Introduction to Numerical Analysis

Lecture Notes for MA 214, Spring 2013

Instructors:

S. Baskar and S. Sivaji Ganesh

Department of Mathematics
Indian Institute of Technology Bombay
Powai, Mumbai 400 076.

Contents

CHAPTER 1

Mathematical Preliminaries

This chapter reviews some of the concepts and results from calculus that are frequently used in this course. We recall important definitions and theorems whose proof is outlined briefly. The readers are assumed to be familiar with a first course in calculus.

In Section 1.1, we introduce sequences of real numbers and discuss the concept of limit and continuity in Section 1.2 with the intermediate value theorem. This theorem plays a basic role in finding initial guesses in iterative methods for solving nonlinear equations. In Section 1.3 we define derivative of a function, and prove Rolle's theorem and mean-value theorem for derivatives. The mean-value theorem for integration is discussed in Section 1.4. These two theorems are crucially used in devising methods for numerical integration and differentiation. Finally, Taylor's theorem is discussed in Section 1.5, which is essential for derivation and error analysis of almost all numerical methods discussed in this course. In Section 1.6 we introduce tools useful in discussing speed of convergence of sequences and rate at which a function $f(x)$ approaches a point $f(x_0)$ as $x \to x_0$.

Let $a, b \in \mathbb{R}$ be such that $a < b$. We use the notation $[a, b]$ for the closed interval

$$[a, b] = \{\, x \in \mathbb{R} \,:\, a \leq x \leq b \,\}.$$

and (a, b) for the open interval

$$(a, b) = \{\, x \in \mathbb{R} \,:\, a < x < b \,\}.$$

1.1 Sequences of Real Numbers

Definition 1.1 (Sequence).

A **sequence** *of real numbers is an ordered list of real numbers*

$$a_1, a_2, \cdots, a_n, a_{n+1}, \cdots$$

In other words, a **sequence** *is a function that associates the real number a_n for each natural number n. The notation $\{a_n\}$ is often used to denote the sequence $a_1, a_2, \cdots, a_n, a_{n+1}, \cdots$*

The concept of convergence of a sequence plays an important role in numerical analysis, for instance when approximating a solution x of a certain problem via an iterative procedure that produces a sequence of approximation. Here, we are interested in knowing the convergence of the sequence of approximate solutions to the exact solution x.

Definition 1.2 (Convergence of a Sequence).

Let $\{a_n\}$ be a sequence of real numbers and let L be a real number. The sequence $\{a_n\}$ is said to **converge** *to L, and we write*

$$\lim_{n\to\infty} a_n = L (or\ a_n \to L\ as\ n \to \infty),$$

if for every $\epsilon > 0$ there exists a natural number N such that

$$|a_n - L| < \epsilon \quad whenever \quad n \geq N.$$

The real number L is called the **limit** *of the sequence $\{a_n\}$.*

Theorem 1.3 (Sandwich Theorem).

Let $\{a_n\}, \{b_n\}, \{c_n\}$ be sequences of real numbers such that

(1) *there exists an $n_0 \in \mathbb{N}$ such that for every $n \geq n_0$, the sequences satisfy the inequalities $a_n \leq b_n \leq c_n$ and*

(2) $\lim_{n\to\infty} a_n = \lim_{n\to\infty} c_n = L$.

Then the sequence $\{b_n\}$ also converges and $\lim_{n\to\infty} b_n = L$. □

Definition 1.4 (Bounded Sequence).

A sequence $\{a_n\}$ is said to be a **bounded sequence** *if there exists a real number M such that*

$$|a_n| \leq M \quad for\ every\ \ n \in \mathbb{N}.$$

Theorem 1.5 (Bolzano-Weierstrass theorem). *Every bounded sequence $\{a_n\}$ has a convergent subsequence $\{a_{n_k}\}$.*

The following result is very useful in computing the limit of a sequence sandwiched between two sequences having a common limit.

Definition 1.6 (Monotonic Sequences).

A sequence $\{a_n\}$ of real numbers is said to be

(1) *an* **increasing sequence** *if $a_n \leq a_{n+1}$, for every $n \in \mathbb{N}$.*

(2) *a* **strictly increasing sequence** *if $a_n < a_{n+1}$, for every $n \in \mathbb{N}$.*

(3) *a* **decreasing sequence** *if $a_n \geq a_{n+1}$, for every $n \in \mathbb{N}$.*

(4) *a* **strictly decreasing sequence** *if $a_n > a_{n+1}$, for every $n \in \mathbb{N}$.*

A sequence $\{a_n\}$ is said to be a **(strictly) monotonic sequence** *if it is either (strictly) increasing or (strictly) decreasing.*

Theorem 1.7. *Bounded monotonic sequences always converge.* □

Note that any bounded sequence need not converge. The monotonicity in the above theorem is very important. The following result is known as "algebra of limits of sequences".

Theorem 1.8. *Let* $\{a_n\}$ *and* $\{b_n\}$ *be two sequences. Assume that* $\lim_{n\to\infty} a_n$ *and* $\lim_{n\to\infty} b_n$ *exist. Then*

(1) $\lim_{n\to\infty} (a_n + b_n) = \lim_{n\to\infty} a_n + \lim_{n\to\infty} b_n$.

(2) $\lim_{n\to\infty} c\, a_n = c \lim_{n\to\infty} a_n$, *for any number* c.

(3) $\lim_{n\to\infty} a_n b_n = \lim_{n\to\infty} a_n \lim_{n\to\infty} b_n$.

(4) $\lim_{n\to\infty} \frac{1}{a_n} = \frac{1}{\lim_{n\to\infty} a_n}$, *provided* $\lim_{n\to\infty} a_n \neq 0$.

1.2 Limits and Continuity

In the previous section, we introduced the concept of limit for a sequences of real numbers. We now define the "limit" in the context of functions.

Definition 1.9 (Limit of a Function).

(1) *Let* f *be a function defined on the left side (or both sides) of* a*, except possibly at* a *itself. Then, we say "the* **left-hand limit** *of* $f(x)$ *as* x *approaches* a*, equals* l*" and denote*

$$\lim_{x\to a-} f(x) = l,$$

if we can make the values of $f(x)$ *arbitrarily close to* l *(as close to* l *as we like) by taking* x *to be sufficiently close to* a *and* x *less than* a*.*

(2) *Let* f *be a function defined on the right side (or both sides) of* a*, except possibly at* a *itself. Then, we say "the* **right-hand limit** *of* $f(x)$ *as* x *approaches* a*, equals* r*" and denote*

$$\lim_{x\to a+} f(x) = r,$$

if we can make the values of $f(x)$ *arbitrarily close to* r *(as close to* r *as we like) by taking* x *to be sufficiently close to* a *and* x *greater than* a*.*

(3) *Let* f *be a function defined on both sides of* a*, except possibly at* a *itself. Then, we say"the* **limit** *of* $f(x)$ *as* x *approaches* a*, equals* L*" and denote*

$$\lim_{x\to a} f(x) = L,$$

if we can make the values of $f(x)$ *arbitrarily close to* L *(as close to* L *as we like) by taking* x *to be sufficiently close to* a *(on either side of* a*) but not equal to* a*.*

Remark 1.10. Note that in each of the above definitions the value of the function f at the point a does not play any role. In fact, the function f need not be defined at the point a. □

In the previous section, we have seen some limit laws in the context of sequences. Similar limit laws also hold for limits of functions. We have the following result, often referred to as "the limit laws" or as "algebra of limits".

Theorem 1.11. *Let f, g be two functions defined on both sides of a, except possibly at a itself. Assume that $\lim_{x\to a} f(x)$ and $\lim_{x\to a} g(x)$ exist. Then*

(1) $\lim_{x\to a} (f(x) + g(x)) = \lim_{x\to a} f(x) + \lim_{x\to a} g(x)$.

(2) $\lim_{x\to a} c\, f(x) = c \lim_{x\to a} f(x)$, *for any number c.*

(3) $\lim_{x\to a} f(x)g(x) = \lim_{x\to a} f(x) \lim_{x\to a} g(x)$.

(4) $\lim_{x\to a} \dfrac{1}{g(x)} = \dfrac{1}{\lim_{x\to a} g(x)}$, *provided* $\lim_{x\to a} g(x) \neq 0$.

Remark 1.12. Polynomials, rational functions, all trigonometric functions wherever they are defined, have property called direct substitution property:

$$\lim_{x\to a} f(x) = f(a).$$ □

The following theorem is often useful to compute limits of functions.

Theorem 1.13. *If $f(x) \leq g(x)$ when x is in an interval containing a (except possibly at a) and the limits of f and g both exist as x approaches a, then*

$$\lim_{x\to a} f(x) \leq \lim_{x\to a} g(x).$$

Theorem 1.14 (Sandwich Theorem). *Let f, g, and h be given functions such that*

(1) $f(x) \leq g(x) \leq h(x)$ *when x is in an interval containing a (except possibly at a) and*

(2) $\lim_{x\to a} f(x) = \lim_{x\to a} h(x) = L$,

then

$$\lim_{x\to a} g(x) = L.$$

We will now give a rigorous definition of the limit of a function. Similar definitions can be written down for left-hand and right-hand limits of functions.

Definition 1.15. *Let f be a function defined on some open interval that contains a, except possibly at a itself. Then we say that the* **limit** *of $f(x)$ as x approaches a is L and we write*

$$\lim_{x\to a} f(x) = L.$$

if for every $\epsilon > 0$ there is a number $\delta > 0$ such that

$$|f(x) - L| < \epsilon \text{ whenever } 0 < |x - a| < \delta.$$

Definition 1.16 (Continuity).

A function f is

(1) **continuous from the right** *at a if*

$$\lim_{x\to a+} f(x) = f(a).$$

(2) **continuous from the left** *at a if*

$$\lim_{x\to a-} f(x) = f(a).$$

(3) **continuous** *at a if*

$$\lim_{x\to a} f(x) = f(a).$$

A function f is said to be continuous on an open interval if f is continuous at every number in the interval. If f is defined on a closed interval $[a, b]$, then f is said to be continuous at a if f is continuous from the right at a and similarly, f is said to be continuous at b if f is continuous from left at b.

Remark 1.17. Note that the definition for continuity of a function f at a, means the following three conditions are satisfied:

(1) The function f must be defined at a. i.e., a is in the domain of f,

(2) $\lim_{x\to a} f(x)$ exists, and

(3) $\lim_{x\to a} f(x) = f(a)$.

Equivalently, for any given $\epsilon > 0$, there exists a $\delta > 0$ such that

$$|f(x) - f(a)| < \epsilon \text{ whenever } |x - a| < \delta. \qquad \square$$

Theorem 1.18. *If f and g are continuous at a, then the functions $f + g$, $f - g$, cg (c is a constant), fg, f/g (provided $g(a) \neq 0$), $f \circ g$ (composition of f and g, whenever it makes sense) are all continuous.*

Thus polynomials, rational functions, trigonometric functions are all continuous on their respective domains.

Theorem 1.19 (Intermediate Value Theorem). *Suppose that f is continuous on the closed interval $[a, b]$ and let N be any number between $f(a)$ and $f(b)$, where $f(a) \neq f(b)$. Then there exists a point $c \in (a, b)$ such that*

$$f(c) = N.$$

1.3 Differentiation

Definition 1.20 (Derivative).

The **derivative** *of a function f at a, denoted by $f'(a)$, is*

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}, \tag{1.1}$$

if this limit exists. We say f is **differentiable** *at a. A function f is said to be* **differentiable** *on (c, d) if f is differentiable at every point in (c, d).*

Remark 1.21. The derivative of a function f at a point $x = a$ can also be given by

$$f'(a) = \lim_{h \to 0} \frac{f(a) - f(a-h)}{h}, \tag{1.2}$$

and

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a-h)}{2h}, \tag{1.3}$$

provided the limits exist. □

If we write $x = a + h$, then $h = x - a$ and $h \to 0$ if and only if $x \to a$. Thus, formula (1.1) can equivalently be written as

$$f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a}.$$

Interpretation: Take the graph of f, draw the line joining the points $(a, f(a))$, $(x, f(x))$. Take its slope and take the limit of these slopes as $x \to a$. Then the point $(x, f(x))$ tends to $(a, f(a))$. The limit is nothing but the slope of the tangent line at $(a, f(a))$ to the curve $y = f(x)$. This geometric interpretation will be very useful in describing the Newton-Raphson method in the context of solving nonlinear equations. □

Theorem 1.22. *If f is differentiable at a, then f is continuous at a.*

Proof:

$$f(x) - f(a) = \frac{f(x) - f(a)}{x - a}(x - a)$$

$$f(x) = \frac{f(x) - f(a)}{x - a}(x - a) + f(a)$$

Taking limit as $x \to a$ in the last equation yields the desired result. □

The converse of Theorem 1.22 is not true. For, the function $f(x) = |x|$ is continuous at $x = 0$ but is not differentiable there.

Theorem 1.23. *Suppose f is differentiable at a. Then there exists a function ϕ such that*

$$f(x) = f(a) + (x - a)f'(a) + (x - a)\phi(x),$$

and

$$\lim_{x \to a} \phi(x) = 0.$$

Proof: Define ϕ by

$$\phi(x) = \frac{f(x) - f(a)}{x - a} - f'(a).$$

Since f is differentiable at a, the result follows on taking limits on both sides of the last equation as $x \to a$. □

Theorem 1.24 (Rolle's Theorem). *Let f be a function that satisfies the following three hypotheses:*

(1) *f is continuous on the closed interval $[a, b]$.*

(2) *f is differentiable on the open interval (a, b).*

(3) *$f(a) = f(b)$.*

Then there is a number c in the open interval (a, b) such that $f'(c) = 0$.

Proof:

If f is a constant function *i.e.,* $f(x) = f(a)$ for every $x \in [a, b]$, clearly such a c exists. If f is not a constant, then at least one of the following holds.

Case 1: The graph of f goes above the line $y = f(a)$ *i.e.,* $f(x) > f(a)$ for some $x \in (a, b)$.
Case 2: The graph of f goes below the line $y = f(a)$ *i.e.,* $f(x) < f(a)$ for some $x \in (a, b)$.

In case (1), *i.e.,* if the graph of f goes above the line $y = f(a)$, then the global maximum cannot be at a or b. Therefore, it must lie in the open interval (a, b). Denote that point by c. That is, global maximum on $[a, b]$ is actually a local maximum, and hence $f'(c) = 0$.

In case (2), *i.e.,* if the graph of f goes below the line $y = f(a)$, then the global minimum cannot be at a or b. Therefore it must lie in the open interval (a, b). Let us call it d. That is, global minimum on $[a, b]$ is actually a local minimum, and hence $f'(d) = 0$. This completes the proof of Rolle's theorem. □

The following theorem is due to J.-L. Lagrange.

Theorem 1.25 (Mean Value Theorem). *Let f be a function that satisfies the following hypotheses:*

(1) *f is continuous on the closed interval $[a, b]$.*

(2) *f is differentiable on the open interval (a, b).*

Then there is a number c in the open interval (a, b) such that

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

or, equivalently,

$$f(b) - f(a) = f'(c)(b - a).$$

Proof: The strategy is to define a new function $\phi(x)$ satisfying the hypothesis of Rolle's theorem. The conclusion of Rolle's theorem for ϕ should yield the conclusion of Mean Value Theorem for f.

Define ϕ on $[a, b]$ by

$$\phi(x) = f(x) - f(a) - \frac{f(b) - f(a)}{b - a}(x - a).$$

We can apply Rolle's theorem to ϕ on $[a, b]$, as ϕ satisfies the hypothesis of Rolle's theorem. Rolle's theorem asserts the existence of $c \in (a, b)$ such that $\phi'(c) = 0$. This concludes the proof of Mean Value Theorem. □

1.4 Integration

In Theorem 1.25, we have discussed the mean value property for the derivative of a function. We now discuss the mean value theorems for integration.

Theorem 1.26 (Mean Value Theorem for Integrals). *If f is continuous on $[a, b]$, then there exists a number c in $[a, b]$ such that*

$$\int_a^b f(x)\,dx = f(c)(b - a).$$

Proof: Let m and M be minimum and maximum values of f in the interval $[a, b]$, respectively. Then,

$$m(b-a) \leq \int_a^b f(x)\,dx \leq M(b-a).$$

Since f is continuous, the result follows from the intermediate value theorem. □

Recall the average value of a function f on the interval $[a, b]$ is defined by

$$\frac{1}{b-a}\int_a^b f(x)\,dx.$$

Observe that the first mean value theorem for integrals asserts that the average of an integrable function f on an interval $[a, b]$ belongs to the range of the function f.

Interpretation: Let f be a function on $[a, b]$ with $f > 0$. Draw the graph of f and find the area under the graph lying between the ordinates $x = a$ and $x = b$. Also, look at a rectangle with base as the interval $[a, b]$ with height $f(c)$ and compute its area. Both values are the same. □

The Theorem 1.26 is often referred to as the **first mean value theorem for integrals**. We now state the second mean value theorem for integrals, which is a general form of Theorem 1.26

Theorem 1.27 (Second Mean Value Theorem for Integrals). *Let f and g be continuous on $[a, b]$, and let $g(x) \geq 0$ for all $x \in \mathbb{R}$. Then there exists a number $c \in [a, b]$ such that*

$$\int_a^b f(x)g(x)\,dx = f(c)\int_a^b g(x)\,dx.$$

Proof: Left as an exercise. □

1.5 Taylor's Theorem

Let f be a real-valued function defined on an interval I. We say $f \in C^n(I)$ if f is n-times continuously differentiable at every point in I. Also, we say $f \in C^\infty(I)$ if f is continuously differentiable of any order at every point in I.

The most important result used very frequently in numerical analysis, especially in error analysis of numerical methods, is the Taylor's expansion of a C^∞ function in a neighborhood of a point $a \in \mathbb{R}$. In this section, we define the Taylor's polynomial and prove an important theorem called the **Taylor's theorem**. The idea of the proof of this theorem is similar to the one used in proving the mean value theorem, where we construct a function and apply Rolle's theorem several times to it.

Definition 1.28 (Taylor's Polynomial for a Function at a Point).

Let f be n-times differentiable at a given point a. The **Taylor's polynomial** *of degree n for the function f at the point a, denoted by T_n, is defined by*

$$T_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k, \ x \in \mathbb{R}. \tag{1.4}$$

Theorem 1.29 (Taylor's Theorem). *Let f be $(n+1)$-times differentiable function on an open interval containing the points a and x. Then there exists a number ξ between a and x such that*

$$f(x) = T_n(x) + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1}, \tag{1.5}$$

where T_n is the Taylor's polynomial of degree n for f at the point a given by (1.4) *and the second term on the right hand side is called the* **remainder term**.

Proof: Let us assume $x > a$ and prove the theorem. The proof is similar if $x < a$.

Define $g(t)$ by

$$g(t) = f(t) - T_n(t) - A(t-a)^{n+1}$$

and choose A so that $g(x) = 0$, which gives

$$A = \frac{f(x) - T_n(x)}{(x-a)^{n+1}}.$$

Note that

$$g^{(k)}(a) = 0 \text{ for } k = 0, 1, \ldots n.$$

Also, observe that the function g is continuous on $[a, x]$ and differentiable in (a, x).

Apply Rolle's theorem to g on $[a, x]$ (after verifying all the hypotheses of Rolle's theorem) to get

$$a < c_1 < x \text{ satisfying } g'(c_1) = 0.$$

Again apply Rolle's theorem to g' on $[a, c_1]$ to get

$$a < c_2 < c_1 \text{ satisfying } g''(c_2) = 0.$$

In turn apply Rolle's theorem to $g^{(2)}$, $g^{(3)}$, $\ldots$, $g^{(n)}$ on intervals $[a, c_2]$, $[a, c_3]$, $\ldots$, $[a, c_n]$, respectively.

At the last step, we get

$$a < c_{n+1} < c_n \text{ satisfying } g^{(n+1)}(c_{n+1}) = 0.$$

But

$$g^{(n+1)}(c_{n+1}) = f^{(n+1)}(c_{n+1}) - A(n+1)!,$$

which gives

$$A = \frac{f^{(n+1)}(c_{n+1})}{(n+1)!}.$$

Equating both values of A, we get

$$f(x) = T_n(x) + \frac{f^{(n+1)}(c_{n+1})}{(n+1)!}(x-a)^{n+1}.$$

This completes the proof. □

Observe that the mean value theorem 1.25 is a particular case of the Taylor's theorem.

Remark 1.30. The representation (1.5) is called the **Taylor's formula** for the function f about the point a.

The Taylor's theorem helps us to obtain an approximate value of a sufficiently smooth function in a small neighborhood of a given point a when the value of f and all its derivatives up to a sufficient order is known at the point a. For instance, if we know $f(a)$, $f'(a)$, $\cdots$, $f^{(n)}(a)$, and we seek an approximate value of $f(a+h)$ for some real number h, then the Taylor's theorem can be used to get

$$f(a+h) \approx f(a) + f'(a)h + \frac{f''(a)}{2!}h + \cdots + \frac{f^{(n)}(a)}{n!}h^n.$$

Note here that we have not added the remainder term and therefore used the approximation symbol $\approx$. Observe that the remainder term

$$\frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}$$

is not known since it involves the evaluation of $f^{(n+1)}$ at some unknown value ξ lying between a and $a+h$. Also, observe that as $h \to 0$, the remainder term approaches to zero, provided $f^{(n+1)}$ is bounded. This means that for smaller values of h, the Taylor's polynomial gives a good approximation of $f(a+h)$. □

Remark 1.31 (Estimate for Remainder Term in Taylor's Formula).

Let f be an $(n+1)$-times continuously differentiable function with the property that there exists an M_{n+1} such that

$$|f^{(n+1)}(\xi)| \leq M_{n+1}, \text{ for all } \xi \in I.$$

Then for fixed points $a, x \in I$, the remainder term in (1.5) satisfies the estimate

$$\left| \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1} \right| \leq \frac{M_{n+1}}{(n+1)!}(x-a)^{n+1}.$$

We can further get an estimate of the reminder term that is independent of x as

$$\left| \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1} \right| \leq \frac{M_{n+1}}{(n+1)!}(b-a)^{n+1},$$

which holds for all $x \in I$. Observe that the right hand side of the above estimate is a fixed number. We refer to such estimates as **remainder estimates**.

In most applications of Taylor's theorem, one never knows ξ precisely. However in view of remainder estimate given above, it does not matter as long as we know that the remainder can be bounded by obtaining a bound M_{n+1} which is valid for all ξ between a and x. □

Definition 1.32 (Truncation Error).

The remainder term involved in approximating $f(x)$ by the Taylor's polynomial $T_n(x)$ is also called the **Truncation error**.

Example 1.33. A second degree polynomial approximation to

$$f(x) = \sqrt{x+1}, \quad x \in [-1, \infty)$$

using the Taylor's formula about $a = 0$ is given by

$$f(x) \approx 1 + \frac{x}{2} - \frac{x^2}{8},$$

where the remainder term is neglected and hence what we obtained here is only an approximate representation of f.

The truncation error is obtained using the remainder term in the formula (1.5) with $n = 2$ and is given by

$$\frac{x^3}{16(\sqrt{1+\xi}\,)^5},$$

for some point ξ between 0 and x.

Note that we cannot obtain a remainder estimate in the present example as f''' is not bounded in $[-1, \infty)$. However, for any $0 < \delta < 1$, if we restrict the domain of f to $[-\delta, \infty)$, then we can obtain the remainder estimate for a fixed $x \in [-\delta, \infty)$ as

$$\frac{x^3}{16(\sqrt{1-\delta}\,)^5}.$$

Further, if we restrict the domain of f to $[-\delta, b]$ for some real number $b > 0$, then we get the remainder estimate independent of x as

$$\frac{b^3}{16(\sqrt{1-\delta}\,)^5}.$$ □

Definition 1.34 (Taylor's Series). *Let f be C^∞ in a neighborhood of a point a. The power series*

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

is called the **Taylor's series** *of f about the point a.*

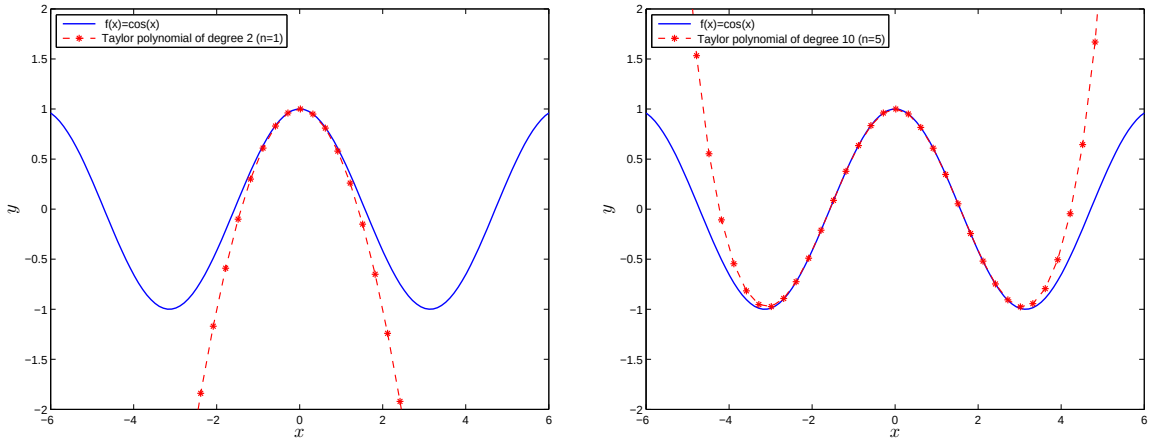

Fig. 1.1. Comparison between the graph of $f(x) = \cos(x)$ and the Taylor polynomial of degree 2 and 10 about the point $a = 0$.

The question now is when this series converges and what is the limit of this series. These questions are answered in the following theorem.

Theorem 1.35. *Let f be $C^\infty(I)$ and let $a \in I$. Assume that there exists an open interval $I_a \subset I$ of the point a such that there exists a constant M (may depend on a)*

$$\left|f^{(k)}(x)\right| \leq M^k,$$

for all $x \in N_a$ and $k = 0, 1, 2, \cdots$. Then for each $x \in I_a$, we have

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

Example 1.36. As another example, let us approximate the function $f(x) = \cos(x)$ by a polynomial using Taylor's theorem about the point $a = 0$. First, let us take the Taylor's series expansion

$$\begin{aligned} f(x) &= \cos(0) - \sin(0)x - \frac{\cos(0)}{2!}x^2 + \frac{\sin(0)}{3!}x^3 + \cdots \\ &= \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!}x^{2k}. \end{aligned}$$

Now, we truncate this infinite series to get an approximate representation of $f(x)$ in a sufficiently small neighborhood of $a = 0$ as

$$f(x) \approx \sum_{k=0}^{n} \frac{(-1)^k}{(2k)!}x^{2k},$$

which is the Taylor polynomial of degree n for the function $f(x) = \cos(x)$ about the point $a = 0$. The remainder term is given by

$$(-1)^{n+1}\frac{\cos(\xi)}{(2(n+1))!}x^{2(n+1)},$$

where ξ lies between 0 and x. It is important to observe here that for a given n, we get the Taylor polynomial of degree $2n$. Figure 1.1 shows the comparison between the Taylor polynomial (red dot and dash line) of degree 2 ($n = 1$) and degree 10 ($n = 5$) for $f(x) = \cos(x)$ about $a = 0$ and the graph of $\cos(x)$ (blue solid line). We observe that for $n = 1$, Taylor polynomial gives a good approximation in a small neighborhood of $a = 0$. But sufficiently away from 0, this polynomial deviates significantly from the actual graph of $f(x) = \cos(x)$. Whereas, for $n = 5$, we get a good approximation in a sufficiently large neighborhood of $a = 0$. □

1.6 Orders of Convergence

In Section 1.1, we defined convergent sequences and discussed some conditions under which a given sequence of real numbers converges. The definition and the discussed conditions never tell us how fast the sequence converges to the limit. Even if we know that a sequence of approximations converge to the exact one (limit), it is very important in numerical analysis to know how fast the sequence of approximate values converge to the exact value. In this section, we introduce two very important notations called **big Oh** and **little oh**, which are basic tools for the study of speed of convergence. We end this section by defining the rate of convergence, which is also called order of convergence.

1.6.1 Big Oh and Little oh Notations

The notions of big Oh and little oh are well understood through the following example.

Example 1.37. Consider the two sequences $\{n\}$ and $\{n^2\}$ both of which are unbounded and tend to infinity as $n \to \infty$. However we feel that the sequence $\{n\}$ grows 'slowly' compared to the sequence $\{n^2\}$. Consider also the sequences $\{1/n\}$ and $\{1/n^2\}$ both of which decrease to zero as $n \to \infty$. However we feel that the sequence $\{1/n^2\}$ decreases more rapidly compared to the sequence $\{1/n\}$. □

The above examples motivate us to develop tools that compare two sequences $\{a_n\}$ and $\{b_n\}$. Landau has introduced the concepts of Big Oh and Little oh for comparing two sequences that we will define below.

Definition 1.38 (Big Oh and Little oh).

Let $\{a_n\}$ and $\{b_n\}$ be sequences of real numbers. Then

(1) *the sequence $\{a_n\}$ is said to be* **Big Oh** *of $\{b_n\}$, and write $a_n = O(b_n)$, if there exists a real number C and a natural number N such that*

$$|a_n| \leq C\,|b_n| \quad \text{for all } n \geq N.$$

(2) *the sequence $\{a_n\}$ is said to be* **Little oh** *(sometimes said to be* **small oh***) of $\{b_n\}$, and write $a_n = o(b_n)$, if for every $\epsilon > 0$ there exists a natural number N such that*

$$|a_n| \leq \epsilon\,|b_n| \quad \text{for all } n \geq N.$$

Remark 1.39.

(1) If $b_n \neq 0$ for every n, then we have $a_n = O(b_n)$ if and only if the sequence $\left\{\frac{a_n}{b_n}\right\}$ is bounded. That is, there exists a constant C such that

$$\left|\frac{a_n}{b_n}\right| \leq C$$

(2) If $b_n \neq 0$ for every n, then we have $a_n = o(b_n)$ if and only if the sequence $\left\{\frac{a_n}{b_n}\right\}$ converges to 0. That is,

$$\lim_{n\to\infty} \frac{a_n}{b_n} = 0.$$

(3) For any pair of sequences $\{a_n\}$ and $\{b_n\}$ such that $a_n = o(b_n)$, it follows that $a_n = O(b_n)$. The converse is not true. Consider the sequences $a_n = n$ and $b_n = 2n + 3$, for which $a_n = O(b_n)$ holds but $a_n = o(b_n)$ does not hold.

(4) Let $\{a_n\}$ and $\{b_n\}$ be two sequences that converge to 0. Then $a_n = O(b_n)$ means the sequence $\{a_n\}$ tends to 0 as fast as the sequence $\{b_n\}$; and $a_n = o(b_n)$ means the sequence $\{a_n\}$ tends to 0 faster than the sequence $\{b_n\}$. □

The Big Oh and Little oh notations can be adapted for functions as follows.

Definition 1.40 (Big Oh and Little oh for Functions).

Let $x_0 \in \mathbb{R}$. Let f and g be functions defined in an interval containing x_0. Then

(1) *the function f is said to be* **Big Oh** *of g as $x \to x_0$, and write $f(x) = O\big(g(x)\big)$, if there exists a real number C and a real number δ such that*

$$|f(x)| \leq C\,|g(x)| \quad \textit{whenever } |x - x_0| \leq \delta.$$

(2) *the function f is said to be* **Little Oh** *(also,* **Small oh***) of g as $x \to x_0$, and write $f(x) = o\big(g(x)\big)$, if for every $\epsilon > 0$ there exists a real number C and a real number δ such that*

$$|f(x)| \leq \epsilon\,|g(x)| \quad \textit{whenever } |x - x_0| \leq \delta.$$

In case of functions also, a remark similar to the Remark 1.39 holds.

Example 1.41. The Taylor's formula for $f(x) = \cos(x)$ about the point $a = 0$ is

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

where ξ lies between x and 0.

Let us denote the remainder term (truncation error) as (for a fixed n)

$$g(x) = (-1)^{n+1} \frac{\cos(\xi)}{(2(n+1))!} x^{2(n+1)}.$$

Clearly, $g(x) \to 0$ as $x \to 0$. The question now is

'How fast does $g(x) \to 0$ as $x \to 0$?'

The answer is

'As fast as $x^{2(n+1)} \to 0$ as $x \to 0$.'

That is,

$$g(x) = O\left(x^{2(n+1)}\right) \text{ as } x \to 0. \qquad \square$$

1.6.2 Rates of Convergence

Let $\{a_n\}$ be a sequence such that $\lim_{n\to\infty} a_n = a$. We would like to measure the speed at which the convergence takes place. For example, consider

$$\lim_{n\to\infty} \frac{1}{2n+3} = 0 \text{ and } \lim_{n\to\infty} \frac{1}{n^2} = 0.$$

We feel that the first sequence goes to zero linearly and the second goes with a much superior speed because of the presence of n^2 in its denominator. We will define the notion of order of convergence precisely.

Definition 1.42 (Rate of Convergence or Order of Convergence).
Let $\{a_n\}$ be a sequence such that $\lim_{n\to\infty} a_n = a$.

(1) *We say that the rate of convergence is* **atleast linear** *if there exists a constant $c < 1$ and a natural number N such that*

$$|a_{n+1} - a| \leq c\,|a_n - a| \quad \textit{for all } n \geq N.$$

(2) *We say that the rate of convergence is* **atleast superlinear** *if there exists a sequence $\{\epsilon_n\}$ that converges to 0, and a natural number N such that*

$$|a_{n+1} - a| \leq \epsilon_n\,|a_n - a| \quad \textit{for all } n \geq N.$$

(3) *We say that the rate of convergence is* **at least quadratic** *if there exists a constant C (not necessarily less than 1), and a natural number N such that*

$$|a_{n+1} - a| \leq C\,|a_n - a|^2 \quad \textit{for all } n \geq N.$$

(4) *Let $\alpha \in \mathbb{R}_+$. We say that the rate of convergence is* **atleast** α *if there exists a constant C (not necessarily less than 1), and a natural number N such that*

$$|a_{n+1} - a| \leq C\,|a_n - a|^\alpha \quad \textit{for all } n \geq N.$$

CHAPTER 2

Error Analysis

Numerical analysis deals with developing methods, called *numerical methods*, to approximate a solution of a given Mathematical problem (whenever a solution exists). The approximate solution obtained by this method will involve an error which is precisely the difference between the exact solution and the approximate solution. Thus, we have

$$\text{Exact Solution} = \text{Approximate Solution} + \text{Error}.$$

We call this error the **mathematical error**.

The study of numerical methods is incomplete if we don't develop algorithms and implement the algorithms as computer codes. The outcome of the computer code is a set of numerical values to the approximate solution obtained using a numerical method. Such a set of numerical values is called the **numerical solution** to the given Mathematical problem. During the process of computation, the computer introduces a new error, called the **arithmetic error** and we have

$$\text{Approximate Solution} = \text{Numerical Solution} + \text{Arithmetic Error}.$$

The error involved in the numerical solution when compared to the exact solution can be worser than the mathematical error and is now given by

$$\text{Exact Solution} = \text{Numerical Solution} + \text{Mathematical Error} + \text{Arithmetic Error}.$$

The **Total Error** is defined as

$$\text{Total Error} = \text{Mathematical Error} + \text{Arithmetic Error}.$$

A digital calculating device can hold only a finite number of digits because of memory restrictions. Therefore, a number cannot be stored exactly. Certain approximation needs to be done, and only an approximate value of the given number will finally be stored in the device. For further calculations, this approximate value is used instead of the exact value of the number. This is the source of arithmetic error.

In this chapter, we introduce the floating-point representation of a real number and illustrate a few ways to obtain floating-point approximation of a given real number. We further introduce different types of errors that we come across in numerical analysis and their effects in the computation. At the end of this chapter, we will be familiar with the arithmetic errors, their effect on computed results and some ways to minimize this error in the computation.

2.1 Floating-Point Representation

Let $\beta \in \mathbb{N}$ and $\beta \geq 2$. Any real number can be represented exactly in **base** β as

$$(-1)^s \times (.d_1 d_2 \cdots d_n d_{n+1} \cdots)_\beta \times \beta^e, \tag{2.1}$$

where $d_i \in \{\, 0, 1, \cdots, \beta - 1 \,\}$ with $d_1 \neq 0$ or $d_1 = d_2 = d_3 = \cdots = 0$, $s = 0$ or 1, and an appropriate integer e called the **exponent**. Here

$$(.d_1 d_2 \cdots d_n d_{n+1} \cdots)_\beta = \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_n}{\beta^n} + \frac{d_{n+1}}{\beta^{n+1}} + \cdots \tag{2.2}$$

is a β-fraction called the **mantissa**, s is called the **sign** and the number β is called the **radix**. The representation (2.1) of a real number is called the **floating-point representation**.

Remark 2.1. When $\beta = 2$, the floating-point representation (2.1) is called the **binary floating-point representation** and when $\beta = 10$, it is called the **decimal floating-point representation**. Throughout this course, we always take $\beta = 10$. □

Due to memory restrictions, a computing device can store only a finite number of digits in the mantissa. In this section, we introduce the floating-point approximation and discuss how a given real number can be approximated.

2.1.1 Floating-Point Approximation

A computing device stores a real number with only a finite number of digits in the mantissa. Although different computing devices have different ways of representing the numbers, here we introduce a mathematical form of this representation, which we will use throughout this course.

Definition 2.2 (n-Digit Floating-point Number).
*Let $\beta \in \mathbb{N}$ and $\beta \geq 2$. An n-**digit floating-point number** in **base** β is of the form*

$$(-1)^s \times (.d_1 d_2 \cdots d_n)_\beta \times \beta^e \tag{2.3}$$

where

$$(.d_1 d_2 \cdots d_n)_\beta = \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_n}{\beta^n} \tag{2.4}$$

where $d_i \in \{\, 0, 1, \cdots, \beta - 1 \,\}$ with $d_1 \neq 0$ or $d_1 = d_2 = d_3 = \cdots = 0$, $s = 0$ or 1, and an appropriate exponent e.

Remark 2.3. When $\beta = 2$, the n-digit floating-point representation (2.3) is called the n-**digit binary floating-point representation** and when $\beta = 10$, it is called the n-**digit decimal floating-point representation**. □

Example 2.4. The following are examples of real numbers in the decimal floating point representation.

(1) The real number $x = 6.238$ is represented in the decimal floating-point representation as
$$6.238 = (-1)^0 \times 0.6238 \times 10^1,$$
in which case, we have $s = 0$, $\beta = 10$, $e = 1$, $d_1 = 6$, $d_2 = 2$, $d_3 = 3$ and $d_4 = 8$.

(2) The real number $x = -0.0014$ is represented in the decimal floating-point representation as
$$x = (-1)^1 \times 0.14 \times 10^{-2}.$$
Here $s = 1$, $\beta = 10$, $e = -2$, $d_1 = 1$ and $d_2 = 4$. □

Remark 2.5. The floating-point representation of the number $1/3$ is
$$\frac{1}{3} = 0.33333\cdots = (-1)^0 \times (0.33333\cdots)_{10} \times 10^0.$$
An n-digit decimal floating-point representation of this number has to contain only n digits in its mantissa. Therefore, the representation (2.3) is (in general) only an approximation to a real number. □

Any computing device has its own memory limitations in storing a real number. In terms of the floating-point representation, these limitations lead to the restrictions in the number of digits in the mantissa (n) and the range of the exponent (e). In section 2.1.2, we introduce the concept of under and over flow of memory, which is a result of the restriction in the exponent. The restriction on the length of the mantissa is discussed in section 2.1.3.

2.1.2 Underflow and Overflow of Memory

When the value of the exponent e in a floating-point number exceeds the maximum limit of the memory, we encounter the overflow of memory, whereas when this value goes below the minimum of the range, then we encounter underflow. Thus, for a given computing device, there are real numbers m and M such that the exponent e is limited to a range
$$m < e < M. \tag{2.5}$$
During the calculation, if some computed number has an exponent $e > M$ then we say, the memory **overflow** occurs and if $e < m$, we say the memory **underflow** occurs.

Remark 2.6. In the case of overflow of memory in a floating-point number, a computer will usually produce meaningless results or simply prints the symbol `inf` or `NaN`. When your computation involves an undetermined quantity (like $0 \times \infty$, $\infty - \infty$, $0/0$), then the output of the computed value on a computer will be the symbol `NaN` (means 'not a number'). For instance, if X is a sufficiently large number that results in an overflow of

memory when stored on a computing device, and x is another number that results in an underflow, then their product will be returned as `NaN`.

On the other hand, we feel that the underflow is more serious than overflow in a computation. Because, when underflow occurs, a computer will simply consider the number as zero without any warning. However, by writing a separate subroutine, one can monitor and get a warning whenever an underflow occurs. □

Example 2.7 (Overflow). Run the following MATLAB code on a computer with 32-bit intel processor:

```
i=308.25471;
fprintf('%f  %f\n',i,10^i);
i=308.25472;
fprintf('%f  %f\n',i,10^i);
```

We see that the first print command shows a meaningful (but very large) number, whereas the second print command simply prints `inf`. This is due to the overflow of memory while representing a very large real number.

Also try running the following code on the MATLAB:

```
i=308.25471;
fprintf('%f  %f\n',i,10^i/10^i);
i=308.25472;
fprintf('%f  %f\n',i,10^i/10^i);
```

The output will be

```
308.254710  1.000000
308.254720  NaN
```

If your computer is not showing `inf` for `i` = `308.25472`, try increasing the value of `i` till you get `inf`. □

Example 2.8 (Underflow). Run the following MATLAB code on a computer with 32-bit intel processor:

```
j=-323.6;
 if(10^j>0)
     fprintf('The given number is greater than zero\n');
 elseif (10^j==0)
     fprintf('The given number is equal to zero\n');
 else
     fprintf('The given number is less than zero\n');
 end
```

The output will be

```
The given number is greater than zero
```

When the value of j is further reduced slightly as shown in the following program

```
j=-323.64;
if(10^j>0)
    fprintf('The given number is greater than zero\n');
elseif (10^j==0)
    fprintf('The given number is equal to zero\n');
else
    fprintf('The given number is less than zero\n');
end
```

the output shows

```
The given number is equal to zero
```

If your computer is not showing the above output, try decreasing the value of j till you get the above output.

In this example, we see that the number $10^{-323.64}$ is recognized as zero by the computer. This is due to the underflow of memory. Note that multiplying any large number by this number will give zero as answer. If a computation involves such an underflow of memory, then there is a danger of having a large difference between the actual value and the computed value. □

2.1.3 Chopping and Rounding a Number

The number of digits in the mantissa, as given in Definition 2.2, is called the **precision** or **length** of the floating-point number. In general, a real number can have infinitely many digits, which a computing device cannot hold in its memory. Rather, each computing device will have its own limitation on the length of the mantissa. If a given real number has infinitely many digits in the mantissa of the floating-point form as in (2.1), then the computing device converts this number into an n-digit floating-point form as in (2.3). Such an approximation is called the **floating-point approximation** of a real number.

There are many ways to get floating-point approximation of a given real number. Here we introduce two types of floating-point approximation.

Definition 2.9 (Chopped and Rounded Numbers).
Let x be a real number given in the floating-point representation (2.1) *as*

$$x = (-1)^s \times (.d_1 d_2 \cdots d_n d_{n+1} \cdots)_\beta \times \beta^e.$$

*The floating-point approximation of x using n-***digit chopping** *is given by*

$$\text{fl}(x) = (-1)^s \times (.d_1 d_2 \cdots d_n)_\beta \times \beta^e. \tag{2.6}$$

*The floating-point approximation of x using n-**digit rounding** is given by*

$$\text{fl}(x) = \begin{cases} (-1)^s \times (.d_1 d_2 \cdots d_n)_\beta \times \beta^e & , \ 0 \le d_{n+1} < \frac{\beta}{2} \\ (-1)^s \times (.d_1 d_2 \cdots (d_n + 1))_\beta \times \beta^e & , \ \frac{\beta}{2} \le d_{n+1} < \beta \end{cases}, \tag{2.7}$$

where

$$(-1)^s \times (.d_1 d_2 \cdots (d_n+1))_\beta \times \beta^e := (-1)^s \times \left((.d_1 d_2 \cdots d_n)_\beta + (. \underbrace{0\,0 \cdots 0}_{(n-1)-\text{times}} \ 1)_\beta \right) \times \beta^e.$$

As already mentioned, throughout this course, we always take $\beta = 10$. **Also, we do not assume any restriction on the exponent** $e \in \mathbb{Z}$.

Example 2.10. The floating-point representation of π is given by

$$\pi = (-1)^0 \times (.31415926 \cdots) \times 10^1.$$

The floating-point approximation of π using **five-digit chopping** is

$$\text{fl}(\pi) = (-1)^0 \times (.31415) \times 10^1,$$

which is equal to 3.1415. Since the sixth digit of the mantissa in the floating-point representation of π is a 9, the floating-point approximation of π using **five-digit rounding** is given by

$$\text{fl}(\pi) = (-1)^0 \times (.31416) \times 10^1,$$

which is equal to 3.1416. □

Remark 2.11. Most of the modern processors, including Intel, uses IEEE 754 standard format. This format uses 52 bits in mantissa, (64-bit binary representation), 11 bits in exponent and 1 bit for sign. This representation is called the **double precision number**.

When we perform a computation without any floating-point approximation, we say that the computation is done using **infinite precision** (also called **exact arithmetic**).□

2.1.4 Arithmetic Using n-Digit Rounding and Chopping

In this subsection, we describe the procedure of performing arithmetic operations using n-digit rounding. The procedure of performing arithmetic operation using n-digit chopping is done in a similar way.

Let $\odot$ denote any one of the basic arithmetic operations '$+$', '$-$', '$\times$' and '$\div$'. Let x and y be real numbers. The process of computing $x \odot y$ **using n-digit rounding** is as follows.

Step 1: Get the n-digit rounding approximation $\text{fl}(x)$ and $\text{fl}(y)$ of the numbers x and y, respectively.

Step 2: Perform the calculation $\text{fl}(x) \odot \text{fl}(y)$ using exact arithmetic.

Step 3: Get the n-digit rounding approximation $\text{fl}(\text{fl}(x) \odot \text{fl}(y))$ of $\text{fl}(x) \odot \text{fl}(y)$.

The result from step 3 is the value of $x \odot y$ using n-digit rounding.

Example 2.12. Consider the function

$$f(x) = x\Big(\sqrt{x+1} - \sqrt{x}\Big).$$

Let us evaluate $f(100000)$ using a six-digit rounding. We have

$$f(100000) = 100000\left(\sqrt{100001} - \sqrt{100000}\right).$$

The evaluation of $\sqrt{100001}$ using six-digit rounding is as follows.

$$\begin{aligned}\sqrt{100001} &\approx 316.229347\\ &= 0.316229347 \times 10^3.\end{aligned}$$

The six-digit rounded approximation of 0.316229347×10^3 is given by 0.316229×10^3. Therefore,

$$\text{fl}(\sqrt{100001}) = 0.316229 \times 10^3.$$

Similarly,

$$\text{fl}(\sqrt{100000}) = 0.316228 \times 10^3.$$

The six-digit rounded approximation of the difference between these two numbers is

$$\text{fl}\Big(\text{fl}(\sqrt{100001}) - \text{fl}(\sqrt{100000})\Big) = 0.1 \times 10^{-2}.$$

Finally, we have

$$\begin{aligned}\text{fl}(f(100000)) &= \text{fl}(100000) \times (0.1 \times 10^{-2})\\ &= (0.1 \times 10^6) \times (0.1 \times 10^{-2})\\ &= 100.\end{aligned}$$

Using six-digit chopping, the value of $\text{fl}(f(100000))$ is 200. □

Definition 2.13 (Machine Epsilon).

The **machine epsilon** *of a computer is the smallest positive floating-point number δ such that*

$$\text{fl}(1+\delta) > 1.$$

For any floating-point number $\hat{\delta} < \delta$, we have $\text{fl}(1+\hat{\delta}) = 1$, and $1+\hat{\delta}$ and 1 are identical within the computer's arithmetic.

Remark 2.14. From Example 2.8, it is clear that the machine epsilon for a 32-bit intel processor lies between the numbers $10^{-323.64}$ and $10^{-323.6}$. It is possible to get the exact value of this number, but it is no way useful in our present course, and so we will not attempt to do this here. □

2.2 Types of Errors

The approximate representation of a real number obviously differs from the actual number, whose difference is called an **error**.

Definition 2.15 (Errors).

(1) *The* **error** *in a computed quantity is defined as*

$$\textbf{Error} = \textbf{True Value - Approximate Value}.$$

(2) *Absolute value of an error is called the* **absolute error**.

(3) *The* **relative error** *is a measure of the error in relation to the size of the true value as given by*

$$\textbf{Relative Error} = \frac{\textbf{Error}}{\textbf{True Value}}.$$

Here, we assume that the true value is non-zero.

(4) *The* **percentage error** *is defined as*

$$\textbf{Percentage Error} = 100 \times |\textbf{Relative Error}|.$$

Remark 2.16. Let x_A denote the approximation to the real number x. We use the following notations:

$$E(x_A) := \text{Error}(x_A) = x - x_A. \tag{2.8}$$

$$E_a(x_A) := \text{Absolute Error}(x_A) = |E(x_A)| \tag{2.9}$$

$$E_r(x_A) := \text{Relative Error}(x_A) = \frac{E(x_A)}{x}, \quad x \neq 0. \tag{2.10}$$

□

The absolute error has to be understood more carefully because a relatively small difference between two large numbers can appear to be large, and a relatively large difference between two small numbers can appear to be small. On the other hand, the relative error gives a percentage of the difference between two numbers, which is usually more meaningful as illustrated below.

Example 2.17. Let $x = 100000$, $x_A = 99999$, $y = 1$ and $y_A = 1/2$. We have

$$E_a(x_A) = 1, \quad E_a(y_A) = \frac{1}{2}.$$

Although $E_a(x_A) > E_a(y_A)$, we have

$$E_r(x_A) = 10^{-5}, \quad E_r(y_A) = \frac{1}{2}.$$

Hence, in terms of percentage error, x_A has only $10^{-3}\%$ error when compared to x whereas y_A has 50% error when compared to y. □

The errors defined above are between a given number and its approximate value. Quite often we also approximate a given function by another function that can be handled more easily. For instance, a sufficiently differentiable function can be approximated using Taylor's theorem 1.29. The error between the function value and the value obtained from the corresponding Taylor's polynomial is defined as **Truncation error** as defined in Definition 1.32.

2.3 Loss of Significance

In place of relative error, we often use the concept of **significant digits** that is closely related to relative error.

Definition 2.18 (Significant β-Digits).
Let β be a radix and $x \neq 0$. If x_A is an approximation to x, then we say that x_A approximates x to r **significant β-digits** *if r is the largest non-negative integer such that*

$$\frac{|x - x_A|}{|x|} \leq \frac{1}{2}\beta^{-r+1}. \tag{2.11}$$

We also say x_A **has r significant β-digits in** *x.* □

Remark 2.19. When $\beta = 10$, we refer significant 10-digits by significant digits. □

Example 2.20.

(1) For $x = 1/3$, the approximate number $x_A = 0.333$ has three significant digits, since

$$\frac{|x - x_A|}{|x|} = 0.001 < 0.005 = 0.5 \times 10^{-2}.$$

Thus, $r = 3$.

(2) For $x = 0.02138$, the approximate number $x_A = 0.02144$ has three significant digits, since

$$\frac{|x - x_A|}{|x|} \approx 0.0028 < 0.005 = 0.5 \times 10^{-2}.$$

Thus, $r = 3$.

(3) For $x = 0.02132$, the approximate number $x_A = 0.02144$ has two significant digits, since

$$\frac{|x - x_A|}{|x|} \approx 0.0056 < 0.05 = 0.5 \times 10^{-1}.$$

Thus, $r = 2$.

(4) For $x = 0.02138$, the approximate number $x_A = 0.02149$ has two significant digits, since

$$\frac{|x - x_A|}{|x|} \approx 0.0051 < 0.05 = 0.5 \times 10^{-1}.$$

Thus, $r = 2$.

(5) For $x = 0.02108$, the approximate number $x_A = 0.0211$ has three significant digits, since

$$\frac{|x - x_A|}{|x|} \approx 0.0009 < 0.005 = 0.5 \times 10^{-2}.$$

Thus, $r = 3$.

(6) For $x = 0.02108$, the approximate number $x_A = 0.02104$ has three significant digits, since

$$\frac{|x - x_A|}{|x|} \approx 0.0019 < 0.005 = 0.5 \times 10^{-2}.$$

Thus, $r = 3$. □

Remark 2.21. Number of significant digits roughly measures the number of leading non-zero digits of x_A that are correct relative to the corresponding digits in the true value x. However, this is not a precise way to get the number of significant digits as it is evident from the above examples. □

The role of significant digits in numerical calculations is very important in the sense that the loss of significant digits may result in drastic amplification of the relative error as illustrated in the following example.

Example 2.22. Let us consider two real numbers

$$x = 7.6545428 = 0.76545428 \times 10^1 \text{ and } \quad y = 7.6544201 = 0.76544201 \times 10^1.$$

The numbers

$$x_A = 7.6545421 = 0.76545421 \times 10^1 \text{ and } \quad y_A = 7.6544200 = 0.76544200 \times 10^1$$

are approximations to x and y, correct to seven and eight significant digits, respectively. The exact difference between x_A and y_A is

$$z_A = x_A - y_A = 0.12210000 \times 10^{-3}$$

and the exact difference between x and y is

$$z = x - y = 0.12270000 \times 10^{-3}.$$

Therefore,

$$\frac{|z - z_A|}{|z|} \approx 0.0049 < 0.5 \times 10^{-2}$$

and hence z_A has only three significant digits with respect to z. Thus, we started with two approximate numbers x_A and y_A which are correct to seven and eight significant digits with respect to x and y respectively, but their difference z_A has only three significant digits with respect to z. Hence, there is a loss of significant digits in the process of subtraction. A simple calculation shows that

$$E_r(z_A) \approx 53581 \times E_r(x_A).$$

Similarly, we have

$$E_r(z_A) \approx 375067 \times E_r(y_A).$$

Loss of significant digits is therefore dangerous. The loss of significant digits in the process of calculation is referred to as **Loss of Significance**. □

Example 2.23. Consider the function

$$f(x) = x(\sqrt{x+1} - \sqrt{x}).$$

From Example 2.12, the value of $f(100000)$ using six-digit rounding is 100, whereas the true value is 158.113. There is a drastic error in the value of the function, which is due to the loss of significant digits. It is evident that as x increases, the terms $\sqrt{x+1}$ and $\sqrt{x}$ comes closer to each other and therefore loss of significance in their computed value increases.

