Dr. Suresan Pareth

Meghadooth, Palayad, Kerala

May 29, 2020

Contents:

Welcome to C++ Programming!

1 Introduction

C++ is a multi-paradigm or versatile programming language that can be considered as a sort of sophisticated weapon for the coding world. This is because it supports structured programming, Object Oriented Programming, and even functional programming patterns. The versatility of C++ undoubtedly makes it the best-suited programming language for the scientific programming.

1.1 C++: History and Applications

C++ is an object oriented programming language, C++ was developed by Bjarne Stroustrup at AT & T Bell lab, USA in early eighties, when Bjarne Stroustrup was doing work for his Ph.D. thesis. C ++ was developed from C and Simula 67 language. One of the languages Stroustrup had the opportunity to work with was a language called Simula, which as the name implies is a language primarily designed for simulations. The Simula 67 language - which was the variant that Stroustrup worked with - is regarded as the first language to support the object-oriented programming paradigm. Stroustrup found that this paradigm was very useful for software development, however the Simula language was far too slow for practical use. Fascinated by object-oriented approach Stroustrup thought of implementing this paradigm in software development, however, the Simula language was far too slow for practical use. So he began working on C with classes i.e. he started working on a new language which would have object-oriented paradigm mixed with the features of C programming language. C++ was early called 'C with classes'. In 1983, it was named C++ and it included some add-on features such as classes, inheritance, in-lining, default function arguments, polymorphism, encapsulation and strong type checking. Since then it has grown to become one of the most polished languages of the computing world.

C++ is a high level programming language, such as C, Fortran, BASIC, PHP, etc. C++ is a general purpose language language. By general purpose what it means that it is designed to be used for developing applications in a wide variety of domains. Some specific features of C++ are as follows:

Specific features of C++

1. Object-oriented: C++ is an object-oriented programming language. This means that the focus is on "objects" and manipulations around these objects. Information about how these manipulations work is abstracted out from the consumer of the object.

2. Rich library support: Through C++ Standard Template Library (STL) many functions are available that help in quickly writing code. For instance, there are standard libraries for various containers like sets, maps, hash tables, etc.

3. Speed: C++ is the preferred choice when latency is a critical metric. The compilation, as well as the execution time of a C++ program, is much faster than most other general purpose programming languages.

4. Compiled: A C++ code has to be first compiled into low-level code and then executed, unlike interpreted programming languages where no compilation is needed.

5. Pointer Support: C++ also supports pointers which are widely used in programming and are often not available in several programming languages.

Applications of C++

Mainly C++ Language is used for Developing Desktop application and system software. Some application of C++ language are given below.

For Develop Graphical related application like computer and mobile games.

To evaluate any kind of mathematical equation use C++ language.

C++ Language are also used for design OS. Like window xp.

Few parts of apple OS X are written in C++ programming language.

Internet browser Firefox are written in C++ programming language.

All major applications of adobe systems are developed in C++ programming language. Like Photoshop, ImageReady, Illustrator and Adobe Premier.

Some of the Google applications are also written in C++, including Google file system and Google Chromium.

C++ are used for design database like MySQL.

1.2 Generic computational approches

Computer programming is the process of designing and building an executable computer program to accomplish a specific computational result. And that involves tasks such as; analysis, generating algorithms, correctness and accuracy of the algorithm and resource consumption, and the implementation of algorithms in a chosen programming language. For programming, programmers write instructions to accomplish their computation tasks

in various programming languages. The source code of a program is written in one or more languages that are intelligible to programmers, rather than machine code, which is directly executed by the central processing unit. The purpose of programming is to find a sequence of instructions that will automate the performance of a task on a computer, often for solving a given problem. Such instructions can be executed directly when they are in the computer manufacturer specific numerical form known as 'machine language', after a simple substitution process when expressed in a corresponding 'assembly language', or after translation from some 'higher-level' language. There are many computer languages such as:

i) Machine level Language

Machine level Language : Machine code or machine language is a set of instructions executed directly by a computer's central processing unit (CPU). Each instruction performs a very specific task, such as a load, a jump, or an ALU operation on a unit of data in a CPU register or memory. Every program directly executed by a CPU is made up of a series of such instructions.

ii) Assembly level Language

An assembly language (or assembler language) is a low-level programming language for a computer, or other programmable device, in which there is a very strong (generally one-to-one) correspondence between the language and the architecture's machine code instructions. Assembly language is converted into executable machine code by a utility program referred to as an assembler; the conversion process is referred to as assembly, or assembling the code.

iii) High level Language

High-level language is any programming language that enables development of a program in much simpler programming context and is generally independent of the computer's hardware architecture. High-level language has a higher level of abstraction from the computer, and focuses more on the programming logic rather than the underlying hardware components such as memory addressing and register utilization.

The first high-level programming languages were designed in the 1950s. Languages like Ada , Algol, BASIC, COBOL, C, C++, JAVA, FORTRAN, LISP, Pascal, and Prolog.are considered high-level because they are closer to human languages and farther from machine languages. In contrast, assembly languages are considered lowlevel because they are very close to machine languages.

The high-level programming languages are broadly categorized in to two categories:

iv) Procedure oriented programming(POP) language.

In the procedure oriented approach, the problem is viewed as sequence of things to be done such as reading , calculation and printing. Procedure oriented programming basically consist of writing a list of instruction or actions for the computer to follow and organizing these instruction into groups known as functions.

The disadvantage of the Procedure Oriented Programming Language is:

1. Global data access.

2. It does not model real word problem very well.
3. No data hiding.

Characteristics of Procedure Oriented Programming Language:

1. Emphasis is on doing things(algorithm).
2. Large programs are divided into smaller programs known as functions.
3. Most of the functions share global data.
4. Data move openly around the system from function to function.
5. Function transforms data from one form to another.
6. Employs top-down approach in program design.

v) Object oriented programming(OOP) language.

Object oriented programming as an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.

Features of the Object Oriented programming:

1. Emphasis is on doing rather than procedure.
2. programs are divided into what are known as objects.
3. Data structures are designed such that they characterize the objects.
4. Functions that operate on the data of an object are tied together in the data structure.
5. Data is hidden and can't be accessed by external functions.
6. Objects may communicate with each other through functions.
7. New data and functions can be easily added.
8. Follows bottom-up approach in program design.

Basic Concepts of Object Oriented Programming:

1. Objects.

2. Classes.

3. Data abstraction and encapsulation.

4. Inheritance.

5. Polymorphism.

6. Dynamic binding.

7. Message passing.

Benefits of Object Oriented Programming:

Object Oriented Programming offers several benefits to both the program designer and the user. Object-oriented contributes to the solution of many problems associated with the development and quality of software products.

The chief advantages of Object Oriented Programming are:

1.Through inheritance we can eliminate redundant code and extend the use of existing classes.

2.We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity.

3.This principle of data hiding helps the programmer to build secure programs that can't be invaded by code in other parts of the program.

4.It is possible to have multiple instances of an object to co-exist with out any interference.

5.It is easy to partition the work in a project based on objects.

6.Object-oriented systems can be easily upgraded from small to large systems.

7.Message passing techniques for communication between objects makes the interface description with external systems much simpler.

8.Software complexity can be easily managed.

Applications Object Oriented Programming are:

The most popular application of oops up to now, has been in the area of user interface design such as windows. There are hundreds of windowing systems developed using oop techniques. Real business systems are often much more complex and contain many more objects with complicated attributes and methods. Oop is useful in this type of applications because it can simplify a complex problem. The promising areas for application of oop includes.

1. Real-Time Systems.
2. Simulation and modeling
3. Object oriented databases.
4. Hypertext, hypermedia and expertext.
5. Al and expert Systems.
6. Neural Networks and Parallel Programming.
7. Dicision Support and Office Automation Systems.
8. CIM / CAM / CAD Systems.

1.3 Basic elements of C++

1.3.1 C++ Comments

C++ introduces a new comment symbol //(double slash). Comments start with a double slash symbol and terminate at the end of line. A comment may start any where in the line and what ever follows till the end of line is ignored. Note that there is no closing symbol. The double slash comment is basically a single line comment. Multi line comments can be written as follows:

```
// this is an example of
// c++ program.
// Enjoy it!!!!
```

The C comment symbols /* ....*/ are still valid and more suitable for multi line comments.

1.3.2 Input Operator

The statement *cin>>name;* is an input statement and causes the program to wait for the user to type a value. The value keyed in is placed in the variable *name*. The *cin* is a predefined object in C++ that corresponds to the standard input stream. Here this stream represents the key board.

The operator >> is known as 'get from' operator. It extracts value from the keyboard and assigns it to the variable on its right.

1.3.3 Output Operator

The statement *cout << "Hello, World"* displays the string with in quotes on the screen. The *cout* can be used to display individual characters, strings and even numbers. It is a predefined object that corresponds to the standard output stream. Stream just refers to a flow of data and the standard output stream normally flows to the screen display. The cout object, whose properties are defined in *iostream* class represents that stream. The insertion operator << also called the 'put to' operator directs the information on its right to the object on its left.

1.3.4 Return Statement

In C++ *main()* returns an integer type value to the operating system. Therefore every *main()* in C++ should end with a return(0) statement, otherwise a warning or an error might occur in linux environment.

Structure of a Program:

In fact the best way to start learning a programming languages is by writing a program. Hence, we will start learning programming through C++, a high level, a general purpose programming language with classic "Hello World" program. This is an example of displaying a some text on the screen. The quotation marks in the program mark the beginning and end of the text to be displayed and they don't appear in the result. Heere is our first program:

Program 1.1:. Program to display Hello World message on the screen.

```
//Program to display Hello World message on the screen.

#include <iostream>
using namespace std;

int main()
{
        cout << "Hello World!";
        return 0;
}
```

```
Hello World!
```

When the program is run, then it will display the text "Hello World!" on the screen.

The first panel shows the source code for our first program. The second one shows the result of the program once compiled and executed. The way to edit and compile a program depends on the compiler you are using. Depending on whether it has a Development Interface or not and on its version. Consult the compilers section and the manual or help included with your compiler if you have doubts on how to compile a C++ console program. The previous program is the typical program that programmer apprentices write for the first time, and its result is the printing on screen of the "Hello World!" text. It is one of the simplest programs that can be written in C++, but it already contains the fundamental components that every C++ program has. We are going to look line by line at the code we have just written:

//Program to display Hello World message on the screen: This is a comment line. All lines beginning with two slash signs (//) are considered comments and do not have any effect on the behavior of the program. The programmer can use them to include short explanations or observations within the source code itself. In this case, the line is a brief description of what our program is.

#include <iostream>: Lines beginning with a hash sign (#) are directives for the preprocessor. They are not regular code lines with expressions but indications for the compiler's preprocessor. In this case the directive #include<iostream> tells the preprocessor to include the iostream standard file. This specific file (iostream) includes the declarations of the basic standard input-output library in C++, and it is included because its functionality is going to be used later in the program.

using namespace std;: All the elements of the standard C++ library are declared within what is called a namespace, the namespace with the name std. So in order to access its functionality we declare with this expression that we will be using these entities. This line is very frequent in C++ programs that use the standard library, and in fact it will be included in most of the source codes included in these tutorials. int main () This line corresponds to the beginning of the definition of the main function. The main function is the point by where all C++ programs start their execution, independently of its location within the source code. It does not matter whether there are other functions with other names defined before or after it – the instructions contained within this function's definition will always be the first ones to be executed in any C++ program. For that same reason, it is essential that all C++ programs have a main function.

The word *main* is followed in the code by a pair of parentheses(). That is because it is a function declaration: In C++, what differentiates a function declaration from other types of expressions are these parentheses that follow its name. Optionally, these parentheses may enclose a list of parameters within them.

Right after these parentheses we can find the body of the main function enclosed in braces{}. What is contained within these braces is what the function does when it is executed.

cout << "Hello World!"; This line is a C++ statement. A statement is a simple or compound expression that can actually produce some effect. In fact, this statement performs the only action that generates a visible effect in our first program.

cout represents the standard output stream in C++, and the meaning of the entire statement is to insert a sequence of characters (in this case the Hello World) into the standard output stream (which usually is the screen). *cout* is declared in the *iostream* standard file within the *std* namespace, so that's why we needed to include that specific file and to declare that we were going to use this specific namespace earlier in our code.

Notice that the statement ends with a semicolon character(;). This character is used to mark the end of the statement and in fact it must be included at the end of all expression statements in all C++ programs(one of the most common syntax errors is indeed to forget to include some semicolon after a statement).

return 0;: The return statement causes the main function to finish. return may be followed by a return code (in our example is followed by the return code 0). A return code of 0 for the main function is generally interpreted as the program worked as expected without any errors during its execution. This is the most usual way to end a C++ console program. You may have noticed that not all the lines of this program perform actions when the code is executed. There were lines containing only comments (those beginning by //). There were lines with directives for the compiler's preprocessor (those beginning by #). Then there were lines that began the declaration of a function (in this case, the main function) and, finally lines with statements (like the insertion into cout), which were all included within the block delimited by the braces{} of the *main* function.

In C++, the separation between statements is specified with an ending semicolon (;) at the end of each one, so the separation in different code lines does not matter at all for this

purpose. We can write many statements per line or write a single statement that takes many code lines. The division of code in different lines serves only to make it more legible and schematic for the humans that may read it.

1.4 Structure of C++ Program.

In general a C++ program will have the following blocks of codes.

1. **Include files.**
2. **Class declarations.**
3. **Class functions definitions.**
4. **Main function of the program.**

We can write program to display name and age on the screen of a computer in the following different ways:

Program 1.2:. Program to display name and age on the screen using the concept of structured programming.

```
//Program to display name and age on the screen.

#include <iostream>
using namespace std;

int main()
{
        char name[20];
        int age;
        name="Bjarne";
        age=55;
        cout<<name<<endl;
        cout<<age;
        return 0;
}
```

```
Bjarne
    55
```

Program 1.3:. Program to display name and age on the screen using concept of procedure oriented programming.

```
//Program to display name and age on the screen.

#include <iostream>
```

(continues on next page)

(continued from previous page)

```
using namespace std;

char name[20];
int age;

void getdata()
{
        name="Bjarne";
        age=55;
}
void display()
{
        cout<<name<<endl;
        cout<<age;
}
int main()
{
        getdata();
        display();
        return 0;
}
```

```
Bjarne
    55
```

Program 1.4.: Program to display student name and age on the screen using the concept of class.

```
//Program to display name and age on the screen.

#include <iostream>
using namespace std;

class student
{
        char name[30];
        int age;
        public:
                void getdata(void);
                void display(void);
};

void  student :: getdata(void)
{
        cout<<''enter name'';
```

(continues on next page)

(continued from previous page)

```
        cin>>name;
        cout<<''enter age'';
        cin>>age;
}

void student :: display()
{
        cout<<''\n name:''<<name;
        cout<<''\n age:''<<age;
}

int main( )
{
        student p;
        p.getdata();
        p.display();
        return(0);
}
```

2 First Step to Programming

2.1 Constants

The identifiers used to identify the memory locaion of data items which remains unchanged during the execution of the program is called constants. We can define constants in C++, using *#define* preprocessor and *const* keyword. Constants can be numerics of *integer* or *floating point* types or non-numeric constants of single character, string or escape sequences. The following is the syntax for defining constants.

Syntax:

```
#define identifier value
const type variable = value;
```

2.2 Variables

A variable is a identifier that refers to a value in the memory that varies. Variable names can be of any length. They can contain both letters and numbers, but they can't begin with a number. It is legal to use uppercase letters, but it is conventional to use only lowercase for variables names. The underscore character, _ , can appear in a varaible name. To create a variable in C++, you have to do specify the variable name, and then assign a value to it using the following syntax.

```
variablename = value;
```

C++ uses = to assign values to variables. There's must need to declare a variable in advance (or to assign a data type to it), assigning a value to a variable itself declares and initializes the variable with that value.

You can not use C++'s keywords as a valid variable name. Rules for variable naming:

1. Variables names must start with a letter or an underscore.
2. The remainder of your variable name may consist of letters, numbers and underscores.
3. Names are case sensitive.

There's need to specify a data type when declaring a variable in C++, while allocating the necessary area in memory for the variable, the C++ Compiler accordingly fix the type for it. Variable assignment works from left to right.

2.3 Assignment

The symbol '=' is used for assignment, it takes the right-hand side (called rvalue) and copy it into the left-hand side (called lvalue).

2.4 Tockens

The smallest individual units in program are known as tokens. C++ has the following tokens.

i) Keywords

ii) Identifiers

iii) Constants

iv) Strings

v) Operators

2.5 Keywords

Note that there are certain tokens that can not be used for naming the varaible. Such tokens are known as C++'s Keywords. The compiler uses Keywords to recognize the structure of the program. The Keywords of C++ are:

Table 1: Keywords

asm	double	new	swtich
auto	else	operator	template
break	enum	private	this
case	extern	protected	throw
catch	float	public	try
char	for	register	typedef
class	friend	return	union
const	goto	short	unsigned
continue	if	signed	virtual
default	inline	sizeof	void
delete	long	static	volatile
do	int	struct	while

Following are some Keywords added by ANSI C++.

Table 2: Keywords

bool	export	reinterpret_cast	typename
const_cast	false	static_cast	using
dynamic_cast	mutable	true	wchar_t
explicit	namespace	typeid	

2.6 Data Types

The basic information that we deal with are nothing but data. And these data can be numeric or alphabets of different sizes. To process these data they have to be stored in appropriate format or types such as character, wide character, integer, floating point, double floating point, Boolean etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the allocated memory. Computer Memory is organized in bytes, and for these variables, a data type is associated. Every identifier is declared with two entities, its type, and its name. The exact sizes and ranges of values for the fundamental types are implementation dependent. There are several data types available in C++. The various data types provided by C++ are;

1. Built-in data types

The basic built-in data types are char, int, float, double, bool, etc as given below.

To deal with integer type data item, C++ is having data type *int* with different variation based on the sign and size as detaild below.

Integer

As you learned in mathematics, the integers include 0, the positive whole numbers, and the negative whole numbers. Integer literals in a C++ program are written without commas, and a leading negative sign indicates a negative value.

int: to store integer values regardless of sign,

signed int: to store integer values regardless of sign,

unsigned int: to store integer values positive or zero and no negative,:

short: to store integer values regardless of sign,

signed short: to store integer values regardless of sign,

unsigned short: to store integer values positive or zero and no negative,

long: to store integer values regardless of sign,

signed long: to store integer values regardless of sign,

unsigned long: to store integer values positive or zero and no negative.

Floating point

A real number in mathematics, such as the value of 3.1416..., consists of a whole number, a decimal point, and a fractional part. Real numbers have infinite precision , which means that the digits in the fractional part can continue forever. Like the integers, real numbers also have an infinite range. However, because a computer's memory is not infinitely large, a computer's memory limits not only the range but also the precision that can be represented for real numbers. C++ uses floating-point numbers to represent real numbers. A floating-point number can be written using either ordinary decimal notation or scientific notation . Scientific notation is often useful for mentioning very large numbers.

float: holds a real number,

double: holds a real number of double precision,

long double: holds a real number of double precision.

Character type

char: any single character including a letter, a digit, a punctuation mark, or a space,

signed char: is same as an ordinary char and has a range from -128 to +127

unsigned char: for the ranges of values from 0 to 255, and no negative value.

Boolean

bool: holds boolean values; true or False.

2. Derived data types

Data type that are derived from the built-in data types are known as derived data types. **Array, Function, References** and **Pointers** are various derived data types provided by C++.

3. User-defined data types

User-defined data types related variables allow us to store Multiple values either of same type or different type or both. This is a data type whose variable can hold more than one value of dissimilar type. C++ provides various user-defined data types, like Structures, Unions, Enumerations and Classes.

Structure & Union: Structure and Union are the significant features of C language, Structure and Union provide a way to group similar or dissimilar data types referred to by a single name. However, C++ has extended the concept of Structure and Union by incorporating some new features in these data types to support object-oriented programming. Structure and Union are declared using the keywords *struct* and *union* respectively.

Class: C++ offers a new user-defined data type known as class, which forms the basis of object-oriented programming. A class acts as a template which defines the data functions that are included in an object of a class. Classes are declared using the keyword *class*. Once a class has been declared, its object can be easily created.

Enumeration: This is an user defined data type having finite set of enumeration constants. The keyword *enum* is used to create enumerated data type.

Besides the above data types, following are another two data types available.

string: holds sequence of characters, and defined in header file.

void: void data type is used for specifying an empty parameter list to the function to indicate that the function does not take any parameter and also with return type for a function to indicate that the function does not return any value. Note that *void* can not be used to declare ordinary variables but can be for declaring generic pointers.

2.7 Operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C++ is rich in built-in operators and provides the following types of operators:

2.7.1 Arithmetic Operators

Table 3: Arithmatic Operators

Operator	Meaning	Syntax
−	Negation	-a
+	Addition	a+b
−	Subtraction	a-b
*	Multiplication	a*b
/	Division	a/b
%	Remainder	a%b
++	Increment	a++
−−	Decrement	a–

2.7.2 Relational Operators

Table 4: Relational Operators

Operator	Meaning	Syntax
==	Is equal to	a==b
! =	Is not equal to	a!=b
>	Greater than	a>b
<	Less than	a<b
>=	Greater than or equal to	a>=b
<=	Less than or equal to	a<=b

2.7.3 Logical Operators

Table 5: Logical Operators

Operator	Meaning	Syntax
&&	And operator. Performs a logical conjunction of two expressions.	a&&b
||	Or operator. Performs a logical disjunction on two expressions.	a||b
!	Not operator. Performs logical negation on an expression.	!b

2.7.4 Bitwise Operators

Bitwise operator works on bits and perform bit-by-bit operation.

Table 6: Operator Precedence

Operator "Description"	
&	Binary AND Operator copies a bit to the result if it exists in both operands
|	Binary OR Operator copies a bit if it exists in either operand
^	Binary XOR Operator copies the bit if it is set in one operand but not both
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits
<<	Binary Left Shift Operator
>>	Binary Right Shift Operator

2.7.5 Assignment Operators

Assignment Operator(=)

Using the operator or symbol '=', we can assign values to identifiers. For example,

```
name = "Tom"
sem = "First Smemester"
age = 21
rollno = "rcet202001"
```

Miscellaneous operators

There are few other operators supported by C++ Language.

1. sizeof ; sizeof operator returns the size of a variable.

2. ? : ; similar to if ... else ...

3. dot(.) and -> ; operators are used to reference individual members of classes, structures, and unions.

4. address operator &, returns the address of a variable.

5. pointer operator(*); is a pointer to a variable.

2.7.6 Cascading of I/O Operator

C++, allows programmers to associate more than one statements in a single line with cout and cin as given below.

```
cout<<"sum = "<<sum<<"\n";
cout<<"sum = "<<sum<<"\n"<<"average = "<<average<<"\n";
cin>>num1>>num2;
```

Operator Precedence in C++

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator:

Table 7: Operator Precedence

Category	Operator	Associativity
Postfix	[];− >;.;++;−−	Left to right
Unary	+;−;!; ;++;−−	Right to left
Multiplicative	∗; /; %	Left to right
Additive	+;−	Left to right
Shift	<<;>>	Left to right
Relational	<;<=;>;>=	Left to right
Equality	==;! =	Left to right
Bitwise	AND &	Left to right
Bitwise	XOR ^	Left to right
Bitwise	OR |	Left to right
Logical	AND &&	Left to right
Logical	OR ||	Left to right
Conditional	? :	Right to left
Assignment	=;+ =;− =;∗ =; / =	Right to left
Comma	,	Left to right

2.8 Expressions and Statements

An expression is a combination of values, variables, and operators. Expression is also known as statements. A value all by itself is considered an expression, and so is a variable, so the following are all legal expressions: When you type a statement, the compiler executes it, which means that it does whatever the statement says.

Arithmetic Expressions

An arithmetic expression consists of operands and operators combined in the usual way. Following example illustrates the use of several arithmetic operators.

2.9 Illustration

Program 1.1: Program to illustrate different arithematic operations.

```
//Program to illustrate different arithematic operations.

#include <iostream>
using namespace std;

int main()
{
        int a=50,b=25, add,sub,mul,div,mod;
        add = a+b;
        sub = a-b;
        mul = a*b;
        div = a/b;
        mod = a%b;
        cout<<"Addition of a, b is : \n"<<add;
        cout<<"Subtraction of a, b is :\n"<<sub;
        cout<<"Multiplication of a, b is :\n"<< mul;
        cout<<"Division of a, b is : \n"<<div;
        cout<<"Modulus of a, b is : \n"<<mod;
}
```

```
Output of the above program is given below:
Addition of a, b is :75
Subtraction of a, b is :25
Multiplication of a, b is :1250
Division of a, b is :2
Modulus of a, b is :0
```

Program 1.2: Program to illustrate bitwise operations.

```
//Program to illustrate different bitwise operations.

#include<iostream>
using namespace std;

main()
{
        unsigned int a = 50;
        unsigned int b = 23;
        int rob = 0;
        rob = a & b;
        cout << "Result after & ing : " << rob << endl ;
        rob = a | b;
```

(continues on next page)

(continued from previous page)

```
        cout << "Result after | ing : " << rob << endl ;
        rob = a ^ b;
        cout << "Result after ^ ing : " << rob << endl ;
        rot = ~a;
        cout << "Result after ~ ing : " << rob << endl ;
        rob = a << 2;
        cout << "Result after << ing by 2: " << rob << endl ;
        rob = a >> 2;
        cout << "Result after >> ing by 2: " << rob << endl ;
}
```

```
Result after & ing : 18
Result after | ing: 55
Result after ^ ing: 37
Result after ~ ing: -51
Result after << ing by 2: 200
Result after >> ing by 2: 12
```

Program 1.3: Program to illustrate relational operations.

```
//Program to illustrate different relational operations.

#include <iostream>
using namespace std;

main()
{
        int a,b;
        cout<<"Enter two numbers\n";
        cin>>a>>b;
        if(a == b)
        {
                cout<<"a is equal to b" << endl ;
        }
        else
        {
                cout<<"a is not equal to b" << endl;
        }
        if(a < b)
        {
                cout<<"a is less than b"<< endl;
        }
        else
        {
                cout<<"a is not less than b"<<endl;
```

(continues on next page)

(continued from previous page)

```
        }
}
```

Program 1.4: Program to illustrate logical operations.

```
//Program to illustrate different logical operations.

#include <iostream>
using namespace std;

#include <iostream>
using namespace std;

main()
{
        int a,b;
        cout<<"Enter two numbers\n";
        cin>>a>>b;
        if(a && b)
        {
                cout<<"Condition is true"<<endl;
        }
        else
        {
                cout<<"Condition is false"<<endl;
        }
        if(a || b)
        {
                cout <<"Condition is true"<<endl;
        }
        else
        {
                cout<<"Condition is false"<<endl;
        }
}
```

Program 2.2:. Program to compute the area of a rectangle by accepting side length as inputs.

```
#include <iostream>
using namespace std;

int main()
{
        float l,b,area;
```

(continues on next page)

(continued from previous page)

```
        cout<<"Enter the length in meters\n";
        cin>>l;
        cout<<"Enter the breadth in meters\n";
        cin>>b;
        area=l*b;
        cout<<"Area="<<area<<" m^2.";
}
```

Upon executing the above program, the following output will be displayed on the screen as and when the programmer inputs the side lengths from the keyboard.

```
enter the length in meters
4
enter the breadth in meters
5
area=20 m^2.
```

Program 2.3:. Program to compute the area of a circle by accepting the radius as input.

```
#include <iostream>
using namespace std;

int main()
{
        float r,pi=3.14159,area,circum;
        cout<<"Enter radius of the circle to find area and its circumference\n";
        cin>>r;
        area=pi*r*r;
        circum=2*pi*r;
        cout<<"Area of the circle = "<<area<<endl;
        cout<<"Circumference of the circle= "<<circum;
}
```

```
Enter radius of the circle to find are and its circumference
2
Area of the circle = 12.566371
Circumference of the circle = 12.566371
```

Program 2.4:. Program to compute the area of a triangle using Heron's formula by accepting side lengths as inputs.

```
#include <iostream>
using namespace std;
#include <math.h>
```

(continues on next page)

(continued from previous page)

```
int main()
{
        float a,b,c,art,s;
        cout<<"Enter the side length a\n";
        cin>>a;
        cout<<"Enter the side length b\n";
        cin>>b;
        cout<<"Enter the side length c\n";
        cin>>c;
        s=(a+b+c)/2;
        art=sqrt(s*(s-a)*(s-b)*(s-c));
        circum=2*pi*r;
        cout<<"Area ="<<art<<" square units";
}
```

```
Enter the side length a
3
Enter the side length b
4
Enter the side length c
5
Area = 6  square units.
```

Program 2.5:. Program to compute the Celsius by accepting Fahrenheit as inputs.

```
#include <iostream>
using namespace std;

int main()
{
        float F,C;
        cout<<"Enter the fahrenheit\n";
        cin>>F;
        C=5*(F-32)/9;
        cout<<"Temperature in Celsius"<<C;
}
```

```
Enter the length
50
Temperature in Celcious =10.00
```

Program 2.6:. Program to compute the Fahrenheit by accepting Celcious as inputs.

```
#include <iostream>
using namespace std;

int main()
{
        float F,C;
        cout<<"Enter the Celcious\n";
        cin>>C;
        F=9*C/5+32;
        cout<<"Temperature in fahrenheit"<<C;
}
```

```
Enter the Celcious
10
Temperature in fahrenheit =50.00
```

2.10 The Software Development Process

There is much more to programming than writing lines of code. The "more" consists of organization and planning, and various conventions for diagramming those plans. Computer scientists refer to the process of planning and organizing a program as software development. There are several approaches to software development. One version is known as the waterfall model. The waterfall model consists of several phases:

1.Customer request —In this phase, the programmers receive a broad statement of a problem that is potentially amenable to a computerized solution. This step is also called the user requirements phase.

2.Analysis —The programmers determine what the program will do. This is sometimes viewed as a process of clarifying the specifications for the problem.

3.Design —The programmers determine how the program will do its task.

4.Implementation —The programmers write the program. This step is also called the coding phase.

5.Integration —Large programs have many parts. In the integration phase, these parts are brought together into a smoothly functioning whole, usually not an easy task.

6.Maintenance —Programs usually have a long life; a life span of 5 to 15 years is common for software. During this time, requirements change, errors are detected, and minor or major modifications are made. Taken together, these phases are also called the software development life cycle. Apart from warterfall model there are other well known models such as incremental model and iterative model. Modern software development is usually incremental and iterative . This means that analysis and design may produce a rough draft,

skeletal version, or prototype of a system for coding, and then back up to earlier phases to fill in more details after some testing.

3 Input/Output Functions

3.1 Unformatted I/O Functions

The stream concerns for how to perform input/output operations. Basically, a stream is a sequence of characters with functions to accept characters from keyboard and put characters into dispaly.

Unformatted Input/output is the most basic form of input/output. Unformatted I/O transfers the internal binary representation of the data directly between memory and the file.

Unformatted I/O operations are *cout*, *cin*. These are predefined objects which are by default connected to I/O devices. The extraction (>>) and insertion (<<) operators define the direction of data flow.

Syntax:

```
cin>>var1>>var2>>...>>varN;
cout<<var1<<var2<<...<<varN;
```

Program 1.1: Program to illustrate cin and cout.

```
#include<iostream>
using namespace std;

int main()
{
        int a,b;
        char ch;
        float av;
        cout<<"Enter the values for a and b\n";
        cin>>a>>b;
        cout<<"Enter a character\n";
```

(continues on next page)

(continued from previous page)

```
        cin>>ch;
        cout<<"Enter a float value\n";
        cin>>av;
        cout<<"a = "<<a<<", b = "<<b<<endl;
        cout<<"character is = "<<ch<<endl;
        cout<<"Float value is = "<<av<<endl;
}
```

3.2 Unformatted Stream I/O Functions

The *ios* class provides the basic support for two kinds of input and output such as formatted and unformatted. The class *istream* provides the facilities for formatted and unformatted input operations whereas the *ostream* class provides the facilities for formatted and unformatted output. The *istream*, class declares input with overloaded >> and *ostream* output with overloaded << functions an both inherits properties of *ios* class. The *iostream* class inherits the properties of *ios*, *istream* and *ostream* classes through multiple inheritance and hence contains all input and output functions.

Unformatted input and output operations can also be carried out using various member functions of cin and cout objects provided by the istream and ostream classes. The functions *get()* and *put()* can be used to handle single character input and output operations, respectively. The general usage of get is as follows;

1. get(char *)

The *get(char *)* function assigns the character read from the keyboard to its argument. Unlike extraction operator >>, it can read blanks spaces and newline characters. Note that >> operator simply skips the blank spaces and newline character while extracting characters from the input stream.

2. get(void)

get(void) returns the character read from the keyboard without assigning it to any variable.

Syntax:

```
char ch;
cin.get(ch);
```

3. put()

put() member function can be used to write one character to the monitor. It may take a character constant or variable as argument. The *put()* function can also take an integer value as input, however rather than displaying the integer value it displays its ASCII equivalent character.

Syntax:

```
char ch;
cout.put(ch);
```

Program 1.2: Program to display a character using get() and put() functions.

```
#include<iostream>
using namespace std;

int main()
{
        char ch;
        cout<<"Enter a character\n";
        cin.get(ch);
        cout<<"character is = "<<endl;
        cout.put(ch);
}
```

Note that, the functions *get()* and *put()* can handle a single character at a time. The *getline()* member function can be used for reading a line of text.

