Numerical Methods

CEGN-2073

Eyaya B

26th March 2019

Contents

Chapter 1

Basic Concepts in Error Estimation

1.1 Introduction

Many problems in Science and Engineering can not be solved analytically on a computer and as a result Numeric solutions are often required. Numeric solutions provide only approximate solutions and they are not unique. Different numerical algorithms might yield different approximations.

Definition 1.1 Numerical Analysis

Numerical Analysis deals with the design, analysis of numeric algorithms that deals with continuous or discrete quantities and considers or analyzes the effects of approximations.

Definition 1.2 An Error

Error-that is, how far an answer or computed value is from the true value.

The reliability of the numerical result will depend on an error estimate or bound, therefore the analysis of error and the sources of error in numerical methods is also a critically important part of the study of numerical technique.

In general, solving of computational problems usually involves the following steps:

- Mathematical modeling: In general, models:
  - are abstractions of reality!

  - are a representations of a particular thing, idea or condition.

Mathematical Models: are simplified representations of some real world entity that can be expressed in mathematical equations or computer code. A Mathematical Model can be broadly defined as a formulation or equation that express the essential features of a physical system or process in mathematical terms. In a very general sense, it can be represented as a functional relationship of the form

$$\texttt{Dvar} = f(\texttt{Indvars}, \texttt{Params}, \texttt{ffuncs}) \tag{1.1}$$

Where, `Dvar`(Dependent variable):is a characteristic that typically reflect the behavior or state of the system,
`Indvar`(Independent variables): are usually dimensions, such as time and space, along with the system's behavior is being determined,
`Params`(Parameters): are reflectives of the system's properties or compositions, and
`ffuncs`(Forcing functions): are external influences acting upon it.
The actual Mathematical expression of equation (1.1) can range from a simple algebraic relationship to large complicated sets of differential equation.

- Algorithm design: involves
  - Building an algorithm to solve the mathematical problem formulation and
  - Analyze the algorithm for its performance.
- Implementation and Evaluation: involves
  - programming implementation of the algorithm and
  - Evaluate its performance with real data

1.2 Errors and approximations in computations

When we use numerical methods or algorithms and computing with finite precision, errors of approximation or rounding and truncation are introduced. It is important to have a notion of their nature and their order. A newly developed method is worthless without an error analysis. Neither does it make sense to use methods which introduce errors with magnitudes larger than the estimated error bound. On the other hand, using a method with very high accuracy might be computationally too expensive to justify the gain in accuracy.

Before we going to discuss these source of errors in detail, let us fist define what an error is and measure of errors in a Numerical calculation.

1.2.1 Sources of Errors

Error in solving an engineering or science problem can arise due to several factors. A paramount goal in numerical analysis is to asses the accuracy of the results of calculations. Errors contained in the numerical answers to problems generally arise in two areas:

Type-I Those inherent in the mathematical formulation of the problems, such as

(a) The error incurred when the mathematical statement of a problem is only an approximation to the physical situation;

(b) The error due to inaccuracies in the physical data.

Type-II Those incurred in the numerical computation process, such as

(a) Programming blunder;

(b) Truncation error, i.e., inexact evaluation of mathematical operators;

(c) Roundoff errors, i.e., inexact arithmetic calculations.

(d) Propagation errors,i.e, errors can be amplified as they propagate through an iterative computation.

Type (1) errors are beyond the control of the calculation and are usually negligible. It is understood, however, that the worth of a computed solution must be carefully weighed against these errors. Programming blunder which results in the correct calculation of the wrong result usually can be detected or verified. It is the last three sources of computational error that chiefly interest us and should be controlled by any feasible Numerical method.

1.2.2 Measure of Errors

An error can be measured in three different ways:

Definition 1.3 Absolute Error

Absolute error, denoted by ϵ_{abs}, is the numerical difference between the computed (or estimated) value of a quantity and the true value. Usually, absolute error is defined in terms of the magnitude of the difference between the true value and the approximate value as:

$$\epsilon_{abs} = \|\boldsymbol{x} - \boldsymbol{x_a}\|, \tag{1.2}$$

where $\boldsymbol{x}$ denotes the exact value and $\boldsymbol{x_a}$ denotes the approximate value. The unit of exact or unit of approximate values expresses the absolute error.

The absolute error doesn't usually signify the measure of an error. For instance a 0.1 pound absolute error is very small error when measuring a person's weight, but the same error measure can be disastrous when measuring the dosage of a medicine. On the other hand, the relative error and percentage error defined bellow, gives a better measure of an error.

Definition 1.4 Relative Error

Relative error, ϵ_{rel}, is the absolute error divided by the true value:

$$\epsilon_{rel} = \frac{\|\boldsymbol{x} - \boldsymbol{x_a}\|}{\|\boldsymbol{x}\|} = \frac{\epsilon_{abs}}{\|\boldsymbol{x}\|}. \tag{1.3}$$

The relative error is dimensionless or it's independent of units.

Definition 1.5 Percentage Error

The percentage error,ϵ_p in $\boldsymbol{x_a}$, committed in approximating the true value of $\boldsymbol{x}$ is given by

$$\epsilon_p = \epsilon_{rel} \times 100\% = \frac{\|\boldsymbol{x} - \boldsymbol{x_a}\|}{\|\boldsymbol{x}\|} \times 100\% \tag{1.4}$$

Example 1.1

Consider $x = 0.33$ and $fl(x) = 0.30$. Clearly,

$$\begin{aligned} \epsilon_{abs} &= 0.03 \text{ and} \\ \epsilon_{rel} &= \frac{0.03}{0.33} \\ &= 0.09091 \approx 9.1\%. \end{aligned}$$

When $x = 0.33 \times 10^{-5}$ and $fl(x) = 0.30 \times 10^{-5}$, we have

$$\begin{aligned} \epsilon_{abs} &= 0.03 \times 10^{-5} \\ &= 3 \times 10^{-7} \end{aligned}$$

but

$$\begin{aligned} \epsilon_{rel} &= 0.09091. \\ &\approx 9.1\%. \end{aligned}$$

Note that the relative error is unchanged, while the absolute error changed by a factor of 10^5.

Therefore, from the above observation we can concluded the following:

- The absolute error is strongly dependent on the magnitude of x.
- The absolute error is misleading unless it is stated what it is an error of.
- The relative error is a measure of the number of significant digits of x that are correct, we will discuss this in detail in Section (1.3.2).
- A relative error has meaning even when x is not known. It is given as a percentage value.

Example 1.2

Three approximate values of the number $\frac{1}{3}$ are given as 0.3, 0.33, and 0.34. Which of these 3 approximations is the best approximation?

Solution: The number which has least absolute error gives the best approximation:

- $E_a^1 = |\frac{1}{3} - 0.30| = \frac{1}{30}$
- $E_a^1 = |\frac{1}{3} - 0.33| = \frac{1}{300}$

- $E_a^1 = |\frac{1}{3} - 0.34| = \frac{1}{150}$

Therefore, since $\frac{1}{300}$ is the smallest of all the absolute errors, 0.33 is the best approximation for $\frac{1}{3}$.

Approximate Numbers

- **Exact Number:** a number with, which no uncertainly is associated, no approximation is taken. Example: $5, 1/2, \frac{21}{6}$.
- **Approximate Number:** There are numbers which are not exact. For example, $e = 2.7182\cdots$,$\sqrt{2} = 1.41421\cdots$ e.t.c. They contain infinitely many non-recurring digits. Therefore, the numbers obtained by retaining a few digits are called **approximate numbers**.

  Example: The approximate numbers, $e \approx 2.718$ and $\pi \approx 3.142$.
- **Significant digits (Figures)**: are the numbers of digits used to express the number. The digits $1, 2, 3, \cdots, 9$ are significant digits and 0 is also a significant digit **except** when it is used to fix the decimal point or used to fill the place of discarded digits.

  Example: $5879, 0.4762$ contains four significant digits, 0.00486 and 0.000382 contains three significant digits and 2.0682 contains five significant digits.

Example 1.3

Find the absolute, relative and percentage errors if x is rounded-off to three decimal digits, given $x = 0.005998$.
Solution: If x is rounded-off to three decimal places we get x = 0.006. Therefore,

$$\begin{aligned}\texttt{Error} &= \texttt{True value} - \texttt{Approximate value}\\ \texttt{Error} &= 0.005998 - 0.006\\ &= -0.000002.\end{aligned}$$

Therefore,

$$\begin{aligned}\texttt{Absolute Error } E_a &= |\texttt{Error}|\\ &= |-0.000002|\\ &= 0.000002,\end{aligned}$$

$$\begin{aligned}\texttt{Relative Error } E_r &= \frac{|\texttt{Error}|}{|\texttt{True value}|}\\ &= \frac{|-0.000002|}{|0.005998|}\\ &= 0.0033344 \text{ and}\\ \texttt{Percentage Error } E_p &= E_r \times 100\%\\ &= 0.0033344 \times 100\%\\ &= 0.33344\%\end{aligned}$$

1.2.3 Approximation of Errors

Often times the true value is unknown to us, which is usually the case in numerical computing. In this case we will have to quantify errors using approximate values only. For example, when an iterative method is used we get a approximate value at the end of each iteration. The approximate error (E_a) is defined as the difference between the current (present) approximate value and the previous approximation. In general,

$$\texttt{approximate error } (E_a) = \texttt{present approximation} - \texttt{previous approximation}.$$

Similarly we can calculate the relative approximate error (E_r) by dividing the `approximate error` by the `present approximate` value.

$$\texttt{relative approximate error}(E_r) = \frac{\texttt{approximate error}}{\texttt{present approximation}}.$$

Assume our iterative method yield a better approximation as the iteration carries on. Often times we can set an acceptable `tolerance` to stop the iteration when the relative approximate error is small enough. We often set the tolerance in terms of the number of significant digits - the number of digits that carry meaningful contribution to its precision. It corresponds to the number of digits in the scientific notation to represent a number's significand or mantissa.

An approximation rule for minimizing the error is as follows: if the absolute relative approximate error is less than or equal to a predefined tolerance then the acceptable error has been reached and no more iterations would be required.

1.3 Types of Errors

1.3.1 Truncation Errors

Truncation error refers to the error in a method, which occurs because some series (finite or infinite) is truncated to a fewer number of terms. Examples of this include the computation of a definite integral through approximation by a sum or the numerical integration of an ordinary differential equation by some finite difference method. Such errors are essentially algorithmic errors and we can predict the extent of the error that will occur in the method. Simply,

Definition 1.6 Truncation Error

Truncation error is defined as the error caused by truncating a mathematical formula or procedure.

Example 1.4

For example, the Maclaurin series for e^x is given as

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

This series has an infinite number of terms but when using this series to calculate e^x, only a finite number of terms can be used. For example, if one uses three terms to calculate e^x , then

$$e^x \approx 1 + x + \frac{x^2}{2!}.$$

Thus, the truncation error for such an approximation is

$$\begin{aligned}\texttt{Truncation Error } (E_T) &= e^x - \left(1 + x + \frac{x^2}{2!}\right)\\ &= \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots\end{aligned}$$

1.3.2 Round-off Errors

When a calculator or digital computer is used to perform numerical calculations, an unavoidable error, called round-off error, must be considered. This error arises due to the fact that floating point numbers are represented by finite precision on a computing device and it occurs because of the computing device's inability to deal with certain numbers. Thus, calculations are performed with approximate representations of the actual numbers which results in a round-off error. Let's see the result of round-off error in a real-world problem.

Example 1.5

Problems created by round off error

- 28 Americans were killed on February 25, 1991 by an Iraqi Scud missile in Dhahran, Saudi Arabia.
- The patriot defense system failed to track and intercept the Scud. Why?

Problem with Patriot missile

- The Patriot defense system consists of an electronic detection device called the range gate. It calculates the area in the air space where it should look for a Scud. To find out where it should aim next, it calculates the velocity of the Scud and the last time the radar detected the Scud. Time is saved in a register that has 24 bits length. Since the internal clock of the system is measured for every one-tenth of a second, 1/10 is expressed in a 24 bit-register as 0.00011001100110011001100. However, this is not an exact representation. In fact, it would need infinite numbers of bits to represent 1/10 exactly. So, the error in the representation in decimal format is

$$
\begin{aligned}
&= \frac{1}{10} - \big(0\times 2^{-1} + 0\times 2^{-2} + 0\times 2^{-3} + 1\times 2^{-4}\cdots + 1\times 2^{-22} \\
&\qquad + 0\times 2^{-23} + 0\times 2^{-24}\big) \\
&= 9.5\times 10^{-8}\frac{s}{0.1s}\times 100hr\times\frac{360s}{1hr} \\
&= 0.342s
\end{aligned}
$$

The battery was on for 100 consecutive hours, hence causing an inaccuracy of

$$
\begin{aligned}
&= 9.5 \times 10^{-8} \frac{s}{0.1s} \times 100hr \times \frac{360s}{1hr} \\
&= 0.342s
\end{aligned}
$$

- The shift calculated in the range gate due to 0.3433 s was calculated as 687 m. For the Patriot missile defense system, the target is considered out of range if the shift was going to more than 137 m.

For example a number like 1/3 may be represented as 0.33333 on a computing device. Then the round off error in this case is $1/3 - 0.33333 = 0.0000033\overline{3}$. There are also other numbers that cannot be represented exactly on a computing machine, for example, π and $\sqrt{2}$ are numbers that need to be approximated in computer calculations.

There are two major approaches in approximating the actual number in a computer: **chopping** and **rounding**:

- When **chopping** a number to a specified number of decimal places, say n, the first n digits of the mantissa are retained, simply chopping off the remain digits.
- When **rounding** a number, the computer chooses the closest number that is representable by the computer.

A natural question one may ask is what error is committed when a number is chopped or rounded to n digits? Consider the number

$$x = 0.d_1 d_2 \cdots d_n d_{n+1} \cdots , \tag{1.5}$$

then chopping to n digits produce and the number

$$fl(x) = 0.d_1 d_2 \cdots d_n, \tag{1.6}$$

with an error

$$
\begin{aligned}
\texttt{Error} = x - fl(x) &= 0.d_{n+1} d_{n+2} \cdots \times 10^{-n} \\
&\leq 0.99999 \cdots \times 10^{-n} \\
&\leq 10^{-n}.
\end{aligned} \tag{1.7}
$$

Does rounding x to n digits increase or decrease this error? We now show that error will decrease. Once again, consider:

$$x = 0.d_1 d_2 \cdots d_n d_{n+1} \cdots , \tag{1.8}$$

If $d_{n+1} < 5$, chop x to n. Then the error is with an error

$$\begin{aligned}\texttt{Error} = x - fl(x) &= 0.d_{n+1}d_{n+2}\cdots \times 10^{-n} \\ &\leq 0.49999\cdots \times 10^{-n} \\ &< \frac{1}{2} \times 10^{-n}.\end{aligned} \tag{1.9}$$

When $5 < d_{n+1} \leq 9$, add 0.5×10^{-n} and chop the result. If $x^* = x + 0.5 \times 10^{-n}$,

$$\texttt{round}(x) = \texttt{chop}(x^*) = 0.d_1d_2\cdots d_n^* \tag{1.10}$$

where $0 < d_n^* < 5$, and $d_n^* = d_n + 1$ The error is therefore,

$$\begin{aligned}|x - \texttt{round}(x)| &= |0.d_1d_2\cdots d_n^* - 0.d_1d_2\cdots d_nd_{n+1}\cdots| \\ &= |d_n^* - d_n - 0.d_{n+1}d_{n+2}\cdots| \times 10^{-n} \\ &= |1 - 0.d_{n+1}d_{n+2}\cdots| \times 10^{-n} \\ &< \frac{1}{2} \times 10^{-n}.\end{aligned} \tag{1.11}$$

The last inequality follows since $5 < d_{n+1} \leq 9$.

When $d_{n+1} = 5$, increase the d_n by unity if it is odd otherwise, leaves it unchanged and apply chopping.

Example: If the number $x = 11.675$ is round-off to two decimal places, then we get $x_a = 11.68$. However, if we round-off the number $x = 11.685$ to two decimal places we $x_a = 11.68$.

In summary, the error committed by rounding a normalized number x to n digits (in **base** 10) is $\frac{1}{2} \times 10^{-n}$, which is half the maximum error committed by chopping.

Definition 1.7 **True relative error**

The formula for the relative error

$$\frac{x - fl(x)}{x} = -\epsilon_{rel}, \tag{1.12}$$

is usually written as

$$fl(x) = (1 + \epsilon_{rel})x.$$

True Relative Error and Significant digits

If the number x is rounded to n significant digits, then $\boldsymbol{\epsilon}_{abs} = \frac{1}{2} \times 10^{-n}$

Example: If $x = 0.51$ and correct to two decimal places. Then $E_a = 0.005$ and the relative accuracy is given by

$$\begin{aligned} E_p = E_r \times 100\% &= \frac{0.005}{0.51} \times 100\% \\ &= 0.98\% \end{aligned}$$

Our intuition tells us that the more accurate a number is, the more digits (that follow the decimal point) will be correct, once the number is expressed in normalized format. To make this statement more precise, we relate the number of correct digits to the relative error.

Definition 1.8

Let p^* approximates p to t significant digits if t is the largest integer > 0 such that

$$E_r = \frac{|p - p^*|}{|p|} \leq 5 \times 10^{-t}.$$

Taking $\log_{10}$ of both sides results in

$$\begin{aligned} \log_{10}(E_r) &\leq \log_{10}\left(5 \times 10^{-t}\right) \\ &= \log_{10} 5 - t, \end{aligned}$$

so that

$$t \leq \log_{10} \frac{5}{E_r}.$$

A related statement is that if x and y are normalized floating-point machine numbers such that $x > y > 0$ and $1 - \frac{x}{y} \geq 2^{-k}$, then at most k significant binary bits are lost in the subtraction $x - y$.

Example 1.6

Given a relative error $E_r = 0.5$, how many significant digits do we have?

Solution:

$$t \leq \log_{10} \frac{5}{5 \times 10^{-1}} = 1,$$

thus, we have one significant digit.

Given a relative error $E_r = 0.1$

$$t \leq \log_{10} \frac{5}{10^{-1}} = \log_{10} 50$$

so that $1 \leq t \leq 2$ and hence we have again one significant digit.

Example 1.7

Consider $x = 3.29$ and $fl(x) = 3.2$. How accurate is the approximation $fl(x)$?

Solution: The true relative error is

$$\begin{aligned} E_r &= \frac{3.29 - 3.2}{3.29} \\ &= \frac{9 \times 10^{-2}}{3.29} \\ &\approx 3 \times 10^{-2}, \end{aligned}$$

so that

$$t \leq \log_{10} \frac{5}{3 \times 10^{-2}} = 2 + \log_{10}\left(\frac{5}{3}\right),$$

which implies that t lies between 2 and 3. Thus, there are 2 significant digits.

The number of significant digits is only weakly dependent on the value of the relative error mantissa. For example, as long as

$$E_r \in \left[0.5 \times 10^{-2}, 5.0 \times 10^{-2}\right], \tag{1.13}$$

there are only two significant digits.

Example 1.8

Another way to pose the question is given $x = 3.2$, what is the worst possible approximation to x which is accurate to 2 significant digits?

Solution: Let the approximation be x^*. Therefore

$$2 = t = \log_{10} \frac{5}{\left|\frac{3.2 - x^*}{3.2}\right|}.$$

Taking the antilogarithm of both sides (10 to the power),

$$100 = \frac{5}{|\frac{3.2-x^*}{3.2}|}.$$

Rearanging terms,

$$\frac{20}{3.2} = |3.2 - x^*|^{-1}$$

or

$$|3.2 - x^*| \approx \frac{1}{6.25} = 0.16$$

The worst approximation to x while retaining 2 significant digits is therefore $x^* = 3.04$ or $x^* = 3.36$.

Example 1.9

Given the solution of a problem as $x_a = 35.25$ with the relative error in the solution at most 0.02. Find, to four decimal digits, the range of values within which the exact value of the solution must lie.

Solution: If x_t is the exact value of the solution, then according to given information in the question we have

$$\begin{aligned}
\Longrightarrow \quad & E_r = \frac{E_a}{|x_t|} = \frac{|x_t - x_a|}{|x_t|} = |\frac{x_t - x_a}{x_t}| < 0.02, \\
\Longrightarrow \quad & |1 - \frac{x_a}{x_t}| < 0.02 \\
\Longrightarrow \quad & -0.02 < 1 - \frac{x_a}{x_t} < 0.02, \\
\Longrightarrow \quad & -0.02 < 1 - \frac{x_a}{x_t} \ \text{ and } \ 1 - \frac{x_a}{x_t} < 0.02, \\
\Longrightarrow \quad & \frac{x_a}{x_t} < 1.02 \ \text{ and } \ 0.98 < \frac{x_a}{x_t}, \\
\Longrightarrow \quad & \frac{x_a}{1.02} < x_t \ \text{ and } \ x_t < \frac{x_a}{0.98}, \\
\Longrightarrow \quad & \frac{x_a}{1.02} < x_t < \frac{x_a}{0.98}
\end{aligned}$$

Now, substituting the approximate value $x_a = 35.25$ we get

$$\begin{aligned}
& \frac{35.25}{1.02} < x_t < \frac{35.25}{0.98} \\
\Longrightarrow \quad & 34.55882353 < x_t < 35.96938775
\end{aligned}$$

Hence, correct to 4 decimal digits, the range of values within which the exact value of the solution lies, is $34.5588 < x_t < 35.9694$.

The last the last tow example demonstrates that the concept of significant digits is not simply a matter of counting the number of digits which are correct.

1.3.3 Propagation Errors

Error analysis of algorithms generally assumes perfect precision, ie. no round-off error. However, it is there and is worth keeping it in mind. Especially if you are doing many sequential calculations where the output from one is input into another. In this way, errors can be propagated and your final answer can be garbage.

If a calculation is made with numbers that are not exact, then the calculation itself will have an error. How do the errors in each individual number propagate through the calculations? Let's look at the concept via some examples.

Example 1.10

Find the bounds for the propagation error in adding two numbers. For example if one is calculating $x + y$ where

$$x = 1.5 \pm 0.05, y = 3.4 \pm 0.04.$$

Solution: By looking at the numbers, the maximum possible value of x and y are $x = 1.55$ and $y = 3.44$. Hence

$$x + y = 1.55 + 3.44 = 4.99$$

is the maximum value of $x + y$.
The minimum possible value of x and y are $x = 1.45$ and $y = 3.36$. Hence

$$x + y = 1.45 + 3.36 = 4.81$$

is the minimum value of $x + y$. Therefore

$$4.81 \leq x + y \leq 4.99.$$

Here, we can see that the exact sum of the two numbers is 4.9 and has an absolute error of 0.09 which is greater than the error in x and y.

Example 1.11

Suppose a real number x is represented by the machine as x^* where $x^* = x(1+\delta)$ and δ (the initial relative error) is small. Suppose we want to calculate x^2. In the machine we will get $(x^*)^2 = x^2(1+\delta)^2$. Now we get an error

$$\begin{aligned} E &= (x^*)^2 - x^2 \\ &= x^2\left[(1+\delta)^2 - 1\right] \qquad \text{since } \delta^2 \ll 1. \\ &\approx x^2(2\delta) \end{aligned}$$

which can be very big, especially if x is big. If we look at relative error$=\frac{E}{x^2}$, we still get a relative error of 2δ . Notice, the relative error doubled. This is an example of an error being propagated.

1.4 General Error Formula

What if the evaluations we are making are function evaluations instead of arithmetic operations? How do we find the value of the propagation error in such cases. In the section, we derive a general formula for the error committed in using a certain formula for a functional relation. Consider

$$\boldsymbol{u} = f(x_1, x_2, \cdots, x_n),$$

be a function of several variables $x_1, x_2, \cdots, x_n$ and let the error in each x_i be $\triangle x_i$. Then the error $\triangle \boldsymbol{u}$ in $\boldsymbol{u}$ is given by

$$\boldsymbol{u} + \triangle \boldsymbol{u} = f(x_1 + \triangle x_1, x_2 + \triangle x_2, \cdots, x_n + \triangle x_n)$$

Using Taylor's series expansion of f about $(x_1, x_2, \cdots, x_n)$ in the above equation we get

$$\begin{aligned}\boldsymbol{u} + \triangle\boldsymbol{u} = {} & f(x_1, x_2, \cdots, x_n) + \\ & \triangle x_1 \frac{\partial \boldsymbol{u}}{\partial x_1} + \triangle x_2 \frac{\partial \boldsymbol{u}}{\partial x_2} + \cdots + \triangle x_n \frac{\partial \boldsymbol{u}}{\partial x_n} + \\ & \frac{1}{2!}\triangle x_1^2 \frac{\partial^2 \boldsymbol{u}}{\partial x_1^2} + \frac{1}{2!}\triangle x_1 \triangle x_2 \frac{\partial^2 \boldsymbol{u}}{\partial x_1 \partial x_2} + \cdots + \frac{1}{2!}\triangle x_1 \triangle x_n \frac{\partial^2 \boldsymbol{u}}{\partial x_1 \partial x_n} + \\ & \frac{1}{2!}\triangle x_1 \triangle x_2 \frac{\partial^2 \boldsymbol{u}}{\partial x_1 \partial x_2} + \frac{1}{2!}\triangle x_2^2 \frac{\partial^2 \boldsymbol{u}}{\partial x_2^2} + \cdots + \frac{1}{2!}\triangle x_2 \triangle x_n \frac{\partial^2 \boldsymbol{u}}{\partial x_2 \partial x_n} + \\ & \vdots \\ & \frac{1}{2!}\triangle x_1 \triangle x_n \frac{\partial^2 \boldsymbol{u}}{\partial x_1 \partial x_n} + \frac{1}{2!}\triangle x_2 \triangle x_n \frac{\partial^2 \boldsymbol{u}}{\partial x_2 \partial x_n} + \cdots + \frac{1}{2!}\triangle x_n^2 \frac{\partial^2 \boldsymbol{u}}{\partial x_n^2} + \\ & \vdots \end{aligned}$$

Assuming the errors $\triangle x_1, \triangle x_2, \cdots, \triangle x_n$ in x_i all are small and that $\frac{\triangle x_i}{x_i} \ll 1$ so that that the terms containing $\triangle x_1^2, \triangle x_2^2, \cdots, \triangle x_n^2$ and higher powers of $\triangle x_1, \triangle x_2, \cdots, \triangle x_n$ are being neglected. Therefore,

$$\boldsymbol{u} + \triangle\boldsymbol{u} \approx f(x_1, x_2, \cdots, x_n) + \triangle x_1 \frac{\partial \boldsymbol{u}}{\partial x_1} + \triangle x_2 \frac{\partial \boldsymbol{u}}{\partial x_2} + \cdots + \triangle x_n \frac{\partial \boldsymbol{u}}{\partial x_n},$$

which implies that

$$\boxed{\triangle\boldsymbol{u} \approx \triangle x_1 \frac{\partial \boldsymbol{u}}{\partial x_1} + \triangle x_2 \frac{\partial \boldsymbol{u}}{\partial x_2} + \cdots + \triangle x_n \frac{\partial \boldsymbol{u}}{\partial x_n}. \qquad (1.14)}$$

Equation (1.14) represents the **general formula for errors**. If we divide the above equation by $\boldsymbol{u}$ on both sides we get the relative error

$$\boxed{E_r = \frac{\triangle\boldsymbol{u}}{\boldsymbol{u}} \approx \frac{\triangle x_1}{\boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial x_1} + \frac{\triangle x_2}{\boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial x_2} + \cdots + \frac{\triangle x_n}{\boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial x_n}, \qquad (1.15)}$$

Also from Equation (1.14), by taking modulus we get **maximum absolute error**,

$$\boxed{|\triangle\boldsymbol{u}| \leq |\triangle x_1 \frac{\partial \boldsymbol{u}}{\partial x_1}| + |\triangle x_2 \frac{\partial \boldsymbol{u}}{\partial x_2}| + \cdots + |\triangle x_n \frac{\partial \boldsymbol{u}}{\partial x_n}|. \qquad (1.16)}$$

In addition, from Equation (1.17), by taking the modulus we get the **maximum relative error** as

$$E_r \leq \left|\frac{\triangle x_1}{\boldsymbol{u}}\frac{\partial \boldsymbol{u}}{\partial x_1}\right| + \left|\frac{\triangle x_2}{\boldsymbol{u}}\frac{\partial \boldsymbol{u}}{\partial x_2}\right| + \cdots + \left|\frac{\triangle x_n}{\boldsymbol{u}}\frac{\partial \boldsymbol{u}}{\partial x_n}\right|, \qquad (1.17)$$

Example 1.12

Let $\boldsymbol{u} = \frac{5xy^2}{z^3}$, with $\triangle x = \triangle y = \triangle z = 0.0001$ and $x = y = z = 1$. Find the maximum absolute error and relative errors.

Solution:

$\frac{\partial \boldsymbol{u}}{\partial x} = \frac{5y^2}{z^3}; \frac{\partial \boldsymbol{u}}{\partial y} = \frac{10xy}{z^3}; \frac{\partial \boldsymbol{u}}{\partial z} = \frac{-15y^2}{z^4}.$

Thus, the absolute error bound is given as

$$\begin{aligned}(\triangle \boldsymbol{u})_{max} &= |\triangle x\frac{5y^2}{z^3}| + |\triangle y\frac{10xy}{z^3}| + |\triangle x\frac{-15y^2}{z^4}| \\ & |0.0001 \times 5| + |0.0001 \times 10| + |0.0001 \times -15| \\ &= 0.003,\end{aligned}$$

and the relative error bound is given by

$$E_r = \frac{(\triangle \boldsymbol{u})}{\boldsymbol{u}} = \frac{0.003}{5} = 0.0006.$$

1.5 Inverse problems of the theory of errors

Definition 1.9

Error Inverse Problem What must the absolute errors of the independent variables of a function be so that the absolute error of the function does not exceed a given quantity?

Given the errors of several independent quantities or approximate numbers, the direct problem requires us to find the error of any function of these quantities.However,the inverse problem requires us to find the allowable errors in several independent quantities in order to obtain a prescribed degree of accuracy in any function.The direct problem is straightforward.The formula to be used is:

$$\Delta \boldsymbol{u} = \frac{\partial f}{\partial x_1}\Delta x_1 + \frac{\partial f}{\partial x_2}\Delta x_2 + \cdots + \frac{\partial f}{\partial x_n}\Delta x_n.$$

However, the inverse problem,namely,finding the allowable errors in $x_1, x_2, \cdots, x_n$ when $\boldsymbol{u}$ is of a desired accuracy, is mathematically indeterminate since there is only one equation for $\Delta \boldsymbol{u}$ and there are several unknowns $\Delta x_1, \Delta_2, \cdots, \Delta x_n$.

The problem is solved with the minimum effort by using what is known as the principle of equal effects.This principle assumes that the partial differentials

$$\frac{\partial f}{\partial x_i}\Delta x_i,\ i = 1, 2, \cdots, n,$$

are equal. Thus, we have

$$\Delta \boldsymbol{u} = n\frac{\partial f}{\partial x_i}\Delta x_i$$
$$\implies \quad \Delta x_i = \frac{\Delta \boldsymbol{u}}{n\frac{f}{\partial x_i}}.$$

Therefore, the absolute error of each independent variable is given by

The Inverse Problem

$$\Delta x_i = \frac{\Delta \boldsymbol{u}}{n\frac{f}{\partial x_i}},\ \text{for } i = 1, 2, \cdots n, \tag{1.18}$$

where n is the number of independent variables.

Example 1.13

The base of a cylinder has radius $r \approx 2m$, the altitude of the cylinder is $h \approx 3m$. With what absolute errors must we determine r and h so that the volume v may be computed within $0.1m^3$?

Solution: We have $v = \pi r^r h$ and $\Delta v = 0.1m^3$. Here we can see see that v is function of three variables, i.e., π, r and h. Thus, putting $r = 2m$, $h = 3m$ and

$\pi = 3.14$, we approximately get:

$$\frac{\partial v}{\partial \pi} = r^2h = 2^2 \times 3m^3 = 12m^3$$
$$\frac{\partial v}{\partial r} = 2\pi rh = 2 \times \pi \times 2 \times 3m^3 = 37.7m^3$$
$$\frac{\partial v}{\partial h} = \pi r^2 = \pi \times 2^2m^3 = 12.6m^3$$

Since $n = 3$, using the inverse formula (1.18) above we have

$$\Delta r = \frac{\Delta v}{3 \times \frac{\partial v}{\partial r}} = \frac{0.1}{3 \times 37.7} = 0.000884173298 < 0.001$$
$$\Delta h = \frac{\Delta v}{3 \times \frac{\partial v}{\partial r}} = \frac{0.1}{3 \times 12.6} = 0.0026455026455 < 0.003$$

Chapter 2

Nonlinear Equations

2.1 Locating Roots

Solving nonlinear equations is one of the most important and challenging problems in science and engineering applications. The root finding problem is one of the most relevant computational problems. It arises in a wide variety of practical applications in Physics, Chemistry, Biosciences, Engineering, etc.

The problem of nonlinear root finding can be stated in an abstract sense as follows:

`Given some function` $f(x)$, `determine the value(s) of` x `such that` $f(x) = 0$.

The solution x is called the **root** of the equation or the **zero** of the function f and the problem is called **root finding** or **zero finding**.

The common root-finding methods include: *Bisection*, *Newton-Raphson*, *False position*, *Secant methods* etc. Different methods converge to the root at different rates. That is, some methods are faster in converging to the root than others. The rate of convergence could be linear, quadratic, etc. The higher the order, the faster the method converges.

The central concept to all root finding methods is **iteration** or **successive approximation**. The idea is that we make some guess at the solution, and then we repeatedly improve upon that guess, using some well-defined operations, until we arrive at an approximate answer that is sufficiently close to actual answer. We refer to this process as an **iterative method**. We call the sequence of approximations the iterates and denote them by $x_0, x_1, x_2, \cdots, x_n, \cdots$.

Iterative methods generally involve an infinite number of steps to obtain the exact solution. However, the beauty and power of these methods is that typically after a finite, relatively small number of steps the iteration can be terminated with the last iterate providing a very good approximation to the actual solution. One of the primary concerns of an iterative method is thus the rate at which it converges to the actual solution, called **order of convergence**.

Definition 2.1 Order of Convergence:

Let $\alpha \in \mathbb{R}$ be the actual solution of $f(x) = 0$. A sequence of iterates,$x_n \in \mathbb{R}, n = 0, 1, 2, \cdots$, is said to be convergent to α if

$$\lim_{n\to\infty} |x_n - \alpha| = 0.$$

Thus, if there exists a constant $c > 0$, an integer $N_0 \geq 0$ and $p \geq 0$ such that for all $n > N_0$ we have

$$|\alpha - x_n| \leq c|\alpha - x_{n-1}|^p, \texttt{i.e.,} \quad \frac{|\alpha - x_n|}{|\alpha - x_{n-1}|^p} \to c \quad \text{as } n \to \infty, \tag{2.1}$$

then we say the sequence of iterates converges with an **order** at least p to α.
If $p = 1$ and $c < 1$ then the convergence is called **linear** and we can write Equation (6.1) as

$$|\alpha - x_n| \approx c^n|\alpha - x_0|, \qquad \text{as } n \to \infty. \tag{2.2}$$

If $p > 1$ then the convergence is called **superlinear** for any $c > 1$. In particular, the values $p = 2$ and $p = 3$ are given the special names **quadratic** and **cubic** convergence, respectively.

Definition 2.2 Error Equation:

Notation: The notation $e_n = |x_n - \alpha|$, is the error in the n^{th} iteration. The equation

$$e_{n+1} = ce_n^p + O(e_n^{p+1}) \tag{2.3}$$

is called the **error equation.** By substituting $e_n = x_n - \alpha$ for all n in any iterative method and simplifying we obtain the error equation for that method. The value of p obtained is called the **order of convergence** of the method.

In addition to the order of convergence, the factors for deciding whether an iterative method for root finding problem is good or not are **accuracy**, **stability**, **efficiency**, and **robustness**. Each of these can be defined as follows:

- **Accuracy**: The error $|\alpha - x_n|$ becomes small as n is increased.
- **Stability**: If the input parameters are changed by small amounts the output of the iterative method should not be wildly different, unless the underlying problem exhibits this type of behavior.
- **Efficiency**: The number of operations and the time required to obtain an approximate answer should be minimized.
- **Robustness**: The iterative method should be applicable to a broad range of inputs.

In the iterative methods that we study we will see how each one of these concepts applies.

2.2 Bisection Method

This is one of the simplest iterative methods and is strongly based on the property of intervals (bracketing). The bisection method is a bracketing method for finding a numerical solution of an equation of the form $f(x) = 0$.

As the name suggests, the method is based on repeated bisections of an interval containing the root. The basic idea is very simple. Suppose we now want to approximate the solution to $f(x) = 0$ for a general *continuous function* $f(x)$. The key to the bisection method is to keep the actual solution bracketed between the guesses. Thus, in addition to being given $f(x)$, we need an interval $a \le x \le b$ where $f(a)$ and $f(b)$ differ in sign. We can write this requirement mathematically as

$$f(a) \times (b) < 0. \tag{2.4}$$

It seems reasonable to conclude that since $f(x)$ is continuous and has different signs at each end of the interval $[a, b]$, there must be at least one point $\alpha \in [a, b]$, such that $f(\alpha) = 0$. Thus, $f(x)$ has at least one root in the interval. This result is in fact known as the corollary of **Intermediate Value Theorem.**

Theorem 2.1 Intermediate value theorem

Suppose f is continuous on a closed interval $[a,b]$. Let p be any number between $f(a)$ and $f(b)$ so that $f(a) \leq p \leq f(b)$ or $f(b) \leq p \leq f(a)$. Then there exists a number c in (a,b) such that $f(c) = p$.

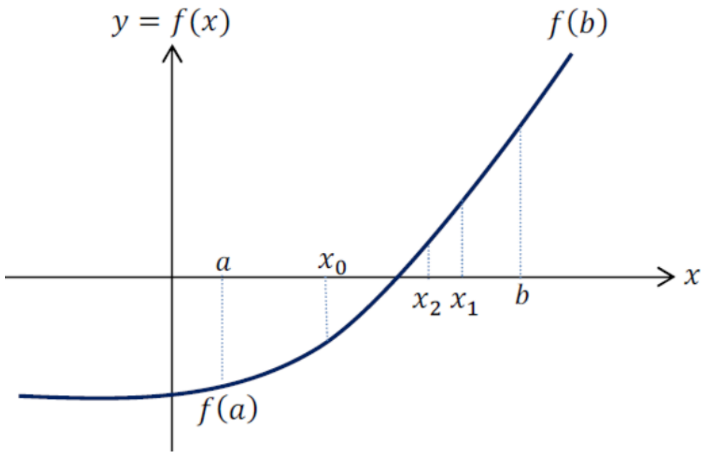


Figure 2.1: Root finding using Bisection method

At each step the method divides the interval in two by computing the midpoint $c = (a+b)/2$ of the interval and the value of the function $f(c)$ at that point. Unless c is itself a root (which is very unlikely, but possible) there are now only two possibilities: either $f(a)$ and $f(c)$ have opposite signs and bracket a root, or $f(c)$ and $f(b)$ have opposite signs and bracket a root. The method selects the sub-interval that is guaranteed to be a bracket as the new interval to be used in the next step. In this way an interval that contains a zero of f is reduced in width by 50% at each step.

The process is continued until the interval is sufficiently small. Explicitly, if $f(a)$ and $f(c)$ have opposite signs, then the method sets c as the new value for b, and if $f(b)$ and $f(c)$ have opposite signs then the method sets c as the new a. (If $f(c) = 0$ then c may be taken as the solution and the process stops.) In both cases, the new $f(a)$ and $f(b)$ have opposite signs, so the method is applicable to this smaller interval.

Now that we have an idea for how the bisection method works for a general problem $f(x) = 0$, it is time to write down a formal procedure for it using well defined operations. We call such a procedure an **algorithm**.

Algorithm 1: Bisection Method

```
Input: Continuous function $f(x)$ ;
       Interval $[a, b]$ such that $f(a)f(b) < 0$ ;
       Error tolerance $\epsilon$. ;
Output: Approximate solution that is within $\epsilon$ of a root of $f(x)$ ;
Step 1. $n = -1$ initialize the counter ;
Step 2. while $b - a \geq 2\epsilon$ do
    $x_{n+1} = \frac{a+b}{2}$ (bisect the interval) ;
    if $f(x_{n+1}) == 0$ then
        return $x_{n+1}$ (we have found a solution);
    end
    if $f(x_{n+1})f(a) \leq 0$ then
        $b = x_{n+1}$
    else
        $a = x_{n+1}$
    end
    $n = n + 1$ (update the counter)
end
Step 3. $x_{n+1} = \frac{b+a}{2}$ (bisect the interval one last time)
 ;
Step 4. return $x_{n+1}$ (return the solution)
```

We can think of an algorithm as a *receipe* for solving some mathematical problem. However, instead of the basic ingredients of flour, sugar, eggs, and salt, the fundamental building blocks of an algorithm are the basic mathematical operations of addition, subtraction, multiplication, and division, as well as the **for**, **if**, and **while** constructs.

In the Bisection Algorithm 1, the line **while** $b - a > 2\epsilon$ in this algorithm is called the **stopping criterion**, and we call ϵ the **error tolerance**. This line says that we are going to continue bisecting the interval until the length of the interval is $\leq 2\epsilon$. This guarantees that the value returned by the algorithm is at most a distance ϵ away from the actual solution. The value for ϵ is given as an input to the algorithm.

Note that the smaller the value of ϵ, the longer it takes the bisection method to converge. Typically, we choose this value to be something small like $\epsilon = 10^{-6}$. The stopping criterion that we have chosen is called an **absolute** criterion. Some other types of criterion are **relative** and **residual**. These correspond to $b - a < 2\epsilon|x_n|$ and $|f(x_n)| \leq \epsilon$, respectively. There is no correct choice.

2.2.1 Number of Iterations Needed in the Bisection Method to Achieve Certain Accuracy:

Let a_n, b_n and x_n denote the n^{th} computed values of a, b and x respectively. Then, we have

$$b_{n+1} - a_{n+1} = \frac{1}{2}(b_n - a_n), \qquad n \geq 1,$$

also

$$b_n - a_n = \frac{1}{2^{n-1}}(b - a), \qquad n \geq 1, \tag{2.5}$$

where $(b - a)$ is the length of the original interval with which we started. Since the root α is in either the interval (a_n, x_n) or (x_n, b_n), we know that

$$|\alpha - x_n| \leq x_n - a_n = b_n - x_n = \frac{1}{2}(b_n - a_n) \tag{2.6}$$

The above Equation (6.5) is the **error bound** for x_n that is used in **Step 2** of Algorithm 1. Now, combining Equation (2.5) and (6.5) we get the further bound

$$|\alpha - x_n| \leq \frac{1}{2^n}(b - a) \tag{2.7}$$

which shows that the iterate x_n converges to α as $n \to \infty$. Therefore, the bisection method **converges linearly** to the solution at a rate of $\frac{1}{2}$. Note that this bound is entirely independent of the function $f(x)$.

Let us now find out what is the **minimum number of iterations** n needed with the bisection method to achieve a certain desired accuracy, say ϵ, suppose we want to have

$$|\alpha - x_n| \leq \frac{1}{2^n}(b - a) \leq \epsilon.$$

Taking logarithms, *(with any convenient base)*, of both sides of the above equation and simplifying the resulting expression we obtain

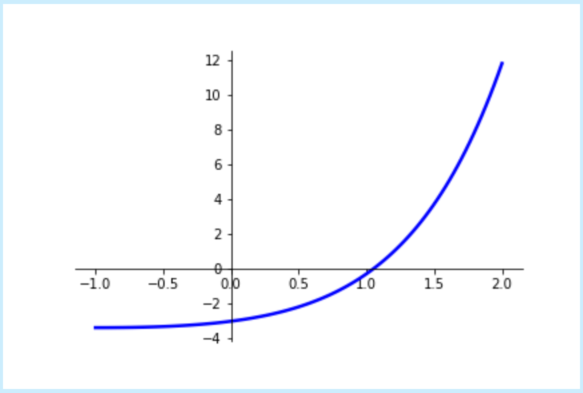
$$\boxed{n \geq \frac{\log \frac{b-a}{\epsilon}}{\log 2} \qquad (2.8)}$$

Example 2.1

Find the largest root of

$$f(x) = x^6 - x - 1 = 0$$

accurate to within $\epsilon = 0.001$. **Solution:** With the help of the following graph, it is easy to check that $1 < \alpha < 2$



We choose $a = 1$, $b = 2$; then $f(a) = -1$, $f(b) = 61$, and Equation (2.4) is satisfied. Thus, applying the Bisection Methods results in

i	**a**	**b**	**c**	$f(a)$	$f(b)$	$f(c)$	$f(a) \times f(c)$	$\|b-a\|$
1	1.0000	2.0000	1.5000	-1.0000	61.0000	8.8906	-8.8906	1.0000
2	1.0000	1.5000	1.2500	-1.0000	8.8906	1.5647	-1.5647	0.5000
3	1.0000	1.2500	1.1250	-1.0000	1.5647	-0.0977	0.0977	0.2500
4	1.1250	1.2500	1.1875	-0.0977	1.5647	0.6167	-0.0603	0.1250
5	1.1250	1.1875	1.1562	-0.0977	0.6167	0.2333	-0.0228	0.0625
6	1.1250	1.1562	1.1406	-0.0977	0.2333	0.0616	-0.0060	0.0312
7	1.1250	1.1406	1.1328	-0.0977	0.0616	-0.0196	0.0019	0.0156
8	1.1328	1.1406	1.1367	-0.0196	0.0616	0.0206	-0.0004	0.0078
9	1.1328	1.1367	1.1348	-0.0196	0.0206	0.0004	-0.0000	0.0039
10	1.1328	1.1348	1.1338	-0.0196	0.0004	-0.0096	0.0002	0.0020

Therefore, the largest root for the given function is 1.134765625 which is 5 correct significant digits

Example 2.2

Approximate $\sqrt{2}$ with an accuracy of $\epsilon = 10^{-3}$ and also compute the minimum number of iterations need.

