Introductory Computational Mathematics

Semester 2, 2022

Dr Elliot Carr

Tarang Janawalkar

Contents

1 Preliminaries

1.1 Errors

Errors in calculations are a common problem in numerical analysis. We can quantify the magnitude of such an error by two measures.

Definition 1.1 (Absolute and relative error)**.** Let $\tilde{x}$ be an approximation of x. Then the **absolute error** is given by

$$\text{absolute error} = |\tilde{x} - x|.$$

The **relative error** is given by

$$\text{relative error} = \frac{|\tilde{x} - x|}{|x|}.$$

It is important to realise that the absolute error can be misleading when comparing different sizes of errors, i.e., it is always small for small values of x and $\tilde{x}$.

1.2 Floating Point Arithmetic

The set of real numbers $\mathbb{R}$ contains uncountably many elements. Computers have a limited number of bits, and can therefore only represent a small subset of these elements.
The most common approximation of real arithmetic used in computers is known as **floating point arithmetic**.

Definition 1.2 (Floating point number system)**.** A floating point number system $\mathbb{F}(\beta,\ k,\ m,\ M)$ is a *finite subset* of the real number system characterised by the parameters:

- $\beta \in \mathbb{N}$: the base
- $k \in \mathbb{N}$: the number of digits in the significand
- $m \in \mathbb{Z}$: the minimum exponent
- $M \in \mathbb{Z}$: the maximum exponent

Definition 1.3 (Floating point numbers)**.** The floating point numbers $f \in \mathbb{F}(\beta,\ k,\ m,\ M)$ are real numbers expressible in the form

$$f = \pm \left(d_1.d_2 d_3 \ldots d_k\right)_\beta \times \beta^e$$

where $e \in \mathbb{Z}$ is the **exponent** satisfying $m \leq e \leq M$. The quantity $d_1.d_2 d_3 \ldots d_k$ is known as the **significand**, where d_i are base-β digits, with $d_1 \neq 0$ unless $f = 0$, to ensure a unique representation of f.

Computers primarily use floating point number systems with base $\beta = 2$ (binary), other common bases include $\beta = 10$ (decimal[1]) and $\beta = 16$ (hexadecimal).

[1] Note that for base-10, we do not need to include the subscript in the significand.

To illustrate the finiteness of the floating point number system, consider the following example:

$$\begin{aligned}\mathbb{F}\,(10,\,3,\,-1,\,1) = \{\,0,&\\ &\pm 1.00\times 10^{-1}, \quad \pm 1.01\times 10^{-1}, \quad \dots, \quad \pm 9.99\times 10^{-1},\\ &\pm 1.00\times 10^{0}, \quad \pm 1.01\times 10^{0}, \quad \dots, \quad \pm 9.99\times 10^{0},\\ &\pm 1.00\times 10^{1}, \quad \pm 1.01\times 10^{1}, \quad \dots, \quad \pm 9.99\times 10^{1}\ \}\\ = \{\,0,&\\ &\pm 0.100, \quad \pm 0.101, \quad \dots, \quad \pm 0.999,\\ &\pm 1.00, \quad \pm 1.01, \quad \dots, \quad \pm 9.99,\\ &\pm 10.0, \quad \pm 10.1, \quad \dots, \quad \pm 99.9\ \}\end{aligned}$$

Note that the numbers in this set are not equally spaced, (smaller spacing for smaller exponents).

Definition 1.4 (Overflow and underflow)**.** Consider the value $x \in \mathbb{R}$, if x is too small in magnitude to be represented in $\mathbb{F}$, an **underflow** occurs which typically causes the number to be replaced by zero.
Similarly, if x is too large in magnitude to be represented in $\mathbb{F}$, an **overflow** occurs which typically causes the number to be replaced by infinity.

Corollary 1.2.0.1. *The smallest and largest values (in magnitude) of* $\mathbb{F}$ *are given by*

$$\begin{aligned}\min_{f\in\mathbb{F}}|f| &= \beta^{m}\\ \max_{f\in\mathbb{F}}|f| &= \left(1-\beta^{-k}\right)\beta^{M+1}.\end{aligned}$$

The cardinality of the positive elements in $\mathbb{F}$*, is given by*

$$|\{f\in\mathbb{F} : f>0\}| = (M-m+1)\,(\beta-1)\,\beta^{k-1}$$

so that by including negative numbers and zero, the cardinality of $\mathbb{F}$ *is given by*

$$|\mathbb{F}| = 2|\{f\in\mathbb{F} : f>0\}| + 1.$$

1.2.1 Representing Real Numbers as Floating Point Numbers

If we wish to represent a real number[2] x that is not exactly representable in $\mathbb{F}$, we can **round** the number to the nearest *representable* number.
The error committed by this process is known as the **roundoff error**.

1.2.2 Converting between Floating Point Number Systems

Consider $fl : \mathbb{R} \to \mathbb{F}\,(\beta,\,k,\,m,\,M)$, defined as function which maps real numbers x to the nearest element in $\mathbb{F}$. To determine $fl\,(x)$:

1. Express x in base β.
2. Express x in scientific form.

[2] x must satisfy $\min(\mathbb{F}) \le x \le \max(\mathbb{F})$.

3. Verify that $m \leq e \leq M$:
   - If $e > M$, then $x = \infty$.
   - If $e < m$, then $x = 0$.
   - Otherwise, round the significand to k digits.

The relative error produced by rounding x to $fl(x)$ is bounded according to

$$\frac{|x - fl(x)|}{|x|} \leq \frac{1}{2}\beta^{1-k}.$$

Definition 1.5 (Unit roundoff)**.** The **unit roundoff** or **machine precision** u of a floating point number system $\mathbb{F}(\beta,\ k,\ m,\ M)$ is given by

$$u = \frac{1}{2}\beta^{1-k}.$$

1.2.3 IEEE Floating Point Standard

IEEE 754 is the standard for floating point arithmetic used by most modern computers.
It is a binary format, with several variants. The most common variant is **IEEE double precision**, which is based on $\mathbb{F}(2,\ 53,\ -1022,\ 1023)$.
The basic properties of this format are summarised in the following table.