Such a loss of significance can be avoided by rewriting the given expression of f in such a way that subtraction of near-by non-negative numbers is avoided. For instance, we can re-write the expression of the function f as

$$f(x) = \frac{x}{\sqrt{x+1} + \sqrt{x}}.$$

With this new form of f, we obtain $f(100000) = 158.114000$ using six-digit rounding. □

Example 2.24. Consider evaluating the function

$$f(x) = 1 - \cos x$$

near $x = 0$. Since $\cos x \approx 1$ for x near zero, there will be loss of significance in the process of evaluating $f(x)$ for x near zero. So, we have to use an alternative formula for $f(x)$ such as

$$\begin{aligned} f(x) &= 1 - \cos x \\ &= \frac{1 - \cos^2 x}{1 + \cos x} \\ &= \frac{\sin^2 x}{1 + \cos x} \end{aligned}$$

which can be evaluated quite accurately for small x. □

Remark 2.25. Unlike the above examples, we may not be able to write an equivalent formula for the given function to avoid loss of significance in the evaluation. In such cases, we have to go for a suitable approximation of the given function by other functions, for instance Taylor's polynomial of desired degree, that do not involve loss of significance. □

2.4 Propagation of Relative Error in Arithmetic Operations

Once an error is committed, it affects subsequent results as this error propagates through subsequent calculations. We first study how the results are affected by using approximate numbers instead of actual numbers and then will take up the effect of errors on function evaluation in the next section.

Let x_A and y_A denote the approximate numbers used in the calculation, and let x_T and y_T be the corresponding true values. We will now see how relative error propagates with the four basic arithmetic operations.

2.4.1 Addition and Subtraction

Let $x_T = x_A + \epsilon$ and $y_T = y_A + \eta$ be positive real numbers. The relative error $E_r(x_A \pm y_A)$ is given by

$$\begin{aligned} E_r(x_A \pm y_A) &= \frac{(x_T \pm y_T) - (x_A \pm y_A)}{x_T \pm y_T} \\ &= \frac{(x_T \pm y_T) - (x_T - \epsilon \pm (y_T - \eta))}{x_T \pm y_T} \end{aligned}$$

Upon simplification, we get

$$E_r(x_A \pm y_A) = \frac{\epsilon \pm \eta}{x_T \pm y_T}. \tag{2.12}$$

The above expression shows that there can be a drastic increase in the relative error during subtraction of two approximate numbers whenever $x_T \approx y_T$ as we have witnessed in Examples 2.22 and 2.23. On the other hand, it is easy to see from (2.12) that

$$|E_r(x_A + y_A)| \leq |E_r(x_A)| + |E_r(y_A)|,$$

which shows that the relative error propagates slowly in addition. Note that such an inequality in the case of subtraction is not possible.

2.4.2 Multiplication

The relative error $E_r(x_A \times y_A)$ is given by

$$\begin{aligned} E_r(x_A \times y_A) &= \frac{(x_T \times y_T) - (x_A \times y_A)}{x_T \times y_T} \\ &= \frac{(x_T \times y_T) - ((x_T - \epsilon) \times (y_T - \eta))}{x_T \times y_T} \\ &= \frac{\eta x_T + \epsilon y_T - \epsilon\eta}{x_T \times y_T} \\ &= \frac{\epsilon}{x_T} + \frac{\eta}{y_T} - \left(\frac{\epsilon}{x_T}\right)\left(\frac{\eta}{y_T}\right) \end{aligned}$$

Thus, we have

$$E_r(x_A \times y_A) = E_r(x_A) + E_r(y_A) - E_r(x_A)E_r(y_A). \tag{2.13}$$

Taking modulus on both sides, we get

$$|E_r(x_A \times y_A)| \leq |E_r(x_A)| + |E_r(y_A)| + |E_r(x_A)|\,|E_r(y_A)|$$

Note that when $|E_r(x_A)|$ and $|E_r(y_A)|$ are very small, then their product is negligible when compared to $|E_r(x_A)| + |E_r(y_A)|$. Therefore, the above inequality reduces to

$$|E_r(x_A \times y_A)| \lessapprox |E_r(x_A)| + |E_r(y_A)|,$$

which shows that the relative error propagates slowly in multiplication.

2.4.3 Division

The relative error $E_r(x_A/y_A)$ is given by

$$\begin{aligned}
E_r(x_A/y_A) &= \frac{(x_T/y_T) - (x_A/y_A)}{x_T/y_T} \\
&= \frac{(x_T/y_T) - ((x_T-\epsilon)/(y_T-\eta))}{x_T/y_T} \\
&= \frac{x_T(y_T-\eta) - y_T(x_T-\epsilon)}{x_T(y_T-\eta)} \\
&= \frac{\epsilon y_T - \eta x_T}{x_T(y_T-\eta)} \\
&= \frac{y_T}{y_T-\eta}(E_r(x_A) - E_r(y_A))
\end{aligned}$$

Thus, we have

$$E_r(x_A/y_A) = \frac{1}{1-E_r(y_A)}(E_r(x_A) - E_r(y_A)). \tag{2.14}$$

The above expression shows that the relative error increases drastically during division whenever $E_r(y_A) \approx 1$. This means that y_A has 100% error when compared to y, which is very unlikely because we always expect the relative error to be very small, ie., very close to zero. In this case the right hand side is approximately equal to $E_r(x_A) - E_r(y_A)$. Hence, we have

$$|E_r(x_A/y_A)| \lessapprox |E_r(x_A) - E_r(y_A)| \leq |E_r(x_A)| + |E_r(y_A)|,$$

which shows that the relative error propagates slowly in division.

2.4.4 Total Error

In Subsection 2.1.4, we discussed the procedure of performing arithmetic operations using n-digit floating-point approximation. The computed value $\text{fl}(\text{fl}(x) \odot \text{fl}(y))$ involves an error (when compared to the exact value $x \odot y$) which comprises of

(1) Error in $\text{fl}(x)$ and $\text{fl}(y)$ due to n-digit rounding or chopping of x and y, respectively, and

(2) Error in $\text{fl}(\text{fl}(x) \odot \text{fl}(y))$ due to n-digit rounding or chopping of the number $\text{fl}(x) \odot \text{fl}(y)$.

The **total error** is defined as

$$(x \odot y) - \text{fl}(\text{fl}(x) \odot \text{fl}(y)) = [(x \odot y) - (\text{fl}(x) \odot \text{fl}(y))] + [(\text{fl}(x) \odot \text{fl}(y)) - \text{fl}(\text{fl}(x) \odot \text{fl}(y))],$$

in which the first term on the right hand side is called the **propagated error** and the second term is called the **floating-point error**. The **relative total error** is obtained by dividing both sides of the above expression by $x \odot y$.

Example 2.26. Consider evaluating the integral

$$I_n = \int_0^1 \frac{x^n}{x+5}\, dx, \quad \text{for} \quad n = 0, 1, \cdots, 30.$$

The value of I_n can be obtained in two different iterative processes, namely,

(1) The **forward iteration** for evaluating I_n is given by

$$I_n = \frac{1}{n} - 5I_{n-1}, \quad I_0 = \ln(6/5).$$

(2) The **backward iteration** for evaluating I_{n-1} is given by

$$I_{n-1} = \frac{1}{5n} - \frac{1}{5}I_n, \quad I_{30} = 0.54046330 \times 10^{-2}.$$

The following table shows the computed value of I_n using both iterative formulas along with the exact value. The numbers are computed using MATLAB using double precision arithmetic and the final answer is rounded to 6 digits after the decimal point.

n	Forward Iteration	Backward Iteration	Exact Value
1	0.088392	0.088392	0.088392
5	0.028468	0.028468	0.028468
10	0.015368	0.015368	0.015368
15	0.010522	0.010521	0.010521
20	0.004243	0.007998	0.007998
25	11.740469	0.006450	0.006450
30	-36668.803026	Not Computed	0.005405

Clearly the backward iteration gives exact value up to the number of digits shown, whereas forward iteration tends to increase error and give entirely wrong values. This is due to the propagation of error from one iteration to the next iteration. In forward iteration, the total error from one iteration is magnified by a factor of 5 at the next iteration. In backward iteration, the total error from one iteration is divided by 5 at the next iteration. Thus, in this example, with each iteration, the total error tends to increase rapidly in the forward iteration and tends to increase very slowly in the backward iteration. □

2.5 Propagation of Relative Error in Function Evaluation

For a given function $f : \mathbb{R} \to \mathbb{R}$, consider evaluating $f(x)$ at an approximate value x_A rather than at x. The question is **how well does $f(x_A)$ approximate $f(x)$?** To answer this question, we compare $E_r(f(x_A))$ with $E_r(x_A)$.

Assume that f is a C^1 function. Using the mean-value theorem, we get

$$f(x) - f(x_A) = f'(\xi)(x - x_A),$$

where ξ is an unknown point between x and x_A. The relative error in $f(x_A)$ when compared to $f(x)$ is given by

$$E_r(f(x_A)) = \frac{f'(\xi)}{f(x)}(x - x_A).$$

Thus, we have

$$E_r(f(x_A)) = \left(\frac{f'(\xi)}{f(x)}x\right) E_r(x_A). \tag{2.15}$$

Since x_A and x are assumed to be very close to each other and ξ lies between x and x_A, we may make the approximation

$$f(x) - f(x_A) \approx f'(x)(x - x_A).$$

In view of (2.15), we have

$$E_r(f(x_A)) \approx \left(\frac{f'(x)}{f(x)}x\right) E_r(x_A). \tag{2.16}$$

The expression inside the bracket on the right hand side of (2.16) is the amplification factor for the relative error in $f(x_A)$ in terms of the relative error in x_A. Thus, this expression plays an important role in understanding the propagation relative error in evaluating the function value $f(x)$ and hence motivates the following definition.

Definition 2.27 (Condition Number of a Function).
The **condition number** *of a continuously differentiable function f at a point $x = c$ is given by*

$$\left|\frac{f'(c)}{f(c)}c\right|. \tag{2.17}$$

The condition number of a function at a point $x = c$ can be used to decide whether the evaluation of the function at $x = c$ is well-conditioned or ill-conditioned depending on whether this condition number is smaller or larger as we approach this point. It is not possible to decide a priori how large the condition number should be to say that the function evaluation is ill-conditioned and it depends on the circumstances in which we are working.

Definition 2.28 (Well-Conditioned and Ill-Conditioned).
The process of evaluating a continuously differentiable function f at a point $x = c$ is said to be **well-conditioned** *if for any given interval I with $c \in I$, there exists a constant $C > 0$ (may depend on I) such that*

$$\left| \frac{f'(x)}{f(x)} x \right| \leq C,$$

for all $x \in I$. The process of evaluating a function at $x = c$ is said to be **ill-conditioned** *if it is not well-conditioned.*

Example 2.29. Consider the function $f(x) = \sqrt{x}$, for all $x \in [0, \infty)$. Then

$$f'(x) = \frac{1}{2\sqrt{x}}, \text{ for all } x \in [0, \infty).$$

The condition number of f is

$$\left| \frac{f'(x)}{f(x)} x \right| = \frac{1}{2}, \text{ for all } x \in [0, \infty)$$

which shows that taking square roots is a well-conditioned process. From (2.16), we have

$$|E_r(f(x_A))| \approx \frac{1}{2} |E_r(x_A)|.$$

Thus, $E_r(f(x_A))$ is more closer to zero than $E_r(x_A)$. □

Example 2.30. Consider the function

$$f(x) = \frac{10}{1 - x^2}, \text{ for all } x \in \mathbb{R}.$$

Then $f'(x) = 20x/(1 - x^2)^2$, so that

$$\begin{aligned} \left| \frac{f'(x)}{f(x)} x \right| &= \left| \frac{(20x/(1 - x^2)^2)x}{10/(1 - x^2)} \right| \\ &= \frac{2x^2}{|1 - x^2|} \end{aligned}$$

and this number can be quite large for $|x|$ near 1. Thus, for x near 1 or -1, the process of evaluating this function is ill-conditioned. □

The above two examples gives us a feeling that if the process of evaluating a function is well-conditioned, then we tend to get less propagating relative error. But, this is not true in general as shown in the following example.

Example 2.31. Consider the function

$$f(x) = \sqrt{x+1} - \sqrt{x}, \text{ for all } x \in (0, \infty).$$

For all $x \in (0, \infty)$, the condition number of this function is

$$\begin{aligned} \left| \frac{f'(x)}{f(x)} x \right| &= \frac{1}{2} \left| \frac{\left(\frac{1}{\sqrt{x+1}} - \frac{1}{\sqrt{x}} \right)}{\sqrt{x+1} - \sqrt{x}} x \right| \\ &= \frac{1}{2} \frac{x}{\sqrt{x+1}\sqrt{x}} \\ &\leq \frac{1}{2}, \end{aligned} \tag{2.18}$$

which shows that the process of evaluating f is well-conditioned for all $x \in (0, \infty)$. But, if we calculate $f(12345)$ using six-digit rounding, we find

$$f(12345) = \sqrt{12346} - \sqrt{12345} = 111.113 - 111.108 = 0.005,$$

while, actually, $f(12345) = 0.00450003262627751 \cdots$. The calculated answer has 10% error. □

The above example shows that a well-conditioned process of evaluating a function at a point is not enough to ensure the accuracy in the corresponding computed value. We need to check for the stability of the computation, which we discuss in the following subsection.

2.5.1 Stable and Unstable Computations

Suppose there are n steps to evaluate a function $f(x)$ at a point $x = c$. Then the total process of evaluating this function is said to have **instability** if atleast one of the n steps is ill-conditioned. If all the steps are well-conditioned, then the process is said to be **stable**.

Example 2.32. We continue the discussion in example 2.31 and check the stability in evaluating the function f. Let us analyze the computational process. The function f consists of the following four computational steps in evaluating the value of f at $x = x_0$:

$$x_1 := x_0 + 1, \quad x_2 := \sqrt{x_1}, \quad x_3 := \sqrt{x_0}, \quad x_4 := x_2 - x_3.$$

Now consider the last two steps where we already computed x_2 and now going to compute x_3 and finally evaluate the function

$$f_4(t) := x_2 - t.$$

At this step, the condition number for f_4 is given by

$$\left|\frac{f_4'(t)}{f_4(t)}t\right| = \left|\frac{t}{x_2 - t}\right|.$$

Thus, f_4 is ill-conditioned when t approaches x_2. Therefore, the above process of evaluating the function $f(x)$ is **unstable**.

Let us rewrite the same function $f(x)$ as

$$\tilde{f}(x) = \frac{1}{\sqrt{x+1} + \sqrt{x}}.$$

The computational process of evaluating $\tilde{f}$ at $x = x_0$ is

$$x_1 := x_0 + 1, \quad x_2 := \sqrt{x_1}, \quad x_3 := \sqrt{x_0}, \quad x_4 := x_2 + x_3, \quad x_5 := 1/x_4.$$

It is easy to verify that the condition number of each of the above steps is well-conditioned. For instance, the last step defines

$$\tilde{f}_5(t) = \frac{1}{x_2 + t},$$

and the condition number of this function is approximately,

$$\left|\frac{\tilde{f}_5'(x)}{\tilde{f}_5(x)}x\right| = \left|\frac{t}{x_2 + t}\right| \approx \frac{1}{2}$$

for t sufficiently close to x_2. Therefore, this process of evaluating $\tilde{f}(x)$ is stable. Recall from example 2.23 that the above expression gives a more accurate value for sufficiently large x. □

Remark 2.33. As discussed in Remark 2.25, we may not be lucky all the time to come out with an alternate expression that lead to stable evaluation for any given function when the original expression leads to unstable evaluation. In such situations, we have to compromise and go for a suitable approximation with other functions with stable evaluation process. For instance, we may try approximating the given function with its Taylor's polynomial, if possible. □

CHAPTER 3

Numerical Linear Algebra

In this chapter, we study the methods for solving system of linear equations, and computing an eigenvalue and the corresponding eigen vector for a matrix. The methods for solving linear systems is categorized into two types, namely, **direct methods** and **iterative methods**. Theoretically, direct methods give exact solution of a linear system and therefore these methods do not involve mathematical error. However, when we implement the direct methods on a computer, because of the presence of arithmetic error, the computed value from a computer will still be an approximate solution. On the other hand, an iterative method generates a sequence of approximate solutions to a given linear system which is expected to converge to the exact solution.

An important direct method is the well-known **Gaussian elimination method**. After a short introduction to linear systems in Section 3.1, we discuss the direct methods in Section 3.2. We recall the Gaussian elimination method in Subsection 3.2.1 and study the effect of arithmetic error in the computed solution. We further count the number of arithmetic operations involved in computing the solution. This operation count revels that this method is more expensive in terms of computational time. In particular, when the given system is of tri-diagonal structure, the Gaussian elimination method can be suitably modified so that the resulting method, called the **Thomas algorithm**, is more efficient in terms of computational time. After introducing the Thomal algorithm in Subsection 3.2.4, we discuss the **LU factorization** methods for a given matrix and solving a system of linear equation using LU factorization in Subsection 3.2.5.

Some matrices are sensitive to even a small error in the right hand side vector of a linear system. Such a matrix can be identified with the help of the **condition number** of the matrix. The condition number of a matrix is defined in terms of the matrix norm. In Section 3.3, we introduce the notion of matrix norms, define condition number of a matrix and discuss few important theorems that are used in the error analysis of iterative methods. We continue the chapter with the discussion of iterative methods to linear system in Section 3.4, where we introduce two basic iterative methods and discuss the sufficient condition under which the methods converge. We end this section with the definition of the residual error and another iterative method called the **residual corrector method**.

Finally in section 3.5 we discuss the power method, which is used to capture the dominant eigenvalue and a corresponding eigen vectors of a given matrix. We end the chapter with the Gerschgorin's Theorem and its application to power method.

3.1 System of Linear Equations

General form of a system of n linear equations in n variables is

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ &\cdots \\ &\cdots \\ &\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n. \end{aligned} \tag{3.1}$$

Throughout this chapter, we assume that the coefficients a_{ij} and the right hand side numbers b_i, $i, j = 1, 2, \cdots, n$ are real.

The above system of linear equations can be written in the matrix notation as

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \tag{3.2}$$

The last equation is usually written in the short form

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{3.3}$$

where A stands for the $n \times n$ matrix with entries a_{ij}, $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$ and the right hand side vector $\boldsymbol{b} = (b_1, b_2, \cdots, b_n)^T$.

Let us now state a result concerning the solvability of the system (3.2).

Theorem 3.1. *Let A be an $n \times n$ matrix and $\boldsymbol{b} \in \mathbb{R}^n$. Then the following statements concerning the system of linear equations $A\boldsymbol{x} = \boldsymbol{b}$ are equivalent.*

(1) $\det(A) \neq 0$

(2) *For each right hand side vector $\boldsymbol{b}$, the system $A\boldsymbol{x} = \boldsymbol{b}$ has a unique solution $\boldsymbol{x}$.*

(3) *For $\boldsymbol{b} = \boldsymbol{0}$, the only solution of the system $A\boldsymbol{x} = \boldsymbol{b}$ is $\boldsymbol{x} = \boldsymbol{0}$.*

We always assume that the coefficient matrix A is invertible. Any discussion of what happens when A is not invertible is outside the scope of this course.

3.2 Direct Methods for Linear Systems

In this section, we discuss two direct methods namely, the Gaussian elimination method and the LU factorization method. We also introduce the Thomas algorithm, which is a particular case of the Gaussian elimination method for tri-diagonal systems.

3.2.1 Naive Gaussian Elimination Method

Let us describe the Gaussian elimination method to solve a system of linear equations in three variables. The method for a general system is similar.

Consider the following system of three linear equations in three variables x_1, x_2, x_3:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3. \end{aligned} \tag{3.4}$$

For convenience, we call the first, second, and third equations by names E_1, E_2, and E_3 respectively.

Step 1: If $a_{11} \neq 0$, then define

$$m_{21} = \frac{a_{21}}{a_{11}}, \quad m_{31} = \frac{a_{31}}{a_{11}}. \tag{3.5}$$

We will now obtain a new system that is equivalent to the system (3.4) as follows:

- Retain the first equation E_1 as it is.
- Replace the second equation E_2 by the equation $E_2 - m_{21}E_1$.
- Replace the third equation E_3 by the equation $E_3 - m_{31}E_1$.

The new system equivalent to (3.4) is given by

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ 0 + a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 &= b_2^{(2)} \\ 0 + a_{32}^{(2)}x_2 + a_{33}^{(2)}x_3 &= b_3^{(2)}, \end{aligned} \tag{3.6}$$

where the coefficients $a_{ij}^{(2)}$, and $b_k^{(2)}$ are given by

$$\left.\begin{aligned} a_{ij}^{(2)} &= a_{ij} - m_{i1}a_{1j}, \quad i,j = 2,3 \\ b_k^{(2)} &= b_k - m_{k1}b_1, \quad k = 2,3. \end{aligned}\right\} \tag{3.7}$$

Note that the variable x_1 has been eliminated from the last two equations.

Step 2: If $a_{22}^{(2)} \neq 0$, then define

$$m_{32} = \frac{a_{32}^{(2)}}{a_{22}^{(2)}}. \tag{3.8}$$

We still use the same names E_1, E_2, E_3 for the first, second, and third equations of the modified system (3.6), respectively. We will now obtain a new system that is equivalent to the system (3.6) as follows:

- Retain the first two equations in (3.6) as they are.
- Replace the third equation by the equation $E_3 - m_{32}E_2$.

The new system is given by

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ 0 + a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 &= b_2^{(2)} \\ 0 + 0 + a_{33}^{(3)}x_3 &= b_3^{(3)}, \end{aligned} \tag{3.9}$$

where the coefficient $a_{33}^{(3)}$, and $b_3^{(3)}$ are given by

$$\begin{aligned} a_{33}^{(3)} &= a_{33}^{(2)} - m_{32}a_{23}^{(2)}, \\ b_3^{(3)} &= b_3^{(2)} - m_{32}b_2^{(2)}. \end{aligned}$$

Note that the variable x_2 has been eliminated from the last equation. This phase of the (Naive) Gaussian elimination method is called **Forward elimination phase**.

Observe that the system (3.9) is readily solvable for x_3 if the coefficient $a_{33}^{(3)} \neq 0$. Substituting the value of x_3 in the second equation of (3.9), we can solve for x_2. Substituting the values of x_1 and x_2 in the first equation, we can solve for x_1. This solution phase of the (Naive) Gaussian elimination method is called **Backward substitution phase**.

The coefficient matrix of the system (3.9) is an upper triangular matrix given by

$$U = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & 0 & a_{33}^{(3)} \end{pmatrix}. \tag{3.10}$$

Define a lower triangular matrix L by

$$L = \begin{pmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{pmatrix}. \tag{3.11}$$

It is easy to verify that $LU = A$.

Remark 3.2 (Why Naive?). We explain why the word "Naive" is used for the method described above.

(1) First of all, we do not know if the method described here can be successfully applied for all systems of linear equations which are uniquely solvable (that is, the coefficient matrix is invertible).

(2) Secondly, even when we apply the method successfully, it is not clear if the computed solution is the exact solution. In fact, it is not even clear that the computed solution is close to the exact solution.

We illustrate these points in Example 3.3 and Example 3.4. □

Example 3.3. Consider the system of equations

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}. \tag{3.12}$$

The Step 1 cannot be started as $a_{11} = 0$. Thus the naive Gaussian elimination method fails. □

Example 3.4. Let $0 < \epsilon \ll 1$. Consider the system of equations

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}. \tag{3.13}$$

Since $\epsilon \neq 0$, after Step 1 of the Naive Gaussian elimination method, we get the system

$$\begin{pmatrix} \epsilon & 1 \\ 0 & 1 - \epsilon^{-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 - \epsilon^{-1} \end{pmatrix}. \tag{3.14}$$

Thus the solution is given by

$$x_2 = \frac{2 - \epsilon^{-1}}{1 - \epsilon^{-1}}, \quad x_1 = (1 - x_2)\epsilon^{-1}. \tag{3.15}$$

Note that for a sufficiently small ϵ, the computer evaluates $2 - \epsilon^{-1}$ as $-\epsilon^{-1}$, and $1 - \epsilon^{-1}$ also as $-\epsilon^{-1}$. Thus, $x_2 \approx 1$ and as a consequence $x_1 \approx 0$. However the exact/correct solution is given by

$$x_1 = \frac{1}{1 - \epsilon} \approx 1, \quad x_2 = \frac{1 - 2\epsilon}{1 - \epsilon} \approx 1. \tag{3.16}$$

Thus, in this particular example, the solution obtained by the naive Gaussian elimination method is completely wrong.

To understand this example better, we instruct the reader to solve the system (3.13) for the cases (1) $\epsilon = 10^{-3}$, and (2) $\epsilon = 10^{-5}$ using 3-digit rounding. □

Let us see a numerical example.

Example 3.5. Consider the linear system

$$\begin{aligned} 6x_1 + 2x_2 + 2x_3 &= -2 \\ 2x_1 + \frac{2}{3}x_2 + \frac{1}{3}x_3 &= 1 \\ x_1 + 2x_2 - x_3 &= 0. \end{aligned} \tag{3.17}$$

Let us solve this system using (naive) Gaussian elimination method using 4-digit rounding. In 4-digit rounding approximation, the above system takes the form

$$\begin{aligned} 6.000x_1 + 2.000x_2 + 2.000x_3 &= -2.000 \\ 2.000x_1 + 0.6667x_2 + 0.3333x_3 &= 1.000 \\ 1.000x_1 + 2.000x_2 - 1.000x_3 &= 0.000 \end{aligned}$$

After eliminating x_1 from the second and third equations, we get (with $m_{21} = 0.3333$, $m_{31} = 0.1667$)

$$\begin{aligned} 6.000x_1 + 2.000x_2 + 2.000x_3 &= -2.000 \\ 0.000x_1 + 0.0001x_2 - 0.3333x_3 &= 1.667 \\ 0.000x_1 + 1.667x_2 - 1.333x_3 &= 0.3334 \end{aligned} \tag{3.18}$$

After eliminating x_2 from the third equation, we get (with $m_{32} = 16670$)

$$\begin{aligned} 6.000x_1 + 2.000x_2 + 2.000x_3 &= -2.000 \\ 0.000x_1 + 0.0001x_2 - 0.3333x_3 &= 1.667 \\ 0.000x_1 + 0.0000x_2 + 5555x_3 &= -27790 \end{aligned}$$

Using back substitution, we get $x_1 = 1.335$, $x_2 = 0$ and $x_3 = -5.003$, whereas the actual solution is $x_1 = 2.6$, $x_2 = -3.8$ and $x_3 = -5$. The difficulty with this elimination process is that the second equation in (3.18), where the coefficient of x_2 should have been zero, but rounding error prevented it and makes the relative error very large. □

The above examples highlight the inadequacy of the Naive Gaussian elimination method. These inadequcies can be overcome by modifying the procedure of Naive Gaussian elimination method. There are many kinds of modification. We will discuss one of the most popular modified methods which is called **Modified Gaussian elimination method with partial pivoting**.

3.2.2 Modified Gaussian Elimination Method with Partial Pivoting

Consider the following system of three linear equations in three variables x_1, x_2, x_3:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3. \end{aligned} \tag{3.19}$$

For convenience, we call the first, second, and third equations by names E_1, E_2, and E_3 respectively.

Step 1: Define $s_1 = \max\{\, |a_{11}|, |a_{21}|, |a_{31}| \,\}$. Note that $s_1 \neq 0$ (why?). Let k be the least number such that $s_1 = |a_{k1}|$. Interchange the first equation and the k^{th} equation. Let us re-write the system after this modification.

$$\begin{aligned} a^{(1)}_{11}x_1 + a^{(1)}_{12}x_2 + a^{(1)}_{13}x_3 &= b^{(1)}_1 \\ a^{(1)}_{21}x_1 + a^{(1)}_{22}x_2 + a^{(1)}_{23}x_3 &= b^{(1)}_2 \\ a^{(1)}_{31}x_1 + a^{(1)}_{32}x_2 + a^{(1)}_{33}x_3 &= b^{(1)}_3. \end{aligned} \tag{3.20}$$

where

$$a^{(1)}_{11} = a_{k1}, a^{(1)}_{12} = a_{k2}, a^{(1)}_{13} = a_{k3}, a^{(1)}_{k1} = a_{11}, a^{(1)}_{k2} = a_{12}, a^{(1)}_{k3} = a_{13}; b^{(1)}_1 = b_k, b^{(1)}_k = b_1, \tag{3.21}$$

and rest of the coefficients $a_{ij}^{(1)}$ are same as a_{ij} as all equations other than the first and k^{th} remain untouched by the interchange of first and k^{th} equation. Now eliminate the x_1 variable from the second and third equations of the system (3.20). Define

$$m_{21} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}}, \quad m_{31} = \frac{a_{31}^{(1)}}{a_{11}^{(1)}}. \tag{3.22}$$

We will now obtain a new system that is equivalent to the system (3.20) as follows:

- The first equation will be retained as it is.
- Replace the second equation by the equation $E_2 - m_{21}E_1$.
- Replace the third equation by the equation $E_3 - m_{31}E_1$.

The new system is given by

$$\begin{aligned} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 &= b_1^{(1)} \\ 0 + a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 &= b_2^{(2)} \\ 0 + a_{32}^{(2)}x_2 + a_{33}^{(2)}x_3 &= b_3^{(2)}, \end{aligned} \tag{3.23}$$

where the coefficients $a_{ij}^{(2)}$, and $b_k^{(2)}$ are given by

$$\begin{aligned} a_{ij}^{(2)} &= a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, \quad i,j = 2,3 \\ b_i^{(2)} &= b_i^{(1)} - m_{i1}b_1^{(1)}, \quad i = 2,3. \end{aligned}$$

Note that the variable x_1 has been eliminated from the last two equations.

Step 2: Define $s_2 = \max\left\{ |a_{22}^{(2)}|, |a_{32}^{(2)}| \right\}$. Note that $s_2 \neq 0$ (why?). Let l be the least number such that $s_l = |a_{l2}^{(2)}|$. Interchange the second row and the l^{th} rows. Let us re-write the system after this modification.

$$\begin{aligned} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 &= b_1^{(1)} \\ 0 + a_{22}^{(3)}x_2 + a_{23}^{(3)}x_3 &= b_2^{(3)} \\ 0 + a_{32}^{(3)}x_2 + a_{33}^{(3)}x_3 &= b_3^{(3)}, \end{aligned} \tag{3.24}$$

where the coefficients $a_{ij}^{(3)}$, and $b_i^{(3)}$ are given by

$$\begin{aligned} a_{22}^{(3)} = a_{l2}^{(2)}, a_{23}^{(3)} = a_{l3}^{(2)}, a_{l2}^{(3)} = a_{22}^{(2)}, a_{l3}^{(3)} = a_{23}^{(2)}, \\ b_2^{(3)} = b_l^{(2)}, b_l^{(3)} = b_2^{(2)} \end{aligned}$$

We still use the same names E_1, E_2, E_3 for the first, second, and third equations of the modified system (3.24), respectively.

In case $l = 2$, both second and third equations stay as they are. Let us now eliminate x_2 from the last equation. Define

$$m_{32} = \frac{a_{32}^{(3)}}{a_{22}^{(3)}} \tag{3.25}$$

We will now obtain a new system that is equivalent to the system (3.24) as follows:

- The first two equations in (3.24) will be retained as they are.
- Replace the third equation by the equation $E_3 - m_{32}E_2$.

The new system is given by

$$\begin{aligned} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 &= b_1^{(1)} \\ 0 + a_{22}^{(3)}x_2 + a_{23}^{(3)}x_3 &= b_2^{(3)} \\ 0 + 0 + a_{33}^{(4)}x_3 &= b_3^{(4)}, \end{aligned} \tag{3.26}$$

where the coefficient $a_{33}^{(4)}$, and $b_3^{(4)}$ are given by

$$\begin{aligned} a_{33}^{(4)} &= a_{33}^{(3)} - m_{32}a_{23}^{(3)}, \\ b_3^{(4)} &= b_3^{(3)} - m_{32}b_2^{(3)}. \end{aligned}$$

Note that the variable x_2 has been eliminated from the last equation. This phase of the modified Gaussian elimination method is called **Forward elimination phase with partial pivoting**.

Now the system (3.26) is readily solvable for x_3 if the coefficient $a_{33}^{(4)} \neq 0$. In fact, it is non-zero (why?). Substituting the value of x_3 in the second equation of (3.26), we can solve for x_2. Substituting the values of x_1 and x_2 in the first equation, we can solve for x_1. This solution phase of the modified Gaussian elimination method with partial pivoting is called **Backward substitution phase**.

3.2.3 Operations Count in Naive Gaussian Elimination Method

It is important to know the length of a computation and for that reason, we count the number of arithmetic operations involved in Gaussian elimination. Let us divide the count into three parts.

(1) **The elimination step:** We now count the additions/subtractions, multiplications and divisions in going from the given system to the triangular system.

Step	Additions/Subtractions	Multiplications	Divisions
1	$(n-1)^2$	$(n-1)^2$	$n-1$
2	$(n-2)^2$	$(n-2)^2$	$n-2$
.	.	.	.
.	.	.	.
.	.	.	.
$n-1$	1	1	1
Total	$\frac{n(n-1)(2n-1)}{6}$	$\frac{n(n-1)(2n-1)}{6}$	$\frac{n(n-1)}{2}$

Here we use the formula

$$\sum_{j=1}^{p} j = \frac{p(p+1)}{2}, \ \sum_{j=1}^{p} j^2 = \frac{p(p+1)(2p+1)}{6}, p \geq 1.$$

Let us explain the first row of the above table. In the first step, computation of m_{21}, $m_{31}, \cdots, m_{n1}$ involve $(n-1)$ divisions. For each i, $j = 2, 3, \cdots, n$, the computation of $a_{ij}^{(2)}$ involves a multiplication and a subtraction. In total, there are $(n-1)^2$ multiplications and $(n-1)^2$ subtractions. Note that we do not count the operations involved in computing the coefficients of x_1 in the 2^{nd} to n^{th} equations (namely, $a_{i1}^{(2)}$), as we do not compute them and simply take them as zero. Similarly, other entries in the above table can be accounted for.

(2) **Modification of the right side:** Proceeding as before, we get

$$\text{Addition/Subtraction} = (n-1) + (n-2) + \cdots + 1 = \frac{n(n-1)}{2}$$

$$\text{Multiplication/Division} = (n-1) + (n-2) + \cdots + 1 = \frac{n(n-1)}{2}$$

(3) **The back substitution:**

$$\text{Addition/Subtraction} = (n-1) + (n-2) + \cdots + 1 = \frac{n(n-1)}{2}$$

$$\text{Multiplication/Division} = n + (n-1) + \cdots + 1 = \frac{n(n+1)}{2}$$

Total number of operations: The total number of operations in obtaining x is

$$\text{Addition/Subtraction} = \frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)}{2} + \frac{n(n-1)}{2} = \frac{n(n-1)(2n+5)}{6}$$

$$\text{Multiplication/Division} = \frac{n(n^2+3n-1)}{3}$$

Even if we take only multiplication and division into consideration, we see that for large value of n, the operation count required for Gaussian elimination is about $\frac{1}{3}n^3$. This means that as n doubled, the cost of solving the linear system goes up by a factor of 8. In addition, most of the cost of Gaussian elimination is in the elimination step. For elimination, we have

$$\text{Multiplication/Division} = \frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)}{2} = \frac{n^3}{3}\left(1 - \frac{1}{n^2}\right) \approx \frac{1}{3}n^3.$$

The operation count in

$$\text{Elimination step} = O(n^3),$$

$$\text{RHS modification} = O(n^2),$$

$$\text{Backward substitution} = O(n^2),$$

as $n \to \infty$. Hence, once the elimination part is completed, it is much less expensive to solve the linear system.

3.2.4 Thomas Method for Tri-diagonal System

The Gaussian elimination method can be simplified in the case of a tri-diagonal system so as to increase the efficiency. The resulting simplified method is called the **Thomas method**.

A tri-diagonal system of linear equations is of the form

$$\begin{array}{llllllllll}
\beta_1 x_1 & +\gamma_1 x_2 & +0x_3 & +0x_4 & +0x_5 & +0x_6 + \cdots & +0x_{n-2} & +0x_{n-1} & +0x_n & = b_1 \\
\alpha_2 x_1 & +\beta_2 x_2 & +\gamma_2 x_3 & +0x_4 & +0x_5 & +0x_6 + \cdots & +0x_{n-2} & +0x_{n-1} & +0x_n & = b_2 \\
0x_1 & +\alpha_3 x_2 & +\beta_3 x_3 & +\gamma_3 x_4 & +0x_5 & +0x_6 + \cdots & +0x_{n-2} & +0x_{n-1} & +0x_n & = b_3 \\
0x_1 & +0x_2 & +\alpha_4 x_3 & +\beta_4 x_4 & +\gamma_4 x_5 & +0x_6 + \cdots & +0x_{n-2} & +0x_{n-1} & +0x_n & = b_4 \\
 & & & & & \cdots & & & & \\
 & & & & & \cdots & & & & \\
 & & & & & \cdots & & & & \\
0x_1 & +0x_2 & +0x_3 & +0x_4 & +0x_5 & +0x_6 + \cdots & +\alpha_{n-1} x_{n-2} & +\beta_{n-1} x_{n-1} & +\gamma_{n-1} x_n & = b_{n-1} \\
0x_1 & +0x_2 & +0x_3 & +0x_4 & +0x_5 & +0x_6 + \cdots & +0x_{n-2} & +\alpha_n x_{n-1} & +\beta_n x_n & = b_n
\end{array} \tag{3.27}$$

Here, we assume that all α_i, $i = 2, 3, \cdots, n$, β_i, $i = 1, 2, \cdots, n$, and γ_i, $i = 1, 2, \cdots, n-1$ are non-zero.

The first equation of the system reads

$$\beta_1 x_1 + \gamma_1 x_2 = b_1.$$

This equation can be rewritten as

$$x_1 + e_1 x_2 = f_1, \quad e_1 = \frac{\gamma_1}{\beta_1}, \quad f_1 = \frac{b_1}{\beta_1}.$$

Eliminating x_1 from the second equation of the system (3.27) by multiplying the above equation by α_2 and subtracting the resulting equation with the second equation of (3.27), we get

$$x_2 + e_2 x_3 = f_2, \quad e_2 = \frac{\gamma_2}{\beta_2 - \alpha_2 e_1}, \quad f_2 = \frac{b_2 - \alpha_2 f_1}{\beta_2 - \alpha_2 e_1}.$$

We now generalize the above procedure by assuming that the j^{th} equation is reduced to the form

$$x_j + e_j x_{j+1} = f_j,$$

where e_j and f_j are known quantity, reduce the $(j+1)^{\text{th}}$ equation in the form

$$x_{j+1} + e_{j+1}x_{j+2} = f_{j+1}, \quad e_{j+1} = \frac{\gamma_{j+1}}{\beta_{j+1} - \alpha_{j+1}e_j}, \quad f_{j+1} = \frac{b_{j+1} - \alpha_{j+1}f_j}{\beta_{j+1} - \alpha_{j+1}e_j},$$

for $j = 1, 2, \cdots, n-2$. Note that for $j = n-2$, we obtain the $j+1 = (n-1)^{\text{th}}$ equation as

$$x_{n-1} + e_{n-1}x_n = f_{n-1}, \quad e_{n-1} = \frac{\gamma_{n-1}}{\beta_{n-1} - \alpha_{n-1}e_{n-2}}, \quad f_{n-1} = \frac{b_{n-1} - \alpha_{n-1}f_{n-2}}{\beta_{n-1} - \alpha_{n-1}e_{n-2}}.$$

To obtain the reduced form of the nth equation of the system (3.27), eliminate x_{n-1} from the n^{th} equation by multiplying the above equation by α_n and subtracting the resulting equation with the nth equation of (3.27), which gives

$$(\alpha_n e_{n-1} - \beta_n)x_n = \alpha_n f_{n-1} - b_n.$$

Thus, the given tri-diagonal system (3.27) is now reduced to the system

$$\begin{aligned}
&x_1 + e_1x_2 + 0x_3 + 0x_4 + 0x_5 + 0x_6 + \cdots + 0x_{n-2} + 0x_{n-1} + 0x_n && = f_1\\
&0x_1 + x_2 + e_2x_3 + 0x_4 + 0x_5 + 0x_6 + \cdots + 0x_{n-2} + 0x_{n-1} + 0x_n && = f_2\\
&\qquad\qquad \cdots\\
&\qquad\qquad \cdots\\
&\qquad\qquad \cdots\\
&0x_1 + 0x_2 + 0x_3 + 0x_4 + 0x_5 + 0x_6 + \cdots + 0x_{n-2} + x_{n-1} + e_{n-1}x_n && = f_{n-1}\\
&0x_1 + 0x_2 + 0x_3 + 0x_4 + 0x_5 + 0x_6 + \cdots + 0x_{n-2} + 0x_{n-1} + (\alpha_n e_{n-1} - \beta_n)x_n && = \alpha_n f_{n-1} - b_n
\end{aligned}$$

which is an upper triangular matrix and hence, by back substitution we can get the solution.

Remark 3.6. If the denominator of any of the e_j's or f_j's is zero, then the Thomas method fails. This is the situation when $\beta_j - \alpha_j e_{j-1} = 0$ which is the coefficient of x_j in the reduced equation. A suitable partial pivoting as done in the modified Gaussian elimination method may sometime help us to overcome this problem. □

3.2.5 LU Factorization

In Theorem 3.1, we have stated that when a matrix is invertible, then the corresponding linear system can be solvable. Let us now ask the next question that

'Can we give examples of a class(es) of invertible matrices for which the system of linear equations (3.2) given by

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

is "easily" solvable?'

There are three types of matrices whose simple structure makes them to be solvable "easily". These matrices are as follows:

(1) **Invertible Diagonal matrices**: These matrices look like

$$\begin{pmatrix} d_1 & 0 & 0 & \cdots & 0 \\ 0 & d_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & d_n \end{pmatrix}$$

and $d_i \neq 0$ for each $i = 1, 2, \cdots, n$. In this case, the solution $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$ is given by $\boldsymbol{x} = \left(\frac{b_1}{d_1}, \frac{b_2}{d_2}, \cdots, \frac{b_n}{d_n}\right)^T$.

(2) **Invertible Lower triangular matrices**: These matrices look like

$$\begin{pmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{pmatrix}$$

and $l_{ii} \neq 0$ for each $i = 1, 2, \cdots, n$. The linear system takes the form

$$\begin{pmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \tag{3.28}$$

From the first equation, we solve for x_1 given by

$$x_1 = \frac{b_1}{l_{11}}.$$

Substituting this value of x_1 in the second equation, we get the value of x_2 as

$$x_2 = \frac{b_2 - l_{21}\dfrac{b_1}{l_{11}}}{l_{22}}.$$

Proceeding in this manner, we solve for the vector $\boldsymbol{x}$. This procedure of obtaining solution may be called the **forward substitution**.

(3) **Invertible Upper triangular matrices**: These matrices look like

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

and $u_{ii} \neq 0$ for each $i = 1, 2, \cdots, n$. The linear system takes the form

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \tag{3.29}$$

From the last equation, we solve for x_n given by

$$x_n = \frac{b_n}{u_{nn}}.$$

Substituting this value of x_n in the penultimate equation, we get the value of x_{n-1} as

$$x_{n-1} = \frac{b_{n-1} - u_{n-1,n} \dfrac{b_n}{u_{nn}}}{u_{n-1,n-1}}.$$

Proceeding in this manner, we solve for the vector $\boldsymbol{x}$. This procedure of obtaining solution may be called the **backward substitution**.

In general, an invertible matrix A need not be in one among the simple structures listed above. However, in certain situations we can always find an invertible lower triangular matrix L and an invertible upper triangular matrix U in such a way that

$$A = LU.$$

In this case, the system $A\boldsymbol{x} = \boldsymbol{b}$ becomes

$$L\big(U\boldsymbol{x}\big) = \boldsymbol{b}.$$

To solve for $\boldsymbol{x}$, we first solve the lower triangular system

$$L\boldsymbol{z} = \boldsymbol{b}$$

for the vector $\boldsymbol{z}$, which can be obtained easily using forward substitution. After obtaining $\boldsymbol{z}$, we solve the upper triangular system

$$U\boldsymbol{x} = \boldsymbol{z}$$

for the vector $\boldsymbol{x}$, which is again obtained easily using backward substitution.

Remark 3.7. In Gaussian elimination method discussed in Section 3.2.1, we have seen that a given matrix A can be reduced to an upper triangular matrix U by an elimination procedure and thereby can be solved using backward substitution. In the elimination procedure, we also obtained a lower triangular matrix L in such a way that

$$A = LU$$

as remarked in this section. □

The above discussion motivates the questions of LU factorizations of matrices and we turn our attention to these questions.

Definition 3.8 (LU factorization). *A matrix A is said to have* **LU factorization** *(or* **decomposition***) if there exists a lower triangular matrix L and an upper triangular matrix U such that*

$$A = LU.$$

Clearly if a matrix has an LU decomposition, the matrices L and U are not unique as

$$A = LU = (LD)(D^{-1}U) = \tilde{L}\tilde{U}$$

for any invertible diagonal matrix D. Note that $A = \tilde{L}\tilde{U}$ is also an LU decomposition of A as $\tilde{L}$ is a lower triangular matrix and $\tilde{U}$ is an upper triangular matrix.

There are three special LU decompositions that we will discuss now.

Doolittle's factorization

Definition 3.9 (Doolittle's factorization). *A matrix A is said to have a Doolittle's factorization if there exists a lower triangular matrix L with all diagonal elements as 1, and an upper triangular matrix U such that*

$$A = LU.$$

We now state the sufficient condition under with the Doolittle's factorization of a given matrix exists. For this, we need the notion of leading principal minors of a given matrix, which we define first and then state the required theorem.

Definition 3.10 (Principal Minors of a Matrix).

(1) *Let A be an $n \times n$ matrix. A* **sub-matrix** *of order k $(< n)$ of the matrix A is a $k \times k$ matrix obtained by removing the $n - k$ rows and $n - k$ columns from A.*

The determinant of such a sub-matrix of order k of A is called **minor** *of order k of the matrix A.*

(2) *The* **principal sub-matrix** *of order k of the matrix A is obtained by removing the last $n - k$ rows and the last $n - k$ columns from A.*

The determinant of the leading principal sub-matrix of order k of A is called a **principal minor** *of order k of the matrix A.*

(3) *A principal sub-matrix and the corresponding principal minor are called* **leading principal sub-matrix** *and* **leading principal minor** *respectively, if $k < n$.*

Example 3.11. Consider the 3×3 matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

There are two leading principal minors corresponding to the matrix A.

- The leading principal minor of order 1 of A is

$$|a_{11}| = a_{11}.$$

- The leading principal minor of order 2 of A is

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}.$$

Theorem 3.12. *Let $n \geq 2$, and A be an $n \times n$ matrix such that all of its first $n-1$ leading principal minors are non-zero. Then A has an LU-decomposition where L is a unit lower triangular matrix and U is an upper triangular matrix. That is, A has a Doolittle's factorization.*

Proof: Let $P(n)$ denote the following statement.

"$P(n)$: Every $n \times n$ matrix all of whose first $n-1$ leading principal minors are non-zero has a Doolittle's factorization."

We want to prove that the statement $P(n)$ is true for all natural numbers that are greater than or equal to 2, using mathematical induction.

Step 1: ($P(2)$ is true) Let $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ be a 2×2 matrix such that its leading principal minor is non-zero. We want to prove that A has a Doolittle's factorization $A = LU$ where

$$L = \begin{pmatrix} 1 & 0 \\ l_{21} & 1 \end{pmatrix}, \; U = \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix}$$

The equality $A = LU$ now becomes

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} u_{11} & u_{12} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} \end{pmatrix} \tag{3.30}$$

Equating the elements of the first rows of the two matrices in the equation (3.30), we get

$$a_{11} = u_{11}, \; a_{12} = u_{12}.$$

That is, the first row of U has been found. Now equating the elements of the second rows of the two matrices in the equation (3.30), we get

$$a_{21} = l_{21}u_{11}, \; a_{22} = l_{21}u_{12} + u_{22}.$$

From the last equation, in view of $a_{11} \neq 0$ (as it is the leading principal minor of A), we get

$$l_{21} = \frac{a_{21}}{u_{11}} = \frac{a_{21}}{a_{11}}, \; u_{22} = a_{22} = l_{21}u_{12} = a_{22} = \frac{a_{21}}{a_{11}}a_{12}.$$

Thus the statement $P(2)$ is true.

Step 2: Assume that $P(k)$ is true. We will prove that $P(k+1)$ is true. Let us assume that A is an $(k+1)\times(k+1)$ matrix such that its first k leading principal minors are non-zero. Let A_k denote the leading sub-matrix consisting of the first k rows and k columns. That is $(A_k)_{ij} = a_{ij}$ for all $1 \le i \le k$ and $1 \le j \le k$. Note that all the leading principal minors of the matrix A_k are non-zero as they are also the leading principal minors of the matrix A.

By induction hypothesis there exist a unit lower triangular $k \times k$ matrix L_k, and an upper triangular $k \times k$ matrix U_k such that $A_k = L_k U_k$. We will now prove that there exists a matrix L and U of the form

$$L = \left(\begin{array}{c|c} L_k & \begin{matrix}0\\ \vdots \\ 0\end{matrix} \\ \hline l_{k+1\,1}\; l_{k+1\,2}\; \cdots\; l_{k+1\,k} & 1 \end{array}\right), \quad U = \left(\begin{array}{c|c} U_k & \begin{matrix}u_{1\,k+1}\\ \\ u_{2\,k+1}\\ \vdots \\ u_{k\,k+1}\end{matrix} \\ \hline 0\,0\;\cdots\;0 & u_{k+1\,k+1} \end{array}\right).$$

such that

$$A = LU.$$

We claim that there exist real numbers $l_{k+1\,1}$, $l_{k+1\,2}$, $\cdots$ $l_{k+1\,k}$, and $u_{1\,k+1}$, $u_{2\,k+1}$, $\cdots$, $u_{k+1\,k+1}$ such that

$$LU = \left(\begin{array}{c|c} L_k & \begin{matrix}0\\ \vdots \\ 0\end{matrix} \\ \hline l_{k+1\,1}\; l_{k+1\,2}\; \cdots\; l_{k+1\,k} & 1 \end{array}\right)\left(\begin{array}{c|c} U_k & \begin{matrix}u_{1\,k+1}\\ \\ u_{2\,k+1}\\ \vdots \\ u_{k\,k+1}\end{matrix} \\ \hline 0\,0\;\cdots\;0 & u_{k+1\,k+1} \end{array}\right) = \left(\begin{array}{c|c} A_k & \begin{matrix}a_{1\,k+1}\\ \\ a_{2\,k+1}\\ \vdots \\ a_{k\,k+1}\end{matrix} \\ \hline a_{k+1\,1}\; a_{k+1\,2}\; \cdots\; a_{k+1\,k} & a_{k+1\,k+1} \end{array}\right).$$

Note that the pq^{th} entry of the matrix LU is given by

$$(LU)_{pq} = \sum_{j=1}^{k+1} l_{pj}u_{jq} = \sum_{j=1}^{\min\{p,q\}} l_{pj}u_{jq}, \tag{3.31}$$

where the last equality follows from the fact that L and U are lower and upper triangular matrices respectively. Thus, for p, q such that $1 \le p \le k$ and $1 \le q \le k$ we get

$$(LU)_{pq} = \sum_{j=1}^{\min\{p,q\}} l_{pj}u_{jq} = (L_kU_k)_{pq} = (A_k)_{pq}\,,$$

as $A_k = L_kU_k$ holds.

For each $s = 1, 2, \cdots, k+1$, on equating the elements in the $(k+1)^{\text{th}}$ column of LU to those of the matrix A, in view of (3.31), we get the equations

$$a_{s\,k+1} = \sum_{j=1}^{\min\{s,k+1\}} l_{sj}u_{j\,k+1} = \sum_{j=1}^{s} l_{sj}u_{j\,k+1} = u_{s\,k+1} + \sum_{j=1}^{s-1} l_{sj}u_{j\,k+1}. \tag{3.32}$$

Note that for $s = 1$, the equation (3.32) reduces to $a_{1\,k+1} = u_{1\,k+1}$, and for $2 \le s \le k+1$, the equation (3.32) gives a recursive formula for computing $u_{s\,k+1}$. Thus the last column of the matrix U is determined.

For each $s = 1, 2, \cdots, k+1$, on equating the elements in the $(k+1)^{\text{th}}$ row of LU to those of the matrix A, in view of (3.31), we get the equations

$$a_{k+1\,s} = \sum_{j=1}^{\min\{s,k+1\}} l_{k+1\,j}\,u_{js} = \sum_{j=1}^{s} l_{k+1\,j}\,u_{js} = l_{k+1\,s}\,u_{ss} + \sum_{j=1}^{s-1} l_{k+1\,j}\,u_{js} \tag{3.33}$$

Note that for $s = 1$, the equation (3.33) reduces to $a_{k+1\,1} = l_{k+1\,1}\,u_{11}$. Since $A_k = L_kU_k$, and A_k is invertible, it follows that $u_{ii} \neq 0$ for each $i = 1, 2, \cdots, k$. Thus we get $l_{k+1\,1} = \frac{a_{k+1\,1}}{u_{11}}$. For $2 \le s \le k$, the equation (3.33) gives a recursive formula for computing $l_{k+1\,s}$. Thus the last row of the matrix L is determined.

This shows that the statement $P(k+1)$ is true. By the principle of mathematical induction, it follows that $P(n)$ is true for all $n \ge 2$. □

Remark 3.13. The above theorem is silent if a matrix A is not invertible. It neither asserts nor rules out existence of an LU-decomposition in this case. □

Example 3.14 (Computing the Doolittle's factorization for a 3×3 matrix).

Let us illustrate the direct computation of the factors L and U of a 3×3 matrix A, whenever its leading principal minors of order 1, 2, and 3 are non-zero. Write $A = LU$ as

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \tag{3.34}$$

The right hand matrix multiplication yields

$$\begin{aligned} a_{11} = u_{11}, a_{12} = u_{12}, a_{13} = u_{13}, \\ a_{21} = l_{21}u_{11}, a_{31} = l_{31}u_{11}. \end{aligned} \tag{3.35}$$

These gives first column of L and the first row of U. Next multiply row 2 of L times columns 2 and 3 of U, to obtain

$$a_{22} = l_{21}u_{12} + u_{22}, \quad a_{23} = l_{21}u_{13} + u_{23}. \tag{3.36}$$

These can be solved for u_{22} and u_{23}. Next multiply row 3 of L to obtain

$$l_{31}u_{12} + l_{32}u_{22} = a_{32}, \quad l_{31}u_{13} + l_{32}u_{23} + u_{33} = a_{33}. \tag{3.37}$$

These equations yield values for l_{32} and u_{33}, completing the construction of L and U. In this process, we must have $u_{11} \neq 0$, $u_{22} \neq 0$ in order to solve for L, which is true because of the assumptions that all the leading principal minors of A are non-zero.