Solution: Let $x = \sqrt{2}$ be a root of a function f. First let's find the function f,

$$\alpha = \sqrt{2} \implies \alpha^2 = 2 \implies \alpha^2 - 2 = 0$$

Thus, let $f(x) = x^2 - 2$ and take the interval $[1, 2]$, we have $f(1) * f(2) < 0$ and $\sqrt{2} \in [1, 2]$, the convergence of the bisection method is guaranteed on this interval. Thus, the number of minimum iteration is given by

$$n \geq \frac{\log \frac{2-1}{10^{-3}}}{\log 2} = \frac{3}{\log_{10} 2} \approx 9.965784$$

Thus, the minimum number iterations required to have the given accuracy is 10. The table below shows the numerical value and error of the first 10 iterates of the bisection algorithm for approximating $\sqrt{2}$ using the function $f(x) = x^2 - 2$ on the interval $[1, 2]$ with an error tolerance $\epsilon = 10^{-3}$.

n	a	b	c	$sign(f(a)) * sign(f(c))$	$b - a$	$\lvert\alpha - c\rvert$
1	1.00000	2.00000	1.50000	$-ve$	1.00000	0.08579
2	1.00000	1.50000	1.25000	+ve	0.50000	0.16421
3	1.25000	1.50000	1.37500	+ve	0.25000	0.03921
4	1.37500	1.50000	1.43750	$-ve$	0.12500	0.02329
5	1.37500	1.43750	1.40625	+ve	0.06250	0.00796
6	1.40625	1.43750	1.42188	$-ve$	0.03125	0.00766
7	1.40625	1.42188	1.41406	+ve	0.01562	0.00015
8	1.41406	1.42188	1.41797	$\vdots$	0.00781	0.00376
9	1.41406	1.41797	1.41602		0.00391	0.00180
10	1.41406	1.41602	1.41504		0.00195	0.00083

As we can see from the table it takes a minimum of 10 iteration to have an accuracy of 10^{-3}, i.e., $\frac{|b-a|}{2} < \epsilon$. In addition we can see that $|\alpha - c| \leq \frac{1}{2}|b - a|$.

The most difficult part about using the bisection method is finding an interval $[a, b]$ where the continuous function $f(x)$ changes sign. Once this is found, the algorithm is guranteed to converge. Thus, we would say that the bisection method is **very robust**. Also, as long as $f(x)$ has only one root between the interval $[a, b]$, and it does not have another root very close to a or b, we can make small changes to a or b and the method will converge to the same solution. Thus, we would say the bisection method is **stable**. Additionally, Equation (2.7) tells us that the error $|\alpha - x_n|$ can be made as small as we like by increasing n. Thus, we would say the bisection method is **accurate**. Finally, the method converges linearly which is acceptable, but, as we will see in the next two sections, it is by no means the best we can do. Thus, we would say that the bisection method is **not very efficient**.

Advantages and disadvantages of the bisection method

- The method is guaranteed to converge
- The error bound decreases by half with each iteration
- The bisection method converges very slowly
- The bisection method cannot detect multiple roots

2.3 False Position (Regular-falsi) Method

The false position method retains the main features of the Bisection method, that the root is trapped in a sequence of intervals of decreasing size. Therefore, The regula falsi method is a bracketing method. This method uses the point where the **secant line** intersect the x-axis. The secant line over the interval $[a, b]$ is the chord between $(a, f(a))$ and $(b, f(b))$ as shown in Figure 6.3. The two right triangles angles in the figure are similar, which mean that we have

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

This implies that

$$c = \frac{af(b) - bf(a)}{f(b) - f(a)} \tag{2.9}$$

then we can compute $f(c)$ and repeat the process with the interval $[a, c]$, if $f(a) \times f(c) < 0$ or to the interval $[c, b]$, if and only if $f(c) \times f(b) < 0$.

The rate of convergence is still linear but faster than that of the bisection method. Both the Bisection and Regular-falsi methods will fail if f has a **double root**.

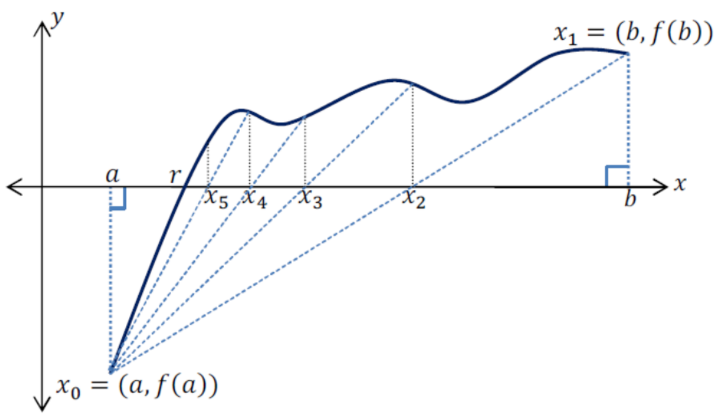


Figure 2.2: Root finding using Regular-Falsi method

The formal procedure of the Regular-falsi method is given below in Algorithm 2.

Algorithm 2: Regular-Falsi Method

```
Input: Continuous function $f(x)$ ;
       Interval $[a, b]$ such that $f(a)f(b) < 0$ ;
       Error tolerance $\epsilon$. ;
Output: Approximate solution that is within $\epsilon$ of a root of $f(x)$ ;
Step 1. $c = b$ (for Residual criteria) ;
Step 2. while $|f(c)| \geq \epsilon$ do
    $c = \dfrac{af(b) - bf(a)}{f(b) - f(a)}$ ;
    if $f(c) == 0$ then
        return $c$ (we have found a solution);
    end
    if $f(a)f(c) \leq 0$ then
        $b = c$
    else
        $a = c$
    end
end
Step 3. return $c$ (return the solution)
```

We can rewrite the above algorithm as follows so that we can use the iteration number. Given an interval $[x_0, x_1]$ such that $sing(f(x_0)) \times sign(f(x_1)) < 0$, then there exists a root on this interval and the next approximate root, x_2, is computed as

$$x_2 = \frac{x_0 \times f(x_1) - x_1 \times f(x_0)}{f(x_1) - f(x_0)}.$$

Now, check if this approximate root is within the given tolerance, i.e., if $f(x_2) < \epsilon$ then we take x_2 as an approximate root other wise we need to find the next new interval depending on the sing of $f(x_0) \times f(x_2)$, if the sign is negative then the root lies in $[x_0, x2]$, otherwise the root lies in $x_2, x_1]$. Thus, the next approximate root, x_3 can be computed as

$$x_3 = \frac{x_0 \times f(x_2) - x_2 \times f(x_0)}{f(x_2) - f(x_0)}.$$

if the root lies in the first interval, x_3 can be computed as

$$x_3 = \frac{x_1 \times f(x_2) - x_2 \times f(x_1)}{f(x_2) - f(x_1)}.$$

We repeat this procedure until $f(x_{n+}) < \epsilon$.

In general, at the n^{th} iteration we have the interval $[x_{n-1}, x_n]$ such that $f(x_{n-1}) \times f(x_n) < 0$ and the next approximate root is given as

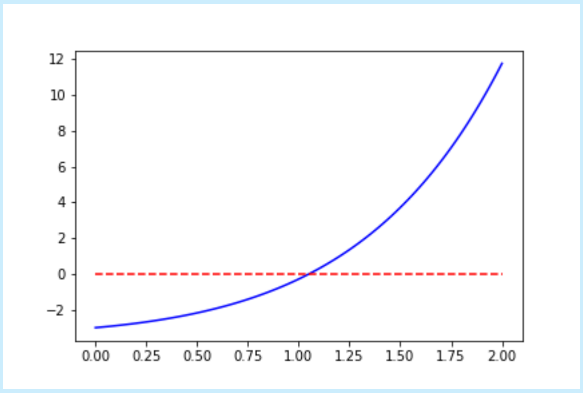
$$x_{n+1} = \frac{x_{n-1} \times f(x_n) - x_n \times f(x_{n-1})}{f(x_n) - f(x_{n-1})}.$$

or,

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

Example 2.3

Find a real root of the equation $f(x) = xe^x - 3$ using Regula-falsi method correct to three decimal places.



Solution: As seen from the graph of f we can assume $x_0 = 1$ and $x_1 = 1.5$ as initial guess values then $f(1) = -0.2817$ and $f(1.5) = 3.7225$ and hence the root lies between 1 and 1.5.

- **Iteration-1**: $x_2 = \dfrac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)} = \dfrac{1(3.7225) - 1.5 \times (-0.2817)}{(3.7225 - (-0.2817))} = 1.0352$

Since, $f(x_2)$ is negative, the next approximate root lies between x_1 and x_2 and also none of the decimal digits in x_1 and x_2 are correct.

- **Iteration-2**: $x_3 = \dfrac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)} = 1.0456$

  Now, $f(x_3)$ is negative and hence the root again lies between x_1 and x_3.

In similar manner, $x_4 = 1.0487$, $x_5 = 1.0496$ and $x_6 = 1.0498$. Hence the approximate root is 1.0498 correct to three decimal places.
The approximate for exact root is $\alpha = 1.0499088949636644$

n	x_{n-1}	x_n	x_{n+1}	$f(x_{n-1}) \times f(x_{n+1})$	$f(x_{n+1})$	$\alpha - x_{n+1}$
1	1.00000	1.50000	1.03518	-ve	0.08535	0.01473
2	1.03518	1.50000	1.04560	-ve	0.02518	0.00431
3	1.04560	1.50000	1.04865	-ve	0.00737	0.00126
4	1.04865	1.50000	1.04954	-ve	0.00215	0.00037
5	1.04954	1.50000	1.04980	-ve	0.00063	0.00011
6	1.04980	1.50000	1.04988	-ve	0.00018	0.00003

The error analysis for the false-position method is not as easy as it is for the bisection method, however, if one of the end points becomes fixed, it can be shown that it is still a linear order of convergence, that is, it is the same rate as the bisection method, usually faster, but possibly slower. For differentiable functions, the closer the fixed end point is to the actual root, the faster the convergence.

Order of convergence of Regular Falsi Method

Let α be the exact root of $f(x) = 0$ and let x_n and x_{n+1} be two successive approximate solutions to the actual root α at step n.If ϵ_n and ϵ_{n+1} are the corresponding errors, thus we have:

$$x_n = \alpha + \epsilon_n \text{ and } x_{n+1} = \alpha + \epsilon_{n+1}.$$

The false position method is given by:

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

$$\implies \quad \alpha + \epsilon_{n+1} = \alpha + \epsilon_n - \frac{x_n - x_{n-1}}{f(\alpha + \epsilon_n) - f(\alpha + \epsilon_{n-1})} \times f(\alpha + \epsilon_n).$$

Expanding $f(\alpha+\epsilon_n)$ and $f(\alpha+\epsilon_{n-1})$ about α using Tylor's series gives

$$\implies \quad \epsilon_{n+1} = \epsilon_n - \frac{(x_n - x_{n-1})\left[f(\alpha)+\epsilon_n f'(\alpha)+\frac{\epsilon_n^2}{2}f''(\alpha)+\cdots\right]}{\left[f(\alpha)+\epsilon_n f'(\alpha)+\frac{\epsilon_n^2}{2}f''(\alpha)+\cdots\right]-\left[f(\alpha)-\epsilon_n f'(\alpha)+\frac{\epsilon_n^2}{2}f''(\alpha)-\cdots\right]}$$

$$\implies \quad \epsilon_{n+1} = \epsilon_n - \frac{(\epsilon_n - \epsilon_{n-1})\left[f(\alpha)+\epsilon_n f'(\alpha)+\frac{\epsilon_n^2}{2}f''(\alpha)\right]}{(\epsilon_n-\epsilon_{n-1})f'(\alpha)+\left(\frac{\epsilon_n^2}{2}-\frac{\epsilon_{n-1}^2}{2}\right)f''(\alpha)} \quad \text{(ignoring higher order terms)}$$

$$\implies \quad \epsilon_{n+1} = \epsilon_n - \frac{(\epsilon_n - \epsilon_{n-1})\left[f(\alpha)+\epsilon_n f'(\alpha)+\frac{\epsilon_n^2}{2}f''(\alpha)\right]}{(\epsilon_n-\epsilon_{n-1})\left[f'(\alpha)+\frac{\epsilon_n+\epsilon_{n-1}}{2}f''(\alpha)\right]}$$

$$\implies \quad \epsilon_{n+1} = \epsilon_n - \frac{f(\alpha)+\epsilon_n f'(\alpha)+\frac{\epsilon_n^2}{2}f''(\alpha)}{f'(\alpha)+\frac{\epsilon_n+\epsilon_{n-1}}{2}f''(\alpha)}$$

$$\implies \quad \epsilon_{n+1} = \epsilon_n - \frac{\epsilon_n f'(\alpha)+\frac{\epsilon_n^2}{2}f''(\alpha)}{f'(\alpha)+\frac{\epsilon_n+\epsilon_{n-1}}{2}f''(\alpha)} \qquad \text{Since } f(\alpha)=0$$

$$\implies \quad \epsilon_{n+1} = \epsilon_n - \frac{\epsilon_n+\frac{\epsilon_n^2}{2}\frac{f''(\alpha)}{f'(\alpha)}}{1+\frac{\epsilon_n+\epsilon_{n-1}}{2}\frac{f''(\alpha)}{f'(\alpha)}} \quad \text{Dividing numerator and denominator by } f'(\alpha)$$

$$\implies \quad \epsilon_{n+1} = \epsilon_n - \left[\epsilon_n+\frac{\epsilon_n^2}{2}\frac{f''(\alpha)}{f'(\alpha)}\right]\left[1+\frac{\epsilon_n+\epsilon_{n-1}}{2}\frac{f''(\alpha)}{f'(\alpha)}\right]^{-1}$$

Now, using the formula $(1+x)^{-1} = 1-x+x^2-x^3+\cdots$ and ignoring higher order powers of x we get

$$\epsilon_{n+1} = \epsilon_n - \left(\epsilon_n+\frac{\epsilon_n^2}{2}\frac{f''(\alpha)}{f'(\alpha)}\right)+\left(\epsilon_n+\frac{\epsilon_n^2}{2}\frac{f''(\alpha)}{f'(\alpha)}\right)\left(\frac{\epsilon_n+\epsilon_{n-1}}{2}\right)\frac{f''(\alpha)}{f'(\alpha)}$$

$$\implies \quad \epsilon_{n+1} = \epsilon_n - \epsilon_n - \frac{\epsilon_n^2}{2}\frac{f''(\alpha)}{f'(\alpha)}+\frac{\epsilon_n(\epsilon_n+\epsilon_{n-1})}{2}\frac{f''(\alpha)}{f'(\alpha)}+\frac{\epsilon_n^2}{2}\frac{f''(\alpha)}{f'(\alpha)}\frac{(\epsilon_n+\epsilon_{n-1})}{2}\frac{f''(\alpha)}{f'(\alpha)}$$

$$\implies \quad \epsilon_{n+1} = -\frac{\epsilon_n^2}{2}\frac{f''(\alpha)}{f'(\alpha)}+\frac{\epsilon_n^2}{2}\frac{f''(\alpha)}{f'(\alpha)}+\frac{\epsilon_n\epsilon_{n-1}}{2}\frac{f''(\alpha)}{f'(\alpha)}+\frac{\epsilon_n^2(\epsilon_n+\epsilon_{n-1})}{4}\left(\frac{f''(\alpha)}{f'(\alpha)}\right)^2$$

$$\implies \quad \epsilon_{n+1} = \frac{\epsilon_n\epsilon_{n-1}}{2}\frac{f''(\alpha)}{f'(\alpha)}+\frac{\epsilon_n^2(\epsilon_n+\epsilon_{n-1})}{4}\left(\frac{f''(\alpha)}{f'(\alpha)}\right)^2$$

$$\implies \quad \epsilon_{n+1} = \epsilon_n\epsilon_{n-1}\left(\frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}\right) \qquad \text{Ignoring higher powers of } \epsilon_n$$

Thus, we have

$$\epsilon_{n+1} = \epsilon_n\epsilon_{n-1}M \tag{2.10}$$

where $M = \left(\frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}\right)$ is constant.

In order to find the order of convergence, it is necessary to find a formula of type

$$\epsilon_{n+1} = c\epsilon_n^p \tag{2.11}$$

$$\begin{aligned}
\implies \quad & \epsilon_n = c\epsilon_{n-1}^p \\
\implies \quad & \epsilon_{n-1} = \left(\frac{\epsilon_n}{c}\right)^{\frac{1}{p}}
\end{aligned}$$

Substituting this value of ϵ_{n-1} into Equation (2.10), we get:

$$\begin{aligned}
\epsilon_{n+1} &= \epsilon_n \left(\frac{\epsilon_n}{c}\right)^{\frac{1}{p}} M \\
\implies \quad c\epsilon_n^p &= \epsilon_n \left(\frac{\epsilon_n}{c}\right)^{\frac{1}{p}} M \\
\implies \quad \epsilon_n^p &= \epsilon_n \left(\frac{\epsilon_n}{c}\right)^{\frac{1}{p}} M \times \frac{1}{c} \\
\implies \quad \epsilon_n^p &= Mc^{-\left(1+\frac{1}{p}\right)}\epsilon_n^{\left(1+\frac{1}{p}\right)}
\end{aligned}$$

Comparing the powers of ϵ_n on both sides we get

$$\begin{aligned}
& p = 1 + \frac{1}{p} \\
\implies \quad & p^2 - p - 1 = 0 \\
\implies \quad & p \approx 1.618 \quad \text{taking the positive root}
\end{aligned}$$

By putting this value of p into Equation (2.13) we get

$$\epsilon_{n+1} = c\epsilon_n^{1.618}$$

Therefore, the rate of convergence of the Regular Falsi Method is 1.618.

2.4 Fixed-point Iteration Method

In this section we give a more general introduction to **iteration methods**, presenting a general theory for **one-point iteration** formulae. This method is also known as substitution method or method of fixed iterations. To find the root of the equation $f(x) = 0$ by successive approximation, we rewrite the given equation in the form of $x = g(x)$ (in an infinite number of ways). A root α of $f(x) = 0$ is also a fixed point of the functiong(x), meaning that α is a number for which $\alpha = g(\alpha)$. So the iteration procedure

$$x_{n+1} = g(x_n) \tag{2.12}$$

converges to α under certain conditions.

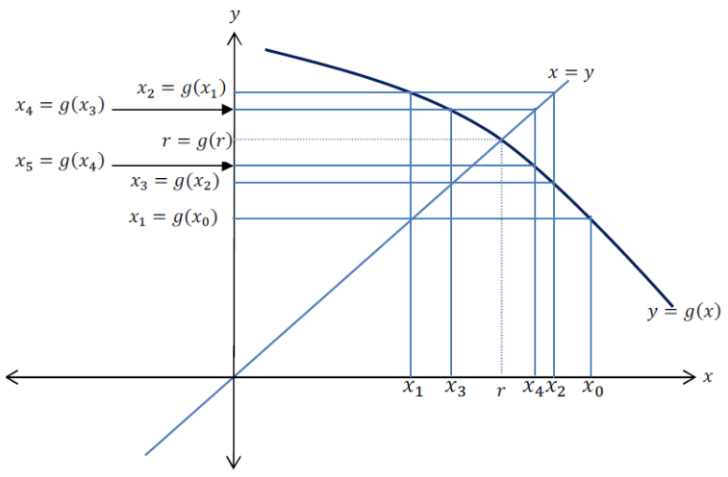


Figure 2.3: Root finding using Fixed-point iteration method

The formal procedure of the Fixed-point iteration Method is given below in Algorithm 3.

Algorithm 3: Fixed-point iteration Method

```
Input: Continuous function $g(x)$ ;
       Initial guess $x_0$ ;
       Error tolerance $\epsilon$. ;
Output: Approximate solution that is within $\epsilon$ of a root of $f(x)$ ;
$n = 0$    (Initialize the counter for iteration) ;
Step 2. while $|f(x_n)| \geq \epsilon$ do
    $x_{n+1} = g(x_n)$ ;
    if $g(x_{n+1}) == 0$ then
        return $x_{n+1}$ (we have found a solution);
    end
    $n = n + 1$;
end
Step 3. return $x_n$ (return the solution)
```

Example: Consider the function $f(x) = x^3 - 2$ and can be written as

$$x = x^3 + x - 2 \tag{2.13}$$

$$x = \frac{2 + 5x - x^3}{5} \tag{2.14}$$

Taking $x_0 = 1.2$ we have

n	Equation (2.13)	Equation (2.14)
1	0.928	1.2544
2	-0.273	1.2596
3	-2.293	1.2599
4	-16.349	1.25992

Thus Equation (2.14) gives the correct root, while Equation (2.13) does not converge.

We begin by asking whether the equation x = g(x) has a solution. For this to occur, the graphs of $y = x$ and $y = g(x)$ must intersect, as seen on the earlier figure 6.3. The following lemma gives conditions under which we are guaranteed there is a fixed point α which is the root of the function f.

Lemma: Let $g(x)$ be a continuous function on the interval $[a, b]$, and suppose it satisfies the property

$$a \leq x \leq b \implies a \leq g(x) \leq b \tag{2.15}$$

Then the equation $x = g(x)$ has at least one solution α in the interval $[a, b]$.

The next question is on what condition does Equation (2.12) will converge to the fixed point α. Suppose $g(x)$ and $g'(x)$ are continuous then from the Tylor series expansion about a point x_n

$$\mathrm{g}(x) = \mathrm{g}(x_n) + (x - x_n)\mathrm{g}'(x_n) + \frac{(x - x_n)^2}{2!}g''(x_n) + \cdots$$

by Taylor's Theorem we have

$$\mathrm{g}(\alpha) = \mathrm{g}(x_n) + (\alpha - x_n)\mathrm{g}'(\xi_n), \quad \text{where } \ x_n \leq \xi_n \leq \alpha. \tag{2.16}$$

Now, let x_0 be an initial guess to the fixed point, then from Equation (2.12) we have

$$\begin{aligned}
& x_1 = g(x_0) \\
\implies & \alpha - x_1 = \alpha - g(x_0) = g(\alpha) - g(x_0) \\
& \quad = (\alpha - x_0)g'(\xi_0), \quad x_0 \le \xi_0 \le \alpha, \texttt{ (using the above Equation } (2.16)) \\
& \alpha - x_2 = g'(\xi_1)(\alpha - x_1) \\
& \quad = g'(\xi_0)g'(\xi_1)(\alpha - x_0), \quad x_1 \le \xi_1 \le \alpha \\
& \quad \vdots \\
& \alpha - x_n = g'(\xi_0)g'(\xi_1)\cdots g'(\xi_{n-1})(\alpha - x_0)
\end{aligned} \tag{2.17}$$

So, if $|g'(\xi_n)| \le M$ for all n, then $|\epsilon_n| \le M^n|\epsilon_0|$ and convergence is assured if $M < 1$, i.e. if $|g'(x)| < 1$ in a neighbourhood containing both α and x_0 ; this condition dictates the version of the method which is to be used. Thus, condition for convergence is given as

Convergence condition The fixed-point iteration method based on the function g converge to α if

- g and g' are continuous functions on an interval I and
- $|g'(x)| < 1 \quad \forall x \in I$

Note: If $g'(x)| > 1$, then the iteration will not converge to α and when $g'(x)| = 1$ no conclusion can be made.

Example 2.4

Find the root of the equation $\cos x = 3x - 1$ correct to three decimal places using fixed-point iteration method.

Solution: Here we have $f(x) = \cos x - 3x + 1$ assume $x_0 = 0$ and $x_1 = \frac{\pi}{2}$ and $f(0) = 2 = +ve$ and $f(\frac{\pi}{2}) = -3(\frac{\pi}{2}) + 1 = -ve$. Thus, the root lies between 0 and $\frac{\pi}{2}$.

Now, the given equation can be rewrite as $x = \dfrac{1}{3}\cos x + 1 = g(x)$ (say) then we can check that $g'(x) = \frac{-\sin x}{3} = |g'(x)| < 1$ in $[0, \frac{\pi}{2}]$ hence, the Fixed-point iteration method can be applied.

Let $x_0 = 0$ be the initial guess then $x_1 = g(x_0) = 0.667$, $x_2 = g(x_1) = 0.5953, \cdots, x_5 = 0.6072, x_6 = 0.6071$. Therefore, since x_5 and x_6 are correct to three decimal places the required root is given by 0.6071.

n	x_{n-1}	x_n	$\lvert x_n - x_{n-1}\rvert$	$\lvert f(x_n)\rvert$
1	0.00000	0.66667	0.66667	0.21411
2	0.66667	0.59530	0.07137	0.04210
3	0.59530	0.60933	0.01403	0.00795
4	0.60933	0.60668	0.00265	0.00151
5	0.60668	0.60718	0.00050	0.00029
6	0.60718	0.60709	0.00010	0.00005

Rate Of Convergence Of Iteration Method

In general Fixed-point Iteration converges linearly with asymptotic error constant $|g'(\alpha)|$, since, by the definition of ξ_n and the continuity of g',

$$\lim_{n\to\infty} \frac{e_{n+1}}{e_n} = \lim_{n\to\infty} |g'(\xi_n)| = |g'(\alpha)|. \tag{2.18}$$

Recall that the conditions we have stated for linear convergence are nearly identical to the conditions for g to have a unique fixed point in $[a, b]$. The only difference is that now, we also require g' to be continuous on $[a, b]$.

Now, suppose that in addition to the previous conditions on g, we assume that $g'(\alpha) = 0$, and that g is twice continuously differentiable on $[a, b]$. Then, using Taylor's Theorem, we obtain

$$e_{n+1} = g(x_n) - g(\alpha) = g'(\alpha)(x_n - \alpha) + g''(\xi_n)(x_n - \alpha)^2 = g''(\xi_n)e_k^2,$$

where ξ_n lies between x_n and α. It follows that for any initial iterate $x_0 \in [a, b]$, Fixed-point iteration converges at least quadratically, with asymptotic error constant $|g''(\alpha)/2|$. This discussion implies the following general result

Theorem 2.2 General Convergence

Let $g(x)$ be a function that is n times continuously differentiable on an interval $[a, b]$. Furthermore, assume that $g(x) \in [f(a), f(b)]$ for $x \in [a, b]$, and that $|g'(x)| \leq M$ on (a, b) for some constant $M < 1$. If the unique fixed point α in $[a, b]$ satisfies

$$g'(\alpha) = g''(\alpha) = \cdots = g^{(n-1)}(\alpha) = 0, \tag{2.19}$$

Then for any $x_0 \in [a, b]$, Fixed-point Iteration converges to α of order n, with asymptotic error constant $|g(n)(\alpha)/n||$.

2.5 Newton-Raphson Method

It is also called Newton's method and it is the general root finding method. This method requires only one appropriate starting point x_0 as an initial assumption of the root of the function $f(x) = 0$. At $(x_0, f(x_0))$ a tangent to $f(x) = 0$ is drawn. Equation of this tangent is given by

$$y = f'(x_0)(x - x_0) + f(x_0) \tag{2.20}$$

The point of intersection, say , of this tangent with x-asis $(y = 0)$ is taken to be the next approximation to the root of f(x) = 0. So on substituting $y = 0$ in the tangent equation we get

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{2.21}$$

If $|f(x_1)| < \epsilon$ we have got an acceptable approximate root of $f(x) = 0$, otherwise we replace x_0 by x_1, and draw a tangent to $f(x) = 0$ at $(x_1, f(x_1))$ and consider its intersection, say , with x-axis as an improved approximation to the root of f(x)=0. If $|f(x_2)| > \epsilon$, we iterate the above process till the convergence criteria is satisfied. This geometrical description of the method may be clearly visualized in the figure below:

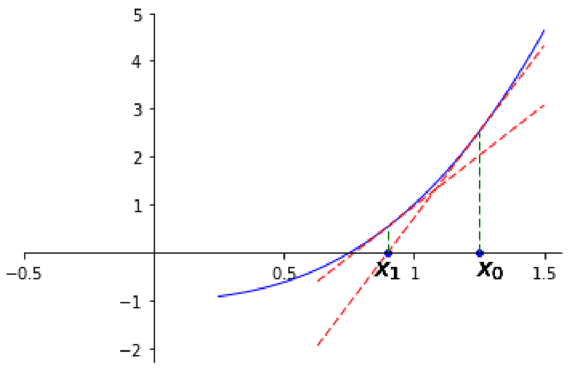


Figure 2.4: Geometric representation of Newton's Method

The various steps involved in calculating the root of $f(x) = 0$ by Newton Raphson Method are described compactly in the algorithm below.

Algorithm 4: Newton-Raphson Method

```
Input: Continuously differentiable function $f(x)$ ;
       Initial guess $x_0$ ;
       Error tolerance $\epsilon$. ;
Output: Approximate solution that is within $\epsilon$ of a root of $f(x)$ ;
Step 1. $n = 0$ initialize the counter ;
Step 2. while $|f(x_n)| \geq \epsilon$ do
    $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ ;
    if $f(x_{n+1}) == 0$ then
        return $x_{n+1}$ (we have found a solution);
    end
    $n = n + 1$
end
Step 3. return $x_n$ (return the solution)
```

Remark (1): This method converges faster than the earlier methods. In fact the method converges at a quadratic rate. We will prove this later.

Remark (2): This method can be also derived directly by the Taylor expansion f(x) in the neighbourhood of the root α of $f(x) = 0$. The starting approximation x_0 to α is to be properly chosen so that the first order Taylor series approximation of $f(x_0 + h)$ in the neighbourhood of x_0 leads to , an improved approximation to α. i.e

$$f(x_0 + h) = f(x_0) + hf^{'}(x_0) + \frac{h^2}{2} f''(x_0) + ... = 0$$

$\because \quad h \;<<\; 1$, neglecting h^2 and its higher powers, we get

$$f(x_0) + hf'(x_0) = 0$$

i.e.

$$h = -\frac{f(x_0)}{f'(x_0)}, \qquad \because h = x_1 - x_0$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Now the successive approximations etc may be calculated by the iterative formula:

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

Example 2.5

Find the root of the equation $\cos x = 3x - 1$ correct to three decimal places using the Newton-Raphson's Method.

Solution: Given

$$f(x) = \cos x - 3x + 1$$

and hence

$$f'(x) = -\sin x - 3$$

assume $x_0 = \frac{\pi}{4} = 0.7854$

Iteration 1:

$$\therefore f(x_0) = -0.6491; \qquad f'(x_0) = -3.7071$$

$$\therefore x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = \frac{\pi}{4} - \frac{-0.6491}{-3.7071} = 0.6103$$

Since $|f(x_1)| = 0.0114 > \frac{1}{2}10^{-3}$ or since none of the digits in x_0 and x_1 are correct, we repeat the Newton-Raphson procedure

Iteration 2:

$$f(x_1) = -0.0114; \quad f'(x_1) = -3.5731$$

We have

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 0.6103 - \frac{-0.0114}{-3.5731} = 0.6071028260$$

Now, we can see that only one correct decimal place between x_1 and x_2 and we need procedure the iteration.

Iteration 3:

$$f(x_2) = -4.20566891868e - 06; \quad f'(x_2) = -3.57049039628$$

We have

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 0.6071 - \frac{-0.0114}{-3.5731} = 0.6071016481$$

n	x_{n-1}	x_n	$\|x_n - x_{n-1}\|$	$\|f(x_n)\|$
1	0.7853981634	0.6103053626	0.1750928008	0.0114430406
2	0.6103053626	0.6071028260	0.0032025367	0.0000042055
3	0.6071028260	0.6071016481	0.0000011778	0.0000000000

Convergence Analysis for Netwon's Method

Using the same approach as with Fixed-point Iteration, we can determine the convergence rate of Newton's Method applied to the equation $f(x) = 0$, where we assume that f is continuously differentiable near the exact solution α , and that f'' exists near α. Using Taylor's Theorem, we obtain

$$\begin{aligned}
e_{n+1} &= x_{n+1} - \alpha \\
&= x_n - \frac{f(x_n)}{f'(x_n)} - \alpha \\
&= e_n - \frac{f(x_n)}{f'(x_n)} \\
&= e_n - \frac{1}{f'(x_n)}\left[f(\alpha) - f'(x_n)(\alpha - x_n) - \frac{1}{2}f''(\xi_n)(\alpha - x_n)^2\right] \\
&= e_n + \frac{1}{f'(x_n)}\left[f'(x_n)(\alpha - x_n) + \frac{1}{2}f''(\xi_n)(\alpha - x_n)^2\right] \text{(MVT)} \\
&= e_n + \frac{1}{f'(x_n)}\left[-f'(x_n)e_n + \frac{1}{2}f''(\xi_n)e_n^2\right]
\end{aligned}$$

Thus, we have

$$e_{n+1} = \frac{f''(\xi_n)}{2f'(x_n)}e_n^2 \tag{2.22}$$

where ξ_n is between x_n and α . We conclude that if $f'(\alpha) \neq 0$, then Newton's Method converges quadratically, with asymptotic error constant $|\frac{f''(\alpha)}{2f'(\alpha)}|$. It is easy to see from this constant, however, that if $f'(\alpha)$ is very small, or zero, then convergence can be very slow or may not even occur.

Example 2.6

Solve $2x^3 - 2.5x - 5 = 0$ for the root in [1,2] by Newton Raphson method with a tolerance $\epsilon = 10^{-6}$.

Solution: Given

$$f(x) = 2x^3 - 2.5x - 5 = 0 \texttt{ and } f'(x) = 6x^2 - 2.5 \qquad \texttt{with } \epsilon = 10^{-6}$$

Iteration 1: Take $x_0 = 2$ as an initial guess

$$\therefore \qquad f(x_0) = 6 \qquad ; \qquad f'(x_0) = 21.5$$

$$\therefore \qquad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 2 - \frac{6.0}{21.5} = 1.72093023$$

Since, $|f(x_1)| = 0.8910913504 > 10^{-6}$ we repeat the process.
Results are tabulated below:

n	x_{n-1}	x_n	$\lvert x_n - x_{n-1}\rvert$	$\lvert f(x_n)\rvert$
0	2.0000000000	1.7209302326	0.2790697674	0.8910913504
1	1.7209302326	1.6625730361	0.0583571965	0.0347669334
2	1.6625730361	1.6601046517	0.0024683844	0.0000607495
3	1.6601046517	1.6601003235	0.0000043282	0.0000000002
4	1.6601003235	1.6601003235	0.0000000000	0.0000000000

Examining the numbers in the table above, we can see that the number of correct decimal places approximately doubles with each iteration, which is typical of quadratic convergence.

Advantages and disadvantages of Newton's Method

- The error decreases rapidly with each iteration
- Newton's method is very fast. (Compare with bisection method!)
- Unfortunately, for bad choices of x_0 (the initial guess) the method can fail to converge! Therefore the choice of x 0 is VERY IMPORTANT!
- Each iteration of Newton's method requires two function evaluations, while the bisection method requires only one.

Note: A good strategy for avoiding failure to converge would be to use the bisection method for a few steps (to give an initial estimate and make sure the sequence of guesses is going in the right direction) followed by Newton's method, which should converge very fast at this point.

2.6 Secant Method

It is the most important variant of Netwon-Raphson method. The idea behind the Secant Method is as follows. Assume we need to find a root of the equation $f(x) = 0$, called α. Consider the graph of the function $f(x)$ and two initial estimates of the root, x_0 and x_1. Unlike the Bisection and Regular-falsi method, the two initial guesses do not need to bracket the root of the equation. Thus, The secant method is an open method but a two-point iteration method and may or may not converge. However, when secant method converges, it will typically converge faster than the Bisection method. However, since the derivative is approximated as given by Equation (2.23), it converges slower than the Newton-Raphson method.

The two points $(x_0, f(x_0))$ and $(x_1, f(x_1))$ on the graph of $f(x)$ determine a straight line, called a **secant line** which can be viewed as an approximation to the graph.

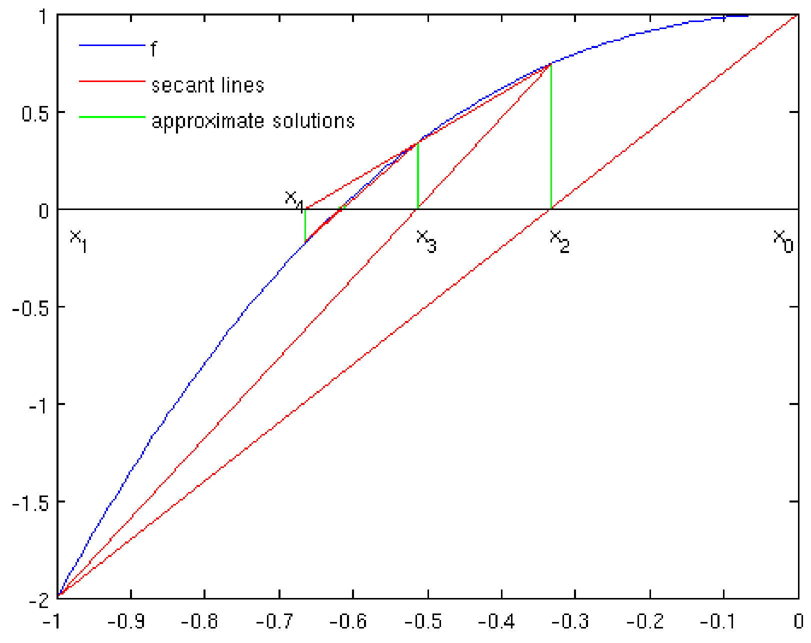


Figure 2.5: Root finding using Fixed-point iteration method

The straight line passing through the two points $(x_0 f(x_0))$ and $(x_1, f(x_1))$can be expressed as

$$\frac{y - f(x_1)}{x - x_1} = \frac{f(x_0) - f(x_1)}{x_0 - x_1}$$

We let $y = 0$ and solve the equation for x, now renamed as x_2, as an approximation of the root

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

The point x_2 where this secant line crosses the $x-$axis is then the next approximation for the root α.

The general procedure for Secant Method is given bellow in Algorithm 5

Algorithm 5: Secant Method

```
Input: Continuous function $f(x)$ ;
       Initial guesses $x_0$ and $x_1$ ;
       Error tolerance $\epsilon$. ;
Output: Approximate solution that is within $\epsilon$ of a root of $f(x)$;
Step 1. $n = 1$    (Initialize the counter);
Step 2. $x_{n-1} = x_0; x_n = x_1$ ;
Step 3. while $|f(x_n)| \geq \epsilon$ do
    $x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$ ;
    if $f(x_{n+1}) == 0$ then
        return $x_{n+1}$ (we have found a solution);
    end
    $n = n + 1$
end
Step 4. return $x_n$ (return the solution)
```

Alternatively, we can think the secant method as in Newton's method, but instead of using $f'(x_n)$, we approximate this derivative by a finite difference or the secant, i.e., the slope of the straight line that goes through the two most recent approximations x_n and x_{n-1}. This slope is given by

$$f'(x_{n-1}) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}\,. \tag{2.23}$$

Inserting this expression for $f'(x_n)$ in Newton's method simply gives us the secant method:

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

or

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \tag{2.24}$$

Comparing Equation (2.24) to the graph in Figure 2.5, we see how two chosen starting points x_0, x_1, and corresponding function values are used to compute x_2. Once we have x_2, we similarly use x_1 and x_2 to compute x_3. As with Newton's method, the procedure is repeated until $|f(x_n)|$ or $|x_{n+1} - x_n|$is below some chosen error tolerance (ϵ).

Example 2.7

Solve $2x^3 - 2.5x - 5 = 0$ for the root with $a = 1, b = 2$ by secant method to an accuracy of 10^{-6}.

Solution:

Iteration 1: Set $x_0 = a = 1 \quad ; \quad x_1 = b = 2$, and we have

$$f(x_0) = f(1) = -5.5 \text{ and } f(x_1) = f(2) = 6.0$$

Therefore,

$$\begin{aligned} x_2 &= \frac{f(x_0)x_{-1} - f(x_{-1})x_0}{f(x_0) - f(x_{-1})} \\ &= \frac{f(2).1 - f(1).2}{f(2) - f(1)}) \\ &= \frac{6.1 - (-5.5).2}{6 - (-5.5)} \\ &= 1.4782608747 \end{aligned}$$

Since, $|f(x_1)| = |-2.2348976135| > 10^{-6}$ we repeat the process with $(x_1, f(x_1))$, $(x_2, f(x_2))$ and so on till you get a $\xi = x_n$ s.t. $|f(\xi)| < \epsilon = 10^{-6}$. These results are tabulated below:

n	x_{n-1}	x_n	x_{n+1}	$\lvert x_n - x_{n-1}\rvert$	$\lvert f(x_n)\rvert$
1	1.0000000000	2.0000000000	1.4782608696	0.5217391304	2.2348976740
2	2.0000000000	1.4782608696	1.6198574765	0.1415966069	0.5488317259
3	1.4782608696	1.6198574765	1.6659486215	0.0460911450	0.0824254417
4	1.6198574765	1.6659486215	1.6599303406	0.0060182810	0.0023855241
5	1.6659486215	1.6599303406	1.6600996200	0.0001692795	0.0000098734
6	1.6599303406	1.6600996200	1.6601003236	0.0000007035	0.0000000012

Convergence Anlysis for Secant Method

We now consider the order of convergence of the secant method. Let α be the true root of the equation $f(\alpha) = 0$, and the error of x_{n+1} is:

$$\begin{aligned}
e_{n+1} &= x_{n+1} - \alpha \\
&= x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) - \alpha \\
&= \frac{(x_{n-1} - \alpha) f(x_n) - (x_n - \alpha) f(x_{n-1})}{f(x_n) - f(x_{n-1})} \\
&= \frac{e_{n-1} f(x_n) - e_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}
\end{aligned}$$

The Taylor expansion of $f(x_n)$

$$f(x_n) = f(\alpha + e_n) = f(\alpha) + f'(\alpha) e_n + \frac{1}{2} f''(\alpha) e_n^2 + O(e_n^3) = f'(\alpha) e_n + \frac{1}{2} f''(\alpha) e_n^2 + O(e_n^3)$$

(as α is the solution for $f(\alpha) = 0$) and similarly

$$f(x_{n-1}) = f'(\alpha) e_{n-1} + \frac{1}{2} f''(\alpha) e_{n-1}^2 + O(e_{n-1}^3)$$

Substituting these into the expression for e_{n+1} we get

$$\begin{aligned}
e_{n+1} &= \frac{e_{n-1}\left[f'(\alpha) e_n + \frac{1}{2} f''(\alpha) e_n^2 + O(e_n^3)\right] - e_n\left[f'(\alpha) e_{n-1} + \frac{1}{2} f''(\alpha) e_{n-1}^2 + O(e_{n-1}^3)\right]}{\left[f'(\alpha) e_n + \frac{1}{2} f''(x) e_n^2 + O(e_n^3)\right] - \left[f'(\alpha) e_{n-1} + \frac{1}{2} f''(\alpha) e_{n-1}^2 + O(e_{n-1}^3)\right]} \\
&= \frac{e_{n-1} e_n f''(\alpha)(e_n - e_{n-1})/2 + O(e_{n-1}^4)}{(e_n - e_{n-1}) f'(\alpha) + (e_n^2 - e_{n+1}^2) f''(\alpha)/2 + O(e_{n-1}^3)} \\
&= \frac{e_{n-1} e_n f''(\alpha)/2 + O(e_{n-1}^3)}{f'(\alpha) + (e_n + e_{n+1}) f''(\alpha)/2 + O(e_{n-1}^2)} \\
&= \frac{e_{n-1} e_n f''(\alpha)/2 + O(e_{n-1}^3)}{f'(\alpha) + O(e_n) + O(e_{n-1}^2)}
\end{aligned} \tag{2.25}$$

When $n \to \infty$, all error terms in both the numerator and denominator of order higher than the lowest order term approach to zero, we have

$$e_{n+1} = \frac{e_{n-1}e_n f''(\alpha)}{2f'(x)} = e_n e_{n-1} C$$

where we have defined $C = f''(\alpha)/2f'(\alpha)$. To find the order of convergence, we need to find p in

$$|e_{n+1}| \leq |e_{n-1}|\ |e_n|\ |C| = \mu |e_n|^p$$

Solving the equation above for $|e_n|$ we get

$$|e_n| = \left(\frac{|C|}{\mu}|e_{n-1}|\right)^{1/(p-1)} = \left(\frac{|C|}{\mu}\right)^{1/(p-1)} |e_{n-1}|^{1/(p-1)}$$

However, when $n \to \infty$ we also have

$$|e_n| = \mu |e_{n-1}|^p$$

Comparing the two equations above we get

$$p = \frac{1}{p-1}, \qquad \mu = \left(\frac{|C|}{\mu}\right)^{1/(p-1)} = \left(\frac{|C|}{\mu}\right)^p$$

Solving these two equations we get

$$p = \frac{1+\sqrt{5}}{2} = 1.618, \qquad \mu = |C|^{p/(p+1)} = C^{(\sqrt{5}-1)/2} = |C|^{0.618}$$

i.e.,

$$|e_{n+1}| = \left|\frac{f''(x)}{2f'(x)}\right|^{0.618} |e_n|^{1.618}$$

We see that the order of convergence $p = 1.618$ of the secant method is better than linear but not worse than quadratic convergence.

Advantages and disadvantages of Secant Method

- The error decreases slowly at first but then rapidly after a few iterations.
- The secant method is slower than Newton's method but faster than the bisection method.
- Each iteration of Newton's method requires two function evaluations, while the secant method requires only one
- The secant method does not require differentiation

Chapter 3

System of Equations

3.1 Introduction

In this chapter we will learn how to solve system of Equations of the form

$$\begin{aligned} f_1(x_1, x_2, \cdots, x_n) &= 0 \\ f_2(x_1, x_2, \cdots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \cdots, x_n) &= 0, \end{aligned} \tag{3.1}$$

where $f_i \in \mathbb{R}^n$ are functions of n-variables of x_i and $b_i \in \mathbb{R}$ are real numbers.

Definition 3.1 Linear and nonlinear functions:

A function $f : \mathbb{R} \to \mathbb{R}$ is defined as being *nonlinear* when it does not satisfy the superposition principle that is

$$f(x_1 + x_2 + \cdots + x_n) \neq f(x_1) + f(x_2) + \cdots + f(x_n).$$

On the other hand, if f satisfies the superposition principle the f is a *linear* function and hence we can rewrite the equation $f(x_1, x_2, \cdots, x_n) = 0$ as

$$a_1 x_1 + a_2 x_2 + \cdots a_n x_n = b.$$

If all the functions f_i in Equation (7.1) are linear equation, then Equation (7.1) called *System of Linear Equations* (SLEs), otherwise if one of the f_i's is nonlinear then it is called *System of Nonlinear Equations* (SNLEs). In the next section subsequent sections of this chapter we will discuss methods for solving SLEs and at the end we will give one particular method,Newton's Method, for solving SNLEs.

In general, we can divide the approaches to the solution of linear algebraic equations into two broad areas. The first of these involve algorithms that lead directly to a solution of the problem after a finite number of steps while the second class involves an initial "guess" which then is improved by a succession of finite steps, each set of which we will call an iteration. If the process is applicable and properly formulated, a finite number of iterations will lead to a solution.

3.2 Direct Methods for System of Linear Equations (SLE)

There are several methods which directly solve system of linear equations, among these are such as Cramer's rule, Gaussian elimination, Gauss Jordan, QR factorization. However, since the Cramer's rule is unsuitable for computer implementation and is not discussed here. Among the direct methods, we only present the Gaussian elimination and Gauss Jordan here in detail.