Syntax:

```
char var[];
cin.getline(var, size);
```

Program 1.3: Program to display name using getline() function.

```
#include<iostream>
using namespace std;

int main()
{
        char name[20];
        cout<<"Enter a string\n";
        cin.getline(name, 20);
        cout<<"Text is  ";
        cout<<name;
}
```

The other member functions used to perform unformatted I/O include *read()* and *write()*. The member function *read()* inputs bytes to a character array in memory; member function *write()* writes output bytes from character array;

Syntax of read() and write()

```
char var[];
cin.read(var, size);
cout.write(var, size);
```

Program 1.4: Program to display name using read() and write() functions.

```
#include<iostream>
using namespace std;

int main()
{
        char name[6];
        cout<<"Enter a string\n";
        cin.read(name,6);
        cout<<"Text is  ";
        cout.write(name,6);
}
```

3.3 Formatted Output Functions

To display the outputs in a specified manner, in C++, there are features defined in *ios* class of C++ such as the following.

1. width()

width(n) function can be used with each output item to specify the width of the field required for displaying in n character space from right. Note that, even if the specified number n is not enough to hold items, *width()* function provides sufficient character space.

Syntax

```
cout.width(n);
```

2. precision()

We can change default number of digits to desired number, for digits after the decimal point, while displaying floating point numbers, using the *precision()* function. Note that, the effect of *precision()* function persists till it is reset.

Syntax

```
cout.precision(n);
```

3. fill()

The blank spaces printed by default in conjection with *width()* function, in the output item can be replaced by character(ch) of user interest using the *fill()* function.

Syntax

```
cout.fill(ch);
```

4. setf()

For, displaying of output item in left-justified, right-justified, etc can be accomplished by using *setf()* function by specifying proper flags as argument. And *unsetf()* can be for resetting of defined flags.

Syntax

```
cout.setf(arg1, arg2);
```

or

```
cout.setf(arg);
```

Program 1.1: Program to illustrate the use of output format functions.

```
#include<iostream>
using namespace std;

int main()
{
        char name[15]="Aaron";
        float pi=3.1415926535;
        cout<<"Name without setting width: "<<name<<endl;
        cout<<"Name after setting width of 10: ";
        cout.width(10);
        cout<<name<<endl;
        cout<<"Name, no effect of previous width setting: "<<name<<endl;
        cout.fill('*');
        cout<<"Name, blank space filled with * using fill('*'): ";
        cout.width(10);
        cout<<name<<"\n";
        cout<<"Value of pi with default digits after . = "<<pi<<endl;
        cout.precision(3);
        cout<<"Value of pi after setting to 2 digits after decimal point
                = "<<pi<<endl;
        cout.precision(8);
        cout<<"Value of pi after setting to 7 idigits after decimal point
                = "<<pi<<endl;
        cout.fill('$');
        cout.setf(ios::left,ios::adjustfield);
        cout<<"Name after left justifying and padding with $: ";
        cout.width(10);
        cout<<name<<"\n";
```

(continues on next page)

(continued from previous page)

```
        cout.setf(ios::right,ios::adjustfield);
        cout<<"Name after right justifying and padding with $: ";
        cout.width(10);
        cout<<name<<"\n";
        cout.precision(3);
        cout.setf(ios::scientific,ios::floatfield);
        cout<<"Value of pi in scientific form with 3 decimal digits: "<<pi<<"\n";
}
```

```
Name without setting width: Aaron
Name after setting width of 10:      Aaron
Name, no effect of previous width setting: Aaron
Name, blank space filled with * using fill('*'): *****Aaron
Value of pi with default digits after . = 3.14159
Value of pi after setting to 2 digits after decimal point = 3.14
Value of pi after setting to 7 idigits after decimal point = 3.1415927
Name after left justifying and padding with $: Aaron$$$$$
Name after right justifying and padding with $: $$$$$Aaron
Value of pi in scientific form with 3 decimal digits: 3.142e+00
```

3.4 Stream Manipulators

The *<iomanip>* header file provides a number of stream manipulating features for formatting the outputs. The following are some of them.

1. **setw(int n):** similar to width(n) of *ios* class.
2. **setprecision(int n):** similar to precision(n) of *ios* class.
3. **setfill(char ch):** similar to fill(ch) of *ios* class.
4. **setiosflags(mask):** similar to setf(n) of *ios* class.
5. **resetiosflags(mask):** similar to unsetf(n) of *ios* class.

Program 1.2: Program to illustrate the use of output format manipulators.

```
#include<iostream>
using namespace std;
#include<iomanip>

int main()
{
        char name[15]="Aaron";
        float pi=3.1415926535;
```

(continues on next page)

(continued from previous page)

```
        cout<<"Name without setting width: "<<name<<endl;
        cout<<"Name after setting width of 10: ";
        cout<<setw(10);
        cout<<name<<endl;
        cout<<"Name, no effect of previous width setting: "<<name<<endl;
        cout<<setfill('*');
        cout<<"Name, blank space filled with * using fill('*'): ";
        cout<<setw(10);
        cout<<name<<"\n";
        cout<<"Value of pi with default digits after . = "<<pi<<endl;
        cout<<setprecision(3);
        cout<<"Value of pi after setting to 2 digits after decimal point
                = "<<pi<<endl;
        cout<<setprecision(8);
        cout<<"Value of pi after setting to 7 idigits after decimal point
                = "<<pi<<endl;
        cout<<setfill('$');
        cout<<setiosflags(ios::left|ios::adjustfield);
        cout<<"Name after left justifying and padding with $: ";
        cout<<setw(10);
        cout<<name<<"\n";
        cout<<setiosflags(ios::right|ios::adjustfield);
        cout<<"Name after right justifying and padding with $: ";
        cout<<setw(10);
        cout<<name<<"\n";
        cout<<setprecision(3);
        cout<<setiosflags(ios::scientific);
        cout<<"Value of pi in scientific form with 3 decimal digits: "<<pi<<"\n";
}
```

```
Name without setting width: Aaron
Name after setting width of 10:      Aaron
Name, no effect of previous width setting: Aaron
Name, blank space filled with * using fill('*'): *****Aaron
Value of pi with default digits after . = 3.14159
Value of pi after setting to 2 digits after decimal point = 3.14
Value of pi after setting to 7 idigits after decimal point = 3.1415927
Name after left justifying and padding with $: Aaron$$$$$
Name after right justifying and padding with $: $$$$$Aaron
Value of pi in scientific form with 3 decimal digits: 3.142e+00
```

4 Control Structures

Control statements(Loop): Statements that allow the computer to select or repeat an action. We begin our study of control statements with repetition statements, also known as loops which repeat an action.

There are two types of loops—those that repeat an action a predefined number of times (definite iteration) and those that perform the action until the program determines that it needs to stop (indefinite iteration). If the program has definite number of iteration, we use the for Loop:Each repetition of the action is known as a pass or an iteration.

4.1 Definite iteration: for loop

In this section, we examine C++'s for loop , the control statement that most easily supports definite iteration. In for loop, initialization happens first and only once, which means that the initialization part of for loop only executes once. Condition in for loop is evaluated on each loop iteration, if the condition is true then the statements inside for loop body gets executed. Once the condition returns false, the statements in for loop does not execute and the control gets transferred to the next statement in the program after for loop.

Note that after every execution of for loop's body, the increment/decrement part of for loop executes and updates the loop counter. After this, the control jumps to re-evaluate test condition and repeats the above process until the loop condition returns false.

The syntax of for Loop is:

```
for (Initialization; TestExpression; Update)
{
        <statement-1>
        .
        .
        .
```

(continues on next page)

(continued from previous page)

```
        <statement-n>
}
```

How for loop works:

Initialization: Is often used to initialize variables; for example: int i=1;

Test Expression: Determines loop termination. i.e., if the condition evaluates to true then the body of loop will be executed and go to Update expression otherwise will exit from the *for* loop; for example: i <= 10;

Update: After executing loop body Update increments/decrements the loop variable by some value; for example: i++;

For **Example 3.1:**

```
#include<iostream>
using namespace std;

int i;
int sum = 0;
for (i=0; i<10; i=i+1)
{
        sum = sum + i;
}
```

Example 3.2:

```
#include<iostream>
using namespace std;

for(i = 0; i <= 3; i++)
cout<<"i="<<i;
```

It outputs as given below.

```
i= 0
i= 1
i= 2
i= 3
```

Program 3.1:. Program to compute the factorial of a number by accepting an integer as inputs using for loop.

```
//Prgoram to find the factorial.

#include<iostream>
```

(continues on next page)

(continued from previous page)

```
using namespace std;

int main()
{
        int i,n,f;
        cout<<"Enter the number to find the facorial\n";
        cin>>n;
        f=1;
        for(i = 1; i <= n; i++)
        f=f*i;
        cout<<"factorial = "<<f;
        return 0;
}
```

```
Enter the number to find the facorial
5
factorial = 120
```

Program 3.2:. Program to compute the nth power of a number using for loop.

```
//Prgoram to find the power of a given number.

#include<iostream>
using namespace std;

int main()
{
        int i,n,p,x;
        cout<<"Enter the base and exponent\n";
        cin>>x;
        cin>>n;
        p=1;
        for(i = 1; i <= n; i++)
        p=p*x;
        cout<<"Power = "<<p;
        return 0;
}
```

```
Enter the base and exponent
3
2
Power = 9
```

Program 3.3:. Program to compute the series sum using for loop.

```
//Prgoram to find find the series sum.

#include<iostream>
using namespace std;

int main()
{
        int i,n,t,x,sum;
        cout<<"Enter the base and exponent\n";
        cin>>x;
        cin>>n;
        t=1;
        sum=1;
        for(i = 1; i <= n; i++)
        {
                t=t*x;
                sum=sum+t;
        }
        cout<<"Sum of the series = "<<sum;
        return 0;
}
```

```
Enter the base and exponent
2
5
Sum of the series = 62
```

4.2 Conditional Iteration:

Earlier we examined the for loop, which executes a set of statements a definite number of times specified by the programmer. In many situations, however, the number of iterations in a loop is unpredictable. The loop eventually completes its work, but only when a condition changes. For example, the user might be asked for a set of input values. In that case, only the user knows the number that to enter. The program's input loop accepts these values until the user enters a special value or sentinel that terminates the input. This type of process is called conditional iteration, meaning that the process continues to repeat as long as a condition remains true. The Conditional iteration requires that a condition be tested within the loop to determine whether the loop should continue. Such a condition is called the loop's continuation condition . If the continuation condition is false, the loop ends. If the continuation condition is true, the statements within the loop are executed again.

4.2.1 while Loop

The while loop is entry controlled loop, i.e., the test condition is evaluated first and if it returns true then the statements inside while loop execute, this happens repeatedly until the condition returns false. When condition returns false, the control comes out of loop and jumps to the next statement in the program after while loop.

Here is the syntax of while loop:

```
while (test condition)
{
        <statement-1>
        .
        .
        .
        <statement-n>
}
```

How while loop works:

The while loop evaluates the test expression.

If the test expression is true, codes inside the body of while loop is evaluated.

Then, the test expression is evaluated again. This process goes on until the test expression is false.

When the test expression is false, while loop is terminated.

The Example 3.2 writen using for loop can be rewritten using while loop as indicated below.

```
#include<iostream>
using namespace std;

i=0;
while(i<=3)
{
        cout<<"i="<<i;
        i=i+1;
}
```

```
i= 0
i= 1
i= 2
i= 3
```

Program 3.4:. Program to compute the factorial of number using while loop.

```
//Prgoram to find the factorial using while loop.

#include<iostream>
using namespace std;

int main()
{
        int i,n,f;
        cout<<"Enter the number to find the facorial\n";
        cin>>n;
        f = 1;
        i = 1;
        while(i <= n)
        {
                f=f*i;
                i++;
        }
        cout<<"factorial = "<<f;
        return 0;
}
```

```
Enter the number to find the facorial
4
factorial = 24
```

Program 3.5:. Program to compute the nth power of a number using while loop.

```
//Prgoram to find the Power using while loop.

#include<iostream>
using namespace std;

int main()
{
        int i,n,f;
        cout<<"Enter the base and exponent\n";
        cin>>x;
        cin>>n;
        p=1;
        i = 1;
        while(i <= n)
        {
                p=p*x;
                i++;
        }
        cout<<"Power = "<<p;
```

(continues on next page)

(continued from previous page)

```
        return 0;
}
```

```
Enter the base and exponent
5
2
Power = 25
```

Program 3.6:. Program to compute the series sum using while loop.

```
//Prgoram to find find the series sum.

#include<iostream>
using namespace std;

int main()
{
        int i,n,t,x,sum;
        cout<<"Enter the base and exponent\n";
        cin>>x;
        cin>>n;
        t=1;
        sum=1;
        i = 1;
        while(i <= n)
        {
                t=t*x;
                sum=sum+t;
                i++;
        }
        cout<<"Sum of the series = "<<sum;
        return 0;
}
```

```
Enter the base and exponent
2
5
Sum of the series = 62
```

4.2.2 do. . . while Loop

The exit controlled, do. . . while loop, is a variant of the while loop with one important difference. The body of do. . . while loop is executed once before the test expression is checked. Note that, in while loop, condition is evaluated first and then the statements inside loop body

gets executed, on the other hand in do-while loop, statements inside do-while gets executed first and then the condition is evaluated. If the condition returns true then the control jumps to the "do" for further repeated execution of it, this happens repeatedly until the condition returns false. Once condition returns false control jumps to the next statement in the program after do-while.

The syntax of do. . . while loop is

```
do
{
        <statement-1>
        .
        .
        .
        <statement-n>
}while (<test condition>);
```

How do. . . while loop works:

The codes inside the body of loop is executed at least once.

Then, only the test expression is checked.

If the test expression is true, the body of loop is executed. This process continues until the test expression becomes false.

When the test expression is false, do. . . while loop is terminated.

The Example 3.2 writen using while loop can be rewritten using do . . . while loop as indicated below.

```
i=0;
do
{
        cout<<"i="<<i;
        i=i+1;
}while(i<=3);
```

```
i= 0
i= 1
i= 2
i= 3
```

Program 3.7:. Program to compute the factorial of a number using do . . . while loop.

```
//Prgoram to find the factorial using while loop.

#include<iostream>
```

(continues on next page)

(continued from previous page)

```
using namespace std;

int main()
{
        int i,n,f;
        cout<<"Enter the number to find the facorial\n";
        cin>>n;
        f=1;
        i = 1;
        do
        {
                f=f*i;
                i++;
        }while(i <= n);
        cout<<"factorial = "<<f;
        return 0;
}
```

```
Enter the number to find the facorial
4
factorial = 24
```

Program 3.8:. Program to compute the nth power of a number using do ... while loop.

```
//Prgoram to find the Power using while loop.

#include<iostream>
using namespace std;

int main()
{
        int i,n,f;
        cout<<"Enter the base and exponent\n";
        cin>>x;
        cin>>n;
        p=1;
        i = 1;
        do
        {
                p=p*x;
                i++;
        }while(i <= n);
        cout<<"Power = "<<p;
        return 0;
}
```

```
Enter the base and exponent
5
2
Power = 25
```

Program 3.9:. Program to compute the series sum using do ... while loop.

```
//Prgoram to find find the series sum.

#include<iostream>
using namespace std;

int main()
{
        int i,n,t,x,sum;
        cout<<"Enter the base and exponent\n";
        cin>>x;
        cin>>n;
        t=1;
        sum=1;
        i = 1;
        do
        {
                t=t*x;
                sum=sum+t;
                i++;
        }while(i <= n);
        cout<<"Sum of the series = "<<sum;
        return 0;
}
```

```
Enter the base and exponent
2
5
Sum of the series = 62
```

4.3 Selection Statements

In real time application, we need to execute a block of statements only when a particular condition is met or not met. This is called decision making, as we are executing a certain code after making a decision in the program logicFor decision making in C++, we have four types of control statements (or control structures), which are as follows:

i) if statement. (or simple if staement)

ii) if... else statement.

iii) if ... else ... if statement.

*iv) nested if statement.**

4.3.1 if statement

The if statement is used to execute a specific statement or set of statements when the given condition is true. There are various forms of if structures, as shown below.The if statement is used to make decisions based on specific conditions occurring during the execution of the program. An action or set of actions is executed if the outcome is true or false otherwise. The general form of a typical decision-making structure found in most programming languages including C++. Any nonzero and nonnull values are considered true in C++, while either zero or null values are considered false.

Syntax of if statement varients are given below:

```
1. if statement:

if(condition)
{
        <statements>
}

2. if ... else Statement:

if(condition)
{
        <statements>
}
else
{
        <statements>
}

3. if ... else ... if Statement:

if (condition)
{
        <statements>
}
else if (condition)
{
        <statements>
}
```

(continues on next page)

(continued from previous page)

```
else
{
        <statements>
}

4. Nested if Statement:

if (condition)
{
        <statements>
        if (condition)
        {
                <statements>
        }
}
```

The illustrative programs for implementing selection statement if; is shown below.

Program 3.10:. Program to check the given number is even using if statement.

```
#include<iostream>
using namespace std;

int main()
{
        int a;
        cout<<"Enter a number\n";
        cin>>a;
        if (a%2==0)
        {
                cout<<"Entered number is even."<<endl;
        }
}
```

```
Enter a number
    6
Entered numberr is even.
```

Program 3.11:. Program to find the greatest of two numbers using if ... else statement.

```
#include<iostream>
using namespace std;

int main()
{
```

(continues on next page)

(continued from previous page)

```
        int a,b,max;
        cout<<"enter a\n";
        cin>>a;
        cout<<"enter b\n";
        cin>>b;
        if (a>b)
        {
                max=a;
        }
        else
        {
                max=b;
        }
        cout<<"Greatest number is = "<<max;
}
```

```
enter a
5
enter b
3
Greatest number is = 5
```

Program 3.12:. Program to find the greatest of three numbers using if ... else ... if statement.

```
#include<iostream.h>

int main()
{
        int a,b,c,max;
        cout<<"enter a\n";
        cin>>a;
        cout<<"enter b\n";
        cin>>b;
        cout<<"enter c\n";
        cin>>c;
        if (a>b)
        {
                max=a;
        }
        else
        {
                max=b;
        }
        if (c>max)
```

(continues on next page)

(continued from previous page)

```
        {
                max=c;
        }
        cout<<"Greatest number is = "<<max;
}
```

```
enter a
3
enter b
7
enter c
1
Greatest number is = 7
```

4.3.2 switch . . . case statement

Program 3.13:. Program to perform arithematic operations using switch . . . case statement.

```
#include<iostream>
using namespace std;

int main()
{
        float x,y;
        int ch;
        cout<<"Addition: 1, Subtraction: 2, Multiplication: 3, Division: 4\n";
        cout<<"enter your choice\n";
        cin>>ch;
        cout<<"enter a number\n";
        cin>>x;
        cout<<"enter a number\n";
        cin>>y;
        switch(ch)
        {
                case 1:
                cout<<x + y;
                break;
                case 2:
                cout<<x - y;
                break;
                case 3:
                cout<<x * y;
```

(continues on next page)

(continued from previous page)

```
                break;
                case 4:
                cout<<x / y;
        }
        return 0;
}
```

4.4 Jump Statements

In the previous sections we have seen conditional control structures. Now we will see jump statements.

The Jump statements are used to change the flow of control. Jump statements defined in C++ are *break, continue, goto* and *return*. A standard library function *exit()* is also used to come out of the program while executing. Details about these statements are stated below.

The break Statement

Using *break* statement we can take the control of execution out of the loop even if the termination condition is not reached. It can be used to end an infinite loop, or to force it to end before its natural end. The *break* statement is mainly used in loops and switch statements. A *break* statement immediately terminates the loop or the switch statement, bypassing the remaining statements. The control then passes to the statement that immediately follows the loop or the *switch* statement. A *break* statement can be used in any of the *for, while* and *do ... while* loops.

Note that a *break* statement used in a nested loop affects only the inner loop in which it is used and not any of the outer loops. Similarly, a *break* statement used in a *switch* statement breaks out of that *switch* statement and not out of any loop that contains the *switch* statement.

The continue Statement:

The *continue* statement causes the program to skip the rest of the loop in the current iteration as if the end of the statement block had been reached, causing it to jump to the start of the following iteration. That is, *continue* statement is used to 'continue' the loop with its next iteration.

The goto Statement:

Goto statement allows to make an absolute jump to another point in the program. It can be used anywhere within a function or a loop. Use of *goto* causes an unconditional jump ignoring any type of nesting limitations. The destination point is identified by a label, which is then used as an argument for the *goto* statement. A label is made of a valid identifier followed by a colon (:). In general, this instruction has no concrete use in structured or object oriented programming.

The exit() Function:

The *exit()* function is a standard library function that terminates the entire program immediately and passes the control to the operating system. The purpose of *exit()* is to terminate the current program with a specific exit code. Its prototype is: *void exit (int exitcode);*. This function takes a single parameter, that is, exit status of the program and returns the same status to the operating system upon termination. The status can be either a zero or non-zero value, where zero shows successful termination and non-zero shows unsuccessful termination of the program.

Return Statement

In programming, *return* is a statement that tells the program to leave the function and return to the return address, directly after where the function was called. This statement does not mandatorily need any conditional statements. As soon as the statement is executed, the flow of the program stops and return the control from where it was called.

4.5 Illustration

The following program illustrate the use of *break* statement while searching for the presence of a particular item in the list provided. On the first occurance of the key that is searching for, the control of the flow will be taken out of the loop.

Program 1.1: Program to search the presence of element in an array using Linear Search method

```
//Prgoram to illustrate the use of break in searching for the item in an array;

#include <iostream>
using namespace std;

int main()
{
        int A[10];
        int i,n,key;
        cout<<"Enter how many numbers\n";
        cin>>n;

        //inputing of numbers
        cout<<"Enter the numbers\n";
        for (i=0; i<n; i=i+1)
        {
                cin>>A[i];
        }
        cout<<"Enter the key\n";
        cin>>key;
```

(continues on next page)

(continued from previous page)

```
        //displying of numbers
        for (i=0; i<n; i=i+1)
        {
                cout<<A[i]<<" ";
        }

        //Linear Search method
        for (i=0; i<n; i=i+1)
        {
                if(A[i]==key)
                {
                        cout<<"Found at "<<i+1;
                        break;
                }
        }
        if(i==n)
        {
                cout<<"Not found\n";
        }

}
```

Program 1.2: Program to illustrate the use of continue.

```
#include<iostream>
using namespace std;

int main()
{
        int i;
        for(i=0;i<10;i++)
        {
                if(i%2==0)
                {
                        continue;
                }
                else
                {
                        cout<<i;
                }
        }
}
```

The above program will skip even numbers and continue with next value of the iterator while iterating through the loop. It will print only the odd numbers.

Program 1.3: Program to illustrate the use of goto.

```
#include<iostream>
using namespace std;

int main()
{
        int n;
        cout<<"Enter a number: ";
        cin>>n;
        if (n%2==0)
        {
                goto loc;
        }
        else
        {
                cout<<"Odd Number";
        }
        loc:
        cout<<"Even Number";
}
```

The following program furnishes the use of *return* statement.

Program 1.4: Program to illustrate the use of return statement.

```
#include<iostream>
using namespace std;

int a=50,b=25;

int add()
{
        return(a+b);
}
int main()
{
        cout<<"Sum of a, b is : \n"<<add();
}
```

Finally we will see the use of *exit()* function. Note that, without executing the statements after the *exit()* function, the program gets terminated.

Program 1.5: Program to illustrate the use of return statement.

```
#include<iostream>
using namespace std;

int main(void)
```

(continues on next page)

(continued from previous page)

```
{
        cout<<"Your are going to exit";
        exit(0);
        cout<<"I am blocked, no use of typing anything here
                since control never enters";
}
```

5 Arrays

An array in C++ is a collection of items stored at contiguous memory locations and elements can be accessed randomly using indices of an array. They are used to store homogeneous type of elements as in the data type must be the same for all elements. Arrays can be used to store collection of primitive data types such as int, float, double, char, etc of any particular type. Array elements are accessed by using an integer index. Array index starts with 0 and goes till size of array minus 1. Arrays can be one dimensional or multidimensional. First we go through 1-D arrays.

5.1 1-Dimensional Arrays

Declaration of 1-D array in C++

```
dataType arrayName[arraySize];
```

For example:

```
int data[10];
float temp[25];
```

5.1.1 Initialization of Array

Array can be populated with data items, known as initialization, during the declaration time itself. An array initializer is the list of values separated by commas and enclosed by curly braces.

For example, see the following initialization:

```
int mark[3] = {34,45,49};
```

Another method to initialize array during declaration:

```
int mark[] = {34,45,49};
```

Note that we use loops to insert data items explicitly to the arrays, and also to traverse through arrays.

As illustration see the following examples.

5.1.2 Illustration

Program 1.1: Program to evaluate the average of set of numbers.

```
//Program to find the Average of set of numbers.;

#include <iostream>
using namespace std;

int main()
{
        int num[10];
        int i,n,sum=0;
        float avg;
        cout<<"Enter how many numbers\n";
        cin>>n;

        //inputing of numbers
        for (i=0; i<n; i=i+1)
        {
                cin>>num[i];
        }

        //summing of the numbers
        for (i=0; i<n; i=i+1)
        {
                sum=sum+num[i];
        }
        avg=sum/n;
        cout<<"Average = "<<avg;
}
```

Program 1.2: Program to sort a set of numbers using Selection sort method.

```
//Program to sort a set of numbers using Selection sort method.

#include <iostream>
using namespace std;

int main()
{
        int A[10];
        int i,j,n;
        cout<<"Enter how many numbers\n";
        cin>>n;

        //inputing of numbers
        cout<<"Enter the numbers\n";
        for (i=0; i<n; i=i+1)
        {
                cin>>A[i];
        }

        //displying of numbers
        cout<<"The numbers before sorting:\n";
        for (i=0; i<n; i=i+1)
        {
                cout<<A[i]<<" ";
        }

        //Selection sort method
        for (i=0; i<n-1; i=i+1)
        {
                for(j=i+1; j<n;j++)
                {
                        if(A[i]>A[j])
                        {
                                tmp=A[i];
                                A[i]=A[j];
                                A[j]=tmp;
                        }
        }
        //displying of numbers
        cout<<"The numbers after sorting:\n";
        for (i=0; i<n; i=i+1)
        {
                cout<<A[i]<<"  ";
        }
}
```

Program 1.3: Program to sort a set of numbers using Bubble sort method.

```
//Program to sort a set of numbers using Bubble sort method.;

#include <iostream>
using namespace std;

int main()
{
        int A[10];
        int i,j,n;
        cout<<"Enter how many numbers\n";
        cin>>n;

        //inputing of numbers
        cout<<"Enter the numbers\n";
        for (i=0; i<n; i=i+1)
        {
                cin>>A[i];
        }

        //displying of numbers
        cout<<"The numbers before sorting:\n";
        for (i=0; i<n; i=i+1)
        {
                cout<<A[i]<<" ";
        }

        //Bubble sort method
        for (i=0; i<n; i=i+1)
        {
                for(j=0; j<n-1-i;j++)
                {
                        if(A[j]>A[j+1])
                        {
                                tmp=A[j];
                                A[j]=A[j+1];
                                A[j+1]=tmp;
                        }
        }
        //displying of numbers
        cout<<"The numbers after sorting\n";
        {
                cout<<A[i]<<"  ";
        }
}
```

Program 1.4: Program to search the presence of element in an array using Linear Search method.

```
//Program to search the presence of element in an array.;

#include <iostream>
using namespace std;

int main()
{
        int A[10];
        int i,n,key;
        cout<<"Enter how many numbers\n";
        cin>>n;

        //inputing of numbers
        cout<<"Enter the numbers\n";
        for (i=0; i<n; i=i+1)
        {
                cin>>A[i];
        }
        cout<<"Enter the key\n";
        cin>>key;
        //displying of numbers
        for (i=0; i<n; i=i+1)
        {
                cout<<A[i]<<" ";
        }

        //Linear Search method
        for (i=0; i<n; i=i+1)
        {
                if(A[i]==key)
                {
                        cout<<"Found at "<<i+1;
                        break;
                }
        }
        if(i==n)
        {
                cout<<"Not found\n";
        }

}
```

Program 1.5: Program to search the presence of element in an array using Binary Search Algorithm.

```
//Program to search the presence of element in an array containing Sorted
        elements using Binary Search.;

#include <iostream>
using namespace std;

int main()
{
        int A[10];
        int i,n,key,count=0;
        int bot,top,mid;
        cout<<"Enter how many numbers\n";
        cin>>n;

        //inputing of numbers
        cout<<"Enter the numbers\n";
        for (i=0; i<n; i=i+1)
        {
                cin>>A[i];
        }
        cout<<"Enter the key\n";
        cin>>key;
        //displying of numbers
        for (i=0; i<n; i=i+1)
        {
                cout<<A[i]<<" ";
        }
        bot=0;
        top=n;
        //Linear Search method
        while(bot<=top)
        {
                mid=(bot+top)/2;
                if(A[mid]==key)
                {
                        cout<<"Found at "<<mid+1;
                        count=1;
                        break;
                }
                else if(A[mid]<key)
                {
                        bot=mid+1;
                }
                else
                {
                        top=mid-1;
```

(continues on next page)

(continued from previous page)

```
                }
        }
        if(count==0)
        {
                cout<<"Not Found!!!!\n";
        }

}
```

5.2 2-Dimensional Arrays

A two-dimensional array is stored in the form of the row-column matrix, where the first index is for the rows and second index for the columns. For each first index, the second index of an array traverse first while accessing the elements of an array.

Declaration of 2-D array in C++

```
dataType arrayName[arraySize1][arraySize2];
```

For example:

```
int data[10][15];
float temp[25][10];
```

5.2.1 Initialization of Array

Array can be populated with data items, known as initialization, during the declaration time itself. An array initializer is the list of values separated by commas and enclosed by curly braces.

For example, see the following initialization:

```
int a[4][2] = {{5,2}, {3,4}, {2,6}, {3,7}};
```

Another method to initialize array during declaration:

```
int mark[][] = {{5,2}, {3,4}, {2,6}, {3,7}};
```

As in the case of 1-D arrays, we use loops to insert data items explicitly to 2-D arrays, and also to traverse it. But we need two loops, one for indexing through rows and other for columns.

5.2.2 Illustration

Program 1.6: Program to display a matrix.

```
//Program to display a matrix.

#include <iostream>
using namespace std;

int main()
{
        int mat[5][5];
        int i,j,m,n;
        cout<<"Enter the order of the matrix\n";
        cin>>m>>n;

        //inputing of numbers
        cout<<"enter the elements of the matrix:\n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cin>>mat[i][j];
                }
        }

        //displaying of the matrix:
        cout<<"Matrix is: \n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cout<<mat[i][j]<<"  ";
                }
                cout<<"\n";
        }
}
```

Program 1.7: Program to add two matrices.

```
//Program to add two matrices.

#include <iostream>
using namespace std;

int main()
{
```

(continues on next page)

(continued from previous page)

```
int A[5][5],B[5][5],S[i][j];
int i,j,m,n;
cout<<"Enter the order of the matrices\n";
cin>>m>>n;

//inputing of matrix A:
cout<<"enter the elements of the matrix A:\n";
for (i=0; i<m; i=i+1)
{
        for (j=0; j<n; j=j+1)
        {
                cin>>A[i][j];
        }
}

//inputing of matrix B:
cout<<"enter the elements of the matrix B:\n";
for (i=0; i<m; i=i+1)
{
        for (j=0; j<n; j=j+1)
        {
                cin>>B[i][j];
        }
}
//displaying of the matrix A:
cout<<"Matrix is A: \n";
for (i=0; i<m; i=i+1)
{
        for (j=0; j<n; j=j+1)
        {
                cout<<A[i][j]<<"  ";
        }
        cout<<"\n";
}
//displaying of the matrix B:
cout<<"Matrix is B: \n";
for (i=0; i<m; i=i+1)
{
        for (j=0; j<n; j=j+1)
        {
                cout<<B[i][j]<<"  ";
        }
        cout<<"\n";
}
//Summing of A and B:
```

(continues on next page)

(continued from previous page)

```
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        S[i][j]=A[i][j]+B[i][j];
                }
        }
        //displaying of the sum matrix S:
        cout<<"Matrix is B: \n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cout<<S[i][j]<<"  ";
                }
                cout<<"\n";
        }
}
```

Program 1.8: Program to product of two matrices.

```
//Program to product of two matrices.

#include <iostream>
using namespace std;

int main()
{
        int A[5][5],B[5][5],P[5][5];
        int i,j,k,m,p,n;
        cout<<"Enter the order of the matrices\n";
        cin>>m>>p>>n;

        //inputing of matrix A:
        cout<<"enter the elements of the matrix A:\n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<p; j=j+1)
                {
                        cin>>A[i][j];
                }
        }

        //inputing of matrix B:
        cout<<"enter the elements of the matrix B:\n";
```

(continues on next page)

(continued from previous page)

```
        for (i=0; i<p; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cin>>B[i][j];
                }
        }
        //displaying of the matrix A:
        cout<<"Matrix is A: \n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<p; j=j+1)
                {
                        cout<<A[i][j]<<"  ";
                }
                cout<<"\n";
        }
        //displaying of the matrix B:
        cout<<"Matrix is B: \n";
        for (i=0; i<p; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cout<<B[i][j]<<"  ";
                }
                cout<<"\n";
        }
        //Product of A and B:
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                    P[i][j]=0;
                        for(k=0;k<p;k=k+1)
                        {
                        P[i][j]=P[i][j]+A[i][k]*B[k][j];
                        }
                }
        }
        //displaying of the sum matrix S:
        cout<<"Matrix is B: \n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
```

(continues on next page)

(continued from previous page)

```
                        cout<<P[i][j]<<"   ";
                }
                cout<<"\n";
        }
}
```

Program 1.9: Program to display the transpose of a matrix.

```
//Program to display transpose of a matrix.

#include <iostream>
using namespace std;

int main()
{
        int mat[5][5];
        int i,j,m,n;
        cout<<"Enter the order of the matrix\n";
        cin>>m>>n;

        //inputing of numbers
        cout<<"enter the elements of the matrix:\n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cin>>mat[i][j];
                }
        }

        //displaying of the matrix:
        cout<<"Matrix is: \n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cout<<mat[i][j]<<"   ";
                }
                cout<<"\n";
        }

        //displaying transpose matrix:
        cout<<"Matrix Transpose is: \n";
        for (i=0; i<n; i=i+1)
        {
```

(continues on next page)

(continued from previous page)

```
                for (j=0; j<m; j=j+1)
                {
                        cout<<mat[j][i]<<"  ";
                }
                cout<<"\n";
        }
}
```

Program 1.10: Program to display Unit matrix I_3.

```
//Program to display Unit matrix.

#include <iostream>
using namespace std;

int main()
{
        int mat[3][3];
        int i,j;

        //inputing of numbers
        for (i=0; i<3; i=i+1)
        {
                for (j=0; j<3; j=j+1)
                {
                        if (i==j)
                        {
                                mat[i][j]=1;
                        }
                        else
                        {
                                mat[i][j]=0;
                        }

                }
        }

        //displaying of the matrix:
        cout<<"Unit Matrix is: \n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cout<<mat[i][j]<<"  ";
                }
```

(continues on next page)

(continued from previous page)

```
                cout<<"\n";
        }
}
```

6 Strings

String is a collection of characters. Two types of string manipulations that are implemented in C++ are the one using the string class and other as a one-dimensional array of characters which is terminated by a null character '\0'. The latter concept is origianted from C language.