Unit roundoff	$u = 1.11 \times 10^{-16}$
Largest representable positive number	1.80×10^{308}
Smallest representable positive number	2.23×10^{-308}
Special values	± 0, $\pm\infty$, `NaN`

1.3 Catastrophic Cancellation

When working with floating point arithmetic, roundoff is a common source of error. Certain operations may bring roundoff errors that are too large to be easily corrected.
Catastrophic cancellation or **cancellation error** is the error that occurs in the floating point subtraction of two numbers that are very close to each other, where at least one of them is not exactly representable.
As an example, the quadratic formula

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

experiences catastrophic cancellation for $b^2 \gg 4ac$, as $b^2 - 4ac \approx b^2$ so that $\sqrt{b^2 - 4ac} = \sqrt{b^2} = |b|$:

$$x_1 = \frac{-b + |b|}{2a} \qquad x_2 = \frac{-b - |b|}{2a}$$

When $b > 0$, $|b| = b$, so that

$$x_1 = \frac{-b + b}{2a} = \frac{b - b}{2a}.$$

And when $b < 0$, $|b| = -b$, so that

$$x_2 = \frac{-b - (-b)}{2a} = \frac{b - b}{2a}.$$

This cancellation can be avoided by taking the product of the two roots to determine the exact result of the root that suffers from catastrophic cancellation.

$$x_1 x_2 = \frac{c}{a}.$$

1.4 Taylor Polynomials

Suppose we have a function $f(x)$ that is n differentiable at the point $x = x_0$. This function can be approximated by a sum of polynomials that agrees with its first n derivatives at that point.

Definition 1.6 (Taylor polynomial)**.** The **Taylor polynomial** of degree n of f, centred at x_0 is defined by

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k.$$

The Taylor polynomial can be used to approximate a function f for values of x near x_0, the following theorem addresses how accurate the approximation is.

Definition 1.7 (Taylor's theorem)**.** Suppose that f is $n+1$ times differentiable on an interval $[a, b]$ containing x_0, and let P_n be the degree n Taylor polynomial for f, centred on x_0. Then for all $x \in [a, b]$, there exists a value $x_0 < c < x$ such that

$$f(x) = P_n(x) + \frac{f^{(n+1)}(c)}{(n+1)!} (x - x_0)^{n+1}.$$

The term

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

is called the **error term** or **remainder term** for P_n.

To determine the absolute error from the Taylor series polynomial, consider the error term:

$$|f(x) - P_n(x)| = |R_n(x)|.$$

The maximum value of $|R_n(x)|$ on the interval $[a, b]$ gives the bound on the maximum error incurred when approximating f by P_n on that interval.

1.5 Taylor Series

Given an infinitely differentiable function f, we can take the limit $n \to \infty$ to find Taylor series representation of f, given by:

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k.$$

When we truncate this series at a finite n, the error from the Taylor series is known as **truncation error**. In this case, the remainder term gives us a *bound* on the size of this truncation error.

2 Ordinary Differential Equations

For certain classes of ordinary differential equations (ODEs), we can obtain analytical closed-form solutions using known techniques. However for most cases we will need to use numerical techniques to obtain approximate solutions.

2.1 Initial Value Problems

While solutions with arbitrary constants, such as $y = Ce^t$ are acceptable for theoretical analysis, we cannot have such variables when determining an approximate solution in the real world. Hence we require **initial conditions** to obtain a definitive solution. An ODE combined with an initial condition is called an **initial value problem** (IVP).

2.2 Time Discretisation

Consider the initial value problem,

$$\begin{aligned} y(a) &= \alpha \\ \frac{\mathrm{d}y(t)}{\mathrm{d}t} &= f(t,\, y(t)), \qquad a \leq t \leq b \end{aligned}$$

Divide the interval $[a, b]$ into n subintervals, each with width $h = (b-a)/n$. Then define $t_i = a + ih$, for $i = 0,\, 1,\, \ldots,\, n$, so that $t_0 = a$ and $t_n = b$. If we then compute $y_i = y(t_i)$ denoted as w_i, then $w_i \approx y_i$ for all $i = 0,\, 1,\, \ldots,\, n$.

2.3 Euler's Method

Euler's method, or the first order Taylor method, uses a Taylor polynomial approximation of y over each subinterval. Assuming y is twice differentiable,

$$y(t_i + h) = y(t_i) + hy'(t_i) + \mathcal{O}(h^2).$$

The remainder term is not shown in its exact form but rather as $\mathcal{O}(h^2)$, "Big-O of h^2", or "order h^2", meaning that the error is proportional to h^2. Using the ODE and substituting $t_{i+1} = t_i + h$ gives

$$y_{i+1} = y_i + hf(t_i,\, y_i) + \mathcal{O}(h^2).$$

By removing the remainder term, we obtain the approximation method known as **Euler's method**:

$$\begin{aligned} w_0 &= \alpha \\ w_{i+1} &= w_i + hf(t_i,\, w_i) \end{aligned}$$

for $i = 0,\, 1,\, \ldots,\, n-1$.

2.4 Local and Global Error

Local error is defined as the error that the method would incur in **one step**, assuming the solution was correct at the previous step. The **global error** is defined as the error in the solution after i steps (at $t = t_i$), and is given by:

$$|y_i - w_i|.$$

It is the accumulation of the local errors from steps 1, 2, ..., i.
The **order** of a method refers to the global error of that method. For Euler's method, the local error is $\mathcal{O}(h^2)$ while the global error is $\mathcal{O}(h)$, hence "*first order* Taylor method".
In general, if the local error is $\mathcal{O}(h^{p+1})$, then the global error is $\mathcal{O}(h^p)$ and the method is said to be a pth order method.

$$\text{Global error} \approx n\mathcal{O}(h^{p+1}) = \frac{b-a}{h}\mathcal{O}(h^{p+1}) = \mathcal{O}(h^p).$$

2.5 Second Order Taylor Method

To improve upon the accuracy of Euler's method, we can use additional Taylor polynomial terms by truncating at a higher order. Assuming y is three times differentiable, we have

$$y(t_i + h) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \mathcal{O}(h^3)$$

which can be rewritten as

$$y_{i+1} = y_i + hf(t_i,\, y_i) + \frac{h^2}{2}f'(t_i\, y_i) + \mathcal{O}(h^3).$$

Again by removing the remainder term, we obtain the approximation w_i of y_i, known as the **second order Taylor method**:

$$\begin{aligned} w_0 &= \alpha \\ w_{i+1} &= w_i + hf(t_i,\, w_i) + \frac{h^2}{2}f'(t_i,\, w_i) \end{aligned}$$

for $i = 0,\, 1,\, \ldots,\, n-1$. This method has a local error of $\mathcal{O}(h^3)$, and therefore a global error of $\mathcal{O}(h^2)$.