In general, we may write a system of linear algebraic equations in the form

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

where $x_1, x_2, \cdots, x_n$ are the **unknowns**, $a_{11}, a_{12}, \cdots, a_{nn}$ are the **coefficients** of the system, and $b_1, b_2, \cdots, b_n$ the **constant terms**. The above system of linear equations (7.2) can be written in **matrix form** as

$$\boldsymbol{Ax} = \boldsymbol{b},$$

where the coefficient matrix $\boldsymbol{A}$, the unknown column vector $\boldsymbol{x}$ and the constant column vector $\boldsymbol{b}$ are given, respectively, as

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{22} & \cdots & a_{2n} \end{pmatrix} \qquad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

3.2.1 Gaussian Method

The Gaussian elimination is arguably the most used method for solving a set of linear algebraic equations. It makes use of the fact that a solution of a special system of linear equations, namely the systems involving triangular matrices, can be constructed very easily.

Forward and back substitution

A matrix equation $\boldsymbol{Lx} = \boldsymbol{b}$ or $\boldsymbol{Ux} = \boldsymbol{b}$ is very easy to solve by an iterative processes called **forward substitution** and **back substitution**, respectively.

Consider the system of equations of the form $\boldsymbol{Lx} = \boldsymbol{b}$,

$$\begin{pmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{12} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & u_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

where, $\boldsymbol{L}$ is an **lower triangular** matrix,i.e., $l_{ij} = 0, \forall i > j$ and $l_{ii} \neq 0 \forall$. We can solve this system by using the forward substitution as follows. The process is so called because, we first computes x_1 from the first equation and then substitutes that forward into the next equation to solve for x_2 , and repeats through to x_n.

Observe that the first equation $l_{11}x_1 = b_1$ only involves x_1, and hence we can compute x_1 directly. The second equation only involves x_1 and x_2, thus we can substitutes the computed value of x_1 to this equation and solver for x_2. Continuing in this way, the j^{th} equation only involves $x_1, x_2, \cdots, x_j$, and one can solve for x_j by substituting the previously solved values $x_1, x_2, \cdots, x_{j-1}$. Thus, we can write this procedure in a formal way as bellow

Forward Substitution (FS):

$$\begin{aligned} x_1 &= \frac{b_1}{l_{11}} \\ x_2 &= \frac{b_2 - l_{21}x_1}{l_{22}} \\ &\vdots \\ x_i &= \frac{1}{l_{ii}}\left[b_i - \sum_{j=1}^{i-1} l_{ij}x_j\right] \text{ for } i = 3, 4, \cdots n. \end{aligned} \tag{3.3}$$

On the other hand, if we consider a system of linear equation given by the following matrix equation

$$\boldsymbol{Ux} = \boldsymbol{\beta} \tag{3.4}$$

Or

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_23 & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

where $\boldsymbol{U}$ is an **upper triangular** matrix such that all elements below the main diagonal are zero and all the diagonal elements are non-zero, i.e., $u_{ii} \neq$ for all i. In an upper triangular matrix, similarly, we can work backwards, first computing x_n, then substituting x_n back into the previous equation to solve for x_{n-1} , and repeating through x_1.

To solve the system (3.4) one can start from the last equation

$$x_n = \frac{\beta_n}{u_{nn}},$$

and then proceed as follows

$$\begin{aligned} x_{n-1} &= \frac{1}{u_{n-1,n-1}}\left[b_{n-1,n-1} - u_{n-1,n}x_n\right], \\ x_{n-2} &= \frac{1}{u_{n-2,n-2}}\left[b_{n-2,n-2} - u_{n-2,n-1}x_{n-1} - u_{n-2,n}x_n\right], \end{aligned}$$

In general, for i^{th} element x_i; we can write

$$x_i = \frac{1}{u_{ii}}\left[b_i - \sum_{j=i+1}^{n} u_{ij}x_j\right]$$

where $i = n-1, n-2; \cdots, 1$. Since we proceed from $i = n$ to $i = 1$, these set of calculations are referred to as **back substitution**. Thus, we have

Backward Substitution(BF):

$$\begin{aligned} x_n &= \frac{b_n}{u_{nn}} \\ x_i &= \frac{1}{u_{ii}}\left[b_i - \sum_{j=i+1}^{n} u_{ij}x_j\right] \end{aligned} \tag{3.5}$$

Thus, the solution procedure for solving this system of equations involving a special type of upper triangular matrix is particularly simple. However, the trouble is that most of the problems encountered in real applications do not have such special form.

Now, suppose we want to solve a system of equations $\boldsymbol{Ax} = \boldsymbol{b}$ where $\boldsymbol{A}$ is a general full rank square matrix without any special form. Then, by some series of transformations, can we convert the system from $\boldsymbol{Ax} = \boldsymbol{b}$ form to $\boldsymbol{Ux} = \boldsymbol{\beta}$ form, i.e. to the desired special form? It is important to carry out this transformation such that the original system of equations and the transformed system of equations have identical solutions, $\boldsymbol{x}$.

The following **row operations** on the augmented matrix of a system produce the augmented matrix of an equivalent system, i.e., a system with the same solution as the original one.

- Interchange any two rows.
- Multiply each element of a row by a nonzero constant.
- Replace a row by the sum of itself and a constant multiple of another row of the matrix.

For these row operations, we will use the following notations.

- $R_i \iff R_j$ means: Interchange row i and row j.

- αR_i means: Replace row i with α times row i.
- $R_i + \alpha R_j$ means: Replace row i with the sum of row i and α times row j.

Now, we shall assume that the system has a unique solution, i.e., we assume $a_{ii} \neq 0$ and proceed to describe the simple **Gaussian Elimination method(GEM)** for finding the solution. The method reduces the system to an upper triangular system using elementary row operations (ERO).

Gaussian Elimination method to solve a system of linear equations is described in the following steps.

Step -1: write the augmented matrix of the system as follows:

$$\boldsymbol{A}^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & | & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & | & b_2^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} & | & b_3^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & | & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} & | & b_n^{(1)} \end{pmatrix}, \quad \text{where } a_{ij}^{(1)} = a_{ij} \text{ and } b_j^{(1)} = b_j.$$

Step -2: Perform elementary row operations to get zeros below the diagonal.

2 -1: Assuming the **pivot element** ,$a_{11}^{(1)} \neq 0$, is nonzero we apply ERO on $\boldsymbol{A}^{(1)}$ to reduce all entries below $a_{11}^{(1)}$ to zero. Let the resulting matrix be denoted by $A^{(2)}$.

$$\boldsymbol{A}^{(1)} \xrightarrow{R_i + m_{i1}^{(1)} R_1} \boldsymbol{A}^{(2)}, \quad \text{where } m_{i1}^{(1)} = -\frac{a_{i1}^{(1)}}{a_{11}^{(1)}}.$$

Here, the m_i's are called multipliers for each row i and also note that $\boldsymbol{A}^{(2)}$ is of the form

$$\boldsymbol{A}^{(2)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & | & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & | & b_2^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & | & b_3^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & | & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} & | & b_n^{(2)} \end{pmatrix},$$

In addition, the above row operations on $\boldsymbol{A}^{(1)}$ can be effected by pre-multiplying

$\boldsymbol{A}^{(1)}$ by $\boldsymbol{M}^{(1)}$ where

$$\boldsymbol{M}^{(1)} = \left(\begin{array}{c|cccc} 1 & 0 & 0 & 0 & 0 \\ \hline m_{21}^{(1)} & & & & \\ m_{31}^{(1)} & & \boldsymbol{I}_{n-1} & & \\ \vdots & & & & \\ m_{n1}^{(1)} & & & & \end{array}\right),$$

where $\boldsymbol{I}_{n-1}$ is an $n-1 \times n-1$ identity matrix. Thus, we have

$$\boldsymbol{M}^{(1)}\boldsymbol{A}^{(1)} = \boldsymbol{A}^{(2)}$$

Then the system $\boldsymbol{Ax} = b$ is equivalent to $\boldsymbol{A}^{(2)}\boldsymbol{x} = \boldsymbol{b}^{(2)}$

2-2: Assume the pivot element $a_{22}^{(2)} \neq 0$ and reduce all entries bellow the $a_{22}^{(2)}$ to zero by ERO

$$\boldsymbol{A}^{(2)} \xrightarrow{R_i + m_{i2}^{(2)} R_1} \boldsymbol{A}^{(3)}, \quad \text{where } m_{i2}^{(2)} = -\frac{a_{i2}^{(2)}}{a_{22}^{(2)}}, \text{ for } i > 2.$$

and $\boldsymbol{M}^{(2)}$ is given by

$$\boldsymbol{M}^{(2)} = \left(\begin{array}{cc|cccc} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \hline 0 & m_{32}^{(2)} & & & & \\ 0 & m_{42}^{(2)} & & \boldsymbol{I}_{n-2} & & \\ \vdots & \vdots & & & & \\ 0 & m_{n2}^{(2)} & & & & \end{array}\right),$$

and hence $\boldsymbol{M}^{(2)}\boldsymbol{A}^{(2)} = \boldsymbol{A}^{(3)}$, where $\boldsymbol{A}^{(3)}$ is of the form

$$\boldsymbol{A}^{(3)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & | & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & | & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & | & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & | & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} & | & b_n^{(3)} \end{pmatrix},$$

Repeat these procedures until you get $\boldsymbol{A}^{(n)}$. These procedures are generalized as follow at the k^{th} step.

2 -k: Assume $a^{(k)_{kk}} \neq 0$, then reduce all entries below $a^{(k)_{kk}}$ to zero by applying

ERO,i.e.,

$$\boldsymbol{A}^{(k)} \xrightarrow{R_i + m_{ik}^{(k)} R_k} \boldsymbol{A}^{(k+1)}, \quad \text{where } m_{ik}^{(k)} = -\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \text{ for } i > k.$$

and $\boldsymbol{M}^{(k)}$ is of the form

$$\boldsymbol{M}^{(k)} = \left(\begin{array}{cccc|cccc} & & & & 0 & 0 & \cdots & 0 \\ & \boldsymbol{I}_k & & & 0 & 0 & \cdots & 0 \\ & & & & \vdots & \vdots & \ddots & \vdots \\ & & & & 0 & 0 & \cdots & 0 \\ \hline 0 & 0 & \cdots & m_{(k+1\ k)}^{(k)} & & & & \\ 0 & 0 & \cdots & m_{(k+2\ k)}^{(k)} & & \boldsymbol{I}_{n-k} & & \\ \vdots & \vdots & \ddots & \vdots & & & & \\ 0 & 0 & \cdots & m_{(n\ k)}^{(k)} & & & & \end{array}\right)$$

and hence $\boldsymbol{M}^{(k)}\boldsymbol{A}^{(k)} = \boldsymbol{A}^{(k+1)}$, where $\boldsymbol{A}^{(k+1)}$ is of the form

$$\boldsymbol{A}^{(k)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} & | & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} & | & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3k}^{(3)} & \cdots & a_{3n}^{(3)} & | & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & & | & \vdots & \\ 0 & 0 & 0 & \cdots & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} & | & b_n^{(k)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & & | & \vdots & \\ 0 & 0 & 0 & \cdots & 0 & \cdots & a_{nn}^{(k)} & | & b_n^{(k)} \end{pmatrix},$$

Continue this procedure until the $(n-1)^{th}$-step and is given bellow.

2 -(n-1): Assuming $a_{n-1,n-1}^{(n-1)} \neq 0$, reduce all entries below $a_{n-1,n-1}^{(n-1)}$ to zero by applying

$$\boldsymbol{A}^{(n-1)} \xrightarrow{R_n + m_{n,n-1}^{(n-1)} R_{n-1}} \boldsymbol{A}^{(n)}, \quad \text{where } m_{n,n-1}^{(n-1)} = -\frac{a_{n,n-1}^{(n-1)}}{a_{n-1,n-1}^{(n-1)}},$$

and $\boldsymbol{M}^{(n-1)}$ is of the form

$$\boldsymbol{M}^{(n-1)} = \left(\begin{array}{cccc|c} & \boldsymbol{I}_{(n-1)} & & & \boldsymbol{0} \\ \hline 0 & 0 & \cdots & m_{(n,n-1)}^{(n-1)} & 1 \end{array}\right)$$

and hence $\boldsymbol{M}^{(n-1)}\boldsymbol{A}^{(n-1)} = \boldsymbol{A}^{(n)}$, where $\boldsymbol{A}^{(n)}$ is of the form

$$\boldsymbol{A}^{(n)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} & | & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} & | & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3k}^{(3)} & \cdots & a_{3n}^{(3)} & | & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & & | & \vdots & \\ 0 & 0 & 0 & \cdots & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} & | & b_n^{(k)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & & | & \vdots & \\ 0 & 0 & 0 & \cdots & 0 & \cdots & a_{nn}^{(n)} & | & b_n^{(n)} \end{pmatrix},$$

At the end of the $(n-1)^{th}$-step, in general, we have

$$\boldsymbol{M}^{(n-1)}\boldsymbol{M}^{(n-2)}\cdots\boldsymbol{M}^{(1)}\boldsymbol{A}^{(1)} = \boldsymbol{A}^{(n)}, \tag{3.6}$$

where $\boldsymbol{A}^{(1)}$ is an augmented matrix of the original matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$.

Step -3: Inspect the resulting matrix and re-interpret it as a system of equations.

a If you get a zero diagonal element then the system can not be solved using GEM or **doe not have a solution**. In this case the matrix is probably a singular matrix.

b If you get less equations than unknowns after the reduction and if there is a solution then there is an **infinite number of solutions**.

c If you get as many equations as unknowns after the reduction and if there is a solution then there is **exactly one solution**.

Step -4: Apply the BSF to get the solution of the given system if part (c) in **step -3** is true.

Note, as a "by-product" of GEM, the simple GEM in addition to to solve the system $\boldsymbol{Ax} = \boldsymbol{b}$ it can also be used to evaluate $\det \boldsymbol{A}$ provided $a_{kk}^{(k)} \neq 0$ for each k. Note further that each $\boldsymbol{M}^{(k)}$ is a lower triangular matrix with all diagonal entries as 1. Thus let $\det \boldsymbol{M}^{(k)}$ is 1 for every k. Now, from Equation (3.6) taking the un-augmented matrix from $\boldsymbol{A}'^{(n)}$ $\boldsymbol{A}'^{(1)}$ and we have

$$\det \boldsymbol{A}'^{(n)} = \det \boldsymbol{M}^{(n-1)} \det \boldsymbol{M}^{(n-2)} \cdots \det \boldsymbol{M}^{(1)} \det \boldsymbol{A}'^{(1)},$$
$$\det \boldsymbol{A}'^{(n)} = \det \boldsymbol{A}'^{(1)} = \det \boldsymbol{A}, \qquad \text{since } \boldsymbol{A} = \boldsymbol{A}'^{(1)}$$

Now $\boldsymbol{A}'^{(n)}$ is an upper triangular matrix and hence its determinant is the product of the diagonal elements

$$a_{11}^{(1)}a_{22}^{(2)}\cdots a_{nn}^{(n)}$$

. Thus det $\boldsymbol{A}$ is given by

$$\det \boldsymbol{A} = a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)}$$

In addition, note that all the matrix $\boldsymbol{M}^{(k)}$ are lower triangular, and nonsingular as their det $= 1 \neq 0$ for all k. They are all therefore invertible and their inverses are all lower triangular, i.e., if

$$\boldsymbol{L} = \boldsymbol{M}^{(n-1)} \boldsymbol{M}^{(n-2)} \cdots \boldsymbol{M}^{(1)}$$

then $\boldsymbol{N}$ is lower triangular, and nonsingular and $\boldsymbol{N}^{-1}$ is also lower triangular. Now

$$\boldsymbol{NA} = \boldsymbol{NA}'^{(1)} = \boldsymbol{M}^{(n-1)} \boldsymbol{M}^{(n-2)} \cdots \boldsymbol{M}^{(1)} \boldsymbol{A}'^{(1)} = \boldsymbol{A}'^{(n)}.$$

Therefore

$$\boldsymbol{A} = \boldsymbol{N}^{-1} \boldsymbol{A}'^{(n)}.$$

Now $\boldsymbol{N}^{-1}$ is lower triangular which we denote by $\boldsymbol{L}$ and $\boldsymbol{A}'^{(n)}$ is upper triangular which we denote by $\boldsymbol{U}$, and we thus get the so called LU decomposition

$$\boldsymbol{A} = \boldsymbol{LU},$$

of a given matrix $\boldsymbol{A}$- as a product of a lower triangular matrix with an upper triangular matrix. This is another application of the simple GEM.

REMEMBER IF AT ANY STAGE WE GET $a_{kk}^{(k)} \neq 0$ WE CANNOT PROCEED FURTHER WITH THE SIMPLE GSM.

Example 3.1:

Solve the following system using Gauss elimination method.

$$\begin{aligned} x + y + z &= 6 \\ 2x - 2y + z &= 3 \\ x + z &= 4 \end{aligned} \tag{3.7}$$

Solution: Step-1: Form the augmented matrix:

$$\boldsymbol{A}^{(1)} = \left(\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 2 & -2 & 1 & 3 \\ 1 & 0 & 1 & 4 \end{array}\right) \tag{3.8}$$

Step -2: Perform elementary row operations to get zeros below the diagonal.

2-1: First compute the multipliers for the second and third row:

$$m_{2,1}^{(1)} = -\frac{a_{21}^{(1)}}{a_{11}^{(1)}} = -\frac{2}{1} = -2 \text{ and } m_{3,1}^{(1)} = -\frac{a_{31}^{(1)}}{a_{11}^{(1)}} = -\frac{1}{1} = -1.$$

Thus,

$$\boldsymbol{M}^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \tag{3.9}$$

Now, multiply matrix (3.8) by matrix (7.7), we get $\boldsymbol{A}^{(2)}$, i.e.,

$$\boldsymbol{A}^{(2)} = \begin{pmatrix} 1 & 1 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & | & 6 \\ 2 & -2 & 1 & | & 3 \\ 1 & 0 & 1 & | & 4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & | & 6 \\ 0 & -3 & -1 & | & -9 \\ 0 & -1 & 0 & | & -2 \end{pmatrix} \tag{3.10}$$

2-2: First compute the multipliers for third row:

$$m_{3,2}^{(2)} = -\frac{a_{32}^{(2)}}{a_{22}^{(2)}} = -\frac{-1}{-3} = -\frac{1}{3}$$

Thus,

$$\boldsymbol{M}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{3} & 1 \end{pmatrix} \tag{3.11}$$

Now, multiply matrix (3.10) by matrix (7.9), we get $\boldsymbol{A}^{(3)}$, i.e.,

$$\boldsymbol{A}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{3} & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & | & 6 \\ 0 & -3 & -1 & | & -9 \\ 0 & -1 & 0 & | & -2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & | & 6 \\ 0 & -3 & -1 & | & -9 \\ 0 & 0 & \frac{1}{3} & | & 1 \end{pmatrix}$$

Step-3 Inspect the resulting system.

un-argumentation of the above matrix $\boldsymbol{A}^{(2)}$ results,

$$\boldsymbol{A}' = \begin{pmatrix} 1 & 1 & 1 \\ 0 & -3 & -1 \\ 0 & 0 & \frac{1}{3} \end{pmatrix} \text{ and } \boldsymbol{b}' = \begin{pmatrix} 6 \\ -9 \\ 1 \end{pmatrix}$$

and hence we have the system of equation

$$\begin{aligned} x+y+z&=6\\ -3y-z&=-9\\ \frac{1}{3}z&=1. \end{aligned} \tag{3.12}$$

Note that Equation (3.23) and Equation (7.4) have the same solution since the applied ERO doesn't change the solutions.

Here, we can see that, there are no zeros on the diagonal of $\boldsymbol{A}'$ and we have the number of equations as the number of unknowns. Thus, the system have a unique solution and the **BSF** method can be applied to solve the system.

Step-4 Solve the system using the **BSF**:

From the last equation we get

$$z=\frac{1}{\frac{1}{3}}=3.$$

Next, from the second equation we solve for y where $z=3$

$$y=\frac{-9-(-z)}{-3}=\frac{-6}{-3}=2$$

Finally, from the first equation we solve for x, where $y=2$ and $z=3$, as

$$x=6-y-z=6-2-3=1$$

Therefore, the solution of the given system of equations (7.4) is

$$x=1,\quad y=2\quad\text{and}\quad z=3.$$

Exercise 3.1:

Solve the following system of linear equations using the GEM:

(a)

$$\begin{aligned} x+y+z&=6\\ 2x-y+z&=3\\ x+z&=4\\ 2x+y+z&=7 \end{aligned}$$

(b)

$$\begin{aligned} x+y+z&=6\\ 2x-y+z&=3\\ x+z&=4\\ 2x+y+z&=8 \end{aligned}$$

(c)

$$\begin{aligned} x+y+z&=6\\ 2x-y+z&=3\\ 2x+2y+2z&=12 \end{aligned}$$

We observed that in order to apply simple GEM we need

$$a_{kk}^{(k)} \neq 0$$

for each stage k. This may not be satisfied always. So we have to modify the simple GEM in order to overcome this situation. Further, even if the condition

$$a_{kk}^{(k)} \neq 0$$

is satisfied at each stage, simple GEM may not be a very accurate method to use. What do we mean by this?

Example 3.2:

Consider, as an example, the following system:

$$\begin{aligned}(0.000003)x + (0.213472)y + (0.332147)z &= 0.235262\\ (0.215512)x + (0.375623)y + (0.476625)z &= 0.127653\\ (0.173257)x + (0.663257)y + (0.625675)z &= 0.285321\end{aligned} \tag{3.13}$$

Solve this system by GEM and perform the computations to 6 significant digits.

Solution:

Step-1: Form the augmented matrix:

$$\boldsymbol{A}^{(1)} = \begin{pmatrix} 0.000003 & 0.213472 & 0.332147 & | & 0.235262\\ 0.215512 & 0.375623 & 0.476625 & | & 0.127653\\ 0.173257 & 0.663257 & 0.625675 & | & 0.285321 \end{pmatrix}$$

Step -2: Perform elementary row operations to get zeros below the diagonal.

2-1: First compute the multipliers for the second and third row. Since $a_{11}^{(1)} \neq 0$, we have

$$m_{2,1}^{(1)} = -\frac{a_{21}^{(1)}}{a_{11}^{(1)}} = -\frac{0.215512}{0.000003} = -71837.3 \text{ and } m_{3,1}^{(1)} = -\frac{a_{31}^{(1)}}{a_{11}^{(1)}} = -\frac{0.173257}{0.000003} = -57752.3.$$

Thus,

$$\boldsymbol{M}^{(1)} = \begin{pmatrix} 1 & 0 & 0\\ -71837.3 & 1 & 0\\ -57752.3 & 0 & 1 \end{pmatrix}$$

Now, multiplying $\boldsymbol{A}^{(1)}$ by $\boldsymbol{M}^{(1)}$ results in $\boldsymbol{A}^{(2)}$, i.e.,

$$\boldsymbol{A}^{(2)} = \begin{pmatrix} 0.000003 & 0.213472 & 0.332147 & \mid & 0.235262 \\ 0 & -15334.9 & -23860.0 & \mid & -16900.5 \\ 0 & -12327.8 & -19181.7 & \mid & -13586.6 \end{pmatrix}$$

2-2: Compute the multipliers for third row, since $a_{22}^{(2)} = -15334.9 \neq 0$ we have

$$m_{3,2}^{(2)} = -\frac{a_{32}^{(2)}}{a_{22}^{(2)}} = -\frac{-12327.8}{-15334.9} = -0.803905$$

Thus,

$$\boldsymbol{M}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.803905 & 1 \end{pmatrix}$$

Thus, $\boldsymbol{A}^{(3)}$ obtained by multiplying $\boldsymbol{A}^{(2)}$ by $\boldsymbol{M}^{(2)}$ and we have

$$\boldsymbol{A}^{(2)} = \begin{pmatrix} 0.000003 & 0.213472 & 0.332147 & \mid & 0.235262 \\ 0 & -15334.9 & -23860.0 & \mid & -16900.5 \\ 0 & 0 & -0.50000 & \mid & -0.20000 \end{pmatrix}$$

Step-3 Inspect the resulting system.

un-argumentation of the above matrix $\boldsymbol{A}^{(2)}$ results,

$$\boldsymbol{A}' = \begin{pmatrix} 0.000003 & 0.213472 & 0.332147 \\ 0 & -15334.9 & -23860.0 \\ 0 & 0 & -0.50000 \end{pmatrix} \text{ and } \boldsymbol{b}' = \begin{pmatrix} 0.235262 \\ -16900.5 \\ -0.20000 \end{pmatrix}$$

and hence we have the system of equation

$$\begin{aligned} 0.000003x + 0.213472y + 0.332147z &= 0.235262 \\ -15334.9y - 23860.0z &= -16900.5 \\ -0.50000z &= -0.20000 \end{aligned} \tag{3.14}$$

Here, we can see that we there are no zeros on the diagonal of $\boldsymbol{A}'$ and we have the number of equations as the number of unknowns. Thus, the system have a unique solution and the **BSF** method can be applied to solve the system.

Step-4 Solve the system using the **BSF**:

From the last equation we get

$$z = \frac{-0.20000}{-0.50000} = 0.400000$$

Next, from the second equation we solve for y where $z = 0.400000$

$$y = \frac{-16900.5 - (-23860.0z)}{-15334.9} = 0.479723$$

Finally, from the first equation we solve for x, where $y = 0.479723$ and $z = 0.400000$, as

$$x = \frac{0.235262 - (0.213472y + 0.332147z)}{0.000003} = -1.33333$$

Therefore, the solution of the above system of equations (7.12) is

$$\begin{aligned} x &= -1.33333, \\ y &= 0.479723, \\ z &= 0.400000 \end{aligned}$$

This compares poorly with the correct answers (to 10 digits) given by

$$\begin{aligned} x &= -0.9912894252 \\ y &= 0.0532039339 \\ z &= 0.6741214691 \end{aligned} \tag{3.15}$$

of the given system of equations (3.13).

Thus we see that the simple Gaussian Elimination method needs modification in order to handle the situations that may lead to $a_{kk}^{(k)} = 0$ for some k or situations as arising in the above example. In order to alleviate such problems we introduce the idea of Partial Pivoting.

3.2.2 Gaussian method with partial pivoting

When performing Gaussian elimination, the diagonal element that one uses during the elimination procedure is called the pivot. To obtain the correct multiple, one uses the pivot as the divisor to the elements below the pivot. Gaussian elimination in this form will fail if the pivot is zero. In this situation, a row interchange must be performed.

Even if the pivot is not identically zero, a small value can result in big round-off errors. For very large matrices, one can easily lose all accuracy in the solution. To avoid these round-off errors arising from small pivots, row interchanges are made, and this technique is called partial pivoting (partial pivoting is in contrast to complete pivoting, where both rows and columns are interchanged).

The idea of partial pivoting is as follows. At the k^{th} stage we shall be trying to reduce all the entries below the k^{th} diagonal to zero as we did in the simple GEM. However, before we do this we look at the entries in the k^{th} diagonal and below it and then pick the one that has the largest absolute value and we bring it to the k^{th} diagonal position by a row interchange, and then reduce the entries below the k^{th} diagonal to zero. When we incorporate this idea at each stage of the Gaussian elimination process we get the **Gaussian Elimination Method with Partial Pivoting.** We now illustrate this with a few examples:

Example 3.3:

Solve the following system using Gauss elimination method.

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

Solution:

Step-1: Form the augmented matrix:

$$\boldsymbol{A}^{(1)} = \left(\begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 2 & -1 & 1 & 2 \\ 1 & 1 & 0 & 3 \end{array}\right)$$

Step -2: Perform elementary row operations to get zeros below the diagonal.

2-1 Select the pivot row by comparing the elements $a_{11}^{(1)}$ and below it and pick the one with the largest absolute value. Thus, the pivot element has to be chosen as $a_{21}^{(1)}2$ as this is the largest absolute valued entry in the first column. Therefore we need to interchange **row-**1 and **row-**2 and hence we have

$$\boldsymbol{M}^{(1)} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

and

$$\boldsymbol{A}^{(2)} = \boldsymbol{M}^{(1)}\boldsymbol{A}^{(1)} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & | & 4 \\ 2 & -2 & 1 & | & 2 \\ 1 & 0 & 1 & | & 3 \end{pmatrix} = \begin{pmatrix} 2 & -1 & 1 & | & 2 \\ 1 & 1 & 2 & | & 4 \\ 1 & 1 & 0 & | & 3 \end{pmatrix}$$

Note, that multiplying the matrix $\boldsymbol{A}^{(1)}$ by the matrix $\boldsymbol{M}^{(1)}$ will simply interchange **row-**1 and **row-**2 of the matrix $\boldsymbol{A}^{(1)}$.

2-2: Now, compute the multipliers for the second and third row:

$$m_{2,1}^{(2)} = -\frac{a_{21}^{(1)}}{a_{11}^{(1)}} = -\frac{1}{2} \text{ and } m_{3,1}^{(2)} = -\frac{a_{31}^{(1)}}{a_{11}^{(1)}} = -\frac{1}{2}.$$

Thus,

$$\boldsymbol{M}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{pmatrix}$$

Now, multiply matrix $\boldsymbol{A}^{(2)}$ by matrix $\boldsymbol{M}^{(2)}$, we get $\boldsymbol{A}^{(3)}$, i.e.,

$$\boldsymbol{A}^{(3)} = \begin{pmatrix} 2 & -1 & 1 & | & 2 \\ 0 & \frac{3}{2} & \frac{3}{2} & | & 3 \\ 0 & \frac{5}{2} & -\frac{1}{2} & | & 2 \end{pmatrix}$$

2-3 Next, the pivot element $a_{32}^3 = \frac{5}{2}$ is selected since this is the entry with the largest absolute value in the 1st column of the next sub matrix. So we have to do another row interchange, i.e., interchange **row-**2 and **row-**3. Let

$$\boldsymbol{M}^{(3)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix},$$

and we have

$$\boldsymbol{A}^{(4)} = \boldsymbol{M}^{(3)}\boldsymbol{A}^{(3)} = \begin{pmatrix} 2 & -1 & 1 & | & 2 \\ 0 & \frac{5}{2} & -\frac{1}{2} & | & 2 \\ 0 & \frac{3}{2} & \frac{3}{2} & | & 3 \end{pmatrix}$$

2-4: Compute the multipliers for third row:

$$m_{3,2}^{(4)} = -\frac{a_{32}^{(4)}}{a_{22}^{(4)}} = -\frac{\frac{3}{2}}{\frac{5}{2}} = -\frac{3}{5}$$

Thus,

$$\boldsymbol{M}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{3}{5} & 1 \end{pmatrix}$$

and here we have

$$\boldsymbol{A}^{(5)} = \boldsymbol{M}^{(4)}\boldsymbol{A}^{(4)} = \begin{pmatrix} 2 & -1 & 1 & | & 2 \\ 0 & \frac{5}{2} & -\frac{1}{2} & | & 2 \\ 0 & 0 & \frac{9}{5} & | & \frac{9}{5} \end{pmatrix}.$$

This completes the reduction and we have that the given system is equivalent to the system

$$\boldsymbol{A}'\boldsymbol{x} = \boldsymbol{b}',$$

where

$$\boldsymbol{A}' = \begin{pmatrix} 2 & -1 & 1 \\ 0 & \frac{5}{2} & -\frac{1}{2} \\ 0 & 0 & \frac{9}{5} \end{pmatrix} \text{ and } \boldsymbol{b}' = \begin{pmatrix} 2 \\ 2 \\ \frac{9}{5} \end{pmatrix}$$

Step-3 Inspect the resulting system.

$$\begin{aligned} 2x - y + z &= 2 \\ \frac{5}{2}y - \frac{1}{2}z &= 2 \\ \frac{9}{5}z &= \frac{9}{5}. \end{aligned}$$

Here, we can see that we there are no zeros on the diagonal of $\boldsymbol{A}'$ and we have the number of equations as the number of unknowns. Thus, the system have a unique solution and the **BSF** method can be applied to solve the system.

Step-4 Solving the above system using the **BSF** results

$$x = 1, \quad y = 1 \quad \text{and} \quad z = 1,$$

Exercise 3.2:

Solve the system of linear equations in Equation (3.13) by using GEM with partial pivoting and compare the results with the correct answer in Equation (7.13):

Determinant Evaluation

Notice that even in the partial pivoting method we get matrices $\boldsymbol{M}^{(k)}, \boldsymbol{M}^{(k-1)} \cdots \boldsymbol{M}^{(1)}$ such that the product $\boldsymbol{M}^{(k)}\boldsymbol{M}^{(k-1)} \cdots \boldsymbol{M}^{(1)}\boldsymbol{A}'$ is upper triangular matrix and therefore $\det \boldsymbol{M}^{(k)} \det \boldsymbol{M}^{(k-1)} \cdots \det \boldsymbol{M}^{(1)} \det \boldsymbol{A}' =$ Product of the diagonal entries in the final upper triangular matrix.

Now if $\boldsymbol{M}^{(i)}$ is used to make entries below a diagonal to zero then $\det \boldsymbol{M}^{(i)} = 1$; and if it is used for a row interchange necessary for a partial pivoting then $\det \boldsymbol{M}^{(i)} = -1$. Therefore, $\det \boldsymbol{M}^{(k)} \det \boldsymbol{M}^{(k-1)} \cdots \det \boldsymbol{M}^{(1)} = (-1)^n$ where n is the number of row interchange effected in the reduction. Hence, $\det \boldsymbol{A} = (-1)^n \times$product of the diagonal elements in the final upper triangular matrix, $\boldsymbol{A}'$.

3.2.3 Gauss Jordan Method

Consider the following matrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

This is an example of a matrix in **reduced row-echelon form**. To be in this form, a matrix must have the following properties:

1. If a row does not consist entirely of zeros, then the first nonzero number in the row is a 1. (We call this the leading 1)
2. If there are any rows that consist entirely of zeros, then they are grouped together at the bottom of the matrix.
3. In any two successive rows that do not consist entirely of zeros, the leading 1 in the lower row occurs farther to the right than the leading 1 in the higher row.
4. Each column that contains a leading 1 has zeros everywhere else.

A matrix having properties **1**, **2** and **3** (but not necessarily **4**) is said to be in **row-echelon**

form. The following two matrices are in **echelon form**

$$\begin{pmatrix} 1 & 2 & 4 & 2 \\ 0 & 7 & -1 & 5 \\ 0 & 0 & 3 & -1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 2 & 4 & 7 & 10 \\ 0 & 0 & 3 & 3 & -6 \\ 0 & 0 & 0 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Typical examples of matrices on reduced echelon form are

$$\begin{pmatrix} 1 & 2 & 0 & 0 & | & 10 \\ 0 & 0 & 1 & 0 & | & -6 \\ 0 & 0 & 0 & 1 & | & -1 \\ 0 & 0 & 0 & 0 & | & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 & | & 2 \\ 0 & 1 & 0 & | & 5 \\ 0 & 0 & 1 & | & -1 \end{pmatrix}.$$

Gaussian elimination can be used to reduce an augmented matrix to **row-echelon form.** A process called Gauss-Jordan elimination (which is an extended version of Gaussian elimination) is used to reduce an augmented matrix to **reduced row-echelon form**. Recall that with Gaussian elimination it was only necessary to get zeros **below** each leading 1. With Gauss-Jordan elimination it is desirable to get zeros **above** and **below** each leading 1 and we could also do the reduction here by partial pivoting.

Consider the following example. Gauss-Jordan elimination is the exact same as Gaussian elimination until the matrix is in row-echelon form.

Example 3.4:

Solve the following system using Gauss-Jordan method.

$$\begin{aligned} x + y + 2z &= 9 \\ 2x + 4y - 3z &= 1 \\ 3x + 6y - 5z &= 0 \end{aligned}$$

Solution:

Step-1: Form the augmented matrix:

$$\boldsymbol{A}^{(1)} = \begin{pmatrix} 1 & 1 & 2 & | & 9 \\ 2 & 4 & -3 & | & 1 \\ 3 & 6 & -5 & | & 0 \end{pmatrix}$$

Step-2 Reduce the matrix to **reduced row-echelon form**.

2-1: Normalize the diagonal element in the first-column to have a leading 1, but it's already a leading 1.

2-2: Perform elementary row operations to get zeros below the diagonal of column-1,i.e., compute the multipliers for the second and third row:

$$m_{2,1}^{(1)} = -\frac{a_{21}^{(1)}}{a_{11}^{(1)}} = -\frac{2}{1} = -2 \text{ and } m_{3,1}^{(1)} = -\frac{a_{31}^{(1)}}{a_{11}^{(1)}} = -\frac{3}{1} = -3.$$

Thus,

$$\boldsymbol{M}^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix}$$

Now, multiply matrix $\boldsymbol{A}^{(1)}$ by matrix $\boldsymbol{M}^{(1)}$, we get $\boldsymbol{A}^{(2)}$, i.e.,

$$\boldsymbol{A}^{(2)} = \begin{pmatrix} 1 & 1 & 2 & | & 9 \\ 0 & 2 & -7 & | & -17 \\ 0 & 3 & -11 & | & -27 \end{pmatrix}$$

2-3 Normalize the diagoanl element in the second column to ge the leading 1. Here, we need to divide the second row by 2. Thus, we have

$$\boldsymbol{M}^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and

$$\boldsymbol{A}^{(3)} = \boldsymbol{M}^{(2)}\boldsymbol{A}^{(2)} = \begin{pmatrix} 1 & 1 & 2 & | & 9 \\ 0 & 1 & -\frac{7}{2} & | & -\frac{17}{2} \\ 0 & 3 & -11 & | & -27 \end{pmatrix}$$

2-3 Compute multiplier for the third-row: $m_{32}^{(3)} = \frac{a_{32}^{(3)}}{a_{22}^{(3)}} = -\frac{3}{1} = -3$

Thus,

$$\boldsymbol{M}^{(3)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix}$$

and hence

$$\boldsymbol{A}^{(4)} = \boldsymbol{M}^{(3)}\boldsymbol{A}^{(3)} = \begin{pmatrix} 1 & 1 & 2 & | & 9 \\ 0 & 1 & -\frac{7}{2} & | & -\frac{17}{2} \\ 0 & 0 & -\frac{1}{2} & | & -\frac{3}{2} \end{pmatrix}.$$

3-4 Normalize the diagonal element of the third row, i.e.,

$$\boldsymbol{M}^{(4)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

and hence

$$\boldsymbol{A}^{(5)} = \boldsymbol{M}^{(4)}\boldsymbol{A}^{(4)} = \begin{pmatrix} 1 & 1 & 2 & | & 9 \\ 0 & 1 & -\frac{7}{2} & | & -\frac{17}{2} \\ 0 & 0 & 1 & | & 3 \end{pmatrix}.$$

This matrix is now in **row-echelon form** (upper triangular). To solve this matrix using Gauss-Jordan method we need to do one more step.

Step-3: Beginning with the last nonzero row and working upwards, add suitable multiples of each row to the rows above it to introduce zeroes above the leading 1's.

3-1 Compute multipliers for the first and and second row using the third row as a pivot:

$$m_{13}^{(5)} = \frac{a_{13}^{(5)}}{a_{33}^{(5)}} = \frac{7}{2} \quad \text{and} \quad m_{23}^{(5)} = \frac{a_{23}^{(5)}}{a_{33}^{(5)}} = -2$$

.

Thus,

$$\boldsymbol{M}^{(5)} = \begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & \frac{7}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

and hence

$$\boldsymbol{A}^{(6)} = \boldsymbol{M}^{(5)}\boldsymbol{A}^{(5)} = \begin{pmatrix} 1 & 1 & 0 & | & 3 \\ 0 & 1 & 0 & | & 2 \\ 0 & 0 & 1 & | & 3 \end{pmatrix}.$$

We are now finished with column 3 as there are all zeros above the leading 1. The final step involves looking at column 2 and getting a zero above the leading 1. To do this we compute multiplier for the first row using the second row as a pivot, i.e.,

$$m_{12}^{(6)} = -\frac{a_{12}^{(6)}}{a_{22}^{(6)}} = -1$$

and hence we have

$$\boldsymbol{M}^{(6)} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and hence

$$\boldsymbol{A}^{(7)} = \boldsymbol{M}^{(6)}\boldsymbol{A}^{(6)} = \begin{pmatrix} 1 & 0 & 0 & | & 1 \\ 0 & 1 & 0 & | & 2 \\ 0 & 0 & 1 & | & 3 \end{pmatrix}.$$

This completes the reduction and we have that the given system is equivalent to the system

$$\boldsymbol{A}'\boldsymbol{x} = \boldsymbol{b}',$$

where

$$\boldsymbol{A}' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } \boldsymbol{b}' = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Step-4 Inspect the resulting system.

$$\begin{aligned} x + 0y + 0z &= 1 \\ 0x + y + 0z &= 2 \\ 0x + 0y + z &= 3. \end{aligned}$$

Here, we can see that we there are no zeros on the diagonal of $\boldsymbol{A}'$ and we have the number of equations as the number of unknowns. Thus, the system have a unique solution is equal to $\boldsymbol{b}'$,i.e.,

$$x = 1, \quad y = 2, \quad \text{and} \quad z = 3.$$

Note that the final matrix is in **reduced row-echelon form** and it's and identity matrix.

Remark:

The natural question that we need to ask at this point is "why the extra computations in Gauss Jordan Method?". The Gauss Jordan Method will result the inverse of the original matrix and also to determine the determinant of the matrix we simply take the product of the determinant $\boldsymbol{M}^{k}$'s. The inverse of a matrix can be computed in two ways, one way is take the product $\boldsymbol{M}^{(k)}\boldsymbol{M}^{(k-1)}\cdots\boldsymbol{M}^{(1)} = \boldsymbol{A}^{-1}$ and the other way is by augmenting the original matrix with and identity matrix. The Gauss Jordan method will reduce the original matrix to an identity matrix and the identity matrix will become the inverse of the original matrix.

3.2.4 Gauss Jordan Method for matrix inversion

As we try to explain above the Gauss Jordan method is also used to find inverse of a given matrix. We simply formalize the procedure using the following example.

Example 3.5:

Solve the following system using Gauss-Jordan method and also find the inverse of the coefficient matrix and it's determinant.

$$\begin{aligned} x+2y+3z&=12\\ 3x+2y+z&=24\\ 2x+y+3z&=36 \end{aligned}$$

Solution:

Step-1: In order to solve and find the inverse of the coefficient matrix the first need we need to do is forming the augmented matrix of $\boldsymbol{A}$, $\boldsymbol{b}$ and $\boldsymbol{I}$, where $\boldsymbol{I}$ is an identity matrix of same order as $\boldsymbol{A}$:

$$\boldsymbol{A}^{(1)}=\begin{pmatrix}1&2&3&|&12&|&1&0&0\\3&2&1&|&24&|&0&1&0\\2&1&3&|&36&|&0&0&1\end{pmatrix}$$

Step-2 Reduce the matrix to **reduced row-echelon form**. Here, we will be a little bit systematic so that we can reduce the computation and we are not going to compute the $\boldsymbol{M}^k$ rather we follow a different form.

2-1: Now normalize the all rows by factoring out the lead elements of the first column so that

$$\boldsymbol{A}^{(1)}\begin{matrix}\xrightarrow{\frac{1}{3}R_2}\\ \xrightarrow{\frac{1}{2}R_3}\end{matrix}\begin{pmatrix}1&2&3&|&12&|&1&0&0\\1&\frac{2}{3}&\frac{1}{3}&|&8&|&0&\frac{1}{3}&0\\1&\frac{1}{2}&\frac{3}{2}&|&18&|&0&0&\frac{1}{2}\end{pmatrix}(1)(3)(2)=\boldsymbol{A}^{(2)}.$$

Note: the product (1)(3)(2) is to for the computation of the the determinant.

- The first row can then be subtracted from the remaining rows (i.e. rows 2 and 3) to yield

$$A^{(2)} \begin{array}{c}\xrightarrow{R2-R_1}\\ \xrightarrow{R3-R_1}\end{array} \begin{pmatrix} 1 & 2 & 3 & | & 12 & | & 1 & 0 & 0\\ 0 & -\frac{4}{3} & -\frac{8}{3} & | & -4 & | & -1 & \frac{1}{3} & 0\\ 0 & -\frac{3}{2} & -\frac{3}{2} & | & 6 & | & -1 & 0 & \frac{1}{2}\end{pmatrix}(6) = A^{(3)}.$$

2-2 Now repeat the cycle normalizing by factoring out the elements of the second column getting

$$A^{(3)} \begin{array}{c}\xrightarrow{\frac{1}{2}R_1}\\ \xrightarrow{-\frac{3}{4}R_2}\\ \xrightarrow{-\frac{2}{3}R_3}\end{array} \begin{pmatrix} \frac{1}{2} & 1 & \frac{3}{2} & | & 6 & | & \frac{1}{2} & 0 & 0\\ 0 & 1 & 2 & | & 3 & | & \frac{3}{4} & -\frac{1}{4} & 0\\ 0 & 1 & 1 & | & -4 & | & \frac{2}{3} & 0 & -\frac{1}{3}\end{pmatrix}(6)(2)(-\frac{4}{3})(-\frac{3}{2}) = A^{(4)}.$$

- Subtracting the second row from the remaining rows (i.e. rows 1 and 3) gives

$$A^{(4)} \begin{array}{c}\xrightarrow{R1-R_2}\\ \xrightarrow{R3-R_2}\end{array} \begin{pmatrix} \frac{1}{2} & 0 & -\frac{1}{2} & | & 3 & | & -\frac{1}{4} & \frac{1}{4} & 0\\ 0 & 1 & 2 & | & 3 & | & \frac{3}{4} & -\frac{1}{4} & 0\\ 0 & 0 & -1 & | & -7 & | & -\frac{1}{12} & \frac{1}{4} & -\frac{1}{3}\end{pmatrix}(24) = A^{(5)}.$$

2-3 Again repeat the cycle normalizing by the elements of the third column so

$$A^{(5)} \begin{array}{c}\xrightarrow{-2R_1}\\ \xrightarrow{\frac{1}{2}R_2}\\ \xrightarrow{-R3}\end{array} \begin{pmatrix} -1 & 0 & 1 & | & -6 & | & \frac{1}{2} & -\frac{1}{2} & 0\\ 0 & \frac{1}{2} & 1 & | & \frac{3}{2} & | & \frac{3}{8} & -\frac{1}{8} & 0\\ 0 & 0 & 1 & | & 7 & | & \frac{1}{12} & -\frac{1}{4} & \frac{1}{3}\end{pmatrix}(24)(-\frac{1}{2})(2)(-1) = A^{(6)}.$$

- and subtract third row from the remaining rows to yield

$$A^{(6)} \begin{array}{c}\xrightarrow{R_1-R3}\\ \xrightarrow{R_2-R3}\end{array} \begin{pmatrix} -1 & 0 & 0 & | & -13 & | & \frac{5}{12} & -\frac{1}{4} & -\frac{1}{3}\\ 0 & \frac{1}{2} & 0 & | & -\frac{11}{2} & | & \frac{7}{24} & \frac{1}{8} & -\frac{1}{3}\\ 0 & 0 & 1 & | & 7 & | & \frac{1}{12} & -\frac{1}{4} & \frac{1}{3}\end{pmatrix}(24) = A^{(7)}.$$

2-4 Finally normalize each row by the diagonal elements so as to produce the unit matrix on the left hand side so that

$$\boldsymbol{A}^{(7)} \begin{array}{c}\xrightarrow{-R_1}\\ \xrightarrow{2R_2}\\ \xrightarrow{R_3}\end{array} \left(\begin{array}{ccc|c|ccc} 1 & 0 & 0 & 13 & -\frac{5}{12} & \frac{1}{4} & \frac{1}{3} \\ 0 & 1 & 0 & -11 & \frac{7}{12} & \frac{1}{4} & -\frac{2}{3} \\ 0 & 0 & 1 & 7 & \frac{1}{12} & -\frac{1}{4} & \frac{1}{3} \end{array}\right)(24)(-1)(\frac{1}{2})(2)(1) = \boldsymbol{A}^{(8)}.$$

Step-3 The solution to the equations is now contained in the center vector while the right hand matrix contains the inverse of the original matrix that was on the left hand side of expression (2.2.14). The scalar quantity accumulating at the front of the matrix is the determinant as it represents factors of individual rows of the original matrix. Thus our complete solution is

$$x = 13, \quad y = -11 \quad \text{and} \quad z = 7.$$

The inverse of the coefficient matrix is

$$\boldsymbol{A}^{-1} = \begin{pmatrix} -\frac{5}{12} & \frac{1}{4} & \frac{1}{3} \\ \frac{7}{12} & \frac{1}{4} & -\frac{2}{3} \\ \frac{1}{12} & -\frac{1}{4} & \frac{1}{3} \end{pmatrix}$$

and also the determinant of the coefficeint matrix is $\det \boldsymbol{A} = -12$.