Syntax of declaring a string data type using C-style and object as string class of C++ is respectively given below:

```
char array_name [size];  (C- style.)
string string_name; (using string class).
```

Defining a string in C-style is shown below.

```
char name[ ]={'A', 'a', 'r', 'o', 'n', '\\0'};
```

or

```
char name[ ]="Aaron";
```

Defining a string as object of string class in different ways are shown below.

```
string s1;
s1= "Aaron";
string s2("Aaron");
string s3="Aaron";
string s4;
```

6.1 String Builtin Library Functions

Following are some of the built in library functions related to string avalable in C++.

1. strcpy()

strcpy(str1, str2) funtion copies string *str2* into string *str1*.

Program 1.6: Program to swap two Strings

```
#include <iostream>
using namespace std;
#include<string.h>

void swapstr(char A[], char B[])
{
        char tmp[10];
        strcpy(tmp,A);
        strcpy(A,B);
        strcpy(B,tmp);
}
int main()
{
        char X[10]="Aaron";
        char Y[10]="Bosh";
        cout<<"Before swapping\n";
        cout<<X<<" and "<<Y;
        swapstr(X,Y);
        cout<<"\nAfter swapping\n";
        cout<<X<<" and "<<Y;
}
```

```
Before swapping
Aaron and Bosh
After swapping
Bosh and Aaron
```

2. strcat()

strcat(str1, str2) function concatenates string *str2* onto the end of string *str1*.

Program 1.7: Program to concatenate string str2 onto the end of string str1.

```
#include <iostream>
using namespace std;
#include<string.h>

void concat(char str1[], char str2[])
{
        strcat(str1,str2);
}
int main()
{
        char str1[10]="Aaron";
```

(continues on next page)

(continued from previous page)

```
        char str2[10]="Bosh";
        cout<<"Before Concatenating\n";
        cout<<str1<<" and "<<str2;
        swapstr(str1,str2);
        cout<<"\nAfter Concatenating\n";
        cout<<str1<<" and "<<str2;
}
```

```
Before Concatenating
Aaron and Bosh
After Concatenating
AaronBosh
```

3. strlen()

strlen(str) function returns the length of string *str*.

Program 1.8: Program to find the length of a string.

```
#include <iostream>
using namespace std;
#include<string.h>

int lenstr(char str[])
{
        return strlen(str);
}
int main()
{
        char str[15]="Aaron Bosh";
        cout<<"Length of the string = "<<lenstr(str);
}
```

```
Length of the string = 10
```

4. strcmp()

strcmp(str1, str2) function returns 0 if *str1* and *str2* are the same; less than 0 if *str1* < *str2*; greater than 0 if *str1* > *str2*.

Program 1.9: Program to compare two strings for their equality.

```
#include <iostream>
using namespace std;
#include<string.h>

int compstr(char str1[], char str2[])
```

(continues on next page)

(continued from previous page)

```
{
        return strcmp(str1,str2);
}
int main()
{
        char str1[10];
        char str2[10];
        cout<<"Enter two string to compare\n";
        cin>>str1;
        cin>>str2;
        if(compstr(str1,str2)==0)
        {
                cout<<"The strings are same";
        }
        else
        {
                cout<<"Strings are different\n";
        }
}
```

```
Enter two string to compare
Aaron
Bosh
Strings are different

Enter two string to compare
Aaron
Aaron
The strings are same
```

6.2 The string Class

Operators defined for string

1. Assign

```
string s1;
string s2;
s1 = s2; // the contents of s2 is copied to s1.
```

2. Append

```
string s1( "Aaron" );
string s2( "Bosh" );
s1 += s2; // after appending s1 = "AaronBosh".
```

3. Equality

```
string s1( "Aaron" );
string s2( "Bosh" );
string s3( "Aaron" );
bool flag1 = ( s1 == s2 ); // flag1 = false now
bool flag2 = ( s1 == s3 ); // flag2 = true now
```

6.3 Illustration

Program 1.1: Program to display a name.

```
//Program to display a name.

#include <iostream>
using namespace std;

int main()
{
        char name[]="Aaron";
        cout<<name;
}
```

On executing the output will be;

```
Aaron
```

Program 1.2: Program to display a name.

```
//Program to display a name.

#include <iostream>
using namespace std;

int main()
{
        char name[20];
        cout<<"Enter the name\n";
        cin>>name;
        cout<<"Name:"<<name;
}
```

```
Enter the name
Aaron
Name:Aaron
```

Program 1.3: Program to display the address of a person.

```
//Program to display the address of a person.

#include <iostream>
using namespace std;

int main()
{
        char name[20];
        char hname[20];
        char post[20];
        int pin;
        cout<<"Enter the name, house name,post and pin\n";
        cin>>name>>hname>>post>>pin;
        cout<<"Name:"<<name<<endl;
        cout<<"House Name:"<<hname<<endl;
        cout<<"Post:"<<post<<endl;
}
```

```
Enter the name, house name,post and pin
Aaron
No.20/153
Trivandrum
670101
Name:Aaron
House Name:No.20/153
Post:Trivandrum
Pin:670101

#include <iostream>
```

Program 1.4: Program to display the address of three person.

```
//Program to display the address of three person.

#include <iostream>
using namespace std;

int main()
{
        char name[20];
```

(continues on next page)

(continued from previous page)

```
        char hname[20];
        char post[20];
        int pin;
        int i;
        for(i=0;i<3;i++)
        {
                cout<<"Enter the name, house name,post and pin\n";
                cin>>name>>hname>>post>>pin;
                cout<<"Address:"<<i+1<<endl;
                cout<<"Name:"<<name<<endl;
                cout<<"House Name:"<<hname<<endl;
                cout<<"Post:"<<post<<endl;
                cout<<"Pin:"<<pin<<endl;
        }
}
```

```
Enter the name, house name,post and pin
Aaron
No.19/234
Tvm
670101
Address:1
Name:Aaron
House Name:No.19/234
Post:Tvm
Pin:670101
Enter the name, house name,post and pin
Haseena
No.17/123
Cochin
670123
Address:2
Name:Haseena
House Name:No.17/123
Post:Cochin
Pin:670123
Enter the name, house name,post and pin
Mohanlal
No.2020
Chennai
456123
Address:3
Name:Mohanlal
House Name:No.2020
Post:Chennai
```

(continues on next page)

(continued from previous page)

```
Pin:456123
```

Program 1.5: Program to display the address of three person using 2-D array.

```
//Program to display the address of three person.

#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
        char name[3][20];
        char hname[3][20];
        char post[3][20];
        int pin[3];
        int i;
        for(i=0;i<3;i++)
        {
                cout<<"Enter the name, house name,post and pin\n";
                cin>>name[i];
                cin>>hname[i];
                cin>>post[i];
                cin>>pin[i];
        }
        cout<<"Name       House Name         Post      Pin\n\n";

        for(i=0;i<3;i++)
        {
        cout<<name[i]<<setw(15)<<hname[i]<<setw(18)<<post[i]<<setw(10)
                <<pin[i]<<endl;
        }
}
```

```
Enter the name, house name,post and pin
Aaron
No.20/IV
Trichur
670456
Enter the name, house name,post and pin
Haseena
No.19/V
Cochin
567543
Enter the name, house name,post and pin
```

(continues on next page)

(continued from previous page)

```
Mohan
2020/VI
Chennai
456432

Name        House Name          Post       Pin

Aaron       No.20/IV           Trichur    670456
Haseena        No.19/V          Cochin    567543
Mohan        2020/VI           Chennai    456432
```

Program 1.6: Program to display the address using string class.

```
//Program to display the address using string class.

#include <iostream>
#include<string>
using namespace std;

int main()
{
        string name;
        name="Aaron";
        string hname="No.2020/IV";
        string post("Cochin");
        cout<<"Name     House No.     Post\n\n";
        cout<<name<<"   "<<hname<<"   "<<post<<endl;
}
```

```
Name     House No.     Post

Aaron   No.2020/IV   Cochin
```

Program 1.7: Program to display the address using string class.

```
//Program to display the address using string class.

#include <iostream>
#include<string>
using namespace std;

int main()
{
        string name,hname,post;
        cout<<"Enter Name, House name, post"<<endl;
```

(continues on next page)

(continued from previous page)

```
        getline(cin,name);
        getline(cin,hname);
        getline(cin,post);
        cout<<"Name     House No.     Post\n\n";
        cout<<name<<"   "<<hname<<"     "<<post<<endl;
}
```

```
Enter Name, House name, post
Aaron
No.20/IV
Cochin
Name     House No.     Post

Aaron    No.20/IV      Cochin
```

Program 1.8: Program to display the address using string class.

```
//Program to display the address using string class.

#include <iostream>
#include<string>
using namespace std;

int main()
{
        string name[3],hname[3],post[3];
        int i;
        for(i=0;i<3;i++)
        {
                cout<<"Enter Name, House Name, Post of Person No. :"<<i+1<<endl;
                getline(cin,name[i]);
                getline(cin,hname[i]);
                getline(cin,post[i]);
        }
        cout<<"Name     House No.     Post\n\n";
        for(i=0;i<3;i++)
        {
                cout<<name[i]<<"   "<<hname[i]<<"   "<<post[i]<<endl;
        }
}
```

```
Enter Name, House Name, Post of Person No. :1
Aaron
No.20/IV
Cochin
```

(continues on next page)

(continued from previous page)

```
Enter Name, House Name, Post of Person No. :2
Edwin
No.2020
New York
Enter Name, House Name, Post of Person No. :3
Putin
No.21/RS
Russia
Name     House No.     Post

Aaron    No.20/IV   Cochin
Edwin   No.2020   New York
Putin   No.21/RS   Russia
```

7 Functions

A function is a set of executable statements that together perform a task. Every C++ program has at least one function, which is main, and all the most trivial programs can define additional functions. We can divide our code into separate functions. How we divide our code among different functions depends on our wish, but logically the division usually is so each function performs a specific task. A function declaration tells the compiler about a function's name, return type, and parameters(optional). A function definition provides the actual body of the function. The C++ standard library provides numerous built-in functions that your program can call. For example, function strcmp to compare two strings, and many more functions. A function is known as with various names like a method or a sub-routine or a procedure etc.

A function being a set of organized, reusable code, it is used to perform a single, related action. Functions provide better modularity for our application and a high degree of code reusing. As we already know, C++ gives us many built-in functions like sqrt(),clrscr(), etc. We can also create our own functions. These functions are called user-defined functions.

7.1 Defining a Function

The general form of a C++ function definition is as follows:

```
return_type function_name(parameter list)
{
        stmt-1;
        stmt-2;
        .
        .
        .
        stmt-n;
}
```

From the above function definition we can see that a function definition consists of a function header and a function body.

All the parts of a function definition is detailed here:

Return Type: A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword void.

Function Name: This is the actual name of the function. The function name and the parameter list together constitute the function signature.

Parameters: A parameter is like a placeholder. When a function is called, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.

Function Body: The function body contains a collection of statements that define what the function does.

7.2 Calling a Function

While creating a C++ function, we give a definition of what the function has to do. To use a function, you will have to call or invoke that function. When a program calls a function, program control is transferred to the called function. A called function performs defined task and when its return statement is executed or when its function ending closing brace is reached, it returns program control back to the main program.

To call a function, we simply need to pass the required parameters along with function name, and if function returns a value, then we can store returned value and for further use.

For example, consider the program for displaying the text "Hello, Aaron" using the concept of function:

We will illustrate, how to program using functions:

To perform the operation of displaying the the text "Hello, Aaron" , we write functions as given below:

1. The text "Hello, Aaron", will be printed using cout routine and it will be defined in a function named as *hello()*.
2. And the *main()* function will call *hello()* function and correspondingly the text will be displayed on the screen.

Program 4.1: Program to illustrate to display the text "Hello, Aaron" using function.

```
#include <iostream>
using namespace std;

void hello()
{
        cout<<"Hello, Aaron";
}
void main()
{
        hello();
}
```

Also note that in c++, functions can be called in two ways, namely;

1. Call by Value &
2. Call by Reference.

7.2.1 Call by Value Method

These two ways are generally differentiated by the type of values passed to them as parameters. The parameters passed to function are called actual parameters whereas the parameters received by function are called formal parameters.

In call by value method, the arguments are passed by value. Which means that when the function is used, the rvalues of arguments are copied into the parameters. The main effect is that even if the argument is a lvalue, modifying the corresponding parameter will not change the argument's value. For example :

Program 1.2: Program to swap two numbers.

```
#include <iostream>
using namespace std;

void swap(int x, int y)
{
        int t;
        t = x;
        x = y;
        y = t;
        cout<<"\nThe value of a and b inside function\n";
        cout<<"x = "<<x<<", y = "<<y;
}

int main()
{
```

(continues on next page)

(continued from previous page)

```
        int a = 10, b = 20;
        cout<<"Before swapping inside main\n";
        cout<<"a = "<<a<<", b = "<<b;
        swap(a, b);
        cout<<"\nAfter swapping inside main\n";
        cout<<"a = "<<a<<", b = "<<b;
        return 0;
}
```

```
Before swapping inside main
a = 10, b = 20
The value of a and b inside function
x = 20, y = 10
After swapping inside main
a = 10, b = 20
```

From the output we can see that, the swapped values are not reflected inside the main by call by value method.

7.2.2 Call by Reference Method

Whereas in call by reference, both the actual and formal parameters refer to same locations, so any changes made inside the function are actually reflected in actual parameters of caller.To do that, the & symbol specifies that the parameter and the argument correspond to the same rvalue. This is called passing an argument by reference.

Program 1.3: Program to increment a number using call by reference.

```
#include <iostream>
using namespace std;

void increment(int &a)
{
        a++;
        cout << "\nvalue inside the function "<< a <<endl;
}

int main()
{
        int x = 5;
        cout<<"Before increment\n";
        cout<<x;
        increment(x);
        cout << "After increment "<< x <<endl;
}
```

```
Before increment 5
value inside the function 6
After increment 6
```

Here using call by reference method the incremented reflected both inside function and the main.

7.3 Illustration

Program 1.4: Program to swap two numbers.

```
#include <iostream>
using namespace std;

void swap(int &x, int &y)
{
        int t;
        t = x;
        x = y;
        y = t;
        cout<<"\nInside the function, after swapping\n";
        cout<<"x = "<<x<<", y = "<<y;
}

int main()
{
        int a = 10, b = 20;
        cout<<"Inside main before swapping\n";
        cout<<"a = "<<a<<", b = "<<b;
        swap(a, b);
        cout<<"Inside main after swapping\n";
        cout<<"a = "<<a<<", b = "<<b;
}
```

```
Inside main before swapping
a = 10, b = 20
Inside the function, after swapping
x = 20, y = 10
Inside main after swapping
a = 20, b = 10
```

Here using call by reference method the swapping reflected both inside function and the main.

Program 4.2: Program to compute the Celsius by accepting Fahrenheit as inputs.

```
#include <iostream>
using namespace std;

int main()
{
        float F,C;
        cout<<"Enter the fahrenheit\n";
        cin>>F;
        C=5*(F-32)/9;
        cout<<"Temperature in Celsius"<<C;
}
```

We will illustrate, how to implement the above program using functions:

To perform the operation of converting the fahrenheit to celsius, we write different functions as given below:

1. To input the value in Celsius. We will write function named as *read()* to accomplish this.
2. To perform the operation of converting the fahrenheit to celsius, using the formula $C = 5 * (F - 32)/9$, we write the function *Fahr2Cel()*.
3. To display the result or the output, we write the function *display()*.
4. Finally the main() function to call i) *read(), ii)Fahr2CEl() and iii) display()*.

Note that the variables involved are C and F, and they are declared globally immediately after preprocessor directives. Hence all the functions can accesss them.

Program 4.3: Program to compute the Celsius by accepting Fahrenheit as inputs using functions.

```
#include<iostream>
using namespace std;

float C,F;

void read()
{
        cout<<"Enter the fahrenheit\n";
        cin>>F;
}

int Fahr2Cel()
{
        C=5*(F-32)/9;
}
```

(continues on next page)

(continued from previous page)

```
void display()
{
        cout<<"Temperature in Celcious = "<<C;
}

int main()
{
        read();
        Fahr2Cel();
        display();
}
```

We have already seen the program for performing various arithematic operations previously as given below:

Program 4.4: Program to illustrate different arithematic operations.

```
//Program to illustrate different arithematic operations using function.

#include <iostream>
using namespace std;

int main()
{
        int a=50,b=25, add,sub,mul;
        add = a+b;
        sub = a-b;
        mul = a*b;
        cout<<"Addition of a, b is : \n"<<add;
        cout<<"Subtraction of a, b is :\n"<<sub;
        cout<<"Multiplication of a, b is :\n"<< mul;
}
```

Now we will see how to incorporate the concept of function for performing arithematic operations like addition, subtration and multiplication;

To perform the arithematic operations, we write different functions as given below:

1. The input values a and b are declared and initialized globally immediately after preprocessor directives. Hence all the functions can accesss them.
2. To perform the operations $a+b$, $a-b$ and $a*b$, we write the function *add(), subtract()* and *mulitly()*.
3. The result or the output will be displayed, upon calling these functions from the main() function.

Program 4.5: Program to illustrate different arithematic operations using function.

```
//Program to illustrate different arithematic operations using function.

#include <iostream>
using namespace std;

int a=50,b=25;

int add()
{
        return(a+b);
}
int subtract()
{
        return(a-b);
}
int multiply()
{
        return(a*b);
}

int main()
{
        cout<<"Sum of a, b is : \n"<<add();
        cout<<"Difference of a, b is :\n"<<subtract();
        cout<<"Multiplication of a, b is :\n"<< multiply();
}
```

Above program can be implemented using function by passing parameters as given below;

Note that instead assigning the input values globally, we will pass these values to the functions through parameters as given below.

Program 4.6: Program to illustrate different arithematic operations using function by passing parameters.

```
//Program to illustrate different arithematic operations by passing parameters.

#include <iostream>
using namespace std;

int add(int a, int b)
{
        return(a+b);
}
int subtract(int a, int b)
```

(continues on next page)

(continued from previous page)

```
{
        return(a-b);
}
int multiply(int x,int y)
{
        return(x*y);
}
int main()
{
        cout<<"Sum of a, b is : \n"<<add(5,10);
        cout<<"Difference of a, b is :\n"<<subtract(25,15);
        cout<<"Multiplication of a, b is :\n"<< multiply(5,15);
}
```

For adding two numbers 5 and 10, we passed these numbers through the function call *add(5,10)* by call by value method from the *main()*. After performing the addition, function *add(5,10)* will return the result and cout routine will display it on the screen. And similarly other operations.

Program 4.7:. Program to compute the factorial of a number by accepting an integer as inputs using for loop.

```
//Prgoram to find the factorial.

#include <iostream.h>
int main()
{
        int i,n,f;
        cout<<"Enter the number to find the facorial\n";
        cin>>n;
        f=1;
        for(i = 1; i <= n; i++)
        f=f*i;
        cout<<"factorial = "<<f;
}
```

```
Enter the number to find the facorial
5
factorial = 120
```

Program 4.8: Program to compute the factorial of a number using function.

```
//Prgoram to find the factorial.

#include <iostream>
```

(continues on next page)

(continued from previous page)

```
using namespace std;

int fact(int n)
{
        int i,f;
        f=1;
        for(i = 1; i <= n; i++)
        {
                f=f*i;
        }
        return(f);
}
int main()
{
        int n;
        cout<<"Enter the number to find the facorial\n";
        cin>>n;
        cout<<"facorial = "<<fact(n);
}
```

The above program can be done by writing seperate functions:

1. To read the input value; by creating function *read()*: By calling the function read(), the input will be captured and that will be passed to the function *fact(input)* upon calling it.
2. To perform the computation; by creating function *fact(input)*: By calling the function *fact(input)*, the factorial computation will be performed and the result stored in *'f'* will be assigned to result and returned to the function display() as parameter, upon calling.
3. To display the result- by creating function *display(result)*: Calling of *display(fact(read()))* inside the *main()* function, will print the result returned by the function *fact(input)* by accepting the input passed by the function read().

Program 4.9:. Program to compute the factorial of a number using function.

```
//Prgoram to find the factorial.

#include <iostream>
using namespace std;

int read()
{
        int input;
        cout<<"Enter the number to find the facorial\n";
        cin>>input;
```

(continues on next page)

(continued from previous page)

```
        return(input);
}
void display(int result)
{
        cout<<"facorial = "<<result;
}

int fact(int input)
{
        int i,f,result;
        f=1;
        for(i = 1; i <= input; i++)
        {
                f=f*i;
        }
        result=f;
        return(result);
}

int main()
{
        display(fact(read()));
}
```

Program 4.10:. Program to compute the x^n using function.

```
//Prgoram to find the factorial.

#include <iostream>
using namespace std;

int x,n;
int read()
{
        cout<<"Enter the number base and exponent\n";
        cin>>x>>n;
}

void display(int result)
{
        cout<<"power = "<<result;
}

int power(int x, int n)
{
```

(continues on next page)

(continued from previous page)

```
        int i,p,result;
        p=1;
        for(i = 1; i <= n; i++)
        {
                p=p*x;
        }
        result=p;
        return(result);
}

int main()
{
        read();
        display(power(x,n));
}
```

Program 4.11:. Program to find the greatest of two numbers.

```
#include <iostream>
using namespace std;

int main()
{
        int a,b,max;
        cout<<"enter a\n";
        cin>>a;
        cout<<"enter b\n";
        cin>>b;
        if (a>b)
        {
                max=a;
        }
        else
        {
                max=b;
        }
        cout<<"Greatest number is = "<<max;
}
```

```
enter a
5
enter b
3
Greatest number is = 5
```

Program 4.12:. Program to find the greatest of two numbers using functions.

```
    #include <iostream>
    using namespace std;

    int a,b;

    void read()
    {
            cout<<"enter a\n";
            cin>>a;
            cout<<"enter b\n";
            cin>>b;
    }

    int max_two(int x, int y)
    {
            if (x>y)
            {
                    return(x);
            }
            else
            {
                    return(y);
            }

    }

int nain()
    {
            read();
            cout<<"Greatest number is = "<<max_two(a,b);
    }
```

```
enter a
5
enter b
3
Greatest number is = 5
```

Program 4.13:. Program to find the greatest of three numbers using if ... else ... if statement.

```
#include <iostream>
using namespace std;

int main()
{
```

(continues on next page)

(continued from previous page)

```
        int a,b,c,max;
        cout<<"enter a\n";
        cin>>a;
        cout<<"enter b\n";
        cin>>b;
        cout<<"enter c\n";
        cin>>c;
        if (a>b)
        {
                max=a;
        }
        else
        {
                max=b;
        }
        if (c>max)
        {
                max=c;
        }
        cout<<"Greatest number is = "<<max;
}
```

```
enter a
3
enter b
7
enter c
1
Greatest number is = 7
```

Program 4.14:. Program to find the greatest of three numbers using functions.

```
#include<iostream>
using namespace std;

int a,b,c;

void read()
{
        cout<<"enter a\n";
        cin>>a;
        cout<<"enter b\n";
        cin>>b;
        cout<<"enter c\n";
        cin>>c;
```

(continues on next page)

(continued from previous page)

```
}

int max_three(int x, int y, int z)
{
        int max;
        if (x>y)
        {
                max=x;
        }
        else
        {
                max=y;
        }
        if (z>max)
        {
                max=z;
        }
        return(max);
}
int main()
{
        read();
        cout<<"Greatest number is = "<<max_three(a,b,c);
}
```

```
enter a
3
enter b
7
enter c
1
Greatest number is = 7
```

Program 4.15:. Program to perform arithematic operations using switch . . . case statement.

```
#include<iostream>
using namespace std;

float a,b;

void read()
{
        cout<<"enter a\n";
        cin>>a;
```

(continues on next page)

(continued from previous page)

```
        cout<<"enter b\n";
        cin>>b;
}

float add(float x, float y)
{
        return(x+y);
}
float subtract(float x, float y)
{
        return(x-y);
}
float multiply(float x, float y)
{
        return(x*y);
}
float divide(float x, float y)
{
        return(x/y);
}

int menu()
{
    int ch;
        cout<<"Addition: 1, Subtraction: 2, Multiplication: 3, Division: 4\n";
        cout<<"enter your choice\n";
        cin>>ch;
        return(ch);
}

void operation()
{
        int ch;
        ch=menu();
        read();
        switch(ch)
        {
                case 1:
                cout<<"Sum = "<<add(a,b);
                break;
                case 2:
                cout<<"Difference = "<<subtract(a,b);
                break;
                case 3:
                cout<<"Product = "<<multiply(a,b);
```

(continues on next page)

(continued from previous page)

```
                break;
                case 4:
                cout<<"Quotient = "<<divide(a,b);
        }
}

int main()
{
        char repeat;
        do
        {
                operation();
                cout<<"Enter y to repeat\n";
                cin>>repeat;
        }
        while (repeat=='y');
}
```

Program 4.16: Program to display the multiplication table upto a given limit using function.

```
//Prgoram to display the multiplication table.

#include <iostream>
using namespace std;

void multdisplay(int n)
{
        for(i = 1; i <= n; i++)
        {
                for(j = 1; j <= n; i++)
                {
                        cout<<i*j<<"  ";
                }
                cout<<"\n";
        }
}
int main()
{
        int n;
        cout<<"Enter the limit\n";
        cin>>n;
        multdisplay(n);
}
```

Program 4.17: Program to evaluate the follwowing series sum

$1 + x + x^2 + ... + x^n$.

```
//Prgoram to find the series sum.

#include <iostream>
using namespace std;

int main()
{
        int i,n,t,x,sum;
        cout<<"Enter the base and exponent\n";
        cin>>x;
        cin>>n;
        t=1;
        sum=1;
        i = 1;
        while(i <= n)
        {
                t=t*x;
                sum=sum+t;
                i++;
        }
cout<<"Sum of the series = "<<sum;
}
```

```
Enter the base and exponent
2
5
Sum of the series = 62
```

Program 4.17: Program to evaluate the follwowing series sum using functions

$1 + x + x^2 + ... + x^n$.

```
//Prgoram to find the series sum.

#include <iostream>
using namespace std;

int x,n,sum=1;

void read()
{
        cout<<"Enter the base and exponent\n";
        cin>>x>>n;
}
```

(continues on next page)

(continued from previous page)

```
void seriessum(int x, int n)
{
        int i,t;
        t=1;
        i = 1;
        while(i <= n)
        {
                t=t*x;
                sum=sum+t;
                i++;
        }
}

void display()
{
        cout<<"Sum of the series = "<<sum;
}

int main()
{
        read();
        seriessum(x,n);
        display();
}
```

```
Enter the base and exponent
2
5
Sum of the series = 63
```

Program 4.18: Program to evaluate the follwowing series sum using functions

$1 - x + x^2 - .... + (-x)^n.$

```
//Prgoram to find the series sum;

#include <iostream>
using namespace std;

int x,n,sum=1;

void read()
{
        cout<<"Enter the base and exponent\n";
        cin>>x>>n;
}
```

(continues on next page)

(continued from previous page)

```
void seriessum(int x, int n)
{
        int i,t;
        t=1;
        i = 1;
        while(i <= n)
        {
                t=-t*x;
                sum=sum+t;
                i++;
        }
}

void display()
{
        cout<<"Sum of the series = "<<sum;
}

int main()
{
        read();
        seriessum(x,n);
        display();
}
```

```
Enter the base and exponent
2
5
Sum of the series = -21
```

7.4 Scope of Variables

Global Variables

Variables that are defined outside of all the functions and are accessible throughout the program are global variables and are said to have global scope. Once declared, these variables can be accessed by any function in the program.

```
#include<iostrem>
using namespace std;

int n;
```

(continues on next page)

(continued from previous page)

```
int main()
{
        cout<<n<< endl;
}
```

Here, n is a global variable since it is declared outside of the main function, that can be used in all the functions in the program.

Local Variables

Scope of the variables that is declared inside a function or a block is only inside that function or block, such variables called local variables. That is, these local variables can only be used within the function or block in which they are declared. In respect of functions, local variable can either be a variable which is declared in the body of that function or can be defined as function parameters in the function definition.

In the following example, variables a and b are declared in the definition of the function *add(int a, int b)* and are used within the function. Hence a and b have no scope outside the function. Similarly the argument x and y is declared locally inside *main()* and passed in respect of call by value method. Thus, any change in the values of a and b does not affect the values of x and y. So, a and b are the local variables for the function *add()* and similary x and y are local to *main()*.

```
include <iostream>
using namespace std;

int add(int a, int b)
{
        return(a + b);
}

int main()
{
        int x = 3, y = 5;
        int s;
        s = add(x, y);
        cout<<"Sum = "<< s << endl;
}
```

7.5 Storage classes

A storage class defines the scope visibility and life time of variables and/or functions within a C++ program. These specifiers precede the type that they modify. Thus a storage class

specifier is used to refine the declaration of a variable, a function, and parameters. There are following storage classes, which can be used in a C++ Program.

1.auto

The default storage class for all local variables is *auto*. And this can be used only with local variables.

2.register

The register storage class is used to define local variables that are frequently used like loop counters. And these variable will be stored in a register instead of RAM.

3.static

In C++, when a variable is declared as static, it causes only one copy of that variable to be shared by all part of the proram. The static modifier may also be applied to global variables. The main difference between local variable and static variable is that, the value of static variable persists till the end of the program.

For the objects declared with the static storage class specifier memory will be allocated when the program begins running and is freed when the program terminates. Static storage duration for a variable can have static duration but local scope.

4.extern

External variables are variables that are recognized globally, rather than locally. In other words, once declared, these variables can be used throughout, in the rest of the program. So an extern variable is nothing but a global variable initialized with a legal value where it is declared in order to be used elsewhere. The main purpose of using extern variables is that they can be accessed between two different files which are part of a large program.

Syntax:

```
storage_class var_data_type var_name;
```

The following program illustrates the use of strorage classes in C++.

Program 1.3: Program to illustrate the use of storage classes

```
#include<iostream>
using namespace std;

float  pi=3.14159;

int sumwithstatic(int a)
{
        static int s=0;
        s+=a;
        return s;
}
```

(continues on next page)

(continued from previous page)

```
int sumnostatic(int a)
{
        int s=0;
        s+=a;
        return s;
}

int main()
{
        extern float pi;
        auto a=5;
        register int i;
        for(i=0;i<5;i++)
        {
                cout<<"With static s, sum = "<<sumwithstatic(10)<<"; value of s,
                        getting retained in each call"<<endl;
        }
        for(i=0;i<5;i++)
        {
                cout<<"No static s, sum = "<<sumnostatic(10)<<"; initial value of s
                        is always 0."<<endl;
        }
        cout<<"At end, loop counter i in register = "<<i<<endl;
        cout<<"auto a = "<<a<<endl;
        cout<<"extern pi = "<<pi<<endl;
}
```

```
With static s, sum = 10; value of s, getting retained in each call
With static s, sum = 20; value of s, getting retained in each call
With static s, sum = 30; value of s, getting retained in each call
With static s, sum = 40; value of s, getting retained in each call
With static s, sum = 50; value of s, getting retained in each call
No static s, sum = 10; initial value of s is always 0.
No static s, sum = 10; initial value of s is always 0.
No static s, sum = 10; initial value of s is always 0.
No static s, sum = 10; initial value of s is always 0.
No static s, sum = 10; initial value of s is always 0.
At end,loop counter i in register = 5
auto a = 5
extern pi = 3.14159
```

7.6 Recursive function

A function can call itself in its definition statement. We call such a function a recursive function. In fact this is possible because at each call, new variables are allocated in the memory. And should ensure that at one point the function will not call itself anymore, so as to terminate.