2.6 Modified Euler Method

Although the second order Taylor method is the more accurate than Euler's method, we require computing $f'(t,\, y)$.
Suppose we use a numerical approximation of the derivative of f.
By definition, the derivative of a function is the limiting value of the slope of the line connecting two nearby points on a curve,

$$f'(t_i,\, y_i) = \lim_{h\to 0}\frac{f(t_{i+1},\, y_{i+1}) - f(t_i,\, y_i)}{h}.$$

For small values of h, we can approximate the derivative with

$$f'(t_i,\, y_i) \approx \frac{f(t_{i+1},\, y_{i+1}) - f(t_i,\, y_i)}{h}.$$

By deriving the error term we find that

$$f'(t_i,\, y_i) = \frac{f(t_{i+1},\, y_{i+1}) - f(t_i,\, y_i)}{h} + \mathcal{O}(h).$$

Hence the second order Taylor polynomial becomes

$$\begin{aligned}
y_{i+1} &= y_i + hf\left(t_i,\, y_i\right) + \frac{h^2}{2} f'\left(t_i,\, y_i\right) + \mathcal{O}\left(h^3\right) \\
&= y_i + hf\left(t_i,\, y_i\right) + \frac{h^2}{2}\left[\frac{f\left(t_{i+1},\, y_{i+1}\right) - f\left(t_i,\, y_i\right)}{h} + \mathcal{O}\left(h\right)\right] + \mathcal{O}\left(h^3\right) \\
&= y_i + hf\left(t_i,\, y_i\right) + \frac{h}{2} f\left(t_{i+1},\, y_{i+1}\right) - \frac{h}{2} f\left(t_i,\, y_i\right) + \mathcal{O}\left(h^3\right) + \mathcal{O}\left(h^3\right) \\
&= y_i + \frac{h}{2} f\left(t_i,\, y_i\right) + \frac{h}{2} f\left(t_{i+1},\, y_{i+1}\right) + \mathcal{O}\left(h^3\right) \\
&= y_i + \frac{h}{2}\left[f\left(t_i,\, y_i\right) + f\left(t_{i+1},\, y_{i+1}\right)\right] + \mathcal{O}\left(h^3\right) \\
&\approx y_i + \frac{h}{2}\left[f\left(t_i,\, y_i\right) + f\left(t_{i+1},\, y_{i+1}\right)\right] \\
w_{i+1} &= w_i + \frac{h}{2}\left[f\left(t_i,\, w_i\right) + f\left(t_{i+1},\, w_{i+1}\right)\right]
\end{aligned}$$

As this formula involves itself, we will use Euler's method on $w_{i+1} = w_i + hf\left(t_i,\, w_i\right)$ on the RHS.

$$\begin{aligned}
w_{i+1} &= w_i + \frac{1}{2}\left[hf\left(t_i,\, w_i\right) + f\left(t_{i+1},\, w_{i+1}\right)\right] \\
&= w_i + \frac{1}{2}\left[\underbrace{hf\left(t_i,\, w_i\right)}_{k_1} + \underbrace{hf\left(t_{i+1},\, w_i + \underbrace{hf\left(t_i,\, w_i\right)}_{k_1}\right)}_{k_2}\right] \\
&= w_i + \frac{1}{2}\left(k_1 + k_2\right)
\end{aligned}$$

This result leads to what is known as the **modified Euler method**.

$$\begin{aligned}
w_0 &= \alpha \\
w_{i+1} &= w_i + \frac{1}{2}\left(k_1 + k_2\right) \\
k_1 &= hf\left(t_i,\, w_i\right) \\
k_2 &= hf\left(t_i + h,\, w_i + k_1\right)
\end{aligned}$$

This method has a local error of $\mathcal{O}\left(h^3\right)$, and therefore a global error of $\mathcal{O}\left(h^2\right)$.

2.7 Runge-Kutta Method

Another popular method that agrees with the Taylor method for high orders is the Runge-Kutta method. It is a fourth order method, referred to as RK4. It has the following form:

$$\begin{aligned} w_0 &= \alpha \\ w_{i+1} &= w_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \\ k_1 &= hf\left(t_i,\, w_i\right) \\ k_2 &= hf\left(t_i + \frac{h}{2},\, w_i + \frac{k_1}{2}\right) \\ k_3 &= hf\left(t_i + \frac{h}{2},\, w_i + \frac{k_2}{2}\right) \\ k_4 &= hf\left(t_i + h,\, w_i + k_3\right) \end{aligned}$$

for $i = 0,\, 1,\, \ldots,\, n-1$. The local error is $\mathcal{O}\left(h^5\right)$, and the global error is $\mathcal{O}\left(h^4\right)$.

3 Interpolation

Definition 3.1 (Interpolating function)**.** A function f is said to interpolate the data points

$$\left\{\left(x_0,\, y_0\right),\, \left(x_1,\, y_1\right),\, \ldots,\, \left(x_n,\, y_n\right)\right\}$$

if it satisfies $f\left(x_0\right) = y_0$, $f\left(x_1\right) = y_1$, …, $f\left(x_n\right) = y_n$.

There are many kinds of functions that may satisfy an interpolating function, however, the most common choice is a **polynomial**.
The following theorem addresses the existence and uniqueness of interpolating polynomials, given distinct x-values, or **abscissas**.

Theorem 3.0.1. *Let the abscissas x_0, x_1, …, x_n be distinct and the function values y_0, y_1, …, y_n be given. Then there exists a* ***unique*** *interpolating polynomial P_n of degree (at most) n, such that $P_n\left(x_i\right) = y_i$ for $i \in [0, n]$.*

The following methods are used to find the interpolating polynomial.

3.1 Lagrange Form

3.1.1 Two Data Points

Consider constructing the interpolating polynomial through the points $\left\{\left(x_0,\, y_0\right),\, \left(x_1,\, y_1\right)\right\}$. We require a polynomial of degree 1 to satisfy the interpolating function. Let $P_1\left(x\right) = a_0 + a_1 x$ satisfy the following conditions:

$$\begin{aligned} P_1\left(x_0\right) &= y_0 \quad &\iff\ a_0 + a_1 x_0 &= y_0 \\ P_1\left(x_1\right) &= y_1 \quad &\iff\ a_0 + a_1 x_1 &= y_1 \end{aligned}$$

solving this linear system of equations, we obtain the following:

$$a_0 = \frac{x_1 y_0 - x_0 y_1}{x_1 - x_0}$$
$$a_1 = \frac{y_1 - y_0}{x_1 - x_0}$$

so that

$$P_1(x) = \frac{x_1 y_0 - x_0 y_1}{x_1 - x_0} + \frac{y_1 - y_0}{x_1 - x_0} x.$$

To generalise this process, consider the following rearrangement of the polynomial:

$$\begin{aligned} P_1(x) &= \frac{x_1 y_0 - x_0 y_1}{x_1 - x_0} + \frac{y_1 - y_0}{x_1 - x_0} x \\ &= \frac{x_1 y_0}{x_1 - x_0} - \frac{x_0 y_1}{x_1 - x_0} + \frac{y_1 x}{x_1 - x_0} - \frac{y_0 x}{x_1 - x_0} \\ &= \left(\frac{-x}{x_1 - x_0} + \frac{x_1}{x_1 - x_0} \right) y_0 + \left(\frac{x}{x_1 - x_0} - \frac{x_0}{x_1 - x_0} \right) y_1 \\ &= \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1. \end{aligned}$$

This form clearly shows that each point satisfies the interpolating function.