3.2.5 Matrix Decomposition Method

As described in the previous sections, Gauss elimination is designed to solve systems of linear algebraic equations

$$\boldsymbol{Ax} = \boldsymbol{b}$$

Although it certainly represents a sound way to solve such systems, it becomes inefficient when solving equations with the same coefficients $\boldsymbol{A}$, but with different right-hand-side constants (the $\boldsymbol{b}$s). Recall that Gauss elimination involves two steps: forward elimination and back-substitution. Of these, the forward-elimination step comprises the bulk of the computational effort. This is particularly true for large systems of equations. On the other hand, **LU** decomposition methods separate the time-consuming elimination of the matrix $\boldsymbol{A}$ from the manipulations of the right-hand side $\boldsymbol{b}$. Thus, once the matrix $\boldsymbol{A}$ has been decomposed, multiple right-hand-side vectors can be evaluated in an efficient manner. Before showing how this can be done, let us first provide a mathematical overview of the decomposition strategy.

Overview of the LU Decomposition

We shall now consider the $\boldsymbol{LU}$ decomposition of matrices. Suppose $\boldsymbol{A}$ is an $n \times n$ matrix. If $\boldsymbol{L}$ and $\boldsymbol{U}$ are lower and upper triangular $n \times n$ matrices respectively such that $\boldsymbol{A} = \boldsymbol{LU}$. We say that this is the $\boldsymbol{LU}$ decomposition of $\boldsymbol{A}$. Note that $\boldsymbol{LU}$ decomposition is **not unique**. For example if $\boldsymbol{A} = \boldsymbol{LU}$ is a decomposition then $\boldsymbol{A} = \boldsymbol{L}_\alpha \boldsymbol{U}_\alpha$ is also an $\boldsymbol{LU}$ decomposition where $\alpha \neq 0$ is any scalar and $\boldsymbol{L}_\alpha = \alpha \boldsymbol{L}$ and $\boldsymbol{U}_\alpha = \frac{1}{\alpha}\boldsymbol{L}$.

A two-step strategy for obtaining solutions of system of equations of the form

$$\boldsymbol{Ab} = \boldsymbol{b},$$

can be based explained as follow:

- **LU decomposition step.** $\boldsymbol{A}$ is factored or decomposed into lower $\boldsymbol{L}$ and upper $\boldsymbol{U}$ triangular matrices.
- **Substitution step.** $\boldsymbol{L}$ and $\boldsymbol{U}$ are used to determine a solution $\boldsymbol{x}$ for a right-hand side column vector $\boldsymbol{b}$. This step itself consists of two steps. First, an intermediate vector $\boldsymbol{y}$ is generated by forward substitution. Then, the result is substituted back to solve for $\boldsymbol{x}$ by back substitution.

$$\boldsymbol{A} = \boldsymbol{LU},$$

$$\boldsymbol{y} = \boldsymbol{Ux},$$

$$\boldsymbol{LUx} = \boldsymbol{Ly} = \boldsymbol{b},$$

- The forward-substitution step can be represented concisely as

$$\begin{aligned} y_1 &= \frac{b_1}{l_{11}}, \\ y_i &= \frac{1}{l_{ii}}\left[b_i - \sum_{j=1}^{i-1} l_{ij} y_j\right], \quad \text{for } i = 2, 3, \cdots, n. \end{aligned} \tag{3.16}$$

- the back-substitution step can be represented concisely as

$$\begin{aligned} x_{nn} &= \frac{y_{nn}}{u_{nn}}, \\ x_i &= \frac{1}{u_{ii}}\left[y_i - \sum_{j=i+1}^{n} u_{ij} y_j\right], \quad \text{for } i = 2, 3, \cdots, n. \end{aligned} \tag{3.17}$$

Further if $\boldsymbol{A} = \boldsymbol{LU}$ is a **LU** decomposition then $\det \boldsymbol{A}$ can be calculated as $\det \boldsymbol{A} = \det \boldsymbol{L} \times \det \boldsymbol{U} = l_{11} l_{22} \cdots l_{nn} u_{11} u_{22} \cdots u_{nn}$, where l_{ii} are the diagonal entries of $\boldsymbol{L}$ and u_{ii} are the diagonal entries of $\boldsymbol{U}$.

We shall now give methods to find **LU** decomposition of a matrix. Basically, we shall be considering three cases. First, we shall consider the the **Doolittles's** and **Crout's method** for a general matrix; secondly the **Cholesky's** method for a symmetric matrix, and thirdly the **decomposition of Tridiagonal matrix.** Before discuss these methods let's discuss the **LU** of a given matrix $\boldsymbol{A}$ using the Gaussian Elimination method.

LU decomposition using Gauss elimination procedure

When the Gauss elimination procedure is applied to a matrix , the elements of the matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ are actually calculated. The upper triangular matrix $\boldsymbol{U}$ is the matrix of coefficients $\boldsymbol{A}$ that is obtained at the end of the Gauss elimination procedure. For the lower triangular matrix $\boldsymbol{L}$, the elements on the diagonal are all 1, and the elements below the diagonal are the negative of multipliers $m_{ij}^{(k)}$ that multiply the pivot equation when it is used to eliminate the elements below the pivot coefficient. For the case of a system of three equations, the decomposition has the form:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ m_{21}^{(1)} & 1 & 0 \\ m_{31}^{(1)} & m_{32}^{(2)} & 1 \end{pmatrix} \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & 0 & a_{33}^{(3)} \end{pmatrix},$$

where

$$m_{21}^{(1)} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}}, \quad m_{31}^{(1)} = \frac{a_{31}^{(1)}}{a_{11}^{(1)}}, \quad \text{and} \quad m_{32}^{(2)} = \frac{a_{32}^{(2)}}{a_{22}^{(2)}}.$$

Let the **LU** decomposition of $\boldsymbol{A}$ is given as:

$$\begin{pmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{12} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & u_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{pmatrix} \text{ and } \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_23 & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix} \tag{3.18}$$

In order to obtain a unique solution it is convenient to choose either $l_{ii} = 1$ or $u_{ii} = 1$. Depending of these choices we will have to methods of the **LU** decomposition.

a) if $l_{ii} = 1$ for $i = 1, 2 \cdots, n$ the method is called the **Doolittle's method** and

b) if $u_{ii} = 1$ for $i = 1, 2 \cdots, n$ the method is called **Crout's method**.

LU decomposition using Doolittle's Method

Let $\boldsymbol{A} = (a_{ij})$. We seek an **LU** decomposition in which the diagonal entries l_i of $\boldsymbol{L}$ are all 1. Let $\boldsymbol{L} = (l_{ij})$; $\boldsymbol{U} = (u_{ij})$. Since $\boldsymbol{L}$ is a lower triangular matrix, we have $l_{ij} = 0$ if $j > i$; and by our choice, $l_{ii} = 1$. Similarly, since $\boldsymbol{U}$ is an upper triangular matrix, we have $u_{ij} = 0$ if $i > j$.

We determine $\boldsymbol{L}$ and $\boldsymbol{U}$ as follows: The 1^{st} row of $\boldsymbol{U}$ and 1^{st} column of $\boldsymbol{L}$ are determined as follows :

$$\begin{aligned} a_{11} &= \sum_{k=1}^{n} l_{1k} u_{k1} \\ &= l_{11} u_{11}, && \text{since } l_{1k} = 0 \text{ for } k > 1, \\ &= u_{11}, && \text{since } l_{11} = 1. \end{aligned}$$

Therefore, in general

$$\boxed{u_{11} = a_{11}. \qquad (3.19)}$$

Now,

$$\begin{aligned} a_{1j} &= \sum_{k=1}^{n} l_{1k} u_{kj} \\ &= l_{11} u_{1j} && \text{since } l_{1k} = 0 \text{ for } k > 1, \\ &= u_{1j} && \text{since } l_{11} = 1. \end{aligned}$$

$$\boxed{\implies u_{1j} = a_{1j} \qquad (3.20)}$$

Thus the first row of $\boldsymbol{U}$ is the same as the first row of $\boldsymbol{A}$. The first column of $\boldsymbol{L}$ is determined as follows:

$$\begin{aligned} a_{j1} &= \sum_{k=1}^{n} l_{jk}u_{k1} \\ &= l_{j1}u_{11} \qquad \text{since } u_{k1} = 0 \text{ for } k > 1, \end{aligned}$$

$$\boxed{\implies l_{j1} = \frac{a_{j1}}{u_{11}} \qquad (3.21)}$$

Note : u_{11} is already obtained from Equation (3.19)

Thus Equation (3.20) and Equation (3.21) determine respectively the first row of $\boldsymbol{U}$ and first column of $\boldsymbol{L}$. The other rows of $\boldsymbol{U}$ and columns of $\boldsymbol{L}$ are determined recursively as given below: Suppose we have determined the first $i-1$ rows of $\boldsymbol{U}$ and the first $i-1$ columns of $\boldsymbol{L}$. Now we proceed to describe how one then determines the i^{th} row of $\boldsymbol{U}$ and i^{th} column of $\boldsymbol{L}$. Since first $i-1$ rows of $\boldsymbol{U}$ have been determined, this means, u_{kj} are all known for $1 \le k \le i-1$; $1 \le j \le n$. Similarly, since first $i-1$ columns are known for L, this means, l_{ik} are all known for $1 \le i \le n$; $1 \le k \le i-1$.

Now,

$$\begin{aligned} a_{ij} &= \sum_{k=1}^{n} l_{ik}u_{kj} \\ &= \sum_{k=1}^{i} l_{ik}u_{kj} \qquad \text{since } l_{ik} = 0 \text{ for } k > i, \\ &= \sum_{k=1}^{i-1} l_{ik}u_{kj} + l_{ii}u_{ij} \\ &= \sum_{k=1}^{i-1} l_{ik}u_{kj} + u_{ij} \qquad \text{since } l_{ii} = 1. \end{aligned}$$

Thus, solving for u_{ij} results in

$$\boxed{\implies u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}. \qquad (3.22)}$$

Note that on the RHS we have a_{ij} which is known from the given matrix. Also the sum on the RHS involves l_{ik} for $1 \leq k \leq i-1$ which are all known because they involve entries in the first $i-1$ columns of $\boldsymbol{L}$; and they also involve u_{kj}; $1 \leq k \leq i-1$ which are also known since they involve only the entries in the first $i-1$ rows of $\boldsymbol{U}$. Thus Equation (3.22) determines the i^{th} row of $\boldsymbol{U}$ in terms of the known given matrix and quantities determined upto the previous stage. Now we describe how to get the i^{th} column of $\boldsymbol{L}$:

$$\begin{aligned} a_{ji} &= \sum_{k=1}^{n} l_{jk}u_{ki} \\ &= \sum_{k=1}^{i} l_{jk}u_{ki} \qquad \text{since } u_{ki} = 0\text{for } k > i, \\ &= \sum_{k=1}^{i-1} l_{jk}u_{ki} + l_{ji}u_{ii}, \end{aligned}$$

solving for l_{ji} gives us:

$$l_{ji} = \frac{1}{u_{ii}}\left[a_{ji} - \sum_{k=1}^{i-1} l_{jk}u_{ki}\right]. \qquad (3.23)$$

Once again we note the RHS involves u_{ii}, which has been determined using Equation (3.22); a_{ij} which is from the given matrix; $l_{jk}; 1 \leq k \leq i-1$ and hence only entries in the first $i-1$ columns of $\boldsymbol{L}$; and $u_{ki}, 1 \leq k \leq i-1$ and hence only entries in the first $i-1$ rows of $\boldsymbol{U}$. Thus RHS in Equation (3.23) is completely known and hence l_{ji}, the entries in the i^{th} column of $\boldsymbol{L}$ are completely determined by Equation (3.23).

Summary:

The summary of Doolittle's procedure is as follows:

Step-1 determining 1^{st} row of $\boldsymbol{U}$ and 1^{st} column of $\boldsymbol{L}$:

a) The diagonal elements of $\boldsymbol{L}$ are 1, i.e.,

$$l_{ii} = 1, \quad \text{for } \; i = 1, 2 \cdots, n.$$

b) The 1^{st} row of $\boldsymbol{U}$ are given by the 1^{st} row of $\boldsymbol{A}$, i.e.,

$$u_{1,j} = a_{1,j} \quad \text{for } \; j = 1, 2, \cdots, n.$$

c). The 1st column of $\boldsymbol{L}$ is given by Equation (3.21), i.e.,

$$l_{j1} = \frac{a_{j1}}{u_{11}} \quad \text{for} \;\; j = 1, 2, \cdots, n.$$

Step-i for $i = 2, 3, \cdots, n$ we determine:

a) Determine the i^{th} row of $\boldsymbol{U}$ using Equation (3.22), i.e.,

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}, \qquad \text{for } j = i, i+1, \cdots, n.$$

Note that for $j < i$ we have $u_{ij} = 0$.

b.) Determine the i^{th} column of $\boldsymbol{L}$ using Equation (3.23), i.e.,

$$l_{ji} = \frac{1}{u_{ii}} \left[a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki} \right], \qquad \text{for } j = i, i+1, \cdots, n.$$

Note that for $j < i$ we have $l_{ji} = 0$.

We observe that the method fails if $u_{ii} = 0$ for some i.

Example 3.6:

Determine the the Doolittle's decomposition for the matrix.

$$A = \begin{pmatrix} 2 & 1 & -1 & 3 \\ -2 & 2 & 6 & -4 \\ 4 & 14 & 19 & 4 \\ 6 & 0 & -6 & 12 \end{pmatrix}$$

Solution:

Step-1 Determine the 1st row of $\boldsymbol{U}$ and 1st column of $\boldsymbol{L}$:

a) The diagonal elements of $\boldsymbol{L}$ are 1, i.e.,

$$l_{ii} = 1, \quad \text{for} \;\; i = 1, 2 \cdots, n.$$

Thus, we have,

$$l_{11} = 1, \; l_{22} = 1, \; l_{33} = 1, \;\; \text{and} \;\; l_{44} = 1$$

b) <u>1st row of $\boldsymbol{U}$</u>

The first row of $\boldsymbol{U}$ is given by the first two of $\boldsymbol{A}$, i.e.,

$$u_{1,j} = a_{1,j} \text{ for } \; j = 1, 2, \cdots, n.$$

Thus we have

$$u_{11} = 2, \; u_{12} = 1, \; u_{13} = -1, \; \text{ and } \; u_{14} = 3$$

c). <u>1st column of $\boldsymbol{L}$</u>

The fist column of $\boldsymbol{L}$ is given by Equation (3.21), i.e.,

$$l_{j1} = \frac{a_{j1}}{u_{11}} \text{for } \; j = 1, 2, \cdots, n.$$

Thus, $j = 1$

$$\implies \qquad l_{11} = 1,$$

$j = 2$

$$\implies \qquad l_{21} = \frac{a_{21}}{u_{11}} = \frac{-2}{2} = -1,$$

$j = 3$

$$\implies \qquad l_{31} = \frac{a_{31}}{u_{11}} = \frac{4}{2} = 2.$$

and $j = 4$

$$\implies \qquad l_{41} = \frac{a_{41}}{u_{11}} = \frac{6}{2} = 3,$$

Step-2 a) <u>2th row of $\boldsymbol{U}$:</u>

The second row of $\boldsymbol{U}$ is given by Equation (3.22) when i=2, i.e.,

$$u_{2j} = a_{2j} - \sum_{k=1}^{2-1} l_{2k} u_{kj}, \qquad \text{for } j = 2, 3, 4.$$

Thus, we have

$j = 2$

$$\implies \qquad u_{22} = a_{22} - l_{21} u_{12} = 2 - (-1)(1) = 3$$

$j = 3$

$$\implies \qquad u_{23} = a_{23} - l_{21} u_{13} = 6 - (-1)(-1) = 5$$

$j = 4$

$$\implies \qquad u_{24} = a_{23} - l_{21} u_{14} = -4 - (-1)(3) = -1$$

b.) <u>2nd column of $\boldsymbol{L}$:</u> The second column of $\boldsymbol{L}$ is given by Equation (3.23) when $i = 2$, i.e.,

$$l_{j2} = \frac{1}{u_{22}}\left[a_{j2} - \sum_{k=1}^{2-1} l_{jk}u_{k2}\right], \qquad \text{for } j = 2, 3, 4.$$

Thus, we have $j = 2$

$$\Longrightarrow \qquad l_{22} = 1$$

$j = 3$

$$\Longrightarrow \qquad l_{32} = \frac{1}{u_{22}}[a_{32} - l_{31}u_{12}] = \frac{1}{3}[14 - (2)(1)] = 4$$

$j = 4$

$$\Longrightarrow \qquad l_{42} = \frac{1}{u_{22}}[a_{42} - l_{41}u_{12}] = \frac{1}{3}[0 - (3)(1)] = -1$$

Step-3 a) <u>3rd row of $\boldsymbol{U}$:</u>

The third row of $\boldsymbol{U}$ is given by Equation (3.22) when i=3, i.e.,

$$u_{3j} = a_{3j} - \sum_{k=1}^{3-1} l_{3k}u_{kj}, \qquad \text{and } j = 3, 4.$$

Thus, we have

$j = 3$

$$\Longrightarrow \qquad u_{33} = a_{33} - (l_{31}u_{13} + l_{32}u_{23}) = 19 - ((2)(-1) + (4)(5)) = 1$$

$j = 4$

$$\Longrightarrow \qquad u_{34} = a_{34} - (l_{31}u_{14} + l_{32}u_{24}) = 4 - ((2)(3) + (4)(-1)) = 2$$

b.) <u>3rd column of $\boldsymbol{L}$:</u> The second column of $\boldsymbol{L}$ is given by Equation (3.23) when $i = 3$, i.e.,

$$l_{j3} = \frac{1}{u_{33}}\left[a_{j3} - \sum_{k=1}^{3-1} l_{jk}u_{k3}\right], \qquad \text{for } j = 3, 4.$$

Thus, we have $j = 3$

$$\Longrightarrow \qquad l_{33} = 1$$

$j = 4$

$$\implies \begin{aligned} l_{43} &= \frac{1}{u_{33}}\left[a_{43} - (l_{41}u_{13} + l_{42}u_{23})\right] \\ &= \frac{1}{1}\left[-6 - ((3)(-1) + (-1)(5))\right] \\ &= 2 \end{aligned}$$

Step-4 a) 4th row of $\boldsymbol{U}$:

The fourth row of $\boldsymbol{U}$ is given by Equation (3.22) when i=4, i.e.,

$$u_{4j} = a_{4j} - \sum_{k=1}^{4-1} l_{4k}u_{kj}, \qquad \text{and } j = 4.$$

Thus, we have

$j = 4$

$$\implies \begin{aligned} u_{44} &= a_{44} - (l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34}) \\ &= 12 - ((3)(3) + (-1)(-) + (2)(2)) \\ &= -2 \end{aligned}$$

b.) 4th column of $\boldsymbol{L}$: The second column of $\boldsymbol{L}$ is given by Equation (3.23) when $i = 4$, i.e.,

$$l_{j4} = \frac{1}{u_{44}}\left[a_{j4} - \sum_{k=1}^{4-1} l_{jk}u_{k4}\right], \qquad \text{for } j = 4.$$

Thus, we have $j = 4$

$$\implies \qquad l_{44} = 1$$

Now, collecting the elements of $\boldsymbol{L}$ and $\boldsymbol{U}$ we get

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 2 & 4 & 1 & 0 \\ 3 & -1 & 2 & 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{U} = \begin{pmatrix} 2 & 1 & -1 & 3 \\ 0 & 3 & 5 & -1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & -2 \end{pmatrix}$$

This completes the **LU** decomposition by Doolittle's method for the given $\boldsymbol{A}$.

LU decomposition using Crout's method

The second method for decomposing a general matrix A into the **LU** factor is the Crout's Method. In this method the matrix is decomposed into the product $\boldsymbol{LU}$, where the diagonal elements of the matrix $\boldsymbol{U}$ are all 1s. It turns out that in this case, the elements of both matrices can be determined using formulas that can be easily programmed just like the Doolittel's method.

A procedure for determining the elements of the matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ using the Crout's method can be written as follow.

Step-1: Calculate the first column of $\boldsymbol{L}$ and the first row of $\boldsymbol{U}$.

a.) Substituting 1s in the diagonal of $\boldsymbol{U}$:

$$u_{ii} = 1 \quad \text{for } i = 1, 2, \cdots n.$$

b.) Calculating the first column of $\boldsymbol{L}$ by using

$$l_{i1} = a_{i1}, \quad \text{for } i = 1, 2, \cdots n.$$

c.) Calculating the fist row of $\boldsymbol{U}$ by using

$$u_{1j} = \frac{a_{1j}}{l_{11}} \quad \text{for } j = 2, \cdots n.$$

Step-i: Calculate the i^{th} column of $\boldsymbol{L}$ and the i^{th} row of $\boldsymbol{U}$, for $i = 2, 3, \cdots, n$.

a.) Calculate the i^{th} column of $\boldsymbol{L}$:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \quad \text{for } j = 2, 3, \cdots i.$$

b.) Calculate the i^{th} row of $\boldsymbol{U}$:

$$u_{ij} = \frac{1}{l_{ii}} \left[a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \right], \qquad \text{for } j = i+1, i+2, \cdots, n$$

Example 3.7:

Solve the following system of linear equations by Crout's Method (**LU** factorization or decomposition method):

$$\begin{aligned}
9x_1 + 3x_2 + 3x_3 + 3x_4 &= 24\\
3x_1 + 10x_2 - 2x_3 - 2x_4 &= 17\\
3x_1 - 2x_2 + 18x_3 + 10x_4 &= 45\\
3x_1 - 2x_2 + 10x_3 + 10x_4 &= 29
\end{aligned}$$

Solution: The given system of equation can be written in matrix form as

$$\begin{pmatrix} 9 & 3 & 3 & 3\\ 3 & 10 & -2 & -2\\ 3 & -2 & 18 & 10\\ 3 & -2 & 10 & 10 \end{pmatrix}\begin{pmatrix} x_1\\ x_2\\ x_3\\ x_4 \end{pmatrix} = \begin{pmatrix} 24\\ 17\\ 45\\ 29 \end{pmatrix}$$

Step-1: Calculate the first column of $\boldsymbol{L}$ and the first row of $\boldsymbol{U}$.

a.) Substituting 1s in the diagonal of $\boldsymbol{U}$:

$$u_{ii} = 1 \quad \text{for } i = 1, 2, \cdots n.$$

Thus, we have

$$u_{11} = 1,\ u_{22} = 1,\ u_{33} = 1,\ \text{and}\ \ u_{44} = 1.$$

b.) Calculating the first column of $\boldsymbol{L}$ by using

$$l_{i1} = a_{i1}, \quad \text{for } i = 1, 2, \cdots n.$$

Here we have

$$l_{11} = a_{11} = 9,\ l_{21} = a_{21} = 3,\ l_{31} = a_{31} = 3,\ \text{and}\ \ l_{41} = a_{41} = 3.$$

c.) Calculating the fist row of $\boldsymbol{U}$ by using

$$u_{1j} = \frac{a_{1j}}{l_{11}} \quad \text{for } j = 2, 3, \cdots n.$$

$j = 2$

$$\Longrightarrow \qquad u_{12} = \frac{l_{12}}{l_{11}} = \frac{3}{9} = \frac{1}{3}$$

$j = 3$

$$\Longrightarrow \qquad u_{13} = \frac{l_{13}}{l_{11}} = \frac{3}{9} = \frac{1}{3}$$

$j = 4$

$$\Longrightarrow \qquad u_{14} = \frac{l_{14}}{l_{11}} = \frac{3}{9} = \frac{1}{3}$$

Step-2: Calculate the 2nd column of $\boldsymbol{L}$ and the 2nd row of $\boldsymbol{U}$, for $i = 2, 3, \cdots, n$.

a.) Calculate the 2nd column of $\boldsymbol{L}$:

$$l_{2j} = a_{2j} - \sum_{k=1}^{j-1} l_{2k}u_{kj} \quad \text{for } j = 2 \cdots i.$$

$j = 2$

$$l_{22} = a_{22} - l_{21}u_{12} = 10 - (3)(\frac{1}{3}) = 9$$

b.) Calculate the 2th row of $\boldsymbol{U}$:

$$u_{2j} = \frac{1}{l_{22}}\left[a_{2j} - \sum_{k=1}^{2-1} l_{2k}u_{kj}, \qquad \text{for} j = 3, 4\right]$$

$j = 3$

$$u_{23} = \frac{1}{l_{22}}[a_{23} - l_{21}u_{13}] = \frac{1}{9}\left[-2 - (3)(\frac{1}{3})\right] = -\frac{1}{3}$$

$j = 4$

$$u_{24} = \frac{1}{l_{22}}[a_{24} - l_{21}u_{14}] = \frac{1}{9}\left[-2 - (3)(\frac{1}{3})\right] = -\frac{1}{3}$$

Step-3: Calculate the 3rd column of $\boldsymbol{L}$ and the 3rd row of $\boldsymbol{U}$, for $i = 2, 3$.

a.) Calculate the 3rd column of $\boldsymbol{L}$:

$$l_{3j} = a_{3j} - \sum_{k=1}^{j-1} l_{3k}u_{kj} \quad \text{for } j = 2, 3 \cdots i.$$

$j = 2$

$$l_{32} = a_{32} - l_{31}u_{12} = -2 - (3)(\frac{1}{3}) = -3$$

$j = 3$

$$l_{33} = a_{33} - [l_{31}u_{13} + l_{32}u_{23}] = 18 - \left[(3)(\frac{1}{3}) + (-3)(-\frac{1}{3})\right] = 16$$

b.) Calculate the 3rd row of $\boldsymbol{U}$:

$$u_{3j} = \frac{1}{l_{33}}\left[a_{3j} - \sum_{k=1}^{3-1} l_{3k}u_{kj}\right], \qquad \text{for} j = 4$$

$j = 4$

$$u_{34} = \frac{1}{l_{33}}\left[a_{34} - (l_{31}u_{14} + l_{32}u_{24})\right] = \frac{1}{16}\left[10 - \left((3)(\frac{1}{3}) + (-3)(-\frac{1}{3})\right)\right] = \frac{1}{2}$$

Step-4: Calculate the 4th column of $\boldsymbol{L}$ and the 4th row of $\boldsymbol{U}$, for $i = 2, 3, 4$.

a.) Calculate the 4th column of $\boldsymbol{L}$:

$$l_{4j} = a_{4j} - \sum_{k=1}^{j-1} l_{4k}u_{kj} \quad \text{for } j = 2, 3 \cdots i.$$

$j = 2$

$$l_{42} = a_{42} - l_{41}u_{12} = -2 - (3)(\frac{1}{3}) = -3$$

$j = 3$

$$l_{43} = a_{43} - [l_{41}u_{13} + l_{42}u_{23}] = 10 - \left[(3)(\frac{1}{3}) + (-3)(-\frac{1}{3})\right] = 8$$

$j = 4$

$$l_{44} = a_{44} - [l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34}] = 10 - \left[(3)(\frac{1}{3}) + (-3)(-\frac{1}{3}) + (8)(\frac{1}{2})\right] = 4$$

b.) Calculate the 4th row of $\boldsymbol{U}$:

$$u_{44} = 1$$

Now, collecting the elements of $\boldsymbol{L}$ and $\boldsymbol{U}$ we get

$$\boldsymbol{L} = \begin{pmatrix} 9 & 0 & 0 & 0 \\ 3 & 9 & 0 & 0 \\ 3 & -3 & 16 & 0 \\ 3 & -3 & 8 & 4 \end{pmatrix} \quad \text{and} \quad \boldsymbol{U} = \begin{pmatrix} 1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 1 & -\frac{1}{3} & -\frac{1}{3} \\ 0 & 0 & 1 & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This completes the **LU** decomposition by Doolittle's method for the given $\boldsymbol{A}$. Now, let us solve the given system

- Forward substitution gives

$$\boldsymbol{Ly} = \boldsymbol{b}$$

we have

$$\begin{pmatrix} 9 & 0 & 0 & 0 \\ 3 & 9 & 0 & 0 \\ 3 & -3 & 16 & 0 \\ 3 & -3 & 8 & 4 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 24 \\ 17 \\ 45 \\ 29 \end{pmatrix}$$

Using the forward substitution formula we get

$$\begin{aligned} &9y_1 = 24 \implies y_1 = \frac{8}{3} \\ &3y_1 + 9y_2 = 17 \implies y_2 = 1 \\ &3y_1 - 3y_2 + 16y_3 = 45 \implies y_3 = \frac{5}{2} \\ &3y_1 - 3y_2 + 8y_3 + 4y_4 = 29 \implies y_4 = 1. \end{aligned}$$

Thus $\boldsymbol{y} = \begin{pmatrix} \frac{8}{3} \\ 1 \\ \frac{5}{2} \\ 1 \end{pmatrix}$ and $\boldsymbol{Ux} = \boldsymbol{y}$ gives

$$\begin{pmatrix} 1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 1 & -\frac{1}{3} & -\frac{1}{3} \\ 0 & 0 & 1 & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \frac{8}{3} \\ 1 \\ \frac{5}{2} \\ 1 \end{pmatrix}.$$

- By using the backward substitution formula we get

$$\begin{aligned} &x_4 = 1 \\ &x_3 + \frac{1}{2}x_4 = \frac{5}{2} \implies x_3 = 2 \\ &x_2 - \frac{1}{3}x_3 - \frac{1}{3}x_4 = 1 \implies x_2 = 2 \\ &x_1 + \frac{1}{3}x_2 + \frac{1}{3}x_3 + \frac{1}{3}x_4 = \frac{8}{3} \implies x_1 = 1. \end{aligned}$$

Therefore, the required solution by Crout's method is

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

Cholesky Decomposition for Symmetric Matrices

Definition 3.2 Symmetric Matrix

A symmetric matrix is a square matrix that in which it's transpose is the matrix it self. Formally, matrix A is symmetric if

$$\boldsymbol{A}^T = \boldsymbol{A}.$$

Definition 3.3 positive definite symmetric Matrix

An $n \times n$ real symmetric matrix $\boldsymbol{A}$ is said to be **positive definite** if the scalar $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$ is positive for every non-zero column vector $\boldsymbol{x}$ of n real numbers.

The Cholesky decomposition of a positive-definite symmetric matrix A is a decomposition of the form

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T,$$

where $\boldsymbol{L}$ is a lower triangular matrix with real positive diagonal elements and $\boldsymbol{L}^T$ is the transpose of $\boldsymbol{L}$. The Cholesky decomposition is unique when $\boldsymbol{A}$ is positive definite; there is only one lower triangular matrix $\boldsymbol{L}$ with strictly positive diagonal entries such that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$. However, the decomposition need not be unique when $\boldsymbol{A}$ is positive semidefinite. The converse holds trivially: if $\boldsymbol{A}$ can be written as $\boldsymbol{L}\boldsymbol{L}^T$ for some lower triangular matrix L, then A is symmetric and positive definite.

The general Algorithm for Cholesky decomposition is given as follow:

$$\begin{aligned}
l_{11} &= \sqrt{a_{11}} \\
l_{i1} &= \frac{a_{i1}}{l_{11}} \quad \text{for } i = 2, 3, \cdots, n. \\
&\text{for } j = 2, 3, \cdots, n \\
l_{jj} &= \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2} \\
&\text{for } i = j+1, j+2, \cdots, n. \\
l_{ij} &= \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right),
\end{aligned} \tag{3.24}$$

Cholesky decomposition is evaluated column by column (starting from the first column) and, in each row, the elements are evaluated from top to bottom. That is, in each column the diagonal element is evaluated first using (2) (the elements above the diagonal are zero) and then the other elements in the same row are evaluated next using (3). This is carried out for each column starting from the first one.

Steps for solving $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A}$ is symmetric, using Cholesky decomposition is given below.

1. Factorize $\boldsymbol{A}$ in to $\boldsymbol{A} = \boldsymbol{LL}^T$.
2. Solve for $\boldsymbol{x}$
   (a) Forward substitution:
   $$\text{Solve for } \boldsymbol{y} \text{ using} \quad \boldsymbol{Ly} = \boldsymbol{b}$$
   (b) Back substitution:
   $$\text{Solve for } \boldsymbol{x} \text{ using} \quad \boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{y}$$

Example 3.8:

Solve the following system of equations using the Cholesky decomposition.

$$\begin{aligned} 4x_1 + 2x_2 + 14x_3 &= 14 \\ 2x_1 + 17x_2 - 5x_3 &= -101 \\ 14x_1 - 5x_2 + 83x_3 &= 155 \end{aligned}$$

Solution: The coefficient matrix is given by

$$\boldsymbol{A} = \begin{bmatrix} 4 & 2 & 14 \\ 2 & 17 & -5 \\ 14 & -5 & 83 \end{bmatrix}$$

In order to solve the above system first we need to get the cholesky decomposition of the coefficient matrix $\boldsymbol{A}$. Now, we use the formulas in Equation (3.24) determine the $\boldsymbol{L}L^T$ decomposition as follows. The computation of $\boldsymbol{L}$ is column by column.

Step-1 First column when $j = 1$:

The diagonal element l_{11} is given by

$$l_{11} = \sqrt{a_{11}} = \sqrt{4} = 2$$

The elements below the diagonal are computed as:

$$l_{i1} = \frac{a_{i1}}{l_{11}}, i = 2, 3$$

Thus, when $i = 2$ we have

$$l_{21} = \frac{a_{21}}{l_{11}} = \frac{2}{2} = 1$$

when $i = 3$

$$l_{31} = \frac{a_{31}}{l_{11}} = \frac{14}{2} = 7$$

Step-2 Compute the 2^{nd} column, i.e., $j = 2$:

The diagonal element l_{22} is given by

$$l_{22} = \sqrt{a_{22} - \sum_{k=1}^{2-1} l_{2k}^2} = \sqrt{17 - l_{21}^2} = \sqrt{17 - 1^2} = 4$$

The elements below l_{22} are computed as:

$$l_{i2} = \frac{1}{l_{22}}\left(a_{i2} - \sum_{k=1}^{2-1} l_{ik}l_{2k}\right), i = 3$$

when $i = 3$ we have

$$l_{32} = \frac{1}{l_{22}}(a_{32} - l_{31}l_{21}) = \frac{1}{4}(-5 - (7)(1)) = -3$$

Step-3: Compute the 3^{rd} column, i.e., $j = 3$ Here we only have one element to compute l_{33} and is given by

$$l_{33} = \sqrt{a_{33} - \sum_{k=1}^{3-1} l_{3k}^2} = \sqrt{a_{33} - (l_{31}^2 + l_{32}^2)} = \sqrt{83 - (49 + 9)} = 5$$

Collecting the elements of $\boldsymbol{L}$ we get

$$\boldsymbol{L} = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 4 & 0 \\ 7 & -3 & 5 \end{bmatrix}$$

Therefore, the Cholesky decomposition of $\boldsymbol{A}$ is given by

$$\boldsymbol{A} = \begin{bmatrix} 4 & 2 & 14 \\ 2 & 17 & -5 \\ 14 & -5 & 83 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 4 & 0 \\ 7 & -3 & 5 \end{bmatrix} \begin{bmatrix} 2 & 1 & 7 \\ 0 & 4 & -3 \\ 0 & 0 & 5 \end{bmatrix} = \boldsymbol{L}\boldsymbol{L}^T.$$

Exercise: Using the above decomposition solve the system given system

3.2.6 Tri-diagonal matrix method

All the methods described so far generally require about n^3 operations to obtain the solution. However, there is one frequently occurring system of equations for which extremely efficient solution algorithms exist. This system of equations is called tri-diagonal because there are never more than three unknowns in any equation and they can be arranged so that the coefficient matrix is composed of non-zero elements on the main diagonal and the diagonal immediately adjacent to either side. Thus such a system would have the form

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

where $\boldsymbol{A}$ is a Tridiagonal matrix given by

$$\boldsymbol{A} = \begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & \mathbf{0} & \\ & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & \mathbf{0} & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{bmatrix}.$$

Here, we can solve the given system using the two-stage strategy:

- **LU decomposition step.** $\boldsymbol{A}$ is factored or decomposed into lower $\boldsymbol{L}$ and upper

$\boldsymbol{U}$ triangular matrices of the form:

$$\boldsymbol{L} = \begin{bmatrix} 1 & & & & & \\ l_2 & 1 & & \mathbf{0} & & \\ & l_3 & 1 & & & \\ & & \ddots & \ddots & & \\ & \mathbf{0} & & l_{n-1} & 1 & \\ & & & & l_n & 1 \end{bmatrix}, \boldsymbol{U} = \begin{bmatrix} u_1 & v_1 & & & & \\ & u_2 & v_2 & & \mathbf{0} & \\ & & u_3 & v_3 & & \\ & & & \ddots & \ddots & \\ & \mathbf{0} & & & u_{n-1} & v_{n-1} \\ & & & & & u_n \end{bmatrix}$$

To determine the matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ we use the following formula:

$$b_1 = u_1 \implies u_1 = b_1$$
$$c_1 = v_1 \implies v_1 = c_1$$

for $k = 2, 3, \cdots, n$

$$\begin{aligned} a_k &= l_k u_{k-1} & &\implies l_k = \frac{a_k}{u_{k-1}}, \\ b_k &= l_k v_{k-1} + u_k & &\implies u_k = b_k - l_k v_{k-1}, \\ c_k &= v_k & &\implies v_k = c_k. \end{aligned}$$

- **Substitution step.** $\boldsymbol{L}$ and $\boldsymbol{U}$ are used to determine a solution $\boldsymbol{x}$ for a right-hand side column vector $\boldsymbol{b}$. This step itself consists of two steps. First, an intermediate vector $\boldsymbol{y}$ is generated by forward substitution. Then, the result is substituted back to solve for $\boldsymbol{x}$ by back substitution.

  - The forward-substitution step to solve

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b},$$

  where $\boldsymbol{y}$ is obtained using:

$$y_1 = b_1,$$

for $k = 2, \cdots n$

$$l_k y_{k-1} + y_k = b_k \implies y_k = b_k - l_k y_{k-1}$$

  - the back-substitution to solve

$$\boldsymbol{U}\boldsymbol{x} = \boldsymbol{y},$$

$$u_n x_n = y_n \implies x_n = \frac{y_n}{u_n},$$

for $k = n-1, n-2, \cdots, 1$:

$$u_k x_k + v_k x_{k+1} = y_k \implies x_k = \frac{y_k - v_k x_{k+1}}{u_k}$$

3.3 Indirect methods for SLE

Introduction

Direct solvers such as Gaussian Elimination and LU decomposition allow for efficient solving. In this section we introduce iterative solutions methods. The choice of a direct method or an indirect method is a combination of the efficiency of the method (and in general iterative methods are more efficient), the particular structure of the matrix system, a trade-off between compute time and memory, and the computer architecture being used. Iterative methods work by refining a guess to the solution and converging as quickly as possible from that guess to the actual solution. You may have met iterative methods previously in, for example, the general purpose solution of non-linear equations– such as bisection or Newton-Raphson techniques (along with their more advanced cousins). Iterative methods for linear systems have become a widespread and powerful tool for solving the most complex scientific and engineering problems and can be extremely effective, especially when starting from a good guess at the final solution – and often effort is expended in making that initial guess as good as possible and which will start you off close to the final solution and yield a more rapid convergence to the answer. Their only drawback is that they may not necessarily converge to a solution for a particular matrix system.

By this approach, we start with some initial guess solution, say $\boldsymbol{x}^{(0)}$; for solution $\boldsymbol{x}$ and generate an improved solution estimate $\boldsymbol{x}^{(k+1)}$ from the previous approximation $\boldsymbol{x}^{(k)}$: This method is a very effective for solving differential equations, integral equations and related problems. Let the residue vector r be defined as

$$r_i^{(k)} = b_i - \sum_{j=1}^{n} a_{ij} x_i^{(k)} \text{ for } i = 1, 2, \cdots n$$

i.e., $\boldsymbol{r}^{(k)} = \boldsymbol{b} - \boldsymbol{A}x^{(k)}$. The iteration sequence $\{\boldsymbol{x}^{(k)} : k := 0, 1, \cdots\}$ is terminated when some norm of the residue $||\boldsymbol{r}^{(k)}|| = ||\boldsymbol{A}\boldsymbol{x}^{(k)} - \boldsymbol{b}||$ becomes sufficiently small.

In this section, to begin with, some well known iterative schemes are presented. Their convergence analysis is presented next. In the derivations that follow, it is implicitly assumed that the diagonal elements of matrix A are non-zero, i.e. $a_{ii} \neq 0$: If this is not the case, simple row exchange is often sufficient to satisfy this condition.

3.3.1 Gauss Jacobi method

Consider the set of equations

$$\begin{aligned} 4x_1 - x_2 + x_3 &= 7 \\ 4x_1 - 8x_2 + x_3 &= -21 \\ -2x_1 + x_2 + 5x_3 &= 15 \end{aligned} \tag{3.25}$$

This could be written as

$$\begin{aligned} x_1 &= \frac{7 + x_2 - x_3}{4} \\ x_2 &= \frac{21 + 4x_1 + x_3}{8} \\ x_3 &= \frac{15 + 2x_1 - x_2}{5} \end{aligned}$$

And we could derive an iteration scheme which cycles through each of the values of x_1, x_2 , and x_3 in turn to refine an initial guess. If k is the k^{th} iteration, then

$$\begin{aligned} x_1^{(k+1)} &= \frac{7 + x_2^{(k)} - x_3^{(k)}}{4} \\ x_2^{(k+1)} &= \frac{21 + 4x_1^{(k)} + x_3^{(k)}}{8} \\ x_3^{(k+1)} &= \frac{15 + 2x_1^{(k)} - x_2^{(k)}}{5} \end{aligned}$$

Starting with an initial guess of $\boldsymbol{x}^{(0)} = [1, 2, 2]^T$ we obtain:

$$\begin{aligned} x_1^{(1)} &= \frac{7 + x_2^{(0)} - x_3^{(0)}}{4} = \frac{7 + 2 - 2}{4} = 1.75 \\ x_2^{(1)} &= \frac{21 + 4x_1^{(0)} + x_3^{(0)}}{8} = \frac{21 + 4 + 1}{8} = 3.375 \\ x_3^{(1)} &= \frac{15 + 2x_1^{(0)} - x_2^{(0)}}{5} = \frac{15 + 2 - 2}{5} = 3.000 \end{aligned}$$

In general we can write the Jacobi Method as:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j\neq i\\ j=1}}^{n} a_{ij}x_j^{(k)}\right) \tag{3.26}$$

The following table shows subsequent iterations of the above example

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	1.0	2.0	2.0
1	1.75	3.375	3.0
2	1.84375	3.875	3.025
3	1.9625	3.925	2.9625
$\vdots$	$\vdots$	$\vdots$	$\vdots$
19	2.0000	4.0000	3.0000

Stopping Criteria

The iterations are stopped when the absolute relative error is less than a respecified tolerance,ϵ, for all unknowns,i.e.,

$$\frac{|x_i^{(k+1)} - x_i^{(k)}|}{|x_i^{(k+1)}|} < \epsilon, \quad \text{for all } i = 1, 2, \cdots, n. \tag{3.27}$$

Convergence

It is a sufficient condition for the matrix to be *strictly diagonally dominant* for the Jacobi method to converge from any given starting vector.

Definition 3.4 Strictly diagonally dominant matrix

A matrix is said to be strictly diagonally dominant if

$$|a_{ii}| > \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| \qquad i = 1, 2, \cdots, n.$$

In the above example we have

$$\boldsymbol{A} = \begin{bmatrix} 4 & -1 & 1 \\ 4 & -8 & 1 \\ -2 & 1 & 5 \end{bmatrix}$$

and

$$\begin{aligned} \text{Row 1:} &\quad |a_{11} = |4| > |-1| + |1| \\ \text{Row 2:} &\quad |a_{22} = |-8| > |4| + |1| \\ \text{Row 3:} &\quad |a_{33} = |5| > |-2| + |1|. \end{aligned}$$

Thus, the matrix is strictly diagonally dominant and the Jacobi method will always converge for any given starting vector or initial guess.

Exercise 3.3:

Solve the linear equation $\boldsymbol{A}_2\boldsymbol{x} = \boldsymbol{b}_2$ using Jacobi Iteration, where

$$\boldsymbol{A}_2 = \begin{bmatrix} -2 & 1 & 5 \\ 4 & -1 & 1 \\ 4 & -8 & 1 \end{bmatrix}, \text{ and } \boldsymbol{b}_2 = \begin{bmatrix} 15 \\ -21 \\ 7 \end{bmatrix}$$

and explain what happens.