Program 1.5: Program to find factorial using recursive function.

```
#include <iostream>
using namespace std;

int fact(int n)
{
        if(n == 0) return 1;
        else
        return n*fact(n-1);
}

int main()
{
        int n,f;
        cout<<"Enter the value of n\n";
        cin>>n;
        f = fact(n);
        cout<<n<<"! = "<< f << "\n";
}
```

```
Enter the value of n
5
5! = 120
```

Program 1.6: Program to print first n fibinocci nubers, using recursive function.

```
#include <iostream>
using namespace std;

int fib(int n)
{
        if (n <= 1)
                return 1;
        else
                return fib(n - 2) + fib(n - 1);
}

int main()
```

(continues on next page)

(continued from previous page)

```
{
        int n,f;
        cout<<"Enter the value of n = ";
        cin>>n;
        for(i=0;i<n;i++)
        {
                f = fib(i);
                cout<< f <<" ";
        }
}
```

```
Enter the value of n = 10
1 1 2 3 5 8 13 21 34 55
```

8 Structures

8.1 Defining a Structure

We have already come across primitive data types like int, float, char, etc and derived data types as array. The another data type that C++ permits is user defined data types. i.e., C++ allows us to create user-defined data types to aggregate data items. One of the simplest aggregate data types is the structure. To create a structure C++ has the facility named as struct. A struct (short for structure) allows us to group variables of mixed data types together into a single unit.

The Syntax of declaring a structure is:

```
struct struct_name
{
        data_type identifier1;
        data_type identifier2;
        .
        .
        .
};
```

Example-1;

```
struct student
{
        char st_name[20];
        char branch[10];
        char rollno[10];
};
```

Example-2;

```
struct faculty
{
        char fac_name[20];
        char dept[10];
        float salary;
};
```

Note the presence of semicolon at the end of the closing brace, while declaring the struct. Also note that struct allows us to aggregate heterogeneous data items unlike array.

The data members of the struct can be accessed using dot "." operator after defining it as given below.

```
student stud;
stud.st_name="Aaron";
stud.branch="Civil";
stud.rollno="MIT20CE01";
```

```
faculty fac;
fac.fac_name="Edwin";
fac.fac_dept="Civil",
fac.salary=250000;
```

What we have done is that, we created two data types namely student and faculty like the primitive data types int, float, etc. Hence we can use these user defined data types to access the data members of the corresponding struct as and when we require in our program.

Program 1.1: Program to find distance between two points in a plane.

```
//Program to find distance between two points in a plane.

#include <iostream>
using namespace std;
#include<math.h>

struct point
{
        int x;
        int y;
};

int main()
{
        point r1,r2;
        float distance;
        cout<<"Cordinates of the point\n";
        cin>>r1.x>>r1.y;
```

(continues on next page)

(continued from previous page)

```
        cout<<"Cordinate of the point\n";
        cin>>r2.x>>r2.y;
        cout<<"Points are \n";
        cout<<"("<<r1.x<<", "<<r1.y<<")"<<endl;
        cout<<"("<<r2.x<<", "<<r2.y<<")"<<endl;
        distance=sqrt((r2.x-r1.x)*(r2.x-r1.x)+(r2.y-r1.y)*(r2.y-r1.y));
        cout<<"Distance between the points ("<<r2.x<<", "<<r2.y<<")
                and ("<<r1.x<<", "<<r1.y<<") = "<<distance<<endl;
}
```

```
Cordinates of the point
1
1
Cordinate of the point
4
5
Points are
(1, 1)
(4, 5)
Distance between the points =(4, 5) and (1, 1) = 5
```

8.2 Array as data members in struct

C++ allows data members have to be arrays, besides primitive data types, inside the struct. This is demonstrated in the follwoing examples.

Program 1.2: Program to display the address of a person.

```
//Program to display the address of a person.

#include <iostream>
using namespace std;

struct addr
{
        char name[20];
        char hname[20];
        char post[20];
        int pin;
};

int main()
{
```

(continues on next page)

(continued from previous page)

```
        addr ad;
        cout<<"Enter the name, house name,post and pin\n";
        cin>>ad.name>>ad.hname>>ad.post>>ad.pin;
        cout<<"Name:"<<ad.name<<endl;
        cout<<"House Name:"<<ad.hname<<endl;
        cout<<"Post:"<<ad.post<<endl;
        cout<<"Pin:"<<ad.pin<<endl;
}
```

```
Enter the name, house name,post and pin
Aaron
No.20/IV
Cochin
670661
Name:Aaron
House Name:No.20/IV
Post:Cochin
Pin:670661
```

Note that in the above program, we have defined a data type of our own, named as **"addr"**, and that is used to declare variable of type *addr* by typing **"addr ad "** and using the **"."** operator we invoked the data members and populated them, for example, **" cin>>ad.name"**.

Program 1.3: Program to display a matrix using struct.

```
//Program to display a matrix using struct.

#include <iostream>
using namespace std;

struct matrix
{
        int mat[5][5];
        int m,n;
};

int main()
{
        matrix A;
        int i,j;
        cout<<"Enter the order of the matrix\n";
        cin>>A.m>>A.n;

        //inputing of numbers
        cout<<"enter the elements of the matrix:\n";
```

(continues on next page)

(continued from previous page)

```
        for (i=0; i<A.m; i=i+1)
        {
                for (j=0; j<A.n; j=j+1)
                {
                        cin>>A.mat[i][j];
                }
        }

        //displaying of the matrix:
        cout<<"Matrix is: \n";
        for (i=0; i<A.m; i=i+1)
        {
                for (j=0; j<A.n; j=j+1)
                {
                        cout<<A.mat[i][j]<<"  ";
                }
                cout<<"\n";
        }
}
```

```
Enter the order of the matrix
3
4
enter the elements of the matrix:
2
1
3
4
2
5
4
6
3
2
1
4
Matrix is:
2  1  3  4
2  5  4  6
3  2  1  4
```

8.3 Array of Structures

We can also make an array of structures. In the previous example in structures, we stored address of a person. Now suppose we need to store the data of more members. Declaring that many separate variables of the structure is not a good practice. In such cases we need to create an array of structures. For example,

Program 1.4: Program to display the address of a three person.

```
//Program to display the address of a person.

#include <iostream>
using namespace std;

struct addr
{
        char name[20];
        char hname[20];
        char post[20];
        int pin;
};

int main()
{
        int i;
        addr ad[3];
        for(i=0;i<3;i++)
        {
                cout<<"Enter the name, house name,post and pin\n";
                cin>>ad[i].name>>ad[i].hname>>ad[i].post>>ad[i].pin;
        }
        for(i=0;i<3;i++)
        {
                cout<<"Name:"<<ad[i].name<<endl;
                cout<<"House Name:"<<ad[i].hname<<endl;
                cout<<"Post:"<<ad[i].post<<endl;
                cout<<"Pin:"<<ad[i].pin<<endl;
        }
}
```

```
Enter the name, house name,post and pin
Aaron
No.2020/VI
Cochin
765674
Enter the name, house name,post and pin
```

(continues on next page)

(continued from previous page)

```
Edwin
No.1920/V
Poland
567545
Enter the name, house name,post and pin
Newton
No.Ru234
Russia
32345
Name:Aaron
House Name:No.2020/VI
Post:Cochin
Pin:765674
Name:Edwin
House Name:No.1920/V
Post:Poland
Pin:567545
Name:Newton
House Name:No.Ru234
Post:Russia
Pin:32345
```

8.4 Nested Structures

When a structure contains another structure, it is called nested structure or structure within a structure. The members of structures are accessed using dot operator. Following example details the implementation of nested structures.

Program 1.5: Program to display the student details.

```
//Program to display the address of a person.

#include <iostream>
using namespace std;

struct addr
{
        char hname[20];
        char post[20];
        int pin;
};
struct dept
{
```

(continues on next page)

(continued from previous page)

```
        char dept_name[20];
        char dept_loc[20];
};
struct student
{
        char st_name[20];
        int age;
        addr ad;
        dept dt;
};
int main()
{
        student stud;
        cout<<"Enter the name and age\n";
        cin>>stud.st_name;
        cin>>stud.age;
        cout<<"Enter House name,post and pin\n";
        cin>>stud.ad.hname>>stud.ad.post>>stud.ad.pin;
        cout<<"Enter dept name and location\n";
        cin>>stud.dt.dept_name>>stud.dt.dept_loc;
        cout<<"Name:"<<stud.st_name<<endl;
        cout<<"Age:"<<stud.age<<endl;
        cout<<"House Name:"<<stud.ad.hname<<endl;
        cout<<"Post:"<<stud.ad.post<<endl;
        cout<<"Pin:"<<stud.ad.pin<<endl;
        cout<<"Dept Name:"<<stud.dt.dept_name<<endl;
        cout<<"Dept Location:"<<stud.dt.dept_loc<<endl;
}
```

```
Enter the name and age
Aaron
16
Enter House name,post and pin
No.20/IV
Cochin
670543
Enter dept name and location
Engineering
Poland
Name:Aaron
Age:16
House Name:No.20/IV
Post:Cochin
Pin:670543
Dept Name:Engineering
```

(continues on next page)

(continued from previous page)

```
Dept Location:Poland
```

Program 1.6: Program to display three students details.

```
//Program to display three students details.

#include <iostream>
using namespace std;

struct addr
{
        char hname[20];
        char post[20];
        int pin;
};
struct dept
{
        char dept_name[20];
        char dept_loc[20];
};
struct student
{
        char st_name[20];
        int age;
        addr ad;
        dept dt;
};
int main()
{
        int i;
        student stud[3];
        for(i=0;i<3;i++)
        {
                cout<<"Enter the name and age\n";
                cin>>stud[i].st_name;
                cin>>stud[i].age;
                cout<<"Enter House name,post and pin\n";
                cin>>stud[i].ad.hname>>stud[i].ad.post>>stud[i].ad.pin;
                cout<<"Enter dept name and location\n";
                cin>>stud[i].dt.dept_name>>stud[i].dt.dept_loc;
        }
        for(i=0;i<3;i++)
        {
                cout<<"Name:"<<stud[i].st_name<<endl;
                cout<<"Age:"<<stud[i].age<<endl;
```

(continues on next page)

(continued from previous page)

```
                cout<<"House Name:"<<stud[i].ad.hname<<endl;
                cout<<"Post:"<<stud[i].ad.post<<endl;
                cout<<"Pin:"<<stud[i].ad.pin<<endl;
                cout<<"Dept Name:"<<stud[i].dt.dept_name<<endl;
                cout<<"Dept Location:"<<stud[i].dt.dept_loc<<endl;
        }
}
```

```
Enter the name and age
Aaron
16
Enter House name,post and pin
No.20/VI
Cochin
65753
Enter dept name and location
Enginering
Poland
Enter the name and age
Edwin
75
Enter House name,post and pin
No.19/VI
Germany
34543
Enter dept name and location
Electrical
Berlin
Enter the name and age
Newton
345
Enter House name,post and pin
No.15/VI
England
5435
Enter dept name and location
Physics
Cambridge

Name:Aaron
Age:16
House Name:No.20/VI
Post:Cochin
Pin:65753
Dept Name:Enginering
```

(continues on next page)

(continued from previous page)

```
Dept Location:Poland
Name:Edwin
Age:75
House Name:No.19/VI
Post:Germany
Pin:34543
Dept Name:Electrical
Dept Location:Berlin
Name:Newton
Age:345
House Name:No.15/VI
Post:England
Pin:5435
Dept Name:Physics
Dept Location:Cambridge
```

8.5 Struct and Functions

C++ have the facility to pass struct as parameters and return types also have to be struct as shown below.

Program 1.7: Program to Add two matrices using functions passing struct as argument.

```
//Program to Add two matrices.

#include <iostream>
using namespace std;

struct matrix
{
        int mat[5][5];
        int m,n;
};

matrix read()
{
        int i,j;
        matrix A;
        cout<<"enter the number of rows and columns\n";
        cin>>A.m>>A.n;
        cout<<"enter the matrix elements\n";
        for(i=0;i<A.m;i++)
```

(continues on next page)

(continued from previous page)

```
        {
                for(j=0;j<A.n;j++)
                {
                        cin>>A.mat[i][j];
                }
        }
        return A;
}

matrix add(matrix X, matrix Y)
{
        matrix T;
        int i,j;
        T.m=X.m;
        T.n=X.n;
        for(i=0;i<T.m;i++)
        {
                for(j=0;j<T.n;j++)
                {
                        T.mat[i][j]=X.mat[i][j]+Y.mat[i][j];
                }
        }
        return T;
}

matrix disp(matrix A)
{
        int i,j;
        for(i=0;i<A.m;i++)
        {
        for(j=0;j<A.n;j++)
                {
        cout<<A.mat[i][j]<<"   ";
                }
        }
        cout<<"\n";
}

int main()
{
        cout<<"\nFor the Matrix A\n";
        matrix A = read();
        cout<<"\nFor the Matrix B\n";
        matrix B = read();
        cout<<"\nMatrix A:\n\n";
```

(continues on next page)

(continued from previous page)

```
        disp(A);
        cout<<"\nMatrix B:\n\n";
        disp(B);
        matrix C=add(A,B);
        cout<<"Sum of the Matrices is :\n";
        disp(C);
}
```

```
For the Matrix A
enter the number of rows and columns
2
3
enter the matrix elements
1
2
3
4
5
6

For the Matrix B
enter the number of rows and columns
2
3
enter the matrix elements
6
5
4
3
2
1

Matrix A:

1   2   3
4   5   6

Matrix B:

6   5   4
3   2   1
Sum of the Matrices is :
7   7   7
7   7   7
```

9 Pointers

We have already come across variables as identifiers for locations in the computer's memory which can be accessed by these identifier names, and not by paying much ado about the physical address of the data in memory.

When a variable is declared, the memory needed to store its value is assigned a specific location in memory (its memory address). In general, it is operating system and not C++ programs, decide the exact memory addresses where these variables have to be stored. To have an explicit knowedge of these memeory addresses, C++, has the mechanism known as *address operator* (&) or reference operator. And correspondingly there is a dereferencing operator known as *pointer(*)* that gives the contents of the memory location referenced by the address operator &.

Hence, pointers are variables, which can hold the address of some other variables.

Syntax of declaring a pointer variable is:

The type of a pointer depends on the type of the variable it points to.

From what follows is that, in connection with pointers we have two unary operators, namely the * operator and the & operator. Hence, if a is a variable, then &a gives the address of the variable and *(&a) gives the contents, the value of a.

Also note that, since pointers are also variables, they can be passed as input parameters to functions. We can pass arrays as input to functions. And arrays are nothing but pointers. Hence we can pass pointers, in place of arrays and vise-versa.

Program 1.1: Program to illustrate use of & and * operators.

```
#include<iostream>
using namespace std;

int main()
{
```

(continues on next page)

(continued from previous page)

```
        int a=5;
        cout<<"a = "<<a;
        cout<<"\nAddress is: "<<&a<<endl;
        cout<<"By pointer, a = "<<*(&a);
}
```

```
a = 5
Address is: 0x7fffd7e7c134
By pointer, a = 5
```

In the above program, *(&a) gives the value of a, i.e, 5. Which is contained in the memory location &a = 0x7fffd7e7c134. Note that we can store this memory location in variable of type pointer and to distinguish this from other normal variable, we use the symbol * before the pointer variable name, eg. * ptra. Hence, the assignement *ptra = &a means, the pointer variable *ptra* holds the address of the variable a, and to retrive the contents, we have to use * with pointer varible. Using this idea, we rewrite the above program in the following manner.

Program 1.2: Program to illustrate use of & and * operators.

```
#include<iostream>
using namespace std;

int main()
{
        int a=10;
        int *ptra;
        ptra=&a;
        cout<<"a = "<<a;
        cout<<"\nAddress is: "<<&a<<endl;
        cout<<"By pointer, a = "<<*ptra<<endl;
}
```

```
a = 10
Address is: 0x7fffefad2dac
By pointer, a = 10
```

9.1 Pointers and Array

Array names essentially are pointers. Array elements are stored in contiguous (consecutive) locations in memory. For example, consider an array, int A[5];

A is a pointer to the first element of the array. That is, *A is the same as A[0].

A+i is a pointer to A[i]. i.e, *(A+i), is equal to A[i].

The program for displaying the contents of an array in the classical way, without using pointers is given below.

Program 1.3: Program to display an array having n elements.

```
#include<iostream>
using namespace std;

int main()
{
        int a[10];
        int i,n;
        cout<<"Enter the value of n\n";
        cin>>n;
        for(i=0;i<n;i++)
        {
                cin>>a[i];
        }
        for(i=0;i<n;i++)
        {
                cout<<a[i]<<"  ";
        }
}
```

With the help of pointers we convert the program in the following way.

Program 1.4: Program to display an array having n elements using pointers.

```
#include<iostream>
using namespace std;

int main()
{
        int a[10];
        int i,n;
        cout<<"Enter the value of n = ";
        cin>>n;
        for(i=0;i<n;i++)
        {
                cin>>*(a+i);
        }
        cout<<"\n Elements are:\n";
        for(i=0;i<n;i++)
        {
                cout<<*(a+i)<<"  ";
        }
}
```

```
Enter the value of n = 5
4
3
7
6
8
Elements are:
4  3  7  6  8
```

Program 1.5: Program to display a matrix using pointers

```
//Prgoram to display a matrix

#include <iostream>
using namespace std;

int main()
{
        int mat[5][5];
        int i,j,m,n;
        cout<<"Enter the order of the matrix\n";
        cin>>m>>n;
        cout<<"enter the elements of the matrix:\n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cin>>*(*(mat + i) +j);
                }
        }
        cout<<"Matrix is: \n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cout<<*(*(mat + i) +j)<<"  ";
                }
                cout<<"\n";
        }
}
```

9.2 Pointers and String

A String is defined as an array of characters and for example,

```
char* s;
```

means that s is a pointer to (or address of) a character, i.e., s is the address of the first character (byte) of the string.

Hence a String in C++ is a char *pointer to a sequence of characters terminated by a null character '\0'.

Declaring a character array

```
char str[10];
```

This means that this is a character array i.e. a string with 10 characters.

Note that this can also be represented as

```
char *str;
```

Initializing a String

A constant string s can be initialized as

```
char* s = "Aaron";
```

9.3 Pointers and Functions

A pointer to function is declared with the * ,the general statement of its declaration is:

```
return_type (*function_name)(arguments)
```

Program 1.10: Program to illustrate the use function pointer.

```
#include <iostream>
using namespace std;

int sum(int a, int b)
{
        return a+b;
}

int main()
{
        int (*fptr)(int,int);
        int x,y;
        cout<<"Enter two numbers\n";
        cin>>x>>y;
```

(continues on next page)

(continued from previous page)

```
        fptr = sum;
        cout<<"Sum = "<<fptr(x,y);
}
```

10 Basics of Object Oriented Programming

10.1 Concepts of OOPS

Object oriented programming is a programming paradigm based on objects (having both data and methods) that aims to incorporate the advantages of modularity and reusability.

Basic Concepts of Object Oriented Programming is detailed below.

1. Objects.

Objects, which are instances of classes are the basic run time entities in an object oriented paradigm. An object may have a physical existence, like a customer, a vehicle, etc.; or an intangible conceptual existence, like a project, a process, etc. The basic idea behind object oriented programing is to combine both data and procedures into a single unit and this is facilitated through objects. The term objects means a combination of data and program that represent some real world entity.

2. Classes.

A class stands for a collection of objects having same characteristics that exhibit common behavior. Class provides the template for the objects that can be created from it. Creation of an object as a member of a class is called instantiation. Thus, an object is an instance of a class.

In C ++ a class is a new data type that contains member variables and member functions that operate on the variables.

3. Data abstraction and encapsulation.

Abstraction is the process of representing essential features without including the back ground details or explanations. Classes use the concept of abstraction and are defined as data members such as size, name, age, salary and functions to operate on these attributes.

The wrapping up of data and functions into a class as single unit is called encapsulation.

The functions that are wrapped inside the class only can access the data members of that class and not by the other outside functions. These functions provide the interface between the objects data and the program.

4. Inheritance.

The ability of one class to acquire the properties from another class is called inheritance. The most important advantage of inheritance is code reusability. In the inheritance process there is a class that shares its properties with the inheriting class. The class that responsible for sharing its properties is called base class and the inheriting class is called derived class. Consequently a derived class will be having additional features of the base class along with those of its own.

5. Polymorphism.

Polymorphism means the ability to take more than one form. The programing language feature that allows a function or operator to be given more than one definition is called poymorphism. Polymorphism allows to redefine the conventional meaning of oparetors. For example, the expression $a + b$ denotes the sum of a and b, for different types of a and b like integers , float and complex numbers. This operator "+" can even redefine to mean the concatenation of two strings. And this kind of polymorphism is known as operator overloading. Similarly C++ allows, function overloading also, by exibiting the use of same function name for different behaviors. The types of the arguments with which the function is invoked determines which definition will be used.

6. Dynamic binding.

Dynamic binding is the process of linking of a function call to the actual code of the function at run time. That is, in dynamic binding, the actual code to be executed is not known to the compiler until run time. The concept of dynamic binding is implemented with the help of inheritance and run time polymorphism.

7. Message passing.

Message passing involves specifying the name of the object, the name of the function (message) and information to be sent.

That is, in object oriented programing, set of objects need to communicate with each other. This is accomplished through message passing. A message for an object is a request for execution of a function to get the result of the execution. To achieve this, the object has to invoke the required function.

Class is a set of objects that share common properties and relationships. In C++, a class provides to create a new data type that contains member variables and member functions that operates on the variables. A class is defined with the keyword *class*. It allows the data to be hidden, if necessary from external use. By defining a class, we are creating a new abstract data type that can be treated like any other built in data types.

The class declaration describes the type and scope of its members. The class function definition describes how the class functions are implemented.

Syntax

```
class class-name
{
        private:
                variable declarations;
                function declaration;
        public:
                variable  declarations;
                function declaration;
};
```

The members that have been declared as private can be accessed only from within the class. Whereas, public members can be accessed from outside the class also. The data members of a class are private by default. Hence, to achieve data hiding, the one of the main features of object orientation, no need of explicitly declaring a data member to be private using the keyword private. Ultimately binding of data and functions together into a single unit with help of class type variable facilitates the encapsulation.

Example

```
class student
{
        char name[20];
        int age;
        public:
                void input(void);
                void display(void);
};
```

The class student contains two data members and two function members, the data members are private by default while both the functions are public by declaration. The function *input()* can be used to assign values to the member variables *name* and *age*, and *display()* for displaying their values. These functions provide the only access to the data members from outside the class.

10.2 Creating Objects

Once a class has been declared we can create variables of that type by using the class name.

Example *student st;*

creates a variables *st* of type *student*.

In C++, the class variables are known as objects. Hence *st* is called an object of type *student*.

Accessing of data members

The private data of a class can be accessed only through the member functions of that class. The *main()* cannot contains statements that the access *name* and *age* directly. Using dot "." operator we can access the member functions.

For example, *st.input();*

Program 1.1: Program to display student name and age.

```
//Program to display student name and age.

#include <iostream>
using namespace std;

#include<iostream>
using namespace std;

class student
{
        char name[20];
        int age;
        public:
                void input()
                {
                        cout<<"Enter the name and age \n";
                        cin>>name>>age;
                }
                void display()
                {
                        cout<<"Name: "<<name<<endl;
                        cout<<"Age : "<<age<<endl;
                }
};
int main()
{
        student st;
        st.input();
        st.display()
}
```

In the above program the member functions *input()* and *display()* are defined inside the the data type *student* itself. It is possible to declare these functions inside and define them outside the class *student* as shown below.

Program 1.2: Program to display student name and age.

```
//Program to display student name and age.
```

(continues on next page)

(continued from previous page)

```
#include <iostream>
using namespace std;

#include<iostream>
using namespace std;

class student
{
        char name[20];
        int age;
        public:
                void input();
                void display();
};

void student::input()
        {
                cout<<"Enter the name and age \n";
                cin>>name>>age;
        }
void student::display()
        {
                cout<<"Name: "<<name<<endl;
                cout<<"Age : "<<age<<endl;
        }
int main()
{
        student st;
        st.input();
        st.display()
}
```

10.3 Constructor

We can initialize an object. A constructor enable us to initialize the objects of its class. The constructor is invoked whenever an object of its associated class is created.

A constructor is declared inside the following class and defined outside the class as given below. Note that the constructor has the same name as that of class.

```
class student
{
```

(continues on next page)

(continued from previous page)

```
        char name[20];
        int age;
        public:
        student(){}
        void input(void);
        void display(void);
}

student :: student(void)
{
        age=16;
}
```

The following program illustrate the use of constructor to initialize the class members *name* and *age* by defining it inside the class.

Program 1.3: Program to illustrate the use of Constructor.

```
//Program to initialize student name and age.

#include<iostream>
using namespace std;
#include<string.h>

class student
{
        char name[20];
        int age;
        public:
                student()
                {
                        age=16;
                        strcpy(name,"Aaron");
                }
                void input();
                void display();
};

void student::display()
{
        cout<<"Name: "<<name<<endl;
        cout<<"Age : "<<age<<endl;
}
int main()
{
        student st;
```

(continues on next page)

(continued from previous page)

```
        st.display();
}
```

We can define the constructor outside the class also.

Program 1.4: Program to illustrate the use of Constructor.

```
//Program to initialize student name and age.

#include<iostream>
using namespace std;

class student
{
        char name[20];
        int age;
        public:
                student();
                void input();
                void display();
};

student()
{
        age=16;
        strcpy(name,"Aaron");
}

void student::display()
{
        cout<<"Name: "<<name<<endl;
        cout<<"Age : "<<age<<endl;
}
int main()
{
        student st;
        st.display();
}
```

Program 1.5: Program to illustrate the use of Constructor by passing parameters.

```
//Program to initialize student name and age.

#include<iostream>
using namespace std;
```

(continues on next page)

(continued from previous page)

```
class student
{
        char name[20];
        int age;
        public:
                student(char[], int);
                void input();
                void display();
};

student(char nm[], int ag)
{
        strcpy(name,nm);
        age=ag;
}

void student::display()
{
        cout<<"Name: "<<name<<endl;
        cout<<"Age : "<<age<<endl;
}
int main()
{
        student st("Aaron",16);
        st.display();
}
```

Program 1.6: Program to find distance between two points in a plane.

```
//Program to find distance between two points in a plane.

#include <iostream>
using namespace std;
#include<math.h>

class point
{
        int x;
        int y;
        public:
                void readxy();
                void dist();
};
```

(continues on next page)

(continued from previous page)

```
void point::readxy()
{
        cout<<"Enter the cordinates\n";
        cin>>x;
        cin>>y;
}
void point::dist()
{
        point r1,r2;
        float d;
        r1.readxy();
        r2.readxy();
        d=sqrt((r2.x-r1.x)*(r2.x-r1.x)+(r2.y-r1.y)*(r2.y-r1.y));
        cout<<"Points are \n";
        cout<<"("<<r1.x<<", "<<r1.y<<")"<<endl;
        cout<<"("<<r2.x<<", "<<r2.y<<")"<<endl;
        cout<<"Distance between the points ("<<r2.x<<", "<<r2.y<<")
                and ("<<r1.x<<", "<<r1.y<<") = "<<d;
}

int main()
{
        point r;
        r.dist();
}
```

Program 1.7: Program to display matrix using class.

```
#include <iostream>
using namespace std;

class matrix
{
        int mat[5][5];
        int m,n;
        public:
                void readMat();
                void readOrder();
                void dispMat();
};

void matrix::readOrder()
{
        cout<<"Enter the order of the matrix\n";
        cin>>m>>n;
```

(continues on next page)

(continued from previous page)

```
}

void matrix::readMat()
{
        int i,j;
        cout<<"enter the elements of the matrix:\n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cin>>mat[i][j];
                }
        }
}

void matrix::dispMat()
{
        int i,j;
        cout<<"Matrix is: \n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cout<<mat[i][j]<<"  ";
                }
                cout<<"\n";
        }
}

int main()
{
        matrix A;
        A.readOrder();
        A.readMat();
        A.dispMat();
}
```

Program 1.8: Program to add two matrices using class.

```
#include <iostream>
using namespace std;

class matrix
{
        int mat[5][5];
```

(continues on next page)

(continued from previous page)

```
        int m,n;
        public:
                void readMat();
                void readOrder();
                void dispMat();
                void addMat();
};

void matrix::readOrder()
{
        cout<<"Enter the order of the matrix\n";
        cin>>m>>n;
}

void matrix::readMat()
{
        int i,j;
        cout<<"enter the elements of the matrix:\n";
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cin>>mat[i][j];
                }
        }
}

void matrix::dispMat()
{
        int i,j;
        for (i=0; i<m; i=i+1)
        {
                for (j=0; j<n; j=j+1)
                {
                        cout<<mat[i][j]<<"  ";
                }
                cout<<"\n";
        }
}

void matrix::addMat()
{
        int i,j;
        matrix S,X,Y;
        X.readOrder();
```

(continues on next page)

(continued from previous page)

```
        X.readMat();
        Y.readOrder();
        Y.readMat();
        S.m=X.m;
        S.n=X.n;
        for(i=0;i<S.m;i++)
        {
                for(j=0;j<S.n;j++)
                {
                        S.mat[i][j]=X.mat[i][j]+Y.mat[i][j];
                }
        }
        cout<<"\nMatrix A:\n";
        X.dispMat();
        cout<<"Matrix B:\n";
        Y.dispMat();
        cout<<"Matrix  A + B :\n";
        S.dispMat();
}

int main()
{
        matrix A;
        A.addMat();
}
```

10.4 Destructor

A destructor, does the work of removing the objects that have been created by a constructor. The destructor is a member function with the same name as that of class name similar to constructor, but difference is that, a destructor name precedes a tilde symbol.

For Example,

```
~student( ){ }
```

From the above declaration it is clear that the destructor does not have any argument or return type. Also note that, even though to release the memory that is no longer accessible, the compiler implicitly calls destructor method upon exiting from the program, it is always better to call explicitly the destructor method in a program so as to make available unused memory for later use.

10.5 Inheritance

Inheritance is the mechanism of deriving new class from existing class. It provides the idea of reusability. C++ supports the concept of reusability. The C++ classes can be used again in several ways. Once a class has been written and tested, it can be adopted by another programmers. Inheritance is often referred to as IS-A relationship because very object of the class being defined 'is' also an object of inherited class. The old class is called 'Base' or 'parent' class and the new one is called 'Derived' or 'child' class.

Defining Derived Classes

A derived class is specified by defining its relationship with the base class in addition to its own details. The general syntax of defining a derived class is as follows:

```
class child : Access specifier parent
{
        members derived class;
};
```

The colon indicates that the class name is derived from the base class name. The access specifier or the visibility mode is optional and it can be public, private or protected. By default it is private. Visibility mode describes the status of derived features.

Example,

```
class parent
{
        members of parent;
};

class child : public parent
{
        members of child ;
};

class child : parent
{
        members of child;
};
```

In the inheritance, some of the base class data elements and member functions are inherited into the derived class. We can add our own data and member functions and thus extend the functionality of the base class. Inheritance, when used to modify and extend the capabilities of the existing classes, becomes a very powerful tool for incremental program development.

11 Files

In the previous sections we have been using the *iostream* standard library, which provides *cin* and *cout* methods for reading from standard input and writing to standard output respectively. Now we will see how to read from and write to a file. This requires another standard C++ library called file stream, which defines three new data types namely *ofstream*, *ifstream* and *fstream*.

ofstream represents the output file stream and is used to create files and to write information to files.

ifstream represents the input file stream and is used to read information from files.

fstream represents the file stream generally, and has the capabilities of both *ofstream* and *ifstream* which means it can create files, write information to files, and read information from files.

The programming that involves file operations for input/output, in general has the followig three steps.

Open the file: Opening an input file allows the program to make available the file for processing by creating connection between the file and the program. Opening an output file usually creates a file on the disk if it already doesn't exist. In C++, we can use *fstream* object to open a file for writing or reading purpose as shown below.

```
fstream fileobj;

fileobj.open("filename.txt", ios::openMode);
```

Note that, *fstream* being a class, we created an object named as *fileobj* and invoked *open()* function that having two parameters such as *filename* and *mode* of access namely *in*/*out*/*app*/*ate*/*trunc* etc. The default value of access mode in *fstream*, *ifstream*, and *ofstream* are *in*/*out*, *in* and *out* respectively.

Process the file: The program either *write* to the file or *read* from the file or can do both depending upon the requirement.

For reading from a file that already exist, we must open the file in a read access mode using *ios::in*. And by invoking *get()* or *getline()* function of *fstream* class on object, we can read from the existing file.

Similarly for writing to a file, we must open the file in a write or append access mode using *ios::out* or *ios::app* and by invoking *put()* function.

Close the file: To remove the connection between the file and program we must close the file.

Even though, C++ implicitely flushes all the streams, release all the allocated memory and close all the opened files, the programmer can also explicitely close all the opened files before the program termination using *close()* function invoked on file object. For example, *fileobj.close()*;.

Another operation that can be done on a file is deletion of it. Using *remove()* function we can delete a file. The basic syntax for deleting file is *remove("fileName")*;.

11.1 File Access Method

There are mainly two ways to access data stored in a file programmatically.

1. Sequnential Access

A Sequential file has to be accessed in the same order the file was written. That is, in sequential access, we access the file data from the beginning of the file to the end of the file. If we need a piece of data that is stored near the end of the file then we have to go traverse all the data before it. It is just like a cassette tape player. We play music in the same order it was recorded. All the above file oerations are applicable to sequential method.

2. Random Access

Random access method is also known as Direct access method. We can directly jump to the desired data without reading the data that comes before it. This method is associated with a file pointer, which can be moved directly to any location in the file. Random access is crucial in certain applications such as databases and indexes. In C++ we can position the input pointer using *seekg()* and output pointer using *seekp()*. The *seekg()* and *seekp()* functions take two parameters. The first parameter is an offset that determines how many bytes to move the file pointer. The second parameter is an *ios flag* that specifies what the offset parameter should be offset from.

The flag *beg* indicates the offset is relative to the beginning of the file and is by default. The flag *cur* indicates the offset is relative to the current location of the file pointer. The flag *end* indicates the offset is relative to the end of the file.