The decomposition we have found is the Doolittle's factorization of A. □

Example 3.15. Consider the matrix

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

Using (3.35), we get

$$u_{11} = 1, \ u_{12} = 1, \quad u_{13} = -1, \quad l_{21} = \frac{a_{21}}{u_{11}} = 1, l_{31} = \frac{a_{31}}{u_{11}} = -2$$

Using (3.36) and (3.37),

$$\begin{aligned} u_{22} &= a_{22} - l_{21}u_{12} = 2 - 1 \times 1 = 1 \\ u_{23} &= a_{23} - l_{21}u_{13} = -2 - 1 \times (-1) = -1 \\ l_{32} &= (a_{32} - l_{31}u_{12})/u_{22} = (1 - (-2) \times 1)/1 = 3 \\ u_{33} &= a_{33} - l_{31}u_{13} - l_{32}u_{23} = 1 - (-2) \times (-1) - 3 \times (-1) = 2 \end{aligned}$$

Thus we obtain the Doolittle's factorization of A as

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

Further, taking $\boldsymbol{b} = (1, 1, 1)^T$, we now solve the system $A\boldsymbol{x} = \boldsymbol{b}$ using LU factorization, with the matrix A given above. As discussed earlier, first we have to solve the lower triangular system

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -2 & 3 & 1 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Forward substitution yields $z_1 = 1, z_2 = 0, z_3 = 3$. Keeping the vector $\boldsymbol{z} = (1, 0, 3)^T$ as the right hand side, we now solve the upper triangular system

$$\begin{pmatrix} 1 & 1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 3 \end{pmatrix}.$$

Backward substitution yields $x_1 = 1, x_2 = 3/2, x_3 = 3/2$. □

Crout's factorization

In Doolittle's factorization, the lower triangular matrix has special property. If we ask the upper triangular matrix to have special property in the LU decomposition, it is known as **Crout's factorization**. We give the definition below.

Definition 3.16 (Crout's factorization). *A matrix A is said to have a* **Crout's factorization** *if there exists a lower triangular matrix L, and an upper triangular matrix U with all diagonal elements as 1 such that*

$$A = LU.$$

The computation of the Crout's factorization for a 3×3 matrix can be done in a similar way as done for Doolittle's factorization in Example 3.14. For invertible matrices, one can easily obtain Doolittle's factorization from Crout's factorization and vice versa. See the Example 3.23 below.

Cholesky's factorization

Before we introduce Cholesky's factorization, we recall the concept of a positive definite matrix.

Definition 3.17 (Positive Definite Matrix).

A symmetric matrix A is said to be **positive definite** *if*

$$\boldsymbol{x}^T A \boldsymbol{x} > 0,$$

for every non-zero vector $\boldsymbol{x}$.

We state below some useful results concerning positive definite matrices.

Lemma 3.18. *The following statements concerning a symmetric $n \times n$ matrix A are equivalent.*

(1) *The matrix A is positive definite.*

(2) *All the principal minors of the matrix A are positive.*

(3) *All the eigenvalues of the matrix A are positive.*

Proof: The statements (1) and (2) are equivalent by definition. The proof of equivalnence of (3) with other two statements is out of the scope of this course. □

Lemma 3.19. *Let A be a symmetric matrix, and B be any matrix. The following statements are equivalent.*

(1) *A is positive definite and B is invertible.*

(2) *BAB^T is positive definite.*

Proof: Left as an exercise. □

We now define Cholesky's factorization.

Definition 3.20 (Cholesky's factorization). *A matrix A is said to have a* **Cholesky's factorization** *if there exists a lower triangular matrix L such that*

$$A = LL^T.$$

Remark 3.21. It is clear from the definition that non-symmetric matrices do not admit Cholesky factorization.

Theorem 3.22. *If A is an $n \times n$ symmetric and positive definite matrix, then A has a unique factorization*

$$A = LL^T$$

where L is a lower triangular matrix with a positive diagonal.

Proof: Since a symmetric matrix A is positive definite if and only if all the leading minors of A are non-zero, and the matrix A itself is invertible, by Theorem 3.12, the matrix A has an LU decomposition. Since the matrix A is symmetric, we have

$$LU = A = A^T = U^T L^T.$$

Since A is invertible, both L and U are invertible. From the above equalities, we get

$$L^{-1}U^T = U\left(L^T\right)^{-1}.$$

Note that the matrix on the left hand side of the above equality is a lower triangular matrix, while the right hand side is an upper triangular matrix. If an upper triangular matrix equals a lower triangular matrix, then both the matrices must be equal to a diagonal matrix D. Thus we have $L^{-1}U^T = D$. Hence we get $U^T = LD$. Since $A = U^T L^T$, we get $A = LDL^T$. Now we can write

$$A = LDL^T = L\sqrt{D}\sqrt{D}L^T = \left(L\sqrt{D}\right)\left(L\sqrt{D}\right)^T \tag{3.38}$$

provided $\sqrt{D}$ makes sense, where $\sqrt{D}$ is defined to be a diagonal matrix whose diagonal entries are given by $\sqrt{d_{ii}}$ where d_{ii} are the diagonal entries of the diagonal matrix D. The proof of Cholesky decomposition is complete provided we prove that all the diagonal entries of the matrix D are non-negative and this follows from Lemma 3.19. □

Example 3.23. Find the Doolittle, Crout, and Cholesky factorizations of the matrix

$$A = \begin{pmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{pmatrix}.$$

(1) The Doolittle factorization is given by

$$A = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & 1 & 1 \end{pmatrix} \begin{pmatrix} 60 & 30 & 20 \\ 0 & 5 & 5 \\ 0 & 0 & \frac{1}{3} \end{pmatrix} \equiv LU.$$

(2) Let us now find Crout factorization from Doolittle factorization as follows:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & 1 & 1 \end{pmatrix} \begin{pmatrix} 60 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \equiv LD\hat{U}.$$

Setting $\hat{L} = LD$, we get the Crout factorization

$$A = \begin{pmatrix} 60 & 0 & 0 \\ 30 & 5 & 0 \\ 20 & 5 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \equiv \hat{L}\hat{U}.$$

(3) The Cholesky factorization is obtained by splitting D as $D = D^{\frac{1}{2}}D^{\frac{1}{2}}$ in the $LD\hat{U}$ factorization above:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & 1 & 1 \end{pmatrix} \begin{pmatrix} 60 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & 1 & 1 \end{pmatrix} \begin{pmatrix} \sqrt{60} & 0 & 0 \\ 0 & \sqrt{5} & 0 \\ 0 & 0 & \frac{1}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} \sqrt{60} & 0 & 0 \\ 0 & \sqrt{5} & 0 \\ 0 & 0 & \frac{1}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Combining the first two matrices into one and the last two matrices into another, we obtain the Cholesky factorization of A:

$$A = \begin{pmatrix} \sqrt{60} & 0 & 0 \\ \frac{\sqrt{60}}{2} & \sqrt{5} & 0 \\ \frac{\sqrt{60}}{3} & \sqrt{5} & \frac{1}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} \sqrt{60} & \frac{\sqrt{60}}{2} & \frac{\sqrt{60}}{3} \\ 0 & \sqrt{5} & \sqrt{5} \\ 0 & 0 & \frac{1}{\sqrt{3}} \end{pmatrix} \equiv \tilde{L}\tilde{L}^T$$

□

It can be observed that the idea of LU factorization method for solving a system of linear equations is parallel to the idea of Gaussian elimination method. However, LU factorization method has an advantage that once the factorization $A = LU$ is done, the solution of the linear system $A\boldsymbol{x} = \boldsymbol{b}$ is reduced to solving two triangular systems. Thus, if we want to solve a large set of linear systems where the coefficient matrix A is fixed but the right hand side vector $\boldsymbol{b}$ varies, we can do the factorization once for A and then use this factorization with different $\boldsymbol{b}$ vectors. This is obviously going to reduce the computation cost drastically as we have seen in the operation counting of Gaussian elimination method that the elimination part is the most expensive part of the computation.

3.3 Matrix Norms and Condition Number of a Matrix

Given any two real numbers α and β, the distance between them may be taken to be $|\alpha - \beta|$. We can also compare their magnitudes $|\alpha|$ and $|\beta|$. Vector Norm may be thought of as a generalization of the Modulus $|\cdot|$ concept to deal with vectors and in general matrices.

Definition 3.24 (Vector Norm).
A **vector norm** *on* $\mathbb{R}^n$ *is a function* $\|\cdot\| : \mathbb{R}^n \to [0, \infty)$ *having the following properties:*

(1) $\|\boldsymbol{x}\| \geq 0$ *for all* $\boldsymbol{x} \in \mathbb{R}^n$.
(2) $\|\boldsymbol{x}\| = 0$ *if and only if* $\boldsymbol{x} = \boldsymbol{0}$.
(3) $\|\alpha\boldsymbol{x}\| = |\alpha|\|\boldsymbol{x}\|$ *for all* $\boldsymbol{x} \in \mathbb{R}^n$ *and for all* $\alpha \in \mathbb{R}$.
(4) $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ *for all* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

The condition 4 in the above definition is called the **triangle inequality**.

We use the following notation for a vector $\boldsymbol{x} \in \mathbb{R}^n$ in terms of its components:

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T.$$

Example 3.25. There can be many vector norms on $\mathbb{R}^n$. We will describe three important vector norms defined on $\mathbb{R}^n$ now.

(1) **Euclidean Norm** on $\mathbb{R}^n$ is denoted by $\|\cdot\|_2$, and is defined by

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}. \tag{3.39}$$

(2) l_∞ **norm**, which is also called **maximum norm**, on $\mathbb{R}^n$ is denoted by $\|\cdot\|_\infty$, and is defined by

$$\|\boldsymbol{x}\|_\infty = \max\{\, |x_1|, |x_2|, \cdots, |x_n| \,\}. \tag{3.40}$$

(3) l_1 **norm** on $\mathbb{R}^n$ is denoted by $\|\cdot\|_1$, and is defined by

$$\|\boldsymbol{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|. \tag{3.41}$$

All the three norms defined above are indeed norms; it is easy to verify that they satisfy the defining conditions of a norm given in Definition 3.24. □

Example 3.26. Let us compute norms of some vectors now. Let $\boldsymbol{x} = (4, 4, -4, 4)^T$, $\boldsymbol{y} = (0, 5, 5, 5)^T$, $\boldsymbol{z} = (6, 0, 0, 0)^T$. Verify that $\|\boldsymbol{x}\|_1 = 16$, $\|\boldsymbol{y}\|_1 = 15$, $\|\boldsymbol{z}\|_1 = 6$; $\|\boldsymbol{x}\|_2 = 8$, $\|\boldsymbol{y}\|_2 = 8.66$, $\|\boldsymbol{z}\|_2 = 6$; $\|\boldsymbol{x}\|_\infty = 4$, $\|\boldsymbol{y}\|_\infty = 5$, $\|\boldsymbol{z}\|_\infty = 6$.

From this example we see that asking which vector is big does not make sense. But once the norm is fixed, this question makes sense as the answer depends on the norm used. In this example each vector is big compared to other two but in different norms. □

Remark 3.27. In our computations, we employ any one of the norms depending on convenience. It is a fact that "all vector norms on $\mathbb{R}^n$ are equivalent"; we will not elaborate further on this. □

We can also define matrix norms on $n \times n$ matrices, which helps us in finding distance between two matrices.

Definition 3.28 (Matrix Norm).

A **matrix norm** *on the vector space of all $n \times n$ real matrices $M_n(\mathbb{R})$ is a function $\|\cdot\| : M_n(\mathbb{R}) \to [0, \infty)$ having the following properties:*

(1) $\|A\| \geq 0$ *for all* $A \in M_n(\mathbb{R})$.

(2) $\|A\| = 0$ *if and only if* $A = 0$.

(3) $\|\alpha A\| = |\alpha|\|A\|$ *for all* $A \in M_n(\mathbb{R})$ *and for all* $\alpha \in \mathbb{R}$.

(4) $\|A + B\| \leq \|A\| + \|B\|$ *for all* $A, B \in M_n(\mathbb{R})$.

As in the case of vector norms, the condition 4 in the above definition is called the **triangle inequality**. For a matrix $A \in M_n(\mathbb{R})$, we use the notation

$$A = (a_{ij})_{1 \leq i \leq n,\ 1 \leq j \leq n}$$

where a_{ij} denotes the element in the i^{th} row and j^{th} column of A.

There can be many matrix norms on $M_n(\mathbb{R})$. We will describe some of them now.

Example 3.29. The following define norms on $M_n(\mathbb{R})$.

(1) $\|A\| = \sqrt{\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij}|^2}$.

(2) $\|A\| = \max\{\, |a_{ij}| \,:\, 1 \leq i \leq n,\, 1 \leq j \leq n \,\}$.

(3) $\|A\| = \sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij}|$.

All the three norms defined above are indeed norms; it is easy to verify that they satisfy the defining conditions of a matrix norm of Definition 3.28. □

Among matrix norms, there are special ones that satisfy very useful and important properties. They are called **Matrix norms subordinate to a vector norm**. As the name suggests, to define them we need to fix a vector norm. We will give a precise definition now.

Definition 3.30 (Matrix Norm subordinate to a vector norm).

Let $\|\cdot\|$ be a vector norm on $\mathbb{R}^n$ and let $A \in M_n(\mathbb{R})$. The matrix norm of A **subordinate** *to the vector norm $\|\cdot\|$ is defined by*

$$\|A\| := \sup\{\|A\boldsymbol{x}\| \,:\, \boldsymbol{x} \in \mathbb{R}^n,\, \|\boldsymbol{x}\| = 1\}. \tag{3.42}$$

The formula (3.42) indeed defines a matrix norm on $M_n(\mathbb{R})$. The proof of this fact is beyond the scope of our course. In this course, by matrix norm, we always mean a norm subordinate to some vector norm. An equivalent and more useful formula for the matrix norm subordinate to a vector norm is given in the following lemma.

Lemma 3.31. *For any $A \in M_n(\mathbb{R})$ and a given vector norm $\|\cdot\|$, we have*

$$\|A\| = \max_{\boldsymbol{z}\neq 0} \frac{\|A\boldsymbol{z}\|}{\|\boldsymbol{z}\|}. \tag{3.43}$$

Proof: For any $\boldsymbol{z} \neq 0$, we have $\boldsymbol{x} = \boldsymbol{z}/||\boldsymbol{z}||$ as a unit vector. Hence

$$\max_{||\boldsymbol{x}||=1} ||A\boldsymbol{x}|| = \max_{||\boldsymbol{z}||\neq 0} \left\| A\left(\frac{\boldsymbol{z}}{||\boldsymbol{z}||}\right)\right\| = \max_{||\boldsymbol{z}||\neq 0} \frac{||A\boldsymbol{z}||}{||\boldsymbol{z}||}.$$

□

The matrix norm subordinate to a vector norm has additional properties as stated in the following theorem whose proof is left as an exercise.

Theorem 3.32. *Let $\|\cdot\|$ be a matrix norm subordinate to a vector norm. Then*

(1) $\|A\boldsymbol{x}\| \leq \|A\|\|\boldsymbol{x}\|$ *for all* $\boldsymbol{x} \in \mathbb{R}^n$.

(2) $\|I\| = 1$ *where I is the identity matrix.*

(3) $\|AB\| \leq \|A\|\,\|B\|$ *for all* $A, B \in M_n(\mathbb{R})$. □

We will now state a few results concerning matrix norms subordinate to some of the vector norms described in Example 3.25. We will not discuss their proofs.

Theorem 3.33 (Matrix norm subordinate to the maximum norm).

The matrix norm subordinate to the l_∞ **norm** (*also called* **maximum norm** *given in* (3.40)) *on $\mathbb{R}^n$ is denoted by $\|A\|_\infty$ and is given by*

$$\|A\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}|. \tag{3.44}$$

The norm $\|A\|_\infty$ given by the formula (3.44) *is called the* **maximum-of-row-sums norm** *of A.* □

Theorem 3.34 (Matrix Norm Subordinate to the l_1-norm).

The matrix norm subordinate to the l_1 **norm** (*given in* (3.41)) *on $\mathbb{R}^n$ is denoted by $\|A\|_1$ and is given by*

$$\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}|. \tag{3.45}$$

The norm $\|A\|_1$ given by the formula (3.45) *is called the* **maximum-of-column-sums norm** *of A.* □

Description and computation of the matrix norm subordinate to the Euclidean vector norm on $\mathbb{R}^n$ is more subtle.

Theorem 3.35 (Matrix norm subordinate to the Euclidean norm).

The matrix norm subordinate to the l_2 **norm** (*also called* **Euclidean norm** *given in* (3.39)) *on $\mathbb{R}^n$ is denoted by $\|A\|_2$ and is given by*

$$\|A\|_2 = \sqrt{\max_{1\leq i\leq n} |\lambda_i|}, \tag{3.46}$$

where $\lambda_1, \lambda_2, \cdots, \lambda_n$ are eigenvalues of the matrix A^TA. The norm $\|A\|_2$ given by the formula (3.46) *is called the* **spectral norm** *of A.* □

Example 3.36. Let us now compute $\|A\|_\infty$ and $\|A\|_2$ for the matrix

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

(1) $\|A\|_\infty = 5$ since

$$\sum_{j=1}^{3} |a_{1j}| = |1| + |1| + |-1| = 3,$$
$$\sum_{j=1}^{3} |a_{2j}| = |1| + |2| + |-2| = 5,$$
$$\sum_{j=1}^{3} |a_{3j}| = |-2| + |1| + |1| = 4.$$

(2) $\|A\|_2 \approx 3.5934$ as the eigenvalues of A^TA are $\lambda_1 \approx 0.0616$, $\lambda_2 \approx 5.0256$ and $\lambda_3 \approx 12.9128$. Hence $\|A\|_2 \approx \sqrt{12.9128} \approx 3.5934$. □

The following theorem motivates the condition number for an invertible matrix which is similar to the condition number defined for a function in Section 2.5.1.

Theorem 3.37. *Let A be an invertible $n \times n$ matrix. Let $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ be the solutions of the systems*

$$A\boldsymbol{x} = \boldsymbol{b} \text{ and } A\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}},$$

respectively, where $\boldsymbol{b}$ and $\tilde{\boldsymbol{b}}$ are given vectors. Then

$$\frac{\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|} \leq \|A\| \, \|A^{-1}\| \, \frac{\|\boldsymbol{b} - \tilde{\boldsymbol{b}}\|}{\|\boldsymbol{b}\|} \tag{3.47}$$

for any fixed vector norm and the matrix norm subordinate to this vector norm.

Proof: Since A is invertible, we have $\boldsymbol{x} - \tilde{\boldsymbol{x}} = A^{-1}\left(\boldsymbol{b} - \tilde{\boldsymbol{b}}\right)$. Taking norms on both sides and using the fact that $\|A\boldsymbol{x}\| \leq \|A\|\|\boldsymbol{x}\|$ (see Theorem 3.32) holds for every $\boldsymbol{x} \in \mathbb{R}^n$, we get

$$\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| \leq \|A^{-1}\|\|\boldsymbol{b} - \tilde{\boldsymbol{b}}\| \tag{3.48}$$

The last inequality (3.48) estimates the error in the solution caused by error on the right hand side vector of the linear system $A\boldsymbol{x} = \boldsymbol{b}$. The inequality (3.47) is concerned with estimating the relative error in the solution in terms of the relative error in the right hand side vector $\boldsymbol{b}$.

Since $A\boldsymbol{x} = \boldsymbol{b}$, we get $\|\boldsymbol{b}\| = \|A\boldsymbol{x}\| \leq \|A\|\|\boldsymbol{x}\|$. Therefore $\|\boldsymbol{x}\| \geq \frac{\|\boldsymbol{b}\|}{\|A\|}$. Using this inequality in (3.48), we get (3.47). □

Remark 3.38.

(1) In the above theorem, it is important to note that a vector norm is fixed and the matrix norm used is subordinate to this fixed vector norm.

(2) The theorem holds no matter which vector norm is fixed as long as the matrix norm subordinate to it is used.

(3) In fact, whenever we do analysis on linear systems, we always fix a vector norm and then use matrix norm subordinate to it. □

Notice that the constant appearing on the right hand side of the inequality (3.47) (which is $\|A\|\,\|A^{-1}\|$) depends only on the matrix A (for a given vector norm). This number is called the **condition number** of the matrix A. Notice that this condition number depends very much on the vector norm being used on $\mathbb{R}^n$ and the matrix norm that is subordinate to the vector norm.

Definition 3.39 (Condition Number of a Matrix). *Let A be an $n \times n$ invertible matrix. Let a matrix norm be given that is subordinate to a vector norm. Then the* **condition number** *of the matrix A (denoted by $\kappa(A)$) is defined as*

$$\kappa(A) := \|A\|\,\|A^{-1}\|. \tag{3.49}$$

Remark 3.40. From Theorem 3.37, it is clear that if the condition number is small, then the relative error in the solution will also be small whenever the relative error in the right hand side vector is small. On the other hand, if the condition number is large, then the relative error could be very large even though the relative error in the right hand side vector is small. We illustrate this in the following example. □

Example 3.41. The linear system

$$\begin{aligned} 5x_1 + 7x_2 &= 0.7 \\ 7x_1 + 10x_2 &= 1 \end{aligned}$$

has the solution $x_1 = 0, x_2 = 0.1$. Let us denote this by $\boldsymbol{x} = (0, 0.1)^T$, and the right hand side vector by $\boldsymbol{b} = (0.7, 1)^T$. The perturbed system

$$5x_1 + 7x_2 = 0.69$$
$$7x_1 + 10x_2 = 1.01$$

has the solution $x_1 = -0.17, x_2 = 0.22$, which we denote by $\tilde{\boldsymbol{x}} = (-0.17, 0.22)^T$, and the right hand side vector by $\tilde{\boldsymbol{b}} = (0.69, 1.01)^T$. The relative error between the solutions of the above systems in the l_∞ vector norm is given by

$$\frac{\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_\infty}{\|\boldsymbol{x}\|_\infty} = 1.7,$$

which is too high compared to the relative error in the right hand side vector which is given by

$$\frac{\|\boldsymbol{b} - \tilde{\boldsymbol{b}}\|_\infty}{\|\boldsymbol{b}\|_\infty} = 0.01.$$

The condition number of the coefficient matrix of the system is 289. Therefore the magnification of the relative error is expected (see the inequality (3.47)). □

Definition 3.42. *A matrix with a large condition number is said to be* **ill conditioned**. *A matrix with a small condition number is said to be* **well conditioned**.

Remark 3.43. An immediate question is that how large should the condition number be to declare that a matrix is ill-conditioned. This quantification is very difficult in practical situations as it depends on how large the relative error in the right hand side vector and also the tolerance level of the user. That is, how much error a user can tolerate in the application for which the linear system is solved. For instance, in finance related applications, even 20% of error may be tolerable, whereas in computing the path of a missile even a 0.2% error may lead to fatal disasters. □

Discussion of condition numbers of matrices is incomplete without the mention of the famous **Hilbert matrix**.

Example 3.44. The **Hilbert matrix** of order n is given by

$$H_n = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \cdot & & & \cdots & \\ \cdot & & & \cdots & \\ \cdot & & & \cdots & \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+1} & \cdots & \frac{1}{2n-1} \end{pmatrix} \tag{3.50}$$

For $n = 4$, we have

$$\kappa(H_4) = \|H_4\|_\infty \|H_4^{-1}\|_\infty = \frac{25}{12} 13620 \approx 28000$$

which may be taken as an ill-conditioned matrix. In fact, as the value of n increases, the corresponding condition number of the Hilbert matrix also increases. □

An interesting and important question is that what kind of matrices could have large condition numbers. A partial answer is stated in the following theorem.

Theorem 3.45. *Let $A \in M_n(\mathbb{R})$ be non-singular. Then, for any singular $n \times n$ matrix B, we have*

$$\frac{1}{\kappa(A)} \leq \frac{\|A - B\|}{\|A\|}. \tag{3.51}$$

Proof. We have

$$\frac{1}{\kappa(A)} = \frac{1}{\|A\|\|A^{-1}\|} = \frac{1}{\|A\|}\left(\frac{1}{\max\limits_{\boldsymbol{x}\neq 0}\frac{\|A^{-1}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}}\right) \leq \frac{1}{\|A\|}\left(\frac{1}{\frac{\|A^{-1}\boldsymbol{y}\|}{\|\boldsymbol{y}\|}}\right)$$

where $\boldsymbol{y}$ is arbitrary. Take $\boldsymbol{y} = A\boldsymbol{z}$, for some arbitrary vector $\boldsymbol{z}$. Then we get

$$\frac{1}{\kappa(A)} \leq \frac{1}{\|A\|}\left(\frac{\|A\boldsymbol{z}\|}{\|\boldsymbol{z}\|}\right).$$

Let $\boldsymbol{z} \neq \boldsymbol{0}$ be such that $B\boldsymbol{z} = 0$ (this is possible since B is singular), we get

$$\begin{aligned}\frac{1}{\kappa(A)} &\leq \frac{\|(A-B)\boldsymbol{z}\|}{\|A\|\|\boldsymbol{z}\|} \\ &\leq \frac{\|(A-B)\|\|\boldsymbol{z}|}{\|A\|\|\boldsymbol{z}\|} \\ &= \frac{\|A-B\|}{\|A\|},\end{aligned}$$

and we are done. □

From the above theorem it is apparent that if A is close to a singular matrix, then the reciprocal of the condition number will be near to zero, ie., $\kappa(A)$ itself will be large. Let us illustrate this in the following example.

Example 3.46. Clearly the matrix

$$B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

is not invertible. For any $\epsilon > 0$, let

$$A = \begin{pmatrix} 1 & 1+\epsilon \\ 1-\epsilon & 1 \end{pmatrix}.$$

As $\epsilon > 0$, A is invertible and we have

$$A^{-1} = \epsilon^{-2}\begin{pmatrix} 1 & -1-\epsilon \\ -1+\epsilon & 1 \end{pmatrix}.$$

Let us use the l_∞ norm on $\mathbb{R}^n$. Then $\|A\|_\infty = 2+\epsilon$ and $\|A^{-1}\|_\infty = \epsilon^{-2}(2+\epsilon)$. Hence

$$\begin{aligned}\kappa(A) &= \|A\|_\infty \|A^{-1}\|_\infty \\ &= \left(\frac{2+\epsilon}{\epsilon}\right)^2 \\ &> \frac{4}{\epsilon^2}.\end{aligned}$$

Thus, if $\epsilon \leq 0.01$, then $\kappa(A) \geq 40,000$. As $\epsilon \to 0$, the matrix A tends to approach the matrix B and consequently, the above inequality says that the condition number $\kappa(A)$ tends to ∞.

Also, we can see that when we attempt to solve the system $A\boldsymbol{x} = \boldsymbol{b}$, then the above inequality implies that a small relative perturbation in the right hand side vector $\boldsymbol{b}$ could be magnified by a factor of at least $40,000$ for the relative error in the solution. □

3.4 Iterative Methods for Linear Systems

In Sections 3.2.1 and 3.2.5 we have discussed methods that obtain exact solution of a linear system $A\boldsymbol{x} = \boldsymbol{b}$ in the absence of floating point errors (*i.e.*, exact arithmetic is used). Such methods are called the **direct methods**. The solution of a linear system can also be obtained using iterative procedures. Such methods are called **iterative methods**. There are many iterative procedures out of which **Jacobi** and **Gauss-Seidel** methods are the simplest ones. In this section we introduce these methods and discuss their convergence.

3.4.1 Jacobi Method

In section 3.2.5, we have seen that when a linear system $A\boldsymbol{x} = \boldsymbol{b}$ is such that the coefficient matrix A is a diagonal matrix, then this system can be solved very easily. We explore this idea to build a new method based on iterative procedure. For this, we first rewrite the matrix A as

$$A = D - C,$$

where $D, C \in M_n(\mathbb{R})$ are such that

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix},$$

where a_{ii}, $i = 1, 2, \cdots, n$ are the diagonal elements of the matrix A. Then the given system of linear equations can be re-written as

$$D\boldsymbol{x} = C\boldsymbol{x} + \boldsymbol{b}. \tag{3.52}$$

If we assume that the right hand side vector is fully known to us, then the above system can be solved very easily as D is a diagonal matrix. But obviously, the right hand side vector cannot be a known quantity because it involves the unknown vector $\boldsymbol{x}$. Rather, if we choose (arbitrarily) some specific value for $\boldsymbol{x}$, say $\boldsymbol{x} = \boldsymbol{x}^{(0)}$, then the resulting system

$$D\boldsymbol{x} = C\boldsymbol{x}^{(0)} + \boldsymbol{b}$$

can be readily solved. Let us call the solution of this system as $\boldsymbol{x}^{(1)}$. That is,

$$D\boldsymbol{x}^{(1)} = C\boldsymbol{x}^{(0)} + \boldsymbol{b}.$$

Now taking $\boldsymbol{x} = \boldsymbol{x}^{(1)}$ on the right hand side of (3.52) we can obtain the solution of this system, which we denote as $\boldsymbol{x}^{(2)}$ and repeat this procedure to get a general iterative procedure as

$$D\boldsymbol{x}^{(k+1)} = C\boldsymbol{x}^{(k)} + \boldsymbol{b}, \quad k = 0, 1, 2, \cdots .$$

If D is invertible, then the above iterative procedure can be written as

$$\boldsymbol{x}^{(k+1)} = B\boldsymbol{x}^{(k)} + \boldsymbol{c}, \quad k = 0, 1, 2, \cdots , \tag{3.53}$$

where $B = D^{-1}C$ and $\boldsymbol{c} = D^{-1}\boldsymbol{b}$. The iterative procedure (3.53) is called the **Jacobi method**.

Example 3.47. Let us illustrate the Jacobi method in the case of 3×3 system

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3, \end{aligned}$$

where $a_{11} \neq 0$, $a_{22} \neq 0$, and $a_{33} \neq 0$. We can rewrite the above system of linear equations as

$$\begin{aligned} x_1 &= \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3) \\ x_2 &= \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3) \\ x_3 &= \frac{1}{a_{33}}(b_3 - a_{31}x_1 - a_{32}x_2) \end{aligned}$$

Let $\boldsymbol{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})^T$ be an initial guess to the true solution $\boldsymbol{x}$, which is chosen arbitrarily. Define a sequence of iterates (for $k = 0, 1, 2, \cdots$) by

$$\left.\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)}) \\ x_2^{(k+1)} &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)}) \\ x_3^{(k+1)} &= \frac{1}{a_{33}}(b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)}). \end{aligned}\right\} \tag{JS}$$

which is the **Jacobi iterative sequence** given by (3.53) in the case of 3×3 system. □

Now, the question is that will the sequence of vectors $\{\boldsymbol{x}^{(k+1)}\}$ generated by the iterative procedure (3.53) always converge to the exact solution $\boldsymbol{x}$ of the given linear system?

The following example gives a system for which the Jacobi iterative sequence converges to the exact solution.

Example 3.48. The Jacobi iterative sequence for the system

$$\begin{aligned} 6x_1 + x_2 + 2x_3 &= -2, \\ x_1 + 4x_2 + 0.5x_3 &= 1, \\ -x_1 + 0.5x_2 - 4x_3 &= 0. \end{aligned}$$

is given by

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{6}(-2 - x_2^{(k)} - 2x_3^{(k)}), \\ x_2^{(k+1)} &= \frac{1}{4}(1 - x_1^{(k)} - 0.5x_3^{(k)}), \\ x_3^{(k+1)} &= \frac{-1}{4}(0 + x_1^{(k)} - 0.5x_2^{(k)}). \end{aligned}$$

The exact solution (upto 6-digit rounding) of this system is

$$\boldsymbol{x} \approx (-0.441176, 0.341176, 0.152941). \tag{3.54}$$

Choosing the initial guess $\boldsymbol{x}^{(0)} = (0, 0, 0)$, we get

$$\begin{aligned} \boldsymbol{x}^{(1)} &\approx (-0.333333, 0.250000, 0.000000), \\ \boldsymbol{x}^{(2)} &\approx (-0.375000, 0.333333, 0.114583), \\ \boldsymbol{x}^{(3)} &\approx (-0.427083, 0.329427, 0.135417), \\ \boldsymbol{x}^{(4)} &\approx (-0.433377, 0.339844, 0.147949), \\ \boldsymbol{x}^{(5)} &\approx (-0.439290, 0.339851, 0.150825), \end{aligned}$$

and so on. We observe from the above computed results that the sequence $\{\boldsymbol{x}^{(k)}\}$ seems to be approaching the exact solution. □

In the following example we discuss a system for which the Jacobi iterative sequence does not converge to the exact solution.

Example 3.49. Consider the system

$$\begin{aligned} x_1 + 4x_2 + 0.5x_3 &= 1, \\ 6x_1 + x_2 + 2x_3 &= -2, \\ -x_1 + 0.5x_2 - 4x_3 &= 0. \end{aligned}$$

which is exactly the same as the system discussed in Example 3.48 but the only difference is the interchange of first and second equation. Hence, the exact solution is same as given in (3.54). The Jacobi iterative sequence for this system is given by

$$
\begin{aligned}
x_1^{(k+1)} &= (1 - 4x_2^{(k)} - 0.5x_3^{(k)}),\\
x_2^{(k+1)} &= (-2 - 6x_1^{(k)} - 2x_3^{(k)}),\\
x_3^{(k+1)} &= \frac{-1}{4}(0 + x_1^{(k)} - 0.5x_2^{(k)}).
\end{aligned}
$$

Choosing the initial guess $\boldsymbol{x}^{(0)} = (0, 0, 0)$, we get

$$
\begin{aligned}
\boldsymbol{x}^{(1)} &\approx (1, -2, 0),\\
\boldsymbol{x}^{(2)} &\approx (9, -8, -0.5),\\
\boldsymbol{x}^{(3)} &\approx (33.25, -55, -3.25),\\
\boldsymbol{x}^{(4)} &\approx (222.625, -195, -15.1875),\\
\boldsymbol{x}^{(5)} &\approx (788.59375, -1307.375, -80.03125),
\end{aligned}
$$

and so on. Here, we observe a diverging trend in the sequence $\{\boldsymbol{x}^{(k)}\}$. □

Thus, we need to look for a condition on the system for which the Jacobi iterative sequence converges to the exact solution. Define the error in the k^{th} iterate $\boldsymbol{x}^{(k)}$ compared to the exact solution by

$$\boldsymbol{e}^{(k)} = \boldsymbol{x} - \boldsymbol{x}^{(k)}.$$

It follows easily that $\boldsymbol{e}^{(k)}$ satisfies the system

$$\boldsymbol{e}^{(k+1)} = B\boldsymbol{e}^{(k)},$$

where B is as defined in (3.53). Using any vector norm and the matrix norm subordinate to it in the above equation, we get

$$\|\boldsymbol{e}^{(k+1)}\| = \|B\boldsymbol{e}^{(k)}\| \leq \|B\|\|\boldsymbol{e}^{(k)}\| \leq \cdots \leq \|B\|^{k+1}\|\boldsymbol{e}^{(0)}\|.$$

Thus, when $\|B\| < 1$, the iteration method (3.53) always converges for any initial guess $\boldsymbol{x}^{(0)}$.

Again the question is

''what are all the matrices A for which the corresponding matrices B in (3.53) have the property $\|B\| < 1$, for some matrix norm subordinate to some vector norm?"

One such class of matrices are the **diagonally dominant** matrices, which we define now.

Definition 3.50 (Diagonally Dominant Matrices). *A matrix A is said to be* **diagonally dominant** *if it satisfies the inequality*

$$\sum_{j=1, j\neq i}^{n} |a_{ij}| < |a_{ii}|, \quad i = 1, 2, \cdots, n.$$

We now prove the sufficient condition for the convergence of the Jacobi method. This theorem asserts that if A is a diagonally dominant matrix, then B in (3.53) of the Jacobi method is such that $\|B\|_\infty < 1$.

Theorem 3.51. *If the coefficient matrix A is diagonally dominant, then the Jacobi method* (3.53) *converges to the exact solution of $A\boldsymbol{x} = \boldsymbol{b}$.*

Proof: Since A is diagonally dominant, the diagonal entries are all non-zero and hence the Jacobi iterating sequence $\boldsymbol{x}^{(k)}$ given by

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \cdots, n. \tag{3.55}$$

is well-defined. Each component of the error satisfies

$$e_i^{(k+1)} = -\sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{a_{ij}}{a_{ii}} e_j^{(k)}, \quad i = 1, 2, \cdots, n. \tag{3.56}$$

which gives

$$|e_i^{(k+1)}| \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right| \|\mathbf{e}^{(k)}\|_\infty. \tag{3.57}$$

Define

$$\mu = \max_{1\leq i\leq n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right|. \tag{3.58}$$

Then

$$|e_i^{(k+1)}| \leq \mu \|\boldsymbol{e}^{(k)}\|_\infty, \tag{3.59}$$

which is true for all $i = 1, 2, \cdots, n$. Therefore, we have

$$\|\boldsymbol{e}^{(k+1)}\|_\infty \leq \mu \|\boldsymbol{e}^{(k)}\|_\infty. \tag{3.60}$$

The matrix A is diagonally dominant if and only if $\mu < 1$. Then iterating the last inequality we get

$$\|\boldsymbol{e}^{(k+1)}\|_\infty \leq \mu^{k+1} \|\boldsymbol{e}^{(0)}\|_\infty. \tag{3.61}$$

Therefore, if A is diagonally dominant, the Jacobi method converges. □

Remark 3.52. Observe that the system given in Example 3.48 is diagonally dominant, whereas the system in Example 3.49 is not so. □

3.4.2 Gauss-Seidel Method

Gauss-Seidel method is a modified version of the Jacobi method discussed in Section 3.4.1. We demonstrate the method in the case of a 3×3 system and the method for a general $n \times n$ system can be obtained in a similar way.

Example 3.53. Consider the 3×3 system

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3. \end{aligned}$$

When the diagonal elements of this system are non-zero, we can rewrite the above system as

$$\begin{aligned} x_1 &= \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3), \\ x_2 &= \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3), \\ x_3 &= \frac{1}{a_{33}}(b_3 - a_{31}x_1 - a_{32}x_2). \end{aligned}$$

Let

$$\boldsymbol{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})^T$$

be an initial guess to the true solution $\boldsymbol{x}$. Define a sequence of iterates (for $k = 0, 1, 2, \cdots$) by

$$\left.\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)}), \\ x_2^{(k+1)} &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)}), \\ x_3^{(k+1)} &= \frac{1}{a_{33}}(b_3 - a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)}). \end{aligned}\right\} \quad \text{(GSS)}$$

This sequence of iterates is called the **Gauss-Seidel iterative sequence** and the method is called **Gauss-Seidel Iteration method**.

Remark 3.54. Compare (JS) and (GSS). □

Theorem 3.55. *If the coefficient matrix is diagonally dominant, then the Gauss-Seidel method converges to the exact solution of the system $A\boldsymbol{x} = \boldsymbol{b}$.*

Proof:

Since A is diagonally dominant, all the diagonal elements of A are non-zero, and hence the Gauss-Seidel iterative sequence given by

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left\{ b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right\}, \quad i = 1, 2, \cdots, n. \tag{3.62}$$

is well-defined. The error in each component is given by

$$e_i^{(k+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} e_j^{(k+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} e_j^{(k)}, \quad i = 1, 2, \cdots, n. \tag{3.63}$$

For $i = 1, 2, \cdots, n,$, define

$$\alpha_i = \sum_{j=1}^{i-1} \left| \frac{a_{ij}}{a_{ii}} \right|,$$
$$\beta_i = \sum_{j=i+1}^{n} \left| \frac{a_{ij}}{a_{ii}} \right|,$$

with the convention that $\alpha_1 = \beta_n = 0$. Note that μ given in (3.58) can be written as

$$\mu = \max_{1 \le i \le n} (\alpha_i + \beta_i)$$

Since A is a diagonally dominant matrix, we have $\mu < 1$. Now

$$|e_i^{(k+1)}| \le \alpha_i \|\boldsymbol{e}^{(k+1)}\|_\infty + \beta_i \|\boldsymbol{e}^{(k)}\|_\infty, \quad i = 1, 2, \cdots, n. \tag{3.64}$$

Let l be such that

$$\|\boldsymbol{e}^{(k+1)}\|_\infty = |e_l^{(k+1)}|.$$

Then with $i = l$ in (3.64),

$$\|\boldsymbol{e}^{(k+1)}\|_\infty \le \alpha_l \|\boldsymbol{e}^{(k+1)}\|_\infty + \beta_l \|\boldsymbol{e}^{(k)}\|_\infty. \tag{3.65}$$

Since $\mu < 1$, we have $\alpha_l < 1$ and therefore the above inequality gives

$$\|\boldsymbol{e}^{(k+1)}\|_\infty \le \frac{\beta_l}{1 - \alpha_l} \|\boldsymbol{e}^{(k)}\|_\infty. \tag{3.66}$$

Define

$$\eta = \max_{1 \le i \le n} \frac{\beta_i}{1 - \alpha_i}. \tag{3.67}$$

Then the above inequality yields

$$\|\boldsymbol{e}^{(k+1)}\|_\infty \le \eta \|\boldsymbol{e}^{(k)}\|_\infty. \tag{3.68}$$

Since for each i,

$$(\alpha_i + \beta_i) - \frac{\beta_i}{1 - \alpha_i} = \frac{\alpha_i [1 - (\alpha_i + \beta_i)]}{1 - \alpha_i} \ge \frac{\alpha_i}{1 - \alpha_i} [1 - \mu] \ge 0, \tag{3.69}$$

we have

$$\eta \le \mu < 1. \tag{3.70}$$

Thus, when the coefficient matrix A is diagonally dominant, Gauss-Seidel method converges. □

Remark 3.56. A careful observation of the proof of the above theorem reveals that the Gauss-Seidel method converges faster than the Jacobi method by comparing (3.70) and (3.61). □

3.4.3 Mathematical Error

Let $\boldsymbol{x}^*$ denote the computed solution using some method. The mathematical error in the approximate solution when compared to the exact solution of a linear system $A\boldsymbol{x} = \boldsymbol{b}$ is given by

$$\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}^*. \tag{3.71}$$

Recall from Chapter 2 that the mathematical error is due to the approximation made in the numerical method where the computation is done without any floating-point approximation (ie., without rounding or chopping). Observe that to get the mathematical error, we need to know the exact solution. But an astonishing feature of linear systems (which is not there in nonlinear equations) is that this error can be obtained exactly without the knowledge of the exact solution. To do this, we first define the **residual vector**

$$\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}^* \tag{3.72}$$

in the approximation of $\boldsymbol{b}$ by $A\boldsymbol{x}^*$. This vector is also referred to as **residual error**. Since $\boldsymbol{b} = A\boldsymbol{x}$, we have

$$\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}^* = A\boldsymbol{x} - A\boldsymbol{x}^* = A(\boldsymbol{x} - \boldsymbol{x}^*).$$

The above identity can be written as

$$A\boldsymbol{e} = \boldsymbol{r}. \tag{3.73}$$

This shows that the error $\boldsymbol{e}$ satisfies a linear system with the same coefficient matrix A as in the original system $A\boldsymbol{x} = \boldsymbol{b}$, but a different right hand side vector. Thus, by having the approximate solution $\boldsymbol{x}^*$ in hand, we can obtain the error $\boldsymbol{e}$ without knowing the exact solution $\boldsymbol{x}$ of the system.

3.4.4 Residual Corrector Method

When we use a computing device for solving a linear system, irrespective to whether we use direct methods or iterative methods, we always get an approximate solution. An attractive feature (as discussed in the above section) of linear systems is that the error involved in the approximate solution when compared to the exact solution can theoretically be obtained exactly. In this section, we discuss how to use this error to develop an iterative procedure to increase the accuracy of the obtained approximate solution using any other numerical method.

There is an obvious difficulty in the process of obtaining $\boldsymbol{e}$ as the solution of the system (3.73), especially on a computer. Since $\boldsymbol{b}$ and $A\boldsymbol{x}^*$ are very close to each other, the computation of $\boldsymbol{r}$ involves loss of significant digits which leads to zero residual error, which is of no use. To avoid this situation, the calculation of (3.72) should be carried out at a higher-precision. For instance, if $\boldsymbol{x}^*$ is computed using single-precision, then $\boldsymbol{r}$ can be computed using double-precision and then rounded back to single precision. Let us illustrate the computational procedure of the residual error in the following example.

Example 3.57 (Computation of residual at higher precision).

Consider the system

$$\begin{aligned} 0.729x_1 + 0.81x_2 + 0.9x_3 &= 0.6867 \\ x_1 + x_2 + x_3 &= 0.8338 \\ 1.331x_1 + 1.210x_2 + 1.100x_3 &= 1.000 \end{aligned}$$

The exact solution of this system with 4-digit rounding is

$$x_1 \approx 0.2245,\ x_2 \approx 0.2814,\ x_3 \approx 0.3279.$$

The solution of the system by Gaussian elimination without pivoting using 4-digit rounding leads to

$$x_1^* \approx 0.2251, \quad x_2^* \approx 0.2790, \quad x_3^* \approx 0.3295.$$

As we have used 4-digit rounding arithmetic in obtaining the approximate solution vector $\boldsymbol{x}^*$, we use 8-digit rounding arithmetic in computing the residual, which is obtained as

$$\boldsymbol{r} = (0.00006210, 0.0002000, 0.0003519)^T.$$

Solving the linear system $A\boldsymbol{e} = \boldsymbol{r}$ using 8-digit rounding arithmetic, we obtain the approximation as

$$\boldsymbol{e}^* = [-0.00044710, 0.00215000, -0.00150400]^T.$$

Compare this to the true error

$$\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}^* = [-0.0007, 0.0024, -0.0016]^T$$

Thus $\boldsymbol{e}^*$ gives a good idea of the size of the error $\boldsymbol{e}$ in the computed solution $\boldsymbol{x}^*$. □

Let us now propose an iterative procedure by first predicting the error by solving the system (3.73) and then correcting the approximate solution $\boldsymbol{x}^*$ by adding the predicted error to the vector $\boldsymbol{x}^*$.

If we take $\boldsymbol{x}^* = \boldsymbol{x}^{(0)}$, and define $\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$, then the error $\boldsymbol{e}^{(0)} = \boldsymbol{x} - \boldsymbol{x}^{(0)}$ can be obtained by solving the linear system

$$A\boldsymbol{e}^{(0)} = \boldsymbol{r}^{(0)}.$$

Now, define

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \boldsymbol{e}^{(0)}.$$

We expect the vector $\boldsymbol{x}^{(1)}$ is more closer to the exact solution than $\boldsymbol{x}^{(0)}$. Again compute the residual error vector $\boldsymbol{r}^{(1)}$ using the formula

$$\boldsymbol{r}^{(1)} = \boldsymbol{b} - A\boldsymbol{x}^{(1)},$$

and solve the corresponding linear system

$$A\boldsymbol{e}^{(1)} = \boldsymbol{r}^{(1)}$$

and define

$$\boldsymbol{x}^{(2)} = \boldsymbol{x}^{(1)} + \boldsymbol{e}^{(1)}.$$

Continuing in this way, we can generate a sequence $\{\boldsymbol{x}^{(k)}\}$ using the formula

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{e}^{(k)}. \tag{3.74}$$

where

$$A\boldsymbol{e}^{(k)} = \boldsymbol{r}^{(k)}$$

with

$$\boldsymbol{r}^{(k)} = \boldsymbol{b} - A\boldsymbol{x}^{(k)},$$

for $k = 0, 1, \cdots$. The above iterative procedure is called th **residual corrector method** (also called the **iterative refinement method**). Note that in computing $\boldsymbol{r}^{(k)}$ and $\boldsymbol{e}^{(k)}$, we use a higher precision than the precision used in computing $\boldsymbol{x}^{(k)}$.

Example 3.58. Using Gaussian elimination with pivoting and four digit rounding in solving the linear system

$$\begin{aligned} x_1 + 0.5x_2 + 0.3333x_3 &= 1 \\ 0.5x_1 + 0.3333x_2 + 0.25x_3 &= 0 \\ 0.3333x_1 + 0.25x_2 + 0.2x_3 &= 0 \end{aligned}$$

we obtain the solution as $\boldsymbol{x} = (9.062,\ -36.32,\ 30.30)^T$. Let us start with an initial guess of

$$\boldsymbol{x}^{(0)} = (8.968,\ -35.77,\ 29.77)^T.$$

Using 8-digit rounding arithmetic, we obtain

$$\boldsymbol{r}^{(0)} = (-0.00534100,\ -0.00435900,\ -0.00053440)^T.$$

After solving the system $A\boldsymbol{e}^{(0)} = \boldsymbol{r}^{(0)}$ using Gaussian elimination with pivoting using 8-digit rounding and the final answer is rounded to four digits, we get

$$\boldsymbol{e}^{(0)} = (0.0922,\ -0.5442,\ 0.5239)^T.$$

Hence, the corrected solution after the first iteration is

$$\boldsymbol{x}^{(1)} = (9.060,\ -36.31,\ 30.29)^T.$$

Similarly, we can predict the error in $\boldsymbol{x}^{(1)}$ when compared to the exact solution and correct the solution to obtain the second iterated vector as

$$\begin{aligned} \boldsymbol{r}^{(1)} &= (-0.00065700,\ -0.00037700,\ -0.00019800)^T, \\ \boldsymbol{e}^{(2)} &= (0.0017,\ -0.0130,\ 0.0124)^T, \\ \boldsymbol{x}^{(2)} &= (9.062,\ -36.32,\ 30.30)^T, \end{aligned}$$

and so on. □

The convergence of the iterative procedure to the exact solution is omitted for this course.

3.4.5 Stopping Criteria

In the iterative methods discussed above, we have a sequence $\{\boldsymbol{x}^{(k)}\}$ that is expected to converge to the exact solution of the given linear system. In practical situation, we cannot go on computing the $\boldsymbol{x}^{(k)}$ indefinitely and we need to terminate our computation once the value of $\boldsymbol{x}^{(k)}$ reaches a desired accuracy for a sufficiently large k. That is, when the error

$$\|\boldsymbol{e}^{(k)}\| = \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\|$$

in the k^{th} iteration in some norm is sufficiently small. Since, we do not know the exact solution $\boldsymbol{x}$, the error given above cannot be computed easily and needs another linear system (3.73) to be solved. Therefore, the question is how to decide where we have to stop our computation (without solving this linear system)? In other words, how do we know whether the computed vector $\boldsymbol{x}^{(k)}$ at the k^{th} iteration is sufficiently close to the exact solution or not. This can be decided by looking at the residual error vector of the k^{th} iteration defined as

$$\boldsymbol{r}^{(k)} = \boldsymbol{b} - A\boldsymbol{x}^{(k)}. \tag{3.75}$$

Thus, for a given sufficiently small positive number ϵ, we stop the iteration if

$$\|\boldsymbol{r}^{(k)}\| < \epsilon,$$

for some vector norm $\|\cdot\|$.

3.5 Eigenvalue Problems

Let A be an $n \times n$ matrix with real entries. Eigenvalues of A are defined as the roots of the equation

$$\det(\lambda I - A) = 0. \tag{3.76}$$

Note that $\det(\lambda I - A)$ is a polynomial in λ of degree n, which is known as **characteristic polynomial of the matrix** A. We know that even if A has real entries, the eigenvalues need not be real numbers. It is also a known fact that for every matrix, there are n eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_n$ (in this list, each eigenvalue is repeated as many times as its algebraic multiplicity, *i.e.,* multiplicity of the eigenvalue as a root of the characteristic polynomial).

When $n = 2$ the characteristic polynomial is a quadratic polynomial for which there is a nice formula for computing roots. When $n = 3$, there is a formula that many of us do not remember. When $n = 4$, none has a formula. But computing eigenvalues is important for applications. Therefore, numerically approximating the eigenvalues is the only way out.

One obvious way of approximating an eigenvalue of a matrix is to first obtain the characteristic polynomial (3.76) explicitly in λ and then use one of the numerical methods discussed in Chapter 4 to compute a root of this polynomial. But this is not an efficient way of computing eigenvalues because of two reasons. One reason is that obtaining explicit form

of (3.76) is itself a difficult task when the dimension of the matrix is very large. Secondly, if we make any small error (like floating-point error) in obtaining the explicit form of the polynomial (3.76), the resulting polynomial may have a root which is entirely different from any of the eigenvalues that we are looking for. This is illustrated in the following example by Wilkinson where we see that "the roots of polynomials are extremely sensitive to perturbations in the coefficients".

Example 3.59 (Wilkinson's example). Let $f(x)$ and $g(x)$ be two polynomials given by

$$f(x) = (x-1)(x-2)\cdots(x-10), \qquad g(x) = x^{10}.$$

The roots of the polynomial $f(x)$ are $1, 2, \cdots, 10$, and all these roots are simple roots. If we perturb this polynomial as $F(x) = f(x) + 0.01g(x)$, then all the roots lie in the interval $[1, 3.5]$ (verified graphically). In fact, the largest root of the polynomial $f(x)$ is 10 and the largest root of the polynomial $F(x)$ is approximately equal to 3.398067. Thus, if the coefficient of x^{10} is perturbed by a small amount of 0.01, the root 10 of $f(x)$ could move as much a distance as approximately 6.6. □

Due to the two reasons discussed above, we look for an alternate method to compute the eigenvalues of a given matrix. One such method is the **power method** that can be used to obtain the eigenvalue which is the largest in magnitude among all the other eigenvalues and the corresponding eigen vector. In Subsection 3.5.1, we present the power method and discuss the condition under which this method can be applied. In Subsection 3.5.2 we prove the Gerschgorin theorem which may be used as a tool to find a class of matrices for which power method can be applied successfully.

3.5.1 Power Method

There are many variations of Power method in the literature. We will present the most elementary form of Power method. We always deal with matrices with real entries, all of whose eigenvalues are real numbers.

Power method is used to obtain a specific eigenvalue called **dominant eigenvalue** and corresponding eigenvector for a given $n \times n$ matrix A. The concept of a dominant eigenvalue plays a very important role in many applications. The Power method provides an approximation to it under some conditions on the matrix. We now define the concept of a dominant eigenvalue.

Definition 3.60 (Dominant Eigenvalue of a Matrix).
An eigenvalue λ of an $n \times n$ matrix A is said to be a **dominant eigenvalue** *of A if*

$$|\lambda| = \max\{\, |z| \,:\, z \text{ is an eigenvalue of } A \,\}.$$

Remark 3.61.

(1) If a dominant eigenvalue of a matrix is equal to zero, then all its eigenvalues are zero and an eigenvector can be found by solving the linear system $A\boldsymbol{x} = \mathbf{0}$.

(2) If λ is a dominant eigenvalue of a matrix A, then there is no other eigenvalue of A whose distance from zero is more than that of λ from zero.

(3) Let $\mu_1, \mu_2, \cdots, \mu_n$ be the eigenvalues of A (repeated according to their algebraic multiplicities) of an $n \times n$ matrix A. These eigenvalues can be re-named (re-labelled, re-indexed) as $\lambda_1, \lambda_2, \cdots, \lambda_n$ such that they satisfy the condition:

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|.$$

Note that λ_1 is a **dominant eigenvalue** of A. □

Example 3.62. A matrix may have a unique dominant eigenvalue or more than one dominant eigenvalues. Further, even if dominant eigenvalue is unique the corresponding algebraic and geometric multiplicities could be more than one, and also both algebraic and geometric multiplicities may not be the same. All these possibilities are illustrated in this example.

(1) The matrix

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

has eigenvalues 1, -1, and -2. The matrix A has a unique dominant eigenvalue, which is -2 as this is the largest in absolute value, of all eigenvalues. Note that the dominant eigenvalue of A is a simple eigenvalue.

(2) The matrix

$$B = \begin{pmatrix} 1 & 3 & 4 \\ 0 & 2 & 1 \\ 0 & 0 & -2 \end{pmatrix}$$

has eigenvalues 1, -2, and 2. According to our definition, the matrix B has two dominant eigenvalues. They are -2 and 2. Note that both the dominant eigenvalues of B are simple eigenvalues.