3.3.2 Gauss Seidel method

When matrix $\boldsymbol{A}$ is large, there is a practical difficulty with the Jacobi method. It is required to store all components of $\boldsymbol{x}^{(k)}$ in the computer memory (as a separate variables) until calculations of $\boldsymbol{x}^{(k+1)}$ is over. The Gauss-Seidel method overcomes this difficulty by using $x_i^{(k+1)}$ immediately in the next equation while computing $x_{i+1}^{(k+1)}$:This modification leads to the following set of equations

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)}\right) \\ x_2^{(k+1)} &= \frac{1}{a_{22}}\left(b_2 - \{a_{21}x_1^{(k+1)}\} - \{a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)}\}\right) \end{aligned}$$

$$x_3^{(k+1)} = \frac{1}{a_{33}}\left(b_3 - \{a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)}\} - \{a_{34}x_4^{(k)} + \cdots + a_{3n}x_n^{(k)}\}\right)$$

In general, for i'th element of $\boldsymbol{x}$, we have

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) \tag{3.28}$$

Now we are using the new values of x as soon as they are available at each iteration. Thus the equations in Equation (3.25) would become:

$$\begin{aligned} x_1^{(k+1)} &= \frac{7 + x_2^{(k)} - x_3^{(k)}}{4} \\ x_2^{(k+1)} &= \frac{21 + 4x_1^{(k+1)} + x_3^{(k)}}{8} \\ x_3^{(k+1)} &= \frac{15 + 2x_1^{(k+1)} - x_2^{(k+1)}}{5} \end{aligned}$$

Making this change and repeating the above makes the iteration to the solution $[2, 4, 3]^T$ take only 10 steps as per the table below.

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	1.0	2.0	2.0
1	1.75	3.75	2.95
2	1.95	3.96875	2.98625
3	1.995625	3.99609375	2.99903125
$\vdots$	$\vdots$	$\vdots$	$\vdots$
10	2.0000	4.0000	3.0000

The stopping criteria of the Gauss Seidel iteration is the same as Gauss Jacobi as given in Equation (3.27).

Convergence

The Gauss Seidel method always convergences if matrix $\boldsymbol{A}$ is either:

- *strictly diagonally dominant*
- *symmetric positive-definite.*

3.4 Systems of non-linear equations using Newton's method

In this section, we will now extend the Newton's Method that we discussed in Section 2.5 of Chapter 2 further to systems of many nonlinear equations. Consider the general system of n linear equations in n unknowns:

$$\begin{aligned} f_1(x_1, x_2, \cdots, x_n) &= 0 \\ f_2(x_1, x_2, \cdots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \cdots, x_n) &= 0, \end{aligned} \tag{3.29}$$

where $f_i \in \mathbb{R}^n$ are nonlinear functions of n-variables of x_i and $b_i \in \mathbb{R}$ are real numbers. For convenience we can think of $(x_1, x_2, x_3, \cdots, x_n)$ as a vector $\boldsymbol{x}$ and $(f_1, f_2, \cdots, f_n)$ as a vector-valued function $\boldsymbol{f}$. With this notation, the system of non-linear equations (3.29) may then be compactly written as

$$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0},$$

in which $\mathbf{0}$ denotes the zero vector $[0, 0, \cdots, 0]^t \in \mathbb{R}^n$. In order to find $\boldsymbol{x}$ such that $\boldsymbol{f}$ goes to $\mathbf{0}$, an initial estimate $\boldsymbol{x}^0$ is chosen, and Newton's iterative method for converging to the solution is used:

$$\boldsymbol{x}^1 = \boldsymbol{x}^0 - \boldsymbol{J}^{-1}(\boldsymbol{x}^0)\boldsymbol{f}(\boldsymbol{x}^0) \tag{3.30}$$

where $\boldsymbol{J}(\boldsymbol{x})$ is the Jacobian matrix of partial derivatives of $\boldsymbol{f}$ with respect to $\boldsymbol{x}$ given as

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\boldsymbol{x}) & \frac{\partial}{\partial x_2} f_1(\boldsymbol{x}) & \cdots & \frac{\partial}{\partial x_n} f_1(\boldsymbol{x}) \\ \frac{\partial}{\partial x_1} f_2(\boldsymbol{x}) & \frac{\partial}{\partial x_2} f_2(\boldsymbol{x}) & \cdots & \frac{\partial}{\partial x_n} f_2(\boldsymbol{x}) \\ & \vdots & & \\ \frac{\partial}{\partial x_1} f_n(\boldsymbol{x}) & \frac{\partial}{\partial x_2} f_n(\boldsymbol{x}) & \cdots & \frac{\partial}{\partial x_1} f_n(\boldsymbol{x}) \end{bmatrix}.$$

The formula in Equation (3.30) is the vector equivalent of the Newton's method formula we learned before. However, in practice we **never use the inverse of a matrix for computations**, so we cannot use this formula directly. Rather, we can do the following. First solve the equation

$$\boldsymbol{J}(\boldsymbol{x})\boldsymbol{h} = -\boldsymbol{f}(\boldsymbol{x}^0).$$

Since $\boldsymbol{J}(\boldsymbol{x}^0)$ is a known matrix and $\boldsymbol{f}(\boldsymbol{x}^0)$ is a known vector, this equation is just a system of linear equations, which can be solved efficiently and accurately. Once we have the solution vector $\boldsymbol{h}$ we can obtain our improved estimate $\boldsymbol{x}^1$ by

$$\boldsymbol{x}^1 = \boldsymbol{x}^0 + \boldsymbol{h}.$$

This, then, fully defines Newton's method for systems of non-linear equations as

1. Choose some initial guess $\boldsymbol{x}^0$
2. For all $k = 0, 1, \cdots$ **until convergence**
    a) Compute the Jacobian matrix $\boldsymbol{J}(\boldsymbol{x}) = D\boldsymbol{f}(\boldsymbol{x})$
    b) Solve the linear system $\boldsymbol{J}\boldsymbol{h}_k = -\boldsymbol{f}(\boldsymbol{x}_k)$ with respect to $\boldsymbol{h}_m$.
    c) set $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \boldsymbol{h}_k$.

The convergence criteria is given by

$$||\boldsymbol{x}^{k+1} - \boldsymbol{x}^k|| < \epsilon,$$

where ϵ is the given error tolerance.

Example 3.6:

Solve the following systems of nonlinear equations using the Newton's method

$$\begin{aligned} x_1 - x_2 + 1 &= 0 \\ x_1^2 + x_2^2 - 4 &= 0 \end{aligned}$$

3.5 Eigenvalue Problems

In this section we will discuss, in some detail, some iterative methods for finding single eigenvalue-eigenvector pairs (eigenpairs is a common term) of a given real matrix A; we will also give an overview of more powerful and general methods that are commonly used to find all the eigenpairs of a given real A.

3.6 Basic Properties of eigenvalues and eigenvectors

The algebraic eigenvalue problem is as follows: Given a matrix $A \in \mathbb{R}^{n \times n}$, find a nonzero vector $x \in \mathbb{R}^n$ and the scalar λ such that

$$\boldsymbol{Ax} = \lambda \boldsymbol{x}$$

Note that this says that the vector $\boldsymbol{Ax}$ is parallel to $\boldsymbol{x}$, with λ being an amplification factor, or gain. Note also that the above implies that

$$(\boldsymbol{A} - \boldsymbol{xI})\boldsymbol{x} = \boldsymbol{0},$$

that $\boldsymbol{A} - \lambda I$ is a singular matrix. Hence, $\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = \boldsymbol{0}$; it is easy to show that this determinant is a polynomial (of degree n) in λ, known as the **characteristic polynomial** of $\boldsymbol{A}$, $p(\lambda)$, so that the eigenvalues are the roots of a polynomial. Although this is not a good way to compute the eigenvalues, it does give us some insight into their properties. Thus, we know that an $n \times n$ matrix has n eigenvalues, that the eigenvalues can be repeated, and that a real matrix can have complex eigenvalues, but these must occur in conjugate pairs. We summarize these and a number of other basic eigenvalue properties in the following theorem, presented without proof.

Theorem 3.1

Basic Eigenvalue Properties Let $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ be given. Then we have the following:

1. There are exactly n eigenvalues, counting multiplicities; complex eigenvalues will occur in conjugate pairs.
2. Eigenvectors corresponding to distinct eigenvalues are linearly independent.
3. If an $n \times n$ matrix $\boldsymbol{A}$ has n independent eigenvectors, then there exists a nonsingular matrix $\boldsymbol{P}$ such that $\boldsymbol{P}^{-1}\boldsymbol{AP} = \boldsymbol{D}$ is diagonal and $\boldsymbol{A}$ is called diagonalizable. Moreover, the columns of $\boldsymbol{P}$ are the eigenvectors of A and the elements $d_{ii} = \lambda_i$ are the eigenvalues of $\boldsymbol{A}$.
4. If $\boldsymbol{A}$ is symmetric ($\boldsymbol{A} = \boldsymbol{A}^T$), then the eigenvalues are real and we can choose the eigenvectors to be real and orthogonal.

5. If $\boldsymbol{A}$ is symmetric, then there is an orthogonal matrix Q such that $\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{D}$ is diagonal, where the elements $d_{ii} = \lambda_i$, are the eigenvalues of $\boldsymbol{A}$.

6. If $\boldsymbol{A}$ is triangular, then the eigenvalues are the diagonal elements, $\lambda_i = a_{ii}$.

3.6.1 Power method for finding dominant eigenvalues

The power method is an iterative technique to find or locater the dominant eigenvalue of a given matrix and also computes the associated eigenvector.

Let $\boldsymbol{A}$ be an $n \times n$ matrix with eigenvalues $\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_n$ not necessary distinct that satisfy the relation

$$\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n.$$

The eigenvalue λ_1 which is the largest in magnitude, is known as *dominant eigenvalue* of the matrix $\boldsymbol{A}$. Furthermore, we assume that the associated eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \cdots, \boldsymbol{v}_n$ are *linearly independent*, and therefore *form a basis* for $\mathbb{R}^n$. It should be noted that not all matrices have eigenvalues and eigenvectors which satisfy the conditions we have assumed here.

Let $\boldsymbol{x}^0$ be a nonzero element of $\mathbb{R}^n$. Since the eigenvectors of $\boldsymbol{A}$ form a basis for $\mathbb{R}$ it follows that $\boldsymbol{x}^0$ can be written as a linear combination of $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \cdots, \boldsymbol{v}_n$, that is, there exists constants $\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_n$ such that

$$\boldsymbol{x}^0 = \alpha_1\boldsymbol{v}_1 + \alpha_2\boldsymbol{v}_2 + \alpha_3\boldsymbol{v}_3 + \cdots + \alpha_n\boldsymbol{v}_n$$

.

Next, construct the sequence $\{\boldsymbol{x}^m$ according to the rule $\boldsymbol{x}^m = \boldsymbol{A}\boldsymbol{x}^{m-1}$ form ≥ 1. By the direct calculation we find

$$\begin{aligned}
\boldsymbol{x}^1 &= \boldsymbol{A}\boldsymbol{x}^0 \\
&= \alpha_1(\boldsymbol{A}\boldsymbol{v}_1) + \alpha_2(\boldsymbol{A}\boldsymbol{v}_2) + \alpha_3(\boldsymbol{A}\boldsymbol{v}_3) + \cdots + \alpha_n(\boldsymbol{A}\boldsymbol{v}_n) \\
&= \alpha_1(\lambda_1\boldsymbol{v}_1) + \alpha_2(\lambda_2\boldsymbol{v}_2) + \alpha_3(\lambda_3\boldsymbol{v}_3) + \cdots + \alpha_n(\lambda_n\boldsymbol{v}_n) \\
\boldsymbol{x}^2 &= \boldsymbol{A}\boldsymbol{x}^1 = \boldsymbol{A}^2\boldsymbol{x}^0 \\
&= \alpha_1(\boldsymbol{A}^2\boldsymbol{v}_1) + \alpha_2(\boldsymbol{A}^2\boldsymbol{v}_2) + \alpha_3(A^2\boldsymbol{v}_3) + \cdots + \alpha_n(\boldsymbol{A}^2\boldsymbol{v}_n) \\
&= \alpha_1(\lambda_1^2\boldsymbol{v}_1) + \alpha_2(\lambda_2^2\boldsymbol{v}_2) + \alpha_3(\lambda_3^2\boldsymbol{v}_3) + \cdots + \alpha_n(\lambda_n^2\boldsymbol{v}_n)
\end{aligned}$$

and, in general

$$\begin{aligned}
\boldsymbol{x}^m &= \boldsymbol{A}\boldsymbol{x}^{m-1} = \cdots = \boldsymbol{A}^m\boldsymbol{x}^0 \\
&= \alpha_1(\boldsymbol{A}^m\boldsymbol{v}_1) + \alpha_2(\boldsymbol{A}^m\boldsymbol{v}_2) + \alpha_3(\boldsymbol{A}^m\boldsymbol{v}_3) + \cdots + \alpha_n(\boldsymbol{A}^m\boldsymbol{v}_n) \\
&= \alpha_1(\lambda_1^m\boldsymbol{v}_1) + \alpha_2(\lambda_2^m\boldsymbol{v}_2) + \alpha_3(\lambda_3^m\boldsymbol{v}_3) + \cdots + \alpha_n(\lambda_n^m\boldsymbol{v}_n)
\end{aligned}$$

In deriving these equations we have made repeated use of the relation $\boldsymbol{A}\boldsymbol{v}_j = \lambda_j\boldsymbol{v}_j$, which flows from the fact that $\boldsymbol{v}_j$ is an eigenvector associated with the eigenvalue λ_j.

Factoring λ_1^m from the right-hand side of the equation for $\boldsymbol{x}^m$ gives

$$\boldsymbol{x}^m = \lambda_1^m\left[\alpha_1 v_2 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right) + \alpha_3\left(\frac{\lambda_3}{\lambda_1}\right) + \cdots \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)\right]. \tag{3.31}$$

By assumption, $|\frac{\lambda_j}{\lambda_1}| < 1$ for each j, so $|\frac{\lambda_j}{\lambda_1}|^m \to 0$ as $m \to \infty$. It therefore follows that

$$\lim_{m\to\infty} \frac{\boldsymbol{x}^m}{\lambda_1^m} = \alpha_1\boldsymbol{v}_1.$$

Since any nonzero constant times and eigenvector is still an eigenvector associated with the same eigenvalue, we see that the scaled sequence $\{\boldsymbol{x}^{(m)}/\lambda_1^m\}$ converges to and eigenvector associated with the dominant eigenvalue provided $\alpha_1 \neq 0$. Furthermore, convergence towards the eigenvector is linear with asymptotic error constant $|\frac{\lambda_2}{\lambda_1}|$.

3.6.2 Approximated dominant eigenvalue

An approximation for the dominant eigenvalue of $\boldsymbol{A}$ can be obtained from the sequence $\{\boldsymbol{x}^{(m)}\}$ as follows. Let i be an index for which $x_i^{(m-1)} \neq 0$, and consider the ratio of the i^{th} element from the vector $\boldsymbol{x}^m$ to the i^{th} element from $\boldsymbol{x}^{(m-1)}$.

By equation (3.31),

$$\frac{x_i^{(m)}}{x_i^{(m-1)}} = \frac{\lambda_1^m \alpha_1 v_{1,i}[1 + O((\lambda_2/\lambda_1)^m)]}{\lambda_1^{m-1} \alpha_1 v_{1,i}[1 + O((\lambda_2/\lambda_1)^{m-1})]} = \lambda_1[1 + O((\lambda_2/\lambda_1)^{m-1})]$$

provided $v_{1,i} \neq 0$, where $v_{1,i}$ denotes the i^{th} element from the vector $\boldsymbol{v}_1$. Hence, the ratio $\frac{x_i^{(m)}}{x_i^{(m-1)}}$ converges towards the dominant eigenvalue, and the convergence is linear with asymptotic rate constant $|\lambda_2/\lambda_1|$.

To avoid overflow or underflow problems when calculating the sequence $\{\boldsymbol{x}^{(m)}$, it is common practice to scale the vectors $\boldsymbol{x}^{(m)}$ so that they all of unit length. Here, we will use the l_∞ norm to measure vector length. Thus, in a practical implementation of the power method, the vectors $\boldsymbol{x}^{(m)}$ would be computed in two steps: First multiply the previous vector by the matrix $\boldsymbol{A}$ and then scale the resulting vector to unit length.

```
Select $\boldsymbol{x}^(0) \neq 0, k = 1$,set $N_{max}, tol, \lambda^{(0)} = 0$;
$\boldsymbol{y}^{(k)} = \boldsymbol{A}\boldsymbol{x}^{(k-1)}, \lambda^{(k)} = y_p^{(k)}$, where $|y_p^{(k)} = \|\boldsymbol{y}^{(k)}\|_\infty$;
instructions;
if $|\lambda^{(k)} - \lambda^{(k-1)}| < tol \ or \ k \geq N_{max}$ then
    Stop;
else
    set $\boldsymbol{x}^{(k)} = \frac{\boldsymbol{y}^{(k)}}{\lambda^{(k)}}$;
    Go to step 2;
end
```

Algorithm 1: Power method algorithms

Theorem 3.2 Determining an Eigenvalue from and Eigenvector

If $\boldsymbol{x}$ is and eigenvector of a matrix $\boldsymbol{A}$, then its corresponding eigenvalue is given by

$$\lambda = \frac{A\boldsymbol{x} \cdot \boldsymbol{x}}{\boldsymbol{x} \cdot \boldsymbol{x}}$$

This quotient is call the **Rayleigh quotient**.

Example 3.1

Approximate dominant eigenvalue using the power method for the matrix

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

Use $\boldsymbol{x}^0 = (1, 1, 1)$ as the initial approximation and $tol = 0.001$

Solution:

Step 1: First iteration of the power method produces

$$\boldsymbol{y}^{(1)} = \boldsymbol{A}\boldsymbol{x}^{(0)} = \begin{bmatrix} 1 & 2 & 0 \\ -2 & 1 & 2 \\ 1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 5 \end{bmatrix}$$

Thus, $\lambda^{(1)} = \|\boldsymbol{y}^1\|_\infty = 5$ and hence we have

$$\boldsymbol{x}^{(1)} = \frac{\boldsymbol{y}^{(1)}}{\lambda^{(1)}} = \begin{bmatrix} 0.60 \\ 0.20 \\ 1.00 \end{bmatrix}$$

Since $|\lambda^{(1)} - \lambda^{(0)}| = 5 \geq tol$ we need to repeat the power method.

Step 2: Second iteration of the power method produces

$$\boldsymbol{y}^{(2)} = \boldsymbol{A}\boldsymbol{x}^{(1)} = \begin{bmatrix} 1 & 2 & 0 \\ -2 & 1 & 2 \\ 1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 0.60 \\ 0.20 \\ 1.00 \end{bmatrix} = \begin{bmatrix} 1.00 \\ 1.00 \\ 2.20 \end{bmatrix}$$

Thus, $\lambda^{(2)} = \|\boldsymbol{y}^2\|_\infty = 2.20$ and hence we have

$$\boldsymbol{x}^{(2)} = \frac{\boldsymbol{y}^{(2)}}{\lambda^{(2)}} = \begin{bmatrix} 0.45 \\ 0.45 \\ 1.00 \end{bmatrix}$$

Since $|\lambda^{(2)} - \lambda^{(1)}| = |2.20 - 5| = 2.80 \geq tol$ we need to repeat the power method.

Continuing this process, you obtain the sequence of approximations shown in Table ??.

$\boldsymbol{x}^{(0)}$	$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$	$\boldsymbol{x}^{(4)}$	$\boldsymbol{x}^{(5)}$	$\boldsymbol{x}^{(6)}$	$\boldsymbol{x}^{(7)}$
$\begin{bmatrix}1.00\\1.00\\1.00\end{bmatrix}$	$\begin{bmatrix}0.60\\0.20\\1.00\end{bmatrix}$	$\begin{bmatrix}0.45\\0.45\\1.00\end{bmatrix}$	$\begin{bmatrix}0.48\\0.55\\1.00\end{bmatrix}$	$\begin{bmatrix}0.51\\0.51\\1.00\end{bmatrix}$	$\begin{bmatrix}0.50\\0.49\\1.00\end{bmatrix}$	$\begin{bmatrix}0.50\\0.50\\1.00\end{bmatrix}$	$\begin{bmatrix}0.50\\0.50\\1.00\end{bmatrix}$
$\lambda^{(0)}$	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$	$\lambda^{(5)}$	$\lambda^{(6)}$	$\lambda^{(7)}$
0	5.0	2.20	2.82	3.13	3.02	2.99	3.00

Then the dominant eigenvalue can be obtained using the Rayleigh quotient after the convergence of the power method. Hence

$$\lambda_1 = \frac{A\boldsymbol{x}^{(7)} \cdot \boldsymbol{x}^{(7)}}{\boldsymbol{x}^{(7)} \cdot \boldsymbol{x}^{(7)}} = 3.0$$

Exercise 3.1

Determine the largest eigenvalue and the corresponding eigenvector of the matrix below using power method

$$\begin{bmatrix}5 & 4\\1 & 2\end{bmatrix} \quad \text{using } \boldsymbol{x}^0 = \begin{bmatrix}1\\0\end{bmatrix} \quad \text{as an initial guess.}$$

Answer: After the 6^{th} iteration you will find that $\lambda^{(6)} = \lambda^{(6)} = 6$ and $\boldsymbol{x}^{(6)} = \boldsymbol{x}^{(5)} = \begin{bmatrix}1\\0.25\end{bmatrix}$.

3.6.3 Inverse power method

Suppose λ is an eigenvalue of matrix $\boldsymbol{A}$, then:

$$\begin{aligned}
& \boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x} \\
\iff\quad & \boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{A}^{-1}\boldsymbol{x} \\
\iff\quad & \boldsymbol{x} = \lambda\boldsymbol{A}^{-1}\boldsymbol{x} \\
\iff\quad & \boldsymbol{A}^{-1}\boldsymbol{x} = \frac{1}{\lambda}\boldsymbol{A}^{-1}\boldsymbol{x} \\
\iff\quad & \frac{1}{\lambda} \text{ is eigenvalue of } \boldsymbol{A}^{-1}.
\end{aligned}$$

Thus if λ is the eigenvalue of $\boldsymbol{A}$ and $\boldsymbol{x}$ is the corresponding eigenvector, then $\frac{1}{\lambda}$ is an eigenvalue of corresponding to the same eigenvector. Hence the reciprocal of the largest eigenvalue of $\boldsymbol{A}^{-1}$ is the smallest eigenvalue of $\boldsymbol{A}$. Therefore to find the smallest eigenvalue of $\boldsymbol{A}$, we apply power method on $\boldsymbol{A}^{-1}$. This process is called **inverse power method.**

Example 3.2

Perform 3 iterations of the inverse power method to obtain the smallest eigenvalue of $\boldsymbol{A} = \begin{bmatrix} 1 & 6 & 1 \\ 1 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$ using $\boldsymbol{x}^{(0)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ as an initial guess.

Chapter 4

Interpolation

4.1 Finite differences

Different types of finite difference operators are used in Numerical Analysis, viz. shift (E), average(mean) (μ),forward difference $(\triangle)$, backward difference $(\triangledown)$ and central difference (δ) are discussed in this chapter.

When a function is known explicitly, it is easy to calculate the value (or values) of $f(x)$, corresponding to a fixed given value . However, when the explicit form of the function is not known, it is possible to obtain an approximate value of the function up to a desired level of accuracy with the help of finite differences. The calculus of finite differences deals with the changes that take place in the value of a function due to finite changes in the independent variable.

Definition 4.1 Finite Differences

If $x_0, x_1, x_2 \cdots, x_n$ be given set of observations and let $y_0 = f(x_0), y_1 = f(x_1), y_2 = f(x_2), \cdots, y_n = f(x_n)$ be their corresponding values for the curve $y = f(x)$, $y_1 - y_0, y_2 - y_1, \cdots, y_n - y_{n-1}$ is called as **finite difference.**

Different types of finite difference operators are defined, among them forward difference, backward difference and central difference operators are widely used for equally spaced data points.

Let the function $y = f(x)$, x begin the independent variable and y a dependent variable, be defined on the closed interval $[a, b]$ and let $x_0, x_1, \cdots, x_n$ be the n discrete values of

x on the given interval. Assumed that these values are equidistant, i.e. $x_i = x_0 + ih$, $i = 0, 1, 2, \cdots, n$; h is a suitable real number called the difference of the interval or step size. When $x = x_i$, the value of y is denoted by y_i and is defined by $y_i = f(x_i)$. The values of independent variable x are called **arguments** and the dependent variable y are called **entries**.

Definition 4.2 **Operator**

An **operator** is a symbol which is operated with a function then it transforms the function or we can say that the operator maps one sequence to other.

4.1.1 Shift operators

If the shift operator E is operating on y_i, the y value is shifted down to the next provided y value.

Definition 4.3 **Shift Operator**

Let $y = f(x)$ be a function x, and let x takes the consecutive values $x, x+h, x+2h, etc.$ We then define an operator having the property

$$Ef(x) = f(x+h).$$

Thus, when E operates on $f(x)$, the result is the next value of the function. Here, E is called the **shift operator.**

If we apply the shift operator twice on $f(x)$, we get

$$\begin{aligned} E^2 f(x) &= E(Ef(x)) \\ &= Ef(x+h) \\ &= f(x+2h) \end{aligned}$$

Thus, in general, if we apply the operator E n times, we get

$$E^n f(x) = f(x+nh),$$

or

$$E^n y_x = y(x+nh),$$

or using the index i

$$E^n y_i = y_{i+n}$$

For example

$$Ey_0 = y_1, \qquad E^2 y_0 = Ey_1 = y_2, \qquad E^4 y_0 = y_4, \cdots, E^2 y_2 = y_4.$$

Definition 4.4 Inverse Shift Operator

The inverse operator E^{-1} is defined as

$$E^{-1} f(x) = f(x-h), \qquad \text{or} \quad E^{-1} y_x = y_{x-h} \qquad \text{or} \quad E^{-1} y_i = y_{i-1}$$

Similarly,

$$E^{-n} f(x) = f(x - nh).$$

4.1.2 Averaging Operator

The averaging operator μ is defined as

$$\mu f(x) = \frac{1}{2}\left[f(x + \frac{1}{2}h) + f(x - \frac{1}{2}h)\right]$$

4.1.3 Differential Operator

The averaging operator D is defined as

$$\begin{aligned} Df(x) &= \frac{\mathrm{d}}{\mathrm{d}x} f(x) = f'(x), \\ D^2 f(x) &= \frac{\mathrm{d}^2}{\mathrm{d}^2 x} f(x) = f''(x). \end{aligned} \tag{4.1}$$

4.1.4 Forward difference operator

Definition 4.5 Forward difference Operator

The forward difference is denoted by $(\triangle)$ and is defined by

$$\triangle f(x) = f(x+h) - f(x). \tag{4.2}$$

When $x = x_i$ then from above equation

$$\triangle f(x_i) = f(x_i + h) - f(x_i), \text{ i.e., } \triangle y_i = y_{i+1} - y_i, i = 0, 1, 2, \cdots, n-1.$$

In particular,

$$\triangle y_0 = y_1 - y_0, \quad \triangle y_1 = y_2 - y_1, \cdots, \triangle y_{n-1} = y_n - y_{n-1}.$$

These are called **First order Forward Differences**. The differences of the first order forward differences are called **Second order Forward Differences** and are denoted by $\triangle^2 y_0, \triangle^2 y_1, \triangle^2 y_2, \cdots, \triangle^2 y_n$. In particulat two second order differences are

$$\triangle^2 y_0 = \triangle y_1 - \triangle y_0 = (y_2 - y_1) - (y_1 - y_0) = y_2 - 2y_1 + y_0.$$
$$\triangle^2 y_1 = \triangle y_2 - \triangle y_1 = (y_3 - y_2) - (y_2 - y_1) = y_3 - 2y_2 + y_1.$$

The third order forward differences are also defined in similar manner, i.e.

$$\begin{aligned}
\triangle^3 y_0 &= \triangle^2 y_1 - \triangle^2 y_0 \\
&= (y_3 - 2y_2 + y_1) - (y_2 - 2y_1 + y_0.) \\
&= y_3 - 3y_2 + 3y_1 - y_0. \\
&= y_3 + (-1)^1 \binom{3}{2} y_2 + (-1)^2 \binom{3}{1} y_1 + (-1)^3 y_0 \binom{3}{0} \\
\triangle^3 y_1 &= y_4 - 3y_3 + 3y_2 - y_1, \\
&= y_4 + (-1)^1 \binom{3}{2} y_3 + (-1)^2 \binom{3}{1} + (-1)^3 y_1 \binom{3}{0},
\end{aligned}$$

where the combination $\binom{n}{k} = \dfrac{n!}{k!(n-k)!}$.

In general, higher order differences can be defined as follows

$$\triangle^n f(x) = \triangle\left[\triangle^{n-1} f(x)\right], \quad \text{i.e.,} \quad \triangle^n\, y_i = \triangle\left[\triangle^{n-1} y_i\right], \quad n = 1, 2, \cdots.$$

Or

$$\triangle^n y_i = y_{n+i} + \sum_{k=1}^{n}\left((-1)^k \binom{n}{n-k} y_{n+i-k}\right). \tag{4.3}$$

It must be remembered that $\triangle^0 \equiv$ identity operator, i.e. $\triangle^0 f(x) = f(x)$ and $\triangle^1 \equiv \triangle$.

Example 4.1

Find $\triangle^4 y_3$?

Solution: Here, we can either use the formula in Equation (4.3) or successive application of the forward difference operator on y_3 four times.

Using the formula, we have $n = 4$ and $i = 3$ hence

$$\begin{aligned}
\triangle^4 y_3 &= y_{4+3} + \sum_{k=1}^{4}\left((-1)^k \binom{4}{4-k} y_{4+3-k}\right) \\
&= y_7 + (-1)^1 \binom{4}{3} y_6 + (-1)^2 \binom{4}{2} y_5 + (-1)^3 \binom{4}{1} y_4 + (-1)^4 \binom{4}{0} y_3 \\
&= y_7 - 4y_6 + 6y_5 - 4y_4 + 1y_3.
\end{aligned}$$

Applying the forward difference operator 4 times on y_3 yields

$$\begin{aligned}
\triangle^4 y_3 &= \triangle^3 y_4 - \triangle^3 y_3 \\
&= \left(\triangle^2 y_5 - \triangle^2 y_4\right) - \left(\triangle^2 y_4 - \triangle^2 y_3\right) \\
&= [(\triangle y_6 - \triangle y_5) - (\triangle y_5 - \triangle y_4)] - [(\triangle y_5 - \triangle y_4) - (\triangle y_4 - \triangle y_3)] \\
&= [((y_7 - y_6) - (y_6 - y_5)) - ((y_6 - y_5) - (y_5 - y_4))] \\
&\quad - [((y_6 - y_5) - (y_5 - y_4)) - ((y_5 - y_4) - (y_4 - y_3))] \\
&= [(y_7 - 2y_6 + y_5) - (y_6 - 2y_5 + y_4)] - [(y_6 - 2y_5 + y_4) - (y_5 - 2y_4 + y_3)] \\
&= [y_7 - 3y_6 + 3y_5 - y_4] - [y_6 - 3y_5 + 3y_4 - y_3] \\
&= y_7 - 4y_6 + 6y_5 - 4y_4 + y_3.
\end{aligned}$$

$$\therefore \quad \triangle^4 y_3 = y_7 - 4y_6 + 6y_5 - 4y_4 + y_3.$$

All the forward differences can be represented in a tabular form, called **the forward difference** or **diagonal difference table**. Let $x_0, x_1, \cdots, x_4$ be four arguments. All the forwarded differences of these arguments are shown in Table 4.1 bellow.

x	y	$\triangle$	$\triangle^2$	$\triangle^3$	$\triangle^4$
x_0	y_0				
		$\triangle y_0$			
x_1	y_1		$\triangle^2 y_0$		
		$\triangle y_1$		$\triangle^3 y_0$	
x_2	y_2		$\triangle^2 y_1$		$\triangle^4 y_0$
		$\triangle y_2$		$\triangle^3 y_1$	
x_3	y_3		$\triangle^2 y_2$		
		$\triangle y_3$			
x_4	y_4				

Table 4.1: Forward difference table.

The first entry, i.e., y_0 is called **leading term** and $\triangle y_0, \triangle^2 y_0, \triangle^3 y_0, \cdots$ are called the **leading differences**.

Error propagation in a difference table

If any entry of the difference table is has an error, then this error spread over the table in convex manner. The propagation of error in a difference table is illustrated in Table 4.2. Let us assumed that y_3 has an error and the amount of the error be ϵ.

x	y	$\triangle$	$\triangle^2$	$\triangle^3$	$\triangle^4$	$\triangle^5$	$\triangle^6$
x_0	y_0						
		$\triangle y_0$					
x_1	y_1		$\triangle^2 y_0$				
		$\triangle y_1$		$\triangle^3 y_0 + \epsilon$			
x_2	y_2		$\triangle^2 y_1 + \epsilon$		$\triangle^4 y_0 - 4\epsilon$		
		$\triangle y_2 + \epsilon$		$\triangle^3 y_1 - 3\epsilon$		$\triangle^5 y_0 + 10\epsilon$	
x_3	$y_3 + \epsilon$		$\triangle^2 y_2 - 2\epsilon$		$\triangle^4 y_1 + 6\epsilon$		$\triangle^6 y_0 - 20\epsilon$
		$\triangle y_3 - \epsilon$		$\triangle^3 y_2 + 3\epsilon$		$\triangle^5 y_1 - 10\epsilon$	
x_4	y_4		$\triangle^2 y_3 + \epsilon$		$\triangle^4 y_2 - 4\epsilon$		
		$\triangle y_4$		$\triangle^3 y_3 - \epsilon$			
x_5	y_5		$\triangle^2 y_4$				
		$\triangle y_5$					
x_6	y_6						

Table 4.2: Error propagation in a finite difference table.

Following observations are noted from Table 4.2.

(i) The error increases with the order of the differences.

(ii) The error is maximum (in magnitude) along the horizontal line through the erroneous tabulated value.

(iii) In the k^{th} difference column, the coefficients of errors are the binomial coefficients in the expansion of $(1-x)^k$. In particular, the errors in the second difference column are $\epsilon, -2\epsilon, \epsilon$, in the third difference column these are $\epsilon, -3\epsilon, 3\epsilon, -\epsilon$, and so on.

(iv) The algebraic sum of errors in any complete column is zero.

Example 4.2

Construct a forward diagonal difference table for the following set of values:

x_i	0	2	4	6	8	10	12	14
y_i	625	81	1	1	81	625	2401	65611

Solution: The forward difference table is given by

x_i	y_i	$\triangle$	$\triangle^2$	$\triangle^3$	$\triangle^4$	$\triangle^5$
0	625					
		-544				
2	81		464			
		-80		-384		
4	1		80		384	
		0		0		0
6	1		80		384	
		80		384		0
8	81		464		384	
		544		768		0
10	625		1232		384	
		1776		1152		
12	2401		2384			
		4160				
14	65611					

Note: If $f(x)$ is a polynomial of degree n in x, then $\triangle^n f(x)$ is a constant and is zero. Conversely, if the $(n+1)^{th}$ difference of a polynomial is zero, then the degree of the polynomial is less or equal to n. **Example:** Let $f(x) = x^2 + 8x - 5$, then

$$\begin{aligned}\triangle f(x) &= f(x+h) - f(x)\\ &= \left[(x+h)^2 + 8(x+h) - 5\right] - \left[x^2 + 8x - 5\right]\\ &= 2xh + h^2 + 8h\end{aligned}$$

Now,

$$\begin{aligned}\triangle^2 f(x) &= \triangle f(x+h) - \triangle f(x)\\ &= \left[2h(x+h) + h^2 + 8h\right] - \left[2hx + h^2 + 8h\right]\\ &= 2h^2, \quad \text{which is a constant.}\end{aligned}$$

Hence,

$$\begin{aligned}\triangle^3 f(x) &= \triangle^2 f(x+h) - \triangle^2 f(x)\\ &= 2h^2 - 2h^2 = 0.\end{aligned}$$

4.1.5 Backward difference operator

Definition 4.6 Backward Difference Operator

The symbol $\bigtriangledown$ is used to represent **backward difference operator**. The backward difference operator is defined as

$$\bigtriangledown f(x) = f(x) - f(x-h). \tag{4.4}$$

Please note that the backward difference of $f(x+h)$ is same as the forward difference of $f(x)$, that is

$$\bigtriangledown f(x+h) = \triangle f(x).$$

When $x = x_i$, the above relation reduces to

$$\bigtriangledown y_i = y_i - y_{i-1}, \quad i = n, n-1, \cdots, 1.$$

In particular,

$$\bigtriangledown y_1 = y_1 - y_0, \bigtriangledown y_2 = y_2 - y_1, \cdots, \bigtriangledown y_n = y_n - y_{n-1},$$

These are called the **first order backward differences.** The second order backward differences are denoted by $\bigtriangledown^2 y_2, \bigtriangledown^2 y_3, \cdots, \bigtriangledown^2 y_n$. First three second order backward differences are

$$\begin{aligned}
\nabla^2 y_2 &= \nabla(\nabla y_2) = \nabla(y_2 - y_1)\\
&= \nabla y_2 - \nabla y_1\\
&= (y_2 - y_1) - (y_1 - y_0)\\
&= y_2 - 2y_1 + y_0,\\
\nabla^2 y_3 &= y_3 - 2y_2 + y_1, \quad \text{and}\\
\nabla^2 y_4 &= y_4 - 2y_3 + y_2.
\end{aligned}$$

The other second order differences can be obtained in similar manner.

In general

$$\nabla^k y_i = \nabla^{k-1} y_i - \nabla^{k-1} y_{i-1}, \quad i = n, n-1, \cdots, k,$$

where $\nabla^0 y_i = y_i$, $\nabla^1 y_i = \nabla y_i$. Like forward differences, these backward differences can be written in a tabular form, called backward difference or horizontal difference table. All backward difference table for the arguments $x_0, x_1, \cdots, x_4$ are shown in Table 4.3.

x	y	∇	∇^2	∇^3	∇^4
x_0	y_0				
		∇y_1			
x_1	y_1		$\nabla^2 y_2$		
		∇y_2		$\nabla^3 y_3$	
x_2	y_2		$\nabla^2 y_3$		$\nabla^4 y_4$
		∇y_3		$\nabla^3 y_4$	
x_3	y_3		$\nabla^2 y_4$		
		∇y_4			
x_4	y_4				

Table 4.3: Backward difference table.

It is observed from the forward and backward difference tables that for a given table of values both the tables are same. Practically, there are no differences among the values of the tables, but, theoretically they have separate significant which will be discussed in the next chapter.

4.1.6 Central difference operator

There is another kind of finite difference operator known as **central difference operator**.

Definition 4.7 Central Difference Operator

The **central difference operator** is denoted by the symbol δ and is defined by

$$\delta f(x) = f(x + \frac{h}{2}) - f(x - \frac{h}{2}). \tag{4.5}$$

When $x = x_i$, then the first order central difference, in terms of ordinates is

$$\delta y_i = y_{i+\frac{1}{2}} - y_{i-\frac{1}{2}}$$

where $y_{i+\frac{1}{2}} = f(x_i + \frac{h}{2})$ and $y_{i-\frac{1}{2}} = f(x_i - \frac{h}{2})$. In particular,

$$\delta y_{\frac{1}{2}} = y_1 - y_0, \quad \delta y_{\frac{3}{2}} = y_2 - y_1, \cdots, \delta y_{n-\frac{1}{2}} = y_n - y_{n-1}.$$

The second order central differences are

$$\begin{aligned}\delta^2 y_i &= \delta y_{i+\frac{1}{2}} - \delta y_{i-\frac{1}{2}} \\ &= (y_{i+1} - y_i) - (y_i - y_{i-1}) \\ &= y_{i+1} - 2y_i + y_{i-1}.\end{aligned}$$

In general,

$$\delta^n y_i = \delta^{n-1} y_{i+\frac{1}{2}} - \delta^{n-1} y_{i-\frac{1}{2}}.$$

All central differences for the five arguments $x_0, x_1, \cdots, x_4$ is shown in Table 4.4.

x	y	δ	δ^2	δ^3	δ^4
x_0	y_0				
		$\delta y_{1/2}$			
x_1	y_1		$\delta^2 y_1$		
		$\delta y_{3/2}$		$\delta^3 y_{3/2}$	
x_2	y_2		$\delta^2 y_2$		$\delta^4 y_2$
		$\delta y_{5/2}$		$\delta^3 y_{5/2}$	
x_3	y_3		$\delta^2 y_3$		
		$\delta y_{7/2}$			
x_4	y_4				

Table 4.4: Central difference table.

It may be observed that all odd (even) order differences have fraction suffices (integral suffices), respectively.

4.1.7 Relations between operators

Lot of useful and interesting results can be derived among the operators discussed above. First of all, we determine the relation between forward and backward difference operators.

$\triangle$ and $\bigtriangledown$

$$\begin{aligned}
\triangle y_i &= y_{i+1} - y_i = \bigtriangledown y_{i+1}.\\
\triangle^2 y_i &= \triangle y_{i+1} - \triangle y_i\\
&= y_{i+2} - 2y_i + y_{i-1}\\
&= \bigtriangledown y_{i+2}.
\end{aligned}$$

In general

$$\boxed{\triangle^n\, y_i = \bigtriangledown^n y_{i+n}, \quad i = 0, 1, 2, \cdots \qquad (4.6)}$$

$\triangle$ and E

There is a good relation between E and $\triangle$ operators.

$$\begin{aligned}
\triangle f(x) &= f(x+h) - f(x)\\
&= Ef(x) - f(x) \qquad (4.7)\\
&= (E-1)f(x).
\end{aligned}$$

From this relation one can conclude that the operators $\triangle$ and $E-1$ are equivalent. That is,

$$\triangle \equiv E - 1, \qquad (4.8)$$

or

$$E \equiv \triangle + 1, \qquad (4.9)$$

The expression for higher order forward differences in terms of function values can be derived as per following way:

$$\begin{aligned}\triangle^3 y_i &= (E-1)^3 y_i \\ &= (E^3 - 3E^2 + 3E - 1)y_i = y_3 - 3y_2 + 3y_1 - y_0.\end{aligned}$$

$\bigtriangledown$ and E

The relation between $\bigtriangledown$ and E operators is derived below:

$$\begin{aligned}\bigtriangledown f(x) &= f(x) - f(x-h) \\ &= f(x) - E^{-1} f(x) \\ &= (1 - E^{-1}) f(x).\end{aligned}$$

That is,

$$\boxed{\bigtriangledown \equiv 1 - E^{-1}, \qquad (4.10)}$$

δ and E

The relation between the operators δ and E is given below:

$$\begin{aligned}\delta f(x) &= f(x + \frac{1}{2}) - f(x - \frac{h}{2}) \\ &= E^{\frac{1}{2}} f(x) - E^{-\frac{1}{2}} f(x) \\ &= \left(E^{\frac{1}{2}} - E^{-\frac{1}{2}}\right) f(x).\end{aligned}$$

That is

$$\boxed{\delta \equiv E^{\frac{1}{2}} - E^{-\frac{1}{2}}. \qquad (4.11)}$$

Every operator defined earlier can be expressed in terms of other operator(s). Few more relations among the operators $\triangle, \bigtriangledown, E$ and δ are given in the following.

	E	$\triangle$	∇	δ	hD
E	E	$\triangle+1$	$(1-\nabla)^{-1}$	$1+\frac{\delta^2}{2}+\delta\sqrt{1+\frac{\delta^2}{4}}$	e^{hD}
$\triangle$	$E-1$	$\triangle$	$(1-\nabla)^{-1}-1$	$\frac{\delta^2}{2}+\delta\sqrt{1+\frac{\delta^2}{4}}$	$e^{hD}-1$
∇	$1-E^{-1}$	$1-(1+\triangle)^{-1}$	∇	$-\frac{\delta^2}{2}+\delta\sqrt{1+\frac{\delta^2}{4}}$	1- e^{-hD}
δ	$E^{\frac{1}{2}}-E^{-\frac{1}{2}}$	$\triangle(1+\triangle)^{-1/2}$	$\nabla(1-\nabla)^{-1/2}$	δ	$2\sinh(hD/2)$
hD	$\log E$	$\log(1+\triangle)$	$-\log(1-\nabla)$	$-2\sinh^{-1}(\delta/2)$	hD

Table 4.5: Relationship between the operators

4.2 Interpolations

An interpolation task usually involves a given set of data points: where the values y_i can,

x_i	x_0	x_1	$\cdots$	x_n
$f(x_i)$	y_0	y_1	$\cdots$	y_n

for example, be the result of some physical measurement or they can come from a long numerical calculation. Thus we know the value of the underlying function $f(x)$ at the set of points $\{x_i\}$, and we want to find an analytic expression for f. In practice, often we can measure a physical process or quantity (e.g., temperature) at a number of points (e.g., time instants for temperature), but we do not have an analytic expression for the process that would let us calculate its value at an arbitrary point. Interpolation provides a simple and good way of estimating the analytic expression, essentially a function, over the range spanned by the measured points.

In interpolation, the task is to estimate $f(x)$ for arbitrary x that lies between the smallest and the largest x_i. If x is outside the range of the x_i's, then the task is called **extrapolation**, which is considerably more hazardous.

Definition 4.8 **Interpolation and extrapolation**

Suppose that the function $f(x)$ is known at $(n+1)$ points $(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)$ where the pivotal points x_i spread out over the interval $[a, b]$ satisfy $a = x_0 < x_1 < \cdots < x_n = b$ and $y_i = f(x_i)$ then finding the value of the function at any non-tabular point $x_s (x_0 < x_s < x_n)$ is called **interpolation**.

Interpolation is done by approximating the required function using simpler functions such as, polynomials. Polynomial approximations assume the data as exact at the $(n+1)$ tabular points and generate an n^{th} degree polynomial passing through these $(n+1)$ points. However, if the given data has some errors then these errors also will reflect in the corresponding approximated function. More accurate approximations can be done using Splines and Chebicheve, Legender and Hermite polynomials but polynomials of degree n or less passing through $(n+1)$ points are easy to develop and useful in understanding numerical differentiation and numerical integral. Hence the present chapter is devoted to developing and using polynomial interpolation formulae to the required functions.

Definition 4.9

The points $x_0, \cdots, x_n$ are called the **interpolation points**. The property of "passing through these points" is referred to as **interpolating the data** or called **interpolation condition**. The function that interpolates the data is an **interpolant** or an **interpolating polynomial** (or whatever function is being used).

The purpose of interpolation are:

1. Replace a set of data points $\{(x_i, y_i)\}$ with a function given analytically. Here we have several aspects
   - Given a set of data points $\{(x_i, y_i)\}$, find a curve passing thru these points that is "pleasing to the eye". In fact, this is what is done continually with computer graphics. How do we connect a set of points to make a smooth curve? Connecting them with straight line segments will often give a curve with many corners, whereas what was intended was a smooth curve.
   - We may want to take function values $f(x)$ given in a table for selected values of x, often equally spaced, and extend the function to values of x not in the table. For example, given numbers from a table of logarithms, estimate the logarithm of a number x not in the table.
   - The data may be from a known class of functions. Interpolation is then used to find the member of this class of functions that agrees with the given data. For example, data may be generated from functions of the form

     $$p(x) = a_0 + a_1 e^x + a_2 x^{2x} + \cdots + a_n e^{nx}$$

     Then we need to find the coefficients $\{a_j\}$ based on the given data values.
2. Approximate functions with simpler ones, usually polynomials or 'piecewise poly-

nomials'. Here, the interpolation is to approximate the function $f(x)$ by a simpler function, perhaps to make it easier to integrate or differentiate $f(x)$. That will be the primary reason for studying interpolation in this course and the application will be discussed in Chapter 6. As an example of why this is important, consider the problem of evaluating

$$\int_0^1 \frac{\mathrm{d}x}{1+x^{10}}$$

This is very difficult to do analytically. But we will look at producing polynomial interpolants of the integrand; and polynomials are easily integrated exactly.