- The coefficient of y_0 is **1** when $x = x_0$, and **0** when $x = x_1$.
- The coefficient of y_1 is **0** when $x = x_0$, and **1** when $x = x_1$.

we can then write $P_1(x)$ as

$$P_1(x) = L_{1,0}(x)\, y_0 + L_{1,1}(x)\, y_1$$

where $L_{1,0}(x) = \frac{x - x_1}{x_0 - x_1}$ and $L_{1,1}(x) = \frac{x - x_0}{x_1 - x_0}$ are the Lagrange basis functions. The first index corresponds to the degree of the polynomial, and the second index corresponds to the index of the point for which it is the coefficient of.

3.1.2 Three Data Points

For three data points, consider the reverse problem:

- The coefficient of y_0 is **1** when $x = x_0$, **0** when $x = x_1$, and **0** when $x = x_2$.
- The coefficient of y_1 is **0** when $x = x_0$, **1** when $x = x_1$, and **0** when $x = x_2$.
- The coefficient of y_2 is **0** when $x = x_0$, **0** when $x = x_1$, and **1** when $x = x_2$.

then solve for the Lagrange basis functions:

$$\begin{aligned} P_2(x) &= L_{2,0}(x)\, y_0 + L_{2,1}(x)\, y_1 + L_{2,2}(x)\, y_2 \\ &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} y_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} y_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} y_2. \end{aligned}$$

3.1.3 Arbitrary Number of Points

Given the abscissas $\{x_0, x_1, \ldots, x_n\}$ and function values $\{y_0, y_1, \ldots, y_n\}$ the polynomial $P_n(x)$ defined by:

$$P_n(x) = \sum_{i=0}^{n} L_{n,i}(x)\, y_i$$

where

$$L_{n,i}(x) = \prod_{j=0,\, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

is called the **Lagrange form** of the interpolating polynomial.
This arises from the fact that we can construct the Lagrange form using

$$\begin{aligned} P_n(x) &= L_{n,0}(x)\, y_0 + L_{n,1}(x)\, y_1 + \cdots + L_{n,n}(x)\, y_n \\ &= \sum_{i=0}^{n} L_{n,i}(x)\, y_i \end{aligned}$$

where the Lagrange basis function is as follows:

$$\begin{aligned} L_{n,i}(x) &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} \\ &= \prod_{j=0,\, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \end{aligned}$$

Note that $L_{n,i}(x)$ is equal to **1** when $x = x_i$, and **0** for all other abscissa:

$$L_{n,i}(x_j) = \delta_{ij}$$

3.2 Error in Polynomial Approximation

Theorem 3.2.1 (Lagrange form of remainder)**.** *Suppose that f is $n+1$ times continuously differentiable on $[a,b]$[3] with distinct abscissas $x_0, x_1, \ldots, x_n$ and interpolating polynomial $P_n(x)$. Then for all $x \in [a,b]$ there exists a $c \in [a,b]$ such that*

$$f(x) = P_n(x) + \frac{f^{(n+1)}(c)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n).$$

where the term

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)$$

is called the Lagrange form of the remainder term for $P_n(x)$.

[3] $a = \min(x_i)$ and $b = \max(x_i)$.

3.3 Newton's Divided Difference Form

While the Lagrange form is straightforward to analyse, it is not ideal for numerical computation. Suppose that we have found the interpolating polynomial through $n+1$ points, and are now interested in including a new point in the interpolation. This requires us to recompute the Lagrange basis functions for all $n+2$ points.

3.3.1 Derivation

The Newton form of the interpolating polynomial is given by:

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

The interpolation conditions:

$$\begin{aligned} P_n(x_0) &= y_0 \\ P_n(x_1) &= y_1 \\ &\vdots \\ P_n(x_n) &= y_n \end{aligned}$$

give the following system of equations for coefficients $a_0, a_1, \ldots, a_n$:

$$\begin{aligned} a_0 &= y_0 \\ a_0 + a_1(x_1 - x_0) &= y_1 \\ a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) &= y_2 \\ &\vdots \\ a_0 + a_1(x_n - x_0) + a_2(x_n - x_0)(x_n - x_1) + \cdots + a_n(x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1}) &= y_n \end{aligned}$$

Solving for the first three of these coefficients gives:

$$\begin{aligned} a_0 &= y_0 \\ a_1 &= \frac{y_1 - y_0}{x_1 - x_0} \\ a_2 &= \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0} \end{aligned}$$

To consisely represent these coefficients, we will need to define divided differences.

3.3.2 Divided Differences

Definition 3.2 (Zeroth divided difference)**.** The zeroth divided difference of f with respect to x_i is denoted $f[x_i]$ and defined by:

$$f[x_i] = y_i$$

Definition 3.3 (kth divided difference)**.** The kth divided difference of f with respect to $x_i, x_{i+1}, \ldots, x_{i+k}$ is denoted $f[x_i, x_{i+1}, \ldots, x_{i+k}]$ and defined by:

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

For example,

$$\begin{aligned} f[x_0] &= y_0 \\ f[x_1] &= y_1 \\ f[x_0,\, x_1] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \\ f[x_1,\, x_2] &= \frac{f[x_2] - f[x_1]}{x_1 - x_0} = \frac{y_2 - y_1}{x_2 - x_1} \\ f[x_0,\, x_1,\, x_2] &= \frac{f[x_1,\, x_2] - f[x_0,\, x_1]}{x_2 - x_0} = \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0} \end{aligned}$$

Using this notation we can rewrite the coefficients of P_n as

$$\begin{aligned} a_0 &= f[x_0] \\ a_1 &= f[x_0,\, x_1] \\ a_2 &= f[x_0,\, x_1,\, x_2] \\ &\vdots \\ a_n &= f[x_0,\, x_1,\, x_2,\, \ldots,\, x_n] \end{aligned}$$

so that

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

Definition 3.4 (Newton's divided difference form)**.** Given the abscissas $x_0,\, x_1,\, \ldots,\, x_n$ and the function values $y_0,\, y_1,\, \ldots,\, y_n$ the polynomial P_n defined by

$$P_n(x) = \sum_{k=0}^{n} f[x_0,\, x_1,\, \ldots,\, x_k] \left(\prod_{i=0}^{k-1} (x - x_i) \right)$$

is called the **Newton divided difference form of the interpolating polynomial**.