For example;

```
file_obj.seekg(10, ios::cur);  moves forward 10 bytes.
file_obj.seekg(-10, ios::cur); moves backwards 10 bytes.
file_obj.seekg(20, ios::beg);  moves to 2oth byte input file.
file_obj.seekg(20); moves to 20th byte input file.
file_obj.seekg(-20, ios::end); moves to the 2oth byte before end of the file.
```

Following are examples of sequential file operations.

11.2 Illustration

Program 1.1: Program to illustrate how to open a file in read mode and close it.

```
#include<iostream>
#include<fstream>
using namespace std;

int main()
{
        fstream fobj;
        cout<<"File open in read mode:"<<endl;
        fobj.open("file1.txt",ios::in);
        fobj.close();
}
```

Program 1.2: Program to check a file is opened or closed.

```
#include<iostream>
#include<fstream>
using namespace std;

void chkOpen(fstream&);
void chkClosed(fstream&);

void chkOpen(fstream &fobj)
{
        if(!fobj)
        {
                cout<<"Error while Opening File: "<<endl;
        }
        else
        {
                cout<<"File Open Successfully"<<endl;
        }
}
```

(continues on next page)

(continued from previous page)

```
void chkClosed(fstream& fobj)
{
        if(fobj.is_open()==0)
        {
                cout<<"File is Closed" <<endl;
        }
        else
        {
                cout<<"File is still Open: "<<endl;
        }
}

int main()
{
        fstream fobj;
        cout<<"In write mode:"<<endl;
        fobj.open("file1.txt", ios::out);
        chkOpen(fobj);
        chkClosed(fobj);
        return 0;
}
```

Program 1.3: Program to open a file in read mode and display the contents.

```
#include<iostream>
#include<fstream>
using namespace std;

void readData(fstream&);

void readData(fstream& fobj)
{
        char ch;
        if(!fobj)
        {
                cout<<"Error while opening File: "<<endl;
        }
        else
        {
                cout<<"File opened Successfully: "<<endl;
                while(!fobj.eof())
                {
                        fobj.get(ch);
                        cout<<ch;
                }
```

(continues on next page)

(continued from previous page)

```
        }
}

int main()
{
        fstream fobj;
        cout<<"WRITING PURPOSE:"<<endl;
        fobj.open("file1.txt",ios::in);
        readData(fobj);
        fobj.close();
}
```

Program 1.4: Program to read from a file and write to another.

```
#include<iostream>
#include<fstream>
using namespace std;

void writeData(fstream&,fstream&);

void write(fstream& inobj, fstream& outobj)
{
        char ch;
        if(!inobj)
        {
                cout<<"Error while opening File: "<<endl;
        }
        if(!inobj)
        {
                cout<<"Error while opening File: "<<endl;
        }
        else
        {
                cout<<"File opened Successfully: "<<endl;
                while(!inobj.eof())
                {
                        inobj.get(ch);
                        outobj.put(ch);
                }
        }
}

int main()
{
        fstream inobj,outobj;
```

(continues on next page)

(continued from previous page)

```
        cout<<"File is opening for writing:"<<endl;
        inobj.open("file1.txt",ios::in);
        outobj.open("file2.txt",ios::out);
        writeData(inobj,outobj);
        inobj.close();
        outobj.close();
}
```

Following program illustrate random accessing of files.

Program 1.5: Program to open a file in read and display the contents in random access method.

```
#include<iostream>
#include<fstream>
using  namespace std;

int main()
{
        fstream fobj;
        fobj.open("file1.txt",ios::in);
        if (!fobj)
        {
                cout<< "File can not be opened for reading!\n";
        }
        string strData;
        fobj.seekg(5);
        getline(fobj, strData);
        cout<<strData<<'\n';
        fobj.seekg(10, ios::cur);getline(fobj, strData);
        cout<<strData<<'\n';
        fobj.seekg(-20, ios::end);
        getline(fobj, strData);
        cout<<strData<<'\n';
        fobj.close();
        return 0;
}
```

Welcome to C++ Programming!

12 Numerical Methods

12.1 Successive Approximation Method for solving equations

For solving the equation $f(x) = 0$, we have an iterative method named as Successive Approximation method. To apply the method of Successive Approximation for solving the equation $f(x) = 0$, we have to rearrange $f(x)$ as of the form $x = \phi(x)$ and have to guess an initial solution, say, x_0. Correspondingly, by taking this initial guess x_0 we will find $x_1, x_2, x_3, ...,$. till we get desired accuracy as given below.

$$x_1 = x_0 \times \phi(x_0)$$
$$x_2 = x_1 \times \phi(x_1)$$
$$x_3 = x_2 \times \phi(x_2)$$
$$.$$
$$.$$
$$.$$

The above iterative procedure can be repeated in particular, the desired accuracy ($= \epsilon$, say) of $|x_{i+1} - x_i| < \epsilon$, for $i = 0, 1, 2, 3...$ is reached.

12.1.1 Illustrative Examples:

Example 1.1

Find a solution of $f(x) = cos(x) - 3x + 1 = 0$, using Successive Approximation method with an error of $\epsilon < 0.0001$.

Solution

We have $f(x) = cos(x) - 3x + 1 = 0$ and error anticipated is $(\epsilon) < 0.0001$.

To apply the method of Successive Approximation we have to rearrange $f(x) = 0$ as of the form $x = \phi(x)$ and we have to guess an initial solution, say, x_0.

So, we have

$$\begin{aligned} f(x) &= cos(x) - 3x + 1 = 0 \\ 3x &= cos(x) + 1 \\ i.e., \qquad x &= \frac{1 + cos(x)}{3} = \phi(x). \end{aligned}$$

And by taking initial guess as $x_0 = 0.5$ we will find x_1.

With this x_1 as new x_0 we will find x_2 and proceed so, till we get a solution of desired accuracy.

Hence the detailed iteration is given below

$$\begin{aligned} x_1 &= \phi(x_0) \\ &= \frac{1 + cos(x_0)}{3} \\ &= \frac{1 + cos(0.5000)}{3} \\ i.e., \qquad x_1 &= 0.6259. \end{aligned}$$

Error $= |x_1 - x_0| = 0.125861$.

$$\begin{aligned} x_2 &= \phi(x_0) \\ &= \frac{1 + cos(x_0)}{3} \\ &= \frac{1 + cos(0.6259)}{3} \\ i.e., \qquad x_2 &= 0.6035. \end{aligned}$$

Error $= |x_2 - x_1| = 0.022374$.

$$\begin{aligned} x_3 &= \phi(x_0) \\ &= \frac{1 + cos(x_0)}{3} \\ &= \frac{1 + cos(0.6035)}{3} \\ i.e., \qquad x_3 &= 0.6078. \end{aligned}$$

Error $= |x_3 - x_2| = 0.004301$.

$$\begin{aligned} x_4 &= \phi(x_0) \\ &= \frac{1 + cos(x_0)}{3} \\ &= \frac{1 + cos(0.6078)}{3} \\ i.e., \qquad x_4 &= 0.6070. \end{aligned}$$

Error $= |x_4 - x_3| = 0.000816$.

$$\begin{aligned} x_5 &= \phi(x_0) \\ &= \frac{1 + cos(x_0)}{3} \\ &= \frac{1 + cos(0.6070)}{3} \\ i.e., \qquad x_5 &= 0.6071. \end{aligned}$$

Error $= |x_5 - x_4| = 0.000155$.

$\therefore$ the solution is $x = 0.607126$ with an error of 0.000155.

Example 1.2

Find a solution of $f(x) = 2x - log(x) - 7 = 0$, using Successive Approximation method with an error of $\epsilon < 0.0001$.

Solution

We have $f(x) = 2x - log(x) - 7 = 0$ and error anticipated is $(\epsilon) < 0.0001$.

To apply the method of Successive Approximation we have to rearrange $f(x) = 0$ as of the form $x = \phi(x)$ and we have to guess an initial solution, say, x_0.

So, we have

$$\begin{aligned} f(x) &= 2x - log_{10}(x) - 7 = 0 \\ 2x &= log_{10}(x) + 7 \\ i.e., \qquad x &= \frac{7 + log_{10}(x)}{2} = \phi(x). \end{aligned}$$

And by taking initial guess as $x_0 = 3$ we will find x_1.

With this x_1 as new x_0 we will find x_2 and proceed so, till we get a solution of desired accuracy.

Hence the detailed iteration is given below

$$\begin{aligned} x_1 &= \phi(x_0) \\ &= \frac{7 + log_{10}(x_0)}{2} \\ &= \frac{7 + log_{10}(3.0000)}{2} \\ i.e., \qquad x_1 &= 3.7386. \end{aligned}$$

Error $= |x_1 - x_0| = 0.738561$.

$$\begin{aligned} x_2 &= \phi(x_0) \\ &= \frac{7 + log_{10}(x_0)}{2} \\ &= \frac{7 + log_{10}(3.7386)}{2} \\ i.e., \qquad x_2 &= 3.7864. \end{aligned}$$

Error $= |x_2 - x_1| = 0.047792$.

$$\begin{aligned} x_3 &= \phi(x_0) \\ &= \frac{7 + log_{10}(x_0)}{2} \\ &= \frac{7 + log_{10}(3.7864)}{2} \\ i.e., \qquad x_3 &= 3.7891. \end{aligned}$$

Error $= |x_3 - x_2| = 0.002758$.

$$\begin{aligned} x_4 &= \phi(x_0) \\ &= \frac{7 + log_{10}(x_0)}{2} \\ &= \frac{7 + log_{10}(3.7891)}{2} \\ i.e., \qquad x_4 &= 3.7893. \end{aligned}$$

Error $= |x_4 - x_3| = 0.000158$.

$\therefore$ the solution is $x = 3.789269$ with an error of 0.000158.

No of iteration is 4.

Example 1.3

Find a solution of $f(x) = 2x - 2sin(x) - 1 = 0$, using Successive Approximation method with an error of $\epsilon < 0.0001$.

Solution

We have $f(x) = 2x - 2sin(x) - 1 = 0$ and error anticipated is $(\epsilon) < 0.0001$.

To apply the method of Successive Approximation we have to rearrange $f(x) = 0$ as of the form $x = \phi(x)$ and we have to guess an initial solution, say, x_0.

So, we have

$$\begin{aligned} f(x) &= 2x - 2sin(x) - 1 = 0 \\ 2x &= 2sin(x) + 1 \\ i.e., \qquad x &= \frac{1 + 2sin(x)}{2} = \phi(x). \end{aligned}$$

And by taking initial guess as $x_0 = 3$ we will find x_1.

With this x_1 as new x_0 we will find x_2 and proceed so, till we get a solution of desired accuracy.

Hence the detailed iteration is given below

$$\begin{aligned} x_1 &= \phi(x_0) \\ &= \frac{1 + 2sin(x_0)}{2} \\ &= \frac{1 + 2 * sin(3.0000)}{2} \\ i.e., \qquad x_1 &= 0.6411. \end{aligned}$$

Error $= |x_1 - x_0| = 2.358880$.

$$\begin{aligned} x_2 &= \phi(x_0) \\ &= \frac{1 + 2sin(x_0)}{2} \\ &= \frac{1 + 2 * sin(0.6411)}{2} \\ i.e., \qquad x_2 &= 1.0981. \end{aligned}$$

Error $= |x_2 - x_1| = 0.456973.$

$$\begin{aligned} x_3 &= \phi(x_0) \\ &= \frac{1 + 2sin(x_0)}{2} \\ &= \frac{1 + 2 * sin(1.0981)}{2} \\ i.e., \qquad x_3 &= 1.3903. \end{aligned}$$

Error $= |x_3 - x_2| = 0.292248.$

$$\begin{aligned} x_4 &= \phi(x_0) \\ &= \frac{1 + 2sin(x_0)}{2} \\ &= \frac{1 + 2 * sin(1.3903)}{2} \\ i.e., \qquad x_4 &= 1.4838. \end{aligned}$$

Error $= |x_4 - x_3| = 0.093421.$

$$\begin{aligned} x_5 &= \phi(x_0) \\ &= \frac{1 + 2sin(x_0)}{2} \\ &= \frac{1 + 2 * sin(1.4838)}{2} \\ i.e., \qquad x_5 &= 1.4962. \end{aligned}$$

Error $= |x_5 - x_4| = 0.012453.$

$$\begin{aligned} x_6 &= \phi(x_0) \\ &= \frac{1 + 2sin(x_0)}{2} \\ &= \frac{1 + 2 * sin(1.4962)}{2} \\ i.e., \qquad x_6 &= 1.4972. \end{aligned}$$

Error $= |x_6 - x_5| = 0.001005.$

$\therefore$ the solution is $x = 1.497220$ with an error of 0.001005.

No of iteration is = 6.

Example 1.4

Find a solution of $f(x) = cos(x) - xe^x = 0$, using Successive Approximation method with an error of $\epsilon < 0.01$.

Solution

We have $f(x) = cos(x) - xe^x = 0$ and error anticipated is $(\epsilon) < 0.01$.

To apply the method of Successive Approximation we have to rearrange $f(x) = 0$ as of the form $x = \phi(x)$ and we have to guess an initial solution, say, x_0.

So, we have

$$\begin{aligned} f(x) = cos(x) - xe^x &= 0 \\ xe^x &= cos(x) \\ x &= \frac{cos(x)}{e^x}. \end{aligned}$$

And by taking initial guess as $x_0 = 0.5$ we will find x_1.

With this x_1 as new x_0 we will find x_2 and proceed so, till we get a solution of desired accuracy.

Hence the detailed iteration is given below

$$\begin{aligned} x_1 &= \phi(x_0) \\ &= \frac{cos(x_0)}{e^{x_0}} \\ &= \frac{cos(0.5000)}{e^{0.5000}} \\ i.e., \qquad x_1 &= 0.5323. \end{aligned}$$

Error $= |x_1 - x_0| = 0.032281$.

$$\begin{aligned} x_2 &= \phi(x_0) \\ &= \frac{cos(x_0)}{e^{x_0}} \\ &= \frac{cos(0.5323)}{e^{0.5323}} \\ i.e., \qquad x_2 &= 0.5060. \end{aligned}$$

Error $= |x_2 - x_1| = 0.026264.$

$$\begin{aligned} x_3 &= \phi(x_0) \\ &= \frac{cos(x_0)}{e^{x_0}} \\ &= \frac{cos(0.5060)}{e^{0.5060}} \\ i.e., \qquad x_3 &= 0.5273. \end{aligned}$$

Error $= |x_3 - x_2| = 0.021322.$

$$\begin{aligned} x_4 &= \phi(x_0) \\ &= \frac{cos(x_0)}{e^{x_0}} \\ &= \frac{cos(0.5273)}{e^{0.5273}} \\ i.e., \qquad x_4 &= 0.5100. \end{aligned}$$

Error $= |x_4 - x_3| = 0.017341.$

$$\begin{aligned} x_5 &= \phi(x_0) \\ &= \frac{cos(x_0)}{e^{x_0}} \\ &= \frac{cos(0.5100)}{e^{0.5100}} \\ i.e., \qquad x_5 &= 0.5241. \end{aligned}$$

Error $= |x_5 - x_4| = 0.014083.$

$$\begin{aligned} x_6 &= \phi(x_0) \\ &= \frac{cos(x_0)}{e^{x_0}} \\ &= \frac{cos(0.5241)}{e^{0.5241}} \\ i.e., \qquad x_6 &= 0.5126. \end{aligned}$$

Error $= |x_6 - x_5| = 0.011451.$

$\therefore$ the solution is $x = 0.512630$ with an error of 0.011451.

12.2 Regula Falsi method or method of false position for solving equations

For solving the equation $f(x) = 0$, we can use Regula Falsi method also. In this method we will choose two points a and b suct that $f(a)$ and $f(b)$ are of opposite signs. Thus $f(x)$ will cross x-axis between these points showing that there is a root lies between $f(a)$ and $f(b)$. Concequently $f(a) \times f(b) < 0$.

The Regula Falsi iterative method is detailed below.

The point x_1 that lying between $f(a)$ and $f(b)$ such that $f(x_1) \approx 0$ is computed using;

$$x_1 = \frac{a * f(b) - b * f(a)}{f(b) - f(a)}.$$

From this point x_1, we will choose the interval (a, x_1) or (x_1, b) according as $f(a) \times f(x_1) < 0$ or $f(x_1) \times f(b) < 0$ and try to find x_2 using

$$x_2 = \frac{a * f(b) - b * f(a)}{f(b) - f(a)}, \quad \text{with} \quad a = a \quad \text{and} \quad b = x_1, \quad \text{if} \quad f(a) \times f(x_1) < 0.$$

(Or)

$$x_2 = \frac{a * f(b) - b * f(a)}{f(b) - f(a)}, \quad \text{with} \quad a = x_1 \quad \text{and} \quad b = b, \quad \text{if} \quad f(x_1) \times f(b) < 0.$$

We will repeat the above iterative procedure that what we did with x_1, with the point x_2 till we get the desired accuracy ($= \epsilon$, say) of $|x_{i+1} - x_i| < \epsilon$, for $i = 1, 2, 3....$

12.2.1 Illustrative Examples:

Example 1.1

Find a solution of $f(x) = x^3 - 18 = 0$ using Regula Falsi method with an error of $\epsilon < 0.0001$.

Solution

We have $f(x) = x^3 - 18 = 0$ and error anticipated is $(\epsilon) < 0.0001$.

We will find an interval (a, b) in which $f(a) \times f(b) < 0$.

(a, b)	$(f(a)), (fb))$	$f(a) * f(b)$
$(0, 1)$	$(18.0000, 17.0000)$	$306 > 0$
$(1, 2)$	$(17.0000, 10.0000)$	$170 > 0$
$(2, 3)$	$(10.0000, -9.0000)$	$-90 < 0$

i.e $f(2) \times f(3) = -90 < 0$.

Hence there is root between 2 and 3. Now start with $a = 2$ and $b = 3$ we have,

$$\begin{aligned} x_1 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{-18.0000}{-9.0000 - 10.0000} \\ &= \frac{-48.0000}{-19.0000} \\ \therefore \quad x_1 &= 2.5263. \end{aligned}$$

Using the solution x_{ 1 } = 2.5263 and b = 3 we can see that $f(x_1) \times f(b) = -16.8873 < 0$.

So the root is between 2.5263 and 3. And by taking $a = x_1 = 2.5263$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_2 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_1 * f(b) - b * f(x_1)}{b - f(x_1)} \\ &= \frac{-28.3659}{-10.8764} \\ \therefore \quad x_2 &= 2.6080. \end{aligned}$$

Error $= |x_2 - x_1| = 0.081719$.

Using the solution x_{ 2 } = 2.6080 and b = 3 we can see that $f(x_2) \times f(b) = -2.3450 < 0$.

So the root is between 2.6080 and 3. And by taking $a = x_2 = 2.6080$ and $b = 3$ it can be

computed as given below

$$\begin{aligned} x_3 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_2 * f(b) - b * f(x_2)}{b - f(x_2)} \\ &= \frac{-24.2540}{-9.2606} \\ \therefore \quad x_3 &= 2.6191. \end{aligned}$$

Error = $|x_3 - x_2| = 0.011028$.

Using the solution x_{ 3 } = 2.6191 and b = 3 we can see that $f(x_3) \times f(b)$ = -0.3111 < 0.

So the root is between 2.6191 and 3. And by taking $a = x_3 = 2.6191$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_4 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_3 * f(b) - b * f(x_3)}{b - f(x_3)} \\ &= \frac{-23.6753}{-9.0346} \\ \therefore \quad x_4 &= 2.6205. \end{aligned}$$

Error = $|x_4 - x_3| = 0.001457$.

Using the solution x_{ 4 } = 2.6205 and b = 3 we can see that $f(x_4) \times f(b)$ = -0.0410 < 0.

So the root is between 2.6205 and 3. And by taking $a = x_4 = 2.6205$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_5 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_4 * f(b) - b * f(x_4)}{b - f(x_4)} \\ &= \frac{-23.5984}{-9.0046} \\ \therefore \quad x_5 &= 2.6207. \end{aligned}$$

Error = $|x_5 - x_4| = 0.000192$.

Using the solution x_{ 5 } = 2.6207 and b = 3 we can see that $f(x_5) \times f(b)$ = -0.0054 < 0.

So the root is between 2.6207 and 3. And by taking $a = x_5 = 2.6207$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_6 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_5 * f(b) - b * f(x_5)}{b - f(x_5)} \\ &= \frac{-23.5882}{-9.0006} \\ \therefore \quad x_6 &= 2.6207. \end{aligned}$$

Error $= |x_6 - x_5| = 0.000025$.

$\therefore$ the solution is x = 2.620738 with an error of 0.000025.

Example 1.2

Find a solution of $f(x) = x^3 - 2 * x - 5 = 0$, using Regula falsi method with an error of $\epsilon < 0.0001$.

Solution

We have $f(x) = x^3 - 2 * x - 5 = 0$ and error anticipated is $(\epsilon) < 0.0001$.

We will find an interval (a, b) in which $f(a) \times f(b) < 0$.

(a, b)	$(f(a)), (fb))$	$f(a) * f(b)$
$(0, 1)$	$(-5.0000, -6.0000)$	$30 > 0$
$(1, 2)$	$(-6.0000, -1.0000)$	$6 > 0$
$(2, 3)$	$(-1.0000, 16.0000)$	$-16 < 0$

i.e $f(2) \times f(3) = -16 < 0$.

Hence there is root between 2 and 3. Now start with $a = 2$ and $b = 3$ we have,

$$\begin{aligned} x_1 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{32.0000}{16.0000 - -1.0000} \\ &= \frac{35.0000}{17.0000} \\ \therefore \quad x_1 &= 2.0588 \end{aligned}$$

Using the solution x_{ 1 } = 2.0588 and b = 3 we can see that $f(x_1) \times f(b) = -6.2528 < 0$.

So the root is between 2.0588 and 3. And by taking $a = x_1 = 2.0588$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_2 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_1 * f(b) - b * f(x_1)}{b - f(x_1)} \\ &= \frac{34.1136}{16.3908} \\ \therefore \quad x_2 &= 2.0813. \end{aligned}$$

Error = $|x_2 - x_1| = 0.022440$.

Using the solution x_{ 2 } = 2.0813 and b = 3 we can see that $f(x_2) \times f(b)$ = -2.3553 < 0.

So the root is between 2.0813 and 3. And by taking $a = x_2 = 2.0813$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_3 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_2 * f(b) - b * f(x_2)}{b - f(x_2)} \\ &= \frac{33.7418}{16.1472} \\ \therefore \quad x_3 &= 2.0896. \end{aligned}$$

Error = $|x_3 - x_2| = 0.008376$.

Using the solution x_{ 3 } = 2.0896 and b = 3 we can see that $f(x_3) \times f(b)$ = -0.8748 < 0.

So the root is between 2.0896 and 3. And by taking $a = x_3 = 2.0896$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_4 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_3 * f(b) - b * f(x_3)}{b - f(x_3)} \\ &= \frac{33.5983}{16.0547} \\ \therefore \quad x_4 &= 2.0927. \end{aligned}$$

Error = $|x_4 - x_3| = 0.003100$.

Using the solution x_{ 4 } = 2.0927 and b = 3 we can see that $f(x_4) \times f(b)$ = -0.3232 < 0.

So the root is between 2.0927 and 3. And by taking $a = x_4 = 2.0927$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_5 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_4 * f(b) - b * f(x_4)}{b - f(x_4)} \\ &= \frac{33.5444}{16.0202} \\ \therefore \quad x_5 &= 2.0939. \end{aligned}$$

Error = $|x_5 - x_4| = 0.001144$.

Using the solution x_{ 5 } = 2.0939 and b = 3 we can see that $f(x_5) \times f(b)$ = -0.1192 < 0.

So the root is between 2.0939 and 3. And by taking $a = x_5 = 2.0939$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_6 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_5 * f(b) - b * f(x_5)}{b - f(x_5)} \\ &= \frac{33.5245}{16.0075} \\ \therefore \quad x_6 &= 2.0943. \end{aligned}$$

Error = $|x_6 - x_5| = 0.000422$.

Using the solution x_{ 6 } = 2.0943 and b = 3 we can see that $f(x_6) \times f(b)$ = -0.0439 < 0.

So the root is between 2.0943 and 3. And by taking $a = x_6 = 2.0943$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_7 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_6 * f(b) - b * f(x_6)}{b - f(x_6)} \\ &= \frac{33.5171}{16.0027} \\ \therefore \quad x_7 &= 2.0945. \end{aligned}$$

Error = $|x_7 - x_6| = 0.000155$.

Using the solution x_{ 7 } = 2.0945 and b = 3 we can see that $f(x_7) \times f(b)$ = -0.0162 < 0.

So the root is between 2.0945 and 3. And by taking $a = x_7 = 2.0945$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_8 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_7 * f(b) - b * f(x_7)}{b - f(x_7)} \\ &= \frac{33.5144}{16.0010} \\ \therefore \quad x_8 &= 2.0945. \end{aligned}$$

Error $= |x_8 - x_7| = 0.000057$.

$\therefore$ the solution is x = 2.094518 with an error of 0.000057.

Example 1.3

Find a solution of $f(x) = x - cos(x) = 0$, using Regula falsi method with an error of $\epsilon < 0.0001$.

Solution

We have $f(x) = x - cos(x) = 0$ and error anticipated is $(\epsilon) < 0.0001$.

We will find an interval (a, b) in which $f(a) \times f(b) < 0$.

(a, b)	$(f(a)), (fb))$	$f(a) * f(b)$
$(0, 1)$	$(-1.0000, 0.4597)$	$-0.4597 < 0$

i.e $f(0) \times f(1) =$ -0.4597 < 0.

Hence there is root between 0 and 1. Now start with $a = 0$ and $b = 1$ we have,

$$\begin{aligned} x_1 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{0.0000}{0.4597 - -1.0000} \\ &= \frac{1.0000}{1.4597} \\ \therefore \quad x_1 &= 0.6851 \end{aligned}$$

Using the solution x_{ 1 } = 0.6851 and b = 1 we can see that $f(x_1) \times f(b) =$ -0.0411 < 0.

So the root is between 0.6851 and 1. And by taking $a = x_1 = 0.6851$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_2 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_1 * f(b) - b * f(x_1)}{b - f(x_1)} \\ &= \frac{0.4042}{0.5490} \\ \therefore \quad x_2 &= 0.7363. \end{aligned}$$

Error = $|x_2 - x_1| = 0.051226$.

Using the solution x_{ 2 } = 0.7363 and b = 1 we can see that $f(x_2) \times f(b)$ = -0.0021 < 0.

So the root is between 0.7363 and 1. And by taking $a = x_2 = 0.7363$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_3 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_2 * f(b) - b * f(x_2)}{b - f(x_2)} \\ &= \frac{0.3431}{0.4644} \\ \therefore \quad x_3 &= 0.7389. \end{aligned}$$

Error = $|x_3 - x_2| = 0.002646$.

Using the solution x_{ 3 } = 0.7389 and b = 1 we can see that $f(x_3) \times f(b)$ = -0.0001 < 0.

So the root is between 0.7389 and 1. And by taking $a = x_3 = 0.7389$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_4 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_3 * f(b) - b * f(x_3)}{b - f(x_3)} \\ &= \frac{0.3399}{0.4599} \\ \therefore \quad x_4 &= 0.7391. \end{aligned}$$

Error = $|x_4 - x_3| = 0.000133$.

Using the solution x_{ 4 } = 0.7391 and b = 1 we can see that $f(x_4) \times f(b)$ = -0.0000 < 0.

So the root is between 0.7391 and 1. And by taking $a = x_4 = 0.7391$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_5 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_4 * f(b) - b * f(x_4)}{b - f(x_4)} \\ &= \frac{0.3398}{0.4597} \\ \therefore \quad x_5 &= 0.7391. \end{aligned}$$

Error $= |x_5 - x_4| = 0.000007$.

$\therefore$ the solution is x = 0.739085 with an error of 0.000007.

Example 1.4

Find a solution of $f(x) = x \times e^x - 2 = 0$, using Regula falsi method with an error of $\epsilon < 0.0001$.

Solution

We have $f(x) = x \times e^x - 2 = 0$ and error anticipated is $(\epsilon) < 0.0001$.

We will find an interval (a, b) in which $f(a) \times f(b) < 0$.

(a, b)	$(f(a)), (fb))$	$f(a) * f(b)$
$(-1, 0)$	$(-2.3679, -2.0000)$	$4.7358 > 0$
$(0, 1)$	$(-2.0000, 0.7183)$	$-1.4366 < 0$

i.e $f(0) \times f(1)$ = -1.4365636569180902 < 0.

Hence there is root between 0 and 1. Now start with $a = 0$ and $b = 1$ we have,

$$\begin{aligned} x_1 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{0.0000}{0.7183 - -2.0000} \\ &= \frac{2.0000}{2.7183} \\ \therefore \quad x_1 &= 0.7358 \end{aligned}$$

Using the solution x_{ 1 } = 0.7358 and b = 1 we can see that $f(x_1) \times f(b)$ = -0.3336 < 0.

So the root is between 0.7358 and 1. And by taking $a = x_1 = 0.7358$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_2 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_1 * f(b) - b * f(x_1)}{b - f(x_1)} \\ &= \frac{0.9929}{1.1827} \\ \therefore \quad x_2 &= 0.8395. \end{aligned}$$

Error = $|x_2 - x_1| = 0.103762$.

Using the solution x_{ 2 } = 0.8395 and b = 1 we can see that $f(x_2) \times f(b)$ = -0.0404 < 0.

So the root is between 0.8395 and 1. And by taking $a = x_2 = 0.8395$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_3 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_2 * f(b) - b * f(x_2)}{b - f(x_2)} \\ &= \frac{0.6593}{0.7746} \\ \therefore \quad x_3 &= 0.8512. \end{aligned}$$

Error = $|x_3 - x_2| = 0.011663$.

Using the solution x_{ 3 } = 0.8512 and b = 1 we can see that $f(x_3) \times f(b)$ = -0.0044 < 0.

So the root is between 0.8512 and 1. And by taking $a = x_3 = 0.8512$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_4 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_3 * f(b) - b * f(x_3)}{b - f(x_3)} \\ &= \frac{0.6176}{0.7245} \\ \therefore \quad x_4 &= 0.8525. \end{aligned}$$

Error = $|x_4 - x_3| = 0.001268$.

Using the solution x_{ 4 } = 0.8525 and b = 1 we can see that $f(x_4) \times f(b)$ = -0.0005 < 0.

So the root is between 0.8525 and 1. And by taking $a = x_4 = 0.8525$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_5 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_4 * f(b) - b * f(x_4)}{b - f(x_4)} \\ &= \frac{0.6130}{0.7190} \\ \therefore \quad x_5 &= 0.8526. \end{aligned}$$

Error = $|x_5 - x_4| = 0.000137$.

Using the solution x_{ 5 } = 0.8526 and b = 1 we can see that $f(x_5) \times f(b)$ = -0.0001 < 0.

So the root is between 0.8526 and 1. And by taking $a = x_5 = 0.8526$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_6 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_5 * f(b) - b * f(x_5)}{b - f(x_5)} \\ &= \frac{0.6125}{0.7184} \\ \therefore \quad x_6 &= 0.8526. \end{aligned}$$

Error = $|x_6 - x_5| = 0.000015$.

$\therefore$ the solution is x = 0.852604 with an error of 0.000015.

Example 1.5

Find a solution of $f(x) = x \times log_{10}(x) - 1.2 = 0$, using Regula falsi method with an error of $\epsilon < 0.0001$.

Solution

We have $f(x) = x \times log_{10}(x) - 1.2 = 0$ and error anticipated is $(\epsilon) < 0.0001$.

We will find an interval (a, b) in which $f(a) \times f(b) < 0$.

(a, b)	$(f(a)), (fb))$	$f(a) * f(b)$
$(2, 3)$	$(-0.5979, 0.2314)$	$-0.1383 < 0$

i.e $f(2) \times f(3)$ = -0.13834 < 0.

Hence there is root between 2 and 3. Now start with $a = 2$ and $b = 3$ we have,

$$\begin{aligned} x_1 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{0.4627}{0.2314 - -0.5979} \\ &= \frac{2.2565}{0.8293} \\ \therefore \quad x_1 &= 2.7210 \end{aligned}$$

Using the solution x_{ 1 } = 2.7210 and b = 3 we can see that $f(x_1) \times f(b)$ = -0.0040 < 0.

So the root is between 2.7210 and 3. And by taking $a = x_1 = 2.7210$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_2 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_1 * f(b) - b * f(x_1)}{b - f(x_1)} \\ &= \frac{0.6808}{0.2485} \\ \therefore \quad x_2 &= 2.7402. \end{aligned}$$

Error = $|x_2 - x_1| = 0.019191$.

Using the solution x_{ 2 } = 2.7402 and b = 3 we can see that $f(x_2) \times f(b)$ = -0.0001 < 0.

So the root is between 2.7402 and 3. And by taking $a = x_2 = 2.7402$ and $b = 3$ it can be computed as given below

$$\begin{aligned} x_3 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_2 * f(b) - b * f(x_2)}{b - f(x_2)} \\ &= \frac{0.6351}{0.2317} \\ \therefore \quad x_3 &= 2.7406. \end{aligned}$$

Error = $|x_3 - x_2| = 0.000431$.

Using the solution x_{ 3 } = 2.7406 and b = 3 we can see that $f(x_3) \times f(b)$ = -0.0000 < 0.

So the root is between 2.7406 and 3. And by taking $a = x_3 = 2.7406$ and $b = 3$ it can be

computed as given below

$$\begin{aligned} x_4 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_3 * f(b) - b * f(x_3)}{b - f(x_3)} \\ &= \frac{0.6341}{0.2314} \\ \therefore \quad x_4 &= 2.7406. \end{aligned}$$

Error = $|x_4 - x_3| = 0.000010$.

therefore hskip 1cm the solution x = 2.740646 with an error of 0.000010.