(3) Consider the matrices

$$C_1 = \begin{pmatrix} 1 & 3 & 4 \\ 0 & 2 & 5 \\ 0 & 0 & 2 \end{pmatrix}, C_2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}, C_3 = \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}.$$

The matrix C_1 has a unique dominant eigenvalue 2, which has algebraic multiplicity 2 and geometric multiplicity 1. The matrix C_2 has a unique dominant eigenvalue 2, whose algebraic and geometric multiplicities equal 2. The matrix C_3 has a unique dominant eigenvalue 2, which has algebraic multiplicity 2 and geometric multiplicity 1. □

As mentioned above the power method is used to compute the dominant eigenvalue and the corresponding eigen vector of a given $n \times n$ matrix provided this eigenvalue is unique. Thus, in the above examples, power method can be used for the matrices A but not for B even though B has distinct eigenvalues. Let us now detail the **power method**.

Assumptions on the matrix for which the power method can work

Assume that an $n \times n$ matrix A has real eigenvalues λ_1, λ_2, $\cdots$, and λ_n (repeated according to their algebraic multiplicities) with the following properties:

(1) The eigenvalues are such that

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n| \tag{3.77}$$

That is, A has a unique dominant eigenvalue λ_1 which is a simple eigenvalue.

(2) There exists a basis of $\mathbb{R}^n$ consisting of eigenvectors of A. That is, there exists $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n$ satisfying $A\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k$ for $k = 1, 2, \cdots, n$; and such that for each $\boldsymbol{v} \in \mathbb{R}^n$ there exists unique real numbers $c_1, c_2, \cdots, c_n$ such that

$$\boldsymbol{v} = c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{v}_n.$$

Equivalently, the matrix A is diagonalizable.

Let us now discuss the key idea behind power method.

- Choose a non-zero vector $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$ arbitrarily so that we can find scalars c_1, c_2, $\cdots$, $c_n \in \mathbb{R}$ such that
$$\boldsymbol{x}^{(0)} = c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{v}_n, \quad c_1 \neq 0.$$
- Pre-multiplying by A and substituting $A\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$, $i = 1, \cdots, n$, we get
$$A\boldsymbol{x}^{(0)} = c_1\lambda_1\boldsymbol{v}_1 + \cdots + c_n\lambda_n\boldsymbol{v}_n = \lambda_1 \left(c_1\boldsymbol{v}_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)\boldsymbol{v}_2 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)\boldsymbol{v}_n \right).$$
Note here that we have assumed $\lambda_1 \neq 0$, which follows from the Assumption (1) above.
- Pre-multiplying by A again and simplying, we get
$$A^2\boldsymbol{x}^{(0)} = \lambda_1^2 \left(c_1\boldsymbol{v}_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^2 \boldsymbol{v}_2 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^2 \boldsymbol{v}_n \right)$$
- For each $k \in \mathbb{N}$, applying A k-times on $\boldsymbol{x}^{(0)}$ yields
$$A^k\boldsymbol{x}^{(0)} = \lambda_1^k \left(c_1\boldsymbol{v}_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k \boldsymbol{v}_2 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k \boldsymbol{v}_n \right) \tag{3.78}$$
- Using the assumption (3.77), we get $|\lambda_k/\lambda_1| < 1$, for $k = 2, \cdots, n$. Therefore, we have
$$\lim_{k\to\infty} \frac{A^k\boldsymbol{x}^{(0)}}{\lambda_1^k} = c_1\boldsymbol{v}_1. \tag{3.79}$$
For $c_1 \neq 0$, the right hand side of the above equation is a scalar multiple of the eigenvector.

- From the above expression for $A^k\boldsymbol{x}^{(0)}$, we also see that

$$\lim_{k\to\infty}\frac{(A^{k+1}\boldsymbol{x}^{(0)})_i}{(A^k\boldsymbol{x}^{(0)})_i}=\lambda_1, \tag{3.80}$$

  where i is any index such that the fractions on the left hand side are meaningful (which is the case when $\boldsymbol{x}^{(0)}\notin\cup_{k=1}^{\infty}\mathrm{K}erA^k$).

The power method generates two sequences $\{\mu_k\}$ and $\{\boldsymbol{x}^{(k)}\}$, using the results (3.79) and (3.80), that converge to the dominant eigenvalue λ_1 and the corresponding eigen vectors $\boldsymbol{v}_1$, respectively. We will now give the method of generating these two sequences.

Step 1: Choose a vector $\boldsymbol{x}^{(0)}$ arbitrarily and set $\boldsymbol{y}^{(1)}:=A\boldsymbol{x}^{(0)}$.
Step 2: Define $\mu_1:=y_i^{(1)}$, where $i\in\{1,\cdots,n\}$ is the least index such that $\|\boldsymbol{y}^{(1)}\|_\infty=|y_i^{(1)}|$ and set $\boldsymbol{x}^{(1)}:=\boldsymbol{y}^{(1)}/\mu_1$.

From $\boldsymbol{x}^{(1)}$, we can obtain μ_2 and $\boldsymbol{x}^{(2)}$ in a similar way.

Power method iterative sequence:

In general, for $k=0,1,\cdots$, we choose the initial vector $\boldsymbol{x}^{(0)}$ arbitrarily and generate the sequences $\{\mu^{(k)}\}$ and $\{\boldsymbol{x}^{(k)}\}$ using the formulas

$$\mu_{k+1}=y_i^{(k+1)},\quad \boldsymbol{x}^{(k+1)}=\frac{\boldsymbol{y}^{(k+1)}}{\mu_{k+1}}, \tag{3.81}$$

where

$$\boldsymbol{y}^{(k+1)}=A\boldsymbol{x}^{(k)},\quad \|\boldsymbol{y}^{(k+1)}\|_\infty=|y_i^{(k+1)}|. \tag{3.82}$$

This iterative procedure is called the **power method**.

Remark 3.63. The scaling factor μ_k introduced in (3.81) makes sure that $\boldsymbol{x}^{(k)}$ has its maximum norm equal to 1, *i.e.,* $\|\boldsymbol{x}^{(k)}\|_\infty=1$. This rules out the possibilities of $\lim_{k\to\infty}\boldsymbol{x}^{(k)}$ being $\mathbf{0}$ or the vector $\boldsymbol{x}^{(k)}$ escaping to infinity. □

We now discuss the sufficient conditions under which the power method converges.

Theorem 3.64 (Convergence Theorem for Power method).

Hypothesis: *Let A be an non-singular $n\times n$ matrix with real eigenvalues having the following properties:*

(H1) *A has a unique dominant eigenvalue λ_1 which is a simple eigenvalue. That is,*

$$|\lambda_1|>|\lambda_2|\geq\cdots\geq|\lambda_n|,$$

where $\lambda_1,\lambda_2,\cdots,\lambda_n$ are the eigenvalues of A (repeated according to their algebraic multiplicities).

(H2) *A has n linearly independent real eigenvectors, $\boldsymbol{v}_i$, $i=1,\cdots,n$.*

(H3) *An initial guess* $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$ *be chosen such that*

$$\boldsymbol{x}^{(0)} = \sum_{j=1}^{n} c_j \boldsymbol{v}_j, \tag{3.83}$$

for some scalars $c_1, c_2, \cdots, c_n \in \mathbb{R}$ *with* $c_1 \neq 0$ *and* $\boldsymbol{x}^{(0)} \notin \bigcup_{k=1}^{\infty} \mathrm{K}erA^k$.

Conclusion: *Then, in the power method* (3.81)-(3.82),

(1) *the sequence* $\{\mu_k\}$ *converges to the dominant eigenvalue* λ_1 *and*

(2) *the sequence* $\{\boldsymbol{x}_k\}$ *converges either to* $+\boldsymbol{v}_1/\|\boldsymbol{v}_1\|_\infty$ *or to* $-\boldsymbol{v}_1/\|\boldsymbol{v}_1\|_\infty$ *or oscillates between these two vectors. Here,* $\boldsymbol{v}_1$ *denotes the eigenvector corresponding to the eigenvalue* λ_1.

Remark 3.65. Note that in Conclusion (2) of the above theorem, we see that whatever may be the case (among the three cases stated), the sequence $\{\boldsymbol{x}_k\}$ approaches the eigenspace associated with the eigenvalue λ_1, as $k \to \infty$. □

Proof. From the definition of $\boldsymbol{x}^{(k+1)}$, we have

$$\boldsymbol{x}^{(k+1)} = \frac{A\boldsymbol{x}^{(k)}}{\mu_{k+1}} = \frac{A\boldsymbol{y}^{(k)}}{\mu_{k+1}\mu_k} = \frac{AA\boldsymbol{x}^{(k-1)}}{\mu_{k+1}\mu_k} = \frac{A^2\boldsymbol{x}^{(k-1)}}{\mu_{k+1}\mu_k} = \cdots = \frac{A^{k+1}\boldsymbol{x}^{(0)}}{\mu_{k+1}\mu_k \cdots \mu_1}.$$

Therefore, we have

$$\boldsymbol{x}^{(k+1)} = m_{k+1} A^{k+1} \boldsymbol{x}^{(0)},$$

where $m_{k+1} = 1/(\mu_1\mu_2 \cdots \mu_{k+1})$. But, $\boldsymbol{x}^{(0)} = \sum_{j=1}^{n} c_j \boldsymbol{v}_j$, $c_1 \neq 0$. Therefore

$$\boldsymbol{x}^{(k+1)} = m_{k+1}\lambda_1^{k+1} \left(c_1\boldsymbol{v}_1 + \sum_{j=2}^{n} c_j \left(\frac{\lambda_j}{\lambda_1}\right)^{k+1} \boldsymbol{v}_j \right).$$

Taking maximum norm on both sides and noting that $\|\boldsymbol{x}^{(k)}\|_\infty = 1$, we get

$$1 = \left| m_{k+1}\lambda_1^{k+1} \right| \left\| c_1\boldsymbol{v}_1 + \sum_{j=2}^{n} c_j \left(\frac{\lambda_j}{\lambda_1}\right)^{k+1} \boldsymbol{v}_j \right\|_\infty.$$

Since $|\lambda_j/\lambda_1|^k \to 0$ as $k \to \infty$, we get

$$\lim_{k\to\infty} \left| m_{k+1}\lambda_1^{k+1} \right| = \frac{1}{|c_1|\|\boldsymbol{v}_1\|_\infty} < \infty.$$

Using this, we get

$$\lim_{k\to\infty} \boldsymbol{x}^{(k+1)} = \lim_{k\to\infty} m_{k+1}\lambda_1^{k+1} c_1 \boldsymbol{v}_1 = \begin{cases} \text{either} & +\dfrac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|_\infty} \\ \text{or} & -\dfrac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|_\infty} \\ \text{or} & \text{oscillates between} \\ & \text{the above two vectors} \end{cases} . \tag{3.84}$$

This completes the proof of Conclusion (2).

Let us now prove that $\mu_k \to \lambda_1$. For this, we first note that

$$\boldsymbol{y}^{(k+1)} = A\boldsymbol{x}^{(k)}.$$

Therefore, from (3.84), we see that (up to a subsequence)

$$\lim_{k\to\infty} \boldsymbol{y}^{(k+1)} = \lim_{k\to\infty} A\boldsymbol{x}^{(k)} = K\lambda_1 \boldsymbol{v}_1,$$

where $K = \pm 1/\|\boldsymbol{v}_1\|_\infty$. Since $\boldsymbol{v}_1$ is an eigen vector, there is at least one non-zero component of $\boldsymbol{v}_1$. We choose one such component of $\boldsymbol{v}_1$ and denote it by $(v_1)_j$. Since $(v_1)_j \neq 0$ and $y_j^{(k+1)} \to K\lambda_1(v_1)_j$, there exists an integer $N > 0$ such that

$$y_j^{(k+1)} \neq 0, \text{ for all } k \geq N.$$

Similarly, we see that

$$x_j^{(k+1)} \neq 0, \text{ for all } k \geq N.$$

Also, we know that

$$\mu_{k+1}\boldsymbol{x}^{(k+1)} = \boldsymbol{y}^{(k+1)}.$$

Therefore, we can write

$$\mu_{k+1} = \frac{y_j^{(k+1)}}{x_j^{(k+1)}} = \frac{(A\boldsymbol{x}^{(k)})_j}{(\boldsymbol{x}^{(k+1)})_j},$$

where j denotes the component as given above. Taking limit, we have

$$\begin{aligned} \lim_{k\to\infty} \mu_{k+1} &= \frac{K(A\boldsymbol{v}_1)_j}{K(\boldsymbol{v}_1)_j} \\ &= \frac{\lambda_1(\boldsymbol{v}_1)_j}{(\boldsymbol{v}_1)_j} \\ &= \lambda_1. \end{aligned}$$

which gives the desired result. □

Note that the above theorem does not guarantee the convergence of the sequence $\{\boldsymbol{x}_n\}$ to an eigenvector of the dominant eigenvalue. However, if the dominant eigenvalue has an eigenvector with a unique dominant component, then this sequence converges as discussed in the following theorem.

Theorem 3.66 (Second Convergence Theorem for Power Method).

Let A be an $n \times n$ matrix satisfying the hypotheses **(H1)**, **(H2)**, *and* **(H3)** *of Theorem 3.64. In addition to these hypotheses,*

(H4) *let $\boldsymbol{v}_1 = (v_{11}, v_{12}, \cdots, v_{1n})^T$ be such that there exists a unique index $j \in \{1, 2, \cdots, n\}$ with the property*

$$|v_{1j}| = \|\boldsymbol{v}_1\|_\infty \tag{3.85}$$

This hypothesis is referred to as $\boldsymbol{v}_1$ **has a single maximal component.**

Conclusion: *Then, in the power method* (3.81)-(3.82),

(1) *the sequence $\{\mu_k\}$ converges to the dominant eigenvalue λ_1 and*

(2) *The sequence of vectors $\boldsymbol{x}^{(k)}$ converges to an eigenvector corresponding to the dominant eigenvalue λ_1.*

Proof. Let us first set up some notation. Let the eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n$ be given by

$$\boldsymbol{v}_j = (v_{j1}, v_{j2}, \cdots, v_{jn})^T, \text{ for } j = 1, 2, \cdots, n. \tag{3.86}$$

Since $\boldsymbol{x}^{(0)} = c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{v}_n$, we have

$$A\boldsymbol{x}^{(0)} = c_1\lambda_1\boldsymbol{v}_1 + c_2\lambda_2\boldsymbol{v}_2 + \cdots + c_n\lambda_n\boldsymbol{v}_n.$$

In coordinate form, we have

$$A\boldsymbol{x}^{(0)} = (\lambda_1 c_1 v_{11} + \lambda_2 c_2 v_{21} + \cdots + \lambda_n c_n v_{n1}, \cdots, \lambda_1 c_1 v_{1n} + \lambda_2 c_2 v_{2n} + \cdots + \lambda_n c_n v_{nn})^T \tag{3.87}$$

In fact, for each $k \in \mathbb{N}$, we have

$$A^k\boldsymbol{x}^{(0)} = (\lambda_1^k c_1 v_{11} + \lambda_2^k c_2 v_{21} + \cdots + \lambda_n^k c_n v_{n1}, \cdots, \lambda_1^k c_1 v_{1n} + \lambda_2^k c_2 v_{2n} + \cdots + \lambda_n^k c_n v_{nn})^T \tag{3.88}$$

Maximum norm of the vector $A^k\boldsymbol{x}^{(0)}$ is going to be the modulus of one of its components. From **(H4)**, we have

$$\frac{|v_{1i}|}{|v_{1j}|} \leq 1 \quad \text{for } i = 1, 2, \cdots, n. \tag{3.89}$$

Observe that

$$\frac{(A^k\boldsymbol{x}^{(0)})_i}{(A^k\boldsymbol{x}^{(0)})_j} = \frac{\lambda_1^k c_1 v_{1i} + \lambda_2^k c_2 v_{2i} + \cdots + \lambda_n^k c_n v_{ni}}{\lambda_1^k c_1 v_{1j} + \lambda_2^k c_2 v_{2j} + \cdots + \lambda_n^k c_n v_{nj}}.$$

The last equation can be written in the form

$$\frac{(A^k\boldsymbol{x}^{(0)})_i}{(A^k\boldsymbol{x}^{(0)})_j} = \frac{v_{1i} + \dfrac{\lambda_2^k c_2}{\lambda_1^k c_1} v_{2i} + \cdots + \dfrac{\lambda_n^k c_n}{\lambda_1^k c_1} v_{ni}}{v_{1j} + \dfrac{\lambda_2^k c_2}{\lambda_1^k c_1} v_{2j} + \cdots + \dfrac{\lambda_n^k c_n}{\lambda_1^k c_1} v_{nj}} \tag{3.90}$$

Note that the RHS in the equation (3.90) converges to $\dfrac{v_{1i}}{v_{1j}}$ as $k \to \infty$. Since,

$$\left|\frac{v_{1i}}{v_{1j}}\right| < 1, \qquad \text{for} \quad i \neq j, \tag{3.91}$$

we can conclude that there exists a $K \in \mathbb{N}$ such that for $k \geq K$,

$$\left|\frac{(A^k \boldsymbol{x}^{(0)})_i}{(A^k \boldsymbol{x}^{(0)})_j}\right| < 1. \tag{3.92}$$

As a consequence, the maximum norm of $A^k \boldsymbol{x}^{(0)}$ is equal to $|(A^k \boldsymbol{x}^{(0)})_j|$ where j is as given in **(H4)**.

For $k \geq K$, the sequence μ_k is given by

$$\mu_k = \frac{(A^k \boldsymbol{x}^{(0)})_j}{(A^{k-1} \boldsymbol{x}^{(0)})_j} = \lambda_1 \frac{v_{1j} + \dfrac{\lambda_2^k c_2}{\lambda_1^k c_1} v_{2j} + \cdots + \dfrac{\lambda_n^k c_n}{\lambda_1^k c_1} v_{nj}}{v_{1j} + \dfrac{\lambda_2^{k-1} c_2}{\lambda_1^{k-1} c_1} v_{2j} + \cdots + \dfrac{\lambda_n^{k-1} c_n}{\lambda_1^{k-1} c_1} v_{nj}}. \tag{3.93}$$

Thus,

$$\lim_{k \to \infty} \mu_k = \lambda_1. \tag{3.94}$$

For $k \geq K$, the sequence $\boldsymbol{x}^{(k)}$ is given by

$$\begin{aligned} \boldsymbol{x}^{(k)} &= \frac{A^k \boldsymbol{x}^{(0)}}{(A^k \boldsymbol{x}^{(0)})_j} \\ &= \left(\frac{(A^k \boldsymbol{x}^{(0)})_1}{(A^k \boldsymbol{x}^{(0)})_j}, \cdots, \frac{(A^k \boldsymbol{x}^{(0)})_{j-1}}{(A^k \boldsymbol{x}^{(0)})_j}, 1, \frac{(A^k \boldsymbol{x}^{(0)})_{j+1}}{(A^k \boldsymbol{x}^{(0)})_j}, \cdots, \frac{(A^k \boldsymbol{x}^{(0)})_n}{(A^k \boldsymbol{x}^{(0)})_j} \right)^T. \end{aligned}$$

In view of (3.90), we now conclude that the sequence $\boldsymbol{x}^{(k)}$ converges to $\frac{1}{v_{1j}} \boldsymbol{v}_1$ which is an eigenvector corresponding to the dominant eigenvalue λ_1. □

We now give a numerical example illustrating power method procedure.

Example 3.67. Consider the matrix

$$A = \begin{pmatrix} 3 & 0 & 0 \\ -4 & 6 & 2 \\ 16 & -15 & -5 \end{pmatrix}.$$

The eigenvalues of this matrix are $\lambda_1 = 3$, $\lambda_2 = 1$ and $\lambda_3 = 0$. The corresponding eigen vectors are $\boldsymbol{v}_1 = (1, 0, 2)^T$, $\boldsymbol{v}_2 = (0, 2, -5)^T$ and $\boldsymbol{v}_3 = (0, 1, -3)^T$. Thus, the hypothesis (H1) and (H2) of the Theorem 3.64 are satisfied. Choose the initial guess $\boldsymbol{x}_0 = (1, 0.5, 0.25)^T$, which also satisfies the hypothesis (H3).

The first ten terms of the iterative sequence in power method given by (3.81)-(3.82) for the given matrix A are as follows:

Iteration No: 1

$$\begin{aligned}\boldsymbol{y}_1 &= A\boldsymbol{x}_0 = (3.000000, -0.500000, 7.250000)^T\\ \mu_1 &= 7.250000\\ \boldsymbol{x}_1 &= \frac{\boldsymbol{y}_1}{\mu_1} = (0.413793, -0.068966, 1.000000)^T\end{aligned}$$

Iteration No: 2

$$\begin{aligned}\boldsymbol{y}_2 &= A\boldsymbol{x}_1 = (1.241379, -0.068966, 2.655172)^T\\ \mu_2 &= 2.655172\\ \boldsymbol{x}_2 &= \frac{\boldsymbol{y}_2}{\mu_2} = (0.467532, -0.025974, 1.000000)^T\end{aligned}$$

Iteration No: 3

$$\begin{aligned}\boldsymbol{y}_3 &= A\boldsymbol{x}_2 = (1.402597, -0.025974, 2.870130)^T\\ \mu_3 &= 2.870130\\ \boldsymbol{x}_3 &= \frac{\boldsymbol{y}_3}{\mu_3} = (0.488688, -0.009050, 1.000000)^T\end{aligned}$$

Iteration No: 4

$$\begin{aligned}\boldsymbol{y}_4 &= A\boldsymbol{x}_3 = (1.466063, -0.009050, 2.954751)^T\\ \mu_4 &= 2.954751\\ \boldsymbol{x}_4 &= \frac{\boldsymbol{y}_4}{\mu_4} = (0.496172, -0.003063, 1.000000)^T\end{aligned}$$

Iteration No: 5

$$\begin{aligned}\boldsymbol{y}_5 &= A\boldsymbol{x}_4 = (1.488515, -0.003063, 2.984686)^T\\ \mu_5 &= 2.984686\\ \boldsymbol{x}_5 &= \frac{\boldsymbol{y}_5}{\mu_5} = (0.498717, -0.001026, 1.000000)^T\end{aligned}$$

Iteration No: 6

$$\begin{aligned}\boldsymbol{y}_6 &= A\boldsymbol{x}_5 = (1.496152, -0.001026, 2.994869)^T\\ \mu_6 &= 2.994869\\ \boldsymbol{x}_6 &= \frac{\boldsymbol{y}_6}{\mu_6} = (0.499572, -0.000343, 1.000000)^T\end{aligned}$$

Iteration No: 7

$$\begin{aligned}\boldsymbol{y}_7 &= A\boldsymbol{x}_6 = (1.498715, -0.000343, 2.998287)^T\\ \mu_7 &= 2.998287\\ \boldsymbol{x}_7 &= \frac{\boldsymbol{y}_7}{\mu_7} = (0.499857, -0.000114, 1.000000)^T\end{aligned}$$

Iteration No: 8

$$\begin{aligned}\boldsymbol{y}_8 &= A\boldsymbol{x}_7 = (1.499571, -0.000114, 2.999429)^T\\ \mu_8 &= 2.999429\\ \boldsymbol{x}_8 &= \frac{\boldsymbol{y}_8}{\mu_8} = (0.499952, -0.000038, 1.000000)^T\end{aligned}$$

Iteration No: 9

$$\begin{aligned}\boldsymbol{y}_9 &= A\boldsymbol{x}_8 = (1.499857, -0.000038, 2.999809)^T\\ \mu_9 &= 2.999809\\ \boldsymbol{x}_9 &= \frac{\boldsymbol{y}_9}{\mu_9} = (0.499984, -0.000013, 1.000000)^T\end{aligned}$$

Iteration No: 10

$$\begin{aligned}\boldsymbol{y}_{10} &= A\boldsymbol{x}_9 = (1.499952, -0.000013, 2.999936)^T\\ \mu_{10} &= 2.999936\\ \boldsymbol{x}_{10} &= \frac{\boldsymbol{y}_{10}}{\mu_{10}} = (0.499995, -0.000004, 1.000000)^T\end{aligned}$$

These ten iterates suggest that the sequence $\{\mu_k\}$ converges to the eigenvalue $\lambda_1 = 3$ and the sequence $\{\boldsymbol{x}^{(k)}\}$ converges to $(0.5, 0, 1) = \frac{1}{2}\boldsymbol{v}_1$. □

Remark 3.68 (Disadvantages of power method).

(1) The Power method requires at the beginning that the matrix has only one dominant eigenvalue, and this information is generally unavailable.

(2) Even when there is only one dominant eigenvalue, it is not clear how to choose the initial guess $\boldsymbol{x}^{(0)}$ such that it has a non-zero component (c_1 in the notation of the theorem) along the eigenvector $\boldsymbol{v}_1$. □

Note that in the above example, all the hypothesis of Theorem 3.64 are satisfied. Now let us ask the question

"What happens when any of the hypotheses of Theorem is violated?"

We discuss these situations through examples.

Example 3.69 (Dominant eigenvalue is not unique (Failure of H1)).

Consider the matrix

$$B = \begin{pmatrix} 1 & 3 & 4\\ 0 & 2 & 1\\ 0 & 0 & -2\end{pmatrix},$$

which has eigenvalues 1, -2, and 2. Clearly, the matrix B has two dominant eigenvalues, namely, -2 and 2. We start with an initial guess $\boldsymbol{x}^{(0)} = (1, 1, 1)$ and the first five iterations generated using power method are given below:

Iteration No: 1

$$\begin{aligned}
\boldsymbol{y}_1 &= A\boldsymbol{x}_0 = (8.000000, 3.000000, -2.000000)^T \\
\mu_1 &= 8.000000 \\
\boldsymbol{x}_1 &= \frac{\boldsymbol{y}_1}{\mu_1} = (1.000000, 0.375000, -0.250000)^T
\end{aligned}$$

Iteration No: 2

$$\begin{aligned}
\boldsymbol{y}_2 &= A\boldsymbol{x}_1 = (1.125000, 0.500000, 0.500000)^T \\
\mu_2 &= 1.125000 \\
\boldsymbol{x}_2 &= \frac{\boldsymbol{y}_2}{\mu_2} = (1.000000, 0.444444, 0.444444)^T
\end{aligned}$$

Iteration No: 3

$$\begin{aligned}
\boldsymbol{y}_3 &= A\boldsymbol{x}_2 = (4.111111, 1.333333, -0.888889)^T \\
\mu_3 &= 4.111111 \\
\boldsymbol{x}_3 &= \frac{\boldsymbol{y}_3}{\mu_3} = (1.000000, 0.324324, -0.216216)^T
\end{aligned}$$

Iteration No: 4

$$\begin{aligned}
\boldsymbol{y}_4 &= A\boldsymbol{x}_3 = (1.108108, 0.432432, 0.432432)^T \\
\mu_4 &= 1.108108 \\
\boldsymbol{x}_4 &= \frac{\boldsymbol{y}_4}{\mu_4} = (1.000000, 0.390244, 0.390244)^T
\end{aligned}$$

Iteration No: 5

$$\begin{aligned}
\boldsymbol{y}_5 &= A\boldsymbol{x}_4 = (3.731707, 1.170732, -0.780488)^T \\
\mu_5 &= 3.731707 \\
\boldsymbol{x}_5 &= \frac{\boldsymbol{y}_5}{\mu_5} = (1.000000, 0.313725, -0.209150)^T
\end{aligned}$$

It is observed that the sequence oscillates even till 1000 iterations as shown below:

Iteration No: 998

$$\begin{aligned}
\boldsymbol{y}_{998} &= A\boldsymbol{x}_{997} = (1.103448, 0.413793, 0.413793)^T \\
\mu_{998} &= 1.103448 \\
\boldsymbol{x}_{998} &= \frac{\boldsymbol{y}_{998}}{\mu_{998}} = (1.000000, 0.375000, 0.375000)^T
\end{aligned}$$

Iteration No: 999

$$\begin{aligned}\boldsymbol{y}_{999} &= A\boldsymbol{x}_{998} = (3.625000, 1.125000, -0.750000)^T\\ \mu_{999} &= 3.625000\\ \boldsymbol{x}_{999} &= \frac{\boldsymbol{y}_{999}}{\mu_{999}} = (1.000000, 0.310345, -0.206897)^T\end{aligned}$$

Iteration No: 1000

$$\begin{aligned}\boldsymbol{y}_{1000} &= A\boldsymbol{x}_{999} = (1.103448, 0.413793, 0.413793)^T\\ \mu_{1000} &= 1.103448\\ \boldsymbol{x}_{1000} &= \frac{\boldsymbol{y}_{1000}}{\mu_{1000}} = (1.000000, 0.375000, 0.375000)^T\end{aligned}$$

and so on. This is a clear indication that the power method is not converging in this case. □

Thus we conclude that the power method when applied to a matrix which has more than one domainant eigenvalue may not converge.

Remark 3.70 (Dominant eigenvalue is not simple).

It is not necessary to have the dominant eigenvalue of algebraic multiplicity 1 in order that the iterative sequences of power method converge. The important thing is to have a unique dominant eigenvalue and it is allowed to have an algebraic multiplicity r with $r > 1$. However it is necessary that the corresponding geometric multiplicity should also be r to satifsy the hypothesis **(H2)** (also see later on, where we discuss the situation where algebraic and geometric multiplicities do not match). In such a case, Power method computes only one eigenvector, as is usual. □

Remark 3.71 (Failure of hypothesis (H2): Matrix is defective).

A matrix is said to be defective if for any of its eigenvalues, the algebraic and geometric multiplicities are not the same. Such matrices are not diagonalizable, and hence cannot satisfy the hypothesis **(H2)** Even the assumption that there is a basis of eigenvectors is not necessary. Interested curious reader is encouraged to think on the following lines. "If I dont have a basis of eigenvectors, the next best thing I know is that there is always a basis consisting of eigenvectors and generalized eigenvectors. If I simply go through the proof of Theorem 3.64 with this kind of basis, will I face obstacles? $\cdots$ " □

Let us now illustrate the situation when the hypothesis (H3) of Theorem 3.64 is violated.

Example 3.72 (Failure of hypothesis (H3): Initial guess $\boldsymbol{x}^{(0)}$ is such that $c_1 = 0$). Consider the matrix (same as in Example 3.67)

$$A = \begin{pmatrix} 3 & 0 & 0\\ -4 & 6 & 2\\ 16 & -15 & -5\end{pmatrix},$$

The eigenvalues of this matrix are $\lambda_1 = 3,\ \lambda_2 = 1$ and $\lambda_3 = 0$. The corresponding eigen vectors are $\boldsymbol{v}_1 = (1, 0, 2)^T,\ \boldsymbol{v}_2 = (0, 2, -5)^T$ and $\boldsymbol{v}_3 = (0, 1, -3)^T$. Thus, the hypothesis (H1) and (H2) of the Theorem 3.64 are satisfied.

Here, we choose a different initial guess $\boldsymbol{x}_0 = (0, 0.5, 0.25)^T$. Note that the hypothesis (H3) of the Theorem 3.64 that $c_1 \neq 0$ is violated here. However, we can see that $c_2 \neq 0$. The first four iterations of the power method are as follows:

Iteration No: 1

$$\begin{aligned}
\boldsymbol{y}_1 &= A\boldsymbol{x}_0 = (0.000000, 3.500000, -8.750000)^T \\
\mu_1 &= -8.750000 \\
\boldsymbol{x}_1 &= \frac{\boldsymbol{y}_1}{\mu_1} = (-0.000000, -0.400000, 1.000000)^T
\end{aligned}$$

Iteration No: 2

$$\begin{aligned}
\boldsymbol{y}_2 &= A\boldsymbol{x}_1 = (0.000000, -0.400000, 1.000000)^T \\
\mu_2 &= 1.000000 \\
\boldsymbol{x}_2 &= \frac{\boldsymbol{y}_2}{\mu_2} = (0.000000, -0.400000, 1.000000)^T
\end{aligned}$$

Iteration No: 3

$$\begin{aligned}
\boldsymbol{y}_3 &= A\boldsymbol{x}_2 = (0.000000, -0.400000, 1.000000)^T \\
\mu_3 &= 1.000000 \\
\boldsymbol{x}_3 &= \frac{\boldsymbol{y}_3}{\mu_3} = (0.000000, -0.400000, 1.000000)^T
\end{aligned}$$

Iteration No: 4

$$\begin{aligned}
\boldsymbol{y}_4 &= A\boldsymbol{x}_3 = (0.000000, -0.400000, 1.000000)^T \\
\mu_4 &= 1.000000 \\
\boldsymbol{x}_4 &= \frac{\boldsymbol{y}_4}{\mu_4} = (0.000000, -0.400000, 1.000000)^T
\end{aligned}$$

Thus, the power method converges to λ_2, which is the second dominant eigenvalue of the given matrix.

Note that in the chosen initial guess, the first coordinate is zero and therefore, c_1 in (3.83) has to be zero. Thus, (3.78) reduces to

$$A^k\boldsymbol{v} = \lambda_2^k \left(c_2\boldsymbol{v}_2 + c_3 \left(\frac{\lambda_3}{\lambda_2}\right)^k \boldsymbol{v}_3 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_2}\right)^k \boldsymbol{v}_n \right).$$

This makes the iteration to converge to λ_2, which is the next dominant eigenvalue. □

Remark 3.73. It is important that we understand the hypothesis **(H3)** on the initial guess $\boldsymbol{x}^{(0)}$ correctly. Note that **(H3)** says that the coefficient of $\boldsymbol{v}_1$ (which was denoted by c_1) should be non-zero when $\boldsymbol{x}^{(0)}$ is expressed as

$$\boldsymbol{x}^{(0)} = c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{v}_n.$$

Note that the coefficients $c_1, c_2, \cdots, c_n$ are unique as $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n$ is a basis for $\mathbb{R}^n$.

For such a choice of $\boldsymbol{x}^{(0)}$, it may happen that the first coordinate may be zero. That is, if $\boldsymbol{x}^{(0)}$ is written in coordinate form as $\boldsymbol{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)})^T$, it is possible that $x_1^{(0)} = 0$ and $c_1 \neq 0$.

Thus, it is not necessary that the power method will converge to the second dominant eigenvalue if the first coordinate of the initial guess is zero. However, we may expect this to happen if $c_1 = 0$. The following example illustrates this fact. □

Example 3.74. Consider the matrix

$$A = \begin{bmatrix} 91.4 & -22.0 & -44.8000 \\ 175.2 & -41.0 & -86.4 \\ 105.2 & -26.0 & -51.4000 \end{bmatrix}.$$

The eigenvalues of this matrix are $\lambda_1 = -5$, $\lambda_2 = 3$ and $\lambda_3 = 1$. The corresponding eigenvectors are $\boldsymbol{v}_1 = (3, 5, 4)^T$, $\boldsymbol{v}_2 = (2, 6, 1)^T$ and $\boldsymbol{v}_3 = (1, -2, 3)^T$.

Note that the matrix A satisfies the hypothesis **(H1)** since -5 is the unique dominant eigenvalue and it is also a simple eigenvalue. The matrix A satisfies the hypothesis **(H2)** as all eigenvalues are distinct and hence eigevectors form a basis for $\mathbb{R}^3$. Thus the fate of the power method iterates depends solely on the choice of the initial guess $\boldsymbol{x}^{(0)}$ and whether it satisfies the hypothesis **(H3)**

- Let us take the initial guess $\boldsymbol{x}^{(0)} = (1, 0.5, 0.25)^T$. Note that $c_1 \neq 0$ for this initial guess. Thus the initial guess satisfies the hypothesis **(H3)**. Therefore by the theorem on Power method (Theorem 3.64), the iterative sequences generated by power method converges to the dominant eigenvalue $\lambda_1 = -5$ and the corresponding eigenvector (with a scalar multiple) $\frac{1}{5}\boldsymbol{v}_1$.

- Let us take the initial guess $\boldsymbol{x}^{(0)} = (0, 0.5, 0.25)^T$. Note that $c_1 \neq 0$ for this initial guess. Thus the initial guess satisfies the hypothesis **(H3)**. Therefore by the theorem on Power method (Theorem 3.64), the iterative sequences generated by power method converges to the dominant eigenvalue $\lambda_1 = -5$ and the corresponding eigenvector (with a scalar multiple) $\frac{1}{5}\boldsymbol{v}_1$. Compare this with Example 3.72. In the present case the first coordinate of the initial guess vector is zero, just as in Example 3.72. In Example 3.72 the power method iterate converged to the second dominant eigenvalue and the corresponding eigenvector, which does not happen in the present case. The reason is that in the Example 3.72, $c_1 = 0$ for the initial guess chosen, but in the current example $c_1 \neq 0$. □

3.5.2 Gerschgorin's Theorem

An important tool in eigenvalue approximation is the ability to localize the eigenvalues, and the most important tool in eigenvalue localization is Gerschgorin's theorem. Gerschgorin's Circle theorem helps us in localization of eigenvalues of a matrix. This theorem explicitly constructs n disks in the complex plane with centers at the diagonal elements of the matrix; and all the eigenvalues of the matrix lie in the union of these disks.

Theorem 3.75 (Gerschgorin's Circle Theorem). *Let A be an $n \times n$ matrix. For each $k = 1, 2, \cdots, n$, define ρ_k by*

$$\rho_k = \sum_{\substack{j=1 \\ j \neq k}}^{n} |a_{kj}|,$$

and D_k denotes the closed disk in the complex plane with centre a_{kk} and radius ρ_k, i.e.,

$$D_k = \big\{ z \in \mathbb{C} \, : \, |z - a_{kk}| \leq \rho_k \big\}.$$

(1) *Each eigenvalue of A lies in one of the disks D_k. That is, no eigenvalue of A lies in $\mathbb{C} \setminus \cup_{k=1}^{n} D_k$.*

(2) *Suppose that among the disks $D_1, D_2, \cdots, D_n$, there is a collection of m disks whose union (denoted by R_1) is disjoint from the union of the rest of the $n-m$ disks (denoted by R_2). Then exactly m eigenvalues lie in R_1 and $n-m$ eigenvalues lie in R_2 (here each eigenvalue is counted as many times as its algebraic multiplicity).*

Proof. We will prove only (i) as it is easy, and the proving (ii) is beyond the scope of this course.

Let λ be an eigenvalue of A. Then there exists a $\boldsymbol{v} = (v_1, v_2, \cdots, v_n) \in \mathbb{R}^n$ and $\boldsymbol{v} \neq \boldsymbol{0}$ such that

$$A\boldsymbol{v} = \lambda \boldsymbol{v} \tag{3.95}$$

Let $1 \leq r \leq n$ be such that $|v_r| = \max\{|v_1|, |v_2|, \cdots, |v_n|\}$. The r^{th} equation of the system of equations (3.95) is given by (actually, of $A\boldsymbol{v} - \lambda \boldsymbol{v} = \boldsymbol{0}$)

$$a_{r1}v_1 + \cdots + a_{r,r-1}v_{r-1} + (a_{rr} - \lambda)v_r + a_{r,r+1}v_{r+1} + \cdots + a_{rn}v_n = 0$$

From the last equation, we get

$$\lambda - a_{rr} = \frac{v_1}{v_r}\, a_{r1} + \cdots + \frac{v_{r-1}}{v_r}\, a_{r,r-1} + \frac{v_{r+1}}{v_r}\, a_{r,r+1} + \cdots + \frac{v_n}{v_r}\, a_{rn} \tag{3.96}$$

Taking modulus on both sides of the equation (3.96), and using the triangle inequality $|a+b| \leq |a| + |b|$ repeatedly we get

$$|\lambda - a_{rr}| \leq \frac{|v_1|}{|v_r|}\, |a_{r1}| + \cdots + \frac{|v_{r-1}|}{|v_r|}\, |a_{r,r-1}| + \frac{|v_{r+1}|}{|v_r|}\, |a_{r,r+1}| + \cdots + \frac{|v_n|}{|v_r|}\, |a_{rn}| \tag{3.97}$$

In view of the choice of r, the components of the vector $\boldsymbol{v}$ satisfy $\frac{|v_s|}{|v_r|} \leq 1$. The last equation (3.97) becomes

$$|\lambda - a_{rr}| \leq |a_{r1}| + \cdots + |a_{r,r-1}| + |a_{r,r+1}| + \cdots + |a_{rn}| \tag{3.98}$$

Observe that the right hand side of the inequality (3.98) is ρ_r. This proves that $\lambda \in D_r$.□

Example 3.76. For the matrix

$$\begin{pmatrix} 4 & 1 & 1 \\ 0 & 2 & 1 \\ -2 & 0 & 9 \end{pmatrix},$$

the **Gerschgorin's disks** are given by

$$D_1 = \big\{z \in \mathbb{C} : |z-4| \leq 2\big\}, \quad D_2 = \big\{z \in \mathbb{C} : |z-2| \leq 1\big\}, \quad D_3 = \big\{z \in \mathbb{C} : |z-9| \leq 2\big\}.$$

Draw a picture of these disks and observe that D_3 neither intersects D_1 nor D_2. By (ii) of Theorem 3.75, D_3 has one eigenvalue and $D_1 \cup D_2$ has two eigenvalues counting multiplicities. Note that the eigenvalues are approximately $4.6318,\ 1.8828 \in D_1 \cup D_2$ and $8.4853 \in D_3$. □

Remark 3.77. Gerschgorin's circle theorem is helpful in finding bound for eigenvalues. For the matrix in Example 3.76, any number z in D_1 satisfies $|z| \leq 6$. Similarly any number z in D_2 satisfies $|z| \leq 3$, and any number z in D_3 satisfies $|z| \leq 11$. Since any eigenvalue λ lies in one of three disks, we can conclude that $|\lambda| \leq 11$. □

Remark 3.78. The main disadvantage of the power method discussed in Section 3.5.1 is that if a given matrix has more than one dominant eigenvalues, then the method may not converge. So, for a given matrix, we do not know whether the power method will converge or not. Also, as the power method is reasonably slow (see Example 3.74 for an illustration) we may have to perform reasonably large number of iterations to come to know that the method is not actually converging.

Thus, a tool to find out whether the given matrix has a unique dominant eigenvalue or not is highly desirable. The Gerschgorin theorem 3.75 can sometime be used to see if power method can be used for a given matrix. For instance, in Example 3.76, we see that the power method can be used to obtain an approximation to the dominant eigenvalue.□

Since the matrices A and its transpose (denoted by A^T) have same eigenvalues, we can apply Gerschgorin Circle Theorem to A^T and conclude the following corollary.

Corollary 3.79. *Let A be an $n \times n$ matrix. For each $k = 1, 2, \cdots, n$, define τ_k by*

$$\tau_k = \sum_{\substack{j=1 \\ j \neq k}}^{n} |a_{jk}|,$$

and B_k denotes the closed disk in the complex plane with centre a_{kk} and radius τ_k. That is,

$$B_k = \big\{z \in \mathbb{C} : |z - a_{kk}| \leq \tau_k\big\}.$$

(1) *Each eigenvalue of A lies in one of the disks B_k. That is, no eigenvalue of A lies in $\mathbb{C} \setminus \cup_{k=1}^{n} B_k$.*

(2) *Suppose that among the disks $B_1, B_2, \cdots, B_n$, there is a collection of m disks whose union (denoted by C_1) is disjoint from the union of the rest of the $n-m$ disks (denoted by C_2). Then exactly m eigenvalues lie in C_1 and $n-m$ eigenvalues lie in C_2 (here each eigenvalue is counted as many times as its algebraic multiplicity).* □

Remark 3.80. Advantage of the above corollary is the following: Let A be an $n \times n$ matrix. Let R denote the region defined by $R = \cup_{k=1}^{n} D_k$ where D_k are as given by Theorem 3.75. Similarly, let C denote the region defined by $C = \cup_{k=1}^{n} B_k$ where B_k are as given by Corollary 3.79. It may happen that $C \subset R$, in which case we will get better estimate for eigenvalues of A. It may not happen for every matrix A. However whenever such a thing happens, we get better bounds for eigenvalues of A. Such bounds which we obtain using both informations R and C may be called **optimum bounds**. Let us illustrate this with two examples. □

Example 3.81. For the matrix

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

the ρ_k, τ_k (in the notations of Theorem 3.75 and Corollary 3.79) are given by

$$\rho_1 = 1,\ \rho_2 = 2,\ \rho_3 = 3, \quad \text{and} \quad \tau_1 = 0,\ \tau_2 = 2,\ \tau_3 = 3$$

Now compute

$$R = \cup_{k=1}^{n} D_k = \{z : |z-3| \leq 1\} \cup \{z : |z+2| \leq 2\} \cup \{z : |z+3| \leq 3\}$$
$$C = \cup_{k=1}^{n} B_k = \{z : |z-3| \leq 0\} \cup \{z : |z+2| \leq 2\} \cup \{z : |z+3| \leq 3\}$$

and clearly $C \subset R$. Hence bounds obtained using C are optimum bounds. Draw both regions C and R. □

CHAPTER 4

Nonlinear Equations

One of the most frequently occurring problems in practical applications is to find the roots of equations of the form

$$f(x) = 0, \tag{4.1}$$

where $f : [a, b] \to \mathbb{R}$ is a given nonlinear function. It is well-known that not all nonlinear equations can be solved explicitly to obtain the exact value of the roots and hence, we need to look for methods to compute approximate value of the roots. By approximate root to (4.1), we mean a point $x^* \in \mathbb{R}$ for which the value $f(x)$ is very near to zero, ie., $f(x^*) \approx 0$.

In this chapter, we introduce various iterative methods to obtain an approximation to a real root of equations of the form (4.1) with f being a continuous nonlinear function. The key idea in approximating the real roots of (4.1) consists of two steps:

(1) **Starting Step:** Take one or more points (arbitrarily or following a procedure) $x_i \in [a, b]$ $(i = 0, 1, \cdots, m,\ m \in \mathbb{N})$ around a root of the equation (4.1). Consider x_m as an approximation to the root of (4.1).

(2) **Improving Step:** If x_m is not 'close' to the required root, then devise a procedure to obtain another point x_{m+1} that is 'more close' to the root than x_m.

Repeat this step until we obtain a point x_n $(n \geq m)$ which is 'sufficiently close' to the required root.

This process of improving the approximation to the root is called the **iterative process** (or **iterative procedure**), and such methods are called **iterative methods**. In an iterative method, we obtain a sequence of numbers $\{x_n\}$ which is expected to converge to the root of (4.1) as $n \to \infty$. We investigate conditions on the function f, its domain, and co-domain under which the sequence of iterates converge to a solution of the equation (4.1).

We classify the iterative methods discussed in this chapter into two types, namely,

(1) **Closed Domain Methods:** As the starting step, these methods need the knowledge of an interval in which at least one root of the given nonlinear equation exists. Further iterations include the restriction of this interval to smaller intervals in which root lies. These methods are also called **bracketing methods**.

(2) **Open Domain Methods:** The x_i's mentioned in the starting step above are chosen arbitrarily and the consecutive iterations are based on a formula.

In the case of closed domain methods, the difficult part is to locate an interval containing a root. But, once this is done, the iterative sequence will surely converge as we will see in Section 4.1. In this section, we discuss two closed domain methods, namely, the bisection method and the regula-falsi method. In the case of open domain methods, it is easy at the starting step as we can choose the x_i's arbitrarily. But, it is not necessary that the sequence converges. We discuss secant method, Newton-Raphson method and fixed point method in Section 4.3, which are some of the open domain methods.

4.1 Closed Domain Methods

The idea behind the closed domain methods is to start with an interval (denoted by $[a_0, b_0]$) in which there exists at least one root of the given nonlinear equations and then reduce the length of this interval iteratively with the condition that there is at least one root of the equation at each iteration.

Note that the initial interval $[a_0, b_0]$ can be obtained using the intermediate value theorem (as we always assume that the nonlinear function f is continuous) by checking the condition that

$$f(a_0)f(b_0) < 0.$$

That is, $f(a_0)$ and $f(b_0)$ are of opposite sign. The closed domain methods differ from each other only by the way they go on reducing the length of this interval at each iteration.

In the following subsections we discuss two closed domain methods, namely, (i) the bisection method and (ii) the regula-falsi method.

4.1.1 Bisection Method

The most simple way of reducing the length of the interval is to sub-divide the interval into two equal parts and then take the sub-interval that contains a root of the equation and discard the other part of the interval. This method is called the **bisection method**. Let us explain the procedure of generating the first iteration of this method.

Step 1: Define x_1 to be the mid-point of the interval $[a_0, b_0]$. That is,

$$x_1 = \frac{a_0 + b_0}{2}.$$

Step 2: Now, exactly one of the following two statements hold.

(1) x_1 solves the nonlinear equation. That is, $f(x_1) = 0$.

(2) Either $f(a_0)f(x_1) < 0$ or $f(b_0)f(x_1) < 0$.

If case (1) above holds, then x_1 is a required root of the given equation $f(x) = 0$ and therefore we stop the iterative procedure. If $f(x_1) \neq 0$, then case (2) holds as $f(a_0)$ and $f(b_0)$ are already of opposite signs. In this case, we define a subinterval $[a_1, b_1]$ of $[a_0, b_0]$ as follows.

$$[a_1, b_1] = \begin{cases} [a_0, x_1], & \text{if } f(a_0)f(x_1) < 0, \\ [x_1, b_0], & \text{if } f(b_0)f(x_1) < 0. \end{cases}$$

The outcome of the first iteration of the bisection method is the interval $[a_1, b_1]$ and the first member of the corresponding iterative sequence is the real number x_1. Observe that

- the length of the interval $[a_1, b_1]$ is exactly half of the length of $[a_0, b_0]$ and
- $[a_1, b_1]$ has at least one root of the nonlinear equation $f(x) = 0$.

Similarly, we can obtain x_2 and $[a_2, b_2]$ as the result of the second iteration and so on.

We now present the algorithm for the bisection method.

Hypothesis: There exists an interval $[a_0, b_0]$ such that the function $f : [a_0, b_0] \to \mathbb{R}$ is continuous, and the numbers $f(a_0)$ and $f(b_0)$ have opposite signs.

Algorithm:

Step 1: For $n = 0, 1, 2, \cdots$, define the iterative sequence of the bisection method as

$$x_{n+1} = \frac{a_n + b_n}{2},$$

which is the midpoint of the interval $[a_n, b_n]$.

Step 2: One of the following two cases hold.

(1) x_{n+1} solves the nonlinear equation. That is, $f(x_{n+1}) = 0$.
(2) Either $f(a_n)f(x_{n+1}) < 0$ or $f(b_n)f(x_{n+1}) < 0$.

Define the subinterval $[a_{n+1}, b_{n+1}]$ of $[a_n, b_n]$ as follows.

$$[a_{n+1}, b_{n+1}] = \begin{cases} [a_n, x_{n+1}], & \text{if } f(a_n)f(x_{n+1}) < 0, \\ [x_{n+1}, b_n], & \text{if } f(b_n)f(x_{n+1}) < 0. \end{cases}$$

Step 3: Stop the iteration if one of the following happens:

- the case (1) in step 2 holds. Then declare the value of x_{n+1} as the required root.
- $(b_{n+1} - a_{n+1})$ is sufficiently small (less than a pre-assigned positive quantity). Then declare the value of x_{n+2} as the required root up to the desired accuracy.

If any of the above stopping criteria does not hold, then go to step 1. Continue this process till one of the above two stopping criteria is fulfilled. □

Remark 4.1. In practice, one may also use any of the stopping criteria listed in section 4.2, either single or multiple criteria. □

Assuming that, for each $n = 1, 2, \cdots$, the number x_n is not a solution of the nonlinear equation $f(x) = 0$, we get a sequence of real numbers $\{x_n\}$. The question is whether this sequence converges to a solution of the nonlinear equation $f(x) = 0$. We now discuss the error estimate and convergence of the iterative sequence generated by the bisection method.

Theorem 4.2 (Convergence and Error Estimate of Bisection Method).
Hypothesis: *Let $f : [a_0, b_0] \to \mathbb{R}$ be a continuous function such that the numbers $f(a_0)$ and $f(b_0)$ have opposite signs.*

Conclusion:

- *There exists an $r \in (a_0, b_0)$ such that $f(r) = 0$ and the iterative sequence $\{x_n\}$ of the bisection method converges to r.*
- *For each $n = 0, 1, 2, \cdots$, we have the following error estimate*

$$|x_{n+1} - r| \leq \left(\frac{1}{2}\right)^{n+1} (b - a). \tag{4.2}$$

Proof: It directly follows from the construction of the intervals $[a_n, b_n]$ that

$$b_n - a_n = \frac{1}{2}(b_{n-1} - a_{n-1}) = \cdots = \left(\frac{1}{2}\right)^{n} (b_0 - a_0).$$

As a consequence, we get

$$\lim_{n\to\infty} (b_n - a_n) = 0.$$

By algebra of limits of sequences, we get

$$\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n.$$

Since for each $n = 0, 1, 2, \cdots$, the number x_{n+1} is the mid-point of the interval $[a_n, b_n]$, we also have

$$a_n < x_{n+1} < b_n.$$

Now by sandwich theorem for sequences, we conclude that the sequence $\{x_n\}$ of mid-points also converges to the same limit as the sequences of end-points. Thus we have

$$\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n = \lim_{n\to\infty} x_n = r \text{ (say)}. \tag{4.3}$$

Since for each $n = 0, 1, 2, \cdots$, we have $f(a_n)f(b_n) < 0$, applying limits on both sides of the inequality and using the continuity of f, we get

$$f(r)f(r) \leq 0, \tag{4.4}$$

from which we conclude that $f(r) = 0$. That is, the sequence of mid-points $\{x_n\}$ defined by the bisection method converges to a root of the nonlinear equation $f(x) = 0$.

Since the sequences $\{a_n\}$ and $\{b_n\}$ are non-decreasing and non-increasing, respectively, for each $n = 0, 1, 2, \cdots$, we have $r \in [a_n, b_n]$. Also, x_{n+1} is the mid-point of the interval $[a_n, b_n]$. Therefore, we have

$$|x_{n+1} - r| \leq \frac{1}{2}(b_n - a_n) = \cdots = \left(\frac{1}{2}\right)^{n+1} (b_0 - a_0),$$

which is the required estimate. □

Corollary 4.3. *Let $\epsilon > 0$ be given. Let f satisfy the hypothesis of bisection method with the interval $[a_0, b_0]$. Let x_n be as in the bisection method, and r be the root of the nonlinear equation $f(x) = 0$ to which bisection method converges. Then $|x_n - r| \leq \epsilon$ whenever n satisfies*

$$n \geq \frac{\log(b_0 - a_0) - \log \epsilon}{\log 2} \tag{4.5}$$

Proof : By the error estimate of bisection method given by (4.2), we are sure that

$$|x_n - r| \leq \epsilon,$$

whenever n is such that

$$\left(\frac{1}{2}\right)^n (b - a) \leq \epsilon.$$

By taking logarithm on both sides of the last inequality, we get the desired estimate on n. □

Remark 4.4. The Corollary 4.3 tells us that if we want an approximation x_n to the root r of the given equation such that the absolute error is less than a pre-assigned positive quantity, then we have to perform n iterations, where n is the least integer that satisfies the inequality (4.5). It is interesting to observe that to obtain this n, we don't need to know the root r. □

Example 4.5. Let us find an approximate solution to the nonlinear equation

$$\sin x + x^2 - 1 = 0$$

using bisection method so that the resultant absolute error is at most $\epsilon = 0.125$.

