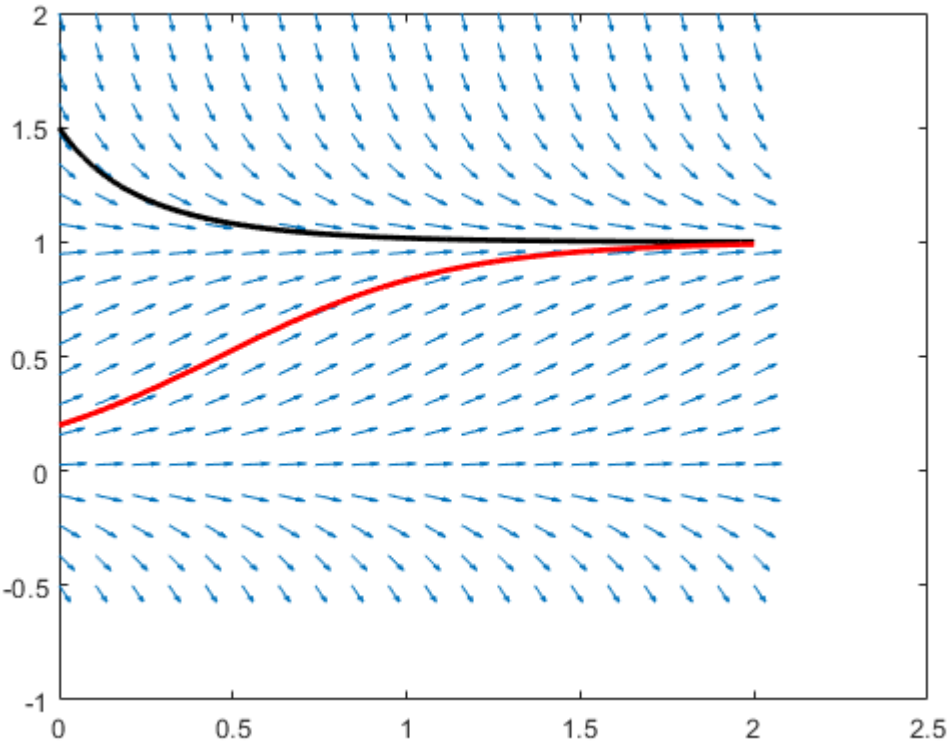


Figure 16: The slope field of the logistic equation with two particular solutions emanating from the initial conditions $y(0) = 1.5$ and $y(0) = 0.2$. Both solutions converge towards $y(t) = 1$ for large t

Using an arrow to plot the slope at each point, we get the **direction field** or **slope field** of the differential equation. If an initial value is specified, the solution of the differential equation can be obtained by following the slope fields; the solution is tangent to the slope field emanating from the initial value.

8.2 Euler's method

The logistic equation has a fairly simple solution. Most differential equations arising in applications do not have an explicit solution formula. The slope field in Figure 16 suggests a computational way of solving differential equations: begin at (t_0, y_0) and follow the slope field. After a small amount of time, re-evaluate the slope and follow the slope-field there and follow the new direction to a point (t_1, y_1). Repeat the process.

Euler's method for solving the initial value problem $y'(t) = f(t, y)$ is derived using the forward difference approximation of the first derivative at (t_n, y_n). Suppose we want to solve the IVP in an interval $[0, b]$. We first discretize the interval $0 = t_0 < t_1 < \cdots < t_N = b$. Let us suppose further that the points are uniform, i.e. $h_n := t_{n+1} - t_n = h$ for every $n = 1, 2, 3, \cdots$. Recall that

$$\frac{y(t_{n+1}) - y(t_n)}{h} = y'(t_n) - \frac{h}{2} f''(c)$$

for some $c \in (t_n, t_{n+1})$. Since $y'(t) = f(t, y)$, and using the approximation $y(t_n) \approx y_n$ we get Euler's method.

$$\boxed{y_{n+1} = y_n + hf(t_n, y_n) = y_n + hf_n},$$

where $f_n = f(t_n, y_n)$. Starting with y_0, we use the above iteration to generate y_n, $n = 1, 2, 3, \cdots$

Example 8.1: Use Euler's method to approximate the solution of the IVP

$$y' = -2t - y, \quad y(0) = 1$$

on the interval $t \in [0, 0.3]$ using a step size of $h = 0.1$.