3.4 Newton's Forward Difference Form

In the case where abscissas are equally spaced, the Newton divided difference form can be simplified.

Definition 3.5 (Forward difference operator)**.** The **forward difference operator** Δ is defined by:

$$\Delta y_i = y_{i+1} - y_i$$

Higher order forward differences are defined by repeated application:

$$\begin{aligned} \Delta^2 y_i &= \Delta(\Delta y_i) = \Delta(y_{i+1} - y_i) = \Delta y_{i+1} - \Delta y_i = y_{i+2} - y_{i+1} - (y_{i+1} - y_i) \\ &= y_{i+2} - 2y_{i+1} + y_i \\ \Delta^3 y_i &= \Delta(\Delta^2 y_i) = \Delta(y_{i+2} - 2y_{i+1} + y_i) = \Delta y_{i+2} - \Delta y_{i+1} - \Delta y_i \\ &= y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i \end{aligned}$$

Theorem 3.4.1 (Simplified divided differences)**.** *If $x_0,\ x_1,\ \ldots,\ x_n$ are equally spaced, then*

$$f\left[x_0,\ x_1,\ \ldots,\ x_k\right] = \frac{\Delta^k y_0}{k!h^k}$$

where $h = x_{i+1} - x_i$ is the spacing between the abscissas.

Using the above theorem and making the substitution $x = x_0 + sh$ (so that $x_i = x_0 + ih$) where $s = \frac{x - x_0}{h}$, we can rewrite the Newton divided difference form as

$$\begin{aligned}
P_n(x) &= \sum_{k=0}^{n} f\left[x_0,\ x_1,\ \ldots,\ x_k\right] \left(\prod_{i=0}^{k-1} (x - x_i)\right) \\
&= \sum_{k=0}^{n} \frac{\Delta^k y_0}{k!h^k} \left(\prod_{i=0}^{k-1} ((x_0 + sh) - (x_0 + ih))\right) \\
&= \sum_{k=0}^{n} \frac{\Delta^k y_0}{k!h^k} \left(\prod_{i=0}^{k-1} (s - i)\, h\right) \\
&= \sum_{k=0}^{n} \frac{\Delta^k y_0}{k!h^k} \left(\prod_{i=0}^{k-1} (s - i) \prod_{i=0}^{k-1} h\right) \\
&= \sum_{k=0}^{n} \frac{\Delta^k y_0}{k!h^k} \left(\prod_{i=0}^{k-1} (s - i)\, h^k\right) \\
&= \sum_{k=0}^{n} \frac{\Delta^k y_0}{k!} s\,(s-1)\,(s-2)\cdots(s-(k-1)) \\
&= \sum_{k=0}^{n} \frac{\Delta^k y_0}{k!} s\,(s-1)\,(s-2)\cdots(s-k+1)\,\frac{(s-k)!}{(s-k)!} \\
&= \sum_{k=0}^{n} \Delta^k y_0 \frac{s!}{k!\,(s-k)!} \\
&= \sum_{k=0}^{n} \binom{s}{k} \Delta^k y_0
\end{aligned}$$

Definition 3.6 (Newton's forward difference form)**.** Given equally spaced abscissas $x_0,\ x_1,\ \ldots,\ x_n$ and the function values $y_0,\ y_1,\ \ldots,\ y_n$ the polynomial P_n defined by

$$P_n(x) = \sum_{k=0}^{n} \binom{\frac{x - x_0}{h}}{k} \Delta^k y_0$$

where $s = \frac{x - x_0}{h}$ with spacing $h = x_{i+1} - x_i$, is called the **Newton forward difference form of the interpolating polynomial**.

4 Root Finding

Given a function $f(x)$, it is often useful to find the values of x for which $f(x) = 0$. These values are called **roots** of that function. The process of finding roots is very straightforward for linear

functions of the form $f(x) = ax + b$, but becomes more complicated for nonlinear functions. In these instances, it is often not possible to find the roots analytically, and numerical methods must be used.
These methods are called **root finding methods**.

4.1 Bisection Method

The bisection method is a simple method based on the intermediate value theorem.

Theorem 4.1.1 (Intermediate Value Theorem)**.** *Let f be a continuous function on the interval $[a, b]$, and let k be any number between $f(a)$ and $f(b)$ inclusive. Then, there exists a number c in $[a, b]$ such that $f(c) = k$.*

$$f(a) \leq k \leq f(b) \iff \exists c \in [a, b] : f(c) = k.$$

Corollary 4.1.1.1. *Let f be a continuous function on the interval $[a, b]$. If $f(a) f(b) < 0$ ($f(a)$ and $f(b)$ have opposite signs), then there exists a number c in $[a, b]$ such that $f(c) = 0$.*

$$f(a) f(b) < 0 \iff \exists c \in [a, b] : f(c) = 0.$$

The bisection method is based on the following algorithm:

1. Find an interval $[a, b]$ such that $f(a) f(b) < 0$[4].
2. Find the midpoint p of the interval $[a, b]$ and evaluate $f(p)$.
   - If $f(p) = 0$, then p is a root of f.
   - Otherwise, the root lies in either $[a, p]$ or $[p, b]$.
     - If $f(a) f(p) < 0$, then p becomes the new b and the root lies in $[a, p]$.
     - If $f(p) f(b) < 0$, then p becomes the new a and the root lies in $[p, b]$.
3. Go to step 2 and repeat until a satisfactory approximation of the root is found.

This can be expressed in a table format:

Iteration	a	b	$p = \frac{a+b}{2}$	$f(a)$	$f(b)$	$f(p)$
1	a_0	b_0	$p_0 = \frac{a_0+b_0}{2}$	$f(a_0)$	$f(b_0)$	$f(p_0)$
2	a_1	b_1	$p_1 = \frac{a_1+b_1}{2}$	$f(a_1)$	$f(b_1)$	$f(p_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

4.2 Fixed-Point Iteration

Fixed-point iteration is based on rewriting $f(x) = 0$ as

$$x = g(x)$$

[4]This procedure is known as **bracketing** the root.

The equation is solved if we can find a number p such that $g(p) = p$, called the **fixed-point** of g. Using an initial guess, x_0, and computing $x_2 = g(x_1)$, $x_3 = g(x_2)$, and so on, we can approximate the fixed-point of g. Under certain conditions, the **sequence**, $\{x_n\}$, will **converge** to the fixed-point of g and thus solve the equation $f(x) = 0$.