To apply Bisection method, we must choose an interval $[a_0, b_0]$ such that the function

$$f(x) = \sin x + x^2 - 1$$

satisfies the hypothesis of bisection method. Note that f satisfies hypothesis of bisection on the interval $[0, 1]$. In order to achieve the required accuracy, we should first decide how many iterations are needed. The inequality (4.5), says that required accuracy is achieved provided n satisfies

$$n \geq \frac{\log(1) - \log(0.125)}{\log 2} = 3$$

Thus we have to compute x_3. We will do it now.

Iteration 1: We have $a_0 = 0$, $b_0 = 1$. Thus $x_1 = 0.5$. Since,

$$f(x_1) = -0.27 < 0, \ f(0) < 0, \text{ and } f(1) > 0,$$

we take $[a_1, b_1] = [x_1, b_0] = [0.5, 1]$.

Iteration 2: The mid-point of $[0.5, 1]$ is $x_2 = 0.75$. Since

$$f(x_2) = 0.24 > 0, \ f(0.5) < 0, \text{ and } f(1) > 0,$$

we take $[a_2, b_2] = [a_1, x_2] = [0.5, 0.75]$.

Iteration 3: The mid-point of $[0.5, 0.75]$ is $x_3 = 0.625$. Since

$$f(x_3) = -0.024 < 0, \ f(0.5) < 0 \text{ and } f(0.75) > 0,$$

we take $[a_3, b_3] = [x_3, b_2] = [0.625, 0.75]$.

As suggested by the formula (4.5), we stop the iteration here and take the approximate root of the given equation as the mid-point of the interval $[a_3, b_3]$, which is $x_4 \approx 0.6875$. Note that the true value is $r \approx 0.636733$. The absolute error is 0.05 which is much less than the required accuracy of 0.125. □

Remark 4.6 (Comments on Bisection method).

(1) Note that the mid-point of an interval $[a, b]$ is precisely the $x-$*coordinate* of the point of intersection of the line joining the points $(a, \text{sgn}(f(a)))$ and $(b, \text{sgn}(f(b)))$ with the x-axis.

(2) Let f, and the interval $[a_0, b_0]$ satisfy the hypothesis of bisection method. Even if the nonlinear equation $f(x) = 0$ has more than one solution in the interval $[a, b]$, the bisection method chooses the root that it tries to approximate. In other words, once we fix the initial interval $[a_0, b_0]$, the bisection method takes over and we cannot control it to find some specific root than what it chooses to find.

(3) Given a function f, choosing an interval $[a, b]$ such that $f(a)f(b) < 0$ is crucial to bisection method. There is no general procedure to find such an interval. This is one of the main drawbacks of bisection method.

(4) Bisection method cannot be used to find zero of functions for which graph touches x-axis but does not cross x-axis.

(5) There is a misconception about bisection method's order of convergence (see Definition 1.42), which is claimed to be 1. However there is no proof of $|x_{n+1} - r| \leq c|x_n - r|$. If the sequence $\{x_n\}$ converges linearly to r, then we get

$$|x_{n+1} - r| \leq c^{n+1}|x_0 - r|.$$

In the case of bisection method, we have this inequality with $c = 1/2$, which is only a necessary condition for the convergence to be linear but not a sufficient condition. □

4.1.2 Regula-falsi Method

The regula-falsi method is similar to the bisection method. Although the bisection method (discussed in the previous subsection) always converges to the solution, the convergence is very slow, especially when the length of the initial interval $[a_0, b_0]$ is very large and the equation has the root very close to the one of the end points. This is because, at every iteration, we are subdividing the interval $[a_n, b_n]$ into two equal parts and taking the mid-point as x_{n+1} (the $(n+1)^{\text{th}}$ member of the iterative sequence). Therefore, it takes several iterations to reduce the length of the interval to a very small number, and as a consequence the distance between the root and x_{n+1}.

The **regula-falsi method** differs from the bisection method only in the choice of x_{n+1} in the interval $[a_n, b_n]$ for each $n = 0, 1, 2, \cdots$. Instead of taking the midpoint of the interval, we now take the x-coordinate of the point of intersection of the line joining the points $(a_n, f(a_n))$ and $(b_n, f(b_n))$ with the x-axis. Let us now explain the procedure of generating the first iteration of this method.

Step 1: Assume the hypothesis of the bisection method and let $[a_0, b_0]$ be the initial interval. The line joining the points $(a_0, f(a_0))$ and $(b_0, f(b_0))$ is given by

$$y = f(a_0) + \frac{f(b_0) - f(a_0)}{b_0 - a_0}(x - a_0),$$

The first member x_1 of the regula-falsi iterative sequence is the x-coordinate of the point of intersection of the above line with the x-axis. Therefore, x_1 satisfies the equation

$$f(a_0) + \frac{f(b_0) - f(a_0)}{b_0 - a_0}(x_1 - a_0) = 0$$

and is given by

$$x_1 = a_0 - f(a_0)\frac{b_0 - a_0}{f(b_0) - f(a_0)},$$

which can also be written as

$$x_1 = \frac{a_0 f(b_0) - b_0 f(a_0)}{f(b_0) - f(a_0)}.$$

Step 2: Now, exactly one of the following two statements hold.

(1) x_1 solves the nonlinear equation. That is, $f(x_1) = 0$.

(2) Either $f(a_0)f(x_1) < 0$ or $f(b_0)f(x_1) < 0$.

If case (1) above holds, then x_1 is a required root of the given equation $f(x) = 0$ and therefore we stop the iterative procedure. If $f(x_1) \neq 0$, then case (2) holds as $f(a_0)$ and $f(b_0)$ are already of opposite signs. We now define a subinterval $[a_1, b_1]$ of $[a_0, b_0]$ as follows.

$$[a_1, b_1] = \begin{cases} [a_0, x_1], & \text{if } f(a_0)f(x_1) < 0, \\ [x_1, b_0], & \text{if } f(b_0)f(x_1) < 0. \end{cases}$$

The outcome of the first iteration of the regula-falsi method is the interval $[a_1, b_1]$ and the first member of the corresponding iterative sequence is the real number x_1. Observe that

- the length of the interval $[a_1, b_1]$ may be (although not always) much less than half of the length of $[a_0, b_0]$ and
- $[a_1, b_1]$ has at least one root of the nonlinear equation $f(x) = 0$.

We now summarize the regula-falsi method.

Hypothesis: Same as bisection method.

Algorithm:

Step 1: For $n = 0, 1, 2, \cdots$, define the iterative sequence of the regula-falsi method as

$$x_{n+1} = a_n - f(a_n)\frac{b_n - a_n}{f(b_n) - f(a_n)} = \frac{a_n f(b_n) - b_n f(a_n)}{f(b_n) - f(a_n)}, \tag{4.6}$$

which is the x-coordinate of the point of intersection of the line joining the points $(a_n, f(a_n))$ and $(b_n, f(b_n))$ (obtained at the n^{th} iteration) with the x-axis.

Step 2: One of the following two cases hold.

(1) x_{n+1} solves the nonlinear equation. That is, $f(x_{n+1}) = 0$.
(2) Either $f(a_n)f(x_{n+1}) < 0$ or $f(b_n)f(x_{n+1}) < 0$.

Define the subinterval $[a_{n+1}, b_{n+1}]$ of $[a_n, b_n]$ as follows.

$$[a_{n+1}, b_{n+1}] = \begin{cases} [a_n, x_{n+1}], & \text{if } f(a_n)f(x_{n+1}) < 0, \\ [x_{n+1}, b_n], & \text{if } f(b_n)f(x_{n+1}) < 0. \end{cases}$$

Step 3: Stop the iteration if the case (1) in step 2 holds and declare the value of x_{n+1} as the required root. Otherwise go to step 1.

Continue this process till a desired accuracy is achieved. □

Remark 4.7 (Stopping criteria).

Unlike in the case of bisection method, there is no clear way of stopping the iteration of regula-falsi method as the length of the interval $[a_n, b_n]$ obtained at the $(n+1)^{\text{th}}$ iteration may not converge to zero as $n \to \infty$. This situation occurs especially when the function f is concave or convex in the interval $[a_0, b_0]$ as illustrated in Example 4.8. □

Assuming that, for each $n = 1, 2, \cdots$, the number x_n is not a solution of the nonlinear equation $f(x) = 0$, we get a sequence of real numbers $\{x_n\}$. The question is whether this sequence converges to a root of the nonlinear equation $f(x) = 0$. Before addressing this question, let us consolidate the information that we have so far.

(1) The sequence of left end points of the intervals $[a_n, b_n]$ is a non-decreasing sequence that is bounded above by b. That is,

$$a_0 \leq a_1 \leq \cdots \leq a_n \cdots \leq b.$$

Hence the sequence $\{a_n\}$ has a limit, *i.e.,* $\lim_{n\to\infty} a_n$ exists.

(2) The sequence of right end points of the intervals $[a_n, b_n]$ is a non-increasing sequence that is bounded below by a. That is,

$$b_0 \geq b_1 \geq \cdots \geq b_n \cdots \geq a.$$

Hence the sequence $\{b_n\}$ has a limit, *i.e.,* $\lim_{n\to\infty} b_n$ exists.

(3) Since $a_n < b_n$ for all $n = 0, 1, 2, \cdots$, we conclude that

$$\lim_{n\to\infty} a_n \leq \lim_{n\to\infty} b_n.$$

If the lengths of the intervals obtained in regula-falsi method tend to zero, then we have

$$\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n,$$

in which case, we also have by sandwich theorem

$$\lim_{n\to\infty} a_n = \lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} b_n,$$

as $a_n < x_{n+1} < b_n$ for all $n = 0, 1, 2, \cdots$. In this case, the common limit will be a root of the nonlinear equation, as is the case with bisection method. □

Unfortunately, it may happen that the lengths of the subintervals chosen by regula-falsi method do not go to zero. In other words, if $a_n \to \alpha$ and $b_n \to \beta$, then it may happen that $\alpha < \beta$. Let us illustrate this case by the following example.

Example 4.8. Consider the nonlinear equation

$$e^x - 2 = 0.$$

Note that the function $f : \mathbb{R} \to \mathbb{R}$ defined by $f(x) = e^x - 2$ satisfies the hypothesis of the regula-falsi method on the interval $[0, 1]$. Let us start the iteration with the initial interval $[a_0, b_0] = [0, 1]$.

Iteration 1: Using the formula (4.6), we get

$$x_1 = \frac{1}{e-1}.$$

Since $f(x_1) \approx -0.21043$, the subinterval chosen by regula-falsi method is the interval $[a_1, b_1] = [x_1, 1]$.

Iteration 2: We have $x_2 \approx 0.67669$. Since $f(x_2) \approx -0.032645$, the subinterval chosen by regula falsi method is the interval $[a_2, b_2] = [x_2, 1]$.

By continuing these computations, we see that the subinterval at every iteration chosen by regula-falsi method is the right half of the interval defined in the previous iteration. In

other words, for $n = 1, 2, \cdots$, we have $[a_n, b_n] = [x_n, 1]$. This is clearly seen geometrically as the function f is convex. A similar situation occurs when the given function f is concave (at least in the initial interval $[a_0, b_0]$). We checked this to be true for $n = 1, 2$ and a proof for a general n is left as an exercise.

Note in this example that $\beta = 1$ and $\alpha \leq r \neq 1$, where r is the root of the given equation $f(x) = 0$ to which the regula-falsi method is expected to converge. □

The last example ruled out any hopes of proving that $\alpha = \beta$. However, it is true that the sequence $\{x_n\}$ converges to a root of the nonlinear equation $f(x) = 0$, and as can be expected, the proof is by contradiction argument. Let us now state the theorem on regula falsi method and prove it.

Theorem 4.9 (Convergence of Regula-falsi method).

Hypothesis: *Let $f : [a_0, b_0] \to \mathbb{R}$ be a continuous function such that the numbers $f(a_0)$ and $f(b_0)$ have opposite signs.*

Conclusion: *If $\{x_n\}$ is the iterative sequence defined by regula-falsi method. Then the sequence $\{x_n\}$ converges to an $r \in (a_0, b_0)$ such that $f(r) = 0$.*

Proof: Let us start by analyzing conditions under which "the sequence $\{x_n\}$ does not converge to a root of the nonlinear equation $f(x) = 0$". This exactly happens when one of the following two situations occur.

(1) The sequence $\{x_n\}$ is not a convergent sequence.

(2) The sequence $\{x_n\}$ converges but its limit is not a root of the nonlinear equation $f(x) = 0$.

We are going to show that none of the above situations can occur, which will then prove the required convergence result.

Step 1: Since the sequence $\{x_n\}$ is a sequence from the interval $[a_0, b_0]$, it is bounded. By Bolzano-Weierstrass theorem, there exists a subsequence $\{x_{n_k}\}$ of $\{x_n\}$ and a number $p \in [a, b]$ such that

$$\lim_{k \to \infty} x_{n_k} = p.$$

We will show that $f(p) = 0$.

Step 2: Suppose that $f(p) \neq 0$. Without loss of generality, assume that $f(p) < 0$. Since f is a continuous function, there exists a $\delta > 0$ such that

$$f(x) < 0 \quad \text{for all } x \in [p - \delta, p + \delta].$$

Since the subsequence $\{x_{n_k}\}$ converges to p, there exists a K such that for every $k \geq K$, we have

$$p - \delta \leq x_{n_k} \leq p + \delta.$$

Note that for each $k \in \mathbb{N}$, $x_{n_k} \in \{a_{n_k}, b_{n_k}\}$. Consider the sets S_a and S_b defined by

$$\begin{aligned} S_a &= \{k \in \mathbb{N} : k \geq K,\, x_{n_k} = a_{n_k}\}, \\ S_b &= \{k \in \mathbb{N} :, k \geq K,\, x_{n_k} = b_{n_k}\}. \end{aligned} \tag{4.7}$$

If both the sets S_a and S_b are non-empty, then we get a contradiction (to what?) immediately (why?). Thus, we may assume that one of the two sets S_a and S_b is empty. Without loss of generality assume that $S_a = \emptyset$ and $S_b \neq \emptyset$. Consequently, $S_b = \{k \in \mathbb{N} : k \geq K\}$.

Claim:

(1) The subsequence $\{b_{n_k}\}_{k \geq K}$ lies to the right of the point p. That is,

$$p < b_{n_k} \text{ for all } k \geq K.$$

(2) The tail of the sequence $\{b_n\}_{n \geq n_K}$ lies in the interval $[p, p + \delta]$.

$$p + \delta \geq b_{n_K} > b_{n_K+1} > b_{n_K+2} > \cdots > p \tag{4.8}$$

As a consequence, $a_n = a_{n_K}$ for all $n \geq n_K$.

(3) We have

$$x_n = b_n \quad \text{for all } n \geq n_K$$

Proof of Claim: (1). If there exists a $K' > K$ such that $b_{n_{K'}} < p$, then p cannot be limit of x_{n_k} as these points have to be one of the end points of subintervals selected by regula falsi method. Thus (1) is proved.

Proof of Claim: (2). Since $\{b_n\}$ is a non-increasing sequence, and the subsequence $\{b_{n_k}\}_{k \geq K}$ lies to the right of the point p, it follows that the sequence $\{b_n\}$ lies to the right of p and consequently, we get (4.8). Since $S_a = \emptyset$, we have $x_{n_k} = b_{n_k}$ for every $k \geq K$. Furthermore, $x_n = b_n$ for every $n \geq n_K$. Thus we also have $a_n = a_{n_K}$ for all $n \geq n_K$.

Incorporating all the information from the last claim into the formula for regula falsi iterative sequence (4.6), we get the following formula for x_{n+1} for every $n \geq n_K$

$$x_{n+1} = \frac{a_{n_K} f(x_n) - x_n f(a_{n_K})}{f(x_n) - f(a_{n_K})}. \tag{4.9}$$

Passing to the limit in the last equation as $n \to \infty$ yields

$$p = \frac{a_{n_K} f(p) - p f(a_{n_K})}{f(p) - f(a_{n_K})}.$$

Simplifying the last equation, we get

$$p f(p) = a_{n_K} f(p)$$

Since $p \neq a_{n_K}$, we must have $f(p) = 0$ which is a contradiction to our assumption that $f(p) \neq 0$. Thus we conclude that $f(p) = 0$.

Thanks to (3) of the previoius claim, we actually proved that the sequence $\{x_n\}$ converges to p, and $f(p) = 0$. This finishes the proof of the theorem. □

Example 4.10. Let us find an approximate solution to the nonlinear equation

$$\sin x + x^2 - 1 = 0$$

using regula-falsi method. Exact solution is approximately 0.637.

We choose the initial interval as $[a_0, b_0] = [0, 1]$ as done in Example 4.5.

Iteration 1: We have $a_0 = 0$, $b_0 = 1$. Thus $x_1 = 0.54304$. Since $f(x_1) = -0.18837 < 0$, $f(0) < 0$, and $f(1) > 0$, we take $[a_1, b_1] = [x_1, b_0] = [0.54304, 1]$.

Iteration 2: Using the regula-falsi formula (4.6), we get $x_2 = 0.62662$. Since $f(x_2) = -0.020937 < 0$, $f(0.54304) < 0$, and $f(1) > 0$, we take $[a_2, b_2] = [x_2, 1] = [0.62662, 1]$.

Iteration 3: Using the regula-falsi formula (4.6), we get $x_3 = 0.63568$. Since $f(x_3) = -0.0021861 < 0$, $f(0.62662) < 0$, and $f(1) > 0$, we take $[a_3, b_3] = [x_3, 1] = [0.63658, 1]$.

Iteration 4: Using the regula-falsi formula (4.6), we get $x_4 = 0.63662$.
and so on.

In this example also, we observe that the lengths of the intervals $[a_n, b_n]$ do not seem to be tending to zero as $n \to \infty$. However, we see that the sequence $\{x_n\}$ is approaching the root of the given equation. □

4.2 Stopping Criteria

The outcome of any iterative method for a given nonlinear equation is a sequence of real numbers that is expected to converges to a root of the equation. When we implement such a methods on a computer, we cannot go on computing the iterations indefinitely and needs to stop the computation at some point. It is desirable to stop computing the iterations when the x_n's are reasonably close to an exact root r for a sufficiently large n. In other words, we want to stop the computation at the n^{th} iteration when the computed value is such that

$$|x_n - r| < \epsilon$$

for some pre-assigned positive number ϵ.

In general, we don't know the root r of the given nonlinear equation to which the iterative sequence is converging. Therefore, we have no idea of when to stop the iteration as we have seen in the case of regula-falsi method. In fact, this situation will be there for any open domain methods discussed in the next section. An alternate way is to look for some criteria that does not use the knowledge of the root r, but gives a rough idea of how close we are to this root. Such a criteria is called the **stopping criteria**. We now list some of the commonly used stopping criteria for iterative methods for nonlinear equations.

Stopping Criterion 1: Fix a $K \in \mathbb{N}$, and ask the iteration to stop after finding x_K. This criterion is borne out of fatigue, as it clearly has no mathematical reason why the K fixed at the beginning of the iteration is more important than any other natural number! If we stop the computation using this criterion, we will declare x_K to be the approximate solution to the nonlinear equation $f(x) = 0$.

Stopping Criterion 2: Fix a real number $\epsilon > 0$ and a natural number N. Ask the iteration to stop after finding x_k such that

$$|x_k - x_{k-N}| < \epsilon.$$

One may interpret this stopping criterion by saying that there is 'not much' improvement improvement in the value of x_k compared to a previous value x_{k-N}. If we stop the computation using this criterion, we will declare x_k to be the approximate solution to the nonlinear equation $f(x) = 0$.

It is more convenient to take $N = 1$ in which case, we get the stopping criteria

$$|x_k - x_{k-1}| < \epsilon.$$

Stopping Criterion 3: Fix a real number $\epsilon > 0$ and a natural number N. Ask the iteration to stop after finding x_k such that

$$\left| \frac{x_k - x_{k-N}}{x_k} \right| < \epsilon.$$

If we stop the computation using this criterion, we will declare x_k to be the approximate solution to the nonlinear equation $f(x) = 0$.

As in the above case, it is convenient to take $N = 1$.

Stopping Criterion 4: Fix a real number $\epsilon > 0$ and ask the iteration to stop after finding x_k such that

$$|f(x_k)| < \epsilon.$$

If we stop the computation using this criterion, we will declare x_k to be the approximate solution to the nonlinear equation $f(x) = 0$. Sometimes the number $|f(x_k)|$ is called the **residual error** corresponding to the approximate solution x_k of the nonlinear equation $f(x) = 0$. □

In practice, one may use any of the stopping criteria listed above, either single or multiple criteria.

Remark 4.11. We can also use any of the above stopping criteria in bisection method.

4.3 Open Domain Methods

In the closed domain methods described in the previous section, we have seen that the iterative sequences always converge to a root of the nonlinear equation. However, initially to start the iteration, we need to give an interval where at least one root of the equation is known to exist. In many practical situations it may be very difficult to obtain this interval manually. Also, it will be very expensive to find this interval using a computer program as we have to adopt a trial and error algorithm for this. Therefore, it is highly desirable to device an iterative method that does not need this information and gives freedom to

choose any arbitrary starting point as done in the iterative method for linear systems in Section 3.4. This is the objective of an open domain method.

The open domain methods do not pre-suppose that a root is enclosed in an interval of type $[a, b]$. These methods always start with a set of initial guesses $x_0, x_1, \cdots, x_m$ distributed anywhere on the real line (but sufficiently close to a root of the equation) and tell us how to compute the values x_{m+1}, x_{m+2}, $\cdots$. in such a way that the resulting iterative sequence $\{x_n\}$ converges to a root of the equation.

In this section we study three open domain iterative methods for approximating a root of a given nonlinear equation.

4.3.1 Secant Method

The straightforward modification of the regula-falsi method is the well-known **secant method**. The only difference in the secant method (when compared to the regula-falsi method) is that here we do not demand the initial guesses a_0 and b_0 to be on the either side of a root. Let us now present the algorithm of the secant method.

Hypothesis: Given any initial values x_0 and x_1 (not necessarily on the either side of a root) such that $f(x_0) \neq f(x_1)$.

Algorithm:

Step 1: For $n = 1, 2, \cdots$, the iterative sequence for secant method is given by

$$x_{n+1} = \frac{f(x_n)x_{n-1} - f(x_{n-1})x_n}{f(x_n) - f(x_{n-1})}. \tag{4.10}$$

This expression can also be written as

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \tag{4.11}$$

Step 2: Choose any one of the stopping criteria (or a combination of them) discussed in Section 4.2. If this criterion is satisfied, stop the iteration. Otherwise, repeat the step 1 until the criterion is satisfied. □

Recall that x_{n+1} for each $n = 1, 2, \cdots$ given by (4.10) is the x-coordinate of the point of intersection of the secant line joining the points $(x_{n-1}, f(x_{n-1}))$ and $(x_n, f(x_n))$ with the x-axis and hence the name **secant method**.

Remark 4.12. It is evident that the secant method fails to determine x_{n+1} if we have $f(x_{n-1}) = f(x_n)$. Observe that such a situation never occurs in regula-falsi method. □

Example 4.13. Consider the equation

$$\sin x + x^2 - 1 = 0.$$

Let $x_0 = 0$, $x_1 = 1$. Then the iterations from the secant method are given by

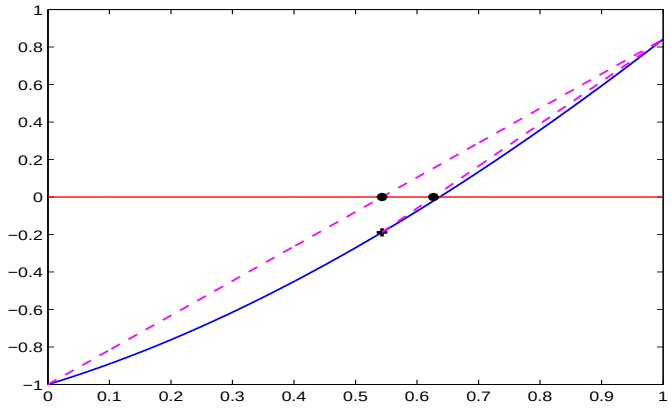

Fig. 4.1. Iterative points of secant method.

n	x_n	ϵ
2	0.543044	0.093689
3	0.626623	0.010110
4	0.637072	0.000339
5	0.636732	0.000001

Figure 4.1 shows the iterative points x_2 and x_3 in black bullet. Recall that the exact value of the root (to which the iteration seems to converge) unto 6 significant digits is $r \approx 0.636733$. Obviously, the secant method is much faster than bisection method in this example. □

We now state the convergence theorem on secant method.

Theorem 4.14 (Convergence of Secant Method).
Hypothesis:

(1) *Let* $f : \mathbb{R} \to \mathbb{R}$ *be a* $C^2(\mathbb{R})$ *function.*

(2) *Let* r *be a simple root of the nonlinear equation* $f(x) = 0$*, that is* $f'(r) \neq 0$*.*

Conclusion: *Then there exists a* $\delta > 0$ *such that for every* $x_0, x_1 \in [r - \delta, r + \delta]$*,*

(1) *the secant method iterative sequence* $\{x_n\}$ *is well-defined.*

(2) *the sequence* $\{x_n\}$ *belongs to the interval* $[r - \delta, r + \delta]$*.*

(3) $\lim_{n\to\infty} x_n = r$*.*

(4) *Further, we have*

$$\lim_{n\to\infty} \frac{|x_{n+1} - r|}{|x_n - r|^\alpha} = \left| \frac{f''(r)}{2f'(r)} \right|^{\alpha/(\alpha+1)}. \tag{4.12}$$

where $\alpha = (\sqrt{5} + 1)/2 \approx 1.62$*.*

The proof for this is omitted for this course.

Remark 4.15. The expression (4.12) implies that the order of convergence (see Definition 1.42) of the iterative sequence of secant method is 1.62. □

4.3.2 Newton-Raphson Method

In Theorem 4.14, we have seen that the secant method has more than linear order of convergence. This method can further be modified to achieve quadratic convergence. To do this, we first observe that when the iterative sequence of the secant method converges, then as n increases, we see that x_{n-1} approaches x_n. Thus, for a sufficiently large value of n, we have

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

provided f is a C^1 function. Thus, if $f(x)$ is differentiable, then on replacing in (4.11), the slope of the secant by the slope of the tangent at x_n, we get the iteration formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4.13}$$

and is called the **Newton-Raphson's Method**.

Remark 4.16 (Geometrical Motivation). If a function f is differentiable at a point x_0, then the tangent ($y = g(x)$) to the graph of f at the point $(x_0, f(x_0))$ is given by

$$g(x) = f'(x_0)(x - x_0) + f(x_0).$$

We may assume that for $x \approx x_0$, $f(x) \approx g(x)$. This can also be interpreted as

"If a function f is differentiable at a point, then the graph of f looks like a straight line, for $x \approx x_0$ on zooming".

Now, if we choose the initial guess x_0 very close to the root r of $f(x) = 0$. That is., if $r \approx x_0$, we have $g(r) \approx f(r) = 0$. This gives (approximately)

$$0 \approx f'(x_0)(r - x_0) + f(x_0).$$

Up on replacing r by x_1 and using '=' symbol instead of '≈' symbol, we get the first iteration of the Newton-Raphson's iterative formula (4.13). □

Recall in secant method, we need two initial guesses x_0 and x_1 to start the iteration. In Newton-Raphson method, we need one initial guess x_0 to start the iteration. The consecutive iteration x_1 is the x-coordinate of the point of intersection of the x-axis and the tangent line at x_0, and similarly for the other iterations. This geometrical interpretation of the Newton-Raphson method is clearly observed in Figure 4.2.

We now derive the Newton-Raphson method under the assumption that f is a C^2 function.

Let x_0 be given. The Taylor's polynomial of degree $n = 1$ with remainder is given by

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{(x - x_0)^2}{2!} f''(\xi),$$

where ξ lies between x_0 and x. When x_0 is very close to x, the last term in the above equation is smaller when compared to the other two terms on the right hand side. By neglecting this term we have

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0). \tag{4.14}$$

Notice that the graph of the function $g(x) = f(x_0) + f'(x_0)(x - x_0)$ is precisely the tangent line to the graph of f at the point $(x_0, f(x_0))$. We now define x_1 to be the x-coordinate of the point of intersection of this tangent line with the x-coordinate. That is, the point x_1 is such that $g(x_1) = 0$, which gives

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}. \tag{4.15}$$

This gives the first member of the iterative sequence of the Newton-Raphson's method.

We now summarize the Newton-Raphson's method.

Hypothesis:

(1) Let the function f be C^1 and r be the root of the equation $f(x) = 0$ with $f'(r) \neq 0$.

(2) The initial guess x_0 is chosen sufficiently close to the root r.

Algorithm:
Step 1: For $n = 0, 1, 2, \cdots$, the iterative sequence of the Newton-Raphson's method is given by (4.13)

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

Step 2: Choose any one of the stopping criteria (or a combination of them) discussed in Section 4.2. If this criterion is satisfied, stop the iteration. Otherwise, repeat the step 1 until the criterion is satisfied. □

Remark 4.17. It is evident that if the initial guess x_0 is such that $f'(x_n) = 0$, for some $n \in \mathbb{N}$, then the Newton-Raphson method fails. Geometrically, this means that the tangent line to the graph of f at the point $(x_n, f(x_n))$ is parallel to the x-axis. Therefore, this line never intersects x-axis and hence x_{n+1} never exists. See Remark 4.12 for the failure of secant method and compare it with the present case. □

Example 4.18. Consider the equation

$$\sin x + x^2 - 1 = 0.$$

Let $x_0 = 1$. Then the iterations from the Newton-Raphson method gives

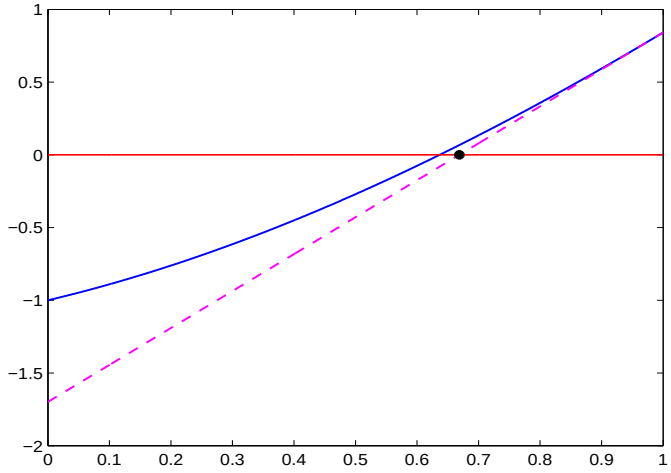

Fig. 4.2. Iteration Procedure of Newton-Raphson's method for $f(x) = \sin(x) + x^2 - 1$.

n	x_n	ϵ
1	0.668752	0.032019
2	0.637068	0.000335
3	0.636733	0.000000

Figure 4.2 shows the iterative points x_2 and x_3 in black bullet. Recall that the exact solution is $x^* \approx 0.636733$. Obviously, the Newton-Raphson method is much faster than both bisection and secant methods in this example. □

Let us now discuss the convergence of the Newton-Raphson method.

Theorem 4.19 (Error Estimates and Convergence of Newton-Raphson method). Hypothesis:

(1) *Let* $f : \mathbb{R} \to \mathbb{R}$ *be a* $C^2(\mathbb{R})$ *function.*

(2) *Let* r *be a simple root of the nonlinear equation* $f(x) = 0$*, that is* $f'(r) \neq 0$*.*

Conclusion: *Then there exists a* $\delta > 0$ *such that for every* $x_0 \in [r - \delta, r + \delta]$*,*

(1) *each term of the Newton-Raphson iterative sequence* $\{x_n\}$ *is well-defined.*

(2) *the sequence* $\{x_n\}$ *belongs to the interval* $[r - \delta, r + \delta]$*.*

(3) $\lim_{n\to\infty} x_n = r$*.*

Proof: The Newton-Raphson sequence is defined by

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

The error $e_{n+1} := r - x_{n+1}$ can be written as

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

Using Taylor's theorem, we have

$$0 = f(r) = f(x_n) + (r - x_n)f'(x_n) + \frac{f''(\xi_n)}{2}(r - x_n)^2, \tag{4.16}$$

where ξ_n lies between r and x_n. Using the information from the last equation, we get the following expression for e_{n+1}:

$$e_{n+1} = -\frac{f''(\xi_n)}{2f'(x_n)}(r - x_n)^2 = -\frac{f''(\xi_n)}{2f'(x_n)}e_n^2. \tag{4.17}$$

Since $f'(r) \neq 0$, there exists a $\delta_1 > 0$ such that $f'(x) \neq 0$ for every $x \in [r - \delta_1, r + \delta_1]$. In particular, there exists $M_{\delta_1} > 0$ such that

$$\min_{x \in [r-\delta_1, r+\delta_1]} |f'(x)| \geq M_{\delta_1}.$$

Also, since f'' is a continuous function, there exists a $L_{\delta_1} > 0$ such that

$$\max_{x \in [r-\delta_1, r+\delta_1]} |f''(x)| \leq L_{\delta_1}.$$

Thus we get

$$|e_{n+1}| \leq \frac{L_{\delta_1}}{2M_{\delta_1}}|e_n|^2. \tag{4.18}$$

Note that $C(\delta_1) := \frac{L_{\delta_1}}{2M_{\delta_1}}$ is a non-increasing function of δ_1, which implies that $C(\delta_1)$ remains bounded as we decrease the value of δ_1. Thus, there exists a $\delta < 1$ such that $\delta < \delta_1$, and $\delta\, C_\delta < 1$. We will now show that this δ has the required property. The inequality (4.18), for this δ reads

$$|e_{n+1}| \leq C_\delta |e_n|^2. \tag{4.19}$$

Proof of (1) **and** (2)**:** Let $x_0 \in [r - \delta, r + \delta]$. In other words, $|r - x_0| \leq \delta$. Then x_1 is well-defined as $f'(x_0) \neq 0$, since x_0 belongs to the interval $[r - \delta, r + \delta]$ on which f' is never equal to zero. We will prove that $|r - x_1| \leq \delta$.

$$|r - x_1| = |e_1| \leq C_\delta |e_0|^2 \leq C_\delta \delta |e_0| \leq |e_0| \leq \delta.$$

By induction, it follows that x_n is well-defined, and $|r - x_n| \leq \delta$ for every $n \in \mathbb{N}$.

Proof of (3)**:** From the inequalities,

$$|e_{n+1}| \leq C_\delta |e_n|^2 \leq \delta C_\delta |e_n| \leq (\delta C_\delta)^2 |e_{n-1}| \leq \cdots \leq (\delta C_\delta)^{n+1} |e_0|,$$

the convergence of the sequence follows as $\delta\, C_\delta < 1$. □

Theorem on Newton-Raphson method says that if we start near-by a root of the non-linear equation, then Newton-Raphson iterative sequence is well-defined and converges. For increasing convex functions, we need not be very careful in choosing the initial guess. For such functions, the Newton-Raphson iterative sequence always converges, whatever may be the initial guess. This is the content of the next theorem.

Theorem 4.20 (Convergence Result for Convex Functions).
Hypothesis: *Let $f : \mathbb{R} \to \mathbb{R}$ be a twice continuously differentiable function such that*

(1) *f is convex, i.e., $f''(x) > 0$ for all $x \in \mathbb{R}$.*

(2) *f is strictly increasing, i.e., $f'(x) > 0$ for all $x \in \mathbb{R}$.*

(3) *there exists an $r \in \mathbb{R}$ such that $f(r) = 0$.*

Conclusion: *Then*

(1) *r is the unique solution of $f(x) = 0$.*

(2) *For every choice of x_0, the Newton-Raphson iterative sequence converges to r.*

Proof:
Proof of (1)**:** Since f is strictly increasing, the function cannot take the same value more than once. Thus $f(x) = 0$ has exactly one solution.

Proof of (2)**:** From (4.17), it follows that $e_{n+1} \leq 0$. This implies that $r \leq x_{n+1}$. Since f is a strictly increasing function, we get $f(r) \leq f(x_{n+1})$. Thus $f(x_{n+1}) \geq 0$ for every $n \in \mathbb{N}$. From (4.13), we get $x_{n+1} \leq x_n$. That is, the sequence $\{x_n\}$ is a non-increasing sequence, and is bounded below by r, and hence converges. Let us denote the limit by x^*.

On the other hand, the sequence $\{e_n\}$ is a non-decreasing sequence, bounded above by 0. Let e^* denote the limit of the sequence $\{e_n\}$.

Passing to the limit as $n \to \infty$ in the equation

$$e_{n+1} = e_n + \frac{f(x_n)}{f'(x_n)},$$

we get

$$e^* = e^* + \frac{f(x^*)}{f'(x^*)}.$$

From the last equality, we get $f(x^*) = 0$. □

Remark 4.21. It follows from (4.19) that the order of convergence of the Newton-Raphson method is 2 (see Definition 1.42(3)), that is the Newton-Raphson iterative sequence converges quadratically. □

To following example illustrates the quadratic convergence of the Newton-Raphson method.

Example 4.22. Start with $x_0 = -2.4$ and use Newton-Raphson iteration to find the root $r = -2.0$ of the polynomial

$$f(x) = x^3 - 3x + 2.$$

The iteration formula is

$$x_{n+1} = \frac{2x_n^3 - 2}{3x_n^2 - 3}.$$

It is easy to verify that $|r - x_{n+1}|/|r - x_n|^2 \approx 2/3$, which shows the quadratic convergence of Newton-Raphson's method as proved in the above theorem. □

4.3.3 Fixed-Point Iteration Method

In fixed point iteration method, search for a solution of nonlinear equation $f(x) = 0$ is replaced by search for a fixed point of a function g, and with the property that if $\alpha \in \mathbb{R}$ is a fixed point of g (*i.e.,* $g(\alpha) = \alpha$), then $f(\alpha) = 0$.

In general, there may be more than one choice of g with this property as illustrated by the following example.

Example 4.23. Note that $\alpha \in \mathbb{R}$ is a solution of the equation $x^2 - x - 2 = 0$ if and only if α is a solution of each of the following equations.

(1) $x = x^2 - 2$
(2) $x = \sqrt{x+2}$
(3) $x = 1 + \dfrac{2}{x}$.

The **fixed-point iteration method** for finding a solution of $g(x) = x$ consists of a sequence of iterates $\{x_n\}$, starting from an initial guess x_0, defined by

$$x_n = g(x_{n-1}) \tag{4.20}$$

As we saw in Example 4.23, for a given nonlinear equation, the iteration function is not unique. The crucial point in this method is to choose a good iteration function $g(x)$. A good iteration function should satisfy the following properties:

(1) For the given starting point x_0, the successive approximations x_n given by (4.20) can be calculated.
(2) The sequence $x_1, x_2, \cdots$ converges to some point ξ.
(3) The limit ξ is a fixed point of $g(x)$, ie., $\xi = g(\xi)$.

Not every iteration function has all these properties. The following example shows that for certain iteration functions, even the sequence of iterates need not be defined.

Example 4.24. Consider the equation

$$x^2 - x = 0.$$

This equation can be re-written as $x = \pm\sqrt{x}$. Let us take the iterative function

$$g(x) = -\sqrt{x}.$$

Since $g(x)$ is defined only for $x > 0$, we have to choose $x_0 > 0$. For this value of x_0, we have $g(x_0) < 0$ and therefore, x_1 cannot be calculated. □

Therefore, the choice of $g(x)$ has to be made carefully so that the sequence of iterates can be calculated.

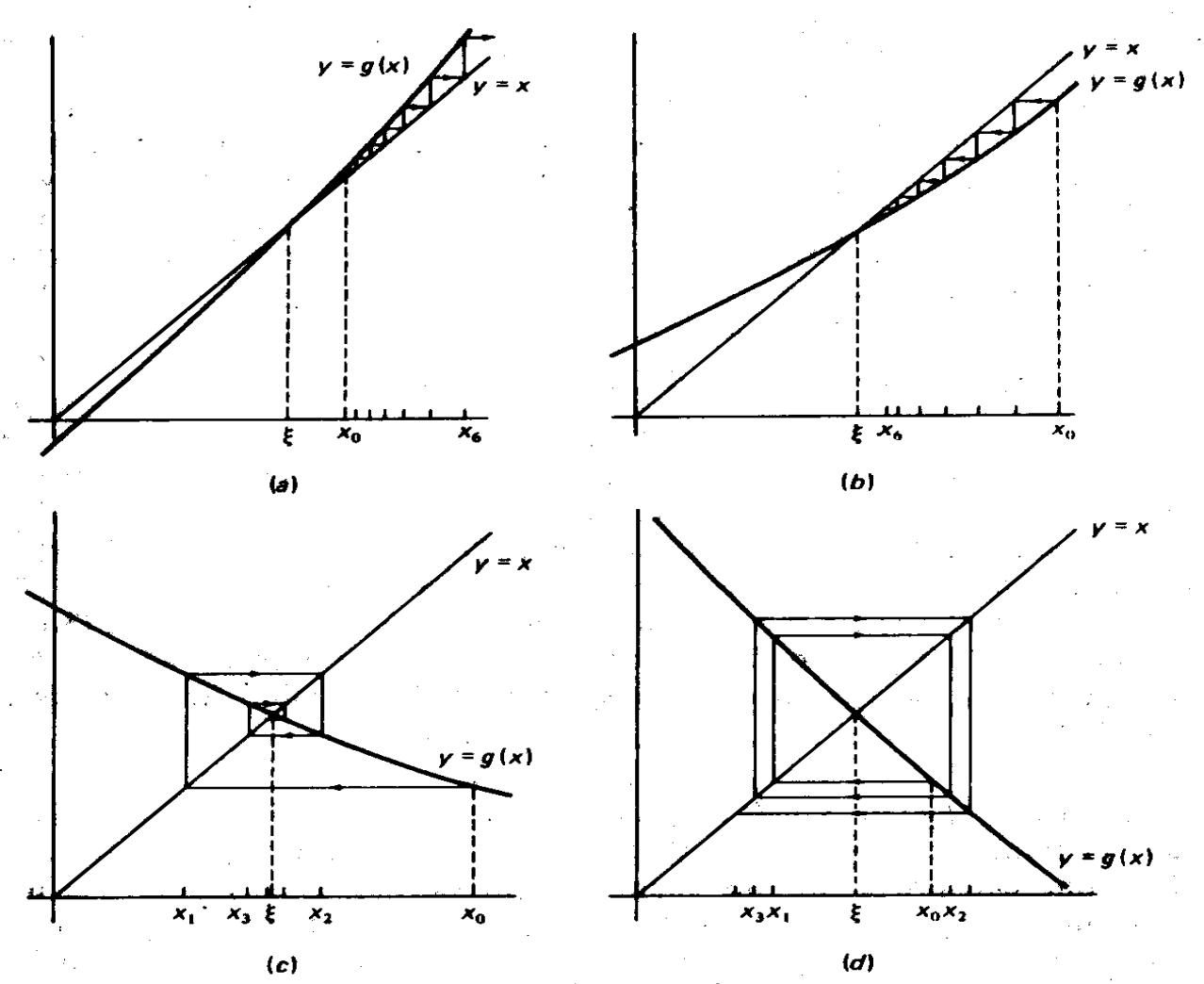

Fig. 4.3. Fixed-point Iteration Procedure.

How to choose such an iteration function $g(x)$?

Note that $x_1 = g(x_0)$, and $x_2 = g(x_1)$. Thus x_1 is defined whenever x_0 belongs to the domain of g. Thus we must take the initial guess x_0 from the domain of g. For defining x_2, we need that x_1 is in the domain of g once again. In fact the sequence is given by

$$x_0,\ g(x_0),\ g \circ g(x_0),\ g \circ g \circ g(x_0),\ \cdots$$

Thus to have a well-defined iterative sequence, we require that

Range of the function g is contained in the domain of g.

A function with this property is called a **self map**. We make our first assumption on the iterative function as

Assumption 1: $a \leq g(x) \leq b$ for all $a \leq x \leq b$.

It follows that if $a \leq x_0 \leq b$, then for all n, $x_n \in [a, b]$ and therefore $x_{n+1} = g(x_n)$ is defined and belongs to $[a, b]$.

Let us now discuss about the point 3. This is a natural expectation since the expression $x = g(x)$, which is the solution of the required equation is precisely the definition of a fixed point. To achieve this, we need $g(x)$ to be a continuous function. For if $x_n \to x^*$ then

$$x^* = \lim_{n\to\infty} x_n = \lim_{n\to\infty} g(x_{n-1}) = g(\lim_{n\to\infty} x_{n-1}) = g(x^*)$$

Therefore, we need

Assumption 2: The iterative function g is continuous.

It is easy to prove that a continuous self map on a bounded interval always has a fixed point. However, the question is whether the sequence (4.20) generated by the iterative function g converges, which is the requirement stated in point 2. This point is well understood geometrically. The Figures 4.3(a) and 4.3(c) illustrate the convergence of the fixed-point iterations whereas the Figures 4.3(b) and 4.3(d) illustrate the diverging iterations. In this geometrical observation, we see that when $g'(x) < 1$, we have convergence and otherwise, we have divergence. Therefore, we make the assumption

Assumption 3: The iteration function $g(x)$ is differentiable on $I = [a, b]$. Further, there exists a constant $0 < K < 1$ such that

$$|g'(x)| \leq K, \quad x \in I. \tag{4.21}$$

Such a function is called the **contraction map**.

Let us now present the algorithm of the fixed point iteration method.

Hypothesis: Assumptions 1, 2, and 3 stated above.

Algorithm: Choose an appropriate iteration function $g : [a, b] \to [a, b]$, where the interval $[a, b]$ is chosen in such a way that g is a self map.

Step 1: Choose an initial guess $x_0 \in [a, b]$.

Step 2: Define the iteration methods as

$$x_{n+1} = g(x_n), \quad n = 0, 1, \cdots$$

Step 3: For a pre-assigned positive quantity ϵ, check for one of the (fixed) stopping criteria discussed in Section 4.2. If the criterion is satisfied, stop the iteration. Otherwise, repeat the step 1 until the criterion is satisfied. □

Theorem 4.25 (Convergence Result for Fixed-Point Iteration Method).

Hypothesis: *Let the iterative function g be chosen so that*

(1) *g is defined on the interval $[a, b]$ and $a \leq g(x) \leq b$. That is, g is a self map on $[a, b]$*

(2) *g is continuously differentiable on $[a, b]$*

(3) *g is a contraction map. That is,*

$$\lambda = \max_{a\leq x\leq b} |g'(x)| < 1. \tag{4.22}$$

Conclusion: *Then*

(1) $x = g(x)$ *has a unique solution* r *in* $[a, b]$.

(2) *For any choice of* $x_0 \in [a, b]$, *with* $x_{n+1} = g(x_n)$, $n = 0, 1, \cdots$,

$$\lim_{n\to\infty} x_n = r.$$

(3) *We further have*

$$|x_n - r| \leq \lambda^n |x_0 - r| \leq \frac{\lambda^n}{1-\lambda}|x_1 - x_0| \tag{4.23}$$

and

$$\lim_{n\to\infty} \frac{r - x_{n+1}}{r - x_n} = g'(r). \tag{4.24}$$

Proof. Proof for (1) is easy.

From mean-value theorem and (4.21), we have

$$|r - x_{n+1}| = |g(r) - g(x_n)| \leq \lambda |r - x_n|. \tag{4.25}$$

By induction, we have

$$|r - x_{n+1}| \leq \lambda^n |x_0 - r|, \quad n = 0, 1, \cdots .$$

Since, as $n \to \infty$, $\lambda^n \to 0$, we have $x_n \to r$. Further, we have

$$\begin{aligned} |x_0 - r| &= |x_0 - x_1 + x_1 - r| \\ &\leq |x_0 - x_1| + |x_1 - r| \\ &\leq \lambda |x_0 - r| + |x_0 - x_1|. \end{aligned}$$

Then solving for $|x_0 - r|$, we get (4.23).

Now we will prove the rate of convergence (4.24). From Mean-value theorem

$$r - x_{n+1} = g(r) - g(x_n) = g'(\xi_n)(r - x_n), \quad n = 0, 1, \cdots .$$

with ξ_n an unknown point between r and x_n. Since $x_n \to r$, we must have $\xi_n \to r$ and therefore,

$$\lim_{n\to\infty} \frac{r - x_{n+1}}{r - x_n} = \lim_{n\to\infty} g'(\xi_n) = g'(r).$$

This completes the proof. □

Remark 4.26. From the inequality (4.25), we see that the fixed point iteration method has linear convergence. In other words, the order of convergence of this method is 1. □

Example 4.27. The nonlinear equation

$$x^3 + 4x^2 - 10 = 0$$

has a unique solution in $[1, 2]$. Note that solution to each of the following fixed-point problems is a solution to the given nonlinear equation.

(1) $x = g_1(x) = x - x^3 - 4x^2 + 10$

(2) $x = g_2(x) = \sqrt{\dfrac{10}{x} - 4x}$

(3) $x = g_3(x) = \dfrac{1}{2}\sqrt{10 - x^3}$

(4) $x = g_4(x) = \sqrt{\dfrac{10}{4 + x}}$

(5) $x = g_5(x) = x - \dfrac{x^3 + 4x^2 - 10}{3x^2 + 8x}$.

We are going to show that among the five equivalent fixed-point formulations of the given nonlinear equation, only some of them turn out to be good iterative functions. Let us implement fixed-point iteration method with each of the five iterating functions, and compare the results which are tabulated below.

n	$x = g_1(x)$	$x = g_2(x)$	$x = g_3(x)$	$x = g_4(x)$	$x = g_5(x)$
0	1.5	1.5	1.5	1.5	1.5
1	-0.875	0.8165	1.286953768	1.348399725	1.373333333
2	6.732	2.9969	1.402540804	1.367376372	1.365262015
3	-469.7	$\sqrt{-8.65}$	1.345458374	1.364957015	1.365230014
4	1.03×10^8		1.375170253	1.365264748	1.365230013
5			1.360094193	1.3652	
6			1.367846968	1.365230576	
7			1.363887004	1.365229942	
8			1.365916734	1.365230022	
9			1.364878217	1.365230012	
10			1.365410062	1.365230014	
15			1.365223680	1.365230013	
20			1.365230236		
25			1.365230006		
30			1.365230013		

From the above table, we conclude that the iterative functions g_1 and g_2 are very bad, while that given by g_5 is the best. However iterative functions g_3 and g_4 are also good but requires more number of iterations compared to g_5.

(1) Note that the iterative function g_1 is given by

$$g_1(x) = x - x^3 - 4x^2 + 10.$$

Note that $g_1(1) = 6$ and $g_1(2) = -12$. Thus range of g_1 is not contained in $[1, 2]$. That is g_1 is not a self map on the interval $[1, 2]$. In fact,

$$g_1'(x) = 1 - 3x^2 - 8x.$$

Let us compute the minimum and maximum values of g_1' on the interval $[1, 2]$. Note that $g_1''(x) = -6x - 8$. Thus derivative of g_1' vanishes at only one point, namely $x = -\frac{4}{3}$ which is not in the interval $[1, 2]$. Thus g_1' has no maxima/minima in the interval $(1, 2)$. Hence maximum and minimum of g_1' are at the points $1, 2$. Note that $g_1'(1) = -10$, $g_1'(2) = -27$. Thus $|g'(x)| \geq 10$ on the interval $[1, 2]$. Thus the map g_1 is not only not contractive on $[1, 2]$ but is an expansive map on $[1, 2]$. This is the reason why the successive iterates using g_1 have increasing moduli.

(2) Note that the iterative function g_2 is given by

$$g_2(x) = \sqrt{\frac{10}{x} - 4x}.$$

It is easy to check that g_2 is not a self map of $[1, 2]$ to itself. In our computation above, we see that the entire iterative sequence is not defined as one of the iterates becomes negative, when the initial guess is taken as 1.5 The exact solution is approximately equal to $x^* = 1.365$ There is no interval containing x^* on which $|g_2'(x)| < 1$. In fact, $g_2'(x^*) \approx 3.4$ and as a consequence $|g_2'(x)| > 3$ on an interval containing x^*. Thus we dont expect a convergent iterative sequence even if the sequence is well-defined!

(3) Note that the iterative function g_3 is given by

$$g_3(x) = \frac{1}{2}\sqrt{10 - x^3}.$$

Note that g_3 is a decreasing function on $[1, 2]$ as

$$g_3'(x) = -\frac{3x^2}{4\sqrt{10 - x^3}} < 0$$

on $[1, 2]$. Thus maximum of g_3 is attained at $x = 1$, which is 1.5; and the minimum is attained at $x = 2$ which is approximately equal to 0.707. Thus g_3 is a self map of $[1, 2]$. But $|g_3'(2)| \approx 2.12$. Thus the condition

$$|g_3'(x)| \leq \lambda < 1$$

is violated on the interval $[1, 2]$. However by restricting to a smaller interval $[1, 1.5]$, we get that g_3 is a self map of $[1, 1.5]$ as g_3 is still decreasing function on $[1, 1.5]$ and $g_3(1) = 1.5$, $g_3(1.5) \approx 1.28$, and also

$$|g_3'(x)| \leq |g_3'(1.5)| \approx 0.66.$$

Thus g_3 satisfies the hypothesis of theorem on fixed-point iteration, and as expected the sequence of iterates converge.

(4) Note that the iterative function g_4 is given by

$$g_4(x) = \sqrt{\frac{10}{4+x}}.$$

We have

$$|g_4'(x)| = \left|\frac{-5}{\sqrt{10}(4+x)^{3/2}}\right| \leq \frac{5}{\sqrt{10}(5)^{3/2}} < 0.15 \text{ for all } x \in [1,2].$$

The bound obtained on the derivative of g_4 is considerably smaller when compared to that of g_3, which explains why the convergence is faster for the iterates obtained using g_4, when compared with those obtained by using g_3.

(5) Note that the iterative function g_5 is given by

$$g_5(x) = x - \frac{x^3 + 4x^2 - 10}{3x^2 + 8x}.$$

This converges much more faster compared to g_3 and g_4. Note that the fixed-point iterative sequence generated by g_5 is nothing but the iterative sequence of Newton-Raphson method for the solution of the nonlinear equation $f(x) = 0$.

Example 4.28. Consider the equation

$$\sin x + x^2 - 1 = 0.$$

Take the initial interval as $[0, 1]$. There are three possible choices for the iteration function, namely,

(1) $g_1(x) = \sin^{-1}(1 - x^2)$,

(2) $g_2(x) = -\sqrt{1 - \sin x}$,

(3) $g_3(x) = \sqrt{1 - \sin x}$.

Here we have

$$g_1'(x) = \frac{-2}{\sqrt{2 - x^2}}.$$

We can see that $|g_1'(x)| > 1$. Taking $x_0 = 0.8$ and denoting the absolute error as ϵ, we have

n	$g_1(x)$	ϵ
0	0.368268	0.268465
1	1.043914	0.407181
2	-0.089877	0.726610
3	1.443606	0.806873

The sequence of iterations is diverging as expected.

If we take $g_2(x)$, clearly the assumption 1 is violated and therefore is not suitable for the iteration process.

