

# Introductory Computational Mathematics

Semester 2, 2022

*Dr Elliot Carr*

Tarang Janawalkar

This work is licensed under a Creative Commons  
“Attribution-NonCommercial-ShareAlike 4.0 International” license.



# Contents

<b>Contents</b>	<b>1</b>
<b>1 Preliminaries</b>	<b>2</b>
1.1 Errors . . . . .	2
1.2 Floating Point Arithmetic . . . . .	2
1.2.1 Representing Real Numbers as Floating Point Numbers . . . . .	3
1.2.2 Converting between Floating Point Number Systems . . . . .	3
1.2.3 IEEE Floating Point Standard . . . . .	4
1.3 Catastrophic Cancellation . . . . .	4
1.4 Taylor Polynomials . . . . .	5
1.5 Taylor Series . . . . .	6
<b>2 Ordinary Differential Equations</b>	<b>6</b>
2.1 Initial Value Problems . . . . .	6
2.2 Time Discretisation . . . . .	6
2.3 Euler's Method . . . . .	6
2.4 Local and Global Error . . . . .	7
2.5 Second Order Taylor Method . . . . .	7
2.6 Modified Euler Method . . . . .	8
2.7 Runge-Kutta Method . . . . .	9
<b>3 Interpolation</b>	<b>9</b>
3.1 Lagrange Form . . . . .	10
3.1.1 Two Data Points . . . . .	10
3.1.2 Three Data Points . . . . .	10
3.1.3 Arbitrary Number of Points . . . . .	11
3.2 Error in Polynomial Approximation . . . . .	11
3.3 Newton's Divided Difference Form . . . . .	12
3.3.1 Derivation . . . . .	12
3.3.2 Divided Differences . . . . .	13
3.4 Newton's Forward Difference Form . . . . .	14
<b>4 Root Finding</b>	<b>15</b>
4.1 Bisection Method . . . . .	15
4.2 Fixed-Point Iteration . . . . .	16
4.3 Newton's Method . . . . .	16
4.4 Secant Method . . . . .	17
4.5 Convergence of Fixed-Point Iteration . . . . .	17

# 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

$$f = \pm (d_1.d_2d_3 \dots d_k)_\beta \times \beta^e$$

where  $e \in \mathbb{Z}$  is the **exponent** satisfying  $m \leq e \leq M$ . The quantity  $d_1.d_2d_3 \dots d_k$  is known as the **significand**, where  $d_i$  are base- $\beta$  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<sup>1</sup>) and  $\beta = 16$  (hexadecimal).

---

<sup>1</sup>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:

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

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*

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

*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<sup>2</sup>  $x$  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  $fl : \mathbb{R} \rightarrow \mathbb{F}(\beta, k, m, M)$ , defined as function which maps real numbers  $x$  to the nearest element in  $\mathbb{F}$ . To determine  $fl(x)$ :

1. Express  $x$  in base  $\beta$ .
2. Express  $x$  in scientific form.

---

<sup>2</sup> $x$  must satisfy  $\min(\mathbb{F}) \leq x \leq \max(\mathbb{F})$ .

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	$1.80 \times 10^{308}$
Smallest representable positive number	$2.23 \times 10^{-308}$
Special values	$\pm 0, \pm \infty, \text{NaN}$

## 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

$$\cos(x) = \sum_{k=0}^n \frac{(-1)^k}{(2k)!} x^{2k}.$$

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

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

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  $n \rightarrow \infty$  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,

$$\begin{aligned} y(a) &= \alpha \\ \frac{dy(t)}{dt} &= f(t, y(t)), \quad a \leq t \leq b \end{aligned}$$

Divide the interval  $[a, b]$  into  $n$  subintervals, each with width  $h = (b - a) / n$ . Then define  $t_i = a + ih$ , for  $i = 0, 1, \dots, n$ , 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  $i = 0, 1, \dots, n$ .

### 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  $i = 0, 1, \dots, n-1$ .

## 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  $p$ th 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  $i = 0, 1, \dots, n-1$ . 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,

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

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

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

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

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

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

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

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

## 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:

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

for  $i = 0, 1, \dots, n-1$ . The local error is  $\mathcal{O}(h^4)$ , and the global error is  $\mathcal{O}(h^3)$ .

## 3 Interpolation

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

$$\{(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)\}$$

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

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  $x_0, x_1, \dots, x_n$  be distinct and the function values  $y_0, y_1, \dots, y_n$  be given. Then there exists a **unique** interpolating polynomial  $P_n$  of degree (at most)  $n$ , such that  $P_n(x_i) = y_i$  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  $\{(x_0, y_0), (x_1, y_1)\}$ . We require a polynomial of degree 1 to satisfy the interpolating function. Let  $P_1(x) = a_0 + a_1x$  satisfy the following conditions:

$$\begin{aligned} P_1(x_0) &= y_0 && \iff a_0 + a_1x_0 = y_0 \\ P_1(x_1) &= y_1 && \iff a_0 + a_1x_1 = y_1 \end{aligned}$$

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

$$\begin{aligned} a_0 &= \frac{x_1y_0 - x_0y_1}{x_1 - x_0} \\ a_1 &= \frac{y_1 - y_0}{x_1 - x_0} \end{aligned}$$

so that

$$P_1(x) = \frac{x_1y_0 - x_0y_1}{x_1 - x_0} + \frac{y_1 - y_0}{x_1 - x_0}x.$$

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

$$\begin{aligned} P_1(x) &= \frac{x_1y_0 - x_0y_1}{x_1 - x_0} + \frac{y_1 - y_0}{x_1 - x_0}x \\ &= \frac{x_1y_0}{x_1 - x_0} - \frac{x_0y_1}{x_1 - x_0} + \frac{y_1x}{x_1 - x_0} - \frac{y_0x}{x_1 - x_0} \\ &= \left( \frac{-x}{x_1 - x_0} + \frac{x_1}{x_1 - x_0} \right) y_0 + \left( \frac{x}{x_1 - x_0} - \frac{x_0}{x_1 - x_0} \right) y_1 \\ &= \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1. \end{aligned}$$

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  $\{x_0, x_1, \dots, x_n\}$  and function values  $\{y_0, y_1, \dots, y_n\}$  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

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

where the Lagrange basis function is as follows:

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

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  $[a, b]$ <sup>3</sup> with distinct abscissas  $x_0, x_1, \dots, x_n$  and interpolating polynomial  $P_n(x)$ . Then for all  $x \in [a, b]$  there exists a  $c \in [a, b]$  such that*

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

---

<sup>3</sup> $a = \min(x_i)$  and  $b = \max(x_i)$ .

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  $P_n(x)$ .

### 3.3 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  $a_0, a_1, \dots, a_n$ :

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

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

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

For example,

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

Using this notation we can rewrite the coefficients of  $P_n$  as

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

so that

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

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

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

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**  $\Delta$  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  $x_0, x_1, \dots, x_n$  are equally spaced, then*

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

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

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

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

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

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  $f(a)f(b) < 0$ <sup>4</sup>.
2. Find the midpoint  $p$  of the interval  $[a, b]$  and evaluate  $f(p)$ .
  - If  $f(p) = 0$ , then  $c$  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.

---

<sup>4</sup>This procedure is known as **bracketing** the root.



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

$$x = g(x)$$

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  $\lambda$  in the long run of each iteration.

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

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

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