Theorem 4.2.1 (Brouwer's fixed-point theorem)**.** *Let g be a continuous function on $[a, b]$ with $g(x)$ in $[a, b]$ for all x. Furthermore, let g be differentiable on (a, b) and let a positive constant $k < 1$ exist such that $|g'(x)| \leq k$ for all x in (a, b).*
Then, g has a unique fixed-point in $[a, b]$, and the iteration $x_{n+1} = g(x_n)$ will converge to this point for all initial guesses x_0 in $[a, b]$.

$$(g(x) \in [a,b] : \forall x \in [a,b]) \wedge (\exists k \in \mathbb{R}^+ : k < 1 : |g'(x)| \leq k : \forall x \in (a,b)) \implies \exists! p \in [a,b] : g(p) = p.$$

4.3 Newton's Method

Newton's method is one of the most widely used methods for solving nonlinear equations. It approximates the solution of $f(x) = 0$ by finding the root of the tangent line to f at each iteration. The next iterate is then the intersection of the tangent line with the x-axis.
The first degree Taylor polynomial of f at x_n gives us the tangent line to f at x_n:

$$f(x) \approx f(x_n) + f'(x_n)(x - x_n)$$

if we solve this equation, we find:

$$\begin{aligned} f(x_n) + f'(x_n)(x - x_n) &= 0 \\ f'(x_n)(x - x_n) &= -f(x_n) \\ x - x_n &= -\frac{f(x_n)}{f'(x_n)} \\ x &= x_n - \frac{f(x_n)}{f'(x_n)} \end{aligned}$$

This gives us the sequence:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

for $n \geq 0$.

4.4 Secant Method

While Newton's method converges rapidly, it requires the derivative of f to be known. The secant method approximates the derivative of f by using the slope of the secant line between two points, $(x_{n-1},\ f(x_{n-1}))$ and $(x_n,\ f(x_n))$. That is, $f'(x_n)$ is approximated by

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

This gives us the sequence

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

for $n \geq 1$. Note that two initial values are required to start the iteration.

4.5 Convergence of Fixed-Point Iteration

When the fixed-point iteration converges, the sequence it generates, $\{x_n\}$, satisfies (for sufficiently large n)

$$|x_{n+1} - p| \approx \lambda |x_n - p|$$

where p is the fixed-point of g and $0 < \lambda < 1$. Therefore the **error** in the solution decreases by a factor of λ in the long run of each iteration.

4.6 Convergence of Newton's Method

When Newton's method converges to the root p, the sequence it generates, $\{x_n\}$, satisfies (for sufficiently large n)

$$|x_{n+1} - p| \approx \lambda |x_n - p|^2$$

for $\lambda > 0$. This means that the number of correct digits approximately *doubles* with each iteration.

4.7 Convergence of the Secant Method

When the secant method converges to the root p, the sequence it generates, $\{x_n\}$, satisfies (for sufficiently large n)

$$|x_{n+1} - p| \approx \lambda |x_n - p|^r$$

for $\lambda > 0$ and $r = \left(1 + \sqrt{5}\right)/2 \approx 1.618$ (Golden ratio). Thus the secant method has a slower *rate of convergence* than Newton's method.

5 Numerical Differentiation

5.1 First Derivative

Often no explicit derivative of a function f can be found, for example when given a table of values for $f(x)$. In this case, the Taylor polynomial can be used.
Provided the function f is sufficiently differentiable, we can express the Taylor polynomial of f at x_0 as

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + \frac{f^{(n+1)}(c)}{(n+1)!}(x - x_0)^{n+1}.$$

If we let $h = x - x_0$, we can write this as

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \cdots + \frac{h^n}{n!}f^{(n)}(x_0) + \frac{h^{n+1}}{(n+1)!}f^{(n+1)}(c).$$

where c is between x_0 and $x_0 + h$.

5.1.1 First Order Approximations

Using the Taylor polynomial of degree 1, we can obtain a first order approximation for $f'(x_0)$. The formula

$$f'(x_0) = \frac{f(x_0+h) - f(x_0)}{h} - \frac{h}{2}f''(c)$$

is known as the **first order forward difference** for $f'(x_0)$. Replacing h with $-h$ gives the **first order backward difference** for $f'(x_0)$. The

$$f'(x_0) = \frac{f(x_0) - f(x_0-h)}{h} + \frac{h}{2}f''(c).$$

In both cases, the error term is $\mathcal{O}(h)$.

5.1.2 Second Order Approximations

Consider the Taylor polynomial of degree 2 for $f(x_0+h)$ and $f(x_0-h)$:

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(c_1)$$
$$f(x_0-h) = f(x_0) - hf'(x_0) + \frac{h^2}{2!}f''(x_0) - \frac{h^3}{3!}f'''(c_2)$$

where c_1 is between x_0 and x_0+h, and c_2 is between x_0-h and x_0. Subtracting the two equations gives

$$f(x_0+h) - f(x_0-h) = 2hf'(x_0) + \frac{h^3}{3!}\left(f'''(c_1) + f'''(c_2)\right)$$
$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h} - \frac{h^2}{6}f'''(c)$$

where $f'''(c) = \frac{f'''(c_1)+f''(c_2)}{2}$, for $c_1 \le c \le c_2$ (Intermediate Value Theorem), and c is between x_0-h and x_0+h. This is known as the **second order central difference** approximation for $f'(x_0)$. The error term is $\mathcal{O}(h^2)$.

5.2 Second Derivative

It is also possible to obtain a second order central difference approximation for the second derivative, $f''(x_0)$ using the Taylor polynomial of degree 3 for $f(x_0+h)$ and $f(x_0-h)$:

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(c_1)$$
$$f(x_0-h) = f(x_0) - hf'(x_0) + \frac{h^2}{2!}f''(x_0) - \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(c_2)$$

where c_1 and c_2 are between x_0 and x_0+h and between x_0-h and x_0, respectively. Adding the two equations gives

$$f(x_0+h) + f(x_0-h) = 2f(x_0) + h^2f''(x_0) + \frac{h^4}{4!}\left(f^{(4)}(c_1) + f^{(4)}(c_2)\right)$$
$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} - \frac{h^2}{12}f^{(4)}(c)$$

where $f^{(4)}(c) = \frac{f^{(4)}(c_1)+f^{(4)}(c_2)}{2}$, for $c_1 \leq c \leq c_2$ (Intermediate Value Theorem), and c is between $x_0 - h$ and $x_0 + h$.