Example 1.6

Find a solution of $f(x) = x \times tan(x) - 1.28 = 0$, using Regula falsi method with an error of $\epsilon < 0.0001$.

Solution

We have $f(x) = x \times tan(x) - 1.28 = 0$ and error anticipated is $(\epsilon) < 0.0001$.

We will find an interval (a, b) in which $f(a) \times f(b) < 0$.

(a, b)	$(f(a)), (fb))$	$f(a) * f(b)$
$(0, 1)$	$(-1.2800, 0.2774)$	$-0.3551 < 0$

i.e $f(0) \times f(1)$ = -0.3550818875582749 < 0.

Hence there is root between 0 and 1. Now start with $a = 0$ and $b = 1$ we have,

$$\begin{aligned} x_1 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{0.0000}{0.2774 - -1.2800} \\ &= \frac{1.2800}{1.5574} \\ \therefore \quad x_1 &= 0.8219 \end{aligned}$$

Using the solution x_{ 1 } = 0.8219 and b = 1 we can see that $f(x_1) \times f(b)$ = -0.1098 < 0.

So the root is between 0.8219 and 1. And by taking $a = x_1 = 0.8219$ and $b = 1$ it can be

computed as given below

$$\begin{aligned} x_2 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_1 * f(b) - b * f(x_1)}{b - f(x_1)} \\ &= \frac{0.6239}{0.6733} \\ \therefore \quad x_2 &= 0.9266. \end{aligned}$$

Error = $|x_2 - x_1| = 0.104729$.

Using the solution x_{ 2 } = 0.9266 and b = 1 we can see that $f(x_2) \times f(b)$ = -0.0128 < 0.

So the root is between 0.9266 and 1. And by taking $a = x_2 = 0.9266$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_3 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_2 * f(b) - b * f(x_2)}{b - f(x_2)} \\ &= \frac{0.3033}{0.3237} \\ \therefore \quad x_3 &= 0.9371. \end{aligned}$$

Error = $|x_3 - x_2| = 0.010495$.

Using the solution x_{ 3 } = 0.9371 and b = 1 we can see that $f(x_3) \times f(b)$ = -0.0013 < 0.

So the root is between 0.9371 and 1. And by taking $a = x_3 = 0.9371$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_4 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_3 * f(b) - b * f(x_3)}{b - f(x_3)} \\ &= \frac{0.2646}{0.2821} \\ \therefore \quad x_4 &= 0.9381. \end{aligned}$$

Error = $|x_4 - x_3| = 0.001039$.

Using the solution x_{ 4 } = 0.9381 and b = 1 we can see that $f(x_4) \times f(b)$ = -0.0001 < 0.

So the root is between 0.9381 and 1. And by taking $a = x_4 = 0.9381$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_5 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_4 * f(b) - b * f(x_4)}{b - f(x_4)} \\ &= \frac{0.2607}{0.2779} \\ \therefore \quad x_5 &= 0.9382. \end{aligned}$$

Error $= |x_5 - x_4| = 0.000103$.

Using the solution x_{ 5 } = 0.9382 and b = 1 we can see that $f(x_5) \times f(b)$ = -0.0000 < 0.

So the root is between 0.9382 and 1. And by taking $a = x_5 = 0.9382$ and $b = 1$ it can be computed as given below

$$\begin{aligned} x_6 &= \frac{a * f(b) - b * f(a)}{f(b) - f(a)} \\ &= \frac{x_5 * f(b) - b * f(x_5)}{b - f(x_5)} \\ &= \frac{0.2603}{0.2775} \\ \therefore \quad x_6 &= 0.9383. \end{aligned}$$

Error $= |x_6 - x_5| = 0.000010$.

therefore hskip 1cm the solution x = 0.938255 with an error of 0.000010.

12.2.2 Practice Problems

Find a solution for the following problems using the Regula Falsi Method with error < 0.0001.

1. $f(x) = 2x^3 - 3x - 6 = 0.$
2. $f(x) = x^3 - x - 1 = 0.$
3. $f(x) = x^3 + x + 1 = 0.$
4. $f(x) = x^2 - 2 = 0.$
5. $f(x) = xe^x - 2 = 0.$
6. $f(x) = x^2 - 2x + 1 = 0.$
7. $f(x) = x^3 - x^2 - x + 1 = 0.$
8. $f(x) = xsin(x) + cos(x) = 0.$
9. $f(x) = xe^x - cos(x) = 0.$
10. $f(x) = xtan(x) - 1.28 = 0.$

12.3 Finding roots of equations by Newton-Raphson method

The Newton-Raphson method, or Newton Method, is a powerful technique for solving equations numerically. Newton Method is based on the idea of linear approximation. The Newton Method, properly used, usually gives a root with devastating efficiency.

Let $f(x)$ be a well-defined function, and let r be a root of the equation $f(x) = 0$. We start with an estimate x_0 of r. From x_0, we estimate x_1. From x_1, we produce a new estimate x_2. From x_2, we try to get a new estimate x_3. We proceed this process till we get a solution of desired accuracy.

The Newton-Raphson iterative formula is given by

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

with the initial estimate or guess x_0, which is close to the solution r.

12.3.1 Illustrative examples:

Example 5.1.

Find a non-zero solution of $x = 2sin(x)$. using the Newton Method.

This equationa can also be written as $x - 2sin(x) = 0$.

Now let $f(x) = x - 2sin(x)$. Then $f'(x) = 1 - 2cos(x)$, and the Newton-Raphson iteration is

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} && (12.1)\\ &= x_n - \frac{x_n - 2sin(x_n)}{1 - 2cos(x_n)} && (12.2)\\ &= \frac{2[sin(x_n) - x_n cos(x_n)]}{1 - 2cos(x_n)} && (12.3)\\ & && (12.4) \end{aligned}$$

Now put $n = 0$ and let $x_0 = 1.5$.

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} && (12.5)\\ &= \frac{2[sin(x_0) - x_0 cos(x_0)]}{1 - 2cos(x_0)} && (12.6)\\ &= \frac{2[sin(1.5) - 1.5cos(1.5)]}{1 - 2cos(1.5)} && (12.7)\\ &= \frac{2[0.9975 - 1.5 \times 0.0707]}{1 - 2 \times 0.0707} && (12.8)\\ i.e., x_1 &= 2.07655; && (12.9)\\ x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} && (12.10)\\ &= \frac{2[sin(x_1) - x_1 cos(x_1)]}{1 - 2cos(x_1)} && (12.11)\\ &= \frac{2[sin(2.07655) - 2.07655cos(2.07655)]}{1 - 2cos(2.07655)} && (12.12)\\ &= \frac{2[0.8748 - 1.5 \times (-0.4844)]}{1 - 2 \times (-0.4844)} && (12.13)\\ i.e., x_2 &= 1.91063; && (12.14)\\ x_3 &= x_2 - \frac{f(x_2)}{f'(x_2)} && (12.15)\\ &= \frac{2[sin(1.91063) - 1.91063cos(1.91063)]}{1 - 2cos(1.91063)} && (12.16)\\ &= \frac{2[0.9428 - 1.91063 \times (-0.3333)]}{1 - 2 \times (-0.3333)} && (12.17)\\ i.e., x_3 &= 1.8955; && (12.18)\\ & && (12.19) \end{aligned}$$

Proceeding like this, we will get the values for x_4, x_5, x_6 as given below.

$$x_4 = 1.895494; \tag{12.20}$$
$$x_5 = 1.895494; \tag{12.21}$$
$$x_6 = 1.895494; \tag{12.22}$$
$$\tag{12.23}$$

Here we can see that iterates agree for six places from x_4 to x_6. So, we will stop the iteration and the solution is given by $x = 1.895494$.

Example 5.2.

Find a non-zero solution of $x^3 - 6x + 4 = 0$, using the Newton-Raphson Method.

Now let $f(x) = x^3 - 6x + 4 = 0$. Then $f'(x) = 3x^2 - 6$, and the Newton-Raphson iteration is

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{12.24}$$
$$= x_n - \frac{x_n^3 - 6x_n + 4}{3x_n^2 - 6} \tag{12.25}$$
$$= \frac{2[x_n^3 - 2]}{3[x_n^2 - 2]} \tag{12.26}$$
$$\text{Now put} \quad n = 0 \ \text{ and } \ x_0 = 1 \tag{12.27}$$
$$\tag{12.28}$$

$$
\begin{aligned}
x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} && (12.29)\\
&= \frac{2[x_0^3 - 2]}{3[x_0^2 - 2]} && (12.30)\\
&= \frac{2[1 - 2]}{3[1 - 2]} && (12.31)\\
i.e., x_1 &= 0.66666; && (12.32)\\
x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} && (12.33)\\
&= \frac{2[x_1^3 - 2]}{3[x_1^2 - 2]} && (12.34)\\
&= \frac{2[0.66666^3 - 2]}{3[0.66666^2 - 2]} && (12.35)\\
i.e., x_2 &= 0.7301587; && (12.36)\\
x_3 &= x_2 - \frac{f(x_2)}{f'(x_2)} && (12.37)\\
&= \frac{2[x_2^3 - 2]}{3[x_2^2 - 2]} && (12.38)\\
&= \frac{2[0.7301587^3 - 2]}{3[0.7301587^2 - 2]} && (12.39)\\
i.e., x_3 &= 0.7320490; && (12.40)\\
x_4 &= x_3 - \frac{f(x_3)}{f'(x_3)} && (12.41)\\
&= \frac{2[x_3^3 - 2]}{3[x_3^2 - 2]} && (12.42)\\
&= \frac{2[0.7320490^3 - 2]}{3[0.7320490^2 - 2]} && (12.43)\\
i.e., x_4 &= 0.7320508; && (12.44)\\
& && (12.45)
\end{aligned}
$$

Proceeding like this, we will get the values for $x_5 = 0.7320508$.

Here we can see that iterates agree for seven places in x_4 and x_5. So, we will stop the iteration and the solution is given by $x = 0.7320508$.

Example 5.3.

Find a non-zero solution of $x - cos(x) = 0$, using the Newton-Raphson Method.

Now let $f(x) = x - cos(x) = 0$. Then $f'(x) = 1 + sin(x)$, and the Newton-Raphson iteration

is

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{12.46}$$

$$= x_n - \frac{x_n - cos(x_n)}{1 + sin(x_n)} \tag{12.47}$$

$$= \frac{x_n sin(x_n) + cos(x_n)}{1 + sin(x_n)} \tag{12.48}$$

$$\text{Now put} \quad n = 0 \ \text{ and } \ x_0 = 0.7 \tag{12.49}$$

$$\tag{12.50}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{12.51}$$

$$= \frac{x_0 sin(x_0) + cos(x_0)}{1 + sin(x_0)} \tag{12.52}$$

$$= \frac{0.7 \times sin(0.7) + cos(0.7)}{1 + sin(0.7)} \tag{12.53}$$

$$= \frac{0.7 \times 0.644217 + 0.764842}{1 + 0.644217} \tag{12.54}$$

$$= \frac{1.215794}{1.644217} \tag{12.55}$$

$$i.e., x_1 = 0.739436; \tag{12.56}$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \tag{12.57}$$

$$= \frac{x_1 sin(x_1) + cos(x_1)}{1 + sin(x_1)} \tag{12.58}$$

$$= \frac{0.739436 \times sin(0.739436) + cos(0.739436)}{1 + sin(0.734936)} \tag{12.59}$$

$$= \frac{0.739436 \times 0.673871 + 0.738848}{1 + 0.673871} \tag{12.60}$$

$$= \frac{1.237133}{1.673871} \tag{12.61}$$

$$i.e., x_2 = 0.739085; \tag{12.62}$$

$$x_3 = \frac{x_2 sin(x_2) + cos(x_2)}{1 + sin(x_2)} \tag{12.63}$$

$$= \frac{0.739085 \times sin(0.739085) + cos(0.739085)}{1 + sin(0.739085)} \tag{12.64}$$

$$= \frac{0.739085 \times 0.673612 + 0.739085}{1 + 0.673612} \tag{12.65}$$

$$= \frac{1.236941}{1.673612} \tag{12.66}$$

$$i.e., x_3 = 0.739085; \tag{12.67}$$

$$\tag{12.68}$$

Here we can see that iterates agree for six places in x_2 and x_3. So, we will stop the iteration and the solution is given by $x = 0.739085$.

Example 5.4.

Find a non-zero solution of $4x - e^x = 0$, using the Newton-Raphson Method.

Now let $f(x) = 4x - e^x) = 0$. Then $f'(x) = 4 - e^x$, and the Newton-Raphson iteration is

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{12.69}$$

$$= x_n - \frac{4x_n - e^{x_n}}{4 - e^{x_n}} \tag{12.70}$$

$$= \frac{e^{x_n}(1 - x_n)}{4 - e^{x_n}} \tag{12.71}$$

$$\text{Now put} \quad n = 0 \ \text{and} \ x_0 = 2 \tag{12.72}$$

$$\tag{12.73}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{12.74}$$

$$= x_0 - \frac{4x_0 - e^{x_0}}{4 - e^{x_0}} \tag{12.75}$$

$$= \frac{e^{x_0}(1 - x_0)}{4 - e^{x_0}} \tag{12.76}$$

$$= \frac{e^2(1 - 2)}{4 - e^2} \tag{12.77}$$

$$= \frac{-7.389056}{-3.389056} \tag{12.78}$$

$$i.e., x_1 = 2.180269; \tag{12.79}$$

$$x_2 = \frac{e^{x_1}(1 - x_1)}{4 - e^{x_1}} \tag{12.80}$$

$$= \frac{e^{2.180269}(1 - 2.180269)}{4 - e^{2.180269}} \tag{12.81}$$

$$= \frac{-10.443830}{-4.848686} \tag{12.82}$$

$$i.e., x_2 = 2.153951; \tag{12.83}$$

$$x_3 = \frac{e^{x_2}(1 - x_2)}{4 - e^{x_2}} \tag{12.84}$$

$$= \frac{e^{2.153951}(1 - 2.153951)}{4 - e^{2.153951}} \tag{12.85}$$

$$= \frac{-9.945723}{-4.618844} \tag{12.86}$$

$$i.e., x_3 = 2.153292; \tag{12.87}$$

$$x_4 = \frac{e^{x_3}(1-x_3)}{4-e^{x_3}} \tag{12.88}$$

$$= \frac{e^{2.153292}(1-2.153292)}{4-e^{2.153292}} \tag{12.89}$$

$$= \frac{-9.933495}{-4.613166} \tag{12.90}$$

$$i.e., x_4 = 2.153292; \tag{12.91}$$

Here the iterates agree for six places in x_3 and x_4. So, we will stop the iteration and the solution is given by $x = 2.153292$.

Example 5.5.

Find a non-zero solution of $2x - log_{10}x - 7 = 0$, that lying between 3 and 4 using the Newton-Raphson Method.

Now let $f(x) = 2x - log_{10}x - 7 = 0$. Then $f'(x) = 2 - \frac{log_{10}e}{x}$, and the Newton-Raphson iteration is

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{12.92}$$

$$= x_n - \frac{2x_n - log_{10}x_n - 7}{2 - \frac{log_{10}e}{x_n}} \tag{12.93}$$

$$= \frac{x_n(log_{10}x_n - log_{10}e + 7)}{2x_n - log_{10}e} \tag{12.94}$$

$$\text{Now put} \quad n = 0 \text{ and } x_0 = 4 \tag{12.95}$$

$$\tag{12.96}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{12.97}$$
$$= \frac{x_0(log_{10}x_0 - log_{10}e + 7)}{2x_0 - log_{10}e} \tag{12.98}$$
$$= \frac{4(log_{10}4 - log_{10}e + 7)}{8 - log_{10}e} \tag{12.99}$$
$$= \frac{4(0.602059 - 0.434294 + 7)}{2 \times 4 - 0.434294} \tag{12.100}$$
$$= \frac{28.671062}{7.565705} \tag{12.101}$$
$$i.e., x_1 = 3.789609; \tag{12.102}$$
$$x_2 = \frac{x_1(log_{10}x_1 - log_{10}e + 7)}{2x_1 - log_{10}e} \tag{12.103}$$
$$= \frac{3.789609(log_{10}3.789609 - log_{10}e + 7)}{2 \times 3.789609 - log_{10}e} \tag{12.104}$$
$$= \frac{3.789609(0.578594 - 0.434294 + 7)}{7.579218 - 0.434294} \tag{12.105}$$
$$= \frac{27.074105}{7.144924} \tag{12.106}$$
$$i.e., x_2 = 3.789278; \tag{12.107}$$
$$x_3 = \frac{x_2(log_{10}x_2 - log_{10}e + 7)}{2x_2 - log_{10}e} \tag{12.108}$$
$$= \frac{3.789278(log_{10}3.789278 - log_{10}e + 7)}{2 \times 3.789278 - log_{10}e} \tag{12.109}$$
$$= \frac{3.789278(0.578556 - 0.434294 + 7)}{7.578556 - 0.434294} \tag{12.110}$$
$$= \frac{27.071596}{7.144262} \tag{12.111}$$
$$i.e., x_3 = 3.789278; \tag{12.112}$$
$$\tag{12.113}$$

Here the iterates agree for six places in x_2 and x_3. So, we will stop the iteration and the solution is given by $x = 3.789278$.

12.3.2 Practice Problems

Find a non-zero solution for the following problems using the Newton-Raphson Method.

1. $f(x) = 2x^3 - 3x - 6 = 0;$ with $x_0 = 2$ (12.114)
2. $f(x) = x^3 - x - 1 = 0;$ with $x_0 = 2$ (12.115)
3. $f(x) = x^3 + x + 1 = 0;$ with $x_0 = -1$ (12.116)
4. $f(x) = x^2 - 2 = 0;$ with $x_0 = 1$ (12.117)
5. $f(x) = xe^x - 1 = 0;$ with $x_0 = 0.5$ (12.118)
6. $f(x) = x^2 - 2x + 1 = 0;$ with $x_0 = 0$ (12.119)
7. $f(x) = x^3 - x^2 - x + 1 = 0;$ with $x_0 = 0.8$ (12.120)
8. $f(x) = xsin(x) + cos(x) = 0;$ with $x_0 = 3$ (12.121)
9. $f(x) = xe^x - cos(x) = 0;$ with $x_0 = 0.5$ (12.122)
10. $f(x) = xtan(x)1.28 = 0;$ with $x_0 = 1$ (12.123)

(12.124)

12.4 Interpolation

12.4.1 Newton Forward Interpolation

Interpolation is the technique of estimating the value of a function for any intermediate value of the independent variable, while the process of computing the value of the function outside the given range is called extrapolation. In order to do this we need to introduce the idea of a difference table.

Difference Table

Suppose that we have a set of values of an unknown function $y_0, y_1, y_2, y_3, ..., y_n$, corresponding to a set of values of the (known) independent variable $x_0, x_1, x_2, x_3, ..., x_n$ where $x_0 < x_1 < x_2 < x_3 < ... < x_n$.

Difference Table is given below:

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
x_0	y_0					
		$\Delta y_0 = y_1 - y_0$				
x_1	y_1		$\Delta y_1 - \Delta y_0$			
		$\Delta y_1 = y_2 - y_1$		$\Delta^2 y_1 - \Delta^2 y_0$		
x_2	y_2		$\Delta y_2 - \Delta y_1$		$\Delta^3 y_1 - \Delta^3 y_0$	
		$\Delta y_2 = y_3 - y_2$		$\Delta^2 y_2 - \Delta^2 y_1$		$\Delta^4 y_1 - \Delta^4 y_0$
x_3	y_3		$\Delta y_3 - \Delta y_2$		$\Delta^3 y_2 - \Delta^3 y_1$	
		$\Delta y_3 = y_4 - y_3$		$\Delta^2 y_3 - \Delta^2 y_2$		
x_4	y_4		$\Delta y_4 - \Delta y_3$			
		$\Delta y_4 = y_5 - y_4$				
x_5	y_5					

Consider that, we have a set of values of an unknown function $y_0, y_1, y_2, y_3, ..., y_n$, corresponding to a set of values of the (known) independent variable $x_0, x_1, x_2, x_3, ..., x_n$ where $x_1 = x_0 + h, x_2 = x_0 + 2h, x_3 = x_0 + 3h, ..., x_n = x_0 + nh$ and h is the equal spacing between the values $x_0, x_1, x_2, x_3, ..., x_n$.

Then Newton Forward interpolation polynomial is given by

$$P(x) = y_0 + \frac{(x - x_0)}{h}\frac{\Delta y_0}{1!} + \frac{(x - x_0)(x - x_1)}{h^2}\frac{\Delta^2 y_0}{2!} + \frac{(x - x_0)(x - x_1)(x - x_2)}{h^3}\frac{\Delta^3 y_0}{3!} + ...$$

$$i.e., P(x) = y_0 + u\frac{\Delta y_0}{1!} + u(u-1)\frac{\Delta^2 y_0}{2!} + u(u-1)(u-2)\frac{\Delta^3 y_0}{3!} + ...; \text{ by taking } u = \frac{x - x_0}{h}.$$

12.4.2 Illustrative Example

Example 6.1:

In the following table, use the Newton Forward Interpolation formula to find $f(2.4)$.

Table 1: Data

x	2	4	6	8	10
y	9.68	10.96	12.32	13.76	15.28

Solution:

We will construct the difference table as given below:

x	y	Δy	$\Delta^2 y$
2	9.68		
		$\Delta y_0 = 10.96 - 9.68 = 1.28$	
4	10.96		0.08
		$\Delta y_1 = 12.32 - 10.96 = 1.36$	
6	12.32		0.08
		$\Delta y_2 = 13.76 - 12.32 = 1.44$	
8	13.76		0.08
		$\Delta y_3 = 15.28 - 13.76 = 1.52$	
10	15.28		

Here we have, $x = 2.4, x_0 = 2, h = 2, u = \frac{x-x_0}{h} = u = \frac{2.4-2}{2} = 0.2$.

Then Newton Forward interpolation polynomial can be written as;

$$f(x) = y_0 + \frac{(x-x_0)}{h}\frac{\Delta y_0}{1!} + \frac{(x-x_0)(x-x_1)}{h^2}\frac{\Delta^2 y_0}{2!} + \frac{(x-x_0)(x-x_1)(x-x_2)}{h^3}\frac{\Delta^3 y_0}{3!} + ...$$

$$i.e., f(x) = y_0 + u\frac{\Delta y_0}{1!} + u(u-1)\frac{\Delta^2 y_0}{2!} + u(u-1)(u-2)\frac{\Delta^3 y_0}{3!} + ...;$$

$$f(2.4) = 9.68 + 0.2\frac{1.28}{1!} + 0.2(0.2-1)\frac{0.08}{2!}$$

$$\implies f(2.4) = 9.68 + 0.2 \times 1.28 - 0.2 \times 0.8 \times 0.04$$

$$\therefore f(2.4) = 9.9296.$$

Example 6.2:

In the following table, use the Newton Forward Interpolation formula to find $e^{0.12}$.

Table 2: Data

x	0.1	0.6	1.1	1.6	2.1
e^x	1.1052	1.8221	3.0042	4.9530	8.1662

Solution:

Here we have, $x = 0.12, x_0 = 0.1, h = 0.5, u = \frac{x-x_0}{h} = u = \frac{0.12-0.1}{0.5} = 0.04$.

We will construct the difference table as given below:

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
0.1	1.1052				
		$\Delta y_0 = 1.8221 - 1.1052 = 0.7169$			
0.6	1.8221		0.4652		
		$\Delta y_1 = 3.0042 - 1.8221 = 1.1821$		0.3015	
1.1	3.0042		0.7667		0.1962
		$\Delta y_2 = 4.9530 - 3.0042 = 1.9488$		0.4977	
1.6	4.9530		1.2644		
		$\Delta y_3 = 8.1662 - 4.9530 = 3.2132$			
2.1	8.1662				

Then Newton Forward interpolation polynomial can be written as;

$$f(x) = y_0 + \frac{(x-x_0)}{h}\frac{\Delta y_0}{1!} + \frac{(x-x_0)(x-x_1)}{h^2}\frac{\Delta^2 y_0}{2!} + \frac{(x-x_0)(x-x_1)(x-x_2)}{h^3}\frac{\Delta^3 y_0}{3!} + ...$$

$$i.e., f(x) = y_0 + u\frac{\Delta y_0}{1!} + u(u-1)\frac{\Delta^2 y_0}{2!} + u(u-1)(u-2)\frac{\Delta^3 y_0}{3!} + ...;$$

$$f(0.12) = 1.1052 + 0.04\frac{0.7169}{1!} + 0.04(0.04-1)\frac{0.4652}{2!}$$
$$+ 0.04(0.04-1)(0.04-2)\frac{0.3015}{3!} + 0.04(0.04-1)(0.04-2)(0.04-3)\frac{0.1962}{4!}$$

$$\implies f(0.12) = 1.1052 + 0.04 \times 0.7169 - 0.04 \times 0.96 \times 0.2328$$
$$+ 0.04 \times 0.96 \times 1.96 \times 0.05025 - 0.04 \times 0.96 \times 1.96 \times 2.96 \times 0.008175$$

$$\therefore \quad f(0.12) = 1.1269.$$

12.4.3 Practice Problems

From the following table, use the Newton Forward Interpolation formula to find $f(x)$ indicated therein.

Table 3: Exercise-1

x	2	3	4	5	6
$f(x)$	3.00	4.28	5.88	7.8	10.04

at $x = 0$.

Table 4: Exercise-2

x	16	18	20	22	24
$f(x)$	261.3	293.7	330.0	372.2	422.3

at $x = 16.4$.

Table 5: Exercise-3

x	0.5	0.6	0.7	0.8
$f(x)$	1.127626	1.185465	1.255169	1.337435

at $x = 0.56$.

Table 6: Exercise-4

Year :x	1971	1981	1991	2001	2011
Population (in lacs): f(x)	12	19	31	47	62

in the year 1975.

Table 7: Exercise-5

x	0	1	2	3	4	5
$f(x)$	1	14	15	10	5	6

at $x = 1.5$.

12.4.4 Newton Backward Interpolation

The formula derived in the previous section is appropriate when the required value (x_0, y_0) lies near the beginning of the tabulated data. When (x_0, y_0) is near the end of the table, it is necessary to find a polynomial satisfying the (n+1) values $(x_i, y_i), i = 0, 1, 2, ..., n$.

Difference Table

As in the case forward interpolation, here also we will construct difference table. Suppose that we have a set of values of an unknown function $y_0, y_1, y_2, y_3, ..., y_n$, corresponding to a set of values of the (known) independent variable $x_0, x_1, x_2, x_3, ..., x_n$ where $x_0 < x_1 < x_2 < x_3 < ... < x_n$.

Difference Table is given below:

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$	$\nabla^5 y$
x_0	y_0					
		$\nabla y_0 = y_1 - y_0$				
x_1	y_1		$\nabla y_1 - \nabla y_0$			
		$\nabla y_1 = y_2 - y_1$		$\nabla^2 y_1 - \nabla^2 y_0$		
x_2	y_2		$\nabla y_2 - \nabla y_1$		$\nabla^3 y_1 - \nabla^3 y_0$	
		$\nabla y_2 = y_3 - y_2$		$\nabla^2 y_2 - \nabla^2 y_1$		$\nabla^4 y_1 - \nabla^4 y_0$
x_3	y_3		$\nabla y_3 - \nabla y_2$		$\nabla^3 y_2 - \nabla^3 y_1$	
		$\nabla y_3 = y_4 - y_3$		$\nabla^2 y_3 - \nabla^2 y_2$		
x_4	y_4		$\nabla y_4 - \nabla y_3$			
		$\nabla y_4 = y_5 - y_4$				
x_5	y_5					

Consider that, we have a set of values of an unknown function $y_0, y_1, y_2, y_3, ..., y_n$, corresponding to a set of values of the (known) independent variable $x_0, x_1, x_2, x_3, ..., x_n$ where $x_1 = x_0 + h, x_2 = x_0 + 2h, x_3 = x_0 + 3h, ..., x_{n+1} = x_0 + nh$ and h is the equal spacing between the values $x_0, x_1, x_2, x_3, ..., x_n$.

Then Newton Backward interpolation polynomial is given by

$$P(x) = y_n + \frac{(x - x_n)}{h}\frac{\nabla y_n}{1!} + \frac{(x - x_n)(x - x_{n-1})}{h^2}\frac{\nabla^2 y_n}{2!} + \frac{(x - x_n)(x - x_{n-1})(x - x_{n-2})}{h^3}\frac{\nabla^3 y_0}{3!} + ...$$

$$i.e., P(x) = y_n + v\frac{\nabla y_n}{1!} + v(v+1)\frac{\nabla^2 y_n}{2!} + v(v+1)(v+2)\frac{\nabla^3 y_n}{3!} + ...; \text{ by taking } v = \frac{x - x_n}{h}.$$

Note that we can use the forward difference table also in the case of backward interpolation. While performing forward interpolation we used the leading top values. But in the case of backward interpolation, instead, we will use the leading bottom values from the forward difference table.

12.4.5 Illustrative Example

Example 6.3:

In the following table, use the Newton Backward Interpolation formula to find $f(8.7)$.

Table 8: Data

x	2	4	6	8	10
y	9.68	10.96	12.32	13.76	15.28

Solution:

We will construct the difference table as given below:

x	y	Δy	$\Delta^2 y$
2	9.68		
		Δy_0 = 10.96 - 9.68 = 1.28	
4	10.96		0.08
		Δy_1 = 12.32 - 10.96 = 1.36	
6	12.32		0.08
		Δy_2 = 13.76 - 12.32 = 1.44	
8	13.76		0.08
		Δy_3 = 15.28 - 13.76 = 1.52	
10	15.28		

Here we have, $x = 8.7, x_n = 10, h = 2, v = \frac{x-x_n}{h} = v = \frac{8.7-10}{2} = -0.65$.

Then Newton Backward interpolation polynomial can be written as;

$$f(x) = y_n + \frac{(x-x_n)}{h}\frac{\nabla y_n}{1!} + \frac{(x-x_n)(x-x_{n-1})}{h^2}\frac{\nabla^2 y_n}{2!} + \frac{(x-x_n)(x-x_{n-1})(x-x_{n-2})}{h^3}\frac{\nabla^3 y_n}{3!} + ...$$

$$i.e., f(x) = y_n + v\frac{\nabla y_n}{1!} + v(v+1)\frac{\nabla^2 y_n}{2!} + v(v+1)(v+2)\frac{\Delta^3 y_n}{3!} + ...;$$

$$f(8.7) = 15.28 - 0.65\frac{1.52}{1!} - 0.65(-0.65+1)\frac{0.08}{2!}$$

$$\implies f(8.7) = 15.28 - 0.65 \times 1.52 - 0.65 \times 0.35 \times 0.04$$

$$\therefore f(8.7) = 14.2829.$$

Example 6.4:

In the following table, use the Newton Bacward Interpolation formula to find e^2.

Table 9: Data

x	0.1	0.6	1.1	1.6	2.1
e^x	1.1052	1.8221	3.0042	4.9530	8.1662

Solution:

Here we have, $x = 2, x_n = 2.1, h = 0.5, v = \frac{x-x_n}{h} = \frac{2-2.1}{0.5} = -0.2$.

We will construct the difference table as given below:

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
0.1	1.1052				
		$\Delta y_0 = 1.8221 - 1.1052 = 0.7169$			
0.6	1.8221		0.4652		
		$\Delta y_1 = 3.0042 - 1.8221 = 1.1821$		0.3015	
1.1	3.0042		0.7667		0.1962
		$\Delta y_2 = 4.9530 - 3.0042 = 1.9488$		0.4977	
1.6	4.9530		1.2644		
		$\Delta y_3 = 8.1662 - 4.9530 = 3.2132$			
2.1	8.1662				

Then Newton Backward interpolation polynomial can be written as;

$$f(x) = y_n + \frac{(x-x_n)}{h}\frac{\nabla y_n}{1!} + \frac{(x-x_n)(x-x_{n-1})}{h^2}\frac{\nabla^2 y_n}{2!} + \frac{(x-x_n)(x-x_{n-1})(x-x_{n-2})}{h^3}\frac{\nabla^3 y_n}{3!} + ...$$

$$i.e., f(x) = y_n + v\frac{\nabla y_n}{1!} + v(v+1)\frac{\nabla^2 y_n}{2!} + v(v+1)(v+2)\frac{\nabla^3 y_n}{3!} + ...;$$

$$f(2) = 8.1662 - 0.2\frac{3.2132}{1!} - 0.2(-0.2+1)\frac{1.2644}{2!}$$
$$- 0.2(-0.2+1)(-0.2+2)\frac{0.4977}{3!} + -0.2(-0.2+1)(-0.2+2)(-0.2+3)\frac{0.1962}{4!}$$

$$\implies f(2) = 8.1662 - 0.2 \times 3.2132 - 0.2 \times 0.8 \times 0.6322$$
$$- 0.2 \times 0.8 \times 1.8 \times 0.08295 - 0.2 \times 0.8 \times 1.8 \times 2.8 \times 0.008175$$

$$\therefore \quad f(2) = 7.3919$$

$$i.e., \quad e^2 \approx 7.3919.$$

12.4.6 Practice Problems

From the following table, use the Newton Backward Interpolation formula to find $f(x)$ indicated therein.

Table 10: Exercise-1

x	2	3	4	5	6
$f(x)$	3.00	4.28	5.88	7.8	10.04

at $x = 5.5$.

Table 11: Exercise-2

x	16	18	20	22	24
$f(x)$	261.3	293.7	330.0	372.2	422.3

at $x = 22.4$.