4.2.1 Linear interpolation

The simplest form of interpolation is probably the straight line, connecting two points by a straight line. Consider the data $(x_0, y_0), (x_1, y_1)$. The problem is to find a function $P_1(x)$ which passes through these two data points. Since there are only two data points available, the maximum degree of the unique polynomial which passes through these points is one. Let us assume that $P_1(x) = ax + b$ is the straight line passing through the two points then

$$ax_0 + b = y_0,$$
$$ax_1 + b = y_1.$$

Solving for a and b gives

$$a = \frac{y_1 - y_0}{x_1 - x_0}$$
$$b = \frac{x_1 y_0 - x_0 y_1}{x_1 - x_0}$$

Thus, $P_1(x)$ can be written in more convenient ways as

$$\begin{aligned} P_1(x) &= \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1 \\ &= \frac{(x_1 - x)y_0 + (x_0 - x)y_1}{x_1 - x_0} \\ &= y_0 + \frac{x - x_0}{x_1 - x_0}[y_1 - y_0] \\ &= y_0 + \left(\frac{y_1 - y_0}{x_1 - x_0}\right)(x - x_0) \end{aligned} \tag{4.12}$$

Check each of these by evaluating them at $x = x_0$ and x_1 to see if the respective values are y_0 and y_1.

As we will see, the interpolating polynomial can be written in a variety of forms, among these are the **Lagrange form** and the **Newton form**. These forms are equivalent in the sense that the polynomial in question is the one and the same (in fact, the solution to the interpolation task is given by a unique polynomial)

Example 4.3

Suppose we have the following velocity versus time data (a car accelerating from a rest position)

Time, s	Velocity, $m.p.h$
0	0
1	10
2	25
3	36
4	52
5	59

Use linear interpolation to estimate the car's velocity at time $t = 1.5$ and at $t = 4.25$.

Solution: Let's denote the discrete times by t_i and velocities by $v_i, i.e.$, velocity at t_i.

To estimate the car's velocity at $t = 1.5$ we used the data points (t_1, v_1) and (t_1, v_2),i.e., $(1, 10)$ and $(2, 25)$ respectively, in Equation (4.12). Thus, we have

$$\begin{aligned} v(t) &= \frac{t-t_2}{t_1-t_2}v_1 + \frac{t-t_1}{t_2-t_1}v_2, \\ v(1.5) &= \frac{1.5-2}{1-2}10 + \frac{1.5-1}{2-1}25, \\ &= 5 + 12.5 = 17.5 \end{aligned}$$

Next, to estimate the velocity at $t = 4.25$ we use the last two data points and we have

$$\begin{aligned} v(t) &= \frac{t-t_5}{t_4-t_5}v_4 + \frac{t-t_4}{t_5-t_4}v_5, \\ v(4.25) &= \frac{4.25-5}{4-5}52 + \frac{4.25-4}{5-4}59, \\ &= 53.75 \end{aligned}$$

Example 4.4

Given $f(x) = e^x$ and $x_0 = 0.82, x_1 = 0.83$, find $P_1(x)$ which approximates e^x on the interval $[0.82, 0.83]$ and evaluate $P_1(0.826)$.

Solution: First we find the interpolation points,i.e., $y_0 = e^{x_0} = 2.2705$ and $y_1 = e^{x_1} = 2.29332$. Hence, we have

$$P_1(x) = \frac{0.83 - x}{0.83 - 0.82} 2.2705 + \frac{x - 0.82}{0.83 - 0.82} 2.29332$$

Hence

$$P_1(0.826) = 2.284192.$$

Remark: In general, if $y_0 = f(x_0)$ and $y_1 = f(x_1)$ for some function f, then $P_1(x)$ is a linear approximation of $f(x)$ for all $x \in [x_0, x_1]$.

4.2.2 Quadratic interpolation

Given the data points $(x_0, y_0), (x_1, y_1), (x_2, y_2)$, are given data we want to find a quadratic polynomial that passes through these points.

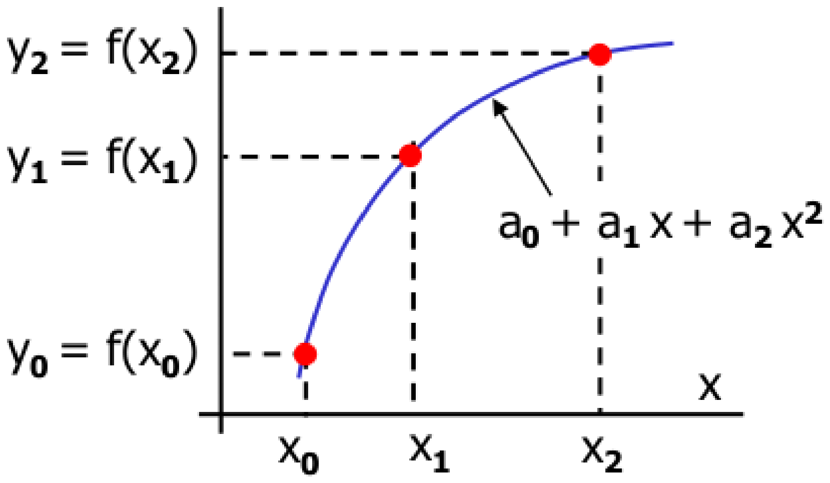


Figure 4.1: Quadratic interpolation

Similar to the linear case, the equation of this parabola is given by

$$P_2(x) = a_0 + a_1 x + a_2 x^2,$$

which satisfies

$$P_2(x_i) = y_i, \quad \text{for} i = 0, 1, 2.$$

for the given data points. One formula for such a polynomial follows:

$$P_2(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) \tag{4.13}$$

with

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}, \qquad L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}, \quad L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)},$$

Equation (4.13) is called **Lagrange's form** of the interpolation polynomial and the functions L_0, L_1 and L_2 are called **Lagrange's interpolating basis functions**. The Lagrange's basis functions have the property that $deg(L_i) \leq 2$ and

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

Here, δ_{ij} is called the Kronecker delta.

Example 4.5

Construct the quadratic polynomial interpolation $P_2(x)$ that interpolates the points $(1, 4), (2, 1)$, and $(5, 6)$.

4.3 Lagrange's interpolation formula

Lagrange's formula is applicable to problems where the independent variable occurs at equal and unequal intervals, but preferably this formula is applied in a situation where there are unequal intervals for the given independent series.

Given $n+1$ discrete data points (x_i, y_i), $i = 0, 1, 2...n$, since there are $(n+1)$ data points (x_i, y_i), we can define a polynomial of degree n as

$$\begin{aligned} P_n(x) =& a_0(x-x_1)(x-x_2)\cdots(x-x_n) + a_1(x-x_0)(x-x_2)\cdots(x-x_n) \\ &+ a_2(x-x_0)(x-x_1)(x-x_3)\cdots(x-x_n) + \cdots + a_n(x-x_0)\cdots(x-x_{n-1}) \end{aligned} \tag{4.14}$$

and we have assumed that, i.e., the interpolation condition:

$$y_i = P_n(x_i) \quad i = 0, 1, ...n$$

For $i = 0$, we have

$$y_0 = P_n(x_0) = a_0(x_0 - x_1)...(x_0 - x_n)$$

$$\therefore a_0 = \frac{y_0}{(x_0 - x_1)...(x_0 - x_n)}$$

For $i = 1$, we get

$$y_1 = P_n(x_1) = a_1(x_1 - x_0)(x_1 - x_2)...(x_1 - x_n)$$

$$\therefore a_1 = \frac{y_1}{(x_1 - x_0)(x_1 - x_2)...(x_1 - x_n)}$$

Similarly for $i = 2.......n - 1$, we get

$$a_i = \frac{y_i}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

and for $i = n$, we get

$$a_n = \frac{y_n}{(x_n - x_0)...(x_n - x_{n-1})}$$

Thus, substituting the a_i's in Equation (4.14) we get

$$\begin{aligned}
P_n(x) =& \frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)}y_0 \\
&+ \frac{(x-x_0)(x-x_2)\cdots(x-x_n)}{(x_1-x_0)(x_1-x_2)\cdots(x_1-x_n)}y_1 \\
&\vdots \\
&+ \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}y_i \\
&\vdots \\
&+ \frac{(x-x_0)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_0)(x_n-x_2)\cdots(x_n-x_{n-1})}y_n
\end{aligned}$$

This can be rewritten in a compact form as:

$$\begin{aligned}
P_n(x) &= L_0(x)y_0 + L_1(x)y_1 + \cdots + L_n(x)y_n \\
&= \sum_{i=0}^{n} L_i(x)y_i
\end{aligned}$$

where

$$L_i(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{((x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$$

Or using the product notation

$$L_i(x) = \frac{\prod_{\substack{k=1\\k\neq i}}^{n}(x-x_k)}{\prod_{\substack{k=1\\k\neq i}}^{n}(x_i-x_k)}$$

Therefore, Lagrange interpolation polynomial of degree n can be written as

$$P_n(x) = \sum_{i=0}^{n} L_i(x)y_i, \tag{4.15}$$

where

$$L_i(x) = \frac{\prod_{\substack{k=1\\k\neq i}}^{n}(x-x_k)}{\prod_{\substack{k=1\\k\neq i}}^{n}(x_i-x_k)} \tag{4.16}$$

Example 4.6

Given the following data table, construct the Lagrange interpolation polynomial $P(x)$, to fit the data and find $f(1.25)$

x_i	0	1	2	3
y_i	1	2.25	3.75	4.25

4.4 Divided difference formula

A major difficulty with the Lagrange Interpolation is that one is not sure about the degree of interpolating polynomial needed to achieve a certain accuracy. Thus, if the accuracy is not good enough with polynomial of a certain degree, one needs to increase the degree of the polynomial, and computations need to be started all over again. Furthermore, computing various Lagrangian polynomials is an expensive procedure. It is, indeed, desirable to have a formula which makes use of $P_{k-1}(x)$ in computing $P_k(x)$.

The following form of interpolation, known as **Newton's interpolation** allows us to do so. The idea is to obtain the interpolating polynomial $P_n(x)$ in the following form:

$$P_n(x) = a_0+a_1(x-x_0)+a_2(x-x_0)(x-x_1)+\cdots+a_n(x-x_0)(x-x_1)\cdots(x-x_{n-1}), \quad (4.17)$$

such that

$$P_n(x_i) = f_i = y_i, \qquad (\textbf{Interpolation Condition}.)$$

Hence, the constants a_0 through a_n can be determined as follows using the interpolation condition.

For $i = 0$ we get

$$f_0 = P_n(x_0) = a_0$$

For $i = 1$ we have

$$f_1 = p_n(x_1) = a_0 + a_1(x_1 - x_0)$$

$$\therefore \quad a_1 = \frac{f_1 - f_0}{x_1 - x_0}$$

For $i = 2$, we have

$$f_2 = p_n(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1)$$

$$\therefore \quad a_2 = \frac{[(f_2 - f_1)/(x_2 - x_1)] - [(f_1 - f_0)/(x_1 - x_0)]}{(x_2 - x_0)}$$

Similarly we can find $a_3.....a_{n-1}$. To express $a_i, i = 0......n - 1$ in a compact manner let us first define the following notation called **divided differences:**

$$f[x_k] = f_k$$

$$f[x_k, x_{k+1}] = \frac{f[x_{k+1}] - f[x_k]}{x_{k+1} - x_k}$$

$$f[x_k, x_{k+1}, x_{k+2}] = \frac{f[x_{k+1}, x_{k+2}] - f[x_k, x_{k+1}]}{x_{k+2} - x_k}$$

Now the co-efficients can be expressed in terms of divided differences as follows:

$$\begin{aligned}
a_0 &= f_0 \\
&= f[x_0] \\
a_2 &= \frac{\frac{f_2 - f_1}{x_2 - x_1} - \frac{f_1 - f_0}{x_1 - x_0}}{x_2 - x_0} \\
&= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\
&= f[x_0, x_1, x_2] \\
a_3 &= \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0} \\
&= f[x_0, x_1, x_2, x_3] \\
&\vdots \\
a_i &= \frac{f[x_1, x_2, \cdots, x_i] - f[x_0, x_1, x_2, \cdots x_{i-1}]}{x_i - x_{x_0}} \\
&= f[x_0, x_1, x_2, \cdots, x_i] \\
&\vdots
\end{aligned}$$

$$\begin{aligned} a_n &= \frac{f[x_1, x_2, \cdots, x_n] - f[x_0, x_1, \cdots, x_{n-1}]}{x_n - x_0} \\ &= f[x_0, x_1, x_2, \cdots, x_n] \end{aligned}$$

Note that a_1 is called as the **first divided difference**, a_2 as the **second divided difference** and so on. Now the polynomial in Equation (4.17) can be rewritten as:

$$\begin{aligned} P_n(x) =& f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &+ \cdots + f[x_0, x_1, \cdots, x_n](x - x_0)(x - x_1)\cdots(x - x_{n-1}) \end{aligned} \tag{4.18}$$

which can be rewritten in a compact form as

$$P_n(x) = \sum_{k=0}^{n} f[x_0, x_1, \cdots, x_k] \prod_{i=0}^{k-1} (x - x_i) \tag{4.19}$$

Thus, Equation (4.18) or (4.19) is called as **Newton's Divided Difference interpolation polynomia**l.

It may also be noted for calculating the higher order divided differences we have used lower order divided differences. In fact starting from the given zeroth order differences ; one can systematically arrive at any of higher order divided differences. For clarity the entire calculation may be depicted in the form of a table called **Newton Divided Difference Table.**

x_i	$f[x_i]$	First order difference	Second order difference	Third order difference	Fourth order difference
x_0	$\mathbf{f[x_0]}$				
		$\mathbf{f[x_0, x_1]}$			
x_1	$f[x_1]$		$\mathbf{f[x_0, x_1, x_2]}$		
		$f[x_1, x_2]$		$\mathbf{f[x_0, x_1, x_2, x_3]}$	
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$		$\mathbf{f[x_0, x_1, x_2, x_3, x_4]}$
		$f[x_2, x_3]$		$f[x_1, x_2, x_3, x_4]$	
x_3	$f[x_3]$		$f[x_2, x_3, x_4]$		
		$f[x_3, x_4]$			
x_4	$f[x_4]$				

Table 4.6: Newton Divided difference table.

In the above Newton divided difference table the bold faces are the coefficients of the polynomial. Again suppose that we are given the data set (x_i, f_i), $i = 0, 1 \cdots, 4$ and that we are interested in finding the 4th order Newton Divided Difference interpolating polynomial. Let us first construct the Newton Divided Difference Table. Wherein one can clearly see how the lower order differences are used in calculating the higher order Divided Differences:

Example 4.7

Construct the Newton Divided Difference Table for generating Newton interpolation polynomial with the following data set:

x_i	0	1	2	3	4
$f(x_i) = y_i$	0	1	8	27	64

Solution: Here $n = 4$. One can find a fourth order Newton Divided Difference interpolation polynomial to the given data. Let us generate Newton Divided Difference Table first; as requested.

x_i	$f[x_i]$	1st order difference	2nd order difference	3^{4d} order difference	4th order difference
0	$\mathbf{0}$				
		$\frac{\mathbf{1-0}}{\mathbf{1-0}} = \mathbf{1}$			
1	1		$\frac{\mathbf{7-1}}{\mathbf{2-0}} = \mathbf{3}$		
		$\frac{8-1}{2-1} = 7$		$\frac{\mathbf{6-3}}{\mathbf{3-1}} = \mathbf{1}$	
2	8		$\frac{19-7}{3-1} = 6$		$\frac{\mathbf{1-1}}{\mathbf{4-0}} = \mathbf{0}$
		$\frac{27-8}{3-2} = 19$		$\frac{9-6}{4-1} = 1$	
3	27		$\frac{37-19}{4-2} = 9$		
		$\frac{64-27}{4-3} = 37$			
4	64				

Note: One may note that the given data corresponds to the cubic polynomial x^3. To fit such a data 3rd order polynomial is adequate. From the Newton Divided Difference table we notice that the fourth order difference is zero. Further the divided differences in the table can be directly used for constructing the Newton Divided Difference interpolation polynomial that would fit the data as follows

$$\begin{aligned} P_3(x) &= \mathbf{0} + \mathbf{1} \times (x-0) + \mathbf{3} \times (x-0)(x-1) + \mathbf{1} \times (x-0)(x-1)(x-2) \\ &= x + 3(x^2 - x) + x^3 - 3x^2 + 2x \\ &= x^3 \end{aligned}$$

The advantage of the above method is that there is no need to start all over again if additional pairs of data are added. We simply need to compute additional divided differences. Since n^{th} order polynomial interpolation of a given $(n+1)$ pairs of data is unique, thus the above polynomial and Lagrangian polynomial are exactly the same.

Example 4.8

Use Newton's Divided Difference formula and evaluate $f(3.0)$ by a third and fourth degree polynomial, given

x_i	3.2	2.7	1.0	4.8	5.6
$f(x_i) = y_i$	22.0	17.8	14.2	38.3	51.7

Solution: Here $n = 4$. One can find a fourth order Newton Divided Difference interpolation polynomial to the given data. Let us generate Newton Divided Difference Table first:

x_i	$f[x_i]$	1^{st} order difference	2^{nd} order difference	3^{4d} order difference	4^{th} order difference
3.2	**22.0**				
		8.400			
2.7	17.8		**2.856**		
		2.118		**-0.5280**	
1.0	14.2		2.012		**0.2560**
		6.342		0.0865	
4.8	38.3		2.63		
		16.750			
5.6	51.7				

The third degree polynomial fitting all points from $x_0 = 3.2$ to $x_3 = 4.8$ is given by

$$\begin{aligned} P_3(x) &= \mathbf{22.0} + \mathbf{8.400}(x-3.2) + \mathbf{2.856}(x-3.2)(x-2.7) \\ &\quad - \mathbf{0.5280}(x-3.2)(x-2.7)(x-1.0) \end{aligned}$$

Hence

$$\begin{aligned} f(3) \approx P_3(3) &= \mathbf{22.0} + \mathbf{8.400}(3-3.2) + \mathbf{2.856}(3-3.2)(3-2.7) \\ &\quad - \mathbf{0.5280}(3-3.2)(3-2.7)(3-1.0) \\ &= \mathbf{20.2120} \end{aligned}$$

The fourth degree polynomial fitting all points from $x_0 = 3.2$ to $x_4 = 5.6$ is also given by

$$\begin{aligned} P_4(x) &= P_3(x) + a_4(x-x_0)(x-x_1)(x-x_2)(x-x_3) \\ &= P_3(x) + \mathbf{0.2560}(x-3.2)(x-2.7)(x-1.0)(x-4.8)(x-5.6) \end{aligned}$$

Hence

$$\begin{aligned} f(3) \approx P_4(3) &= \mathbf{20.2120} + \mathbf{0.2560}(3-3.2)(3-2.7)(3-1.0)(3-4.8)(3-5.6) \\ &= \mathbf{20.2120} + \mathbf{0.055296} \\ &= \mathbf{20.267296}. \end{aligned}$$

4.5 The Newton-Gregory Interpolation Formulae (with equidistant data points)

If the data points are equally spaced, that is,

$$x_{i+1} = x_i + h, \qquad i = 0, 1, n-1$$

then we have

$$x_i = x_0 + ih, \qquad i = 1, 2, \cdots n.$$

Thus, we have then $n+1$ data points as

$$(x_0, y_0), (x_0+h, y_1), (x_0+2h, y_2), \cdots, (x_0+nh, y_n)$$

Using these we can easily simplify the Newton divided differences using the forward, backward and central difference and we can obtain the corresponding interpolating polynomial formula.

4.5.1 The Newton-Gregory Forward Interpolation Formula

Recalling the Divided difference and the forward difference we have the following:

$$\begin{aligned}
f[x_0, x_1] &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\
&= \frac{\triangle f(x_0)}{h} \\
f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\
&= \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2 - x_0} \\
&= \frac{f(x_2) - 2f(x_1) + f(x_0)}{2h^2} \\
&= \frac{\triangle^2 f(x_0)}{2h^2}
\end{aligned}$$

$$\begin{aligned}
f[x_0, x_1, x_2, x_3] &= \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0} \\
&= \frac{\frac{f(x_3)-2f(x_2)+f(x_1)}{2h^2} - \frac{f(x_2)-2f(x_1)+f(x_0)}{2h^2}}{x_3 - x_0} \\
&= \frac{\frac{f(x_3)-3f(x_2)+3f(x_1)-f(x_0)}{2h^2}}{3h} \\
&= \frac{\triangle^3 f(x_0)}{3!h^3}
\end{aligned}$$

Or in general we have

$$f[x_0, x_1, \cdots x_n] = \frac{\triangle^n f(x_0)}{n!h^n}.$$

with this notation the Newton's Divided Difference interpolating polynomial 4.18 can be written as

$$P_n(x) = f(x_0) + \frac{\triangle f(x_0)}{h}(x - x_0) + \frac{\triangle^2 f(x_0)}{2!h^2}(x - x_0)(x - x_1) + \cdots + \frac{\triangle^n f(x_0)}{n!h^n}(x - x_0)(x - x_1)\cdots(x - x_{n-1}) \tag{4.20}$$

simplifies to

$$P_n(x) = f(x_0) + k \triangle f(x_0) + k(k-1)\frac{\triangle^2 f(x_0)}{2!} + \cdots + k(k-1)\cdots(k-n+1)\frac{\triangle^n f(x_0)}{n!}, \tag{4.21}$$

where

$$k = \frac{(x - x_0)}{h}.$$

This is called the **Newton's forward interpolation formula** or **forward Newton-Gregory formula**.

By looking at the forward difference table 4.1 we can see that this formula uses the values along the diagonal of the differences of y - it is a FORWARD DIFFERENCE formula. It is therefore used for interpolation **near the beginning** of a table where k is small.

Example 4.9

In the following table, use the Newton-Gregory Forward Interpolation formula to find

(a) $f(2.4)$

(b) $f(8.7)$

x_i	2	4	6	8	10
$f(x_i) = y_i$	9.86	10.96	12.32	13.76	15.28

Solution: Form a difference table and note that all forward differences > 2 are zero.

x_i	$f(x_i)$	$\triangle$	$\triangle^2$
2	**9.68**		
		1.28	
4	10.96		**0.08**
		1.36	
6	12.32		0.08
		1.44	
8	13.76		0.08
		1.52	
10	15.28		

(a) Here we have $x = 2.4, x_0 = 2, h = 2$ and $k = \dfrac{x - x_0}{h} = \dfrac{2.4 - 2}{2} = 0.2$. Using Equation (4.20) we have

$$\begin{aligned} P_2(x) &= f(x_0) + (x - x_0)\frac{\triangle f(x_0)}{h} + (x - x_0)(x - x_1)\frac{\triangle^2 f(x_0)}{2!h^2} \\ &= 9.68 + (x - 2)\frac{1.28}{2} + (x - 2)(x - 4)\frac{0.08}{2! \times 2^2}. \end{aligned}$$

Hence

$$\begin{aligned} f(2.4) &\approx P_2(2.4) \\ &= 9.68 + (2.4 - 2)\frac{1.28}{2} + (2.4 - 2)(2.4 - 4)\frac{0.08}{2! \times 2^2}. \\ &= 9.9296. \end{aligned}$$

Or using Equation (4.21) we get

$$P_2(x) = f(x_0) + k \triangle f(x_0) + k(k - 1)\frac{\triangle^2}{2!}$$

Hence

$$\begin{aligned} f(2.4) &\approx P_2(2.4) \\ &= 9.68 + 0.2 \times 1.28 + 0.2(0.2 - 1)\frac{0.08}{2!} \\ &= 9.9296 \end{aligned}$$

(b) Here we have $x = 8.7; x_0 = 2; h = 2$ and $k = \frac{8.7-2}{2} = 3.35$. Now using

Equation (4.21) we get

$$f(8.7) \approx P_2(8.7) = 9.68 + 3.35(1.28) + 3.35(3.35-1)\frac{0.08}{2!}$$
$$= 14.2829$$

Example 4.10

IIn the following table of e^x use the Newton-Gregory formula of forward interpolation to calculate

(a) $f(0.12)$ (b) $f(2)$

x_i	0.1	0.6	1.1	1.6	2.1
e_i^x	1.1052	1.8221	3.0042	4.9530	8.1662

Solution: First let's construct the forward difference table:

x_i	$f(x_i)$	$\triangle$	$\triangle^2$	$\triangle^3$	$\triangle^4$
0.1	**1.1052**				
		0.7169			
0.6	1.8221		**0.4652**		
		1.1821		**0.3015**	
1.1	3.0042		0.7667		**0.1962**
		1.9488		0.4977	
1.6	4.9530		1.2644		
		3.2132			
2.1	8.1662				

Note that in this case there is no difference column that is constant. This is to be expected since ex cannot be represented by a polynomial function of finite degree.

(a) Here we have $x = 0.12, x_0 = 0.1, h = 0.5$ and $k = 0.04$.

$$\begin{aligned} e^{0.12} &\approx 1.1052 + 0.04(0.7169) + 0.04(0.04-1)\frac{0.4652}{2!} \\ &+ 0.04(0.04-1)(0.04-2)\frac{0.3015}{3!} \\ &+ 0.04(0.04-1)(0.04-2)(0.04-3)\frac{0.1962}{4!} \\ &= 1.1269. \qquad \text{(correct value to 5 d.p. is 1.12750)} \end{aligned}$$

(b) Here we have $x = 2; x_0 = 0.1; h = 0.5$ and $k = 3.8$ and we get

$$\begin{aligned} e^{2.00} &\approx 1.1052 + 2.72422 + 2.47486 + 0.96239 + 0.12525 \\ &= 7.3919 \end{aligned}$$

In Example 4.5.1 the interpolation formula is identical with $f(x)$, which is a quadratic function, and the results for $f(2.4)$ and $f(8.7)$ will therefore be correct to the number of decimal places retained. In example 4.5.1 the function e^x is replaced by a 4^{th} degree polynomial which takes the value of e^x at the five given entries. Because the successive difference decrease, higher differences are relatively small and the value of the estimate converges. From direct calculation it turns out that the error in the estimate for $e^{0.12}$ is about 0.05 percent and for $e^{2.00}$ it is about 0.04 percent.

4.5.2 The Newton-Gregory Backward Interpolation Formula

For interpolating the value of the function $y = f(x)$ near the end of the given data points and also to extrapolate value of the function a short distance forward from y_n, Newton's backward interpolation formula is used.

Let $y = f(x)$ represent a function which assumes the values $y_0, y_1, \cdots, y_n$ corresponding to the equidistant interpolation points $x_0, x_1, \cdots, x_n$ of the argument x, the common difference being h. With these $n+1$ data points, we may consider a polynomial of degree n as our interpolation formula. Suppose the n^{th} degree polynomial is takes the form:

$$P_n(x) = a_n + a_{n-1}(x - x_n) + a_{n-2}(x - x_n)(x - x_{n-1}) + \cdots + a_0(x - x_n)(x - x_{n-1}) \cdots (x - x_1). \tag{4.22}$$

Now, the constants $a_n, a_{n-1}, \cdots, a_0$ are computed such that

$$P_n(x_n) = y_n, P_n(x_{n-1}) = y_{n-1} \cdots P_n(x_0) = y_0.$$

Thus, evaluating Equation (4.22) at $x = x_n$ we get

$$P_n(x_n) = a_n \implies a_n = y_n$$

Again,

$$\begin{aligned} P_n(x_{n-1}) &= a_n + a_{n-1}(x_{n-1} - x_n) \\ \implies \qquad y_{n-1} &= y_n + a_{n-1}(x_{n-1} - x_n) \\ \implies \qquad a_{n-1} &= \frac{y_n - y_{n-1}}{x_n - x_{n-1}} \\ &= \frac{\nabla y_n}{h}. \end{aligned}$$

When we evaluate $P_n(x)$ at $x = x_{n-2}$ we get

$$\begin{aligned} P_n(x_{n-2}) &= a_n + a_{n-1}(x_{n-2} - x_n) + a_{n-1}(x_{n-2} - x_n)(x_{n-2} - x_{n-1}) \\ \implies \qquad y_{n-2} &= y_n + \frac{y_n - y_{n-1}}{h}(2h) + a_{n-2}(2h)(h) \\ &= y_n + 2(y_{n-1} - y_n) + a_{n-2}(2h^2) \\ \implies \qquad a_{n-2} &= \frac{y_{n-2} - 2(y_{n-1} - y_n) - y_n}{2h^2} \\ &= \frac{y_{n-2} - 2y_{n-1} + y_n}{2!h^2} \\ &= \frac{\nabla^2 y_n}{2!h^2}. \end{aligned}$$

In general, for each $x = x_i$ we get

$$a_i = \frac{\nabla^{n-i} y_n}{(n-i)!h^{n-i}}, \qquad i = n-3, n-4, \cdots, 0$$

Substituting the value of the a_i's in Equation (4.22) results in

$$P_n(x) = y_n + \frac{\nabla y_n}{h}(x - x_n) + \frac{\nabla^2 y_n}{2!h^2}(x - x_n)(x - x_{n-1}) + \cdots + \frac{\nabla^n y_n}{n!h^n}(x - x_n)(x - x_{n-1}) \cdots (x - x_0). \tag{4.23}$$

Now, setting

$$k = \frac{x - x_n}{h},$$

then $x - x_n = kh$ and $x - x_{n-1} = x - (x_n - h) = x - x_n + h = kh + h = (k+1)h$ Similarly,

$$\begin{aligned} x - x_{n-2} &= (k+2)h \\ x - x_{n-3} &= (k+3)h \\ &\vdots \\ x - x_1 &= (k + (n-1))h \end{aligned}$$

Thus, Equation (4.23) reduces to

$$P_n(x) = y_n + k \nabla y_n + k(k+1)\frac{\nabla^2 y_n}{2!} + \cdots + k(k+1)\cdots(k+(n-1))\frac{\nabla^n y_n}{n!}, \quad (4.24)$$

where

$$k = \frac{(x - x_n)}{h}$$

This is called the **Newton's back interpolation formula** or **Newton-Gregory backward formula**.

Example 4.11

For the following table of values, estimate $f(7.5)$

x_i	1	2	3	4	5	6	7	8
$y_i = f(x_i)$	1	8	27	64	125	216	343	512

Solution: The value to be interpolated is at the end of the table. Hence, it is appropriate to use Newton's backward interpolation formula. Let's first construct the backward difference table for the given data.

x_i	$f(x_i)$	∇	∇^2	∇^3	∇^4
1	1				
		7			
2	8		12		
		19		6	
3	27		18		0
		37		6	
4	64		24		0
		61		6	
5	125		30		0
		91		6	
6	216		36		**0**
		127		**6**	
7	343		**42**		
		169			
8	**512**				

Since the 4th and higher order differences are zero, the required Newton's backward interpolation formula is

$$y_x = y_n + k \bigtriangledown y_n + k(k+1)\frac{\bigtriangledown^2 y_n}{2!} + k(k+1)(k+1)\frac{\bigtriangledown^3 y_n}{3!}. \tag{4.25}$$

In this problem,

$$k = \frac{x - x_n}{1} = \frac{7.5 - 8}{1} = -0.5,$$

hence, we have

$$\begin{aligned} y_{7.5} &\approx 512 + (-0.5)169 + (-0.5)(-0.5+1)\frac{42}{2} \\ &+ (-0.5)(-0.5+1)(-0.5+2)\frac{6}{3!} \\ &= 512 - 84.5 - 5.25 - 0.375 \\ &= 421.875 \end{aligned} \tag{4.26}$$

4.6 Error in Polynomial Interpolation

Our goal in this section is to provide estimates on the “error" we make when interpolating data that is taken from sampling an underlying function f (x). While the interpolant and the function agree with each other at the interpolation points, there is, in general, no reason to expect them to be close to each other elsewhere. Nevertheless, we can estimate the difference between them, a difference which we refer to as the **interpolation error**. The interpolating polynomial P_n is an approximation to f , but unless f itself is a polynomial of degree n, there will be a nonzero error $e(x) = f(x) - P_n(x)$. At times it is useful to have an explicit expression for the error and is given by the following theorem which we are not going to prove.

Theorem 4.1

Let f be a given function on $[a, b]$ and P_n be the polynomial of degree less than or equal to n interpolating the f at the $n+1$ data points $x_0, x_1, x_2, \cdots, x_n$ in $[a, b]$. If f has $n+1$ continuous derivatives and x_i are distinct, then

$$e_n(x) = |f(x) - P_n(x)| = \prod_{i=0}^{n} (x - x_k)\, f^{(n+1)}(\xi_x), \tag{4.27}$$

where $a \le x \le b$ and ξ_x is between the minimum and maximum of $x_0, x_1, \cdots, x_n$.

In addition to the interpretation of the divided difference of order n as the coefficient of x_n in some interpolation polynomial, it can also be characterized in another important way. Consider, e.g., the first-order divided difference

$$f[x_0, x_] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

Since the order of the points does not change the value of the divided difference, we can assume, without any loss of generality, that $x_0 < x_1$. If we assume, in addition, that $f(x)$ is continuously differentiable in the interval $[x_0, x_1]$, then this divided difference equals to the derivative of $f(x)$ at an intermediate point, i.e.,

$$f[x_0, x_1] = f'(\xi), \quad \xi \in (x_0, x_1)$$

In other words, the first-order divided difference can be viewed as an approximation of the first derivative of $f(x)$ in the interval. It is important to note that while this interpretation is based on additional smoothness requirements from $f(x)$ (i.e. its being differentiable), the divided differences are well defined also for non-differentiable functions. This notion can be extended to divided differences of higher order as stated bellow

Let $x, x_0, \cdots, x_{n-1}$ be $n-1$ distinct points. Let $a = \min(x, x_0, \cdots, x_{n-1})$ and $b = \max(x, x_0, \cdots, x_{n-1})$. Assume that $f(x)$ has a continuous derivative of order n in the interval (a, b). Then

$$f[x, x_0, x_1, \cdots, x_{n-1}] = \frac{f^n(\xi)}{n!}, \tag{4.28}$$

where $\xi \in (a, b)$.

Example 4.12

If $P(x)$ is the polynomial that interpolates the function $f(x) = \sin(x)$ at 10 points on the interval $[0, 1]$, what is the greatest possible error?

Solution: In this example we have $n + 1 = 10$ data points, hence $n = 9$ and

$$f^{(n+1)}(x) = f^{(10)}(x) = -\sin(x).$$

Thus, the largest possible error would be the maximum of $e_{10}(x)$, i.e.,

$$\left|\frac{1}{10!}f^{(10)}(\xi_x)\prod_{i=0}^{n}(x-x_i)\right|,$$

for $x, x_0, x_1, \cdots, x_9, \xi_x \in [0,1]$. Clearly, on the interval $[0,1]$

$$\max|x-x_i| = 1$$

and

$$\max\left|f^{(10)}(\xi_x)\right| = \max|-s\sin(x)| = 1$$

so the maximum error would be

$$\frac{1}{10!}(1)(1)^{n+1} \approx 2.8\times 10^{-7}.$$

Example 4.13

Determine the spacing h in a table of evenly spaced values of the function $f(x) = \sqrt{x}$ between 1 and 2, so that interpolation with a second-degree polynomial in this table will yield a desired accuracy ϵ.

Solution: Such a table contains the values $f(x_i), i = 0, 1, \cdots, n = \frac{1}{h}$, at points $x_i = 1 + ih$. If $x \in [x_{i-1}, x_{i+1}]$, then we approximate $f(x)$ with $P_2(x)$, where $P_2(x)$ is the polynomial of degree 2 that interpolates f at x_{i-1}, x_i, x_{i+1}. Thus, by Theorem 4.6, the error is

$$e_2(x) = \frac{f'''(\xi_x)}{3!}(x-x_{i-1})(x-x_i)(x-x_{i+1}),$$

where ξ_x depends on x. Though ξ_x is unknown to us, we can drive the following error bound:

$$\begin{aligned} f'''(\xi_x) &\le \max_{1\le x\le 2}|f'''(x)| \\ &= \max_{1\le x\le 2}\frac{3}{8}x^{\frac{-5}{2}}, \\ &= \frac{3}{8} \end{aligned}$$

and for any $x \in [x_{i-1}, x_i]$,

$$\begin{aligned}|(x - x_{i-1})(x - x_i)(x - x_{i+1})| &\le \left|\max_{y\in[-h,h]}(y - h)h(y + h)\right| \\ &= \frac{2}{3\sqrt{3}}h^3.\end{aligned}$$

In the last step, the maximum absolute value of $g(y) = (y-h)y(y+h)$ over $[-h, h]$ is obtained as follows. Since

$$g(-h) = g(h) = g(0) = 0$$

$g(y)$ achieves its maximum (or minimum) in the interior of the interval. We have

$$g'(h) = 3y^2 - h^2.$$

Thus, $g'(h) = 0$ yields $y = \pm\frac{h}{\sqrt{3}}$, where the maximum of $|g(y)|$ is attained. Hence we have derived a bound on the interpolation error:

$$\begin{aligned}|e_2(x)| &\le \frac{1}{3!}\cdot\frac{3}{8}\cdot\frac{h}{\sqrt{3}} \\ &= \frac{h^3}{2\sqrt{3}}.\end{aligned} \tag{4.29}$$

Therefore, we have

$$\frac{h^3}{2\sqrt{3}} < \epsilon.$$

In particular, suppose we want the accuracy of at least 7 places after zero. We should choose h such that

$$\frac{h^3}{24\sqrt{3}} < 5 \times 10^{-7}.$$

This gives us $h \approx 0.0127619$. And the number of entries is about 79.

Exercise 4.1

Construct the Lagrange and Newton forms of the interpolating polynomial $P_3(x)$ for the function $f(x) = \sqrt[3]{x}$ which passes through the points $(0, 0), (1, 1), (8, 2)$ and $(27, 3)$. Calculate the interpolation error at $x = 5$ and compare with the theoretical error bound.

4.6.1 Spline Interpolation

Linear Spline

Definition 4.10

Function S is called a spline of degree one or linear spline on $[a, b]$ if:

i. S is continuous on $[a, b]$.

ii. There is a partitioning of the interval $a = x_0 < x_1 < x_2 < \cdots < x_n = b$ such that S is a linear polynomial on each sub interval.

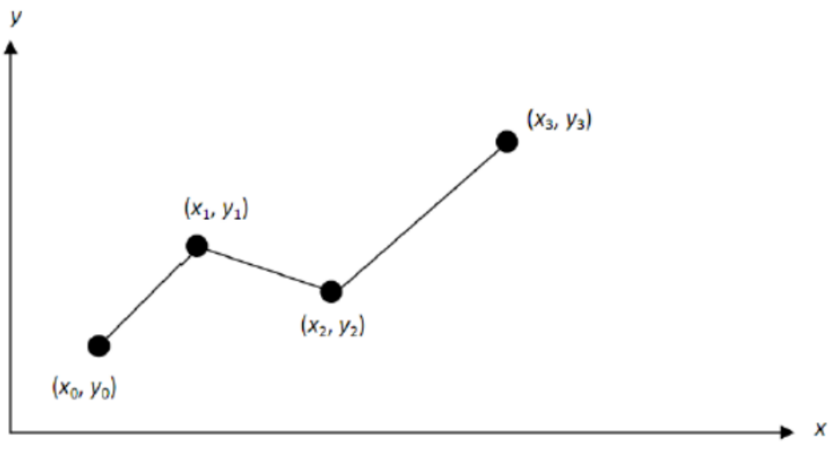


Figure 4.2: Linear Spline

Example 4.14

Determine whether the following function is linear spline function or not

$$s(x) = \begin{cases} x, & -1 \le x \le 0 \\ 1 - x, & 0 < x < 1 \\ 2x - 2, & 1 \le x \le 2 \end{cases}$$

Solution:

Example 4.15

State whether the following piecewise polynomials are linear splines or not.

$$s(x) = \begin{cases} x+1, & -1 \le x \le 0 \\ 2x+1, & 0 < x < 1 \\ 4-x, & 1 \le x \le 2 \end{cases}$$

Solution:

Linear spline can be used for interpolation

Example 4.16

Obtain the piecewise linear interpolating polynomial (linear spline) for the function $f(x)$ defined by the data:

x	1	2	4	8
$f(x)$	3	7	21	73

and estimate the value of $f(3)$ and $f(7)$.

Solution: For the interval $[1,2]$ (i.e., for points $(1,3)$ and $(2,7)$) we have:

$$\begin{aligned} P_1(x) &= l_0(x)y_0 + l_1(x)y_1 \\ \implies P_1(x) &= \frac{x-x_1}{x_0-x_1}y_0 + \frac{x-x_0}{x_1-x_0}y_1 \\ &= \frac{x-2}{1-2}(3) + \frac{x-1}{2-1}(7) \\ &= -3x+6+7x-7 \\ \implies P_1(x) &= 4x-1, \quad 1 \le x \le 2 \end{aligned}$$

For the interval $[2,4]$ (i.e., for points $(2,7)$ and $(4,21)$) we have:

$$\begin{aligned} P_2(x) &= l_0(x)y_0 + l_1(x)y_1 \\ \implies P_2(x) &= \frac{x-x_1}{x_0-x_1}y_0 + \frac{x-x_0}{x_1-x_0}y_1 \\ &= \frac{x-4}{2-4}(7) + \frac{x-2}{4-2}(21) \\ &= 7x-7 \\ \implies P_2(x) &= 7x-7, \quad 2 \le x \le 4 \end{aligned}$$

For the interval $[4,8]$ (i.e., for points $(4,21)$ and $(8,73)$) we have:

$$\begin{aligned}
P_2(x) &= l_0(x)y_0 + l_1(x)y_1 \\
\implies P_2(x) &= \frac{x-x_1}{x_0-x_1}y_0 + \frac{x-x_0}{x_1-x_0}y_1 \\
&= \frac{x-8}{4-8}(21) + \frac{x-2}{8-4}(73) \\
&= 13x-31 \\
\implies P_2(x) &= 13x-31, \quad 4 \le x \le 8
\end{aligned}$$

Hence the linear spline interpolating the above data is:

$$P(x) = \begin{cases} 4x-1, & 1 \le x \le 2 \\ 7x-7, & 2 \le x \le 4 \\ 13x-31, & 4 \le x \le 8 \end{cases}$$

Since 3 is in $2 \le x \le 4$, $f(3) = 7(3) - 7 = 14$. In addition, since 7 in $4 \le x \le 8, ; f(7) = 13(7) - 31 = 60$

Example 4.17

Approximate the function $f(x) = 4^x$ on $[-1,1]$ by linear spline method and find the value of $f(0.125)$.

4.6.2 Cubic Spline

Definition 4.11

A cubic spline,$S(x)$ is defined by the following conditions:

i. $S(x)$ is a cubic polynomial in each of the sub intervals $[x_{j-1}, j]$, $j = 1, 2, \cdots, n$ and

ii. $S(x)$, $S'(x)$ and $S''(x)$ are continuous at each point.

If the data

x	x_1	x_2	$\cdots$	x_1
$f(x)$	y_3	y_2	$\cdots$	y_1

is given then the cubic spline $s(x)$ on this given data is found by finding the spline on $[x_{j-1}, x_j]$, $j = 1, 2, \cdots, n$ using the formula:

$$\begin{aligned} s(x) =& \frac{1}{6h}\left((x_j - x)^3 M_{j-1} + (x - x_{j-1})^3 M_j\right) + \frac{1}{h}(x_j - x)\left(y_{j-1} - \frac{h^2}{6}M_{j-1}\right) \\ &+ \frac{1}{h}(x - x_{j-1})\left(y_j - \frac{h^2}{6}M_j\right) \end{aligned} \tag{4.30}$$

And

$$\begin{aligned} & M_{j-1} + 4M_j + M_j = \frac{6}{h^2}(y_{j-1} - 2y_j + y_{j+1}), \quad j = 1, 2, \cdots n-1, \\ \text{where} \quad & s''(x_j) = M_j. \end{aligned} \tag{4.31}$$

Equation (**??**) gives a system of $(n11)$ linear equations in the $(n+1)$ unknowns $M_0, M_1, M_2, \cdots, M_n$. Two more conditions, called the **end conditions**, have to be prescribed to obtain $(n+1)$ equations in $(n+1)$ unknowns $M_0, M_1, M_2, \cdots, M_n$.

Different types of cubic splines are obtained when different end conditions are supplied.

1. When we assume the condition that $M_0 = 0$ and $M_n = 0$ the cubic interpolation is called the **Natural Cubic Spline.**

2. We may also assume that $S''(x)$ to be constant near end points i.e. $M_0 = M_1$ and $M_n = M_{n-1}$

3. We can impose the first derivative condition at the end points as follows:

$$M_0 = y_0' \qquad \text{and} \quad M_n = y_n'$$

4. Similarly ce can impose the second derivative condition at the end points as follows:

$$M_0 = y_0'' \qquad \text{and} \quad M_n = y_n''$$

Example 4.18

The following values of x and y are given:

1	2	3	4
1	2	5	11

Find the natural cubic spline and evaluate $y(1.5)$ and $y\prime(3)$

Solution:

Chapter 5

Numerical Differentiation and Integration

5.1 Differentiation

Differentiation is a very important mathematical process and a great deal of effort has been devoted to the development of analytic techniques of finding the derivatives of various mathematical functions. It often occurs, however, that it is not possible to utilise these traditional methods. This happens when:

(i) The function is too complex

(ii) The function is unknown (when data are collected from some experiment).

In this section numerical techniques are described which provide an estimate of the derivative of a tabulated function. The main concept of numerical differentiation is stated below:

construct an appropriate interpolation polynomial from the given set of values of x and y and then differentiate it at any value of x.

Like interpolation, lot of formulae are available for differentiation. Based on the given set of values, different types of formulae can be constructed. The common formulae are based on Lagrange's and Newton's interpolation formulae. These formulae are discussed in this section.

5.1.1 Differentiation based on Newton's forward interpolation formula

We know that the Newton's forward and backward interpolation formulae are applicable only when the arguments are in equispaced. So, we assumed that the given arguments are equispaced.