Let us take $g_3(x)$. Here, we have

$$g_3'(x) = \frac{-\cos x}{\sqrt{1-\sin x}}.$$

Therefore,

$$\begin{aligned} |g_3'(x)| &= \frac{\sqrt{1-\sin^2 x}}{2\sqrt{1-\sin x}} \\ &= \frac{\sqrt{1+\sin x}}{2} \\ &\leq \frac{1}{\sqrt{2}} < 1. \end{aligned}$$

Taking $x_0 = 0.8$ and denoting the absolute error as ϵ, we have

n	$g_3(x)$	ϵ
0	0.531643	0.105090
1	0.702175	0.065442
2	0.595080	0.041653
3	0.662891	0.026158

The sequence is converging. □

4.4 Comparison and Pitfalls of Iterative Methods

Closed domain methods: Bisection and Regula falsi methods

(1) In both these methods, where we are trying to find a solution of the nonlinear equation $f(x) = 0$, we are required to find an interval $[a, b]$ such that $f(a)$ and $f(b)$ have opposite signs. This calls for a complete study of the function f. In case the function has no solutions on the real line, this search for an interval will be futile. There is no way to realize this immediately, thus necessitating a full fledged understanding of the funtion f.

(2) Once it is known that we can start these methods, then surely the iterative sequences converge to a solution of the nonlinear equation.

(3) In bisection method, we can keep track of the error by means of an upper bound. But such a thing is not available for regula falsi method. In general convergence of bisection method iterates is slower compared to that of regula falsi method iterates.

(4) If the initial interval $[a, b]$ is such that the equation $f(x) = 0$ has a unique solution in it, then both the methods converge to the solution. If there are more than one solutions in $[a, b]$, then usually both methods find different solutions. The only way of finding the desired root is to find an interval in which there is exactly one solution to the nonlinear equation.

Open domain methods: Secant, Newton-Raphson, and Fixed point methods

(1) The main advantage of the open domain methods when compared to closed domain methods is that we don't need to locate a root in an interval. Rather, we can start the iteration with an arbitrarily chosen initial guess(es).

(2) The disadvantage of the open domain methods is that the iterative sequence may not be well-defined for all initial guesses. Even if the sequence is well-defined, it may not converge. Even if it converges, it may not converge to a specific root of interest.

(3) In situations where both open and closed domain methods converge, open domain methods are generally faster compared to closed domain methods. Especially, Newton-Raphson's method is faster than other methods as the order of convergence of this method is 2. In fact, this is the fastest method known today.

(4) In these methods, it may happen that we are trying to find a particular solution of the nonlinear equation, but the iterative sequence may converge to a different solution. Thus we have to be careful in choosing the initial guess. If the initial guess is far away from the expected root, then there is a danger that the iteration converges to another root of the equation.

In the case of Newton-Raphson's method, this usually happens when the slope $f'(x_0)$ is small and the tangent line to the curve $y = f(x)$ is nearly parallel to the x-axis. Similarly, in the case of secant method, this usually happens when the slope of the secant joining $(x_0, f(x_0))$ and $(x_1, f(x_1))$ is nearly parallel to the x-axis.

For example, if

$$f(x) = \cos x$$

and we seek the root $x^* = \pi/2$ and start with $x_0 = 3$, calculation reveals that

$$x_1 = -4.01525, \quad x_2 = -4.85266, \cdots,$$

and the iteration converges to $x = -4.71238898 \approx -3\pi/2$. The iterative sequence for $n = 1, 2$ is depicted in Figure 4.4.

(5) Suppose that $f(x)$ is positive and monotone decreasing on an unbounded interval $[a, \infty)$ and $x_0 > a$. Then the sequence might diverge. For example, if $f(x) = xe^{-x}$ and $x_0 = 2$, then

$$x_1 = 4.0, \quad x_2 = 5.333333..., \quad \cdots, p_{15} = 19.72354..., \cdots.$$

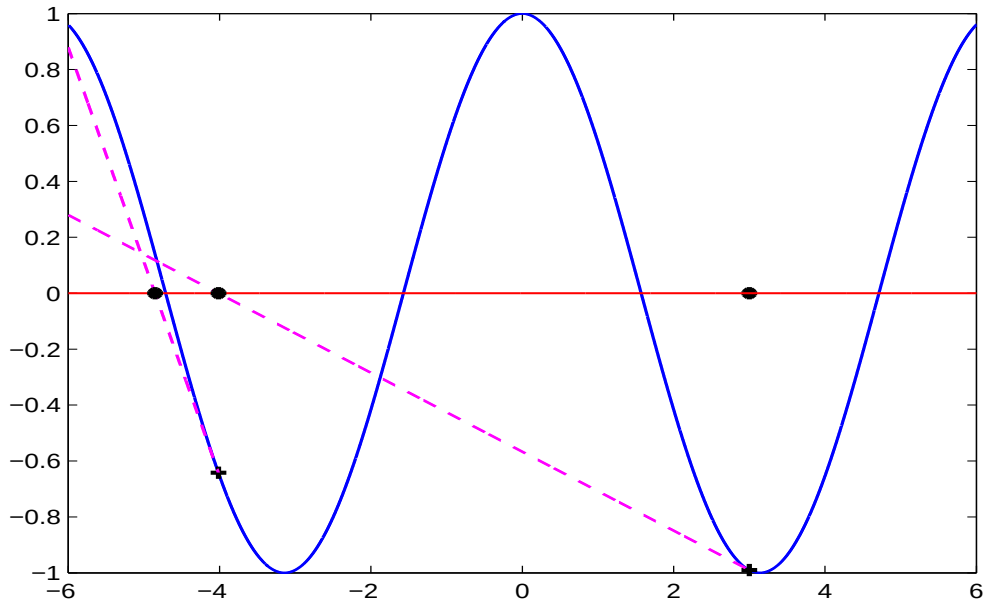

Fig. 4.4. Iteration Procedure of Newton-Raphson's method for $f(x) = \cos(x)$.

and the sequence diverges to $+\infty$. This particular function has another suprising problem. The value of $f(x)$ goes to zero rapidly as x gets large, for example $f(x_{15}) = 0.0000000536$, and it is possible that p_{15} could be mistaken for a root as per the residual error. Thus, using residual error for iterative methods nonlinear equations is often not preferred.

(6) The method can stuck in a cycle. For instance, let us compute the iterative sequence generated by the Newton-Raphson's method for the function $f(x) = x^3 - x - 3$ with the initial guess $x_0 = 0$. The iterative sequence is

$$x_1 = -3.00, \quad x_2 = -1.961538, \quad x_3 = -1.147176, \quad x_4 = -0.006579,$$

$$x_5 = -3.000389, \quad x_6 = -1.961818, \quad x_7 = -1.147430, \cdots$$

and we are stuck in a cycle where $x_{n+4} \approx x_k$ for $k = 0, 1, \cdots$. But if we start with a value x_0 sufficiently close with the root $r \approx 1.6717$, then the convergence is obtained. The proof of this is left as an exercise.

(7) If $f(x)$ has no real root, then there is no indication by these methods and the iterative sequence may simply oscillate. For example compute the Newton-Raphson iteration for

$$f(x) = x^2 - 4x + 5.$$

CHAPTER 5

Interpolation

Let a physical experiment be conducted and the outcome is recorded only at some finite number of times. If we want to know the outcome at some intermediate time where the data is not available, then we may have to repeat the whole experiment once again to get this data. In the mathematical language, suppose that the finite set of values

$$\{f(x_i) \,:\, i = 0, 1, \cdots, n\}$$

of a function f at a given set of points

$$\{x_i \,:\, i = 0, 1, \cdots, n\}$$

is known and we want to find the value of $f(x)$, where $x \in (x_j, x_k)$, for some $j = 1, 2, \cdots, n$ and $k = 1, 2, \cdots, n$. One way of obtaining the value of $f(x)$ is to compute this value directly from the expression of the function f. Often, we may not know the expression of the function explicitly and only the **data**

$$\{(x_i, y_i) \,:\, i = 0, 1, \cdots, n\}$$

is known, where $y_i = f(x_i)$. In terms of the physical experiments, repeating an experiment will quite often be very expensive. Therefore, one would like to get at least an approximate value of $f(x)$ (in terms of experiment, an approximate value of the outcome at the desired time). This is achieved by first constructing a function whose value at x_i coincides exactly with the value $f(x_i)$ for $i = 0, 1, \cdots, n$ and then finding the value of this constructed function at the desired points. Such a process is called **interpolation** and the constructed function is called the **interpolating function** for the given data.

In certain circumstances, the function f may be known explicitly, but still too difficult to perform certain operations like differentiation and integration. Thus, it is useful to restrict the class of interpolating functions to polynomials, where the differentiation and integration can be done more easily.

In Sectoin 5.1, we introduce the basic problem of polynomial interpolation and prove the existence and uniqueness of polynomial interpolating the given data. There are at least two ways to obtain the unique polynomial interpolating a given data, one is the **Lagrange** and another one is the **Newton**. In Section 5.1.2, we introduce Lagrange form of interpolating polynomial. Section 5.1.3 introduces the notion of divided differences and

Newton form of interpolating polynomial. The error analysis of the polynomial interpolation is studied in Section 5.3. In certain cases, the interpolating polynomial can differ significantly from the exact function. This is illustrated by Carl Runge and is called the **Runge Phenomenon**. In Section 5.3.4 we present the example due to Runge and state a few results on convergence of the interpolating polynomials. The concept of **piecewise polynomial interpolation** and **Spline interpolation** are discussed in Section 5.5.

5.1 Polynomial Interpolation

Polynomial interpolation is a concept of fitting a polynomial to a given data. Thus, to construct an interpolating polynomial, we first need a set of points at which the data values are known.

Definition 5.1. *Any collection of distinct real numbers $x_0, x_1, \cdots, x_n$ (not necessarily in increasing order) is called* **nodes**.

Definition 5.2 (Interpolating Polynomial). *Let $x_0, x_1, \cdots, x_n$ be the given nodes and $y_0, y_1, \cdots, y_n$ be real numbers. A polynomial $p_n(x)$ of degree less than or equal to n is said to be a* **polynomial interpolating the given data** *or an* **interpolating polynomial for the given data** *if*

$$p_n(x_i) = y_i, \quad i = 0, 1, \cdots n. \tag{5.1}$$

The condition (5.1) *is called the* **interpolation condition**. □

Remark 5.3. Let $x_0, x_1, \cdots, x_n$ be given nodes, and $y_0, y_1, \cdots, y_n$ be real numbers. Let $p_n(x)$ be a polynomial interpolating the given data. Then the graph of $p_n(x)$ passes through the set of $(n+1)$ distinct points in the xy-plane given by the table

x	x_0	x_1	x_2	x_3	$\cdots$	x_n
y	y_0	y_1	y_2	y_3	$\cdots$	y_n

We call the set $\{(x_i, y_i), i = 0, 1, \cdots, n\}$ as **data** and quite often we represent this set in the above form of a table. □

5.1.1 Existence and Uniqueness of Interpolating Polynomial

The following result asserts that an interpolating polynomial exists and is unique.

Theorem 5.4 (Existence and Uniqueness of Interpolating polynomial).

Let $x_0, x_1, \cdots, x_n$ be given nodes, and $y_0, y_1, \cdots, y_n$ be real numbers.

(1) *Then there exists a polynomial $p_n(x)$ of degree less than or equal to n such that*

$$p_n(x_i) = y_i, \quad i = 0, 1, \cdots, n.$$

That is, $p_n(x)$ is an interpolating polynomial for the data $\{(x_i, y_i), i = 0, 1, \cdots, n\}$.

(2) *Such a polynomial as above is unique.*

Proof:
Proof of uniqueness: Assume that $p_n(x)$ and $q_n(x)$ are interpolating polynomials of degree less than or equal to n that satisfies the interpolation condition (5.1). Let

$$r_n(x) = p_n(x) - q_n(x).$$

Then, $r_n(x)$ is also a polynomial of degree less than or equal to n, and by the interpolation condition, we have

$$r_n(x_i) = 0,$$

for every $i = 0, 1, \cdots, n$. Thus, $r_n(x)$ is a polynomial of degree less than or equal to n with $n+1$ distinct roots. By the fundamental theorem of algebra, we conclude that $r_n(x)$ is the zero polynomial. That is, the interpolating polynomial is unique.

Proof of existence: Existence of interpolating polynomial is proved using mathematical induction. If $n = 0$, then the constant polynomial

$$p_0(x) = y_0$$

is the required polynomial and its degree is less than or equal to 0. Assume that the result is true for $n = k$. We will now prove that the result is true for $n = k + 1$.

Let the data be given by

x	x_0	x_1	x_2	x_3	$\cdots$	x_k	x_{k+1}
y	y_0	y_1	y_2	y_3	$\cdots$	y_k	y_{k+1}

By the assumption, there exists a polynomial $p_k(x)$ of degree less than or equal to k such that the first k interpolating conditions

$$p_k(x_i) = y_i, \quad i = 0, 1, \cdots, k$$

hold. Define a polynomial $p_{k+1}(x)$ of degree less than or equal to $k + 1$ by

$$p_{k+1}(x) = p_k(x) + c(x - x_0)(x - x_1) \cdots (x - x_k), \tag{5.2}$$

where the constant c is such that the $(k+1)^{\text{th}}$ interpolation condition $p_{k+1}(x_{k+1}) = y_{k+1}$ holds. This is achieved by choosing

$$c = \frac{y_{k+1} - p_k(x_{k+1})}{(x_{k+1} - x_0)(x_{k+1} - x_1) \cdots (x_{k+1} - x_k)}.$$

Note that $p_{k+1}(x_i) = y_i$ for $i = 0, 1, \cdots, k$ and therefore $p_{k+1}(x)$ is an interpolating polynomial for the given data. This proves the result for $n = k + 1$. By the principle of mathematical induction, the result is true for any natural number n. □

Remark 5.5. A special case is when the data values y_i, $i = 0, 1, \cdots, n$ are the values of a function f at given nodes x_i, $i = 0, 1, \cdots, n$. In such a case, a polynomial interpolating the given data

x	x_0	x_1	x_2	x_3	$\cdots$	x_n
y	$f(x_0)$	$f(x_1)$	$f(x_2)$	$f(x_3)$	$\cdots$	$f(x_n)$

is said to be the **polynomial interpolating the given function** or the **interpolating polynomial for the given function** and has a special significance in applications of Numerical analysis for computing approximate solutions of differential equations and numerically computing complicated integrals. □

Example 5.6. Let the following data represent the values of f:

x	0	0.5	1
$f(x)$	1.0000	0.5242	-0.9037

The questions are the following:

(1) What is the exact expression for the function f?

(2) What is the value of $f(0.75)$?

We cannot get the exact expression for the function f just from the given data, because there are infinitely many functions having same value at the given set of points. Due to this, we cannot expect an exact value for $f(0.75)$, in fact, it can be any real number. On the other hand, if we look for f in the class of polynomials of degree less than or equal to 2, then Theorem 5.4 tells us that there is exactly one such polynomial and hence we can obtain a unique value for $f(0.75)$.

The interpolating polynomial happens to be

$$p_2(x) = -1.9042x^2 + 0.0005x + 1$$

and we have

$$p_2(0.75) = -0.0707380.$$

The function used to generate the above table of data is

$$f(x) = \sin\left(\frac{\pi}{2}e^x\right).$$

With this expression of f, we have (using 7-digit rounding)

$$f(0.75) \approx -0.1827495.$$

The relative error is given by

$$E_r(p_2(0.75)) = \frac{f(0.75) - p_2(0.75)}{f(0.75)} \approx 0.6129237.$$

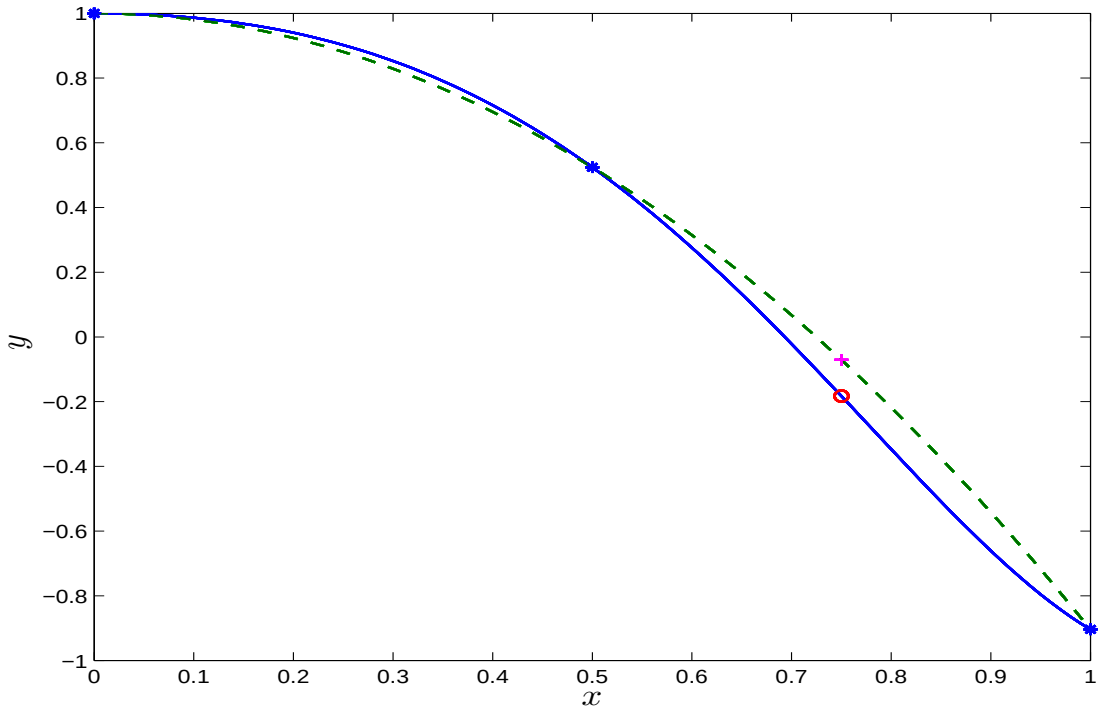

Fig. 5.1. The function $f(x) = \sin\left(\frac{\pi}{2}e^x\right)$ (blue solid line) and $p_2(x)$ (green dash line). Blue dots represent the given data, magenta '+' symbol indicates the value of $p_2(0.75)$ and the red 'O' symbol represents the value of $f(0.75)$.

That is, at the point $x = 0.75$ the polynomial approximation to the given function f has more than 61% error. The graph of the function f (blue solid line) and p_2 (green dash line) are depicted in Figure 5.1. The blue dots denote the given data, magenta '+' symbol indicates the value of $p_2(0.75)$ and the red 'O' symbol represents the value of $f(0.75)$. It is also observed from the graph that if we approximate the function f for $x \in [0, 0.5]$, then we obtain a better accuracy than approximating f in the interval $(0.5, 1)$. □

5.1.2 Lagrange's Form of Interpolating Polynomial

Definition 5.7 (Lagrange's Polynomial).

Let $x_0, x_1, \cdots, x_n$ be given nodes. For each $k = 0, 1, \cdots, n$, the polynomial $l_k(x)$ defined by

$$l_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)} \tag{5.3}$$

is called the k^{th} **Lagrange Polynomial** *or the k^{th}* **Lagrange Cardinal Function**.

Remark 5.8. Note that the k^{th} Lagrange polynomial depends on all the $n+1$ nodes $x_0, x_1, \cdots, x_n$. □

Theorem 5.9 (Lagrange's form of Interpolating Polynomial).
Hypothesis:

(1) *Let $x_0, x_1, \cdots, x_n$ be given nodes.*

(2) *Let the values of a function f be given at these nodes.*

(3) *For each $k = 0, 1, \cdots, n$, let $l_k(x)$ be the k^{th} Lagrange polynomial.*

(4) *Let $p_n(x)$ (of degree $\leq n$) be the polynomial interpolating the function f at the nodes $x_0, x_1, \cdots, x_n$.*

Conclusion: *Then, $p_n(x)$ can be written as*

$$p_n(x) = \sum_{i=0}^{n} f(x_i) l_i(x). \tag{5.4}$$

This form of the interpolating polynomial is called the **Lagrange's form of Interpolating Polynomial**.

Proof: Firstly, we will prove that $q(x) := \sum_{i=0}^{n} f(x_i) l_i(x)$ is an interpolating polynomial for the function f at the nodes $x_0, x_1, \cdots, x_n$. Since

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

we get $q(x_j) = f(x_j)$ for each $j = 0, 1, \cdots, n$. Thus, $q(x)$ is an interpolating polynomial. Since interpolating polynomial is unique by Theorem 5.4, the polynomial $q(x)$ must be the same as $p_n(x)$. This completes the proof of the theorem. □

Example 5.10. Consider the case $n = 1$ in which we have two distinct points x_0 and x_1. Then

$$l_0(x) = \frac{x - x_1}{x_0 - x_1}, \quad l_1(x) = \frac{x - x_0}{x_1 - x_0}$$

and

$$\begin{aligned} p_1(x) &= f(x_0) l_0(x) + f(x_1) l_1(x) \\ &= f(x_0) \frac{x - x_1}{x_0 - x_1} + f(x_1) \frac{x - x_0}{x_1 - x_0} \\ &= \frac{f(x_0)(x - x_1) - f(x_1)(x - x_0)}{x_0 - x_1} \\ &= f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0). \end{aligned} \tag{5.5}$$

This is the **linear interpolating polynomial** of the function f. Similarly, if we are given three nodes with corresponding values, then we can generate the **quadratic interpolating polynomial** and so on.. □

Example 5.11. Let the values of the function $f(x) = e^x$ be given at $x_0 = 0.82$ and $x_1 = 0.83$ by

$$e^{0.82} \approx 2.270500, \quad e^{0.83} \approx 2.293319.$$

In this example, we would like to obtain an approximate value of $e^{0.826}$ using the polynomial $p_1(x)$ that interpolates f at the nodes x_0, x_1. The polynomial $p_1(x)$ is given by

$$p_1(x) = 2.270500 + \frac{2.293319 - 2.270500}{0.83 - 0.82}(x - 0.82) = 2.2819x + 0.399342.$$

The approximate value of $e^{0.826}$ is taken to be $p_1(0.826)$, which is given by

$$p_1(0.826) \approx 2.2841914.$$

The true value of $e^{0.826}$ is

$$e^{0.826} \approx 2.2841638.$$

Note that the approximation to $e^{0.826}$ obtained using the interpolating polynomial $p_1(x)$, namely 2.2841914, approximates the exact value to at least five significant digits.

If we are given an additional node $x_2 = 0.84$ and the value of f at x_2 as $f(x_2) \approx 2.316367$, then we would like to use the quadratic interpolation polynomial p_2 to obtain an approximate value of $e^{0.826}$. In fact,

$$p_2(0.826) \approx 2.2841639.$$

Note that the approximation to $e^{0.826}$ obtained using the interpolating polynomial $p_2(x)$, namely 2.2841639, approximates the exact value to at least eight significant digits. □

Remark 5.12. The above example gives us a feeling that if we increase the number of nodes, and thereby increasing the degree of the interpolating polynomial, the polynomial approximates the orginal function more accurately. But this is not true in general, and we will discuss this further in Section 5.3.4. □

Remark 5.13. Let $x_0, x_1, \cdots, x_n$ be nodes, and f be a function. Recall that computing an interpolating polynomial in Lagrange's form requires us to compute for each $k = 0, 1, \cdots, n$, the k^{th} Lagrange's polynomial $l_k(x)$ which depends on the given nodes $x_0, x_1, \cdots, x_n$. Suppose that we have found the corresponding interpolating polynomial $p_n(x)$ of f in the Lagrange's form for the given data. Now if we add one more node x_{n+1}, the computation of the interpolating polynomial $p_{n+1}(x)$ in the Lagrange's form requires us to compute a new set of Lagrange's polynomials corresponding to the set of $(n + 1)$ nodes, and no advantage can be taken of the fact that p_n is already available. This will obviously increasing the computational costs.

An alternative form of the interpolating polynomial, namely **Newton's form of interpolating polynomial**, avoids this problem, and will be discussed in the next section. □

5.1.3 Newton's Form of Interpolating Polynomial

We saw in the last section that it is easy to write the Lagrange form of the interpolating polynomial once the Lagrange polynomials associated to a given set of nodes have been written. However we observed in Remark 5.13 that the knowledge of p_n (in Lagrange form) cannot be utilized to construct p_{n+1} in the Lagrange form. In this section we describe **Newton's form of interpolating polynomial**, which uses the knowledge of p_n in constructing p_{n+1}.

Theorem 5.14 (Newton's form of Interpolating Polynomial).
Hypothesis:

(1) *Let $x_0, x_1, \cdots, x_n$ be given nodes.*

(2) *Let the values of a function f be given at these nodes.*

(3) *Let $p_n(x)$ (of degree $\leq n$) be the polynomial interpolating the function f at the nodes $x_0, x_1, \cdots, x_n$.*

Conclusion: *Then, $p_n(x)$ can be written as*

$$p_n(x) = A_0 + A_1(x-x_0) + A_2(x-x_0)(x-x_1) + A_3\prod_{i=0}^{2}(x-x_i) + \cdots + A_n\prod_{i=0}^{n-1}(x-x_i) \quad (5.6)$$

where $A_0, A_1, \cdots, A_n$ are constants.

This form of the interpolating polynomial is called the **Newton's form of Interpolating Polynomial**.

Proof: Recall that in the proof of Theorem 5.4, we proved the existence of an interpolating polynomial using mathematical induction. In fact, we have given an algorithm for constructing interpolating polynomial. The interpolating polynomial given by (5.2) was precisely the Newton's form of interpolating polynomial. □

Remark 5.15. Let us recall the equation (5.2) from the proof of Theorem 5.4 now.

(1) It says that for each $n \in \mathbb{N}$, we have

$$p_n(x) = p_{n-1}(x) + A_n\prod_{i=0}^{n-1}(x-x_i) \quad (5.7)$$

for some constant A_n. This shows the recursive nature of computing Newton's form of interpolating polynomial.

(2) Indeed A_n is given by

$$A_n = \frac{f(x_n) - p_{n-1}(x_n)}{(x_n-x_0)(x_n-x_1)\cdots(x_n-x_{n-1})}. \quad (5.8)$$

From the last equality, note that A_0 depends only on $f(x_0)$. A_1 depends on the values of f at x_0 and x_1 only. In general, A_n depends on the values of f at $x_0, x_1, x_2, \cdots, x_n$ only.

(3) To compute Newton's form of interpolating polynomial $p_n(x)$, it is enough to compute A_k for $k = 0, 1, \cdots, n$. However note that the formula (5.8) is not well-suited to compute A_k because we need to evaluate all the successive interpolating polynomials $p_k(x)$ for $k = 0, 1, \cdots, n-1$ and then evaluate them at the node x_k which is computationally costly. It then appears that we are in a similar situation to that of Lagrange's form of interpolating polynomial as far as computational costs are concerned. But this is not the case, as we shall see shortly that we can compute A_n directly using the given data (that is, the given values of the function at the nodes), and this will be done in Section 5.2. □

5.2 Newton's Divided Difference Formulas

Definition 5.16 (Divided Differences). *Let $x_0, x_1, \cdots, x_n$ be distinct nodes. Let $p_n(x)$ be the polynomial interpolating a function f at the nodes $x_0, x_1, \cdots, x_n$. The coefficient of x^n in the polynomial $p_n(x)$ is denoted by $f[x_0, x_1, \cdots, x_n]$, and is called an n^{th}* **divided difference of** *f.*

Remark 5.17.

(1) Since the interpolating polynomial for the function f at the nodes $x_0, x_1, \cdots, x_n$ is unique, there is one and only one coefficient of x^n; even though interpolation polynomial may have many forms like Lagrange's form and Newton's form. Thus the quantity $f[x_0, x_1, \cdots, x_n]$ is well-defined.

(2) More generally, the divided difference $f[x_i, x_{i+1}, \cdots, x_{i+k}]$ is the coefficient of x^k in the polynomial interpolating f at the nodes $x_i, x_{i+1}, \cdots, x_{i+k}$.

(3) The Newton's form of interpolating polynomial may be written, using divided differences, as

$$p_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \cdots, x_k] \prod_{i=0}^{k-1} (x - x_i) \tag{5.9}$$

□

Example 5.18. As a continuation of Example 5.10, let us contruct the linear interpolating polynomial of a function f in the Newton's form. In this case, the interpolating polynomial is given by

$$p_1(x) = f[x_0] + f[x_0, x_1](x - x_0),$$

where

$$f[x_0] = f(x_0), \quad f[x_0, x_1] = \frac{f(x_0) - f(x_1)}{x_0 - x_1} \tag{5.10}$$

are **zeroth** and **first order divided differences**, respectively. Observe that this polynomial is exactly the same as the interpolating polynomial obtained using Lagrange's form in Example 5.10. □

The following result is concerning the symmetry properties of divided differences.

Theorem 5.19 (Symmetry). *The divided difference is a symmetric function of its arguments. That is, if $z_0, z_1, \cdots, z_n$ is a permutation of $x_0, x_1, \cdots, x_n$, then*

$$f[x_0, x_1, \cdots, x_n] = f[z_0, z_1, \cdots, z_n] \tag{5.11}$$

Proof: Since $z_0, z_1, \cdots, z_n$ is a permutation of $x_0, x_1, \cdots, x_n$, which means that the nodes $x_0, x_1, \cdots, x_n$ have only been re-labelled as $z_0, z_1, \cdots, z_n$, and hence the polynomial interpolating the function f at both these sets of nodes is the same. By definition $f[x_0, x_1, \cdots, x_n]$ is the coefficient of x^n in the polynomial interpolating the function f at the nodes $x_0, x_1, \cdots, x_n$, and $f[z_0, z_1, \cdots, z_n]$ is the coefficient of x^n in the polynomial interpolating the function f at the nodes $z_0, z_1, \cdots, z_n$. Since both the interpolating polynomials are equal, so are the coefficients of x^n in them. Thus, we get

$$f[x_0, x_1, \cdots, x_n] = f[z_0, z_1, \cdots, z_n].$$

This completes the proof. □

The following result helps us in computing recursively the divided differences of higher order.

Theorem 5.20 (Higher-order divided differences). *Divided differences satisfy the equation*

$$f[x_0, x_1, \cdots, x_n] = \frac{f[x_1, x_2, \cdots, x_n] - f[x_0, x_1, \cdots, x_{n-1}]}{x_n - x_0} \tag{5.12}$$

Proof: Let us start the proof by setting up the following notations.

- Let $p_n(x)$ be the polynomial interpolating f at the nodes x_0, x_1, $\cdots$, x_n.
- Let $p_{n-1}(x)$ be the polynomial interpolating f at the nodes x_0, x_1, $\cdots$, x_{n-1}.
- Let $q(x)$ be the polynomial interpolating f at the nodes x_1, x_2, $\cdots$, x_n.

Claim: We will prove the following relation between p_{n-1}, p_n, and q:

$$p_n(x) = p_{n-1}(x) + \frac{x - x_0}{x_n - x_0}\big(q(x) - p_{n-1}(x)\big) \tag{5.13}$$

Since both sides of the equality in (5.13) are polynomials of degree less than or equal to n, and $p_n(x)$ is the polynomial interpolating f at the nodes $x_0, x_1, \cdots, x_n$, the equality in (5.13) holds for all x if and only if it holds for $x \in \{ x_0, x_1, \cdots, x_n \}$ and both sides of the equality reduce to $f(x)$ for $x \in \{ x_0, x_1, \cdots, x_n \}$. Let us now verify the equation (5.13) for $x \in \{ x_0, x_1, \cdots, x_n \}$.

(1) When $x = x_0$,

$$p_{n-1}(x_0) + \frac{x_0 - x_0}{x_n - x_0}\big(q(x_0) - p_{n-1}(x_0)\big) = p_{n-1}(x_0) = f(x_0) = p_n(x_0).$$

(2) When $x = x_k$ for $1 \leq k \leq n-1$, $q(x_k) = p_{n-1}(x_k)$ and thus we have

$$p_{n-1}(x_k) + \frac{x_k - x_0}{x_n - x_0}\big(q(x_k) - p_{n-1}(x_k)\big) = p_{n-1}(x_k) = f(x_k) = p_n(x_k).$$

(3) When $x = x_n$, we have

$$p_{n-1}(x_n) + \frac{x_n - x_0}{x_n - x_0}\big(q(x_n) - p_{n-1}(x_n)\big) = p_{n-1}(x_n) + \big(f(x_n) - p_{n-1}(x_n)\big) = f(x_n) = p_n(x_n).$$

This finishes the proof of the Claim.

The coefficient of x^n in the polynomial $p_n(x)$ is $f[x_0, x_1, \cdots, x_n]$. The coefficient of x^n using the right hand side of the equation (5.13) is given by

$$\Big(\text{coefficient of } x^n \text{ in } p_{n-1}(x)\Big) + \frac{1}{x_n - x_0}\Big(\text{coefficient of } x^n \text{ in } (x - x_0)\big(q(x) - p_{n-1}(x)\big)\Big).$$

On noting that the coefficient of x^{n-1} in the polynomial p_{n-1} is $f[x_0, x_1, \cdots, x_{n-1}]$, the coefficient of x^{n-1} in the polynomial q is $f[x_1, x_2, \cdots, x_n]$, and the coefficient of x^n in the polynomial p_{n-1} is zero, we get that the coefficient of x^n using the right hand side of the equation (5.13) becomes

$$\frac{f[x_1, x_2, \cdots, x_n] - f[x_0, x_1, \cdots, x_{n-1}]}{x_n - x_0}.$$

Comparing the coefficients of x^n in the left and right hand sides of the equation (5.13) yields

$$f[x_0, x_1, \cdots, x_n] = \frac{f[x_1, x_2, \cdots, x_n] - f[x_0, x_1, \cdots, x_{n-1}]}{x_n - x_0}.$$

□

Remark 5.21. Let $i, j \in \mathbb{N}$. Applying Theorem 5.20 to a set of nodes $x_i, x_{i+1}, \cdots, x_{i+j}$, we conclude

$$f[x_i, x_{i+1}, \cdots, x_{i+j}] = \frac{f[x_{i+1}, x_{i+2}, \cdots, x_{i+j}] - f[x_i, x_{i+1}, \cdots, x_{i+j-1}]}{x_{i+j} - x_i} \tag{5.14}$$

Note that the divided differences $f[x_0, x_1, \cdots, x_n]$ are defined only for distinct nodes $x_0, x_1, \cdots, x_n$. □

5.2.1 Divided Differences Table

Given a collection of $(n+1)$ nodes $x_0, x_1, \cdots, x_n$ and the values of the function f at these nodes, we can construct the Newton's form of interpolating polynomial $p_n(x)$ using divided differences. As observed earlier, the Newton's form of interpolation polynomial has the formula

$$p_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \cdots, x_k] \prod_{i=0}^{k-1} (x - x_i) \tag{5.15}$$

One can explicitly write the formula (5.15) for $n = 1, 2, 3, 4, 5, \cdots$. For instance, when $n = 5$, the formula (5.15) reads

$$\left.\begin{aligned} p_5(x) = \quad & \boldsymbol{f[x_0]} \\ & + \boldsymbol{f[x_0, x_1]}(x - x_0) \\ & + \boldsymbol{f[x_0, x_1, x_2]}(x - x_0)(x - x_1) \\ & + \boldsymbol{f[x_0, x_1, x_2, x_3]}(x - x_0)(x - x_1)(x - x_2) \\ & + \boldsymbol{f[x_0, x_1, x_2, x_3, x_4]}(x - x_0)(x - x_1)(x - x_2)(x - x_3) \\ & + \boldsymbol{f[x_0, x_1, x_2, x_3, x_4, x_5]}(x - x_0)(x - x_1)(x - x_2)(x - x_3)(x - x_4) \end{aligned}\right\} \tag{5.16}$$

For easy computation of the divided differences in view of the formula (5.12), it is convenient to write the divided differences in a table form. For $n = 5$, the divided difference table is given by

x_0	$\boldsymbol{f(x_0)}$	$\boldsymbol{f[x_0, x_1]}$				
x_1	$f(x_1)$		$\boldsymbol{f[x_0, x_1, x_2]}$			
		$f[x_1, x_2]$		$\boldsymbol{f[x_0, x_1, x_2, x_3]}$		
x_2	$f(x_2)$		$f[x_1, x_2, x_3]$		$\boldsymbol{f[x_0, x_1, x_2, x_3, x_4]}$	
		$f[x_2, x_3]$		$f[x_1, x_2, x_3, x_4]$		$\boldsymbol{f[x_0, x_1, x_2, x_3, x_4, x_5]}$
x_3	$f(x_3)$		$f[x_2, x_3, x_4]$		$f[x_1, x_2, x_3, x_4, x_5]$	
		$f[x_3, x_4]$		$f[x_2, x_3, x_4, x_5]$		
x_4	$f(x_4)$		$f[x_3, x_4, x_5]$			
		$f[x_4, x_5]$				
x_5	$f(x_5)$					

Comparing the above divided differences table and the interpolating polynomial p_5 given by (5.16), we see that the leading members of each column (denoted in bold font) are the required divided differences used in $p_5(x)$.

5.2.2 Divided Difference Formula for Repeated Nodes

It follows from the symmetric property (Theorem 5.19) that the n^{th} order divided difference formula (5.12) is not well-defined if at least two nodes coincide. But such situations do occur quite common in numerical analysis, for instance in the error analysis of quadrature formulas. So, we need to interpret the define of the divided difference in such a way that it is still well defined if two or more nodes coincide. For this, the following theorem is useful.

Theorem 5.22 (Hermite-Genocchi Formula).
Hypothesis:

(1) *Let* x_0, x_1, $\cdots$, $x_n \in [a, b]$ *be distinct nodes.*

(2) *Let* f *be* n*-times continuously differentiable function on the interval* $[a, b]$.

Conclusion: *Then*

$$f[x_0, x_1, \cdots, x_n] = \int_{\tau_n} \cdots \int f^{(n)}(t_0 x_0 + \cdots + t_n x_n)\, dt_1 \cdots dt_n, \tag{5.17}$$

where

$$\tau_n = \left\{ (t_1, t_2, \cdots, t_n) \ / \ t_i \geq 0, i = 1, 2, \cdots, n; \quad \sum_{i=1}^{n} t_i \leq 1 \right\}, \quad t_0 = 1 - \sum_{i=1}^{n} t_i. \tag{5.18}$$

Proof: We prove the theorem by induction.

Step 1 : First, we prove (5.17) for $n = 1$. From (5.18), we see that $\tau_1 = [0, 1]$. Using the expression for t_0 in (5.18), we have

$$\begin{aligned}\int_0^1 f'(t_0 x_0 + t_1 x_1) dt_1 &= \int_0^1 f'(x_0 + t_1(x_1 - x_0))\, dt_1 = \frac{1}{x_1 - x_0} f(x_0 + t_1(x_1 - x_0))\Big|_{t_1=0}^{t_1=1} \\ &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f[x_0, x_1].\end{aligned}$$

Thus, we have proved the result (5.17) for $n = 1$.

Step 2 : Now assume that the formula (5.17) holds for $n = k \geq 1$ and prove the formula for $n = k + 1$. We have

$$\begin{aligned}&\int_{\tau_{k+1}} \cdots \int f^{(k+1)}(t_0 x_0 + t_1 x_1 + \cdots + t_{k+1} x_{k+1})\, dt_1 \cdots dt_{k+1} \\ &= \int_{\tau_k} \cdots \int \left(\int_0^{1-(t_1+\cdots+t_k)} f^{(k+1)}(x_0 + t_1(x_1 - x_0) + \cdots + t_{k+1}(x_{k+1} - x_0))\, dt_{k+1} \right) dt_1 \cdots dt_k \\ &= \int_{\tau_k} \cdots \int \frac{1}{x_{k+1} - x_0} \left[f^{(k)}(x_0 + t_1(x_1 - x_0) + \cdots + t_{k+1}(x_{k+1} - x_0)) \right]_{t_{k+1}=0}^{t_{k+1}=1-(t_1+\cdots+t_k)} dt_1 \cdots dt_k \\ &= \frac{1}{x_{k+1} - x_0} \left[\int_{\tau_k} \cdots \int f^{(k)}(x_{k+1} + t_1(x_1 - x_{k+1}) + \cdots + t_k(x_k - x_{k+1}))\, dt_1 \cdots dt_k \right. \\ &\qquad \left. - \int_{\tau_k} \cdots \int f^{(k)}(x_0 + t_1(x_1 - x_0) + \cdots + t_k(x_k - x_0))\, dt_1 \cdots dt_k \right] \\ &= \frac{f[x_1, \cdots, x_k, x_{k+1}] - f[x_0, x_1, \cdots, x_k]}{x_{k+1} - x_0} \\ &= f[x_0, x_1, \cdots, x_{k+1}].\end{aligned}$$

By the principle of mathematical induction, the result is true for all $n \in \mathbb{N}$. □

Remark 5.23. Since f is n-times continuously differentiable, the right hand side of (5.17) is meaningful for any set of points $x_0, x_1, \cdots, x_n$, which are not necessarily distinct. This gives a meaning, which can be used to define the n^{th} order divided difference when $x_0, x_1, \cdots, x_n$ are not distinct (*i.e.*, one or more points are repeated). □

Following results are direct consequences of the above theorem.

Corollary 5.24.
Hypothesis:

(1) *Let $x_0, x_1, \cdots, x_n \in [a,b]$ be a set of points, not necessarily distinct.*
(2) *Let f be n times continuously differentiable function on the interval $[a,b]$.*

Conclusion:

(1) *The function $(x_0, x_1, \cdots, x_n) \longmapsto f[x_0, x_1, \cdots, x_n]$ is continuous on $\mathbb{R}^{n+1}$.*
(2) *For any $x \in [a,b]$, the limit*

$$\lim_{(x_0,x_1,\cdots,x_n)\to(x,x,\cdots,x)} f[x_0, x_1, \cdots, x_n]$$

exists and this limit is the n^{th}-order divided difference $f[x, x, \cdots, x]$ (x repeated $(n+1)$-times). Further, we have

$$f[x, x, \cdots, x] = \frac{f^{(n)}(x)}{n!}. \tag{5.19}$$

Proof.

(1) The proof is out side the scope of this course and hence omitted.
(2) The proof follows from the fact that

$$\int_{\tau_n} \cdots \int 1 \; dt_1 \cdots dt_n = \frac{1}{n!}$$

and the Hermite-Genocchi formula (5.17). □

The following theorem gives an expression for the divided difference of an $(n+2)$-times differentiable function when two nodes are repeated.

Theorem 5.25.
Hypothesis:

(1) *Let $x_0, x_1, \cdots, x_n$ be given (distinct) nodes in an interval $[a,b]$.*
(2) *Let $x \in [a,b]$, $x \notin \{x_0, x_1, \cdots, x_n\}$.*

Conclusion: *The* $(n+2)^{\text{nd}}$*-order divided difference* $f[x_0, x_1, \cdots, x_n, x, x]$ *of an* $(n+2)$*-times continuously differentiable function* f *is given by*

$$f[x_0, x_1, \cdots, x_n, x, x] = \frac{d}{dx} f[x_0, x_1, \cdots, x_n, x]. \tag{5.20}$$

Proof: It follows from (5.17) that the function $F(x) = f[x_0, x_1, \cdots, x_n, x]$ for all $x \in [a, b]$ is well-defined and continuous. Therefore, using the symmetry property of the divided differences, we get

$$\begin{aligned} f[x_0, x_1, \cdots, x_n, x, x] &= \lim_{h \to 0} f[x_0, x_1, \cdots, x_n, x, x+h] \\ &= \lim_{h \to 0} f[x, x_0, x_1, \cdots, x_n, x+h]. \end{aligned}$$

As all the points used on the right hand side are distinct, we can use the formula (5.12) to get

$$\begin{aligned} f[x_0, x_1, \cdots, x_n, x, x] &= \lim_{h \to 0} \frac{f[x_0, x_1, \cdots, x_n, x+h] - f[x, x_0, x_1, \cdots, x_n]}{h} \\ &= \lim_{h \to 0} \frac{f[x_0, x_1, \cdots, x_n, x+h] - f[x_0, x_1, \cdots, x_n, x]}{h} \\ &= \frac{d}{dx} f[x_0, x_1, \cdots, x_n, x]. \end{aligned}$$

This completes the proof. □

5.3 Error in Polynomial Interpolation

Let f be a function defined on an interval $I = [a, b]$. Let $p_n(x)$ be a polynomial of degree less than or equal to n that interpolates the function f at $n+1$ nodes x_0, x_1, $\cdots$, x_n in I. How well does $p_n(x)$ approximate f on the interval I? This question leads to the analysis of **interpolation error**.

As we discussed at the beginning of Chapter 2, the error given by

$$\text{ME}_n(x) = f(x) - p_n(x). \tag{5.21}$$

is the **mathematical error** involved in the interpolating polynomial. But, when we perform the calculations using finite precision floating-point arithmetic, then the polynomial obtained, denoted by $\tilde{p}_n(x)$, need not be the same as the interpolating polynomial $p_n(x)$. The error

$$\text{AE}_n(x) = p_n(x) - \tilde{p}_n(x). \tag{5.22}$$

is the **arithmetic error** involved in the interpolating polynomial.

The **total error**, denoted by $\text{TE}_n(x)$, involved in the polynomial interpolation is therefore given by

$$\text{TE}_n(x) = f(x) - \tilde{p}_n(x) = (f(x) - p_n(x)) + (p_n(x) - \tilde{p}_n(x)) = \text{ME}_n(x) + \text{AE}_n(x).$$

In Subsection 5.3.1, we derive the mathematical error involved in polynomial interpolation. We analyze the effect of arithmetic error in Subsection 5.3.2.

5.3.1 Mathematical Error

The following theorem provides a formula for the interpolation error when we assume that the necessary data are given exactly without any floating-point approximation.

Theorem 5.26 (Mathematical Error in Interpolation). *Let $p_n(x)$ be the polynomial interpolating a function $f \in C^{n+1}[a,b]$ at the nodes $x_0, x_1, \cdots, x_n$ lying in $I = [a,b]$. Then for each $x \in I$, there exists a $\xi_x \in (a,b)$ such that*

$$\mathrm{ME}_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^{n} (x - x_i) \tag{5.23}$$

Proof: If x is one of the nodes, then (5.23) holds trivially for any choice of $\xi_x \in (a,b)$. Therefore, it is enough to prove the theorem when x is not a node. The idea is to obtain a function having at least $n+2$ distinct zeros; and then apply Rolle's theorem $n+1$ times to get the desired conclusion.

For a given $x \in I$ with $x \neq x_i$ $(i = 0, 1, \cdots, n)$, define a new function ψ on the interval I by

$$\psi(t) = f(t) - p_n(t) - \lambda \prod_{i=0}^{n} (t - x_i), \quad t \in I, \tag{5.24}$$

where λ is chosen so that $\psi(x) = 0$. This gives

$$\lambda = \frac{f(x) - p_n(x)}{\prod_{i=0}^{n} (x - x_i)}.$$

Note that $\psi(x_i) = 0$ for each $i = 0, 1, \cdots, n$. Thus, ψ has at least $n+2$ distinct zeros. By Rolle's theorem, the function ψ' has at least $n+1$ distinct zeros in (a,b). A repeated application of Rolle's theorem $n+1$ times to the function ψ gives that $\psi^{(n+1)}$ has at least one zero in (a,b); call it ξ_x. Differentiating (5.24) $(n+1)$-times and noting that

$$p_n^{(n+1)}(t) = 0, \quad \left(\prod_{i=0}^{n} (t - x_i) \right)^{(n+1)} = (n+1)!, \quad \psi^{(n+1)}(\xi_x) = 0,$$

we see that ξ_x satisfies

$$0 = \psi^{(n+1)}(\xi_x) = f^{(n+1)}(\xi_x) - \frac{f(x) - p_n(x)}{\prod_{i=0}^{n} (x - x_i)} (n+1)\,!.$$

Thus,

$$f^{(n+1)}(\xi_x) - \frac{\mathrm{ME}_n(x)}{\prod_{i=0}^{n}(x - x_i)}(n+1)\,! = 0.$$

The last equation yields (5.23). □

The following theorem plays an important role in the error analysis of numerical integration. The idea behind the proof of this theorem is similar to the idea used in the above theorem and is left as an exercise.

Theorem 5.27. *If $f \in C^{n+1}[a,b]$ and if $x_0, x_1, \cdots, x_n$ are distinct nodes in $[a,b]$, then there exists a point $\xi_x \in (a,b)$ such that*

$$f[x_0, x_1, \cdots, x_n, x] = \frac{f^{(n+1)}(\xi_x)}{(n+1)\,!}. \tag{5.25}$$

Remark 5.28. It is interesting to note that when all the nodes coincide, then (5.25) reduces to (5.19). □

Definition 5.29 (Infinity Norm).
If f is continuous on a closed interval $I = [a,b]$, then the **infinity norm** *of f, denoted by $\|f\|_{\infty,I}$, is defined as*

$$\|f\|_{\infty,I} = \max_{x \in I} |f(x)|. \tag{5.26}$$

Example 5.30. Let us obtain an upper bound of the mathematical error for the linear interpolating polynomial with respect to the infinity norm. As in Example 5.10, the linear interpolating polynomial for f at x_0 and x_1 ($x_0 < x_1$) is given by

$$p_1(x) = f(x_0) + f[x_0, x_1](x - x_0),$$

where $f[x_0, x_1]$ is given by (5.10). For each $x \in I := [x_0, x_1]$, using the formula (5.23), the error $\mathrm{ME}_1(x)$ is given by

$$\mathrm{ME}_1(x) = \frac{(x - x_0)(x - x_1)}{2}\ f''(\xi_x),$$

where $\xi_x \in (x_0, x_1)$ depends on x. Therefore,

$$|\mathrm{ME}_1(x)| \le |(x - x_0)(x - x_1)|\ \frac{\|f''\|_{\infty,I}}{2}.$$

Note that the maximum value of $|(x - x_0)(x - x_1)|$ as x varies in the interval $[x_0, x_1]$, occurs at $x = (x_0 + x_1)/2$. Therefore, we have

$$|(x - x_0)(x - x_1)| \le \frac{(x_1 - x_0)^2}{4}.$$

Using this inequality, we get an upper bound

$$|\mathrm{ME}_1(x)| \leq (x_1 - x_0)^2 \ \frac{\|f''\|_{\infty,I}}{8}, \text{ for all } x \in [x_0, x_1].$$

Note that the above inequality is true for all $x \in [x_0, x_1]$. In particular, this inequality is true for an x at with the function $|\mathrm{ME}_1|$ attains its maximum. Thus, we have

$$\|\mathrm{ME}_1\|_{\infty,I} \leq (x_1 - x_0)^2 \ \frac{\|f''\|_{\infty,I}}{8}.$$

The right hand side quantity, which is a real number, is an upper bound for the mathematical error in linear interpolation with respect to the infinity norm. □

Example 5.31. Let the function

$$f(x) = \sin x$$

be approximated by an interpolating polynomial $p_9(x)$ of degree less than or equal to 9 for f at the nodes $x_0, x_1, \cdots, x_9$ in the interval $I := [0, 1]$. Let us obtain an upper bound for $\|\mathrm{ME}_9\|_{\infty,I}$, where (from (5.23))

$$\mathrm{ME}_9(x) = \frac{f^{(10)}(\xi_x)}{10!} \prod_{i=0}^{9} (x - x_i).$$

Since $|f^{(10)}(\xi_x)| \leq 1$ and $\prod_{i=0}^{9} |x - x_i| \leq 1$, we have

$$|\sin x - p_9(x)| = |\mathrm{ME}_9(x)| \leq \frac{1}{10!} < 2.8 \times 10^{-7}, \text{ for all } x \in [0, 1].$$

Since this holds for all $x \in [0, 1]$, we have

$$\|\mathrm{ME}_9\|_{\infty,I} < 2.8 \times 10^{-7}.$$

The right hand side number is the upper bound for the mathematical error ME_9 with respect to the infinity norm on the interval I □

5.3.2 Arithmetic Error

Quite often, the polynomial interpolation that we compute is based on the function data subjected to floating-point approximation. In this subsection, we analyze the arithmetic error arising due to the floating-point approximation $\mathrm{fl}(f(x_i))$ of $f(x_i)$ for each node point x_i, $i = 0, 1, \cdots, n$ in the interval $I = [a, b]$.

The Lagrange form of interpolating polynomial that uses the values $\mathrm{fl}(f(x_i))$ instead of $f(x_i)$, $i = 0, 1, \cdots, n$ is given by

$$\tilde{p}_n(x) = \sum_{i=0}^{n} \mathrm{fl}(f(x_i)) \ l_i(x).$$

We now analyze the arithmetic error. Denoting

$$\epsilon_i := f(x_i) - \text{fl}(f(x_i); \quad ||\epsilon||_\infty = \max\{|\epsilon_i| \, : \, i = 0, 1, \cdots, n\},$$

we have

$$|\text{AE}_n(x)| = |p_n(x) - \tilde{p}_n(x)| = \left| \sum_{i=0}^{n} \Big(f(x_i) - \text{fl}(f(x_i)) \Big) \, l_i(x) \right| \leq ||\epsilon||_\infty \sum_{i=0}^{n} ||l_i||_{\infty,I},$$

for all $x \in I$. Therefore,

$$\|\text{AE}_n\|_{\infty,I} = \|p_n - \tilde{p}_n\|_{\infty,I} \leq ||\epsilon||_\infty \sum_{i=0}^{n} ||l_i||_{\infty,I}. \tag{5.27}$$

The upper bound in (5.27) might grow quite large as n increases, especially when the nodes are equally spaced as we will study now.

Assume that the nodes are equally spaced in the interval $[a, b]$, with $x_0 = a$ and $x_n = b$, and $x_{i+1} - x_i = h$ for all $i = 0, 1, \cdots, n-1$. Note that $h = (b-a)/n$. We write

$$x_i = a + ih, \quad i = 0, 1, \cdots, n.$$

Any $x \in I$ can be written as

$$x = a + \eta h,$$

where $0 \leq \eta \leq n$. Here η is not necessarily an integer. Therefore, for any $x \in I$, we have

$$l_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)} = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(\eta - i)}{(k - i)}, \quad k = 0, \cdots, n.$$

Clearly, the Lagrange polynomials are not dependent on the choice of a, b, and h. They depend entirely on n and η (which depends on x). The Figure 5.2 (a), (b) and (c) shows the function

$$l(x) = \sum_{i=0}^{n} |l_i(x)| \tag{5.28}$$

for $n = 2, 8$ and 18. It is observed that as n increases, the maximum of the function l increases. In fact, as $n \to \infty$, the maximum of l tends to infinity and it is observed in Figure 5.2 (d) which depicts n in the x-axis and the function

$$M_n = \sum_{i=0}^{n} ||l_i||_{\infty,I} \tag{5.29}$$

in the y-axis, which shows that the upper bound of the arithmetic error AE_n given in (5.27) tends to infinity as $n \to \infty$. This gives the possibility that the arithmetic may tend to increase as n increases. Thus, as we increase the degree of the interpolating polynomial, the approximation may go worser due to the presence of floating-point approximation. In fact, this behavior of the arithmetic error in polynomial interpolation can also be analyzed theoretically, but this is outside the scope of the present course.