Solution: We need to compute y_1, y_2 and y_3.

$$\begin{aligned} y_1 &= y_0 + hf(t_0, y_0) \\ & \quad y_0 = 1, h = 0.1, f(t_0, y_0) = -2t_0 - y_0 = -2(0) - 1 = -1. \\ y_1 &= 1 + 0.1(-1) = 1 - 0.1 = 0.9 \end{aligned}$$

The calculation of y_2 and y_3 is similar.

$$\begin{aligned} y_2 &= y_1 + hf(t_1, y_1) = 0.790 \\ y_3 &= y_2 + hf(t_2, y_2) = 0.671 \end{aligned}$$

The exact solution is $y(t) = 2 - 2t - e^{-t}$, and the correct value is $y(0.3) = 0.659182$. The absolute error at $y(0.3)$ is:

$$|y(0.3) - y_3| = 0.011818.$$

8.3 Implicit Euler's Method

Euler's method above is an **explicit** method because the solution at the next time step depends on the solution at the previous time step. The stability of Euler's method can be improved by considering an **implicit** modification. Implicit means that the solution at the current time step depends on both the past and the current time.

In order to derive the implicit Euler's method, we approximate y' by a backwards difference formula.

$$\frac{y_{n+1} - y_n}{h} \approx y'(t_{n+1}) = f(t_{n+1}, y_{n+1}).$$

The implicit Euler's method is then

$$\boxed{y_{n+1} = y_n + hf_{n+1}}$$

Exercise 8.2:

Find an approximation of $y(0.03)$ of the IVP

$$y' = -2t - y, \quad y(0) = 1.0$$

using the Implicit Euler's method.

Solution:

$$\begin{aligned}
y_{n+1} &= y_n + hf_{n+1} = y_n + h[-2t_{n+1} - y_{n+1}] \\
y_{n+1}(1+h) &= y_n - 2ht_{n+1} \\
y_{n+1} &= \frac{y_n - 2ht_{n+1}}{1+h} \\
y_1 &= \frac{y_0 - 2ht_1}{1+h} = \frac{1 - 2 \cdot 0.1^2}{1+0.1} = 0.890909 \\
y_2 &= 0.773554 \\
y_3 &= 0.648685
\end{aligned}$$

The correct value is $y(0.3) = 0.659182$ and the absolute error is $|y(0.3) - y_3| = 0.010497$.

8.4 Modified Euler's method

This is also known as Heun's method or Improved Euler, or Euler-Cauchy's method. Essentially the function f is overaged over the two endpoints:

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

where $\tilde{y}_{n+1}$ is then approximated by Euler's method $y_{n+1} = y_n + hf(t_n, y_n)$ so the modified Euler's method takes the form

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

Example 8.3: Find an approximation to $y(0.3)$ for the IVP

$$y' = -2t - y, \quad y(0) = 1.0$$

using the modified Euler's method and step size $h = 0.1$.

Solution:

$$\begin{aligned}
y_1 &= y_0 + \frac{h}{2}\left[f(t_0, y_0) + f(t_1, y_0 + hf(t_0, y_0)\right] \\
f(t_0, y_0) &= -2t_0 - y_0 = -1 \\
f(t_1, y_0 + hf(t_0, y_0)) &= -2t_1 - (y_0 + hf(t_0, y_0)) = -2 \cdot 0.1 - (1 - 0.1 \cdot 1) = -1.1 \\
y_1 &= 1 + \frac{0.1}{2}(-1 - 1.1) = 0.895000 \\
y_2 &= 0.780975 \\
y_3 &= 0.658782
\end{aligned}$$

The exact value is $y(0.3) = 0.659182$. The absolute error is then $|y(0.3) - y_3| = 0.000399$.