5.3 Instability

When performing numerical differentiation, it is important to realise that as h decreases, the first and second derivative formulas will require the subtraction of nearly equal values, which can lead to **numerical instability**. It is therefore important to choose a value of h so that it is not too large, which results in inaccuracies due to truncation error, or too small, which results in inaccuracies due to roundoff error.

6 Numerical Integration

Indefinite integration is defined as the anti-derivative of a function f, denoted by $F(x)$, such that

$$F'(x) = f(x).$$

The definite integral of f over the interval $[a, b]$ is defined as

$$I = \int_a^b f(x)\,\mathrm{d}x.$$

Numerical integration, also known as **quadrature**, is concerned with approximating the definite integral of a function f over a given interval $[a, b]$. It approximates the integral $\int_a^b f(x)\,\mathrm{d}x$ by a weighted sum of function values.

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} w_i f(x_i)$$

where w_i are the weights and x_i are the abscissas. The primary rule for developing integral approximations is the interpolating polynomial.

6.1 Trapezoidal Rule

Consider dividing the interval $[a, b]$ into n subintervals, each of width h. Then $h = \frac{b-a}{n}$, and let $x_i = a + ih$ for $i = 0, 1, \ldots, n$, so that $x_0 = a$ and $x_n = b$.
The **trapezoidal rule** uses the interpolating polynomial of degree 1 to approximate $f(x)$ over each subinterval $[x_i, x_{i+1}]$. Consider the interpolating polynomial which passes through the points $[x_0, y_0]$ and $[x_1, y_1]$ where $y_0 = f(x_0)$ and $y_1 = f(x_1)$. The interpolating polynomial is

$$P_1(x) = y_0 + s\Delta y_0$$

using the Newton forward difference form. Using the change of variables $x = x_0 + sh$, $\mathrm{d}x = h\,\mathrm{d}s$,

and the limits of integration become $[0,1]$. The integral of P_1 over $[0,1]$ is

$$\begin{aligned}\int_{x_0}^{x_1} f(x)\,\mathrm{d}x &= \int_0^1 P_1(x)\,\mathrm{d}x \\ &= \int_0^1 (y_0 + s\Delta y_0)\,h\,\mathrm{d}s \\ &= h\int_0^1 y_0 + s\Delta y_0\,\mathrm{d}s \\ &= h\left[y_0 s + (y_1 - y_0)\,\frac{s^2}{2}\right]_0^1 \\ &= h\left[y_0 + \frac{y_1 - y_0}{2}\right] \\ &= \frac{h}{2}(y_0 + y_1)\end{aligned}$$

It follows that:

$$\int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x \approx \frac{h}{2}\left[f(x_{i-1}) + f(x_i)\right]$$

for all $i = 1, 2, \ldots, n$. The trapezoidal rule is therefore

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &= \int_{x_0}^{x_1} f(x)\,\mathrm{d}x + \int_{x_1}^{x_2} f(x)\,\mathrm{d}x + \cdots + \int_{x_{n-1}}^{x_n} f(x)\,\mathrm{d}x \\ &\approx \frac{h}{2}\left[f(x_0) + f(x_1)\right] + \frac{h}{2}\left[f(x_1) + f(x_2)\right] + \cdots + \frac{h}{2}\left[f(x_{n-1}) + f(x_n)\right] \\ &= \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right] \\ &= \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]\end{aligned}$$

Therefore, for a twice continuously differentiable function f on $[a,b]$, with $h = \frac{b-a}{n}$ and $x_i = a + ih$, for $i = 0, 1, \ldots, n$,

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right] - \frac{(b-a)\,h^2}{12} f''(c)$$

where c is between a and b. The error in the trapezoidal rule is $\mathcal{O}(h^2)$, and this method is exact for polynomials of degree one or less, as the error term is zero.

6.2 Simpson's Rule

The **Simpson's rule** uses the interpolating polynomial of degree 2 to approximate $f(x)$ over each subinterval $[x_i, x_{i+2}]$. Consider the interpolating polynomial which passes through the points $[x_0, y_0]$, $[x_1, y_1]$, and $[x_2, y_2]$ where $y_i = f(x_i)$. The interpolating polynomial is

$$P_2(x) = y_0 + s\Delta y_0 + \frac{s(s-1)}{2}\Delta^2 y_0$$

using the Newton forward difference form. Using the change of variables $x = x_0 + sh$, $\mathrm{d}x = h\,\mathrm{d}s$, and the limits of integration become $[0, 2]$. The integral of P_2 over $[0, 2]$ is

$$\begin{aligned}
\int_{x_0}^{x_2} f(x)\,\mathrm{d}x &= \int_0^2 P_2(x)\,\mathrm{d}x \\
&= \int_0^2 \left(y_0 + s\Delta y_0 + \frac{s(s-1)}{2}\Delta^2 y_0\right) h\,\mathrm{d}s \\
&= h\int_0^1 y_0 + s\Delta y_0 + \left(\frac{1}{2}s^2 - \frac{1}{2}s\right)\Delta^2 y_0\,\mathrm{d}s \\
&= h\left[y_0 s + (y_1 - y_0)\frac{s^2}{2} + \left(\frac{1}{6}s^3 - \frac{1}{4}s^2\right)(y_2 - 2y_1 + y_0)\right]_0^2 \\
&= h\left[2y_0 + 2(y_1 - y_0) + \left(\frac{8}{6} - \frac{1}{4}4\right)(y_2 - 2y_1 + y_0)\right] \\
&= \frac{h}{3}(y_0 + 4y_1 + y_2)
\end{aligned}$$

It follows that:

$$\int_{x_{2i-2}}^{x_{2i}} f(x)\,\mathrm{d}x \approx \frac{h}{3}\left[f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})\right]$$

for all $i = 1, 2, \ldots, n/2$. Simpson's rule is therefore

$$\begin{aligned}
\int_a^b f(x)\,\mathrm{d}x &= \int_{x_0}^{x_2} f(x)\,\mathrm{d}x + \int_{x_2}^{x_4} f(x)\,\mathrm{d}x + \cdots + \int_{x_{n-2}}^{x_n} f(x)\,\mathrm{d}x \\
&\approx \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right] + \frac{h}{3}\left[f(x_2) + 4f(x_3) + f(x_4)\right] + \cdots \\
&\quad + \frac{h}{3}\left[f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right] \\
&= \frac{h}{3}\left[f(x_0) + 4\left(f(x_1) + f(x_3) + \cdots + f(x_{n-1})\right)\right. \\
&\quad \left. + 2\left(f(x_2) + f(x_4) + \cdots + f(x_{n-2})\right) f(x_n)\right] \\
&= \frac{h}{3}\left[f(x_0) + 4\sum_{i=1}^{\frac{n}{2}} f(x_{2i-1}) + 2\sum_{i=1}^{\frac{n}{2}-1} f(x_{2i}) + f(x_n)\right]
\end{aligned}$$

Therefore, for a four times continuously differentiable function f on $[a, b]$, with $h = \frac{b-a}{n}$ (with n even) and $x_i = a + ih$, for $i = 0, 1, \ldots, n$,

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{3}\left[f(x_0) + 4\sum_{i=1}^{\frac{n}{2}} f(x_{2i-1}) + 2\sum_{i=1}^{\frac{n}{2}-1} f(x_{2i}) + f(x_n)\right] - \frac{(b-a)h^4}{180} f^{(4)}(c)$$

where c is between a and b. The error in Simpson's rule is $\mathcal{O}(h^4)$, and this method is exact for polynomials of degree three or less, as the error term is zero.