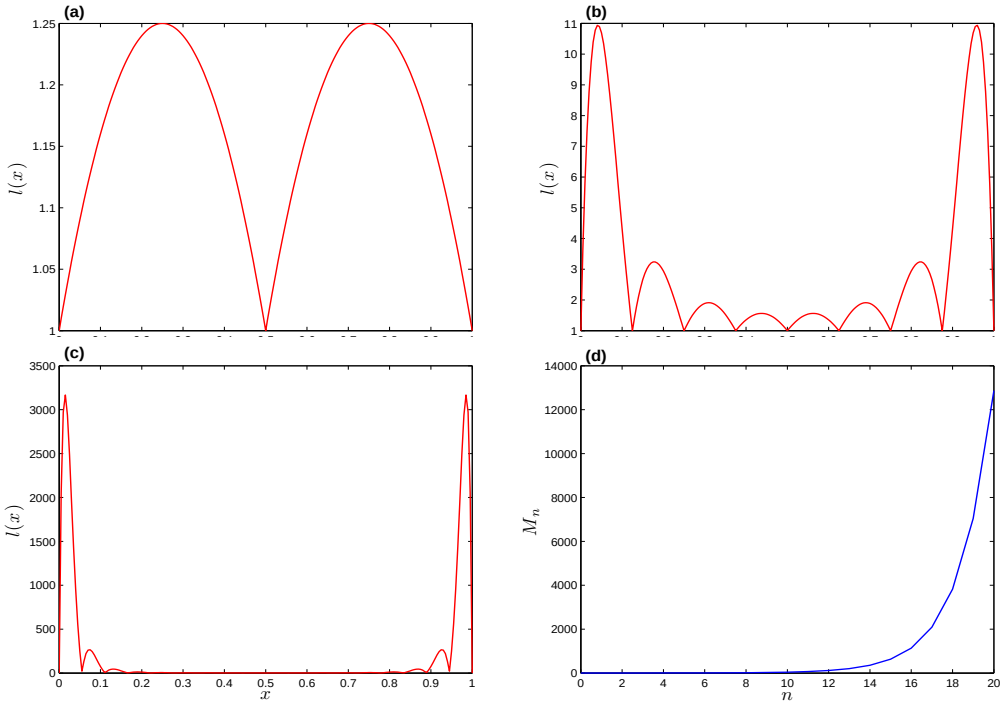

Fig. 5.2. (a) to (c) depicts the graph of function l given by (5.28) for $x \in [0, 1]$ when $n = 2, 8$, and 18. (d) depicts the function n in the x-axis and M_n given by (5.29) in the y-axis.

5.3.3 Total Error

Let us now estimate the total error, which is given by

$$\mathrm{TE}_n(x) = f(x) - \tilde{p}_n(x) = (f(x) - p_n(x)) + (p_n(x) - \tilde{p}_n(x)). \tag{5.30}$$

Taking infinity norm on both sides of the equation (5.30) and using triangle inequality, we get

$$\|\mathrm{TE}_n(x)\|_{\infty,I} = ||f - \tilde{p}||_{\infty,I} \leq ||f - p_n||_{\infty,I} + ||p_n - \tilde{p}||_{\infty,I} \leq ||f - p_n||_{\infty,I} + ||\epsilon||_\infty M_n.$$

It is clear from the Figure 5.2 (d) that M_n increases exponentially with respect to n. This implies that even if $||\epsilon||_\infty$ is very small, a large enough n can bring in a significantly large error in the interpolating polynomial.

Thus, for a given function and a set of equally spaced nodes, even if the mathematical error is bounded, the presence of floating-point approximation in the given data can lead to significantly large arithmetic error for larger values of n.

5.3.4 Runge Phenomenon

In the previous section, we have seen that even a small arithmetic error may lead to a drastic magnification of total error as we go on increasing the degree of the polynomial. This gives us a feeling that if the calculation is done with infinite precision (that is, without

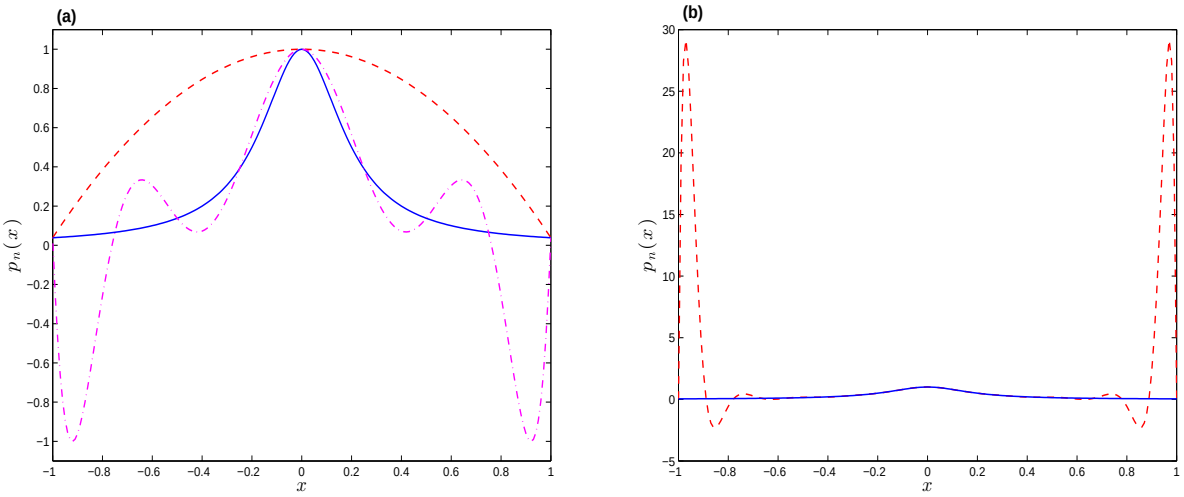

Fig. 5.3. Runge Phenomenon is illustrated. Figure (a) depicts the graph of the function f given by (5.31) (blue solid line) along with the interpolating polynomial of degree 2 (red dash line) and 8 (magenta dash dot line) with equally spaced nodes. Figure (b) shows the graph of f (blue solid line) along with the interpolating polynomial of degree 18 (red dash line) with equally spaced nodes.

any finite digit floating point arithmetic) and the function f is smooth, then we always have a better approximation for a larger value of n. In other words, we expect

$$\lim_{n\to\infty} \|f - p_n\|_{\infty,I} = 0.$$

But this is not true in the case of equally spaced nodes. This was first shown by Carl Runge, where he discovered that there are certain functions for which, as we go on increasing the degree of interpolating polynomial, the total error increases drastically and the corresponding interpolating polynomial oscillates near the boundary of the interval in which the interpolation is done. Such a phenomenon is called the **Runge Phenomenon**. This phenomenon is well understood by the following example given by Runge.

Example 5.32 (Runge's Function). Consider the **Runge's function** defined on the interval $[-1, 1]$ given by

$$f(x) = \frac{1}{1 + 25x^2}. \tag{5.31}$$

The interpolating polynomials with $n = 2$, $n = 8$ and $n = 18$ are depicted in Figure 5.3. This figure clearly shows that as we increase the degree of the polynomial, the interpolating polynomial differs significantly from the actual function especially, near the end points of the interval. □

In the light of the discussion made in Subsection 5.3.2, we may think that the Runge phenomenon is due to the amplification of the arithmetic error. But, even if the calculation is done with infinite precision (that is, without any finite digit floating point arithmetic), we may still have the Runge phenomenon due to the amplification in mathematical error. This can be observed by taking infinity norm on both sides of the formula (5.23). This gives the upper bound of the infinity norm of $\mathrm{ME}_n(x)$ as

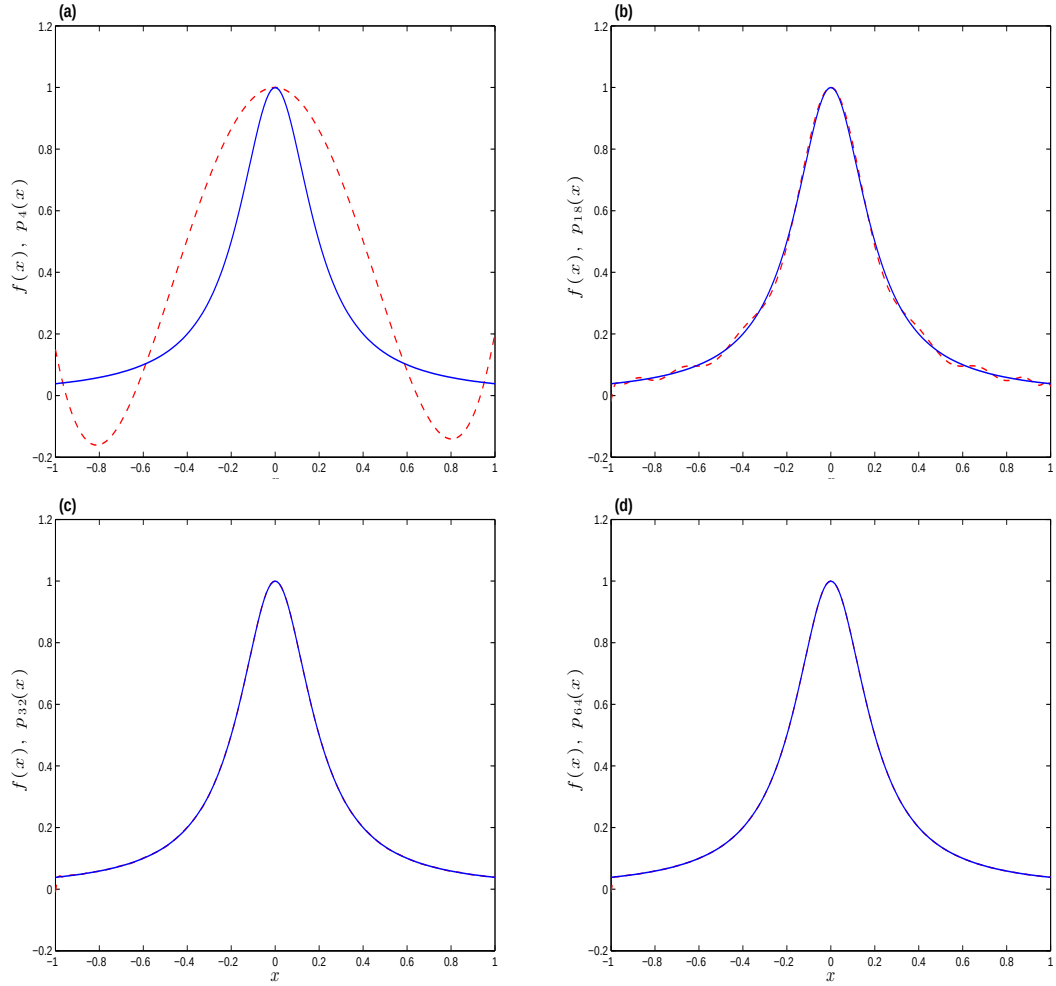

Fig. 5.4. Runge Phenomenon is illustrated with Chebyshev nodes. Figure (a) to (d) shows the graph of the Runge function (blue solid line) and the interpolating polynomial with Chebyshev nodes (red dash line) for $n = 4$, 18, 32 and 64 respectively. Note that the two graphs in Figure (c) and (d) are indistinguishable.

$$\|\mathrm{ME}_n\|_{\infty,I} \leq \frac{(b-a)^{n+1}}{(n+1)!}\|f^{(n+1)}\|_{\infty,I}.$$

Although the first part, $(b-a)^{n+1}/(n+1)!$ in the upper bound tends to zero as $n \to \infty$, if the second part, $\|f^{(n+1)}\|_{\infty,I}$ increases significantly as n increases, then the upper bound can still increase and makes it possible for the mathematical error to be quite high.

A more deeper analysis is required to understand the Runge phenomenon more rigorously. But this is outside the scope of this course and therefore is omitted.

5.3.5 Convergence of Sequence of Interpolating Polynomials

We end this section by stating without proof a positive and a negative result concerning the convergence of sequence of interpolating polynomials.

Theorem 5.33 (Faber). *For each $n \in \mathbb{N}$, let the sequence of nodes*

$$a \leq x_0^{(n)} < x_1^{(n)} < \cdots < x_n^{(n)} \leq b$$

be given. Then there exists a continuous function f defined on the interval $[a, b]$ such that the polynomials $p_n(x)$ that interpolate the function f at these nodes have the property that $\|p_n - f\|_{\infty,[a,b]}$ **does not** *tend to zero as $n \to \infty$.*

Example 5.34. In fact, the interpolating polynomial $p_n(x)$ for the Runge's function goes worser and worser as shown in Figure 5.3 for increasing values of n with equally spaced nodes. That is, $\|f - p_n\|_{\infty,[-1,1]} \to \infty$ as $n \to \infty$ for any sequence of equally spaced nodes.

Let us now state a positive result concerning the convergence of sequence of interpolating polynomials to a given continuous function.

Theorem 5.35. *Let f be a continuous function on the interval $[a, b]$. Then for each $n \in \mathbb{N}$, there exists a sequence of nodes*

$$a \leq x_0^{(n)} < x_1^{(n)} < \cdots < x_n^{(n)} \leq b$$

such that the polynomials $p_n(x)$ that interpolate the function f at these nodes satisfy $\|p_n - f\|_{\infty,[a,b]} \to 0$ as $n \to \infty$.

Example 5.36. The Theorem 5.35 is very interesting because it implies that for the Runge's function, we can find a sequence of nodes for which the corresponding interpolating polynomial yields a good approximation even for a large value of n. For instance, define a sequence of nodes

$$x_i^{(n)} = \cos\left(\frac{(2i+1)\pi}{2(n+1)}\right), i = 0, 1, \cdots, n \tag{5.32}$$

for each $n = 0, 1, 2, \cdots$. When $n = 4$, the nodes are $x_0^{(4)} = \cos(\pi/10)$, $x_1^{(4)} = \cos(3\pi/10)$, $x_2^{(4)} = \cos(5\pi/10)$, $x_3^{(4)} = \cos(7\pi/10)$ and $x_4^4 = \cos(9\pi/10)$. The nodes $x_i^{(n)}$ defined by (5.32) are called **Chebyshev nodes**.

Figure 5.4 depicts $p_n(x)$ for $n = 4$, 18, 32, and 64 along with the Runge's function. From these figures, we observe that the interpolating polynomial $p_n(x)$ agrees well with the Runge's function. □

5.4 Piecewise Polynomial Interpolation

Quite often polynomial interpolation will be unsatisfactory as an approximation tool. This is true if we insist on letting the order of the polynomial get larger and larger. However, if we keep the order of the polynomial fixed, and use different polynomials over different intervals, with the length of the intervals getting smaller and smaller, then the resulting interpolating function approximates the given function more accurately.

Let us start with linear interpolation over an interval $I = [a, b]$ which leads to

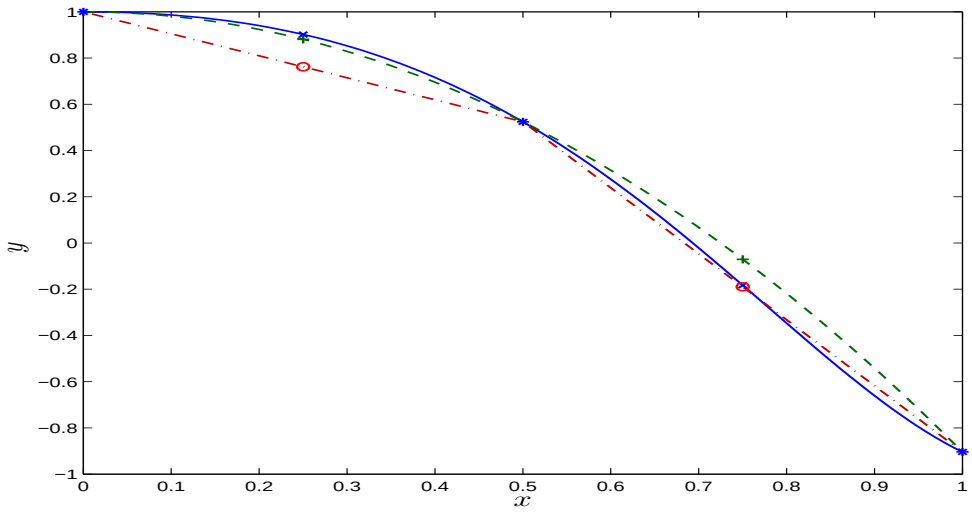

Fig. 5.5. The function $f(x) = \sin\left(\frac{\pi}{2}e^x\right)$ (blue line), $p_2(x)$ (green bash line) and the piecewise linear interpolation $s(x)$ (red dash and dot line) are shown. Blue dots represent the given data, blue 'x' symbol indicates the value of $f(0.25)$ and $f(0.75)$, green '+' symbol indicates the value of $p_2(0.25)$ and $p_2(0.75)$, and the red 'O' symbol represents the value of $s(0.25)$ and $s(0.75)$.

$$p_1(x) = f(a) + f[a,b](x-a) = f(a) + \frac{f(b)-f(a)}{b-a}(x-a) = \frac{x-b}{a-b}f(a) + \frac{x-a}{b-a}f(b),$$

where the nodes are $x_0 = a$, $x_2 = b$. In addition to these two nodes, we now choose one more point x_1 such that $x_0 < x_1 < x_2$. With these three nodes, can obtain a quadratic interpolation polynomial. Instead, we can interpolate the function $f(x)$ in $[x_0, x_1]$ by a linear polynomial with nodes x_0 and x_1, and in $[x_1, x_2]$ by another linear polynomial with nodes x_1 and x_2. Such polynomials are given by

$$p_{1,1}(x) := \frac{x-x_1}{x_0-x_1}f(x_0) + \frac{x-x_0}{x_1-x_0}f(x_1), \quad p_{1,2}(x) := \frac{x-x_2}{x_1-x_2}f(x_1) + \frac{x-x_1}{x_2-x_1}f(x_2)$$

and the interpolating function is given by

$$s(x) = \begin{cases} p_{1,1}(x) \,, x \in [x_0, x_1] \\ p_{1,2}(x) \,, x \in [x_1, x_2] \end{cases}.$$

Note that $s(x)$ is a continuous function on $[x_0, x_2]$, which interpolates $f(x)$ and is linear in $[x_0, x_1]$ and $[x_1, x_2]$. Such an interpolating function is called **piecewise linear interpolating function**.

In a similar way as done above, we can also obtain piecewise quadratic, cubic interpolating functions and so on.

Example 5.37. Consider the Example 5.1, where we have obtained the quadratic interpolating polynomial for the function

$$f(x) = \sin\left(\frac{\pi}{2}e^x\right).$$

The piecewise linear polynomial interpolating function for the data

x	0	0.5	1
$f(x)$	1.0000	0.5242	-0.9037

is given by

$$s(x) = \begin{cases} 1 - 0.9516\,x & , x \in [0, 0.5] \\ 1.9521 - 2.8558\,x & , x \in [0.5, 1]. \end{cases}$$

The following table shows the value of the function f at $x = 0.25$ and $x = 0.75$ along with the values of $p_2(x)$ and $s(x)$ with relative errors.

x	$f(x)$	$p_2(x)$	$s(x)$	$E_r(p_2(x))$	$E_r(s(x))$
0.25	0.902117	0.881117	0.762105	0.023278	0.155203
0.75	-0.182750	-0.070720	-0.189732	0.613022	0.038204

Figure 5.5 depicts the graph of f, $p_2(x)$ and $s(x)$. In this figure, we observe that the quadratic polynomial $p_2(x)$ agrees well with $f(x)$ than $s(x)$ for $x \in [0, 0.5]$, whereas $s(x)$ agrees well with $f(x)$ than $p_2(x)$ for $x \in [0.5, 1]$. □

5.5 Spline Interpolation

Although a piecewise interpolating function introduced in Section 5.4 is continuous, it may not be differentiable at the nodes. We wish to find an interpolating function that is sufficiently smooth and does a better approximation to $f(x)$. This can be achieved by **spline interpolation**.

Definition 5.38 (Spline Interpolating Function).

A **spline interpolating function of degree** d *with nodes* x_i, $i = 0, 1, \cdots, n$ ($x_0 < x_1 < \cdots < x_n$) *is a function* $s(x)$ *with the following properties*

(1) *On each subinterval* $[x_{i-1}, x_i]$, $i = 1, 2, \cdots, n$, $s(x)$ *is a polynomial of degree less than or equal to* d.

(2) $s(x)$ *and its first* $(d-1)$ *derivatives are continous on* $[x_0, x_n]$.

(3) *The interpolation conditions* $s(x_i) = f(x_i)$, $i = 0, 1, \cdots, n$ *are satisfied.*

We shall now study how we can obtain the interpolation of a function f as spline interpolating functions instead of polynomials. For the sake of simplicity, we restrict only to $d = 3$, called the **cubic spline interpolating function**.

Remark 5.39. Note that in each subinterval $[x_{i-1}, x_i]$, $i = 1, 2, \cdots, n$, we only know $f(x_{i-1})$ and $f(x_i)$. But we look for a cubic polynomial in this subinterval. Therefore, we cannot follow the Lagrange's or Newton's interpolating formula, as these formulas demand the function values at four distinct nodes in the subinterval. We need to adopt a different method for the construction of the polynomial in each subinterval in order to obtain the spline interpolation. □

Construction of a Cubic Spline

The construction of a cubic spline interpolating function $s(x)$ of a function $f(x)$ is as follows:

Step 1: Let us denote by $M_1, \cdots, M_n$,

$$M_i = s''(x_i), \quad i = 0, 1, \cdots, n$$

and first obtain $s(x)$ in terms of M_i's which are unknowns.

Step 2: Since $s(x)$ is cubic on each $[x_{i-1}, x_i]$, the function $s''(x)$ is linear on the interval such that

$$s''(x_{i-1}) = M_{i-1}, \quad s''(x_i) = M_i.$$

Therefore, it is given by

$$s''(x) = \frac{(x_i - x)M_{i-1} + (x - x_{i-1})M_i}{x_i - x_{i-1}}, \quad x_{i-1} \le x \le x_i \tag{5.33}$$

Integrating (5.33) two times with respect to x, we get

$$s(x) = \frac{(x_i - x)^3 M_{i-1}}{6(x_i - x_{i-1})} + \frac{(x - x_{i-1})^3 M_i}{6(x_i - x_{i-1})} + K_1 x + K_2,$$

where K_1 and K_2 are integrating constants to be determined by using the interpolation conditions $s(x_{i-1}) = f(x_{i-1})$ and $s(x_i) = f(x_i)$. We have

$$K_1 = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} - \frac{(M_i - M_{i-1})(x_i - x_{i-1})}{6}$$
$$K_2 = \frac{x_i f(x_{i-1}) - x_{i-1} f(x_i)}{x_i - x_{i-1}} - \frac{(M_{i-1}x_i - M_i x_{i-1})(x_i - x_{i-1})}{6}$$

Substituting these values in the above equation, we get

$$\begin{aligned} s(x) = \quad & \frac{(x_i - x)^3 M_{i-1} + (x - x_{i-1})^3 M_i}{6(x_i - x_{i-1})} + \frac{(x_i - x)f(x_{i-1}) + (x - x_{i-1})f(x_i)}{x_i - x_{i-1}} \\ & - \frac{1}{6}(x_i - x_{i-1})[(x_i - x)M_{i-1} + (x - x_{i-1})M_i], \quad x_{i-1} \le x \le x_i \end{aligned} \tag{5.34}$$

Formula (5.34) applies to each of the intervals $[x_1, x_2], \cdots, [x_{n-1}, x_n]$. The formulas for adjacent intervals $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$ will agree at their common point $x = x_i$ because of the interpolating condition $s(x_i) = f(x_i)$. This implies that $s(x)$ is continuous over the entire interval $[a, b]$. Similarly, formula (5.33) for $s''(x)$ implies that it is continuous on $[a, b]$.

Step 3: All that remains is to find the values of M_i for all $i = 0, 1, \cdots, n$. This is obtained by ensuring the continuity of $s'(x)$ over $[a, b]$, ie., the formula for $s'(x)$ on $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$ are required to give the same value at their common point $x = x_i$, for $i = 1, 2, \cdots, n-1$. After simplification, we get the system of linear equations for $i = 1, 2, \cdots n-1$

$$\frac{x_i - x_{i-1}}{6} M_{i-1} + \frac{x_{i+1} - x_{i-1}}{3} M_i + \frac{x_{i+1} - x_i}{6} M_{i+1} = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} - \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}. \quad (5.35)$$

These $n-1$ equations together with the assumption that

$$M_0 = M_n = 0 \quad (5.36)$$

leads to the values of M_0, M_1, $\cdots$, M_n and hence to the interpolation function $s(x)$.

A spline constructed above is called a **natural spline**.

Example 5.40. Calculate the natural cubic spline interpolating the data

$$\left\{(1,1), \left(2, \frac{1}{2}\right), \left(3, \frac{1}{3}\right), \left(4, \frac{1}{4}\right)\right\}.$$

The number of points is $n = 4$ and all $x_i - x_{i-1} = 1$.
Step 1: Here, we have

$$M_0 = s''(1),\ M_1 = s''(2),\ M_2 = s''(3),\ M_3 = s''(4).$$

Step 2: The function $s''(x)$ is given by

$$s''(x) = \begin{cases} (2-x)M_0 + (x-1)M_1\ , x \in [1,2] \\ (3-x)M_1 + (x-2)M_2\ , x \in [2,3] \\ (4-x)M_2 + (x-3)M_3\ , x \in [3,4] \end{cases}.$$

Integrating $s''(x)$ two times with respect to x, we get

$$s(x) = \begin{cases} \dfrac{(2-x)^3 M_0}{6} + \dfrac{(x-1)^3 M_1}{6} + K_1 x + K_2\ , x \in [1,2] \\ \dfrac{(3-x)^3 M_1}{6} + \dfrac{(x-2)^3 M_2}{6} + K_1 x + K_2\ , x \in [2,3] \\ \dfrac{(4-x)^3 M_2}{6} + \dfrac{(x-3)^3 M_3}{6} + K_1 x + K_2\ , x \in [3,4] \end{cases}.$$

where K_1 and K_2 are integrating constants to be determined by using the conditions $s(x_{i-1}) = f(x_{i-1})$ and $s(x_i) = f(x_i)$. We have

$$K_1 = \begin{cases} -\dfrac{1}{2} - \dfrac{(M_1 - M_0)}{6}\ , x \in [1,2] \\ -\dfrac{1}{6} - \dfrac{(M_2 - M_1)}{6}\ , x \in [2,3] \\ -\dfrac{1}{12} - \dfrac{(M_3 - M_2)}{6}\ , x \in [3,4] \end{cases}.$$

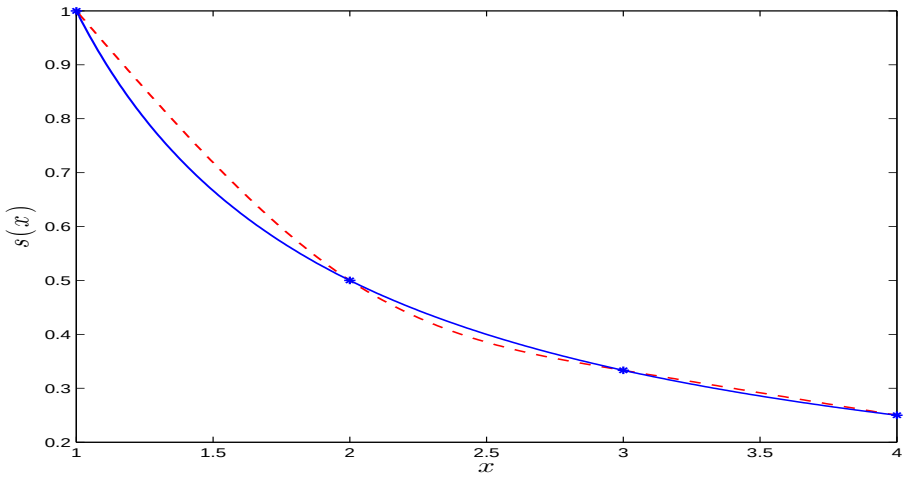

Fig. 5.6. The function $f(x) = 1/x$ (blue line) and the corresponding cubic natural spline interpolating function $s(x)$ (red dash line) are shown. The data are represented by blue dots.

$$K_2 = \begin{cases} \dfrac{3}{2} - \dfrac{2M_0 - M_1}{6} & , x \in [1,2] \\ \dfrac{5}{6} - \dfrac{3M_1 - 2M_2}{6} & , x \in [2,3] \\ \dfrac{7}{12} - \dfrac{4M_2 - 3M_3}{6} & , x \in [3,4] \end{cases}.$$

Substituting these expressions in the expression of $s(x)$, we get the required cubic spline as given in (5.34).

Step 3: Since we are constructing the natural spline, we take $M_0 = M_3 = 0$. The system (5.35) gives

$$\begin{aligned} \frac{2}{3}M_1 + \frac{1}{6}M_2 &= \frac{1}{3}, \\ \frac{1}{6}M_1 + \frac{2}{3}M_2 &= \frac{1}{12}. \end{aligned}$$

Solving this system ,we get $M_1 = \frac{1}{2}, \quad M_2 = 0$. Substituting these values into (5.34), we obtain

$$s(x) = \begin{cases} \dfrac{1}{12}x^3 - \dfrac{1}{4}x^2 - \dfrac{1}{3}x + \dfrac{3}{2} & , x \in [1,2] \\ -\dfrac{1}{12}x^3 + \dfrac{3}{4}x^2 - \dfrac{7}{3}x + \dfrac{17}{6} & , x \in [2,3] \\ -\dfrac{1}{12}x + \dfrac{7}{12} & , x \in [3,4] \end{cases}$$

which is the required natural cubic spline approximation to the given data. A comparison result is depicted in Figure 5.6. □

CHAPTER 6

Numerical Integration and Differentiation

There are two reasons for approximating derivatives and integrals of a function $f(x)$. One is when the function is very difficult to differentiate or integrate, or only the tabular values are available for the function. Another reason is to obtain solution of a differential or integral equation. In this chapter we introduce some basic methods to approximate integral and derivative of a function either explicitly or by tabulated values.

In Section 6.1, we obtain numerical methods for evaluating the integral of a given integrable function f defined on the interval $[a, b]$. Section 6.2 introduces various ways to obtain numerical formulas for approximating derivatives of a given differentiable function.

6.1 Numerical Integration

In this section we derive and analyze numerical methods for evaluating definite integrals. The problem is to evaluate the number

$$I(f) = \int_a^b f(x)dx. \tag{6.1}$$

Most such integrals cannot be evaluated explicitly, and with many others, it is faster to integrate numerically than explicitly. The process of approximating the value of $I(f)$ is usually referred to as **numerical integration** or **quadrature**.

The idea behind numerical integration is to approximate the integrand f by a simpler function that can be integrated easily. One obvious approximation is the interpolation by polynomials. Thus, we approximate $I(f)$ by $I(p_n)$, where $p_n(x)$ is the interpolating polynomial for the integrand f at some appropriately chosen nodes $x_0, \cdots, x_n$. The general form of the approximation is

$$I(f) \approx I(p_n) = w_0 f(x_0) + w_1 f(x_1) + \cdots + w_n f(x_n),$$

where the **weights** are given by

$$w_i = I(l_i),$$

with $l_i(x)$ the i^{th} Lagrange polynomial.

Without going through interpolation, now propose a general formula

$$I(f) \approx w_0 f(x_0) + w_1 f(x_1) + \cdots + w_n f(x_n), \tag{6.2}$$

where $x_0, \cdots, x_n$ are distinct real numbers (called **quadrature points**) and $w_0, \cdots, w_n$ are real numbers (called **quadrature weights**). When the quadrature points are equally spaced, the quadrature formula of the form (6.2) is called the **Newton-Cotes formula**.

The Newton-Cotes formula (6.2) gives rise to different quadrature formulas depending on the degree of the interpolating polynomial n and also on the choice of the nodes. We now study few simple quadrature formulas.

6.1.1 Rectangle Rule

We now consider the case when $n = 0$. Then, the corresponding interpolating polynomial is the constant function $p_0(x) = f(x_0)$, and therefore

$$I(p_0) = (b - a) f(x_0).$$

From this, we can obtain two quadrature rules depending on the choice of x_0.

- If $x_0 = a$, then this approximation becomes

  $$I(f) \approx I_R(f) := (b - a) f(a) \tag{6.3}$$

  and is called the **rectangle rule**. The geometrical interpretation of the rectangle rule is illustrated in Figure 6.1.

- If $x_0 = (a + b)/2$, we get

  $$I(f) \approx I_M(f) := (b - a) f\left(\frac{a + b}{2}\right) \tag{6.4}$$

  and is called the **mid-point rule**.

  We now obtain the mathematical error in rectangle rule, given by,

  $$\mathrm{ME}_{\mathrm{R}}(f) := I(f) - I(p_0).$$

Theorem 6.1 (Error in Rectangle Rule). *Let $f \in C^1[a, b]$. The mathematical error $\mathrm{ME}_{\mathrm{R}}(f)$ of the rectangle rule takes the form*

$$\mathrm{ME}_{\mathrm{R}}(f) = \frac{f'(\eta)(b - a)^2}{2}, \tag{6.5}$$

for some $\eta \in (a, b)$.

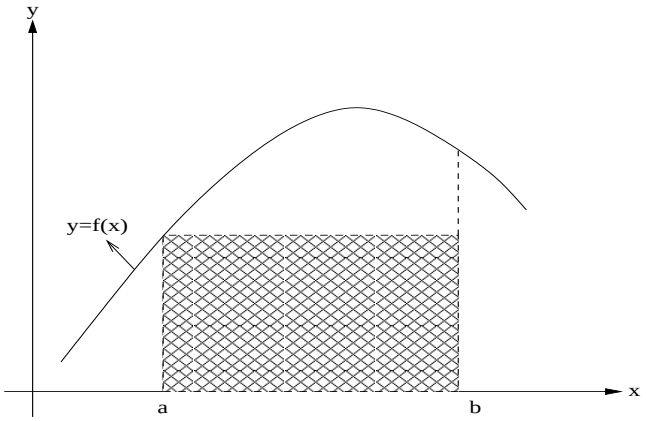

Fig. 6.1. Geometrical Interpretation of the Rectangle Rule.

Proof: For each $x \in (a, b]$, the linear interpolating polynomial for f at the nodes a and x is given by

$$f(x) = p_0(x) + f[a, x](x - a).$$

Therefore, the mathematical error in the rectangle rule is given by

$$\mathrm{ME_R}(f) = I(f) - I_R(f) = \int_a^b f[a, x](x - a)\, dx.$$

Using mean-value theorem for integrals, we get

$$\mathrm{ME_R}(f) = f[a, \xi] \int_a^b (x - a)\, dx,$$

for some $\xi \in (a, b)$. By mean value theorem for derivatives, $f[a, \xi] = f'(\eta)$ for some $\eta \in (a, \xi)$. Thus, we get

$$\mathrm{ME_R}(f) = \frac{f'(\eta)(b - a)^2}{2},$$

for some $\eta \in (a, b)$. □

6.1.2 Trapezoidal Rule

We now consider the case when $n = 1$. Then

$$p_1(x) = f(x_0) + f[x_0, x_1](x - x_0),$$

and therefore

$$I(f) \approx I_T(f) := \int_a^b \left(f(x_0) + f[x_0, x_1](x - x_0)\right)\, dx.$$

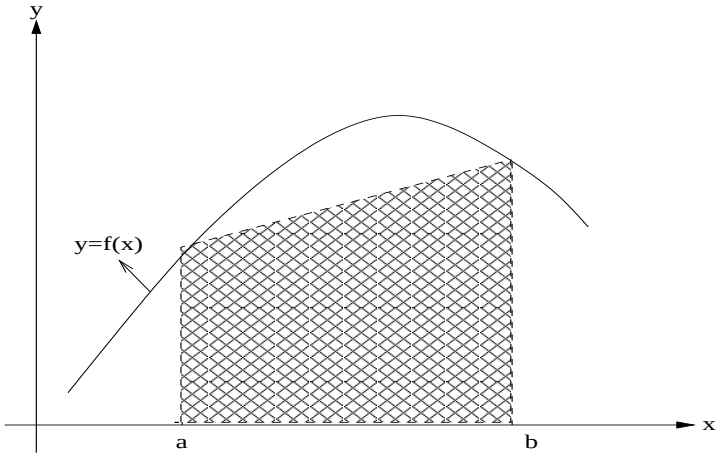

Fig. 6.2. Trapezoidal Rule.

Taking $x_0 = a$ and $x_1 = b$, we get

$$I_T(f) = (b-a)\left(\frac{f(a)+f(b)}{2}\right) \tag{6.6}$$

and is called the **Trapezoidal Rule**. The Trapezoidal rule is illustrated in Figure 6.2. We now obtain the mathematical error in Trapezoidal rule, given by

$$\mathrm{ME_T}(f) := I(f) - I(p_1).$$

Theorem 6.2 (Error in Trapezoidal Rule). *Let $f \in C^2[a,b]$. The mathematical error $\mathrm{ME_T}(f)$ of the trapezoidal rule takes the form*

$$\mathrm{ME_T}(f) = -\frac{f''(\eta)(b-a)^3}{12}, \tag{6.7}$$

for some $\eta \in (a,b)$.

Proof: We have

$$f(x) = f(a) + f[a,b](x-a) + f[a,b,x](x-a)(x-b).$$

Integrating over the interval $[a,b]$, we get

$$I(f) = I_T(f) + \int_a^b f[a,b,x](x-a)(x-b)dx.$$

Therefore the mathematical error is given by

$$\mathrm{ME_T}(f) = I(f) - I_T(f) = \int_a^b f[a,b,x](x-a)(x-b)dx. \tag{6.8}$$

From Corollary 5.24 (Conclusion 1), we see that the function $x \longmapsto f[a,b,x]$ is continuous. Therefore, from the mean value theorem for integrals (after noting that $(x-a)(x-b)$ is non-negative for all $x \in [a,b]$), the expression (6.8) for the mathematical error takes the form

$$\mathrm{ME}_{\mathrm{T}}(f) = f[a,b,\eta] \int_a^b (x-a)(x-b)dx,$$

for some $\eta \in (a,b)$. The formula (6.7) now follows from (5.25) and a direct evaluation of the above integral. □

Example 6.3. For the function $f(x) = 1/(x+1)$, we approximate the integral

$$I = \int_0^1 f(x)dx,$$

using trapezoidal rule to get

$$I_T(f) = \frac{1}{2}\left(1 + \frac{1}{2}\right) = \frac{3}{4} = 0.75.$$

The true value is $I(f) = \log(2) \approx 0.693147$. Therefore, the error is $\mathrm{ME}_{\mathrm{T}}(f) \approx -0.0569$. Using the formula (6.7), we get the bounds for $\mathrm{ME}_{\mathrm{T}}(f)$ as

$$-\frac{1}{6} < \mathrm{ME}_{\mathrm{T}}(f) < -\frac{1}{48}$$

which clearly holds in the present case. □

Composite Trapezoidal Rule

We can improve the approximation of trapezoidal rule by breaking the interval $[a,b]$ into smaller subintervals and apply the trapezoidal rule (6.6) on each subinterval. We will derive a general formula for this.

Let us subdivide the interval $[a,b]$ into n equal subintervals of length

$$h = \frac{b-a}{n}$$

with endpoints of the subintervals as

$$x_j = a + jh, \quad j = 0, 1, \cdots, n.$$

Then, we get

$$I(f) = \int_a^b f(x)dx = \int_{x_0}^{x_n} f(x)dx = \sum_{j=0}^{n-1} \int_{x_j}^{x_{j+1}} f(x)dx.$$

Using Trapezoidal rule (6.6) on the subinterval $[x_j, x_{j+1}]$, we get

$$\int_{x_j}^{x_{j+1}} f(x)dx \approx h\left(\frac{f(x_j)+f(x_{j+1})}{2}\right),\ j=0,1,\cdots,n-1.$$

Substituting this in the above equation, we get

$$I(f) \approx h\left[\frac{f(x_0)+f(x_1)}{2}\right] + h\left[\frac{f(x_1)+f(x_2)}{2}\right] + \cdots + h\left[\frac{f(x_{n-1})+f(x_n)}{2}\right].$$

The terms on the right hand side can be combined to give the simpler formula

$$I_T^n(f) := h\left[\frac{1}{2}f(x_0) + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + \frac{1}{2}f(x_n)\right]. \tag{6.9}$$

This rule is called the **Composite Trapezoidal rule**.

Example 6.4. Using composite trapezoidal rule with $n = 2$, let us approximate the integral

$$I = \int_0^1 f(x)dx,$$

where

$$f(x) = \frac{1}{1+x}.$$

As we have seen in Example 6.3, the true value is $I(f) = \log(2) \approx 0.693147$. Now, the composite trapezoidal rule with $x_0 = 0$, $x_1 = 1/2$ and $x_2 = 1$ gives

$$I_T^2(f) \approx 0.70833.$$

Thus the error is -0.0152. Recall from Example 6.3 that with $n = 1$, the trapezoidal rule gave an error of -0.0569. □

6.1.3 Simpson's Rule

We now calculate $I(p_2(x))$ to obtain the formula for the case when $n = 2$. Let us choose $x_0 = a$, $x_1 = (a+b)/2$ and $x_2 = b$. The Lagrange form of interpolating polynomial is

$$p_2(x) = f(x_0)l_0(x) + f(x_1)l_1(x) + f(x_2)l_2(x).$$

Then

$$\int_a^b p_2(x)dx = f(x_0)\int_a^b l_0(x)\,dx + f(x_1)\int_a^b l_1(x)\,dx + f(x_2)\int_a^b l_2(x)\,dx.$$

Using the change of variable

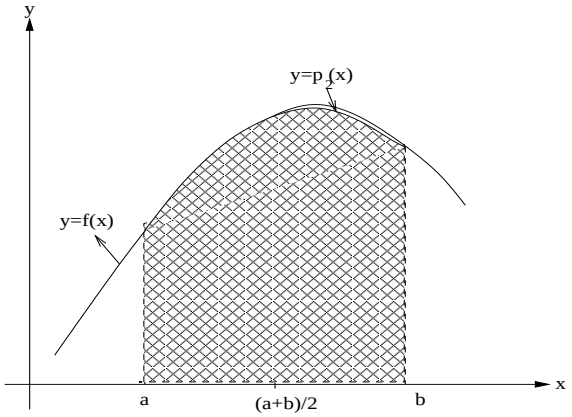

Fig. 6.3. Simpson Rule.

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

we get

$$\begin{aligned}\int_a^b l_0(x)\,dx &= \int_a^b \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}\,dx \\ &= \int_{-1}^{1} \frac{t(t-1)}{2}\frac{b-a}{2}\,dt \\ &= \frac{b-a}{6}.\end{aligned}$$

Similarly, we can see

$$\int_a^b l_1(x)\,dx = \frac{4}{6}(b-a),$$
$$\int_a^b l_2(x)\,dx = \frac{b-a}{6}.$$

We thus arrive at the formula

$$I(f) \approx I_S(f) := \int_a^b p_2(x)dx = \frac{b-a}{6}\left\{f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right\} \tag{6.10}$$

which is the famous **Simpson's Rule**. The Simpson's rule is illustrated in Figure 6.3.

We now obtain the mathematical error in Simpson's rule, given by,

$$\mathrm{ME_S}(f) := I(f) - I(p_2).$$

Theorem 6.5 (Error in Simpson's Rule). *Let $f \in C^4[a,b]$. The mathematical error $\mathrm{ME_S}(f)$ of the Simpson's rule takes the form*

$$\mathrm{ME_S}(f) = -\frac{f^{(4)}(\eta)(b-a)^5}{2880}, \tag{6.11}$$

for some $\eta \in (a,b)$.

Proof: Consider the collection of nodes $a,\ (a+b)/2,\ b,\ x$ in the interval $[a,b]$. Let $x_0 \in [a,b]$ be a given point different from a, $(a+b)/2$, b, and x. From the fourth order divided difference formula, we have

$$f[a,(a+b)/2,b,x] = f[x_0,a,(a+b)/2,b] + f[x_0,a,(a+b)/2,b,x](x-x_0).$$

The mathematical error in Simpson's method can therefore be written as

$$\mathrm{ME_S}(f) = \int_a^b f[x_0,a,(a+b)/2,b]\phi(x)dx + \int_a^b f[x_0,a,(a+b)/2,b,x](x-x_0)\phi(x),$$

where

$$\phi(x) = (x-a)(x-(a+b)/2)(x-b).$$

A direct integration shows

$$\int_a^b \phi(x)dx = \int_a^b (x-a)\left(x-\frac{a+b}{2}\right)(x-b)dx = 0.$$

Thus, we have

$$\mathrm{ME_S}(f) = \int_a^b f[x_0,a,(a+b)/2,b,x](x-x_0)\phi(x),$$

Recall in the proof of theorem 6.2, we used mean value theorem 1.27 at this stage to arrive at the conclusion. But, in the present case, we cannot use this theorem because the function $(x-x_0)\phi(x)$ need not be of one sign on $[a,b]$. However, the choice $x_0 = (a+b)/2$ makes this function one signed (non-negative). Now using Corollary 5.24 (Conclusion 1) and following the idea of the proof of Theorem 6.2, we arrive at the formula (6.11) (this is left as an exercise). □

Example 6.6. We now use the Simpson's rule to approximate the integral

$$I(f) = \int_0^1 f(x)\,dx,$$

where

$$f(x) = \frac{1}{1+x}.$$

The true value is $I(f) = \log(2) \approx 0.693147$. Using the Simpson's rule (6.10), we get

$$I_S(f) = \frac{1}{6}\left(1 + \frac{8}{3} + \frac{1}{2}\right) = \frac{25}{36} \approx 0.694444.$$

Therefore, the error is $\mathrm{ME_S}(f) \approx 0.001297$. □

Composite Simpson's Rule

Let us now derive the **composite Simpson's rule**. First let us subdivide the interval $[a, b]$ into n equal parts as we discussed in composite trapezoidal rule (Subsection 6.1.2). Taking $a = x_{i-1}$, $b = x_i$, $x_{i-1/2} = (x_i + x_{i-1})/2$ and $x_i - x_{i-1} = h$ in Simpson rule, we get

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx \frac{h}{6}\left\{f(x_{i-1}) + 4f(x_{i-1/2}) + f(x_i)\right\}.$$

Summing for $i = 1, \cdots, n$, we get

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)dx \\ &\approx \frac{h}{6}\sum_{i=1}^{n}\left\{f(x_{i-1}) + 4f(x_{i-1/2}) + f(x_i)\right\}.\end{aligned}$$

Therefore, the **composite Simpson's rule** takes the form

$$I_S^n(f) = \frac{h}{6}\left[f(x_0) + f(x_n) + 2\sum_{i=1}^{n-1} f(x_i) + 4\sum_{i=1}^{n} f(x_{i-1/2})\right] \tag{6.12}$$

6.1.4 Method of Undetermined Coefficients

All the rules so far derived are of the form

$$I(f) = \int_a^b f(x)dx \approx w_0 f(x_0) + w_1 f(x_1) + \cdots + w_n f(x_n) \tag{6.13}$$

where w_i's are weights. In deriving those rules, we have fixed the nodes x_0, x_1, $\cdots$, x_n ($n = 0$ for rectangle rule, $n = 1$ for trapezoidal rule and $n = 2$ for Simpson's rule), and used interpolating polynomials to obtain the corresponding weights. Instead, we may use another approach in which for a fixed set of nodes weights are determined by imposing the condition that the resulting rule is exact for polynomials of degree less than or equal to n. Such a method is called the **method of undetermined coefficients**.

Example 6.7. Let us find w_0, w_1, and w_2 such that the approximate formula

$$\int_a^b f(x)\,dx \approx w_0 f(a) + w_1 f\left(\frac{a+b}{2}\right) + w_2 f(b) \tag{6.14}$$

is exact for all polynomials of degree less than or equal to 2.

Since integral of sum of functions is the sum of the integrals of the respective functions, the formula (6.14) is exact for all polynomials of degree less than or equal to 2 if and only if the formula (6.14) is exact for the polynomials $1, x$, and x^2.

- The condition that the formula (6.14) is exact for the polynomial $p(x) = 1$ yields

$$b - a = \int_a^b 1\,dx = w_0 + w_1 + w_2$$

- The condition that the formula (6.14) is exact for the polynomial $p(x) = x$ yields

$$\frac{b^2 - a^2}{2} = \int_a^b x\,dx = aw_0 + \left(\frac{a+b}{2}\right) w_1 + bw_2.$$

- The condition that the formula (6.14) is exact for the polynomial $p(x) = x^2$ yields

$$\frac{b^3 - a^3}{3} = \int_a^b x^2\,dx = a^2 w_0 + \left(\frac{a+b}{2}\right)^2 w_1 + b^2 w_2$$

Thus, we have a linear system of three equations satisfied by w_0, w_1, and w_2. By solving this system, we get

$$w_0 = \frac{1}{6}(b-a),$$

$$w_1 = \frac{2}{3}(b-a),$$

$$w_2 = \frac{1}{6}(b-a),$$

which gives us the familiar Simpson's rule. □

This motivates the following definition.

Definition 6.8 (Degree of Precision).

The **degree of precision** *(also called* **order of exactness***) of a quadrature formula is the largest positive integer n such that the formula is exact for all polynomials of degree less than or equal to n.*

Example 6.9. Let us determine the degree of precision of Simpson's rule. It will suffice to apply the rule over the interval $[0, 2]$ (in fact any interval is good enough and we chose this interval for the sake of having easy computation).

$$\begin{aligned}
\int_0^2 dx &= 2 = \frac{2}{6}(1+4+1),\\
\int_0^2 x dx &= 2 = \frac{2}{6}(0+4+2),\\
\int_0^2 x^2 dx &= \frac{8}{3} = \frac{2}{6}(0+4+4),\\
\int_0^2 x^3 dx &= 4 = \frac{2}{6}(0+4+8),\\
\int_0^2 x^4 dx &= \frac{32}{5} \neq \frac{2}{6}(0+4+16) = \frac{20}{3}.
\end{aligned}$$

Therefore, the degree of precision of Simpson's rule is 3. □

Remark 6.10. In Example 6.7 we have obtained the Simpson's rule using the method of undetermined coefficients by requiring the exactness of the rule for polynomials of degree less than or equal to 2, the above example shows that the rule is exact for polynomials of degree three as well. □

6.1.5 Gaussian Rules

In Example 6.7 we have fixed the nodes and obtained the weights in the quadrature rule (6.14) such that the rule is exact for polynomials of degree less than or equal to 2. In general, by fixing the nodes, we can obtain the weights in (6.13) such that the rule is exact for polynomials of degree less than or equal to n. But it is also possible to derive a quadrature rule such that the rule is exact for polynomials of degree less than or equal to $2n+1$ by choosing the $n+1$ nodes and the weights appropriately. This is the basic idea of **Gaussian rules**.

Let us consider the special case

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i). \tag{6.15}$$

The weights w_i and the nodes x_i $(i = 0, \cdots, n)$ are to be chosen in such a way that the rule (6.15) is exact, that is

$$\int_{-1}^{1} f(x)dx = \sum_{i=0}^{n} w_i f(x_i), \tag{6.16}$$

whenever $f(x)$ is a polynomial of degree less than or equal to $2n+1$. Note that (6.16) holds for every polynomial $f(x)$ of degree less than or equal to $2n+1$ if and only if (6.16) holds for $f(x) = 1, x, x^2, \cdots, x^{2n+1}$.

Case 1: $(n = 0)$. In this case, the quadrature formula (6.15) takes the form

$$\int_{-1}^{1} f(x)dx \approx w_0 f(x_0).$$

The condition (6.16) gives

$$\int_{-1}^{1} 1\, dx = w_0 \text{ and } \int_{-1}^{1} x\, dx = w_0 x_0.$$

These conditions give $w_0 = 2$ and $x_0 = 0$. Thus, we have the formula

$$\int_{-1}^{1} f(x)dx \approx 2f(0) =: I_{G_0}(f), \tag{6.17}$$

which is the required Gaussian rule for $n = 0$.

Case 2: $(n = 1)$. In this case, the quadrature formula (6.15) takes the form

$$\int_{-1}^{1} f(x)dx \approx w_0 f(x_0) + w_1 f(x_1).$$

The condition (6.16) gives

$$\begin{aligned} w_0 + w_1 &= 2, \\ w_0 x_0 + w_1 x_1 &= 0, \\ w_0 x_0^2 + w_1 x_1^2 &= \frac{2}{3}, \\ w_0 x_0^3 + w_1 x_1^3 &= 0. \end{aligned}$$

A solution of this nonlinear system is $w_0 = w_1 = 1$, $x_0 = -1/\sqrt{3}$ and $x_1 = 1/\sqrt{3}$. This lead to the formula

$$\int_{-1}^{1} f(x)dx \approx f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) =: I_{G1}(f), \tag{6.18}$$

which is the required Gaussian rule for $n = 1$.

Case 3: (General). In general, the quadrature formula is given by (6.15), where there are $2(n+1)$ free parameters x_i and w_i for $i = 0, 1, \cdots, n$. The condition (6.16) leads to the nonlinear system

$$\sum_{j=0}^{n} w_j x_j^i = \begin{cases} 0 & , \ i = 1, 3, \cdots, 2n+1 \\ \dfrac{2}{i+1} & , \ i = 0, 2, \cdots, 2n \end{cases}.$$

These are nonlinear equations and their solvability is not at all obvious and therefore the discussion is outside the scope of this course.

So far, we derived Gaussian rule for integrals over $[-1, 1]$. But this is not a limitation as any integral on the interval $[a, b]$ can easily be transformed to an integral on $[-1, 1]$ by using the linear change of variable

$$x = \frac{b + a + t(b-a)}{2}, \quad -1 \le t \le 1. \tag{6.19}$$

Thus, we have

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

Example 6.11. We now use the Gaussian rule to approximate the integral

$$I(f) = \int_0^1 f(x)\, dx,$$

where

$$f(x) = \frac{1}{1+x}.$$

Note that the true value is $I(f) = \log(2) \approx 0.693147$.

To use the Gaussian quadrature, we first need to make the linear change of variable (6.19) with $a = 0$ and $b = 1$ and we get

$$x = \frac{t}{2}, \quad -1 \le t \le 1.$$

Thus the required integral is

$$I(f) = \int_0^1 \frac{dx}{1+x} = \int_{-1}^{1} \frac{dt}{3+t}.$$

We need to take $f(t) = 1/(3+t)$ in the Gaussian quadrature formula (6.18) and we get

$$\int_0^1 \frac{dx}{1+x} = \int_{-1}^{1} \frac{dt}{3+t} \approx f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) \approx 0.692308 \approx I_{G1}(f).$$

Therefore, the error is $I(f) - I_{G1}(f) \approx 0.000839$. □

6.2 Numerical Differentiation

The aim of this section is to obtain formulas to approximate the values of derivatives of a given function at a given point. Such a formula for the first derivative of a function can be obtained directly using the definition of the derivative, namely, the difference quotients of the function. This will be discussed in Subsection 6.2.1. But this idea cannot be adopted for higher order derivatives. Approximating formulas for derivatives can be obtained in at least two ways, namely,

(1) Methods based on Interpolation

(2) Methods based on Undetermined Coefficients

These methods are discussed in Subsections 6.2.2 and 6.2.3, respectively.

6.2.1 Approximations of First Derivative

Forward Difference Formula

The most simple way to obtain a numerical method for approximating the derivative of a C^1 function f is to use the definition of derivative

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

The approximating formula can therefore be taken as

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} =: D_h^+ f(x) \tag{6.20}$$

for a sufficiently small value of $h > 0$. The formula $D_h^+ f(x)$ is called the **forward difference formula** for the derivative of f at the point x.

Theorem 6.12. *Let $f \in C^2[a,b]$. The mathematical error in the forward difference formula is given by*

$$f'(x) - D_h^+ f(x) = -\frac{h}{2} f''(\eta) \tag{6.21}$$

for some $\eta \in (x, x+h)$.