Table 12: Exercise-3

x	0.5	0.6	0.7	0.8
$f(x)$	1.127626	1.185465	1.255169	1.337435

at $x = 0.76$.

Table 13: Exercise-4

Year :x	1971	1981	1991	2001	2011
Population (in lacs): f(x)	12	19	31	47	62

in the year 2005.

Table 14: Exercise-5

x	0	1	2	3	4	5
$f(x)$	1	14	15	10	5	6

at $x = 4.5$.

12.4.7 Lagranges interpolation

Consider that, we have a set of values of an unknown function $y_0, y_1, y_2, y_3, ..., y_n$, corresponding to a set of values of the (known) independent variable $x_0, x_1, x_2, x_3, ..., x_n$, in general for both equal and unequal spacing.

Then the Lagrangian interpolating polynomial is given by

$$P(x) = \sum_{i=0}^{n} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

Note that when $n = 1, 2, 3$, $P(x)$ are given below:

$$\begin{aligned} P(x) &= \sum_{i=0}^{1} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \\ &= \frac{(x - x_1)}{(x_0 - x_1)} y_0 + \frac{(x - x_0)}{(x_1 - x_0)} y_1. \end{aligned}$$

$$\begin{aligned} P(x) &= \sum_{i=0}^{2} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \\ &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} y_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} y_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} y_2. \end{aligned}$$

and

$$\begin{aligned}P(x) &= \sum_{i=0}^{3} y_i \prod_{j\neq i} \frac{x-x_j}{x_i-x_j} \\ &= \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}y_0 + \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}y_1 \\ &+ \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}y_2 + \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}y_3.\end{aligned}$$

12.4.8 Illustrative Example

Example 6.5

Find the value of $f(x)$ at $x = 0$ given some set of values $(-2, 5), (1, 7), (3, 11), (7, 34)$.

Solution

Here we have $n = 3$, $x_0 = -2, x_1 = 1, x_2 = 3, x_3 = 7$ and $y_0 = 5, y_1 = 7, y_2 = 11, y_3 = 34$. Ans we arsked find the value of $f(x)$ at $x = 0$.

we use Langranges Interpolation polynomial with $n = 3$.

$$\begin{aligned}P(x) &= \sum_{i=0}^{3} y_i \prod_{j\neq i} \frac{x-x_j}{x_i-x_j} \\ &= \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}y_0 + \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}y_1 \\ &+ \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}y_2 + \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}y_3 \\ &= \frac{(x-1)(x-3)(x-7)}{(-2-1)(-2-3)(-2-7)} \times 5 + \frac{(x+2)(x-3)(x-7)}{(1+2)(1-3)(1-7)} \times 7 \\ &+ \frac{(x+2)(x-1)(x-7)}{(3+2)(3-1)(3-7))} \times 11 + \frac{(x+2)(x-1)(x-3)}{(7+2)(7-1)(7-3)} \times 34 \\ \therefore\ P(x) &= \frac{43x^3}{1080} + \frac{101x^2}{540} + \frac{793x}{1080} + \frac{1087}{180},\end{aligned}$$

which is the required Lagranges interpolation polynomial.

Now the value of $f(x)$ at $x = 0$ is given by $P(0) = \frac{1087}{180}$.

Example 6.6

Use appropriate Lagrange polynomial of degree 3 to approximate $f(10)$ if $f(5) = 12, f(6) = 13, f(9) = 14, f(11) = 16$.

Solution

Here we have $n = 3$, $x_0 = 5, x_1 = 6, x_2 = 9, x_3 = 11$ and $y_0 = 12, y_1 = 13, y_2 = 14, y_3 = 16$. Ans we arsked find the value of $f(x)$ at $x = 0$.

we use Langranges Interpolation polynomial with $n = 3$.

$$\begin{aligned}
P(x) &= \sum_{i=0}^{3} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \\
&= \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} y_0 + \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} y_1 \\
&+ \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} y_2 + \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} y_3 \\
&= \frac{(x-6)(x-9)(x-11)}{(5-6)(5-9)(5-11)} \times 12 + \frac{(x-5)(x-9)(x-11)}{(6-5)(6-9)(6-11)} \times 13 \\
&+ \frac{(x-5)(x-6)(x-11)}{(9-5)(9-6)(9-11))} \times 14 + \frac{(x-5)(x-6)(x-9)}{(11-5)(11-6)(11-9)} \times 16 \\
\therefore \quad P(x) &= \frac{x^3}{20} - \frac{7x^2}{6} + \frac{557x}{60} - \frac{23}{2},
\end{aligned}$$

which is the required Lagranges interpolation polynomial.

Now the value of $f(x)$ at $x = 10$ is given by $P(0) = \frac{44}{3} = 14.6666$.

12.4.9 Practice Problems

From the following table, use the Lagranges Interpolation formula to find $f(x)$ indicated therein.

Table 15: Exercise-1

x	2	4	5	6
$f(x)$	3.00	5.88	7.8	10.04

at $x = 3$.

Table 16: Exercise-2

x	16	20	22	24
$f(x)$	261.3	330.0	372.2	422.3

at $x = 23$.

Table 17: Exercise-3

x	0.5	0.6	0.8
$f(x)$	1.127626	1.185465	1.337435

at $x = 0.7$.

Table 18: Exercise-4

Year :x	1971	1981	1991	2011
Population (in lacs): f(x)	12	19	31	62

in the year 2005.

Table 19: Exercise-5

x	0	2	3	5
$f(x)$	1	15	10	6

at $x = 4$.

12.5 Solution of system of equations

12.5.1 Gauss Elimination Method

Gaussian Elimination, a method to solve a system of linear equations $AX = b$, , except for certain special cases. Gaussian elimination is summarized by the following three steps:

1.Write the system of equations in matrix form. Form the augmented matrix. We omit the symbols for the variables, the equal signs, and just write the coefficients and the unknowns in a matrix.

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

3.An elementary row operation is one of the following:

i)Multiply each element of the row by a non-zero constant.

ii)Switch two rows.

iii)Add (or subtract) a non-zero constant times a row to another row.

12.5.2 Illustrative Example

Example 1.1

Use the Gauss Elimination Method to solve the following system of equations.

$$\begin{aligned} 1.0x_1 + 2.0x_2 + 3.0x_3 &= 6.0 \\ 2.0x_1 + 2.0x_2 + 1.0x_3 &= 5.0 \\ 1.0x_1 + 1.0x_2 + 2.0x_3 &= 4.0 \end{aligned}$$

Solution

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 \\ 2.0 & 2.0 & 1.0 \\ 1.0 & 1.0 & 2.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6.0 \\ 5.0 \\ 4.0 \end{bmatrix}$$

The augmented matrix A:B is given by

$$A : B = \begin{bmatrix} 1.0 & 2.0 & 3.0 & 6.0 \\ 2.0 & 2.0 & 1.0 & 5.0 \\ 1.0 & 1.0 & 2.0 & 4.0 \end{bmatrix}$$

Now we will transform the above matrix into a matrix having only zeros below the diagonal.

The elementary row operations are indicated below the transormed matrices.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 6.0 \\ 0.0 & -2.0 & -5.0 & -7.0 \\ 1.0 & 1.0 & 2.0 & 4.0 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - 2.0 * R_1.$

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 6.0 \\ 0.0 & -2.0 & -5.0 & -7.0 \\ 0.0 & -1.0 & -1.0 & -2.0 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - 1.0 * R_1.$

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 6.0 \\ 0.0 & -2.0 & -5.0 & -7.0 \\ 0.0 & 0.0 & 1.5 & 1.5 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - 0.5 * R_2.$

After performing the back substitution, we get the solution as,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Example 1.2

Use the Gauss Elimination Method to solve the following system of equations.

$$\begin{aligned} 1.0x_1 + 2.0x_2 + 3.0x_3 &= 9.0 \\ 4.0x_1 + 5.0x_2 + 6.0x_3 &= 24.0 \\ 3.0x_1 + 1.0x_2 - 2.0x_3 &= 4.0 \end{aligned}$$

Solution

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 3.0 & 1.0 & -2.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} 9.0 \\ 24.0 \\ 4.0 \end{bmatrix}$$

The augmented matrix A:B is given by

$$A : B = \begin{bmatrix} 1.0 & 2.0 & 3.0 & 9.0 \\ 4.0 & 5.0 & 6.0 & 24.0 \\ 3.0 & 1.0 & -2.0 & 4.0 \end{bmatrix}$$

Now we will transform the above matrix into a matrix having only zeros below the diagonal.

The elementary row operations are indicated below the transformed matrices.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 9.0 \\ 0.0 & -3.0 & -6.0 & -12.0 \\ 3.0 & 1.0 & -2.0 & 4.0 \end{bmatrix}$$

By performing the row operation $R_2 \leftarrow R_2 - 4.0 * R_1$.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 9.0 \\ 0.0 & -3.0 & -6.0 & -12.0 \\ 0.0 & -5.0 & -11.0 & -23.0 \end{bmatrix}$$

By performing the row operation $R_3 \leftarrow R_3 - 3.0 * R_1$.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 9.0 \\ 0.0 & -3.0 & -6.0 & -12.0 \\ 0.0 & 0.0 & -1.0 & -3.0 \end{bmatrix}$$

By performing the row operation $R_3 \leftarrow R_3 - 1.667 * R_2$.

After performing the back substitution, we get the solution as,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \\ 3 \end{bmatrix}$$

Example 1.3

Use the Gauss Elimination Method to solve the following system of equations.

$$\begin{aligned} 2.0x_1 + 3.0x_2 - 1.0x_3 &= 5.0 \\ 4.0x_1 + 4.0x_2 - 3.0x_3 &= 3.0 \\ 2.0x_1 - 3.0x_2 + 2.0x_3 &= 2.0 \end{aligned}$$

Solution

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 2.0 & 3.0 & -1.0 \\ 4.0 & 4.0 & -3.0 \\ 2.0 & -3.0 & 2.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} 5.0 \\ 3.0 \\ 2.0 \end{bmatrix}$$

The augmented matrix A:B is given by

$$A : B = \begin{bmatrix} 2.0 & 3.0 & -1.0 & 5.0 \\ 4.0 & 4.0 & -3.0 & 3.0 \\ 2.0 & -3.0 & 2.0 & 2.0 \end{bmatrix}$$

Now we will transform the above matrix into a matrix having only zeros below the diagonal.

The elementary row operations are indicated below the transformed matrices.

$$\begin{bmatrix} 2.0 & 3.0 & -1.0 & 5.0 \\ 0.0 & -2.0 & -1.0 & -7.0 \\ 2.0 & -3.0 & 2.0 & 2.0 \end{bmatrix}$$

By performing the row operation $R_2 \leftarrow R_2 - \frac{4.0}{2.0} \times R_1$.

$$\begin{bmatrix} 2.0 & 3.0 & -1.0 & 5.0 \\ 0.0 & -2.0 & -1.0 & -7.0 \\ 0.0 & -6.0 & 3.0 & -3.0 \end{bmatrix}$$

By performing the row operation $R_3 \leftarrow R_3 - \frac{2.0}{2.0} \times R_1$.

$$\begin{bmatrix} 2.0 & 3.0 & -1.0 & 5.0 \\ 0.0 & -2.0 & -1.0 & -7.0 \\ 0.0 & 0.0 & 6.0 & 18.0 \end{bmatrix}$$

By performing the row operation $R_3 \leftarrow R_3 - \frac{-6.0}{-2.0} \times R_2$.

After performing the back substitution, we get the solution as,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Example 1.4

Use the Gauss Elimination Method to solve the following system of equations.

$$\begin{aligned} 1.0x_1 + 1.0x_2 + 5.0x_3 &= 7.0 \\ 2.0x_1 + 10.0x_2 + 1.0x_3 &= 13.0 \\ 10.0x_1 + 1.0x_2 + 1.0x_3 &= 12.0 \end{aligned}$$

Solution

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 1.0 & 1.0 & 5.0 \\ 2.0 & 10.0 & 1.0 \\ 10.0 & 1.0 & 1.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} 7.0 \\ 13.0 \\ 12.0 \end{bmatrix}$$

The augmented matrix A:B is given by

$$A : B = \begin{bmatrix} 1.0 & 1.0 & 5.0 & 7.0 \\ 2.0 & 10.0 & 1.0 & 13.0 \\ 10.0 & 1.0 & 1.0 & 12.0 \end{bmatrix}$$

Now we will transform the above matrix into a matrix having only zeros below the diagonal.

The elementary row operations are indicated below the transformed matrices.

$$\begin{bmatrix} 1.0 & 1.0 & 5.0 & 7.0 \\ 0.0 & 8.0 & -9.0 & -1.0 \\ 10.0 & 1.0 & 1.0 & 12.0 \end{bmatrix}$$

By performing the row operation $R_2 \leftarrow R_2 - \frac{2.0}{1.0} \times R_1$.

$$\begin{bmatrix} 1.0 & 1.0 & 5.0 & 7.0 \\ 0.0 & 8.0 & -9.0 & -1.0 \\ 0.0 & -9.0 & -49.0 & -58.0 \end{bmatrix}$$

By performing the row operation $R_3 \leftarrow R_3 - \frac{10.0}{1.0} \times R_1$.

$$\begin{bmatrix} 1.0 & 1.0 & 5.0 & 7.0 \\ 0.0 & 8.0 & -9.0 & -1.0 \\ 0.0 & 0.0 & -59.125 & -59.125 \end{bmatrix}$$

By performing the row operation $R_3 \leftarrow R_3 - \frac{-9.0}{8.0} \times R_2$.

After performing the back substitution, we get the solution as,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Example 1. 5

Use the Gauss Elimination Method to solve the following system of equations.

$$\begin{aligned}
1.0x_1 + 2.0x_2 + 3.0x_3 + 4.0x_4 &= 20.0\\
3.0x_1 - 2.0x_2 + 8.0x_3 + 4.0x_4 &= 26.0\\
2.0x_1 + 1.0x_2 - 4.0x_3 + 7.0x_4 &= 10.0\\
2.0x_1 + 1.0x_2 - 4.0x_3 - 2.0x_4 &= 1.0
\end{aligned}$$

Solution

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 \\ 3.0 & -2.0 & 8.0 & 4.0 \\ 2.0 & 1.0 & -4.0 & 7.0 \\ 2.0 & 1.0 & -4.0 & -2.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \begin{bmatrix} 20.0 \\ 26.0 \\ 10.0 \\ 1.0 \end{bmatrix}$$

The augmented matrix A:B is given by

$$A : B = \begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 20.0 \\ 3.0 & -2.0 & 8.0 & 4.0 & 26.0 \\ 2.0 & 1.0 & -4.0 & 7.0 & 10.0 \\ 2.0 & 1.0 & -4.0 & -2.0 & 1.0 \end{bmatrix}$$

Now we will transform the above matrix into a matrix having only zeros below the diagonal.

The elementary row operations are indicated below the transformed matrices.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 20.0 \\ 0.0 & -8.0 & -1.0 & -8.0 & -34.0 \\ 2.0 & 1.0 & -4.0 & 7.0 & 10.0 \\ 2.0 & 1.0 & -4.0 & -2.0 & 1.0 \end{bmatrix}$$

By performing the row operation $R_2 \leftarrow R_2 - \frac{3.0}{1.0} \times R_1$.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 20.0 \\ 0.0 & -8.0 & -1.0 & -8.0 & -34.0 \\ 0.0 & -3.0 & -10.0 & -1.0 & -30.0 \\ 2.0 & 1.0 & -4.0 & -2.0 & 1.0 \end{bmatrix}$$

By performing the row operation $R_3 \leftarrow R_3 - \frac{2.0}{1.0} \times R_1$.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 20.0 \\ 0.0 & -8.0 & -1.0 & -8.0 & -34.0 \\ 0.0 & -3.0 & -10.0 & -1.0 & -30.0 \\ 0.0 & -3.0 & -10.0 & -10.0 & -39.0 \end{bmatrix}$$

By performing the row operation $R_4 \leftarrow R_4 - \frac{2.0}{1.0} \times R_1$.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 20.0 \\ 0.0 & -8.0 & -1.0 & -8.0 & -34.0 \\ 0.0 & 0.0 & -9.625 & 2.0 & -17.25 \\ 0.0 & -3.0 & -10.0 & -10.0 & -39.0 \end{bmatrix}$$

By performing the row operation $R_3 \leftarrow R_3 - \frac{-3.0}{-8.0} \times R_2$.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 20.0 \\ 0.0 & -8.0 & -1.0 & -8.0 & -34.0 \\ 0.0 & 0.0 & -9.625 & 2.0 & -17.25 \\ 0.0 & 0.0 & -9.625 & -7.0 & -26.25 \end{bmatrix}$$

By performing the row operation $R_4 \leftarrow R_4 - \frac{-3.0}{-8.0} \times R_2$.

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 20.0 \\ 0.0 & -8.0 & -1.0 & -8.0 & -34.0 \\ 0.0 & 0.0 & -9.625 & 2.0 & -17.25 \\ 0.0 & 0.0 & 0.0 & -9.0 & -9.0 \end{bmatrix}$$

By performing the row operation $R_4 \leftarrow R_4 - \frac{-9.625}{-9.625} \times R_3$.

After performing the back substitution, we get the solution as,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

12.5.3 Practice Problems

Use the Gauss Elimination Method to solve the following system of equations.

Exercise 1.1

$$\begin{aligned} 1.0x_1 + 1.0x_2 + 5.0x_3 &= 16.0 \\ 2.0x_1 + 3.0x_2 + 1.0x_3 &= 4.0 \\ 4.0x_1 + 1.0x_2 - 1.0x_3 &= 4.0 \end{aligned}$$

Exercise 1.2

$$\begin{aligned} 1.0x_1 - 1.0x_2 + 1.0x_3 &= 6.0 \\ 2.0x_1 + 4.0x_2 + 1.0x_3 &= 3.0 \\ 3.0x_1 + 2.0x_2 - 2.0x_3 &= -2.0 \end{aligned}$$

Exercise 1.3

$$\begin{aligned} 2.0x_1 + 1.0x_2 + 4.0x_3 &= 12.0 \\ 8.0x_1 - 3.0x_2 + 2.0x_3 &= 20.0 \\ 4.0x_1 + 11.0x_2 - 1.0x_3 &= 33 \end{aligned}$$

Exercise 1.4

$$\begin{aligned} 1.0x_1 + 1.0x_2 + 1.0x_3 &= 3.0 \\ 2.0x_1 + 3.0x_2 + 4.0x_3 &= 9.0 \\ 3.0x_1 - 3.0x_2 + 1.0x_3 &= 1.0 \end{aligned}$$

Exercise 1.5

$$\begin{aligned} 1.0x_1 + 1.0x_2 - 1.0x_3 &= 1.0 \\ 3.0x_1 + 1.0x_2 + 1.0x_3 &= 1.0 \\ 4.0x_1 + 3.0x_2 + 2.0x_3 &= -1.0 \end{aligned}$$

12.6 Curve fitting

12.6.1 Method of Least-squares for fitting a straight line

Fitting a straight line of best fit $y = a + bx$, from the sample data $(x_i, y_i), i = 0, 1, 2, 3, ..., n$ is a common problem in estimation. It is also known as Linear Regression. The Method of Least Squares is a procedure to determine the best fit line to data based on the principles from calculus and linear algebra.

This method assures that, as long as the $x's$ are not all equal, the best fit values of a and b are obtained by solving a linear system of equations known as Normal equations that are given below.

$$\begin{aligned} n \times a + \Sigma x \times b &= \Sigma y \\ \Sigma x \times a + \Sigma x^2 \times b &= \Sigma xy \end{aligned}$$

Once a best fit linear regression is obtained, that it can be used for predicting the values of y for the given value of x.

12.6.2 Illustrative Example

Example 1.1

Use the least square method to fit a straight line of the form $y = a + bx$ using the data given below and find the value of y at $x = 5$.

x :	0.0	0.5	1.0	1.5	2.0	2.5
y :	0.0	1.5	3.0	4.5	6.0	7.5

Solution

The data points are given by,

x :	0.0	0.5	1.0	1.5	2.0	2.5
y :	0.0	1.5	3.0	4.5	6.0	7.5

Hence the corresponding normal equations can be obtained using the equations,

$$n \times a + \Sigma x \times b = \Sigma y$$
$$\Sigma x \times a + \Sigma x^2 \times b = \Sigma xy$$

By substituting the values obtained from the samples into the above normal equations, we have

$$6.0a + 7.5b = 22.5$$
$$7.5a + 13.75b = 41.25$$

Now we have to solve these equations for a and b .

We will use Gauss elimination method for solving them.

For,

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 6.0 & 7.5 \\ 7.5 & 13.75 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} 22.5 \\ 41.25 \end{bmatrix}$$

$$A : B = \begin{bmatrix} 6.0 & 7.5 & 22.5 \\ 7.5 & 13.75 & 41.25 \end{bmatrix}$$

By permorming the elementary row operations as indicated below so as to get upper triangular matrix, we have

$$\begin{bmatrix} 6.0 & 7.5 & 22.5 \\ 0.0 & 4.375 & 13.125 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{7.5}{6.0} \times R_1.$

By back substitution, we get the value of a and b as,

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0.0 \\ 3.0 \end{bmatrix}$$

$\therefore$ the required straight line is given by,

$$\begin{aligned} y &= a + bx \\ y &= 0.0 + 3.0 \times x. \end{aligned}$$

Value of y at x = 5.0 is = 15.0000.

Example 1.2

Use the least square method to fit a straight line of the form $y = a + bx$ using the data given below and find the value of y at $x = 7$.

x :	1.0	2.0	3.0	4.0	5.0
y :	3.0	5.0	7.0	9.0	11.0

Solution

The data points are given by,

x :	1.0	2.0	3.0	4.0	5.0
y :	3.0	5.0	7.0	9.0	11.0

Hence the corresponding normal equations can be obtained using the equations,

$$\begin{aligned} n \times a + \Sigma x \times b &= \Sigma y \\ \Sigma x \times a + \Sigma x^2 \times b &= \Sigma xy \end{aligned}$$

By substituting the values obtained from the samples into the above normal equations, we have

$$\begin{aligned} 5.0a + 15.0b + &= 35.0 \\ 15.0a + 55.0b + &= 125.0 \end{aligned}$$

Now we have to solve these equations for a and b .

We will use Gauss elimination method for solving them.

For,

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 5.0 & 15.0 \\ 15.0 & 55.0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} 35.0 \\ 125.0 \end{bmatrix}$$

$$A : B = \begin{bmatrix} 5.0 & 15.0 & 35.0 \\ 15.0 & 55.0 & 125.0 \end{bmatrix}$$

By permorming the elementary row operations as indicated below so as to get upper triangular matrix, we have

$$\begin{bmatrix} 5.0 & 15.0 & 35.0 \\ 0.0 & 10.0 & 20.0 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{15.0}{5.0} \times R_1.$

By back substitution, we get the value of a and b as,

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1.0 \\ 2.0 \end{bmatrix}$$

$\therefore$ the required straight line is given by,

$$\begin{aligned} y &= a + bx \\ y &= 1.0 + 2.0 \times x. \end{aligned}$$

Value of y at x = 7.0 = 15.0000.

Example 1.3

Use the least square method to fit a straight line of the form $y = a + bx$ using the data given below and find the value of y at $x = 4$.

x :	0.0	1.0	2.0	3.0	4.0
y :	-1.0	2.0	5.0	8.0	11.0

Solution

The data points are given by,

x :	0.0	1.0	2.0	3.0	4.0
y :	-1.0	2.0	5.0	8.0	11.0

Hence the corresponding normal equations can be obtained using the equations,

$$\begin{aligned} n \times a + \Sigma x \times b &= \Sigma y \\ \Sigma x \times a + \Sigma x^2 \times b &= \Sigma xy \end{aligned}$$

By substituting the values obtained from the samples into the above normal equations, we have

$$\begin{aligned} 5.0a + 10.0b + &= 25.0 \\ 10.0a + 30.0b + &= 80.0 \end{aligned}$$

Now we have to solve these equations for a and b .

We will use Gauss elimination method for solving them.

For,

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 5.0 & 10.0 \\ 10.0 & 30.0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} 25.0 \\ 80.0 \end{bmatrix}$$

$$A : B = \begin{bmatrix} 5.0 & 10.0 & 25.0 \\ 10.0 & 30.0 & 80.0 \end{bmatrix}$$

By permorming the elementary row operations as indicated below so as to get upper triangular matrix, we have

$$\begin{bmatrix} 5.0 & 10.0 & 25.0 \\ 0.0 & 10.0 & 30.0 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{10.0}{5.0} \times R_1.$

By back substitution, we get the value of a and b as,

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} -1.0 \\ 3.0 \end{bmatrix}$$

$\therefore$ the required straight line is given by,

$$\begin{aligned} y &= a + bx \\ y &= -1.0 + 3.0 \times x. \end{aligned}$$

Value of y at x = 4 = 11.0000.

Example 1.4

Use the least square method to fit a straight line of the form $y = a + bx$ using the data given below and find the value of y at $x = 2.5$.

x :	-1.0	0.0	1.0	2.0	3.0
y :	0.0	2.0	4.0	8.0	6.0

Solution

The data points are given by,

x :	-1.0	0.0	1.0	2.0	3.0
y :	0.0	2.0	4.0	8.0	6.0

Hence the corresponding normal equations can be obtained using the equations,

$$\begin{aligned} n \times a + \Sigma x \times b &= \Sigma y \\ \Sigma x \times a + \Sigma x^2 \times b &= \Sigma xy \end{aligned}$$

By substituting the values obtained from the samples into the above normal equations, we have

$$5.0a + 5.0b + = 20.0$$
$$5.0a + 15.0b + = 38.0$$

Now we have to solve these equations for a and b .

We will use Gauss elimination method for solving them.

For,

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 5.0 & 5.0 \\ 5.0 & 15.0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} 20.0 \\ 38.0 \end{bmatrix}$$

$$A : B = \begin{bmatrix} 5.0 & 5.0 & 20.0 \\ 5.0 & 15.0 & 38.0 \end{bmatrix}$$

By permorming the elementary row operations as indicated below so as to get upper triangular matrix, we have

$$\begin{bmatrix} 5.0 & 5.0 & 20.0 \\ 0.0 & 10.0 & 18.0 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{5.0}{5.0} \times R_1.$

By back substitution, we get the value of a and b as,

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 2.2 \\ 1.8 \end{bmatrix}$$

$\therefore$ the required straight line is given by,

$$\begin{aligned} y &= a + bx \\ y &= 2.2 + 1.8 \times x. \end{aligned}$$

Value of y at x = 2.5 = 6.7000.

Example 1.5

Use the least square method to fit a straight line of the form $y = a + bx$ using the data given below and find the value of y at $x = 5.5$.

x :	0.0	5.0	10.0	15.0	20.0
y :	25.0	20.0	15.0	10.0	5.0

Solution

The data points are given by,

$$\begin{array}{lccccc} x: & 0.0 & 5.0 & 10.0 & 15.0 & 20.0 \\ y: & 25.0 & 20.0 & 15.0 & 10.0 & 5.0 \end{array}$$

Hence the corresponding normal equations can be obtained using the equations,

$$n \times a + \Sigma x \times b = \Sigma y$$
$$\Sigma x \times a + \Sigma x^2 \times b = \Sigma xy$$

By substituting the values obtained from the samples into the above normal equations, we have

$$5.0a + 50.0b + = 75.0$$
$$50.0a + 750.0b + = 500.0$$

Now we have to solve these equations for a and b .

We will use Gauss elimination method for solving them.

For,

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 5.0 & 50.0 \\ 50.0 & 750.0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} 75.0 \\ 500.0 \end{bmatrix}$$

$$A : B = \begin{bmatrix} 5.0 & 50.0 & 75.0 \\ 50.0 & 750.0 & 500.0 \end{bmatrix}$$

By permorming the elementary row operations as indicated below so as to get upper triangular matrix, we have

$$\begin{bmatrix} 5.0 & 50.0 & 75.0 \\ 0.0 & 250.0 & -250.0 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{50.0}{5.0} \times R_1.$

By back substitution, we get the value of a and b as,

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 25.0 \\ -1.0 \end{bmatrix}$$

$\therefore$ the required straight line is given by,

$$\begin{aligned} y &= a + bx \\ y &= 25.0 + -1.0 \times x. \end{aligned}$$

Value of y at x = 5.5 = 19.5000.

12.6.3 Method of Least-squares for fitting a parabola

Best fitting of a parabola $y = a + bx + cx^2$, from the sample data $(x_i, y_i), i = 0, 1, 2, 3, ..., n$ is a common problem in estimation. It is also known as quadratic polynomial fitting. Using the method of least squares we can also determine the best fit parabola from the data provided.

Similar to the fitting of straight line, we have the following normal equations.

$$\begin{aligned} n \times a + \Sigma x \times b + \Sigma x^2 \times b &= \Sigma y \\ \Sigma x \times a + \Sigma x^2 \times b + \Sigma x^3 \times b &= \Sigma xy \\ \Sigma x^2 \times a + \Sigma x^3 \times b + \Sigma x^4 \times b &= \Sigma x^2 y \end{aligned}$$

Once a best fit quadratic polynomial is obtained, that can be used for predicting the values of y for the given set of value of x.

12.6.4 Illustrative Example

Example 1.1

Use the least square method to fit a parabola of the form $y = a + bx + cx^2$ using the data given below and find the value of y at $x = 2.0$.

x :	0.0	0.5	1.0	1.5	2.0	2.5
y :	0.0	1.5	3.0	4.5	6.0	7.5

Solution

The data points are given by,

x :	0.0	0.5	1.0	1.5	2.0	2.5
y :	0.0	1.5	3.0	4.5	6.0	7.5

Hence the corresponding normal equations can be obtained using the equations,

$$\begin{aligned} n \times a + \Sigma x \times b + \Sigma x^2 \times b &= \Sigma y \\ \Sigma x \times a + \Sigma x^2 \times b + \Sigma x^3 \times b &= \Sigma xy \\ \Sigma x^2 \times a + \Sigma x^3 \times b + \Sigma x^4 \times b &= \Sigma x^2 y \end{aligned}$$

By substituting the values obtained from the samples into the above normal equations, we have

$$\begin{aligned} 6.0a + 7.5b + 13.75c &= 22.5 \\ 7.5a + 13.75b + 28.125c &= 41.25 \\ 13.75a + 28.125b + 61.1875c &= 84.375 \end{aligned}$$

Now we have to solve these equations for a and b .

We will use Gauss elimination method for solving them.

For,

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 6.0 & 7.5 & 13.75 \\ 7.5 & 13.75 & 28.125 \\ 13.75 & 28.125 & 61.1875 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 22.5 \\ 41.25 \\ 84.375 \end{bmatrix}$$

$$A : B = \begin{bmatrix} 6.0 & 7.5 & 13.75 & 22.5 \\ 7.5 & 13.75 & 28.125 & 41.25 \\ 13.75 & 28.125 & 61.1875 & 84.375 \end{bmatrix}$$

By permorming the elementary row operations as indicated below so as to get upper triangular matrix, we have

$$\begin{bmatrix} 6.0000 & 7.5000 & 13.7500 & 22.5000 \\ 0.0000 & 4.3750 & 10.9375 & 13.1250 \\ 13.7500 & 28.1250 & 61.1875 & 84.3750 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{7.5}{6.0} \times R_1.$

$$\begin{bmatrix} 6.0000 & 7.5000 & 13.7500 & 22.5000 \\ 0.0000 & 4.3750 & 10.9375 & 13.1250 \\ 0.0000 & 10.9375 & 29.6771 & 32.8125 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{13.75}{6.0} \times R_1.$

$$\begin{bmatrix} 6.0000 & 7.5000 & 13.7500 & 22.5000 \\ 0.0000 & 4.3750 & 10.9375 & 13.1250 \\ 0.0000 & 0.0000 & 2.3333 & 0.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{10.9375}{4.375} \times R_2.$

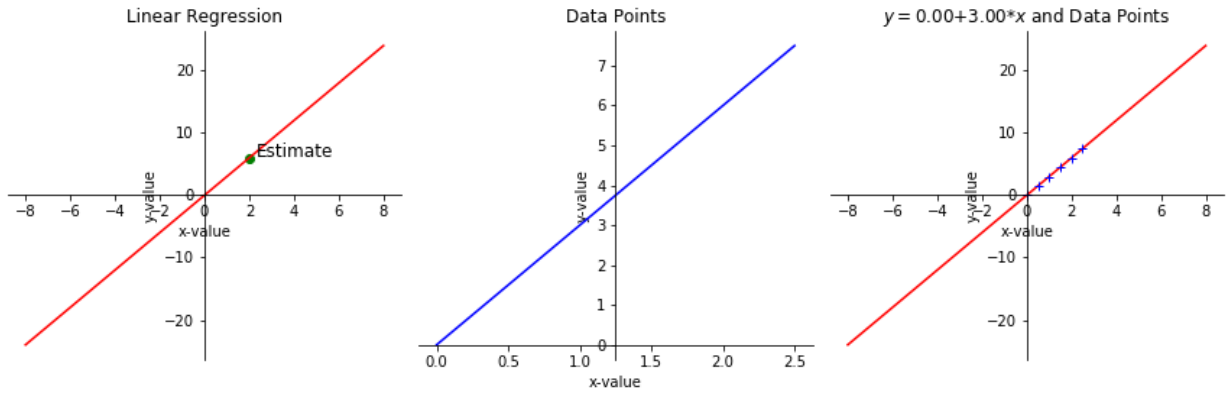
By back substitution, we get the value of a and b as,

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0.0 \\ 3.0 \\ 0.0 \end{bmatrix}$$

$\therefore$ the required straight line is given by,

$$\begin{aligned} y &= a + bx + cx^2 \\ y &= 0.0 + 3.0 \times x + 0.0 \times x^2. \end{aligned}$$

Value of y at x = 2.0 = 6.