Let the function $y = f(x)$ be known at the $(n+1)$ equispaced arguments $x_0, x_1, \cdots, x_n$ and $y_i = f(x_i)$ for $i = 0, 1, \cdots, n$. Since the arguments are in equispaced, therefore one can write $x_i = x_0 + ih$. Also, let $k = \frac{x - x_0}{h}$ where h is called the spacing. For this data set, the Newton's forward interpolation formula is

$$\begin{aligned} P_n(x) &= y_0 + k \triangle y_0 + \frac{k(k-1)}{2!} \triangle^2 y_0 + \cdots + \frac{k(k-1)\cdots(k-n+1)}{n!} \triangle^n y_0 \\ &= y_0 + k \triangle y_0 + \frac{k^2 - k}{2!} \triangle^2 y_0 + \frac{k^3 - 3k^2 + 2k}{3!} \triangle^3 y_0 \\ &+ \frac{k^4 - 6k^3 + 11k^2 - 6k}{4!} \triangle^4 y_0 + \frac{k^5 - 10k^4 + 35k^3 - 50k^2 + 24k}{5!} \triangle^5 y_0 + \cdots \end{aligned} \tag{5.1}$$

The error term of this interpolation formula is

$$E_(x) = \frac{k(k-1)(k-2)\cdots(k-n)}{(n+1)!} h^{n+1} f^{n+1}(\xi), \tag{5.2}$$

where $\min\{x, x_0, x_1, \cdots, x_n\} < \xi < \{x, x_0, x_1, \cdots, x_n\}$.

Differentiating Equation (5.1) thrice to get the first three derivatives of

$$\begin{aligned} P_n'(x) = \frac{1}{h} \Bigg[&\triangle y_0 + \frac{2k-1}{2!} \triangle^2 y_0 + \frac{3k^2 - 6k + 2)}{3!} \triangle^3 y_0 + \frac{4k^3 - 18k^2 + 22k - 6}{4!} \triangle^4 y_0 \\ &+ \frac{5k^4 - 40k^3 + 105k^2 - 100k + 24}{5!} \triangle^5 y_0 + \cdots \Bigg] \qquad \left(\because \frac{\mathrm{d}k}{\mathrm{d}x} = \frac{1}{h} \right) \end{aligned} \tag{5.3}$$

$$\begin{aligned} P_n''(x) = \frac{1}{h^2} \Bigg[&\triangle^2 y_0 + \frac{6k-6}{3!} \triangle^3 y_0 + \frac{12k^2 - 36k + 22}{4!} \triangle^4 y_0 \\ &+ \frac{20k^3 - 120k^2 + 210k - 100}{5!} \triangle^5 y_0 + \cdots \Bigg] \end{aligned} \tag{5.4}$$

$$P_n'''(x) = \frac{1}{h^3}\left[\triangle^3 y_0 + \frac{24k-36}{4!}\triangle^4 y_0 + \frac{60k^2-240k+210}{5!}\triangle^5 y_0 + \cdots\right] \tag{5.5}$$

and so on.

In this way, we can find all other derivatives. It may be noted that $\triangle y_0, \triangle^2 y_0, \triangle^3 y_0, \cdots$ are constants.

The above three formulae give the first three (approximate) derivatives of $f(x)$ at any arbitrary argument x where $x = x_0 + kh$. The above formulae become simple when $x = x_0$, i.e. $k = 0$. That is,

$$P_n'(x_0) = \frac{1}{h}\left[\triangle y_0 - \frac{1}{2}\triangle^2 y_0 + \frac{1}{3}\triangle^3 y_0 - \frac{1}{4}\triangle^4 y_0 + \frac{1}{5}\triangle^5 y_0 - \cdots\right] \tag{5.6a}$$

$$P_n''(x_0) = \frac{1}{h}\left[\triangle^2 y_0 - \triangle^3 y_0 + \frac{11}{12}\triangle^4 y_0 - \frac{5}{6}\triangle^5 y_0 + \cdots\right] \tag{5.6b}$$

$$P_n'''(x_0) = \frac{1}{h}\left[\triangle^3 y_0 - \frac{3}{2}\triangle^4 y_0 + \frac{7}{5}\triangle^5 y_0 - \cdots\right] \tag{5.6c}$$

Error in Newton's forward differentiation formula

The error in Newton's forward differentiation formula is calculated from the expression of error in Newton's forward interpolation formula. The error in Newton's forward interpolation formula is given by

$$E_n(x) = k(k-1)(k-2)\cdots(k-n)h^{n+1}\frac{f^{(n+1)}(\xi)}{(n+1)!}$$

Differentiating this expression with respect to x, then we have

$$\begin{aligned} E_n'(x) &= h^{n+1}\frac{f^{(n+1)}(\xi)}{(n+1)!}\frac{\mathrm{d}}{\mathrm{d}k}\left[k(k-1)\cdots(k-n)\right]\frac{1}{h} + \frac{k(k-1)\cdots(k-n)}{(n+1)!}h^{n+1}\frac{\mathrm{d}}{\mathrm{d}x}\left[f^{(n+1)}(\xi)\right] \\ &= h^n\frac{f^{(n+1)}(\xi)}{(n+1)!}\frac{\mathrm{d}}{\mathrm{d}k}\left[k(k-1)\cdots(k-n)\right] + \frac{k(k-1)\cdots(k-n)}{(n+1)!}h^{n+1}f^{(n+2)}(\xi_1) \end{aligned}$$

where ξ and ξ_1 are two quantities depend on x and $\min\{x, x_0, \cdots, x_n\} < \xi, \xi_1 < \max\{x, x_0, \cdots, x_n\}$.

The expression for error at the starting argument $x = x_0$, i.e., $k = 0$ is evaluated as

$$\begin{aligned}
E_n'(x_0) &= h^n \frac{f^{(n+1)}(\xi)}{(n+1)!} \frac{\mathrm{d}}{\mathrm{d}k}\left[k(k-1)\cdots(k-n)\right]_{k=0} + 0 \\
&= \frac{h^n(-1)^n n! f^{(n+1)}(\xi)}{(n+1)!} \qquad \left[\text{as } \frac{\mathrm{d}}{\mathrm{d}k}\left[k(k-1)\cdots(k-n)\right]_{k=0} = (-1)^n n!\right] \\
&= \frac{(-1)^n h^n f^{(n+1)}(\xi)}{n+1},
\end{aligned}$$

where ξ lies between $\min\{x, x_0, \cdots, x_n\}$ and $\max\{x, x_0, \cdots, x_n\}$.

Example 5.1

Consider the following table

x	:	1.0	1.5	2.0	2.5	3.0	3.5
y	:	1.234	2.453	7.625	12.321	18.892	23.327

Find the value of $\frac{\mathrm{d}y}{\mathrm{d}x}, \frac{\mathrm{d}^2y}{\mathrm{d}x^2}$ at $x = 1$ and $\frac{\mathrm{d}y}{\mathrm{d}x}$ when $x = 1.2$

Solution: The forward difference table is

x	y	$\triangle y$	$\triangle^2 y$	$\triangle^3 y$	$\triangle^4 y$	$\triangle^5 y$
1.0	1.234					
		3.425				
1.5	2.453		3.095			
		6.520		0.36		
2.0	7.625		3.455		0.71	
		9.975		1.07		−0.680
2.5	12.321		4.525		0.03	
		14.500		1.10		
3.0	18.892		5.625			
		20.125				
3.5	23.327					

Here $x = x_0 = 1$ and $h = 0.5$. Then $k = 0$ hence we can use Equation (5.6a) to find the first derivative at $x = 1$. Thus,

$$\begin{aligned}
y'(1) &\approx \frac{1}{h}\left[\triangle y_0 - \frac{1}{2}\triangle^2 y_0 + \frac{1}{3}\triangle^3 y_0 - \frac{1}{4}\triangle^4 y_0 + \frac{1}{5}\triangle^5 y_0\right] \\
&= \frac{1}{0.5}\left[3.425 - \frac{1}{2}\times 3.095 + \frac{1}{3}\times 0.36 - \frac{1}{4}\times 0.71 + \frac{1}{5}\times(-0.680)\right] \\
&= 3.36800.
\end{aligned}$$

Similarly, using Equation (5.6b) we can compute the second derivative at $x = 1$:

$$\begin{aligned}
y''(1) &\approx \frac{1}{0.5}\left[\triangle^2 y_0 - \triangle^3 y_0 + \frac{11}{12}\triangle^4 y_0 - \frac{5}{6}\triangle^5 y_0\right] \\
&= 4.0 \times \left[3.095 - \times 0.36 + \frac{11}{12} \times 0.71 + \frac{5}{6} \times (-0.680)\right] \\
&= 15.8100.
\end{aligned}$$

Now, at $x = 1.2, h = 0.5, k = \dfrac{x - x_0}{h} = \dfrac{1.2 - 1}{0.5} = 0.4$
Therefore, using Equation (5.3) we have:

$$\begin{aligned}
y'(1.2) &= \frac{1}{0.5}\left[\triangle y_0 + \frac{2k-1}{2!}\triangle^2 y_0 + \frac{3k^2 - 6k - 2)}{3!}\triangle^3 y_0 + \frac{4k^3 - 18k^2 + 22k - 6}{2!}\triangle^4 y_0 \right. \\
&\quad \left. + \frac{5k^4 - 40k^3 + 105k^2 - 100k + 24}{2!}\triangle^5 y_0\right] \\
&= \frac{1}{0.5}\left[3.425 + \frac{2 \times 0.4 - 1}{2!} \times 3.095 + \frac{3(0.4)^2 - 6(0.4) - 2)}{3!}0.36 \right. \\
&\quad + \frac{4(0.4)^3 - 18(0.4)^2 + 22(0.4) - 6}{4!} \times 0.71 \\
&\quad \left. + \frac{5(0.4)^4 - 40(0.4)^3 + 105(0.4)^2 - 100(0.4) + 24}{5!} \times (-0.68)\right] \\
&= 6.26948.
\end{aligned}$$

5.1.2 Differentiation based on Newton's backward interpolation formula

Like Newton's forward differentiation formula one can derive **Newton's backward differentiation formula** based on Newton's backward interpolation formula.

Suppose the function $y = f(x)$ is not know explicitly, but it is known at $(n+1)$ arguments $x_0, x_1, \cdots, x_n$. That is, $y_i = f(x_i), i = 0, 1, 2, \cdots, n$ are given. Since the Newton's backward interpolation formula is applicable only when the arguments are equispaced, therefore, $x_i = x_0 + ih, i = 0, 1, 2, \cdots, n$ and $k = \dfrac{x - x_n}{h}$.

The Newton's backward interpolation formula is

$$P_n(x) = y_n + k\nabla y_n + \frac{k(k+1)}{2!}\nabla^2 y_n + \frac{k(k+1)(k+2)}{3!}\nabla^3 y_n \\ + \frac{k(k+1)(k+2)(k+3)}{4!}\nabla^4 y_n + \frac{k(k+1)(k+2)(k+3)(k+4)}{5!}\nabla^5 y_n + \cdots$$

Differentiating this formula with respect to x successively, the formulae for derivatives of different order can be derived as

$$P_n'(x) = \frac{1}{h}\left[\nabla y_n + \frac{2k+1}{2!}\nabla^2 y_n + \frac{3k^2+6k+2}{3!}\nabla^3 y_n + \frac{4k^3+18k^2+22k+6}{4!}\nabla^4 y_n \right. \\ \left. + \frac{5k^4+40k^3+105k^2+100k+24}{5!}\nabla^5 y_n + \cdots\right] \tag{5.7}$$

$$P_n''(x) = \frac{1}{h^2}\left[\nabla^2 y_n + \frac{6k+6}{3!}\nabla^3 y_n + \frac{12k^2+36k+22}{4!}\nabla^4 y_n \right. \\ \left. + \frac{20k^3+120k^2+210k+100}{5!}\nabla^5 y_n + \cdots\right] \tag{5.8}$$

$$P_n'''(x) = \frac{1}{h^2}\left[\nabla^3 y_n + \frac{24k+36}{4!}\nabla^4 y_n + \frac{60k^2+240k+210}{5!}\nabla^5 y_n + \cdots\right] \tag{5.9}$$

and so on.

The above formulae give the approximate value of $\frac{\mathrm{d}y}{\mathrm{d}x}, \frac{\mathrm{d}^2y}{\mathrm{d}x^2}, \frac{\mathrm{d}^3y}{\mathrm{d}x^3}$, and so on, at any point value of x, where $\min\{x, x_0, \cdots, x_n\} < x < \max\{x, x_0, \cdots, x_n\}$.

When $x = x_n$ then $v = 0$. In this particular case, the above formulae reduced to the following form.

$$P_n'(x_n) = \frac{1}{h}\left[\nabla y_n + \frac{1}{2}\nabla^2 y_n + \frac{1}{3}\nabla^3 y_n + \frac{1}{4}\nabla^4 y_n + \frac{1}{5}\nabla^5 y_n + \cdots\right] \tag{5.10a}$$

$$P_n''(x_n) = \frac{1}{h}\left[\nabla^2 y_n + \nabla^3 y_n + \frac{11}{12}\nabla^4 y_n + \frac{5}{6}\nabla^5 y_n + \cdots\right] \tag{5.10b}$$

$$P_n'''(x_n) = \frac{1}{h}\left[\nabla^3 y_n + \frac{3}{2}\nabla^4 y_n + \frac{7}{5}\nabla^5 y_n + \cdots\right] \tag{5.10c}$$

Error in Newton's backward differentiation formula

The error can be calculated by differentiating the error in Newton's backward interpolation formula. Such error is given by

$$E_n(x) = k(k+1)(k+2)\cdots(k+n)h^{n+1}\frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

where $k = \dfrac{x-x_n}{h}$ and ξ lies between $\min\{x, x_0, \cdots, x_n\}$ and $\max\{x, x_0, \cdots, x_n\}$. Differentiating $E_n(x)$, we get

$$\begin{aligned} E_n'(x) &= h^n \frac{\mathrm{d}}{\mathrm{d}k}\left[k(k+1)(k+2)\cdots(k+n)\right]\frac{f^{(n+1)}(\xi)}{(n+1)!} \\ &\quad + h^{n+1}\frac{k(k+1)(k+2)\cdots(k+n)}{(n+1)!}f^{(n+2)}(\xi_1) \end{aligned} \tag{5.11}$$

where ξ and ξ_1 are two quantities depend on x and $\min\{x, x_0, \cdots, x_n\} < \xi, \xi_1 < \max\{x, x_0, \cdots, x_n\}$.

This expression gives the error in differentiation at any argument x. In particular, when $x = x_n$, i.e. when $k = 0$ then

$$\begin{aligned} E_n'(x_n) &= h^n \frac{f^{(n+1)}(\xi)}{(n+1)!}\frac{\mathrm{d}}{\mathrm{d}k}\left[k(k+1)\cdots(k+n)\right]_{k=0} + 0 \\ &= \frac{h^n n!}{(n+1)!}f^{(n+1)}(\xi) \qquad \left[\text{as } \frac{\mathrm{d}}{\mathrm{d}k}\left[k(k+1)\cdots(k+n)\right]_{k=0} = n!\right] \\ &= \frac{h^n f^{(n+1)}(\xi)}{n+1}, \end{aligned}$$

Example 5.2

A slider in a machine moves along a fixed straight rod. It's distance x (in *cm*) along the rod are given in the following table for various values of the time t (in second):

$(sec)\ t$	:	0	2	4	6	8
$(cm)\ x$	:	20	50	80	120	180

Find the velocity and acceleration of the slider at time $t = 8$.

Solution: The backward difference table is

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$
0	20				
		30			
2	50		0		
		30		10	
4	80		10		**0**
		40		**10**	
6	120		**20**		
		60			
8	**180**				

Here $t = 8 = t_n$ and $h = 2$,the velocity at $t = 8$ is given by

$$\begin{aligned}
v(8) = \frac{\mathrm{d}x}{\mathrm{d}t}\Big|_{t=8} &\approx \frac{1}{h}\left[\nabla y_4 + \frac{1}{2}\nabla^2 y_4 + \frac{1}{3}\nabla^3 y_4 + \frac{1}{4}\nabla^4 + \frac{1}{5}\nabla^5 + \cdots\right] \\
&= \frac{1}{2}\left[60 + \frac{1}{2}\times 20 + \frac{1}{3}\times 10 + 0\right] \\
&= 0.5 \times \left[70 + \frac{10}{3}\right] \\
&= 36.66667
\end{aligned}$$

Thus, the velocity of the slider when $t = 8$ is $v = 36.6667$.

The acceleration at $t = 8$ is

$$\begin{aligned}
a(8) = \frac{\mathrm{d}^2x}{\mathrm{d}t^2}\Big|_{t=8} &\approx \frac{1}{h^2}\left[\nabla^2 y_4 + \nabla^3 y_4 + \frac{11}{12}\nabla^4 y_4 + \frac{5}{6}\nabla^5 y_4 + \cdots\right] \\
&= \frac{1}{2}\left[20 + 10 + 0\right] \\
&= \frac{2}{2^2} \times \left[70 + \frac{10}{3}\right] \\
&= 7.50
\end{aligned}$$

Choice of differentiation formula

Choice of differentiation formula is same as choice of interpolation formula. That is, if the given argument is at the beginning of the table then the Newton's forward differentiation formula is used. Similarly, when the given argument is at the end of the table then the Newton's backward differentiation formula is used. The Lagrange's differentiation formula is used for any argument.

5.2 Integration (Trapezoidal and Simpson's rule)

Integration is a very common and fundamental tool of integral calculus. But, finding of integration is not easy for all kind of functions, even the function is known completely. Again, in many real life problems, only a set of values of x and y are available and we have to find the integration of such functions. In this situations, separate methods are developed and these methods are known as **numerical integration** or **quadrature**.

The problem of numerical integration is stated below:

Given a set of points $(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)$ *of a function* $y = f(x)$*. The problem is to find the value of the definite integral*

$$I = \int_{x_0}^{x_n} f(x)\,\mathrm{d}x.$$

The function $f(x)$ *is replaced by a suitable interpolating polynomial* $P_n(x)$*.*

Then the approximate value of the definite integral is then evaluated by the following formula

$$\int_{x_0}^{x_n} f(x)\,\mathrm{d}x \approx \int_{x_0}^{x_n} P_n(x)\,\mathrm{d}x.$$

A quadrature formula is said to be of **closed type**, if the limits of integration $a(= x_0)$ and $b(x_n)$ are taken as two interpolating points. If a and b are not included in the interpolating polynomial, then the formula is known as **open type** formula.

Newton's cotes Quadrature formula

Let $f(x)$ be an unknown function whose numerical values are given at $(n+1)$ equidistant points x_i in the interval $[a, b]$, where $x_i = x_0 + ih,\ i = 0, 1, \cdots, n$ such that $a = x_0$ and $b = x_n$, i.e., $b - a = nh$. Then

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &\approx \int_{x_0}^{x_n} P_n(x)\,\mathrm{d}x \\ &= h\int_0^n P_n(p)\,\mathrm{d}p, \qquad \text{where} \quad x = x_0 + ph,\ \text{and}\ \mathrm{d}x = h\,\mathrm{d}p.\end{aligned}$$

Expressing $P_n(p)$ by Newton's forward difference formula (5.12) , we get

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &\approx h\int_0^n \left[f_0 + p\Delta f_0 + \frac{p(p-1)}{2!}\Delta^2 f_0 + \cdots \frac{p(p-1)\cdots(p-n+1)}{n!}\Delta^n f_0\right]\mathrm{d}k \\ &= h\left[pf_0 + \frac{p^2}{2}\Delta f_0 + \frac{1}{2!}\left(\frac{p^3}{3} - \frac{p^2}{2}\right)\Delta^2 f_0 + \cdots + \text{last term}\right]_0^n\end{aligned}$$

Thus, the Newton's Cotes quadrature formula is given by:

$$\boxed{\int_a^b f(x)\,\mathrm{d}x \approx h\left[nf_0 + \frac{n^2}{2}\Delta f_0 + \frac{1}{2!}\left(\frac{n^3}{3} - \frac{n^2}{2}\right)\Delta^2 f_0 + \cdots + \text{last term}\right] \qquad (5.12)}$$

From this formula one can derive many simple formulae for different values of $n = 1, 2, 3, \cdots$. Some particular cases are discussed below.

5.2.1 Trapeziodal Rule

One of the simple quadrature formula is trapezoidal formula. To obtain this formula, we substitute $n = 1$ to the Equation (5.12). Obviously, we can fit a straight line through these two points or we can say, one can obtain only first order differences from two points then neglecting second and high order differences in Equation (5.12) we get,

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x \approx h\left[f_0 + \frac{1}{2}\Delta f_0\right] &= h\left[f_0 + \frac{1}{2}(f_1 - f_0)\right] \\ &= \frac{h}{2}(f_0 + f_1)\end{aligned}$$

Hence

$$\boxed{\int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{2}(f_0 + f_1) \qquad (5.13)}$$

This is known as **Trapezoidal quadrature formula** or the **Trapezoidal rule.**

Note that the formula is very simple and it gives a very rough approximation of the integral. So, if the interval $[a, b]$ is divided into some subintervals and the formula is applied to each of these subintervals, then much better approximate result may be obtained. This formula is known as **composite trapezoidal formula**, described below.

Composite Trapezoidal formula

Suppose the interval $[a, b]$ be divided into n equal subintervals as $a = x_0, x_1, x_2, \cdots, x_n = b$. That is, $x_i = x_0 + ih, i = 1, 2, \cdots, n$, where h is the length of the intervals.

Now, the trapezoidal formula is applied to each of the subintervals, and we obtained the composite formula as follows:

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &= \int_{x_0}^{x_1} f(x)\,\mathrm{d}x + \int_{x_1}^{x_2} f(x)\,\mathrm{d}x + \cdots + \int_{x_{n-1}}^{x_n} f(x)\,\mathrm{d}x \\ &\approx \frac{h}{2}(f_0 + f_1) + \frac{h}{2}(f_1 + f_2) + \cdots + \frac{h}{2}(f_{n-1} + f_n) \\ &= \frac{h}{2}\left[f_0 + 2(f_1 + f_2 + \cdots + f_{n-1}) + f_n\right].\end{aligned}$$

Therefore, the Composite trapezoidal rule is given by

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{2}\left[f_0 + 2\sum_{j=1}^{n-1} f_j + f_n\right]. \qquad (5.14)$$

Example 5.3

The following points were found empirically.

x	2.1	2.4	2.7	3.0	3.3	3.6
y	3.2	2.7	2.9	3.5	4.1	5.2

Use the trapezoidal rule to estimate

$$\int_{2.1}^{3.6} y\,\mathrm{d}x$$

Solution: Here we have $h = 0.3$, therefore we have

$$\begin{aligned}\int_{2.1}^{3.6} y\,\mathrm{d}x &= \frac{0.3}{2}\left[3.2 + 2 \times (2.7 + 2.9 + 3.5 + 4.1) + 5.2\right] \\ &= 5.22\end{aligned}$$

Error in Trapezoidal Rule

Since Trapezoidal rule is a numerical formula, it must have an error. The error in trapezoidal formula is calculated below.

Error of Trapezoidal in one step, i.e., Local error is

$$\begin{aligned} E_{L_0} &= \int_{x_0}^{x_1} \frac{p(p-1)}{2!} h^2 f''(\xi_1)\, \mathrm{d}p \\ &= \frac{h^3}{2} f''(\xi_1) \int_0^1 \left(p^2 - p\right) \mathrm{d}p = \quad = -\frac{h^3}{12} f''(\xi_1) \to \mathcal{O}(h^3), \end{aligned}$$

i.e., the local error of Trapezoidal rule is $\mathcal{O}(h^3)$. Now, Global error in Trapezoidal rule means the sum of all n local errors, which is

$$\sum_{i=0}^{n-1} E_{L_i} = -\frac{h^3}{12} \left[f''(\xi_1) + f''(\xi_2) + \cdots + f''(\xi_n)\right], x_i \le \xi_i \le x_{i+1}, i = 0, 1, 2 \cdots, n-1.$$

If we assume that $f''(x)$ is continuous on (a, b) then there exists some value of x in (a, b), say ξ such that $\sum_{i=1}^{n} f''(\xi_i) = n f''(\xi)$.

Therefore, the global error of Trapezoidal rule is given by

$$E_T = -\frac{h^3}{12} n f''(\xi) = -\frac{b-a}{12} h^2 f''(\xi) \to \mathcal{O}(h^2) \qquad \text{since } nh = b - a, \qquad (5.15)$$

which is one less to the order of local error.

Note: The error term in trapezoidal formula indicates that if the second and higher order derivatives of the function $f(x)$ vanish, then the trapezoidal formula gives exact result. That is, the trapezoidal formula gives exact result when the integrand is linear.

Geometrical interpretation of trapezoidal formula

In trapezoidal formula, the integrand $y = f(x)$ is replaced by the straight line, let AB joining the points (x_i, y_i) and (x_1, y_1) (see Figure 5.1). Then the area bounded by the

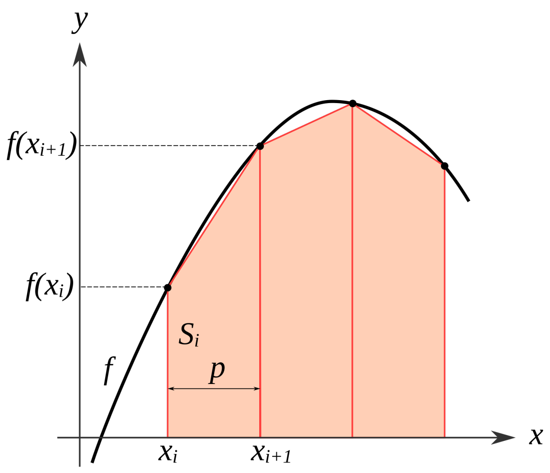


Figure 5.1: Geometrical interpretation of trapezoidal formula

curve $y = f(x)$, the ordinates $x = x_i$, $x = x_{i+1}$ and the x-axis is approximated by the area of the trapezium bounded by the straight line AB, the straight lines $x = xx_i$, $x = x_{i+1}$ and x-axis. That is, the value of the integration $\int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x$ obtained by the a trapezoidal formula is nothing but the area of the trapezium.

5.2.2 Simpson's Rule

When substituting $n = 2$ in the formula (5.12) and similar to $n = 1$, neglecting third and higher order differences in (5.12), we get

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &\approx h\left[2f_0 + \frac{2^2}{2}\Delta f_0 + \frac{1}{2!}\left(\frac{2^3}{3} - \frac{2^2}{2}\right)\Delta^2 f_0\right]\\ &= h\left[2f_0 + 2(f_1 - f_0) + \frac{1}{3}(f_2 - 2f_1 + f_0)\right]\\ &= \frac{h}{3}\left[f_0 + 4f_1 + f_2\right]\end{aligned}$$

Hence, the Simpson's 1/3 formula is given by

$$\boxed{\int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{3}\left[f_0 + 4f_1 + f_2\right] \qquad (5.16)}$$

Composite Simpson's 1/3 Rule

In the above formula, the interval of integration $[a, b]$ is divided into two subdivisions. Now, we divide the interval $[a, b]$ into n (even number) equal subintervals by the arguments $x_0, x_1, x_2, \cdots, x_n$, where $x_i = x_0 + ih,\ i = 1, 2, \cdots, n$.

$$\begin{aligned}
\int_a^b f(x)\,\mathrm{d}x &= \int_{x_0}^{x_2} f(x)\,\mathrm{d}x + \int_{x_2}^{x_4} f(x)\,\mathrm{d}x + \cdots + \int_{x_{n-2}}^{x_n} f(x)\,\mathrm{d}x \\
&\approx \frac{h}{3}\left(f_0 + 4f_1 + f_2\right) + \frac{h}{3}\left(f_2 + 4f_3 + f_4\right) + \cdots + \frac{h}{3}\left(f_{n-2} + 4f_{n-1} + f_n\right) \\
&= \frac{h}{3}\left[f_0 + 4\left(f_1 + f_3 + \cdots + f_{n-1}\right) + 2\left(f_2 + f_4 + \cdots + f_{n-2}\right) + f_n\right]. \\
&= \frac{h}{3}\left[f_0 + 4\,(\text{sum of } f_i \text{ with odd subscripts}) + 2\,(\text{sum of } f_i \text{ with even subscripts}) + f_n\right].
\end{aligned} \tag{5.17}$$

Thus, the **Simpson's 1/3 composite quadrature formula** is given by

$$\boxed{\int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{3}\left[f_0 + 4\sum_{j=1}^{\frac{n}{2}} f_{2j-1} + 2\sum_{j=1}^{\frac{n}{2}-1} f_{2j} + f_n\right]. \qquad (5.18)}$$

Note: Simpson's 1/3 -rule requires the division of the whole range into an even number of subintervals of width h.

Error in Simpson's 1/3 quadrature formula

The Local error expression in Simpson's 1/3 rule on the interval $[x_0, x_2]$ is given by

$$E_L \approx -\frac{h^5}{90} f^{(iv)}(\xi)$$

where $x_0 < \xi < x_2$ and the Global error in the composite Simpson's 1/3 rule is given by

$$E_T = -\frac{b-a}{180} h^4 f^{(iv)}(\xi), \quad \text{where} \quad f^{(iv)(\xi)} = \max\{f^{(iv)}(x_0), f^{(iv)}(x_1), \cdots, f^{(iv)}(x_n)\} \tag{5.19}$$

Example 5.4

Approximate the value of the integral

$$\int_0^1 e^{-x}\,\mathrm{d}x$$

using the composite Simpson's rule with $n = 4$ subintervals. Determine an upper bound for the absolute error using the error term. Verify that the absolute error is within this bound.

Solution: Here $n = 4$ and hence $h = \dfrac{1-0}{4} = \dfrac{1}{4}$. Using the Simpson's composite rule we get

$$\int_0^1 e^{-x}\,\mathrm{d}x \approx \frac{1}{3\times 4}\left(e^0 + 4e^{-0.25} + 2e^{-0.5} + 4e^{-0.75} + e^{-1}\right)$$
$$\approx 0.6321342$$

Next, we need to find an upper bound for the absolute error using the general error term for the composite Simpson's rule given by

$$E = -\frac{b-a}{180} h^4 f^{(4)}(\xi)$$

where $a < \xi < b$. Taking absolute values, and inserting $a = 0, b = 1$ and $h = 1/4$, the absolute error E is

$$E = -\frac{1}{180\times 4^4}|f^{(4)}(\xi)|.$$

Since $|f^{(4)}(\xi) = e^{-x}|$ is a decreasing positive function, we have the bound $|f^{(4)}(\xi)| < e^{-0} = 1$. Therefore the error bound is given by

$$E \le \frac{1}{46080} \approx 2.2\times 10^{-5}.$$

To verify that the bound holds here, we easily compute the exact value of the integral

$$\int_0^1 e^{-x}\,\mathrm{d}x = \left[-e^{-x}\right]_0^1 = e^{-1} - e^0 = 1 - e^{-1} = 0.6321206$$

Thus the actual absolute error is (correct to the digits used)

$$|0.6321206 - 0.6321342 \approx 1.4 \times 10^{-5}$$

so the absolute error is within our bound. The bound quite closely bounds the actual absolute error in this case.

Exercise 5.1

Evaluate $\int_1^3 (x+1)e^{x^2}\,\mathrm{d}x$ taking 10 intervals, by (i) Trapezoidal, and (ii) Simpson's 1/3 rule. **Ans.** (i) 6149.2217 (ii) 5557.9445

Simpson's 3/8 Rule:

We put $n = 3$ in Equation (5.12), and we will have the four points $(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$ so that all forward differences higher than third order in Equation (5.12) will be zero. Hence we obtain:

$$\begin{aligned}
\int_a^b f(x)\,\mathrm{d}x &\approx h\left[3f_0 + \frac{3^2}{2}\Delta f_0 + \frac{1}{2!}\left(\frac{3^3}{3} - \frac{3^2}{2}\right)\Delta^2 f_0 + \frac{1}{3!}\left(\frac{3^4}{4} - 3^3 + 3^2\right)\Delta^3 f_0\right] \\
&= h\left[3f_0 + \frac{9}{2}\Delta f_0 + \frac{9}{4}\Delta^2 f_0 + \frac{9}{24}\Delta^3 f_0\right] \\
&= \frac{3}{8}h\left[8f_0 + 12\Delta f_0 + 6\Delta^2 f_0 + \Delta^3 f_0\right] \\
&= \frac{3}{8}h\left[8f_0 + 12(f_1 - f_0) + 6(f_2 - 2f_1 + f_0) + (f_3 - 3f_2 + 3f_1 - f_0)\right] \\
&= \frac{3}{8}h\left[f_0 + 3f_1 + 3f_2 + f_3\right]
\end{aligned}$$

Therefore, the **Simpson's 3/8 Rule** is given by

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{3}{8}h\left[f_0 + 3f_1 + 3f_2 + f_3\right] \qquad (5.20)$$

Composite Simpson's 3/8 Rule

In the above formula, the interval of integration $[a, b]$ is divided into three subdivisions. Now, we divide the interval $[a, b]$ into n (a multiple of three) equal subintervals by the arguments $x_0, x_1, x_2, \cdots, x_n$, where $x_i = x_0 + ih,\ i = 1, 2, \cdots, n$.

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &= \int_{x_0}^{x_3} f(x)\,\mathrm{d}x + \int_{x_3}^{x_6} f(x)\,\mathrm{d}x + \cdots + \int_{x_{n-3}}^{x_n} f(x)\,\mathrm{d}x \\ &\approx \frac{3}{8}h\left(f_0 + 3f_1 + 3f_2 + f_3\right) + \frac{3}{8}h\left(f_3 + 3f_4 + 3f_5 + f_6\right) + \cdots \\ &\quad + \frac{3}{8}h\left(f_{n-3} + 3f_{n-2} + 3f_{n-1} + f_n\right) \\ &= \frac{3}{8}h\left[f_0 + 3\left(f_1 + f_2 + f_4 + f_5 + \cdots f_{n-1}\right) + 2\left(f_3 + f_6 + f_9 + \cdots + f_{n-3}\right) + f_n\right]\end{aligned}$$

Thus, the **composite Simpson's 3/8 quadrature formula** is given by

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{3}{8}h\left[f_0 + 3\left(f_1 + f_2 + f_4 + f_5 + \cdots f_{n-1}\right) + 2\left(f_3 + f_6 + \cdots + f_{n-3}\right) + f_n\right] \tag{5.21}$$

In using Equation (5.21) the number of subintervals should be taken as a multiple of 3.

Example 5.5

Approximate the value of the integral

$$\int_0^6 \frac{1}{1+x^2}\,\mathrm{d}x$$

by dividing the interval[0,6] in to six equal subintervals using:

i. Trapezoidal rule

ii. Simpson's 1/3 rule

iii. Simpson's 3/8 rule

Solution: Since we have six subintervals, i.e. $n = 6$, we can obtain the the step size h as

$$h = \frac{6-0}{6} = 1.$$

As a result we obtain the values of the function $f(x) = \frac{1}{1+x^2}$ at the nodal points

x_i	0	1	2	3	4	5	6
$f(x_i)$	1	0.5	0.2	0.1	0.0588	0.0385	0.027

i. **Trapezoidal rule:**

$$\begin{aligned}\int_0^6 \frac{1}{1+x^2}\,\mathrm{d}x &\approx \frac{1}{2}\left[f_0 + 2\left(f_1+f_2+f_3+f_4+f_5\right)+f_6\right]\\ &\approx \frac{1}{2}\left[1+2\left(0.5+0.2+0.1+0.0588+0.0385\right)+0.027\right]\\ &\approx 1.4108\end{aligned}$$

ii. **Simpson's** $1/3$ **rule:**

$$\begin{aligned}\int_0^6 \frac{1}{1+x^2}\,\mathrm{d}x &\approx \frac{1}{3}\left[f_0 + 4\left(f_1+f_3+f_5\right)+2\left(f_2+f_4\right)+f_6\right]\\ &\approx \frac{1}{3}\left[1+2\left(0.5+0.1+0.0385\right)+\left(0.2+0.0588\right)+0.027\right]\\ &\approx 1.3662\end{aligned}$$

iii. **Simpson's** $3/8$ **rule:**

$$\begin{aligned}\int_0^6 \frac{1}{1+x^2}\,\mathrm{d}x &\approx \frac{3}{8}\left[f_0 + 3\left(f_1+f_2+f_4+f_5\right)+2\left(f_3\right)+f_6\right]\\ &\approx \frac{3}{8}\left[1+2\left(0.5+0.2+0.0588+0.0385\right)+2\left(0.1+\right)+0.027\right]\\ &\approx 1.3571\end{aligned}$$

A more illuminating explanation of why Simpson's rule is "more accurate than it ought to be" can be had by looking at the extent to which it integrates polynomials exactly. This leads us to the notion of **degree of precision** for a quadrature rule.

Definition 5.1 Degree of Precision

The **degree of precision** of a quadrature formula is the positive integer n such that $E(P_m) = 0$ for all polynomials $P_m(x)$ of degree $\leq n$, but for which $E(P_{n+1}) \neq 0$ for some polynomial $P_{n+1}(x)$ of degree $n+1$.

Example 5.6

Determine the degree of precision of Simpson's rule.

Solution: It will suffice to apply the rule over the interval $[0, 2]$.

$$\int_0^2 \mathrm{d}x = 2 = \frac{1}{3}(1+4+1), \qquad \int_0^2 \mathrm{d}x = 2 = \frac{1}{3}(0+4+2)$$
$$\int_0^2 x^2\,\mathrm{d}x = \frac{8}{3} = \frac{1}{3}(0+4+4), \qquad \int_0^2 x^3\,\mathrm{d}x = 4 = \frac{1}{3}(0+4+8)$$

but,

$$\int_0^2 x^4\,\mathrm{d}x = \frac{32}{5} \neq \frac{1}{3}(0+4+16) = \frac{20}{3},$$

Therefore, the degree of precision is 3.

Chapter 6

Least Squares Method

6.1 Discrete Least Squares Approximation

Given a set of data points $(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)$, a normal and useful practice in many applications in statistics, engineering and other applied sciences is to construct a curve that is considered to be the "*best fit*" for the data, in some sense. So far, we have discussed two data-fitting techniques, *polynomial interpolation* and piecewise *polynomial interpolation*. Interpolation techniques, of any kind, construct functions that agree exactly with the data. That is, given points $(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)$, interpolation yields a function $f(x)$ such that $f(x_i) = y_i$ for $i = 1, 2, \cdots, m$.

However, fitting the data exactly may not be the best approach to describing the data with a function. We have seen that high-degree polynomial interpolation can yield oscillatory functions that behave very differently than a smooth function from which the data is obtained. Also, it may be pointless to try to fit data exactly, for if it is obtained by previous measurements or other computations, it may be erroneous. Therefore, we consider revising our notion of what constitutes a "*best fit*" of given data by a function.

One alternative approach to data fitting is to solve the minimax problem, which is the problem of finding a function $f(x)$ of a given form for which

$$\max_{1 \le i \le m} |f(x_i) - y_i|,$$

is minimized. However, this is a very difficult problem to solve.

Another approach is to minimize the total absolute deviation of $f(x)$ from the data. That is, we seek a function $f(x)$ of a given form for which

$$\sum_{i=1}^{m} |f(x_i) - y_i|,$$

is minimized. However, we cannot apply standard minimization techniques to this function, because, like the absolute value function that it employs, it is not differentiable.

This defect is overcome by considering the problem of finding $f(x)$ of a given form for which

$$\sum_{i=1}^{m} [f(x_i) - y_i]^2,$$

is minimized. This is known as the **least squares problem**. In summary, the problem of least squares is the following

Discrete Least-Squares Approximation Problem

Given a set of n discrete data points $(x_i, y_i), i = 1, 2, \cdots, m$.

Find the algebraic polynomial

$$P_n(x) = a_0 + a_1x + a_2x^2 + \cdots a_nx^n \quad (n \le m),$$

such that the error $E(a_0, a_1, a_2, \cdots, a_n)$ in the least-squares sense is minimized; that is,

$$E(a_0, a_1, a_2, \cdots, a_n) = \sum_{i=1}^{m} \left[a_0 + a_1x + a_2x^2 + \cdots a_nx^n - y_i\right]^2$$

is minimum.

Here $E(a_0, a_1, a_2, \cdots, a_n)$ is a function of $(n+1)$ variables: $a_0, a_1, a_2, \cdots a_n$

We will first show how this problem is solved for the case where $f(x)$ is a linear function of the form $f(x) = a_1x + a_0$, and then generalize this solution to other types of functions.

6.1.1 Linear Least-Squares

When $f(x)$ is linear, the least squares problem is the problem of finding constants a_0 and a_1 such that the function

$$E(a_0, a_1)) = \sum_{i=1}^{m} [y_i - a_0 + a_1 x]^2$$

is minimum. In order to minimize this function of a_0 and a_1, we must compute its partial derivatives with respect to a_0 and a_1 and set these partial derivatives to zero. This yields

$$\frac{\partial E(a_0, a_1)}{\partial a_0} = 2\sum_{i=1}^{m} [y_i - a_0 + a_1 x], \quad \frac{\partial E(a_0, a_1)}{\partial a_1} = 2\sum_{i=1}^{m} [y_i - a_0 + a_1 x] x_i.$$

At a minimum, both of these partial derivatives must be equal to zero. This yields the system of linear equations

$$\begin{aligned} na_0 + \left(\sum_{i=1}^{m} x_i\right) a_1 &= \sum_{i=1}^{m} y_i, \\ \left(\sum_{i=1}^{m} x_i a_0\right) + \left(\sum_{i=1}^{m} x_i^2\right) a_1 &= \sum_{i=1}^{m} x_i y_i, \end{aligned}$$

Using the formula for the inverse of a 2×2 matrix,

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix},$$

we obtain the solution

$$\begin{aligned} a_0 &= \frac{\left(\sum_{i=1}^{m} x_i^2\right)\left(\sum_{i=1}^{m} y_i\right) - \left(\sum_{i=1}^{m} x_i\right)\left(\sum_{i=1}^{m} x_i y_i\right)}{n\sum_{i=1}^{m} x_i^2 - \left(\sum_{i=1}^{m} x_i\right)^2} \\ a_1 &= \frac{n\sum_{i=1}^{m} x_i y_i - \left(\sum_{i=1}^{m} x_i\right)\left(\sum_{i=1}^{m} y_i\right)}{n\sum_{i=1}^{m} x_i^2 - \left(\sum_{i=1}^{m} x_i\right)^2} \end{aligned}$$

Example 6.1

We wish to find the linear function $y = a_1 x + a_0$ that best approximates the data shown in the following table, in the least-squares sense.

i	x_i	y_i
1	2.0774	3.3123
2	2.0774	3.8982
3	3.0125	4.6500
4	4.7092	6.5576
5	5.5016	7.5173
6	5.8704	7.0415
7	6.2248	7.7497
8	8.4431	11.0451
9	8.7594	9.8179
10	9.3900	12.2477

Using the summations

$$a_0 = \frac{380.5426 \times 73.8373 - 56.2933 \times 485.9487}{10 \times 380.5426 - 56.2933^2} = \frac{742.5703}{636.4906} = 1.1667,$$
$$a_1 = \frac{10 \times 485.9487 - 56.2933 \times 73.8373}{10 \times 380.5426 - 56.2933^2} = \frac{702.9438}{636.4906} = 1.1044$$

We conclude that the linear function that best fits this data in the least-squares sense is

$$y = 1.1044 + 1.1667x.$$

The data, and this function, are shown in Figure 6.1 bellow.

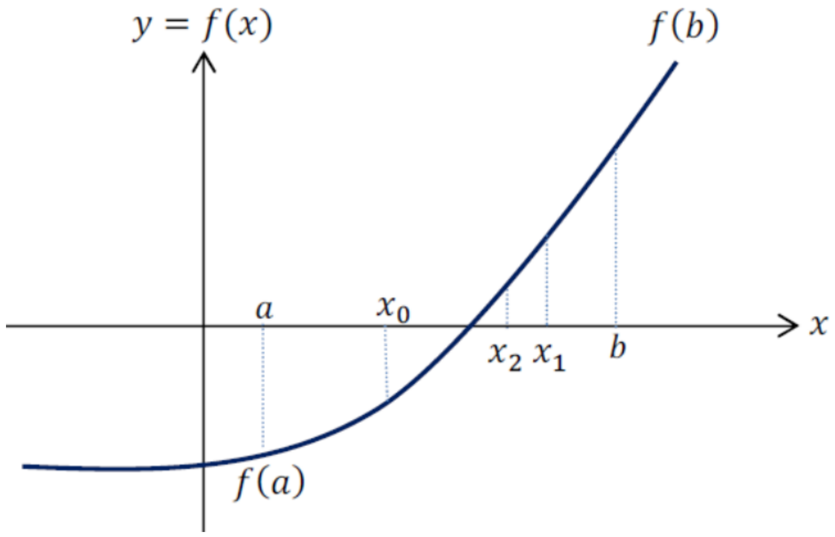


Figure 6.1: Data points (x_i, y_i) (circles) and least-squares line (solid line)

6.1.2 Non-linear least-squares (polynomial and exponential) Approximations

In the last lecture, we learned how to compute the coefficients of a linear function that best fit given data, in a least-squares sense. We now consider the problem of finding a polynomial of degree n or exponential function that gives the best least-squares fit.

Polynomial least-square method

As before, let $(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)$ be given data points that need to be approximated by a polynomial of degree n. We assume that $n < m - 1$, for otherwise, we can use polynomial interpolation to fit the points exactly.

Let the least-squares polynomial have the form

$$P_n(x) = \sum_{j=0}^{m} a_j x^j,$$

Our goal is to minimize the sum of squares of the deviations in $P_n(x)$ from each $y-$value,

$$E(\boldsymbol{a}) = \sum_{i=1}^{m} [P_n(x) - y_i]^2 = \sum_{i=1}^{m} \left[\sum_{j=0}^{n} a_j x_i^j - y_i \right]^2,$$

where $\boldsymbol{a}$ is a column vector of the unknown coefficients of $P_n(x)$,

$$\boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$$

Since $E(a_0, a_1, \cdots, a_n)$ is a function of the variables, $a_0, a_1, \cdots, a_n$, for this function to be minimum, we must have:

$$\frac{\partial E}{\partial a_j} = 0, \quad j = 0, 1, \cdots, n$$

Now, simple computations of these partial derivatives yield:

$$\begin{aligned}
\frac{\partial E}{\partial a_0} &= -2\sum_{i=1}^{m}\left(y_i - a_0 - a_1x_i - a_2x_i^2 - \cdots - a_nx_i^n\right)\\
\frac{\partial E}{\partial a_1} &= -2\sum_{i=1}^{m}x_i\left(y_i - a_0 - a_1x_i - a_2x_i^2 - \cdots - a_nx_i^n\right)\\
&\vdots\\
\frac{\partial E}{\partial a_n} &= -2\sum_{i=1}^{m}x_i^n\left(y_i - a_0 - a_1x_i - a_2x_i^2 - \cdots - a_nx_i^n\right)
\end{aligned}$$

Setting these equations to be zero, we have

$$\begin{aligned}
&a_0\sum_{i=1}^{m}1 + a_1\sum_{i=1}^{m}x_i + \cdots a_n\sum_{i=1}^{m}x_i^n = \sum_{i=1}^{m}y_i\\
&a_0\sum_{i=1}^{m}x_i + a_1\sum_{i=1}^{m}x_i^2 + \cdots a_n\sum_{i=1}^{m}x_i^{n+1} = \sum_{i=1}^{m}x_iy_i\\
&\vdots\\
&a_0\sum_{i=1}^{m}x_i^n + a_1\sum_{i=1}^{m}x_i^{n+1} + \cdots a_n\sum_{i=1}^{m}x_i^{2n} = \sum_{i=1}^{m}x_i^ny_i
\end{aligned}$$

Set

$$\begin{aligned}
s_k &= \sum_{i=1}^{m}x_i^k, \quad k = 0, 1, \cdots, 2n\\
b_k &= \sum_{i=1}^{m}x_i^ky_i, \quad k = 0, 1, \cdots, n
\end{aligned}$$

Using these notations, the above equations can be written as:

$$\begin{aligned}
&s_0a_0 + s_1a_1 + \cdots + s_na_n = b_0\\
&s_1a_0 + s_2a_1 + \cdots + s_{n+1}a_n = b_0\\
&\vdots\\
&s_na_0 + s_{n+1}a_1 + \cdots + s_{2n}a_n = b_0
\end{aligned} \tag{6.1}$$

This is a system of $(n+1)$ equations in $(n+1)$ unknowns $a_0, a_1, \cdots, a_n$. These equations are called **Normal Equations**. This system now can be solved to obtain these (n + 1) unknowns, provided a solution to the system exists. We will not show that this system has a unique solution if x_i's are distinct.