Trapezoidal method (Crank-Nicholson)

Consider the first order IVP

$$y'(t) = f(t, y), \quad y(0) = y_0,$$

on the time interval $[t_n, t_{n+1}]$. Integrating both sides of the differential equation on the interval $[t_n, t_n + h]$ and applying the Fundamental Theorem of Calculus, we

obtain

$$\int_{t_n}^{t_{n+1}} y'(s)\, ds \;=\; \int_{t_n}^{t_{n+1}} f(s, y(s))\, ds$$

The left hand side evaluates to $y(t_{n+1}) - y(t_n)$. The integral on the right hand is approximated using the Trapezoidal rule

$$\int_{t_n}^{t_{n+1}} f(s, y(s))\, ds = \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right] - \frac{h^3}{12} f^{(2)}(c_n)$$

for some $c_n \in (t_n, t_{n+1})$. Dropping the higher-order error term we get the Trapezoidal (Crank-Nicholson) scheme

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

Exercise 8.4: Find an approximation to $y(0.3)$ for the IVP

$$y' = -2t - y, \quad y(0) = 1$$

using the Trapezoidal method with step size $h = 0.1$

Solution:

$$\begin{aligned}
y_{n+1} &= y_n + \frac{h}{2}\left[f_n + f_{n+1}\right] \\
&= y_n + \frac{h}{2}\left[-2t_n - y_n - 2t_{n+1} - y_{n+1}\right] \\
y_{n+1}(1 + h/2) &= y_n(1 - h/2) - h(t_n + t_{n+1}) \\
y_{n+1} &= \frac{y_n(1 - h/2) - h(t_n + t_{n+1})}{1 + h/2} \\
y_1 &= 0.895238 \\
y_2 &= 0.781406 \\
y_3 &= 0.659367
\end{aligned}$$

The exact solution is $y(0.3) = 0.659182$. The absolute error is $y(0.3) - y_3| = 0.000224$.

8.5 Runge Kutta methods

These are some of the most commonly used methods for approximating solutions of differential equations. There are several levels of Runge-Kutta methods with the convergence rates increasing with each level.

RK-1:

$$\boxed{y_{n+1} = y_n + hf(t_n, y_n)}$$

Notice that RK-1 is the (forward) Euler's method.

RK-2:

Define

$$k_1 = f(t_n, y_n)$$
$$k_2 = f(t_n + h, y_n + hk_1).$$

Then the RK-2 method is defined by the iteration

$$\boxed{y_{n+1} = y_n + \frac{h}{2}\left[k_1 + k_2\right]}.$$

RK-2 is the modified Euler's method.

RK-3:

Define k_1, k_2 and k_3 as:

$$k_1 = f(t_n, y_n)$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$
$$k_3 = f(t_n + h, y_n + 2hk_1 - hk_2)$$

.

The RK-3 method is then defined as

$$\boxed{y_{n+1} = y_n + \frac{h}{6}\left[k_1 + 4k_2 + k_3\right]}$$

RK-4:

Define $k1, k_2, k_3$ and k_4 as follows:

$$k_1 = f(t_n, y_n)$$
$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$$
$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$$
$$k_4 = f(t_n + h, y_n + hk_3)$$

$$\boxed{y_{n+1} = y_n + \frac{h}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right]}$$

Exercise 8.5:

Find an approximation to $y(0.03)$ for the IVP

$$y'(t) = -2t - y, \quad y(0) = 1$$

using RK-4 using a step size of $h = 0.1$. Compute the absolute errors.

Solution:

RK-4:

$$k_1 = f(t_0, y_0) = -2t_0 - y_0 = -2(0) - 1 = -1$$
$$k_2 = f(t_0+\frac{h}{2}, y_0+\frac{h}{2}k_1) = -2(t_0+h/2)-(y_0+\frac{h}{2}k_1) = -2\cdot\frac{0.1}{2}-\left(1+\frac{0.1}{2}(-1)\right) = -1.05$$
$$k_3 = f(t_0+\frac{h}{2}, y_0+\frac{h}{2}k_2) = -2(t_0+h/2)-(y_0+\frac{h}{2}k_2) = -2\cdot\frac{0.1}{2}-\left(1+\frac{0.1}{2}(-1.05)\right) = -1.0475$$
$$k_4 = f(t_0 + h, y_0 + hk_3) = -2(0.1) - (1 + 0.1(-1.0475)) = -1.09525$$

$$y_1 = 1 + \frac{0.1}{6}\left[-1 + 2(-1.05) + 2(-1.0475) + (-1.09525)\right] = 0.8951625$$
$$y_2 = 0.781269098594$$
$$y_3 = 0.6591851577999$$

The exact solution at $t = 0.3$ is $y(0.3) = 0.659181779318$. The absolute error is $|y(0.3) - y_3| = 0.000000201319 = 2.01319 \times 10^{-7}$.