Proof: By Taylor's theorem, we have

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(\eta) \tag{6.22}$$

for some $\eta \in (x, x+h)$. Using (6.20) and (6.22), we obtain

$$D_h^+ f(x) = \frac{1}{h}\left\{\left[f(x) + hf'(x) + \frac{h^2}{2} f''(\eta)\right] - f(x)\right\} = f'(x) + \frac{h}{2} f''(\eta).$$

This completes the proof. □

Remark 6.13. If we consider the left hand side of (6.21) as a function of h, *i.e.,* if

$$g(h) = f'(x) - D_h f(x),$$

then we see that

$$\left|\frac{g(h)}{h}\right| = \frac{1}{2}|f''(\eta)|.$$

Let $M > 0$ be such that $|f''(x)| \leq M$ for all $x \in [a, b]$. Then we see that

$$\left|\frac{g(h)}{h}\right| \leq \frac{M}{2}.$$

That is, $g = O(h)$ as $h \to 0$. We say that the forward difference formula $D_h^+ f(x)$ is of order 1 (order of accuracy). □

Backward Difference Formula

The derivative of a function f is also given by

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

Therefore, the approximating formula for the first derivative of f can also be taken as

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} =: D_h^- f(x) \tag{6.23}$$

The formula $D_h^- f(x)$ is called the **backward difference formula** for the derivative of f at the point x.

Deriving the mathematical error for backward difference formula is similar to that of the forward difference formula. It can be shown that the backward difference formula is of order 1.

Central Difference Formula

The derivative of a function f is also given by

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}.$$

Therefore, the approximating formula for the first derivative of f can also be taken as

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} =: D_h^0 f(x), \tag{6.24}$$

for a sufficiently small value of $h > 0$. The formula $D_h^0 f(x)$ is called the **central difference formula** for the derivative of f at the point x.

The central difference formula is of order 2 as shown in the following theorem.

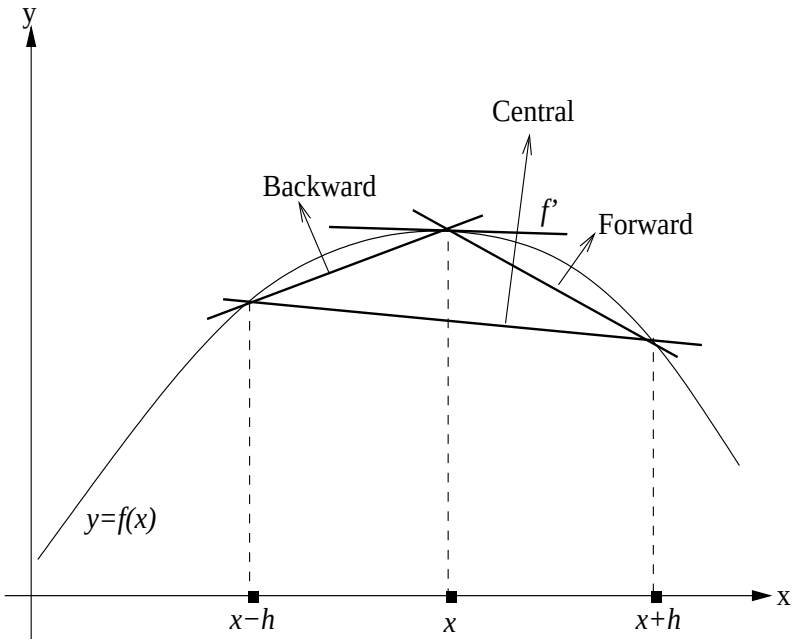

Fig. 6.4. Geometrical interpretation of difference formulae.

Theorem 6.14. *Let $f \in C^3[a,b]$. The mathematical error in the central difference formula is given by*

$$f'(x) - D_h^0 f(x) = -\frac{h^2}{6} f'''(\eta), \tag{6.25}$$

where $\eta \in (x-h, x+h)$.

Proof: Using Taylor's theorem, we have

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f'''(\eta_1)$$

and

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!} f''(x) - \frac{h^3}{3!} f'''(\eta_2),$$

where $\eta_1 \in (x, x+h)$ and $\eta_2 \in (x-h, x)$.

Therefore, we have

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{h^3}{3!}(f'''(\eta_1) + f'''(\eta_2)).$$

Since $f'''(x)$ is continuous, by intermediate value theorem applied to f'', we have

$$\frac{f'''(\eta_1) + f'''(\eta_2)}{2} = f'''(\eta)$$

for some $\eta \in (x-h, x+h)$. This completes the proof. □

Geometric interpretation of the three primitive difference formulae (forward, backward, and central) is shown in Figure 6.4.

Example 6.15. To find the value of the derivative of the function given by $f(x) = \sin x$ at $x = 1$ with $h = 0.003906$, we use the three primitive difference formulas. We have

$$\begin{aligned} f(x-h) &= f(0.996094) = 0.839354, \\ f(x) &= f(1) = 0.841471, \\ f(x+h) &= f(1.003906) = 0.843575. \end{aligned}$$

(1) Backward difference: $D_h^- f(x) = \dfrac{f(x) - f(x-h)}{h} = 0.541935.$

(2) Central Difference: $D_h^0 f(x) = \dfrac{f(x+h) - f(x-h)}{2h} = 0.540303.$

(3) Forward Difference: $D_h^+ f(x) = \dfrac{f(x+h) - f(x)}{h} = 0.538670.$

Note that the exact value is $f'(1) = \cos 1 = 0.540302$.

6.2.2 Methods based on Interpolation

Using the polynomial interpolation we can obtain formula for derivatives of any order for a given function. For instance, to calculate $f'(x)$ at some point x, we use the approximate formula

$$f'(x) \approx p_n'(x),$$

where $p_n(x)$ denotes the interpolating polynomial for $f(x)$. Many formulas can be obtained by varying n and by varying the placement of the nodes $x_0, \cdots, x_n$ relative to the point x of interest.

Let us take $n = 1$. The linear interpolating polynomial is given by

$$p_1(x) = f(x_0) + f[x_0, x_1](x - x_0).$$

Hence, we have the formula

$$f'(x) \approx p_1'(x) = f[x_0, x_1]. \tag{6.26}$$

In particular,

- if we take $x_0 = x$ and $x_1 = x + h$ for a small value $h > 0$, we obtain the forward difference formula $D_h^+ f(x)$.
- if we take $x_0 = x - h$ and $x_1 = x$ for small value of $h > 0$, we obtain the backward difference formula $D_h^- f(x)$.
- if we take $x_0 = x - h$ and $x_1 = x + h$ for small value of $h > 0$, we get the central difference formula $D_h^0 f(x)$.

We next prove the formula for mathematical error in approximating the first derivative using interpolating polynomial.

Theorem 6.16 (Mathematical Error).
Hypothesis:

(1) *Let f be an $(n+2)$-times continuously differentiable function on the interval $[a,b]$.*

(2) *Let $x_0, x_1, \cdots, x_n$ be $n+1$ distinct nodes in $[a,b]$.*

(3) *Let $p_n(x)$ denote the polynomial that interpolates f at the nodes $x_0, x_1, \cdots, x_n$.*

(4) *Let x be any point in $[a,b]$ such that $x \notin \{x_0,\ x_1\ \cdots,\ x_n\}$.*

Conclusion: *Then*

$$f'(x) - p_n'(x) = w_n(x)\frac{f^{(n+2)}(\eta_x)}{(n+2)!} + w_n'(x)\frac{f^{(n+1)}(\xi_x)}{(n+1)!} \tag{6.27}$$

with $w_n(x) = \prod_{i=0}^{n}(x - x_i)$ and ξ_x and η_x are points in between the maximum and minimum of $x_0,\ x_1\ \cdots,\ x_n$ and x, that depend on x.

Proof. For any $x \in [a,b]$ with $x \notin \{x_0,\ x_1\ \cdots,\ x_n\}$, by Newton's form of interpolating polynomial, we have

$$f(x) = p_n(x) + f[x_0, \cdots, x_n, x]w_n(x),$$

where $p_n(x)$ is the polynomial (of degree $\leq n$) that interpolates f at $x_0, \cdots, x_n$. Taking derivative on both sides, we get

$$f'(x) = p_n'(x) + w_n(x)\frac{d}{dx}f[x_0, \cdots, x_n, x] + w_n'(x)f[x_0, \cdots, x_n, x].$$

But we know that (Theorem 5.25)

$$\frac{d}{dx}f[x_0, \cdots, x_n, x] = f[x_0, \cdots, x_n, x, x].$$

Therefore, we have

$$f'(x) = p_n'(x) + w_n(x)f[x_0, \cdots, x_n, x, x] + w_n'(x)f[x_0, \cdots, x_n, x].$$

Further, from Theorem 5.27, we see that there exists an $\xi_x \in (a,b)$ such that

$$f[x_0, \cdots, x_n, x] = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}.$$

Using Theorem 5.27 along with the Hermite-Genocchi formula, we see that there exists an $\eta_x \in (a,b)$ such that

$$f[x_0, \cdots, x_n, x, x] = \frac{f^{(n+2)}(\eta_x)}{(n+2)!}.$$

Therefore, we get

$$f'(x) - p_n'(x) = w_n(x)\frac{f^{(n+2)}(\eta_x)}{(n+2)!} + w_n'(x)\frac{f^{(n+1)}(\xi_x)}{(n+1)!}$$

which is what we wish to show. □

Difference formulas for higher order derivatives and their mathematical error can be obtained similarly. The derivation of the mathematical error for the formulas of higher order derivatives are omitted for this course.

Example 6.17. Let x_0, x_1, and x_2 be the given nodes. Then, the Newton's form of interpolating polynomial for f is given by

$$p_2(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x^2 - (x_0 + x_1)x + x_0x_1).$$

Therefore, we take the first derivative of f as

$$f'(x) \approx p_2'(x) = f[x_0, x_1] + f[x_0, x_1, x_2](2x - x_0 - x_1).$$

- If we take $x_0 = x - h$, $x_1 = x$, and $x_2 = x + h$ for any given $x \in [a, b]$, we obtain the central difference formula $D_h^0(f)$ and the corresponding error obtained from (6.27) is precisely the error given in (6.25).
- If we take $x_0 = x$, $x_1 = x + h$ and $x_2 = x + 2h$ for any given $x \in [a, b]$, we obtain the difference formula

  $$f'(x) \approx \frac{-3f(x) + 4f(x + h) - f(x + 2h)}{2h}$$

  with mathematical error obtained using (6.27) as

  $$f'(x) - p_2'(x) = \frac{h^2}{3} f'''(\xi),$$

  for some $\xi \in (x, x + 2h)$. □

6.2.3 Methods based on Undetermined Coefficients

Another method to derive formulas for numerical differentiation is called the **method of undetermined coefficients**. The idea behind this method is similar to the one discussed in deriving quadrature formulas.

Suppose we seek a formula for $f^{(k)}(x)$ that involves the nodes x_0, x_1, $\cdots$, x_n. Then, write the formula in the form

$$f^{(k)}(x) \approx w_0 f(x_0) + w_1 f(x_1) + \cdots + w_n f(x_n) \tag{6.28}$$

where w_i, $i = 0, 1, \cdots, n$ are free variables that are obtained by imposing the condition that this formula is exact for polynomials of degree less than or equal to n.

Example 6.18. We will illustrate the method by deriving the formula for $f''(x)$ at nodes $x_0 = x - h$, $x_1 = x$ and $x_2 = x + h$ for a small value of $h > 0$.

For a small value of $h > 0$, let

$$f''(x) \approx D_h^{(2)} f(x) := w_0 f(x - h) + w_1 f(x) + w_2 f(x + h) \tag{6.29}$$

where w_0, w_1 and w_2 are to be obtained so that this formula is exact when $f(x)$ is a polynomial of degree less than or equal to 2. This condition is equivalent to the exactness for the three polynomials 1, x and x^2.

Step 1: When $f(x) = 1$ for all x. Then the formula of the form (6.29) is assumed to be exact and we get

$$w_0 + w_1 + w_2 = 0. \tag{6.30}$$

Step 2: When $f(x) = x$ for all x. Then the formula of the form (6.29) is assumed to be exact and we get

$$w_0(x-h) + w_1 x + w_2(x+h) = 0.$$

Using (6.30), we get

$$w_2 - w_0 = 0. \tag{6.31}$$

Step 3: When $f(x) = x^2$ for all x. Then the formula of the form (6.29) is assumed to be exact and we get

$$w_0(x-h)^2 + w_1 x^2 + w_2(x+h)^2 = 2.$$

Using (6.30) and (6.31), we get

$$w_0 + w_2 = \frac{2}{h^2}. \tag{6.32}$$

Solving the linear system of equations (6.30), (6.31), and (6.32), we get

$$w_0 = w_2 = \frac{1}{h^2}, \quad w_1 = -\frac{2}{h^2}.$$

Substituting these into (6.29), we get

$$D_h^{(2)} f(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}, \tag{6.33}$$

which is the required formula.

Let us now derive the mathematical error involved in this formula. For this, we use the Taylor's series

$$f(x \pm h) = f(x) \pm h f'(x) + \frac{h^2}{2!} f''(x) \pm \frac{h^3}{3!} f^{(3)}(x) + \cdots$$

in (6.33) to get

$$\begin{aligned} D_h^{(2)} f(x) = \frac{1}{h^2}\left(f(x) + h f'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f^{(3)}(x) + \frac{h^4}{4!} f^{(4)}(x) + \cdots \right) - \frac{2}{h^2} f(x) \\ + \frac{1}{h^2}\left(f(x) - h f'(x) + \frac{h^2}{2!} f''(x) - \frac{h^3}{3!} f^{(3)}(x) + \frac{h^4}{4!} f^{(4)}(x) - \cdots \right). \end{aligned}$$

After simplification, we get

$$D_h^{(2)} f(x) = f''(x) + \frac{h^2}{24}[(f^{(4)}(x) + \cdots) + (f^{(4)}(x) - \cdots)].$$

Now treating the fourth order terms on the right hand side as remainders in Taylor's series, we get

$$D_h^{(2)} f(x) = f''(x) + \frac{h^2}{24}[f^{(4)}(\xi_1) + f^{(4)}(\xi_2)],$$

for some $\xi_1, \xi_2 \in (x-h, x+h)$. Using intermediate value theorem for the function $f^{(4)}$, we get the mathematical error as

$$f''(x) - D_h^{(2)} f(x) = -\frac{h^2}{12} f^{(4)}(\xi) \tag{6.34}$$

for some $\xi \in (x-h, x+h)$, which is the required mathematical error. □

6.2.4 Arithmetic Error in Numerical Differentiation

Difference formulas are useful when deriving methods for solving differential equations. But they can lead to serious errors when applied to function values that are subjected to floating-point approximations. Let

$$f(x_i) = f_i + \epsilon_i, \quad i = 0, 1, 2.$$

To illustrate the effect of such errors, we choose the approximation $D_h^{(2)} f(x)$ given by (6.33) for the second derivative of f with $x_0 = x - h$, $x_1 = x$ and $x_2 = x + h$. Instead of using the exact values $f(x_i)$, we use the appoximate values f_i in the difference formula (6.33). That is,

$$\bar{D}_h^{(2)} f(x_1) = \frac{f_2 - 2f_1 + f_0}{h^2}.$$

The total error committed is

$$\begin{aligned} f''(x_1) - \bar{D}_h^{(2)} f(x_1) &= f''(x_1) - \frac{f(x_2) - 2f(x_1) + f(x_0)}{h^2} + \frac{\epsilon_2 - 2\epsilon_1 + \epsilon_0}{h^2} \\ &= -\frac{h^2}{12} f^{(4)}(\xi) + \frac{\epsilon_2 - 2\epsilon_1 + \epsilon_0}{h^2}. \end{aligned}$$

Using the notation $\epsilon_\infty := \max\{|\epsilon_0|, |\epsilon_1|, |\epsilon_3|\}$, we have

$$|f''(x_1) - \bar{D}_h^{(2)} f(x_1)| \le \frac{h^2}{12} |f^{(4)}(\xi)| + \frac{4\epsilon_\infty}{h^2}. \tag{6.35}$$

The error bound in (6.35) clearly shows that, although the first term (bound of mathematical error) tends to zero as $h \to 0$, the second term (bound of arithmetic error) can tend to infinity as $h \to 0$. This gives a possibility for the total error to be as large as possible when $h \to 0$. In fact, there is an optimal value of h to minimize the right side of (6.35) (as shown in Figure 6.5), which we will illustrate in the following example.

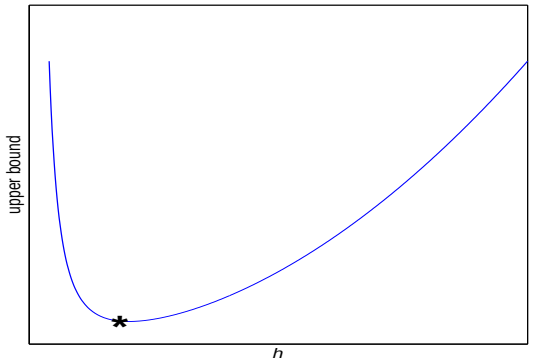

Fig. 6.5. A sketch of the upper bound in total error as given in (6.35) as a function of h. The black star indicates the optimal value of h.

Example 6.19. In finding $f''(\pi/6)$ for the function $f(x) = \cos x$, if we use the function values f_i that has six significant digits when compared to $f(x_i)$, then

$$\frac{|f(x_i) - f_i|}{|f(x_i)|} \leq 0.5 \times 10^{-5}.$$

Since $|f(x_i)| = |\cos(x_i)| \leq 1$, we have $|f(x_i) - f_i| \leq 0.5 \times 10^{-5}$.

We now use the formula $\bar{D}_h^{(2)} f(x)$ to approximate $f''(x)$. Assume that other than the approximation in the function values, the formula $\bar{D}_h^{(2)} f(x)$ is calculated exactly. Then the bound for the absolute value of the total error given by (6.35) takes the form

$$|f''(\pi/6) - \bar{D}_h^{(2)} f(\pi/6)| \leq \frac{h^2}{12}|f^{(4)}(\xi)| + \frac{4\epsilon_\infty}{h^2},$$

where $\epsilon_\infty \leq 0.5 \times 10^{-5}$ and $\xi \approx \pi/6$. Thus, we have

$$|f''(\pi/6) - \bar{D}_h^{(2)} f(\pi/6)| \leq \frac{h^2}{12}\cos\left(\frac{\pi}{6}\right) + \frac{4}{h^2}(0.5 \times 10^{-5}) \approx 0.0722h^2 + \frac{2 \times 10^{-5}}{h^2} =: E(h).$$

The bound $E(h)$ indicates that there is a smallest value of h, call it h^*, such that the bound increases rapidly for $0 < h < h^*$ when $h \to 0$. To find it, let $E'(h) = 0$, with its root being h^*. This leads to $h^* \approx 0.129$. Thus, for close values of $h > h^* \approx 0.129$, we have less error bound than the values $0 < h < h^*$. This behavior of $E(h)$ is observed in the following table. Note that the true value is $f''(\pi/6) = -\cos(\pi/6) \approx -0.86603$.

h	$\bar{D}_h^{(2)} f(\pi/6)$	Total Error	$E(h)$
0.2	-0.86313	-0.0029	0.0034
0.129	-0.86479	-0.0012	0.0024
0.005	-0.80000	-0.0660	0.8000
0.001	0.00000	-0.8660	20

When h is very small, $f(x-h)$, $f(x)$ and $f(x+h)$ are very close numbers and therefore their difference in the numerator of the formula (6.33) tend to have loss of significance. This is clearly observed in the values of $\bar{D}_h^{(2)} f(\pi/6)$ when compared to the true value where, we are not loosing much number of significant digits for $h > 0.129$, whereas for $h < 0.129$, there is a drastic loss of significant digits. □

CHAPTER 7

Numerical Ordinary Differential Equations

Consider a first order ordinary differential equation (ODE) of the form

$$y' = f(x, y),$$

where $y = y(x)$ is an unknown variable, $x \in \mathbb{R}$ is an independent variable and the function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is given. Here, we used the notation $y' := dy/dx$. The objective is to find the solution y for all $x \in [a, b]$ subject to an **initial condition**

$$y(x_0) = y_0, \quad x_0 \in [a, b].$$

We call the problem of solving the above ODE along with this initial condition, the **initial value problem**.

It is well-known that there are many ODEs of physical interest that cannot be solved exactly although we know that such problems have unique solutions. If one wants the solution of such problems, then the only way is to obtain it approximately. One common way of obtaining an approximate solution to a given initial value problem is to numerically compute the solution using a numerical method (or numerical scheme). In this chapter, we introduce some basic numerical methods for approximating the solution of a given initial value problem.

In Section 7.1, we review the exact solvability of a given initial value problem and motivate the need of numerical methods. In Section 7.3, we introduce a basic numerical method called **Euler's method** for obtaining approximate solution of an initial value problem and discussed the error involved in this method. We also show in this section that the Euler method is of order 1. Modified forms of Euler method can be obtained using numerical quadrature formulas as discussed in Section 7.4. Taylor approximation with higher order terms can be used to obtain numerical methods of higher order accuracy. However, such methods involve the higher order derivatives of the unknown function and are called **Taylor methods**. Taylor methods are hard to implement on a computer as they involve higher order derivatives. An alternate way of achieving higher order accuracy without using higher order derivatives of the unknown function is the famous **Runge-Kutta methods**. Runge-Kutta method of order 2 is derived in full detail in section 7.5 and the formula for the Runge-Kutta method of order 4 is stated.

7.1 Review of Theory

In this section, we will discuss solving an initial value problem (IVP) for an ordinary differential equation. Let f be a continuous function of two variables defined on a domain $D \subset \mathbb{R}^2$. Let $(x_0, y_0) \in D$ be given. The initial value problem for an ODE is given by

$$y' = f(x, y), \quad (x, y) \in D, \tag{7.1a}$$

$$y(x_0) = y_0. \tag{7.1b}$$

The following result is concerning the equivalence of the initial value problem (7.1) with an integral equation.

Lemma 7.1. *A continuous function y defined on an interval I containing the point x_0 is a solution of the initial value problem* (7.1) *if and only if y satisfies the integral equation*

$$y(x) = y_0 + \int_{x_0}^{x} f(s, y(s))\, ds, \tag{7.2}$$

for all $x \in I$.

Proof: If y is a solution of the initial value problem (7.1), then we have

$$y'(x) = f(x, y(x)). \tag{7.3}$$

Integrating the above equation from x_0 to x yields the integral equation (7.2).

On the other hand let y be a solution of integral equation (7.2). Observe that, due to continuity of the function $x \to y(x)$, the function $s \to f(s, y(s))$ is continuous on I. Thus by fundamental theorem of integral calculus, the right hand side of (7.2) is a differentiable function with respect to x and its derivative is given by the function $x \to f(x, y(x))$, which is a continuous function. From (7.2), we see that y is continuously differentiable and hence a direct differentiation of this equation yields the ODE (7.1a). Evaluating (7.2) at $x = x_0$ gives the initial condition $y(x_0) = y_0$. □

Remark 7.2. The following are the consequences of Lemma 7.1.

- If f is a function of x only, ie., $f = f(x)$, which can be integrated explicitly, then the integration on the right hand side of (7.2) can be obtained explicitely to get the solution y of the IVP (7.1a) exactly.

  For instance, if $f(x, y) = e^x$, then the solution of (7.1a) is $y(x) = y_0 + e^x - e^{x_0}$.
- If f depends on y, ie., $f = f(x, y)$, then (7.2) is an integral equation which in general cannot be solved explicitly. However, there are some particular cases, for instance when (7.1a) is linear, separable or exact, then we can obtain a solution of (7.1a). There are certain other cases where an integral factor can be obtained through which the equation can be made exact. But, there are many ODEs for which none of the above mentioned methods can be used to obtain solution.

For instance, the equation

$$y' = x^2 + y^2$$

is clearly neither linear nor separable nor exact. Also any of the standard ways of finding integrating factor will not work for this equation.

When standard method cannot be used to obtain solutions exactly, several approximation procedures may be proposed for finding an approximate solution to the initial value problem (7.1). In these procedures, one may use the ODE (7.1a) itself or its equivalent integral formulation (7.2). If we choose the ODE (7.1a), then the derivative may be approximated by a numerical differentiation formula. In case we choose the integral formulation (7.2), then we may use various numerical integration techniques (quadrature rules) to adopt a numerical method for approximating the solution of the initial value problem (7.1). □

Before going for a numerical method, it is important to ensure that the given initial value problem has a unique solution. Otherwise, we will be solving an initial value problem numerically, which actually does not have a solution or it may have many solutions and we do not know which solution the numerical method obtains. Let us illustrate the non-uniqueness of solution of an initial value problem.

Example 7.3 (Peano). Let us consider the initial value problem

$$y' = 3\,y^{2/3}, \quad y(0) = 0. \tag{7.4}$$

Note that $y(x) = 0$ for all $x \in \mathbb{R}$ is clearly a solution for this initial value problem. Also, $y(x) = x^3$ for all $x \in \mathbb{R}$. This initial value problem has infinite family of solutions parametrized by $c \geq 0$, given by

$$y_c(x) = \begin{cases} 0 & \text{if } x \leq c, \\ (x-c)^3 & \text{if } x > c, \end{cases}$$

defined for all $x \in \mathbb{R}$. Thus, we see that a solution to an initial value problem need not be unique. □

However by placing additional conditions on f, we can achieve uniqueness as stated in the following theorem. The proof of this theorem is omitted for this course.

Theorem 7.4 (Cauchy-Lipschitz-Picard's Existence and Uniqueness Theorem). *Let $D \subseteq \mathbb{R}^2$ be a domain and $I \subset \mathbb{R}$ be an interval. Let $f : D \to \mathbb{R}$ be a continuous function. Let $(x_0, y_0) \in D$ be a point such that the rectangle R defined by*

$$R = \{x \,:\, |x - x_0| \leq a\} \times \{y \,:\, |y - y_0| \leq b\} \tag{7.5}$$

is contained in D. Let f be Lipschitz continuous with respect to the variable y on R, i.e., there exists a $K > 0$ such that

$$|f(x, y_1) - f(x, y_1)| \leq K|y_1 - y_2| \quad \forall (x, y_1),\, (x, y_2) \in R.$$

Then the initial value problem (7.1) has at least one solution on the interval $|x - x_0| \leq \delta$ where $\delta = \min\{a, \frac{b}{M}\}$. Moreover, the initial value problem (7.1) has exactly one solution on this interval.

Remark 7.5. We state without proof that the function $f(x, y) = y^{2/3}$ in Example 7.3 is not a Lipschitz function on any rectangle containing the point $(0, 0)$ in it.

While verifying that a given function f is Lipschitz continuous is a little difficult, one can easily give a set of alternative conditions that guarantee Lipschitz continuity, and also these conditions are easy to verify.

Lemma 7.6. *Let $D \subseteq \mathbb{R}^2$ be an open set and $f : D \to \mathbb{R}$ be a continuous function such that its partial derivative with respect to the variable y is also continuous on D, i.e., $\frac{\partial f}{\partial y} : D \to \mathbb{R}$ is a continuous function. Let the rectangle R defined by*

$$R = \{x : |x - x_0| \leq a\} \times \{y : |y - y_0| \leq b\}$$

be contained in D. Then f is Lipschitz continuous with respect to the variable y on R.

Proof: Let (x, y_1), $(x, y_2) \in R$. Applying mean value theorem with respect to the y variable, we get

$$f(x, y_1) - f(x, y_1) = \frac{\partial f}{\partial y}(x, \xi)(y_1 - y_2), \tag{7.6}$$

for some ξ between y_1 and y_2. Since we are applying mean value theorem by fixing x, this ξ will also depend on x. However since $\frac{\partial f}{\partial y} : D \to \mathbb{R}$ is a continuous function, it will be bounded on R. That is, there exists a number $L > 0$ such that

$$\left|\frac{\partial f}{\partial y}(x, y)\right| \leq L \quad \text{for all } (x, y) \in R. \tag{7.7}$$

Taking modulus in the equation (7.6), and using the last inequality we get

$$|f(x, y_1) - f(x, y_1)| = \left|\frac{\partial f}{\partial y}(x, \xi)\right| \, |(y_1 - y_2)| \leq L \, |(y_1 - y_2)|.$$

This finishes the proof of lemma. □

The following theorem can be used as a tool to check the existence and uniqueness of solution of a given initial value problem (7.1).

Corollary 7.7 (Existence and Uniqueness Theorem).

Let $D \subseteq \mathbb{R}^2$ be a domain and $I \subseteq \mathbb{R}$ be an interval. Let $f : D \to \mathbb{R}$ be a continuous function. Let $(x_0, y_0) \in D$ be a point such that the rectangle R defined by

$$R = \{x : |x - x_0| \leq a\} \times \{y : |y - y_0| \leq b\} \tag{7.8}$$

is contained in D.

If the partial derivative $\partial f/\partial y$ is also continuous in D, then there exists a unique solution $y = y(x)$ of the initial value problem (7.1) defined on the interval $|x - x_0| \leq \delta$ where $\delta = \min\{a, \frac{b}{M}\}$.

We state an example (without details) of an initial value problem that has a unique solution but the right hand side function in the differential equation is not a Lipschitz function. Thus this example illustrates the fact that condition of Lipschitz continuity is only a sufficient condition for uniqueness and by no means necessary.

Example 7.8. The IVP

$$y' = \begin{cases} y \sin \frac{1}{y} & \text{if} \quad y \neq 0 \\ 0 & \text{if} \quad y = 0, \end{cases} \qquad y(0) = 0.$$

has unique solution, despite the RHS not being Lipschitz continuous with respect to the variable y on any rectangle containing $(0, 0)$. □

Remark 7.9 (On Existence and Uniqueness Theorem).

(1) Though we have stated existence theorems without proof, we can confirm that their proofs do not give an explicit soluton of the initial value problem being considered. In fact, the proofs give a sequence of functions that converge to a solution of the initial value problem.

(2) Even when the RHS of the ordinary differential equation is a function of x only, we do not have explicit solutions. For example,

$$y' = e^{-x^2}, \; y(0) = 0.$$

The only alternative is to approximate its solution by a numerical procedure. □

7.2 Discretization Notations

A numerical method gives approximate value of the solution of the initial value problem (7.1) at only a discrete set of point. Thus, if we are interested in obtaining solution for (7.1a) in an interval $[a, b]$, then we first discretize the interval as

$$a = x_0 < x_1 < \cdots < x_n = b, \tag{7.9}$$

where each point x_i, $i = 0, 1, \cdots, n$ is called a **node**. Unless otherwise stated, we always assume that the nodes are equally spaced. That is,

$$x_j = x_0 + jh, \quad j = 0, 1, \cdots n \tag{7.10}$$

for a sufficiently small positive real number h. We use the notation for the approximate solution as

$$y_j = y_h(x_j) \approx y(x_j), \quad j = 0, 1, \cdots, n. \tag{7.11}$$

7.3 Euler's Method

The most simplest numerical method for a first order ordinary differential equation (7.1a) is obtained by replace the first order derivative of the unknown function by its finite difference formula. Assume that we know the value of the unknown function y at a point $x = x_0$. For obtaining the value of y at the point $x_1 = x_0 + h$, we use the forward difference formula for the derivative given by

$$y'(x) \approx \frac{1}{h}(y(x+h) - y(x))$$

in (7.1a) to get

$$\frac{1}{h}(y(x_1) - y(x_0)) \approx f(x_0, y(x_0)).$$

This can be used to obtain the value of $y(x_1)$ in terms of x_0 and $y(x_0)$ as

$$y(x_1) \approx y(x_0) + hf(x_0, y(x_0)).$$

Since we assumed that we know the value of $y(x_0)$, the right hand side is fully known and hence $y(x_1)$ can now be computed explicitly.

In general, if you know the value of $y(x_j)$, $j = 0, 1, \cdots, n$, we can obtain the value of $y(x_{j+1})$ by using the forward difference formula in (7.1a) at $x = x_j$ to get

$$\frac{1}{h}(y(x_{j+1}) - y(x_j)) \approx f(x_j, y(x_j)).$$

Denoting the approximate value of $y(x_j)$ by y_j, we can adopt the formula

$$y_{j+1} = y_j + hf(x_j, y_j), \quad j = 0, 1, \cdots, n-1 \tag{7.12}$$

for obtaining the values of the solution y at the discrete points x_j, $j = 1, 2, \cdots, n$ by taking the value of y_0 from the initial condition (7.1b). The formula (7.12) is called the **forward Euler's method**.

The **backward Euler's method** can be obtained by using the backward difference formula for the derivative of y in (7.1) and is given by

$$y_{j-1} = y_j - hf(x_j, y_j), \quad j = 0, -1, \cdots, -n+1. \tag{7.13}$$

By **Euler's method** we mean either forward or backward Euler's method depending on the context.

Geometrical Interpretation:
A geometric insight into Euler's method is shown in Figure 7.1 The tangent line to the graph of $y(x)$ at $x = x_j$ has slope $f(x_j, y_j)$. The Euler's method approximates the value of $y(x_{j+1})$ by the corresponding value of tangent line at the point $x = x_{j+1}$. □

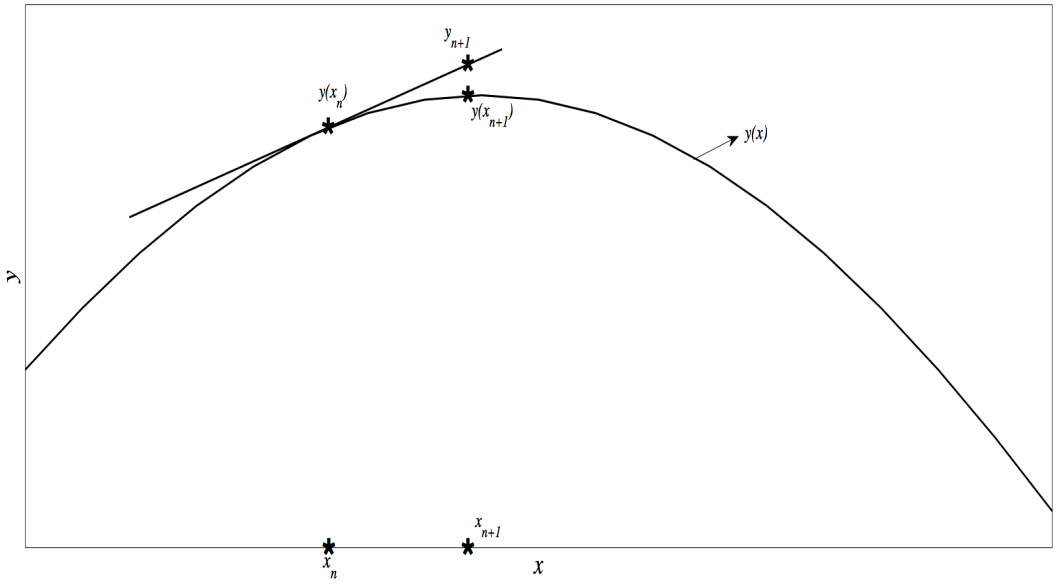

Fig. 7.1. Geometrical interpretation of Euler's method

Example 7.10. Consider the initial value problem

$$y' = y, \quad y(0) = 1.$$

The Euler method (7.12) for this equation takes the form

$$y_{j+1} = y_j + hy_j = (1 + h)y_j.$$

Note that the exact solution for the given initial value problem is $y(x) = e^x$.

On applying Euler's method with $h = 0.01$ and using 7-digit rounding, we get

$$\begin{aligned}
y(0.01) &\approx y_1 = 1 + 0.01 = 1.01 \\
y(0.02) &\approx y_2 = 1.01 + 0.01(1.01) = 1.0201 \\
y(0.03) &\approx y_3 = 1.0201 + 0.01(1.0201) = 1.030301 \\
y(0.04) &\approx y_4 = 1.030301 + 0.01(1.030301) = 1.040604
\end{aligned}$$

The numerical results along with the error is presented in the following table for $h = 0.01$.

h	x	$y_h(x)$	Exact Solution	Error	Relative Error
0.01	0.00	1.000000	1.000000	0.000000	0.000000
0.01	0.01	1.010000	1.010050	0.000050	0.000050
0.01	0.02	1.020100	1.020201	0.000101	0.000099
0.01	0.03	1.030301	1.030455	0.000154	0.000149
0.01	0.04	1.040604	1.040811	0.000207	0.000199
0.01	0.05	1.051010	1.051271	0.000261	0.000248

Since the exact solution of this equation is $y = e^x$, the correct value at $x = 0.04$ is 1.040811.

By taking smaller values of h, we may improve the accuracy in the Euler's method. The numerical results along with the error is shown in the following table for $h = 0.005$.

h	x	$y_h(x)$	Exact Solution	Error	Relative Error
0.005	0.00	1.000000	1.000000	0.000000	0.000000
0.005	0.00	1.005000	1.005013	0.000013	0.000012
0.005	0.01	1.010025	1.010050	0.000025	0.000025
0.005	0.01	1.015075	1.015113	0.000038	0.000037
0.005	0.02	1.020151	1.020201	0.000051	0.000050
0.005	0.02	1.025251	1.025315	0.000064	0.000062
0.005	0.03	1.030378	1.030455	0.000077	0.000075
0.005	0.03	1.035529	1.035620	0.000090	0.000087
0.005	0.04	1.040707	1.040811	0.000104	0.000100
0.005	0.04	1.045910	1.046028	0.000117	0.000112
0.005	0.05	1.051140	1.051271	0.000131	0.000125

7.3.1 Error in Euler's Method

In Example (7.10), we illustrated that as we reduce the step size h, we tend to get more accurate solution of a given IVP at a given point $x = x_j$. The truncation error confirms this illustration when y'' is a bounded function. However, the mathematical error which involves the truncation error in the computed solution y_j and the propagating error from the computation of the solution at $x = x_i$ for $i = 0, 1, \cdots j - 1$. In addition to the mathematical error, we also have arithmetic error due to floating-point approximation in each arithmetic operation. In this section, we study the total error involved in forward Euler's method. Total error involved in backward Euler's method can be obtained in a similar way.

Using Taylor's theorem, write

$$y(x_{j+1}) = y(x_j) + hy'(x_j) + \frac{h^2}{2}y''(\xi_j)$$

for some $x_j < \xi_j < x_{j+1}$. Since $y(x)$ satisfies the ODE $y' = f(x, y(x))$, we get

$$y(x_{j+1}) = y(x_j) + hf(x_j, y(x_j)) + \frac{h^2}{2}y''(\xi_j).$$

Thus, the local **truncation error** in forward Euler's method is

$$T_{j+1} = \frac{h^2}{2}y''(\xi_j), \tag{7.14}$$

which is the error involved in obtaining the value $y(x_{j+1})$ using the exact value $y(x_j)$. The forward Euler's method uses the approximate value y_j in the formula and therefore the

finally computed value y_{j+1} not only involves the truncation error but also the propagated error involved in computing y_j. Thus, the local **mathematical error** in the forward Euler's method is given by

$$\mathrm{ME}(y_{j+1}) := y(x_{j+1}) - y_{j+1} = y(x_j) - y_j + h(f(x_j, y(x_j)) - f(x_j, y_j)) + \frac{h^2}{2} y''(\xi_j).$$

Here, $y(x_j) - y_j + h(f(x_j, y(x_j)) - f(x_j, y_j))$ is the propagated error.

The propagated error can be simplified by applying the mean value theorem to $f(x, z)$ considering it as a functin of z:

$$f(x_j, y(x_j)) - f(x_j, y_j) = \frac{\partial f(x_j, \eta_j)}{\partial z}[y(x_j) - y_j],$$

for some η_j lying between $y(x_j)$ and y_j. Using this, we get the mathematical error

$$\mathrm{ME}(y_{j+1}) = \left[1 + h\frac{\partial f(x_j, \eta_j)}{\partial z}\right] \mathrm{ME}(y_j) + \frac{h^2}{2} y''(\xi_j) \tag{7.15}$$

for some $x_j < \xi_j < x_{j+1}$, and η_j lying between $y(x_j)$ and y_j.

We now assume that over the interval of interest,

$$\left|\frac{\partial f(x_j, y(x_j))}{\partial z}\right| < L, \quad |y''(x)| < Y,$$

where L and Y are fixed positive constants. On taking absolute values in (7.15), we obtain

$$|\mathrm{ME}(y_{j+1})| \le (1 + hL)|\mathrm{ME}(y_j)| + \frac{h^2}{2} Y. \tag{7.16}$$

Applying the above estimate recursively, we get

$$\begin{aligned}
|\mathrm{ME}(y_{j+1})| &\le (1 + hL)^2 |\mathrm{ME}(y_{j-1})| + (1 + (1 + hL))\frac{h^2}{2} Y \\
&\le \cdots \\
&\le \cdots \\
&\le (1 + hL)^{j+1} |\mathrm{ME}(y_0)| + \Big(1 + (1 + hL) + (1 + hL)^2 + \cdots + (1 + hL)^j\Big)\frac{h^2}{2} Y.
\end{aligned}$$

Using the formulas

(1) For any $\alpha \neq 1$,

$$1 + \alpha + \alpha^2 + \cdots + \alpha^j = \frac{\alpha^{j+1} - 1}{\alpha - 1}$$

(2) For any $x \ge -1$,

$$(1 + x)^N \le e^{Nx},$$

in the above inequality, we have proved the following theorem.

Theorem 7.11. *Let $y \in C^2[a,b]$ be a solution of the IVP (7.1) with*

$$\left|\frac{\partial f(x,y)}{\partial y}\right| < L, \quad |y''(x)| < Y,$$

for all x and y, and some constants $L > 0$ and $Y > 0$. The mathematical error in the forward Euler's method at a point $x_j = x_0 + jh$ satisfies

$$|\mathrm{ME}(y_j)| \leq \frac{hY}{2L}\left(e^{(x_n - x_0)L} - 1\right) + e^{(x_n - x_0)L}|y(x_0) - y_0| \tag{7.17}$$

Example 7.12. Consider the initial value problem

$$y' = y, \ y(0) = 1, \quad x \in [0,1].$$

Let us now find the upper bound for the mathematical error of forward Euler's method in solving this problem.

Here $f(x,y) = y$. Therefore, $\partial f/\partial y = 1$ and hence we can take $L = 1$.

Since $y = e^x$, $y'' = e^x$ and $|y''(x)| \leq e$ for $0 \leq x \leq 1$. Therefore, we take $Y = e$.

We now use the estimate (7.17) with $x_0 = 0$ and $x_n = 1$ to obtain

$$|\mathrm{ME}(y_j)| \leq \frac{he}{2}(e-1) \approx 2.3354h.$$

Here, we assume that there is no approximation in the initial condition and therefore the second term in (7.17) is zero.

To validate the upper bound obtained above, we shall compute the approximate solution of the given IVP using forward Euler's method. The method for the given IVP reads

$$y_{j+1} = y_j + hf(x_j, y_j) = (1+h)y_j.$$

The solution of this difference equation satisfing $y(0) = 1$ is

$$y_j = (1+h)^j.$$

Now, if $h = 0.1$, $n = 10$, we have $y_j = (1.1)^{10}$. Therefore, the forward Euler's method gives $y(1) \approx y_{10} \approx 2.5937$. But the exact value is $y(1) = e \approx 2.71828$. The error is 0.12466, whereas the bound obtained from (7.17) was 0.2354. □

Remark 7.13. The error bound (7.17) is valid for a large family of the initial value problems. But, it usually produces a very poor estimate due to the presence of the exponential terms. For instance, in the above example, if we take x_n to be very large, then the corresponding bound will also be very large. □

The above error analysis assumes that the numbers used are of infinite precision and no floating point approximation is assumed. When we include the floating point approximation that $y_n = \tilde{y}_n + \epsilon_n$, then the bound for total error is given in the following theorem. The proof of this theorem is left as an exercise.

Theorem 7.14. *Let $y \in C^2[a,b]$ be a solution of the IVP (7.1) with*

$$\left|\frac{\partial f(x,y)}{\partial y}\right| < L, \quad |y''(x)| < Y,$$

for all x and y, and some constants $L > 0$ and $Y > 0$. Let y_j be the approximate solution of (7.1) computed using the forward Euler's method (7.12) with infinite precision and let $\tilde{y}_j$ be the corresponding computed value using finite digit floating-point arithmetic. If

$$y_j = \tilde{y}_j + \epsilon_j,$$

then the total error $\mathrm{TE}(y_j) := y(x_j) - \tilde{y}_j$ in forward Euler's method at a point $x_j = x_0 + jh$ satisfies

$$|\mathrm{TE}(y_j)| \leq \frac{1}{L}\left(\frac{hY}{2} + \frac{\epsilon}{h}\right)\left(e^{(x_n - x_0)L} - 1\right) + e^{(x_n - x_0)L}|\epsilon_0|, \tag{7.18}$$

where $\epsilon := \max\{|\epsilon_i|/i = 0, 1, \cdots, n\}$. □

7.4 Modified Euler's Methods

The Euler's method derived in the previous section can also be derived using the equivalent integral form of the IVP (7.1) as discussed in Lemma 7.1. Using this integral form in the interval $[x_j, x_{j+1}]$, we get

$$y(x_{j+1}) = y(x_j) + \int_{x_j}^{x_{j+1}} f(s,y)ds. \tag{7.19}$$

The Euler's method can be obtained by replacing the integral on the right hand side by the rectangle rule.

Using the integral form in the interval $[x_{j-1}, x_{j+1}]$ and using the **mid-point** quadrature formula given by

$$\int_{x_{j-1}}^{x_{j+1}} f(s,y)ds \approx f(x_j, y_j)(x_{j+1} - x_{j-1}),$$

we get the **Euler's mid-point method**

$$y_{j+1} = y_{j-1} + 2hf(x_j, y_j). \tag{7.20}$$

To compute the value of y_{j+1}, we need to know the value of y_{j-1} and y_j. Note that the above formula cannot be used to compute the value of y_1. Hence, we need another method to obtain y_1 and then y_j, for $j = 2, 3, \cdots, n$ can be obtained using (7.20). This method belongs to the class of **2-step methods**.

Example 7.15. Consider the initial-value problem

$$y' = y, \quad y(0) = 1.$$

To obtain the approximate value of $y(0.4)$ with $h = 0.01$:

We first use Euler's method to get

$$y(0.01) \approx y_1 = 1 + 0.01 = 1.01$$

Next use Euler's mid-point method to get

$$\begin{aligned} y(0.02) &\approx y_2 = y_0 + 2 \times h \times y_1 = 1 + 2 \times 0.01 \times 1.01 = 1.0202 \\ y(0.03) &\approx y_3 = y_1 + 2 \times h \times y_2 = 1.01 + 2 \times 0.01 \times 1.0202 \approx 1.030404 \\ y(0.04) &\approx y_4 = y_2 + 2 \times h \times y_3 = 1.040808 \end{aligned}$$

Since the exact solution of this equation is $y = e^x$, the correct value at $x = 0.04$ is 1.040811. The error is 0.000003.

Recall the error in Euler method was 0.000199. □

The methods derived above are **explicit methods** in the sense that the value of y_{j+1} is computed using the known values. If we using the trapezoidal rule for the integration on the right hand side of (7.19), we get

$$y_{j+1} = y_j + \frac{h}{2}(f(x_j, y_j) + f(x_{j+1}, y_{j+1})). \tag{7.21}$$

This method is called the **Euler's Trapezoidal method**. Here, we see that the formula (7.21) involves an implicit relation for y_{j+1}. Such methods are referred to as **implicit methods**.

Although the Euler's Trapezoidal method gives an implicit relation for y_{j+1}, sometimes it is explicit to compute the values y_{j+1} as illustrated in the following example.

Example 7.16. Let us use the Euler's trapezoidal rule with $h = 0.2$ to obtain the appoximate solution of the initial value problem

$$y' = xy, \quad y(0) = 1.$$

We have $y_0 = 1$ and

$$y_1 = y_0 + \frac{h}{2}(x_0 y_0 + x_1 y_1) = 1 + 0.1(0 + 0.2y_1),$$

which gives $(1 - 0.02)y_1 = 1$, and this implies $y_1 \approx 1.0204$. Similarly,

$$y_2 = y_1 + \frac{h}{2}(x_1 y_1 + x_2 y_2) = 1.0204 + 0.1(0.2 \times 1.0204 + 0.4y_2),$$

and

$$y(0.4) \approx y_2 = \frac{1.0408}{1 - 0.04} \approx 1.0842.$$

□

In general, the Euler's trapezoidal rule gives a nonlinear equation for y_{j+1} as illustrated below.

Example 7.17. Consider the initial value problem

$$y' = e^{-y}, \quad y(0) = 1.$$

We use the Euler's trapezoidal rule with $h = 0.2$ to solve the above problem. We have,

$$y_1 = y_0 + \frac{h}{2}(e^{-y_0} + e^{-y_1}) = 1 + 0.1(e^{-1} + e^{-y_1}),$$

which gives the nonlinear equation

$$g(y_1) = y_1 - 0.1e^{-y_1} - (1 + 0.1e^{-1}) = 0,$$

and the solution of this equation is the approximate value of the solution $y(x_1)$ of the given initial value problem. □

7.5 Runge-Kutta Methods

Although Euler's method is easy to implement, this method is not so efficient in the sense that to get a better approximation, one needs a very small step size. One way to get a better accuracy is to include the higher order terms in the Taylor expansion to get an approximation to y'. But the higher order terms involve higher derivatives of y. The **Runge-Kutta methods** attempts to obtain higher order accuracy and at the same time avoid the need for higher derivatives, by evaluating the function $f(x, y)$ at selected points on each subintervals. We first derive the Runge-Kutta method of order 2. The derivation of the Runge-Kutta method of order 4 can be done in a similar way. So, we skip the derivation of this method and present only final formula.

7.5.1 Order Two

Let y be a solution of the ODE (7.1a). The Runge-Kutta method of order 2 is obtained by truncating the Taylor expansion of $y(x+h)$ after the quadratic term. We derive now a formula for this method. Taylor expansion of $y(x + h)$ at the point x upto the quadratic term is given by

$$y(x + h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + O(h^3). \tag{7.22}$$

Since y satisfies the given ODE $y' = f(x, y)$, by differentiating this ODE with respect to x both sides gives

$$y''(x) = \frac{\partial f}{\partial x}(x, y(x)) + y'(x)\frac{\partial f}{\partial y}(x, y(x)) \tag{7.23}$$

Substituting the values of y', y'' in (7.22), we get

$$y(x+h) = y(x) + hf(x,y(x)) + \frac{h^2}{2}\left[\frac{\partial f}{\partial x}(x,y(x)) + f(x,y(x))\frac{\partial f}{\partial y}(x,y(x))\right] + O(h^3).$$

The last equation is re-written as

$$\begin{aligned} y(x+h) = y(x) &+ \frac{h}{2}f(x,y(x)) \\ &+ \frac{h}{2}\left[f(x,y(x)) + h\frac{\partial f}{\partial x}(x,y(x)) + hf(x,y(x))\frac{\partial f}{\partial y}(x,y(x))\right] \\ &+ O(h^3) \end{aligned} \quad (7.24)$$

Taking $x = x_j$ for $j = 0, 1, \cdots, n-1$ with $x_{j+1} = x_j + h$ in (7.24), we get

$$\begin{aligned} y(x_{j+1}) = y(x_j) &+ \frac{h}{2}f(x_j,y(x_j)) \\ &+ \frac{h}{2}\left[f(x_j,y(x_j)) + h\frac{\partial f}{\partial x}(x_j,y(x_j)) + hf(x_j,y(x_j))\frac{\partial f}{\partial y}(x_j,y(x_j))\right] \\ &+ O(h^3). \end{aligned} \quad (7.25)$$

Let us now expand the function $f = f(s,t)$, which is a function of two variables, into its Taylor series at the point (ξ,τ) and truncate the series after the linear term. It is given by

$$f(s,t) = f(\xi,\tau) + (s-\xi)\frac{\partial f}{\partial s}(\xi,\tau) + (t-\tau)\frac{\partial f}{\partial t}(\xi,\tau) + O\big((s-\xi)^2\big) + O\big((t-\tau)^2\big).$$

Taking $(\xi,\tau) = (x_j, y(x_j))$ and comparing the above equation with the term in the square brackets in the equation (7.25), we get

$$y(x_{j+1}) = y(x_j) + \frac{h}{2}f(x_j,y(x_j)) + \frac{h}{2}\left[f(x_{j+1}, y(x_j) + hf(x_j,y(x_j)))\right] + O(h^3). \quad (7.26)$$

Truncating the higher order terms and denoting the approximate value of $y(x_{j+1})$ as y_{j+1}, we get

$$y_{j+1} = y_j + \frac{h}{2}f(x_j,y_j) + \frac{h}{2}\left[f(x_{j+1}, y_j + hf(x_j,y_j))\right]. \quad (7.27)$$

Although the terms dropped from (7.26) to get (7.27) are of order 3 (namely, $O(h^3)$), the resultant approximation to y' is of order 2 as is evident from the Taylor's formula (7.22). The equation (7.27) is therefore known as **Runge-Kutta method of order 2**. To facilitate easy memorizing, the formula (7.27) may be written as

$$y_{j+1} = y_j + \frac{1}{2}(k_1 + k_2),$$

where

$$\begin{aligned} k_1 &= h\, f(x_j, y_j) \\ k_2 &= h\, f(x_{j+1}, y_j + k_1). \end{aligned}$$

The truncation error of Runge-Kutta method of order 2 is of order $O(h^3)$ whereas the Euler's method is of order $O(h^2)$. Therefore, for a fixed $h > 0$ we expect to get more accurate result from Runge-Kutta method order 2 when compared to Euler's method.

Example 7.18. Consider the initial-value problem

$$y' = y, \quad y(0) = 1.$$

Using Runge-Kutta method of order 2, we obtain

x	y	k_1	k_2
0.000000	1.000000	0.010000	0.010100
0.010000	1.010050	0.010000	0.010100
0.020000	1.020201	0.010100	0.010202
0.030000	1.030454	0.010202	0.010304
0.040000	1.040810	0.010305	0.010408

Recall the exact solution is $y(x) = e^x$ and $y(0.04) \approx 1.040811$. Therefore, the error involved is 0.000001 which is much less than the error (0.000199) obtained in Euler's method for $h = 0.01$. □

7.5.2 Order Four

We state without derivation, the formula for the **Runge-Kutta method of order 4**.

$$y_{j+1} = y_j + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where

$$\begin{aligned}
k_1 &= hf(x_j, y_j), \\
k_2 &= hf\left(x_j + \frac{h}{2}, y_j + \frac{k_1}{2}\right), \\
k_3 &= hf\left(x_j + \frac{h}{2}, y_j + \frac{k_2}{2}\right), \\
k_4 &= hf(x_j + h, y_j + k_3)
\end{aligned}$$

The local truncation error of the 4^{th} order Runge-Kutta Method is of $O(h^5)$.

Example 7.19. Consider the initial-value problem

$$y' = y, \quad y(0) = 1.$$

Using Runge-Kutta method of order 4, we obtain

x_j	y_j	Exact Solution	Error	Relative Error
0.00	1.000000	1.000000	0.000000	0.000000
0.01	1.010050	1.010050	0.000000	0.000000
0.02	1.020201	1.020201	0.000000	0.000000
0.03	1.030455	1.030455	0.000000	0.000000
0.04	1.040811	1.040811	0.000000	0.000000

Note that the exact solution is $y(x) = e^x$. □

Index