7 Linear Systems

Given a linear system with n knowns and n unknowns, we can represent this system using matrix notation:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

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

where $\mathbf{A}$ is the **coefficient matrix**, $\boldsymbol{x}$ is the **solution vector**, and $\boldsymbol{b}$ is the **right hand side vector**. Alternatively, this equation can be written using an **augmented matrix**, by augmenting $\mathbf{A}$ with $\boldsymbol{b}$:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right]$$

the vertical bar separates the coefficient matrix from the right hand side vector.

7.1 Triangular Systems

A linear system that takes the form

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{nn} & b_n \end{array}\right]$$

is known as an **upper triangular system**. Similarly, a linear system that takes the form

$$\left[\begin{array}{cccc|c} a_{11} & 0 & \cdots & 0 & b_1 \\ a_{21} & a_{22} & \ddots & \vdots & b_2 \\ \vdots & \vdots & \ddots & 0 & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right]$$

is known as a **lower triangular system**.

7.2 Backward Substitution

When solving an upper triangular system, we can use **backward substitution** to solve for the unknowns. This method starts with the last unknown, x_n, and solves for it using the last equation in the system. Then, we use the second to last equation to solve for x_{n-1}, and so on. This leads to

the following algorithm:

$$\begin{aligned} x_n &= \frac{b_n}{a_{nn}} \\ x_{n-1} &= \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}} \\ &\vdots \\ x_i &= \frac{b_i - a_{in}x_n - a_{i,n-1}x_{n-1} - \cdots - a_{i,i+1}x_{i+1}}{a_{ii}} \\ &= \frac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}} \end{aligned}$$

7.3 Forward Substitution

When solving a lower triangular system, we can use **forward substitution** to solve for the unknowns. This method starts with the first unknown, x_1, and solves for it using the first equation in the system. Then, we use the second equation to solve for x_2, and so on. This leads to the following algorithm:

$$\begin{aligned} x_1 &= \frac{b_1}{a_{11}} \\ x_2 &= \frac{b_2 - a_{21}x_1}{a_{22}} \\ &\vdots \\ x_i &= \frac{b_i - a_{i,1}x_1 - a_{i,2}x_2 - \cdots - a_{i,i-1}x_{i-1}}{a_{ii}} \\ &= \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j}{a_{ii}} \end{aligned}$$

7.4 LU Decomposition

The **LU decomposition** is a method for factoring a square matrix into a lower triangular matrix **L** and an upper triangular matrix **U**:

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

to solve the linear system:

$$\begin{aligned} \mathbf{A}\boldsymbol{x} &= \boldsymbol{b} \\ \mathbf{LU}\boldsymbol{x} &= \boldsymbol{b} \\ \mathbf{L}\boldsymbol{z} &= \boldsymbol{b} \qquad\qquad (\mathbf{U}\boldsymbol{x} = \boldsymbol{z}) \end{aligned}$$

where $\boldsymbol{z}$ can be solved using forward substitution, and then $\boldsymbol{x}$ can be solved using backward substitution. The matrices $\mathbf{L}$ and $\mathbf{U}$ can be found by considering the following:

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

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{bmatrix}$$

If we perform this product, we get the following:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} & \cdots & l_{21}u_{1n} + u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1}u_{11} & l_{n1}u_{12} + l_{n2}u_{22} & \cdots & l_{n1}u_{1n} + l_{n2}u_{2n} + \cdots + u_{nn} \end{bmatrix}$$

which allows us to solve for u_{ij} and l_{ij}. This can be done by considering the first row, then the first column, then the second row, then the second column, and so on.
In general, the equations to consider are, for l_{ij}

$$l_{ij} = \begin{cases} \frac{a_{ij}}{u_{jj}} & \text{if } i > j \text{ and } j = 1 \\ \frac{a_{ij} - \sum_{k=1}^{j} l_{ik}u_{kj}}{u_{jj}} & \text{if } i > j \text{ and } j > 1 \\ 1 & \text{if } i = j \\ 0 & \text{if } i < j \end{cases}$$

and for u_{ij}

$$u_{ij} = \begin{cases} a_{ij} & \text{if } i \leq j \text{ and } i = 1 \\ a_{ij} - \sum_{k=1}^{i} l_{ik}u_{kj} & \text{if } i \leq j \text{ and } i > 1 \\ 0 & \text{if } i > j \end{cases}$$

7.5 Cholesky Decomposition

The **Cholesky decomposition** is a method for factoring a symmetric, positive definite matrix into a lower triangular matrix $\mathbf{L}$ and its transpose:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T$$

to solve the linear system:

$$\begin{aligned} \mathbf{A}\boldsymbol{x} &= \boldsymbol{b} \\ \mathbf{L}\mathbf{L}^\top\boldsymbol{x} &= \boldsymbol{b} \\ \mathbf{L}\boldsymbol{z} &= \boldsymbol{b} \qquad (\mathbf{L}^\top\boldsymbol{x} = \boldsymbol{z}) \end{aligned}$$

where $\boldsymbol{z}$ can be solved using forward substitution, and then $\boldsymbol{x}$ can be solved using backward substitution. The matrix $\mathbf{L}$ can be found by considering the following:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^\top$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & \cdots & l_{n,n-1} & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ 0 & l_{22} & \cdots & l_{n2} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & l_{nn} \end{bmatrix}$$

If we perform this product, we get the following:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11}^2 & l_{11}l_{21} & \cdots & l_{11}l_{n1} \\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & \cdots & l_{21}l_{n1} + l_{22}l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1}l_{11} & l_{n1}l_{21} + l_{n2}l_{22} & \cdots & l_{n1}^2 + l_{n2}^2 + \cdots + l_{nn}^2 \end{bmatrix}$$

which allows us to solve for l_{ij}. This can be done by solving each column of $\mathbf{L}$.