The system (6.1) can be written in the following matrix form:

$$\begin{bmatrix} s_0 & s_1 & \cdots & s_n \\ s_1 & s_2 & \cdots & s_{n+1} \\ \vdots & & \ddots & \vdots \\ s_n & s_{n+1} & \cdots & s_{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}, \tag{6.2}$$

or

$$\boldsymbol{Sa} = \boldsymbol{b}, \tag{6.3}$$

where

$$\boldsymbol{S} = \begin{bmatrix} s_0 & s_1 & \cdots & s_n \\ s_1 & s_2 & \cdots & s_{n+1} \\ \vdots & & \ddots & \vdots \\ s_n & s_{n+1} & \cdots & s_{2n} \end{bmatrix}, \quad \boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}.$$

Define

$$\boldsymbol{V} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ 1 & x_3 & x_3^2 & \cdots & x_3^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{bmatrix}$$

Then the system (6.3) has the form:

$$\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{a} = \boldsymbol{b}. \tag{6.4}$$

The matrix V is known as the **Vandermonde matrix**, and this matrix has full rank if x_i's are distinct. In this case, the matrix $\boldsymbol{S} = \boldsymbol{V}^T\boldsymbol{V}$ is *symmetric* and *positive definite* [**Exercise**] and is therefore nonsingular. Thus, if x_i's are distinct, the equation (6.3) has a unique solution.

Theorem 6.1 Existence and uniqueness of Discrete Least-Squares Solutions

Let $(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)$ be m distinct points. Then the discrete least-square approximation problem has a unique solution.

Example 6.2

We wish to find the quadratic function $y = a_2x^2 + a_1x + a_0$ that best approximates the data shown in the following table, in the least-squares sense.

i	x_i	y_i
1	2.0774	3.3123
2	2.0774	3.8982
3	3.0125	4.6500
4	4.7092	6.5576
5	5.5016	7.5173
6	5.8704	7.0415
7	6.2248	7.7497
8	8.4431	11.0451
9	8.7594	9.8179
10	9.3900	12.2477

First let's compute the summation

i	x_i	y_i	x_i^2	x_i^3	x_i^4	x_iy_i	$x_i^2y_i$
1	2.0774	2.7212	4.3156	8.9652	18.6243	5.6530	11.7436
2	2.3049	3.7798	5.3126	12.2449	28.2233	8.7121	20.0804
3	3.0125	4.8774	9.0752	27.3389	82.3585	14.6932	44.2632
4	4.7092	6.6596	22.1766	104.4339	491.8000	31.3614	147.6870
5	5.5016	10.5966	30.2676	166.5202	916.1278	58.2983	320.7337
6	5.8704	9.8786	34.4616	202.3034	1187.6016	57.9913	340.4323
7	6.2248	10.5232	38.7481	241.1994	1501.4180	65.5048	407.7544
8	8.4431	23.3574	71.2859	601.8743	5081.6849	197.2089	1665.0542
9	8.7594	24.0510	76.7271	672.0833	5887.0461	210.6723	1845.3632
10	9.3900	27.4827	88.1721	827.9360	7774.3192	258.0626	2423.2074
Sum	**56.2933**	**123.9275**	**380.5423**	**2864.8995**	**22969.2037**	**908.1578**	**7226.3193**

Thus, the matrix $\boldsymbol{S}$ and the vector $\boldsymbol{b}$ are given by

$$\boldsymbol{S} = \begin{bmatrix} 10 & 56.2933 & 380.5423 \\ 56.2933 & 380.5423 & 2864.8995 \\ 380.5423 & 2864.8995 & 22969.2037 \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} 123.9275 \\ 908.1578 \\ 7226.3193 \end{bmatrix}.$$

solving the normal equations

$$\boldsymbol{Sa} = \boldsymbol{b}$$

we obtain the coefficients

$$a_0 = 4.7681, \quad a_1 = -1.5193, \quad a_2 = 0.4251.$$

and conclude that the quadratic function that best fits this data in the least-squares sense is

$$y = 0.4251x^2 - 1.5193x + 4.7681.$$

The data, and this function, are shown in Figure 6.2.

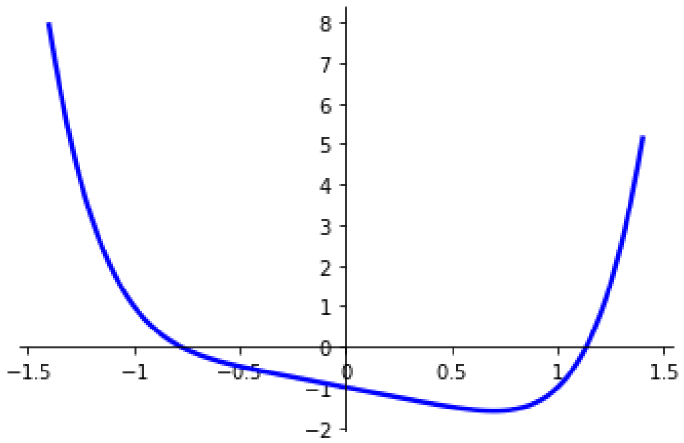


Figure 6.2: Data points (x_i, y_i) (circles) and least-squares line (solid line)

Exponential least-square method

Least-squares fitting can also be used to fit data with functions that are not linear combinations of functions such as polynomials. Suppose we believe that given data points can best be matched to an exponential function of the form $y = be^{ax}$, where the constants a and b are unknown. Taking the natural logarithm of both sides of this equation yields

$$\ln y = \ln b + ax.$$

If we define $z = \ln y$ and $c = \ln b$, then the problem of fitting the original data points $\{(x_i, y_i)\}_{i=1}^{m}$ with an exponential function is transformed into the problem of fitting the data points $\{(x_i, y_i)\}_{i=1}^{m}$ with a linear function of the form $c + ax$, for unknown constants a and c.

Similarly, suppose the given data is believed to approximately conform to a function of the form $y = bx^a$, where the constants a and b are unknown. Taking the natural logarithm of both sides of this equation yields

$$\ln y = \ln b + a \ln x.$$

If we define $z = \ln y$, $c = \ln b$ and $w = \ln x$, then the problem of fitting the original data points $\{(x_i, y_i)\}_{i=1}^m$ with a constant times a power of x is transformed into the problem of fitting the data points $\{(w_i, z_i)\}_{i=1}^m$ with a linear function of the form $c + aw$, for unknown constants a and c.

Example 6.3

We wish to find the exponential function $y = be^{ax}$ that best approximates the data shown in the following table , in the least-squares sense.

i	x_i	y_i
1	2.0774	1.4509
2	2.3049	2.8462
3	3.0125	2.1536
4	4.7092	4.7438
5	5.5016	7.7260

First let's compute the summation

i	x_i	y_i	$z_i = \ln y_i$	x_i^2	$x_i z_i$
1	2.0774	1.4509	0.3722	4.3156	0.7732
2	2.3049	2.8462	1.0460	5.3126	2.4109
3	3.0125	2.1536	0.7671	9.0752	2.3110
4	4.7092	4.7438	1.5568	22.1766	7.3315
5	5.5016	7.7260	2.0446	30.2676	11.2485
Sum	**17.6056**	18.9205	**5.7867**	**71.1475**	**24.0751**

By defining

$$\boldsymbol{S} = \begin{bmatrix} 5 & 17.6056 \\ 17.6056 & 71.1475 \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} 5.7867 \\ 24.0751 \end{bmatrix}$$

solving the normal equations

$$\boldsymbol{Sc} = \boldsymbol{b}$$

we obtain the coefficients

$$c_0 = -0.2653, \quad c_1 = 0.4040$$

and we obtain $a = c_1 = 0.4040, \quad b = e^{c_0} = e^{-0.2653} = 0.7670,$. and conclude that the exponential function that best fits this data in the least-squares sense is

$$y = 0.7670e^{0.4040x}.$$

The data, and this function, are shown in Figure 6.3.

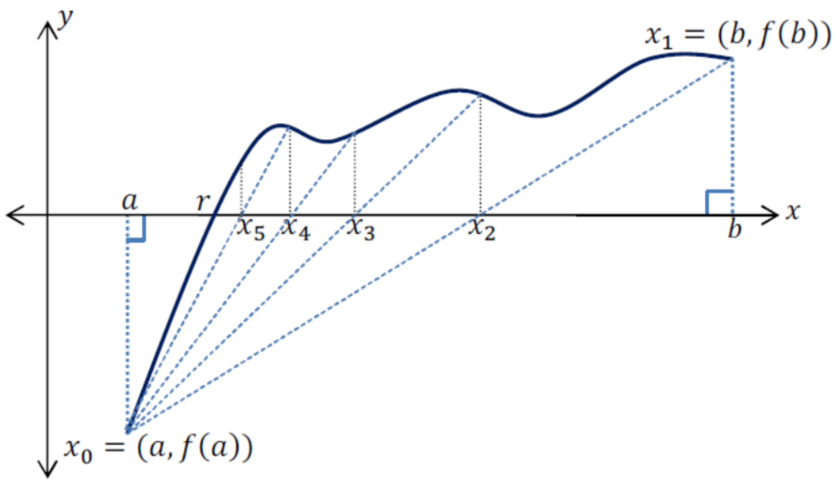


Figure 6.3: Data points (x_i, y_i) (circles) and least-squares line (solid line)

6.2 Continuous Least-Squares Approximation

In the previous section, we have described least-squares approximation to fit a set of **discrete data**. Here we first describe **continuous least-square approximations** of a function $f(x)$ by using polynomials and later in the subsequent sections using orthogonal polynomials and Fourier series.

6.2.1 Approximation by Polynomials

First, consider approximation by a polynomial with **monomial basis**: $\{1, x, x^2, \cdots, x^n\}$.

Least-Square Approximations of a Function Using Monomial Polynomials

Given a function $f(x)$, continuous on $[a, b]$, find a polynomial $P_n(x)$ of degree at most n:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_n x^n \quad (n \leq m),$$

such that the integral of the square of the error is minimized. That is,

$$E(a_0, a_1, a_2, \cdots, a_n) = \int_a^b \left[f(x) - P_n(x)\right]^2 \mathrm{d}x,$$

is minimized.

The polynomial $P_n(x)$ is called the **Least-Squares Polynomial.** For minimization, we must have

$$\frac{\partial E}{\partial a_i} = 0, \quad i = 0, 1, \cdots, n.$$

As before, these conditions will give rise to a system of $(n+1)$ **normal equations** in $(n+1)$ unknowns: $a_0, a_1, \cdots, a_n$. Solution of these equations will yield the unknowns: $a_0, a_1, \cdots, a_n$.

Setting up the Normal Equations

Since

$$E = \int_a^b \left[f(x) - \left(a_0 + a_1 x + a_2 x^2 + \cdots a_n x^n\right)\right]^2 \mathrm{d}x,$$

differentiating E with respect to each a_i results in

$$\begin{aligned}
\frac{\partial E}{\partial a_0} &= -2 \int_a^b \left[f(x) - a_0 - a_1 x - a_2 x^2 - \cdots - a_n x^n\right] \mathrm{d}x, \\
\frac{\partial E}{\partial a_1} &= -2 \int_a^b x \left[f(x) - a_0 - a_1 x - a_2 x^2 - \cdots - a_n x^n\right] \mathrm{d}x, \\
&\vdots \\
\frac{\partial E}{\partial a_n} &= -2 \int_a^b x^n \left[f(x) - a_0 - a_1 x - a_2 x^2 - \cdots - a_n x^n\right] \mathrm{d}x.
\end{aligned}$$

Thus, we have

$$\frac{\partial E}{\partial a_0} = 0 \quad \Longrightarrow \quad a_0 \int_a^b 1 \,\mathrm{d}x + a_1 \int_a^b x \,\mathrm{d}x + a_2 \int_a^b x^2 \,\mathrm{d}x + \cdots + a_n \int_a^b x^n \,\mathrm{d}x = \int_a^b f(x).$$

Similarly,

$$\frac{\partial E}{\partial a_i}=0 \quad\Longrightarrow\quad a_0\int_a^b x^i\,\mathrm{d}x + a_1\int_a^b \mathrm{d}x^{i+1}x\,\mathrm{d}x + a_2\int_a^b x^{i+2}\,\mathrm{d}x+\cdots+a_n\int_a^b x^{i+n}\,\mathrm{d}x=\int_a^b x^i f(x).$$

$$i=0,1,2,\cdots n.$$

So, the $(n+1)$ normal equations in this case are:

$$\begin{aligned}
i=0:&\quad a_0\int_a^b 1\,\mathrm{d}x + a_1\int_a^b x\,\mathrm{d}x + a_2\int_a^b x^2\,\mathrm{d}x+\cdots+a_n\int_a^b x^n\,\mathrm{d}x=\int_a^b f(x).\\
i=1:&\quad a_0\int_a^b x\,\mathrm{d}x + a_1\int_a^b x^2x\,\mathrm{d}x + a_2\int_a^b x^3\,\mathrm{d}x+\cdots+a_n\int_a^b x^{n+1}\,\mathrm{d}x=\int_a^b xf(x).\\
&\quad\vdots\\
i=n:&\quad a_0\int_a^b x^n\,\mathrm{d}x + a_1\int_a^b x^{n+1}x\,\mathrm{d}x + a_2\int_a^b x^{n+2}\,\mathrm{d}x+\cdots+a_n\int_a^b x^{2n}\,\mathrm{d}x=\int_a^b x^nf(x).
\end{aligned}$$

Denoting

$$\int_a^b x^i\,\mathrm{d}x=s_i,\quad i=0,1,2,\cdots,2n,\ \text{ and }\ b_i=\int_a^b x^if(x)\,\mathrm{d}x,\quad i=0,1,2,\cdots n,$$

the above $(n+1)$ equations can be written as

$$\begin{aligned}
&s_0a_0+s_1a_1+\cdots+s_na_n=b_0\\
&s_1a_0+s_2a_1+\cdots+s_{n+1}a_n=b_0\\
&\ \vdots\\
&s_na_0+s_{n+1}a_1+\cdots+s_{2n}a_n=b_0
\end{aligned}$$

or in matrix notation

$$\begin{bmatrix} s_0 & s_1 & \cdots & s_n\\ s_1 & s_2 & \cdots & s_{n+1}\\ \vdots & & \ddots & \vdots\\ s_n & s_{n+1} & \cdots & s_{2n}\end{bmatrix}\begin{bmatrix}a_0\\a_1\\\vdots\\a_n\end{bmatrix}=\begin{bmatrix}b_0\\b_1\\\vdots\\b_n\end{bmatrix}.$$

Hence, we have the system of normal equations

$$\boldsymbol{Sa}=\boldsymbol{b}, \tag{6.5}$$

where

$$\boldsymbol{S}=\begin{bmatrix} s_0 & s_1 & \cdots & s_n\\ s_1 & s_2 & \cdots & s_{n+1}\\ \vdots & & \ddots & \vdots\\ s_n & s_{n+1} & \cdots & s_{2n}\end{bmatrix},\quad \boldsymbol{a}=\begin{bmatrix}a_0\\a_1\\\vdots\\a_n\end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix}b_0\\b_1\\\vdots\\b_n\end{bmatrix}.$$

The solution of Equation (6.5) will yield the coefficients $a_0, a_1, \cdots, a_n$ of the least-squares polynomial $P_n(x)$.

Method-1 Least-Squares Approximation using Monomial Polynomials

Inputs: (i) $f(x)$ - A continuous function on $[a, b]$.

(ii) n - The degree of the desired least-squares polynomial

Output: The coefficients $a_0, a_1, \cdots, a_n$ of the desired least-squares polynomial: $P_n(x) = a_0 + a_1 x + \cdots + a_n xn$.

Step 1: **Compute** $s_0, s_1, \cdots, s_{2n}$:

```
for $i = 0, 1, 2, \cdots, 2n$ do
    $s_i = \int_a^b x^i \, dx$
end
```

Step 2: **Compute** $b_0, b_1, \cdots, b_n$:

```
for $i = 0, 1, 2, \cdots, n$ do
    $b_i = \int_a^b x^i f(x) \, dx$
end
```

Step 3: Form the matrix $\boldsymbol{S}$ from the numbers $s_0, s_1, \cdots, s_{2n}$ and the vector $\boldsymbol{b}$ from the numbers$b_0, b_1, \cdots, b_n$, i.e.,

$$\boldsymbol{S} = \begin{bmatrix} s_0 & s_1 & \cdots & s_n \\ s_1 & s_2 & \cdots & s_{n+1} \\ \vdots & & \ddots & \vdots \\ s_n & s_{n+1} & \cdots & s_{2n} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}.$$

Step 4: **Solve** the $(n+1) \times (n+1)$ system of equations for $a_0, a_1, \cdots, a_n$:

$$\boldsymbol{S}\boldsymbol{a} = \boldsymbol{b}, \quad \text{where} \quad \boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}.$$

A Special Case:

Let the interval be $[0, 1]$. Then

$$s_i = \int_0^1 x^i \, \mathrm{d}x = \frac{1}{i+1}, \quad i = 0, 1, 2, \cdots, 2n.$$

Thus, in this case the matrix of the normal equations

$$\boldsymbol{S} = \begin{bmatrix} 1 & \frac{1}{2} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \vdots & & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \cdots & \frac{1}{2n} \end{bmatrix},$$

which is a **Hilbert Matrix**. It is well-known to be **ill-conditioned.**

Example 6.4

Find Linear and Quadratic least-squares approximations to $f(x) = e^x$ on $[-1, 1]$.

Solution:

Linear approximation: $n = 1$ $P_1(x) = a_0 + a_1 x$

Step 1:

$$s_0 = \int_{-1}^1 1 \, \mathrm{d}x = 2,$$

$$s_1 = \int_{-1}^1 x \, \mathrm{d}x = \left[\frac{x^2}{2}\right]_{-1}^1 = \frac{1}{2} - \left(\frac{1}{2}\right) = 0,$$

$$s_2 = \int_{-1}^1 x^2 \, \mathrm{d}x = \left[\frac{x^3}{3}\right]_{-1}^1 = \frac{1}{3} - \left(\frac{-1}{3}\right) = \frac{2}{3}.$$

Step 2:

$$b_0 = \int_{-1}^1 1 \, \mathrm{d}e^x = [e^x]_{-1}^1 = e - \frac{1}{e} = 2.3504,$$

$$b_1 = \int_{-1}^1 x e^x \, \mathrm{d}x = \frac{2}{e} = 0.7358,$$

Step 3: From the matrix $\boldsymbol{S}$ and vector $\boldsymbol{b}$:

$$\boldsymbol{S} = \begin{bmatrix} 2 & 0 \\ 0 & \frac{2}{3} \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 2.3504 \\ 0.7358 \end{bmatrix}$$

Step 4: Solve the normal system is:

$$\begin{bmatrix} 2 & 0 \\ 0 & \frac{2}{3} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 2.3504 \\ 0.7358 \end{bmatrix}$$

This gives

$$a_0 = 1.1752, \quad a_1 = 1.1037$$

The linear least-squares polynomial $P_1(x) = 1.1752 + 1.1037x$.

Accuracy Check:

$$P_1(0.5) = 1.7270, \quad e^{0.5} = 1.6487.$$

Relative Error:

$$\frac{|e^{0.5} - P_1(0.5)|}{|e^{0.5}|} = \frac{|1.6487 - 1.7270|}{|1.6487|} = 0.0475.$$

Quadratic fitting n=2; $P_2(x) = a_0 + a_x + a_2x^2$

Step 1: Compute s_i's

$$s_0 = 2, \quad s_1 = 0, \quad s_2 = \frac{2}{3}$$
$$s_3 = \int_{-1}^{1} x^3 \, \mathrm{d}x = \left[\frac{x^4}{4}\right]_{-1}^{1} = \frac{1}{4} - \left(\frac{1}{4}\right) = 0,$$
$$s_4 = \int_{-1}^{1} x^4 \, \mathrm{d}x = \left[\frac{x^5}{5}\right]_{-1}^{1} = \frac{1}{5} - \left(\frac{-1}{5}\right) = \frac{2}{5}.$$

Step 2: Compute b_i's

$$b_0 = 2.3504, \quad b_1 = 0.7358,$$
$$b_2 = \int_{-1}^{1} x^2 e^x \, \mathrm{d}x = e - \frac{5}{e} = 0.8789.$$

Step 3: From the matrix $\boldsymbol{S}$ and vector $\boldsymbol{b}$:

$$\boldsymbol{S} = \begin{bmatrix} 2 & 0 & \frac{2}{3} \\ 0 & \frac{2}{3} & 0 \\ \frac{2}{3} & 0 & \frac{2}{5} \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 2.3504 \\ 0.7358 \\ 0.8789 \end{bmatrix}$$

Step 4: Solve the normal system is:

$$\begin{bmatrix} 2 & 0 & \frac{2}{3} \\ 0 & \frac{2}{3} & 0 \\ \frac{2}{3} & 0 & \frac{2}{5} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 2.3504 \\ 0.7358 \\ 0.8789 \end{bmatrix}$$

This gives

$$a_0 = 0.9963, \quad a_1 = 1.1037, \quad a_2 = 0.5368.$$

The linear least-squares polynomial $P_2(x) = 0.9963 + 1.1037x + 0.5368x^2$.

Accuracy Check:

$$P_2(0.5) = 1.6889, \quad e^{0.5} = 1.6487.$$

Relative Error:

$$\frac{|e^{0.5} - P_1(0.5)|}{|e^{0.5}|} = \frac{|1.6487 - 1.6889|}{|1.6487|} = 0.0204.$$

Chapter 7

Numerical methods for ODEs

7.1 Introduction

Consider $y(x)$ to be a function of a variable x. **A first order Ordinary differential equation** is an equation relating y, x and its first order derivatives. The most general form is :

$$F(x, y(x), y'(x)) = 0$$

The variable y is known as a dependent variable and x is independent variable. The equation is of first order as it is the order of highest derivative present in the equation. Sometimes it is possible to rewrite the equation in the form

$$y'(x) = f(x, y(x)). \tag{7.1}$$

$y = g(x)$ is a solution of the first order differential equation (7.1) means

i) $y(x)$ is differentiable

ii) Substitution of $y(x)$ and $y'(x)$ in (7.1) satisfies the differential equation identically.

The differential equations are commonly obtained as mathematical representations of many real world problems. Then the solution of the underlying problem lies in the solution of differential equation. Finding solution of the differential equation is then critical to that real world problem. In This chapter we are concerned with the problem of solving differential equations, **numerically**.

7.2 Initial Value problem

At first we are concentrate on the so-called **first order Initial Value Problem (IVP)**. A first order differential equation together with specified initial condition at $x = x_0$ is defined as

$$y'(x) = f(x, y(x)) \text{ with } y(x_0) = y_0. \tag{7.2}$$

There exist several methods for finding solutions of differential equations. However, all differential equations are not solvable. The following well known theorem from theory of differential equations establishes the **existence** and **uniqueness** of solution of the IVP:

Theorem 7.1 Existence and Uniqueness Theorem

Let $f(x, y(x))$ be continuous in a domain $D = \{(x, y(x)) : a \le x \le b, c \le y \le d\} \subseteq R^2$. If f satisfies **Lipschitz condition** on the variable y and (x_0, y_0) in D, then IVP has a unique solution $y = y(x)$ on the some interval $a \le x \le b$. (The function f satisfies **Lipschitz condition** means that there exists a positive constant L such that $|f(x, y) - f(x, w)| < L|y - w|$)

The theorem gives conditions on function $f(x, y)$ for **existence** and **uniqueness of the solution**. But the solution has to be obtained by available methods. It may not be possible to obtain analytical solution (in closed form) of a given first order differential equation by known methods even when the above theorem guarantees its existence. Sometimes it is very difficult to obtain the solution. In such cases, the approximate solution of given differential equation can be obtained using Numerical methods.

Discretization

The aim of this chapter is to device numerical methods to obtain an approximate solution of the initial value problem (7.2) at only a **discrete set of point**. That is, if we are interested in obtaining solution for (7.2) in an interval $[a, b]$, then we first discretize the interval as

$$a = x_0 < x_1 < \cdots < x_N = b,$$

where each point x_i, $\quad i = 0, 1, \cdots, N$ is called a **node**. Unless otherwise stated, we always assume that the nodes are equally spaced. That is,

$$x_i = x_0 + ih, i = 0, 1, \cdots N$$

for a sufficiently small positive real number h, called **the stepsize**. We use the notation for the **approximate solution** as

$$y_i = y_h(x_i) \approx y(x_i), \qquad i = 0, 1, \cdots, N.$$

In both cases, let us assume that we somehow have found solutions $y_i \approx y(x_i)$, for $i = 0, 1, \cdots, n$, and we want to find an approximation $y_{n+1} \approx y(x_n + 1)$ where $x_{n+1} = x_n + h$. Basically, there are two different classes of methods in practical use.

1). **One-step methods**: Only y_n is used to find the approximation y_{n+1} . One-step methods usually require more than one function evaluation pre-step.

They can all be put in a general abstract form

$$y_{n+1} = y_n + h\phi(x_n, y_n; h).$$

2.) **Linear multistep methods:** y_{n+1} is approximated from $y_{n-k+1}, \cdots, y_n$.

In the next section a very basic one-step method known as Euler method is being discussed.

7.2.1 Euler's Method

Euler's method is the natural starting point for any discussion of numerical methods for IVPs. Although it is not the most accurate of the methods we study, it is by far the simplest, and much of what we learn from analyzing Euler's method in detail carries over to other methods without a lot of difficulty.

Euler's Method assumes our solution is written in the form of a Taylor's Series (**??**). This gives us a reasonably good approximation if we take plenty of terms, and if the value of h is reasonably small.

For Euler's Method, we just take the first 2 terms only.

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2!}y''(\eta), \qquad \text{where } x < \eta < x+h.$$

Using the fact that $y' = f(x, y(x))$, we obtain a numerical scheme by truncating the Taylor series after the second term.

Method-2 Euler's method

For approximating the solution to the initial-value problem

$$y'(x) = f(x,y), \qquad y(x_0) = y_0$$

at the points $x_{i+1} = x_0 + ih$ $(i = 0, 1, 2, \cdots, N)$ on $[x_0, x_N]$ is

$$y_{i+1} = y_i + hf(x_i, y_i). \qquad \text{with } y_0 = y(x_0). \tag{7.3}$$

Example 7.1

Consider the initial-value problem

$$y' = y - x, \qquad y(0) = \frac{1}{2}$$

Use Euler's method (a) with $h = 0.1$ and (b) with $h = 0.05$ to obtain an approximation to $y(1)$. Given the exact solution to the initial value problem is

$$y(x) = x + 1 - \frac{1}{2}e^x$$

compare the errors in the two approximations to y(1).

7.2.2 Other Examples of One-Step Method

Assume that x_n, y_n is known. The exact solution $y(x_{n+1})$ with $x_{n+1} = x_n + h$ of equation (7.1) passing through this point is given by

$$y(x_n + h) = y_n + \int_{x_n}^{x_{n+1}} y'(\tau)d\tau = y_n + \int_{x_n}^{x_n+h} f(\tau, y(\tau))d\tau. \tag{7.4}$$

The idea is to find approximations to the last integral. The simplest idea is to use $f(\tau, y(\tau)) \approx f(x_n, y_n)$, in which case we get the Euler method again:

$$y_{n+1} = y_n + hf(x_n, y_n).$$

Modified Euler Method

The integral (7.4) can be approximated by **Midpoint rule** or **Rectangular rule** of integration as

$$\int_{x_n}^{x_n+h} f(\tau, y(\tau))d\tau = h\left[f\left(x_n + h/2, y\left(x_n + h/2\right)\right)\right].$$

By inserting the forward Euler step for the missing value $y(x_n + h/2)$

$$y(x_n + h/2) = y_n + \frac{h}{2}f(x_n, y_n)$$

we obtain the **Modified Euler's method** as

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

which we can rewrite as:

Method-3 **Modified Euler's Method**

$$\begin{aligned} y_{n+1} &= y_n + k_2, \\ k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + h/2, y_n + k_1/2) \end{aligned} \tag{7.5}$$

Improved Euler Method

The above numerical method can be improved for a more accurate solution by using the trapezoidal rule instead of using the rectangular rule. Approximating the integral in Equation (7.4) by **Trapezoidal rule** results in

$$y_{n+1} = y_n + \int_{x_n}^{x_n+h} f(x_n, y(\tau))d\tau = y_n + \frac{h}{2}\left(f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right). \tag{7.6}$$

Here y_{n+1} is available by solving a (usually) nonlinear system of equations. Such methods are called **implicit methods.** To avoid this extra difficulty, we could replace y_{n+1} on the right hand side by the approximation from Eulers method. The method that we consider here is an example of what is called a **predictor-corrector method**. The idea is to use the formula from Euler's method to obtain a first approximation to the solution $y(x_{n+1})$, we denote this approximation as

$$y_{n+1} = y_n + hf(x_n, y_n).$$

Hence, Equation (7.7) becomes

$$y_{n+1} = y_n + \frac{h}{2}\left(f(x_n, y_n) + f(x_n + h, y_n + hf(x_n, y_n))\right), \tag{7.7}$$

which can be rewrite as:

Method-4 **Improved Euler Method**

$$y_{n+1} = y_n + \frac{1}{2}\left(k_1 + k_2\right),$$

where

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + h, y_n + k_1)$$

Example 7.2

Apply (i) Euler's method, (ii) Modified Euler's Method and (ii) Improved Euler's method to compute $y(x)$ at $x = 0.3$ with step-size $h = 0.1$ for the initial value problem:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = 2x\left(1 - y\right), \quad y(0) = 2$$

Compare the errors $e_n = |y(x_n) - y_n|$ at each step with the exact solution $y(x) = 1 + e^{-x^2}$.

Solution: Since we are solving the problem on the interval $[0, 0.3]$ with step-size $h = 0.1$ we have the nodes $x_0 = 0, x_1 = 0.1, x_2 = 0.2$ and $x_3 = 0.3$ and the initial value $y_0 = 2$. In addition we have $f(x, y) = 2x(1 - y)$.

i.) **Euler's method:** is given by

$$y_{n+1} = y_n + hf(x_n, y_n),$$

where

$$f(x_n, y_n) = 2x_n\left(1 - y_n\right)$$

When $n = 0$, i.e., at $x = 0.1$ we have

$$\begin{aligned} y_1 &= y_0 + hf(x_0, y_0) = y_0 + h\left[2x_0\left(1 - y_0\right)\right] \\ &= 2 + 0.1 \times [2 \times 0 \times (1 - 2)] \\ &= 2 \end{aligned}$$

When $n = 1$, i.e., at $x = 0.2$ we have

$$\begin{aligned} y_2 &= y_1 + hf(x_1, y_1) = y_1 + h\left[2x_1\left(1 - y_1\right)\right] \\ &= 2 + 0.1 \times [2 \times 2 \times (1 - 2)] \\ &= 1.98 \end{aligned}$$

When $n = 2$, i.e., at $x = 0.3$ we have

$$\begin{aligned} y_3 &= y_2 + hf(x_2, y_2) = y_2 + h\left[2x_2\left(1 - y_2\right)\right] \\ &= 1.98 + 0.1 \times [2 \times 1.98 \times (1 - 1.98)] \\ &= 1.9408 \end{aligned}$$

ii.) **Modified Euler's method:** is given by

$$y_{n+1} = y_n + k2,$$

where

$$\begin{aligned} k1 &= hf(x_n, y_n) \\ k2 &= hf(x_n + h/2, y_n + k1/2) \end{aligned}$$

When $n = 0$, i.e., at $x = 0.1$ we have

$$\begin{aligned}
k_1 &= hf(x_0, y_0)\\
&= h\left(2x_0\left(1 - y_0\right)\right)\\
&= 0.1 \times (2 \times 0 \times (1 - 2))\\
&= 0\\
k_2 &= hf(x_0 + h/2, y_0 + k1/2)\\
&= h\left(2(x_0 + h/2)\left(1 - (y_0 - k1/2)\right)\right)\\
&= 0.1 \times (2 \times (0 + 0.1/2) \times (1 - (2 + 0/2)))\\
&= -0.01\\
y_1 &= y_0 + k2\\
&= 2 + (-0.01)\\
&= 1.99
\end{aligned}$$

When $n = 1$, i.e., at $x = 0.2$ we have

$$\begin{aligned}
k_1 &= hf(x_1, y_1) = h\left(2x_1\left(1 - y_1\right)\right)\\
&= 0.1 \times [2 \times 0.1 \times (1 - 1.99)]\\
&= -0.0198
\end{aligned}$$

$$\begin{aligned}
k_2 &= hf(x_1 + h/2, y_1 + k1/2)\\
&= h\left(2(x_1 + h/2)\left(1 - (y_1 - k1/2)\right)\right)\\
&= 0.1 \times [2 \times (0.1 + 0.1/2) \times (1 - (1.99 + (-0.0198)/2))]\\
&= -0.0294\\
y_2 &= y_0 + k2\\
&= 1.99 + (-0.0294)\\
&= 1.9606
\end{aligned}$$

When $n = 2$, i.e., at $x = 0.3$ we have

$$\begin{aligned}
k_1 &= hf(x_2, y_2) = h\left(2x_2\left(1 - y_2\right)\right) \\
&= 0.1 \times [2 \times 0.2 \times (1 - 1.9606)] \\
&= -0.0384 \\
k_2 &= hf(x_2 + h/2, y_2 + k2/2) \\
&= h\left(2(x_2 + h/2)\left(1 - (y_2 - k1/2)\right)\right) \\
&= 0.1 \times [2 \times (0.2 + 0.1/2) \times (1 - (1.9606 + (-0.0384)/2))] \\
&= -0.0471 \\
y_3 &= y_0 + k2 \\
&= 1.9606 + (-0.0384) \\
&= 1.9135
\end{aligned}$$

iii.) **Improved Euler's method:** is given by

$$y_{n+1} = y_n + \frac{1}{2}\left(k1 + k2\right),$$

where

$$\begin{aligned}
k1 &= hf(x_n, y_n) \\
k2 &= hf(x_n + h, y_n + k1)
\end{aligned}$$

When $n = 0$, i.e., at $x = 0.1$ we have

$$\begin{aligned}
k_1 &= hf(x_0, y_0) \\
&= h\left(2x_0\left(1 - y_0\right)\right) \\
&= 0.1 \times (2 \times 0 \times (1 - 2)) \\
&= 0
\end{aligned}$$

$$\begin{aligned}
k_2 &= hf(x_0+h, y_0+k1)\\
&= h\left(2(x_0+h)\left(1-(y_0-k1)\right)\right)\\
&= 0.1\times(2\times(0+0.1)\times(1-(2+0)))\\
&= -0.02\\
y_1 &= y_0+\frac{1}{2}\left(k1+k2\right)\\
&= 2+0.5\times(0+(-0.02))\\
&= 1.99
\end{aligned}$$

When $n=1$, i.e., at $x=0.2$ we have

$$\begin{aligned}
k_1 &= hf(x_1,y_1) = h\left(2x_1\left(1-y_1\right)\right)\\
&= 0.1\times[2\times0.1\times(1-1.99)]\\
&= -0.0198\\
k_2 &= hf(x_1+h, y_1+k1)\\
&= h\left(2(x_1+h)\left(1-(y_1-k1)\right)\right)\\
&= 0.1\times[2\times(0.1+0.1)\times(1-(1.99+(-0.0198)))]\\
&= -0.0388\\
y_2 &= y_0+\frac{1}{2}\left(k1+k2\right)\\
&= 1.99+0.5\times(-0.0198-0.0388)\\
&= 1.9607
\end{aligned}$$

When $n=2$, i.e., at $x=0.3$ we have

$$\begin{aligned}
k_1 &= hf(x_2,y_2) = h\left(2x_2\left(1-y_2\right)\right)\\
&= 0.1\times[2\times0.2\times(1-1.9607)]\\
&= -0.0384\\
k_2 &= hf(x_2+h, y_2+k2)\\
&= h\left(2(x_2+h)\left(1-(y_2-k1)\right)\right)\\
&= 0.1\times[2\times(0.2+0.1)\times(1-(1.9607+(-0.0384)))]\\
&= -0.0553\\
y_1 &= y_0+\frac{1}{2}\left(k1+k2\right)\\
&= 1.9606+0.5\times(-0.0384-0.0553) = 1.9138
\end{aligned}$$

Summary

x	Exact value	Euler's	Modified	Improved
0.1	1.9900	2	1.99	1.99
0.2	1.9608	1.9800	1.9606	1.9607
0.3	1.9139	1.9408	1.9135	1.9138
Error	0	-0.0269	0.0004	0.0001

7.2.3 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 need a very small step size. One way to get a better accuracy is to include the higher order terms in the Taylor expansion in the formula. But the higher order terms involve higher derivatives of y. The Runge-Kutta methods attempt to obtain greater 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. A general Runge Kutta algorithm is given as

$$y_{n+1} = y_n + h\phi(x_n, y_n, h) \tag{7.8}$$

The function ϕ is termed as *increment function*. The m^{th} order Runge-Kutta method gives accuracy of order $O(h^m)$. The function ϕ is chosen in such a way that when expanded the right hand side of (7.8) matches with the Taylor series up to desired order. This means that for a second order Runge-Kutta method the right side of (7.8) matches up to second order terms of Taylor series.

Second Order Runge-Kutta

The Second order Runge Kutta methods are known as **RK2** methods. For the derivation of second order Runge Kutta methods, it is assumed that ϕ is the weighted average of two functional evaluations at suitable points in the interval $[x_n, x_{n+1}]$, i.e., $\phi(x_n, y_n, h) = w_1k_1 + w_2k_2$. Thus, we have:

$$y_{n+1} = y_n + [w_1k_1 + w_2k_2] \tag{7.9}$$

where

$$k_1 = hf(x_n, y_n),\ k_2 = hf(x_n + \alpha h, y_n + \beta k_1) \tag{7.10}$$

Here w_1, w_2, α and β are constants to be determined so that equation (7.9) agrees with the Taylor algorithm of a possible higher order.

Now, let's write down the Taylor series expansion of y in the neighborhood of x_n correct to the h^2 term i.e

$$y(x_{n+1}) = y(x_n) + hf(x_n, y(x_n)) + \frac{h^2}{2} f'(x_n, y(x_n)) + O(h^3) \tag{7.11}$$

Then, using chain rule for the derivative $f'(x_n, y(x_n))$ we get

$$f'(x_n, y(x_n)) = \frac{\partial f(x_n, y(x_n))}{\partial x} + f(x_n, y(x_n))\frac{\partial f(x_n, y(x_n))}{\partial y},$$

Thus we have

$$y(x_{n+1}) = y(x_n) + hf(x_n, y(x_n)) + \frac{h^2}{2}\left[\frac{\partial f(x_n, y(x_n))}{\partial x} + f(x_n, y(x_n))\frac{\partial f(x_n, y(x_n))}{\partial y}\right] + O(h^3) \tag{7.12}$$

In addition, equation (7.9) and (7.10) can be rewritten as:

$$\begin{aligned} y_{n+1} &= y_n + w_1 hf(x_n, y_n) + w_2 hf(x_n + \alpha h, y_n + \beta hf(x_n, y_n)) \\ &= y_n + w_1 hf(x_n, y_n) + w_2 h\left[f(x_n, y_n) + \alpha h\frac{\partial f(x_n, y_n)}{\partial x} + \beta hf(x_n, y_n)\frac{f(x_n, y_n)}{\partial y} + O(h^2)\right] \\ &= y_n + h(w_1 + w_2)f(x_n, y_n) + h^2\left[w_2\alpha\frac{\partial f(x_n, y_n)}{\partial x} + w_2\beta f(x_n, y_n)\frac{\partial f(x_n, y_n)}{\partial y}\right] + O(h^3) \end{aligned}$$

Therefore,

$$y_{n+1} = y_n + h(w_2 + w_2)f(x_n, y_n) + h^2\left[w_2\alpha\frac{\partial f(x_n, y_n)}{\partial x} + w_2\beta f(x_n, y_n)\frac{\partial f(x_n, y_n)}{\partial y}\right] + O(h^3). \tag{7.13}$$

Assuming $y(x_n) \approx y_n$ and comparing equations (7.12) and (7.13) yields

$$w_1 + w_2 = 1, \qquad w_2\alpha = \frac{1}{2} \qquad \text{and} \qquad w_2\beta = \frac{1}{2}. \tag{7.14}$$

Observe that four unknowns are to be evaluated from three equations. Accordingly many solutions are possible for (7.14). Two examples of second-order Runge-Kutta methods of the form (7.9) and (7.10) are the modified Euler method and the improved Euler method.

(a) **The modified Euler method** In this case we take $\beta = \frac{1}{2}$ obtain

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

(b) **The improved Euler method, usually called RK2** This is arrived at by choosing $\beta = 1$ which gives

$$\begin{aligned} k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + h, y_n + k_1), \\ y_{n+1} &= y_n + \frac{1}{2}(k_1 + k_2). \end{aligned}$$

Fouth Order Runge-Kutta

A similar but more complicated analysis is used to construct Runge-Kutta methods of higher order. One of the most frequently used methods of the Runge-Kutta family is often known as the classical **fourth-order** method, **RK4**, given by:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{7.15}$$

where

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\ k_3 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\ k_4 &= hf(x_n + h, y_n + k_3). \end{aligned}$$

Example 7.3

Consider the initial value problem

$$y' = y, \quad y(0) = 1.$$

Approximate y(0.05) with a step-size $h = 0.01$ using RK2 and RK4.

Solution: Here $f(x, y) = y, x_0 = 0, \quad y_0 = 1, \quad$ and $\quad h = 0.01$. First let's use RK2 for $n = 0, 1, \cdots, 5$.

A $x = 0.01$ or when $n = 0$:

$$k_1 = hf(x_0, y_0) = hy_0 = 0.010000$$
$$k_2 = hf(x_0 + h, y_0 + k_1) = h(y_0 + k_1) = 0.01(1 + 0.01) = 0.010100$$
$$y_1 = y_0 + \frac{1}{2}(k_1 + k_2) = 1.0 + 0.5(0.010000 + 0.010100) = 1.010050$$

At $x = 0.02$ or when $n = 1$:

$$k_1 = hf(x_1, y_1) = hy_1 = 0.01(1.010050) = 0.010100$$
$$k_2 = hf(x_1 + h, y_1 + k_1) = h(y_1 + k_1) = 0.01(1.010050 + 0.010100) = 0.010202$$
$$y_2 = y_1 + \frac{1}{2}(k_1 + k_2) = 1.010050 + 0.5(0.010100 + 0.010202) = 1.020201$$

Repeating this until $x = 0.05$ we get the following

x_i	k_1	k_2	y_i
0.0	–	–	1.000000
0.1	0.010000	0.010100	1.010050
0.2	0.010100	0.010202	1.020201
0.3	0.010202	0.010304	1.030454
0.4	0.010305	0.010408	1.040810
0.5	0.010408	0.010512	1.051270

Therefore, $y(0.05) = 1.051270$

Now, let us use RK4 for $n = 0, 1, \cdots 5$.

A $x = 0.01$ or when $n = 0$:

$$k_1 = hf(x_0, y_0) = hy_0 = 0.010000$$
$$k_2 = hf(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}k_1) = h(y_0 + \frac{1}{2}k_1) = 0.01(1 + 0.005) = 0.010050$$
$$k_3 = hf(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}k_2) = h(y_0 + \frac{1}{2}k_2) = 0.01(1 + 0.005025) = 0.010050$$
$$k_4 = hf(x_0 + h, y_0 + k_3) = h(y_0 + k_3) = 0.01(1 + 0.010050) = 0.010101$$
$$y_1 = y_0 + \frac{1}{2}(k_1 + 2k_2 + 2k_3 + k_4) = 1.0 + \frac{1}{6}(0.010000 + 2 \times 0.010050 + \times 0.010050 + 0.010101)$$
$$= 1.010050$$

At $x = 0.02$ or when $n = 1$:

$$k_1 = hf(x_1, y_1) = hy_1 = 0.01(1.010050) = 0.010101$$
$$k_2 = hf(x_1 + \frac{1}{2}h, y_1 + \frac{1}{2}k_1) = h(y_1 + \frac{1}{2}k_1) = 0.01(1.010050 + 0.005050) = 0.010151$$
$$k_3 = hf(x_1 + \frac{1}{2}h, y_1 + \frac{1}{2}k_2) = h(y_1 + \frac{1}{2}k_2) = 0.01(1.010050 + 0.5 \times 0.010151) = 0.010151$$
$$k_4 = hf(x_1 + \frac{1}{2}h, y_1 + k_3) = h(y_1 + k_3) = 0.01(1.010050 + 0.010151) = 0.010202$$
$$y_2 = y_1 + \frac{1}{2}(k_1 + 2k_2 + 2k_3 + k_4) = 1.010050 + \frac{1}{6}(0.010101 + 2 \times 0.010151 + 2 \times 0.010151 + 0.01020$$
$$= 1.020201.$$

Repeating the above procedure until $x = 0.05$ we get the following

x_i	k_1	k_2	k_3	k_4	y_i
0.0	–	–	–	–	1.000000
0.1	0.010000	0.010050	0.010050	0.010101	1.010050
0.2	0.010101	0.010151	0.010151	0.010202	1.020201
0.3	0.010202	0.010253	0.010253	0.010305	1.030455
0.4	0.010305	0.010356	0.010356	0.010408	1.040811
0.5	0.010408	0.010460	0.010460	0.010513	1.051271

Therefore, $y(0.05) = 1.051271$