Example 1.2

Use the least square method to fit a parabola of the form $y = a + bx + cx^2$ using the data given below and find the value of y at $x = 1.5$.

$x:$	0.0	1.0	2.0	3.0	4.0
$y:$	1.0	6.0	17.0	34.0	57.0

Solution

The data points are given by,

$x:$	0.0	1.0	2.0	3.0	4.0
$y:$	1.0	6.0	17.0	34.0	57.0

Hence the corresponding normal equations can be obtained using the equations,

$$n \times a + \Sigma x \times b + \Sigma x^2 \times b = \Sigma y$$
$$\Sigma x \times a + \Sigma x^2 \times b + \Sigma x^3 \times b = \Sigma xy$$
$$\Sigma x^2 \times a + \Sigma x^3 \times b + \Sigma x^4 \times b = \Sigma x^2 y$$

By substituting the values obtained from the samples into the above normal equations, we have

$$5.0a + 10.0b + 30.0c = 115.0$$
$$10.0a + 30.0b + 100.0c = 370.0$$
$$30.0a + 100.0b + 354.0c = 1292.0$$

Now we have to solve these equations for a and b .

We will use Gauss elimination method for solving them.

For,

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 5.0 & 10.0 & 30.0 \\ 10.0 & 30.0 & 100.0 \\ 30.0 & 100.0 & 354.0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 115.0 \\ 370.0 \\ 1292.0 \end{bmatrix}$$

$$A : B = \begin{bmatrix} 5.0 & 10.0 & 30.0 & 115.0 \\ 10.0 & 30.0 & 100.0 & 370.0 \\ 30.0 & 100.0 & 354.0 & 1292.0 \end{bmatrix}$$

By permorming the elementary row operations as indicated below so as to get upper triangular matrix, we have

$$\begin{bmatrix} 5.0000 & 10.0000 & 30.0000 & 115.0000 \\ 0.0000 & 10.0000 & 40.0000 & 140.0000 \\ 30.0000 & 100.0000 & 354.0000 & 1292.0000 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{10.0}{5.0} \times R_1.$

$$\begin{bmatrix} 5.0000 & 10.0000 & 30.0000 & 115.0000 \\ 0.0000 & 10.0000 & 40.0000 & 140.0000 \\ 0.0000 & 40.0000 & 174.0000 & 602.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{30.0}{5.0} \times R_1.$

$$\begin{bmatrix} 5.0000 & 10.0000 & 30.0000 & 115.0000 \\ 0.0000 & 10.0000 & 40.0000 & 140.0000 \\ 0.0000 & 0.0000 & 14.0000 & 42.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{40.0}{10.0} \times R_2.$

By back substitution, we get the value of a and b as,

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \end{bmatrix}$$

$\therefore$ the required straight line is given by,

$$\begin{aligned} y &= a + bx + cx^2 \\ y &= 1.0 + 2.0 \times x + 3.0 \times x^2. \end{aligned}$$

Value of y at x = 1.5 = 10.7500.

Example 1.3

Use the least square method to fit a parabola of the form $y = a + bx + cx^2$ using the data given below and find the value of y at $x = 2.5$.

x :	0.0	1.0	2.0	3.0	4.0
y :	0.0	1.0	8.0	21.0	40.0

Solution

The data points are given by,

$$\begin{array}{lccccc} x: & 0.0 & 1.0 & 2.0 & 3.0 & 4.0 \\ y: & 0.0 & 1.0 & 8.0 & 21.0 & 40.0 \end{array}$$

Hence the corresponding normal equations can be obtained using the equations,

$$\begin{aligned} n \times a + \Sigma x \times b + \Sigma x^2 \times b &= \Sigma y \\ \Sigma x \times a + \Sigma x^2 \times b + \Sigma x^3 \times b &= \Sigma xy \\ \Sigma x^2 \times a + \Sigma x^3 \times b + \Sigma x^4 \times b &= \Sigma x^2 y \end{aligned}$$

By substituting the values obtained from the samples into the above normal equations, we have

$$\begin{aligned} 5.0a + 10.0b + 30.0c &= 70.0 \\ 10.0a + 30.0b + 100.0c &= 240.0 \\ 30.0a + 100.0b + 354.0c &= 862.0 \end{aligned}$$

Now we have to solve these equations for a and b .

We will use Gauss elimination method for solving them.

For,

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 5.0 & 10.0 & 30.0 \\ 10.0 & 30.0 & 100.0 \\ 30.0 & 100.0 & 354.0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 70.0 \\ 240.0 \\ 862.0 \end{bmatrix}$$

$$A : B = \begin{bmatrix} 5.0 & 10.0 & 30.0 & 70.0 \\ 10.0 & 30.0 & 100.0 & 240.0 \\ 30.0 & 100.0 & 354.0 & 862.0 \end{bmatrix}$$

By permorming the elementary row operations as indicated below so as to get upper triangular matrix, we have

$$\begin{bmatrix} 5.0000 & 10.0000 & 30.0000 & 70.0000 \\ 0.0000 & 10.0000 & 40.0000 & 100.0000 \\ 30.0000 & 100.0000 & 354.0000 & 862.0000 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{10.0}{5.0} \times R_1.$

$$\begin{bmatrix} 5.0000 & 10.0000 & 30.0000 & 70.0000 \\ 0.0000 & 10.0000 & 40.0000 & 100.0000 \\ 0.0000 & 40.0000 & 174.0000 & 442.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{30.0}{5.0} \times R_1.$

$$\begin{bmatrix} 5.0000 & 10.0000 & 30.0000 & 70.0000 \\ 0.0000 & 10.0000 & 40.0000 & 100.0000 \\ 0.0000 & 0.0000 & 14.0000 & 42.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{40.0}{10.0} \times R_2.$

By back substitution, we get the value of a and b as,

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0.0 \\ -2.0 \\ 3.0 \end{bmatrix}$$

$\therefore$ the required straight line is given by,

$$\begin{aligned} y &= a + bx + cx^2 \\ y &= 0.0 - 2.0 \times x + 3.0 \times x^2. \end{aligned}$$

Value of y at x = 2.5 = 13.7500.

Example 1.4

Use the least square method to fit a parabola of the form $y = a + bx + cx^2$ using the data given below and find the value of y at $x = 5.0$.

$x:$	-2.0	-1.0	0.0	1.0	2.0
$y:$	5.0	2.0	1.0	2.0	5.0

Solution

The data points are given by,

$x:$	-2.0	-1.0	0.0	1.0	2.0
$y:$	5.0	2.0	1.0	2.0	5.0

Hence the corresponding normal equations can be obtained using the equations,

$$\begin{aligned} n \times a + \Sigma x \times b + \Sigma x^2 \times b &= \Sigma y \\ \Sigma x \times a + \Sigma x^2 \times b + \Sigma x^3 \times b &= \Sigma xy \\ \Sigma x^2 \times a + \Sigma x^3 \times b + \Sigma x^4 \times b &= \Sigma x^2 y \end{aligned}$$

By substituting the values obtained from the samples into the above normal equations, we have

$$\begin{aligned} 5.0a + 0.0b + 10.0c &= 15.0 \\ 0.0a + 10.0b + 0.0c &= 0.0 \\ 10.0a + 0.0b + 34.0c &= 44.0 \end{aligned}$$

Now we have to solve these equations for a and b .

We will use Gauss elimination method for solving them.

For,

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 5.0 & 0.0 & 10.0 \\ 0.0 & 10.0 & 0.0 \\ 10.0 & 0.0 & 34.0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 15.0 \\ 0.0 \\ 44.0 \end{bmatrix}$$

$$A : B = \begin{bmatrix} 5.0 & 0.0 & 10.0 & 15.0 \\ 0.0 & 10.0 & 0.0 & 0.0 \\ 10.0 & 0.0 & 34.0 & 44.0 \end{bmatrix}$$

By permorming the elementary row operations as indicated below so as to get upper triangular matrix, we have

$$\begin{bmatrix} 5.0000 & 0.0000 & 10.0000 & 15.0000 \\ 0.0000 & 10.0000 & 0.0000 & 0.0000 \\ 10.0000 & 0.0000 & 34.0000 & 44.0000 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{0.0}{5.0} \times R_1.$

$$\begin{bmatrix} 5.0000 & 0.0000 & 10.0000 & 15.0000 \\ 0.0000 & 10.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 14.0000 & 14.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{10.0}{5.0} \times R_1.$

$$\begin{bmatrix} 5.0000 & 0.0000 & 10.0000 & 15.0000 \\ 0.0000 & 10.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 14.0000 & 14.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{0.0}{10.0} \times R_2.$

By back substitution, we get the value of a and b as,

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.0 \\ 1.0 \end{bmatrix}$$

$\therefore$ the required straight line is given by,

$$\begin{aligned} y &= a + bx + cx^2 \\ y &= 1.0 + 0.0 \times x + 1.0 \times x^2. \end{aligned}$$

Value of y at x = 5.0 = 26.

Example 1.5

Use the least square method to fit a parabola of the form $y = a + bx + cx^2$ using the data given below and find the value of y at $x = 3.0$.

$x:$	-2.0	-1.0	0.0	2.0	4.0
$y:$	12.0	7.0	4.0	4.0	12.0

Solution

The data points are given by,

$x:$	-2.0	-1.0	0.0	2.0	4.0
$y:$	12.0	7.0	4.0	4.0	12.0

Hence the corresponding normal equations can be obtained using the equations,

$$\begin{aligned} n \times a + \Sigma x \times b + \Sigma x^2 \times b &= \Sigma y \\ \Sigma x \times a + \Sigma x^2 \times b + \Sigma x^3 \times b &= \Sigma xy \\ \Sigma x^2 \times a + \Sigma x^3 \times b + \Sigma x^4 \times b &= \Sigma x^2 y \end{aligned}$$

By substituting the values obtained from the samples into the above normal equations, we have

$$\begin{aligned} 5.0a + 3.0b + 25.0c &= 39.0 \\ 3.0a + 25.0b + 63.0c &= 25.0 \\ 25.0a + 63.0b + 289.0c &= 263.0 \end{aligned}$$

Now we have to solve these equations for a and b .

We will use Gauss elimination method for solving them.

For,

The system of equations can be written as $AX = b$ as given below.

$$\begin{bmatrix} 5.0 & 3.0 & 25.0 \\ 3.0 & 25.0 & 63.0 \\ 25.0 & 63.0 & 289.0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 39.0 \\ 25.0 \\ 263.0 \end{bmatrix}$$

$$A : B = \begin{bmatrix} 5.0 & 3.0 & 25.0 & 39.0 \\ 3.0 & 25.0 & 63.0 & 25.0 \\ 25.0 & 63.0 & 289.0 & 263.0 \end{bmatrix}$$

By permorming the elementary row operations as indicated below so as to get upper triangular matrix, we have

$$\begin{bmatrix} 5.0000 & 3.0000 & 25.0000 & 39.0000 \\ 0.0000 & 23.2000 & 48.0000 & 1.6000 \\ 25.0000 & 63.0000 & 289.0000 & 263.0000 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{3.0}{5.0} \times R_1.$

$$\begin{bmatrix} 5.0000 & 3.0000 & 25.0000 & 39.0000 \\ 0.0000 & 23.2000 & 48.0000 & 1.6000 \\ 0.0000 & 48.0000 & 164.0000 & 68.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{25.0}{5.0} \times R_1.$

$$\begin{bmatrix} 5.0000 & 3.0000 & 25.0000 & 39.0000 \\ 0.0000 & 23.2000 & 48.0000 & 1.6000 \\ 0.0000 & 0.0000 & 64.6897 & 64.6897 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{48.0}{23.2} \times R_2.$

By back substitution, we get the value of a and b as,

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 4.0 \\ -2.0 \\ 1.0 \end{bmatrix}$$

$\therefore$ the required straight line is given by,

$$\begin{aligned} y &= a + bx + cx^2 \\ y &= 4.0 + -2.0 \times x + 1.0 \times x^2. \end{aligned}$$

Value of y at x = 3.0 = 7.

12.7 Numerical Integration

12.7.1 The Trapezoidal Rule

For employing the trapezoidal rule to evaluate $\int_a^b f(x)dx$, we divide the interval (a, b) into n subintervals of equal width, such that $a = x_0 < x_1 < x_2 < x_3 < ... < x_n = b$ with $x_1 - x_0 = h, x_2 - x_1 = h, x_3 - x_2 = h, ..., x_n - x_{n-1} = h = \frac{b-a}{n}$.

The corresponding Trapezoidal formula is given by;

$$\begin{aligned} \int_a^b f(x)dx &= \frac{h}{2}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + ... + f(x_{n-1})] + f(x_n)\Big] \\ &= \frac{h}{2}\Big[y_0 + 2[y_1 + y_2 + y_3 + ... + y_{n-1}] + y_n\Big] \\ &= \frac{b-a}{2n}\Big[y_0 + 2[y_1 + y_2 + y_3 + ... + y_{n-1}] + y_n\Big]. \end{aligned}$$

12.7.2 Illustrative Example

Example 6.7

Use the trapezoidal rule to numerically integrate $f(x) = 0.2 + 25x$ from $a = 0$ to $b = 2$ by taking $n = 4$.

Solution

We have,

$$\begin{aligned}
\int_a^b f(x)dx &= \int_0^2 (0.2 + 25x)dx \\
\int_a^b f(x)dx &= \frac{h}{2}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + ... + f(x_{n-1})] + f(x_n)\Big] \\
&= \frac{b-a}{2\times 4}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3)] + f(x_4)\Big] \\
&= \frac{2-0}{2\times 4}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3)] + f(x_4)\Big] \\
&= \frac{2-0}{8}\Big[0.20 + 2\times(12.7 + 25.2 + 37.7) + 50.2\Big] \\
&= 0.25 \times 201.6 \\
&= 50.4.
\end{aligned}$$

Example 6.8

Use the trapezoidal rule to numerically integrate $f(x) = \frac{1}{1+x}$ from $a = 0$ to $b = 1$ by taking $n = 4$.

Solution

We have,

$$\begin{aligned}
\int_a^b f(x)dx &= \int_0^1 \frac{1}{1+x}dx \\
\int_a^b f(x)dx &= \frac{h}{2}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + ... + f(x_{n-1})] + f(x_n)\Big] \\
&= \frac{b-a}{2\times 4}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3)] + f(x_4)\Big] \\
&= \frac{1-0}{2\times 4}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3)] + f(x_4)\Big] \\
&= \frac{1}{8}\Big[1 + 2\times(0.8 + 0.6667 + 0.5714) + 0.5\Big] \\
&= \frac{1}{8} \times 5.5762 \\
&= 0.6970.
\end{aligned}$$

Example 6.9

Use the trapezoidal rule to numerically integrate $f(x) = \frac{1}{1+x^2}$ from $a = 0$ to $b = 1$ by taking $n = 5$.

Solution

We have,

$$\begin{aligned}
\int_a^b f(x)dx &= \int_0^1 \frac{1}{1+x^2}dx \\
\int_a^b f(x)dx &= \frac{h}{2}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + ... + f(x_{n-1})] + f(x_n)\Big] \\
&= \frac{b-a}{2\times 5}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + f(x_4)] + f(x_5)\Big] \\
&= \frac{1-0}{2\times 5}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + f(x_4)] + f(x_5)\Big] \\
&= \frac{1}{10}\Big[1 + 2\times(0.9615 + 0.8621 + 0.7353 + 0.6098) + 0.5\Big] \\
&= \frac{1}{10}\times 7.8373 \\
&= 0.7838.
\end{aligned}$$

Example 6.10

Use the trapezoidal rule to numerically integrate $f(x) = sin(x)$ from $a = 0$ to $b = \pi$ by taking $n = 5$.

Solution

We have,

$$\begin{aligned}
\int_a^b f(x)dx &= \int_0^\pi sin(x)dx \\
\int_a^b f(x)dx &= \frac{h}{2}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + ... + f(x_{n-1})] + f(x_n)\Big] \\
&= \frac{b-a}{2\times 5}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + f(x_4)] + f(x_5)\Big] \\
&= \frac{\pi-0}{2\times 5}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + f(x_4)] + f(x_5)\Big] \\
&= \frac{\pi}{10}\Big[0 + 2\times(0.5878 + 0.9511 + 0.9511 + 0.5878) + 0\Big] \\
&= \frac{\pi}{10}\times 6.1554 \\
&= 1.9338.
\end{aligned}$$

Example 6.11

Use the trapezoidal rule to numerically integrate $f(x) = e^x$ from $a = 0$ to $b = 1$ by taking $n = 5$.

Solution

We have,

$$\begin{aligned}
\int_a^b f(x)dx &= \int_0^1 e^x dx \\
\int_a^b f(x)dx &= \frac{h}{2}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + ... + f(x_{n-1})] + f(x_n)\Big] \\
&= \frac{b-a}{2\times 5}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + f(x_4)] + f(x_5)\Big] \\
&= \frac{1-0}{2\times 5}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + f(x_4)] + f(x_5)\Big] \\
&= \frac{1}{10}\Big[1 + 2\times(1.2214 + 1.4918 + 1.8221 + 2.2255) + 2.7183\Big] \\
&= \frac{1}{10}\times 17.24 \\
&= 1.724.
\end{aligned}$$

Example 6.12

Use the trapezoidal rule to numerically integrate $f(x) = log(x)$ from $a = 4$ to $b = 5.2$ by taking $n = 4$.

Solution

We have,

$$\begin{aligned}
\int_a^b f(x)dx &= \int_4^{5.2} log(x)dx \\
\int_a^b f(x)dx &= \frac{h}{2}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3) + ... + f(x_{n-1})] + f(x_n)\Big] \\
&= \frac{b-a}{2\times 4}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3)] + f(x_4)\Big] \\
&= \frac{5.2-4}{2\times 4}\Big[f(x_0) + 2[f(x_1) + f(x_2) + f(x_3)] + f(x_4)\Big] \\
&= \frac{1.2}{8}\Big[1.3863 + 2\times(1.4586 + 1.5261 + 1.5892) + 1.6487\Big] \\
&= \frac{1}{8}\times 12.1828 \\
&= 1.8274.
\end{aligned}$$

12.7.3 Practice Problems

Use the trapezoidal rule to numerically integrate by taking $n = 4$.

1. $\int_1^2 log(x)dx$
2. $\int_2^3 \frac{1}{x-1}dx$
3. $\int_0^1 e^{x^2}dx$
4. $\int_0^{\frac{\pi}{2}} [cos(x) + sin(x)]dx$
5. $\int_1^{\frac{\pi}{4}} tan(x)dx$

12.7.4 The Simpsons $\frac{1}{3}$ Rule

For employing the Simpsons $\frac{1}{3}$ rule to evaluate $\int_a^b f(x)dx$, we divide the interval (a, b) into $2n$ (i.e.,even) number of subintervals having equal width, such that $a = x_0 < x_1 < x_2 < x_3 < ... < x_{2n} = b$ with $x_1 - x_0 = h, x_2 - x_1 = h, x_3 - x_2 = h, ..., x_{2n} - x_{2n-1} = h = \frac{b-a}{2n}$.

The corresponding Simpsons $\frac{1}{3}$ formula is given by;

$$\begin{aligned}\int_a^b f(x)dx &= \frac{h}{3}\Big[f(x_0) + f(x_{2n}) + 4 \times [f(x_1) + f(x_3) + f(x_5) + ... + f(x_{2n-1})] \\ &\qquad + 2 \times [f(x_2) + f(x_4) + f(x_6) + ... + f(x_{2n-2})]\Big] \\ &= \frac{h}{3}\Big[y_0 + y_{2n} + 4 \times (y_1 + y_3 + y_5 + ... + y_{2n-1}) \\ &\qquad + 2 \times (y_2 + y_4 + y_6 + ... + y_{2n-2})\Big]\end{aligned}$$

12.7.5 Illustrative Example

Example 6.13

Use the Simpsons $\frac{1}{3}$ rule to numerically integrate $f(x) = 0.2 + 25x$ from $a = 0$ to $b = 2$ by taking $n = 2$.

Solution

We have,

$$\begin{aligned}\int_a^b f(x)dx &= \int_0^2 (0.2+25x)dx \\ &= \frac{h}{3}\Big[f(x_0)+f(x_{2n})+4\times[f(x_1)+f(x_3)+f(x_5)+...+f(x_{2n-1})] \\ &\qquad\qquad +2\times[f(x_2)+f(x_4)+f(x_6)+...+f(x_{2n-2})]\Big] \\ &= \frac{b-a}{3\times 4}\Big[f(x_0)+f(x_4)+4\times[f(x_1)+f(x_3)]+2\times f(x_2)\Big] \\ &= \frac{2-0}{3\times 4}\Big[f(x_0)+f(x_4)+4\times[f(x_1)+f(x_3)]+2\times f(x_2)\Big] \\ &= \frac{2}{12}\Big[0.20+50.2+4\times(12.7+37.7)+2\times 25.2\Big] \\ &= \frac{1}{6}\times 302.4 \\ &= 50.40.\end{aligned}$$

Example 6.14

Use the Simpsons $\frac{1}{3}$ rule to numerically integrate $f(x)=\frac{1}{1+x}$ from $a=0$ to $b=1$ by taking $n=2$.

Solution

We have,

$$\begin{aligned}\int_a^b f(x)dx &= \int_0^1 \frac{1}{1+x}dx \\ &= \frac{h}{3}\Big[f(x_0)+f(x_{2n})+4\times[f(x_1)+f(x_3)+f(x_5)+...+f(x_{2n-1})] \\ &\qquad\qquad +2\times[f(x_2)+f(x_4)+f(x_6)+...+f(x_{2n-2})]\Big] \\ &= \frac{b-a}{3\times 4}\Big[f(x_0)+f(x_4)+4\times[f(x_1)+f(x_3)]+2\times f(x_2)\Big] \\ &= \frac{1-0}{3\times 4}\Big[f(x_0)+f(x_4)+4\times[f(x_1)+f(x_3)]+2\times f(x_2)\Big] \\ &= \frac{1}{12}\Big[1+0.5+4\times(0.8+0.5714)+2\times 0.6667\Big] \\ &= \frac{1}{12}\times 8.3190 \\ &= 0.6932.\end{aligned}$$

Example 6.15

Use the Simpsons $\frac{1}{3}$ rule to numerically integrate $f(x)=\frac{1}{1+x^2}$ from $a=0$ to $b=1$ by taking $n=3$.

Solution

We have,

$$\begin{aligned}
\int_a^b f(x)dx &= \int_0^1 \frac{1}{1+x^2}dx \\
&= \frac{h}{3}\Big[f(x_0)+f(x_{2n})+4\times[f(x_1)+f(x_3)+f(x_5)+...+f(x_{2n-1})] \\
&\qquad\qquad +2\times[f(x_2)+f(x_4)+f(x_6)+...+f(x_{2n-2})]\Big] \\
&= \frac{b-a}{3\times 6}\Big[f(x_0)+f(x_6)+4\times[f(x_1)+f(x_3)+f(x_5)]+2\times(f(x_2)+f(x_4))\Big] \\
&= \frac{1-0}{3\times 6}\Big[f(x_0)+f(x_6)+4\times[f(x_1)+f(x_3)+f(x_5)]+2\times(f(x_2)+f(x_4))\Big] \\
&= \frac{1}{18}\Big[1+0.5+4\times(0.9730+0.0.8+0.5902)+2\times(0.9+0.6923)\Big] \\
&= \frac{1}{18}\times 14.1372 \\
&= 0.7854.
\end{aligned}$$

Example 6.16

Use the Simpsons $\frac{1}{3}$ rule to numerically integrate $f(x) = sin(x)$ from $a = 0$ to $b = \pi$ by taking $n = 3$.

Solution

We have,

$$\begin{aligned}
\int_a^b f(x)dx &= \int_0^\pi sin(x)dx \\
&= \frac{h}{3}\Big[f(x_0)+f(x_{2n})+4\times[f(x_1)+f(x_3)+f(x_5)+...+f(x_{2n-1})] \\
&\qquad\qquad +2\times[f(x_2)+f(x_4)+f(x_6)+...+f(x_{2n-2})]\Big] \\
&= \frac{b-a}{3\times 6}\Big[f(x_0)+f(x_6)+4\times[f(x_1)+f(x_3)+f(x_5)]+2\times(f(x_2)+f(x_4))\Big] \\
&= \frac{\pi-0}{3\times 6}\Big[f(x_0)+f(x_6)+4\times[f(x_1)+f(x_3)+f(x_5)]+2\times(f(x_2)+f(x_4))\Big] \\
&= \frac{\pi}{18}\Big[0+0+4\times(0.5+1+0.5)+2\times(0.8660+0.8660)\Big] \\
&= \frac{\pi}{18}\times 11.4641 \\
&= 2.0008.
\end{aligned}$$

Example 6.17

Use the Simpsons $\frac{1}{3}$ rule to numerically integrate $f(x) = e^x$ from $a = 0$ to $b = 1$ by taking $n = 3$.

Solution

We have,

$$\begin{aligned}\int_a^b f(x)dx &= \int_0^1 e^x dx \\ &= \frac{h}{3}\Big[f(x_0)+f(x_{2n})+4\times[f(x_1)+f(x_3)+f(x_5)+...+f(x_{2n-1})] \\ &\qquad\qquad +2\times[f(x_2)+f(x_4)+f(x_6)+...+f(x_{2n-2})]\Big] \\ &= \frac{b-a}{3\times 6}\Big[f(x_0)+f(x_6)+4\times[f(x_1)+f(x_3)+f(x_5)]+2\times(f(x_2)+f(x_4))\Big] \\ &= \frac{1-0}{3\times 6}\Big[f(x_0)+f(x_6)+4\times[f(x_1)+f(x_3)+f(x_5)]+2\times(f(x_2)+f(x_4))\Big] \\ &= \frac{1}{18}\Big[1+2.7183+4\times(1.18136+1.6487+2.3010)+2\times(1.3956+1.9477)\Big] \\ &= \frac{1}{18}\times 30.9292 \\ &= 1.7183.\end{aligned}$$

Example 6.18

Use the Simpsons $\frac{1}{3}$ to numerically integrate $f(x) = log(x)$ from $a = 4$ to $b = 5.2$ by taking $n = 3$.

Solution

We have,

$$\begin{aligned}\int_a^b f(x)dx &= \int_4^{5.2} log(x)dx \\ &= \frac{h}{3}\Big[f(x_0)+f(x_{2n})+4\times[f(x_1)+f(x_3)+f(x_5)+...+f(x_{2n-1})] \\ &\qquad\qquad +2\times[f(x_2)+f(x_4)+f(x_6)+...+f(x_{2n-2})]\Big] \\ &= \frac{b-a}{3\times 6}\Big[f(x_0)+f(x_6)+4\times[f(x_1)+f(x_3)+f(x_5)]+2\times(f(x_2)+f(x_4))\Big] \\ &= \frac{5.2-4}{3\times 6}\Big[f(x_0)+f(x_6)+4\times[f(x_1)+f(x_3)+f(x_5)]+2\times(f(x_2)+f(x_4))\Big] \\ &= \frac{1.2}{18}\Big[1.3863+1.6487+4\times(1.4351+1.5261+1.6094)+2\times(1.4816+1.5686)\Big] \\ &= \frac{1.2}{18}\times 27.4177 \\ &= 1.8278.\end{aligned}$$

12.7.6 Practice Problems

Use the Simpsons $\frac{1}{3}$ rule to numerically integrate by taking $n = 3$ (i.e., by dividing into 6 subintervals).

1. $\int_1^2 log(x)dx$
2. $\int_2^3 \frac{1}{x-1}dx$
3. $\int_0^1 e^{x^2}dx$
4. $\int_0^{\frac{\pi}{2}} [cos(x) + sin(x)]dx$
5. $\int_1^{\frac{\pi}{4}} tan(x)dx$

12.7.7 The Guass's Quadrature method for evaluating the integrals

The Guass's Quadrature method for evaluating the integrals of the form $\int_a^b f(x)dx$ is such that the integral be exact for polynomials $f(x)$ of degree as large as possible. In particular we have to find weights w_i and nodes x_i such that $\int_{-1}^1 f(x)dx = \sum_1^n w_i f(x_i)$, known as the quadrature formula, and that involves n unknowns of weights w_i and that of n nodes x_i. Hence it require $2n$ conditions or equations to obtain these unknowns. That is we require the quadrature formula to be exact for $f(x) = x^i,\ \ i = 0, 1, 2, ..., 2n-1$.

Hence we have the system of equations,

$$\sum_{j=1}^{n} w_j x_j^i = \int_{-1}^{1} f(x)dx = \int_{-1}^{1} x^i dx,\ \ i = 0, 1, 2, ...2n-1 \tag{12.125}$$

$$= \begin{cases} \frac{2}{i+1}, & \text{for } i = 0, 2, 4, ..., 2n-2; \\ 0, & \text{for } i = 1, 3, 5, ..., 2n-1. \end{cases} \tag{12.126}$$

From the Guass's Quadrature formula, we can obtain the values for weights and nodes when $n = 1$ and $n = 2$ in the following way.

Case $n = 1$:

Here we require two equations, namely,

$$\sum_{j=1}^{n} w_j x_j^i = \int_{-1}^{1} f(x)dx = \int_{-1}^{1} x^i dx,\ \ i = 0, 1 \tag{12.127}$$

$$w_1 x_1^0 = \int_{-1}^{1} x^0 dx = 2; \text{ for } i = 0 \text{ and} \tag{12.128}$$

(12.129)

$$w_1 x_1^1 = \int_{-1}^{1} x^1 dx = 0; \text{ for } i = 1 \tag{12.130}$$

$$\implies x_1 = 0; \because w_1 = 2 \tag{12.131}$$

$$i.e., \; w_1 = 2, x_1 = 0 \tag{12.132}$$

$$\therefore \int_{-1}^{1} f(x)dx \approx 2 \times f(0). \tag{12.133}$$

Case $n = 2$, the requied four equations are given by,

$$\sum_{j=1}^{n} w_j x_j^i = \int_{-1}^{1} f(x)dx = \int_{-1}^{1} x^i dx; \; i = 0, 1, 2, ..., 2n-1 \tag{12.134}$$

$$\sum_{j=1}^{2} w_j x_j^i = \int_{-1}^{1} x^i dx; \; i = 0, 1, 2, 3 \tag{12.135}$$

$$\implies w_1 + w_2 = \int_{-1}^{1} 1dx = 2; \text{ for } i = 0, \tag{12.136}$$

$$w_1 x_1 + w_2 x_2 = \int_{-1}^{1} x dx = 0; \text{ for } i = 1, \tag{12.137}$$

$$w_1 x_1^2 + w_2 x_2^2 = \int_{-1}^{1} x^2 dx = \frac{2}{3}; \text{ for } i = 2 \text{ and} \tag{12.138}$$

$$w_1 x_1^3 + w_2 x_2^3 = \int_{-1}^{1} x^3 dx = 0; \text{ for } i = 3. \tag{12.139}$$

$$\tag{12.140}$$

That is,

$$w_1 + w_2 = 2 \tag{12.141}$$

$$w_1 x_1 + w_2 x_2 = 0 \tag{12.142}$$

$$w_1 x_1^2 + w_2 x_2^2 = \frac{2}{3} \tag{12.143}$$

$$w_1 x_1^3 + w_2 x_2^3 = 0. \tag{12.144}$$

$$\tag{12.145}$$

Solving the above equations, we get $w_1 = w_2 = 1$, $x_1 = \frac{-1}{\sqrt{3}}$ and $x_2 = \frac{1}{\sqrt{3}}$

$$\int_{-1}^{1} f(x)dx \approx f(\frac{-1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}}). \tag{12.146}$$

Note that for the evaluation of the integral of the from $\int_a^b . \; dx$, we have to transform it into $\int_{-1}^{1} . \; dt$ by taking $x = \frac{b+a+(b-a)t}{2}$.

For consider,

$$x = \frac{b+a+(b-a)t}{2} \tag{12.147}$$

$$\therefore dx = \frac{b-a}{2}dt \tag{12.148}$$

$$\implies \int_a^b f(x)dx = \int_{-1}^{1} f(\frac{b+a+(b-a)t}{2})\frac{b-a}{2}dt \tag{12.149}$$

$$i.e., \int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f\Big(\frac{b+a+(b-a)t}{2}\Big)dt \tag{12.150}$$

$$\therefore \int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big] \tag{12.151}$$

$$\tag{12.152}$$

12.7.8 Illustrative Example

Example 1.1

Use the Guass's Quadrature Formula to $\int_{-1}^{1}(3x^2+2)dx$.

Solution

We have the integral is given by

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1}(3x^2+2)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_{-1}^{1} f(x)dx \approx f(\frac{1}{\sqrt{3}}) + f(\frac{-1}{\sqrt{3}})$$

$$\int_{-1}^{1}(3x^2+2)dx = 3\times(\frac{1}{\sqrt{3}})^2 + 2 + 3\times(\frac{-1}{\sqrt{3}})^2 + 2$$

$$\int_{-1}^{1}(3x^2+2)dx = 6.0$$

Example 1.2

Use the Guass's Quadrature Formula to $\int_{0}^{1}(3x^2+2)dx$.

Solution

We have the integral is given by

$$\int_{0}^{1} f(x)dx = \int_{0}^{1}(3x^2+2)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_0^1 (3x^2+2)dx \approx \frac{1-0}{2}\Big[f\Big(\frac{1+0+\frac{1-0}{\sqrt{3}}}{2}\Big)+f\Big(\frac{1+0-\frac{(1-0)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{1}{2}\Big[f\Big(\frac{1+\frac{1}{\sqrt{3}}}{2}\Big)+f\Big(\frac{1-\frac{1}{\sqrt{3}}}{2}\Big)\Big]$$

$$i.e, \int_0^1 (3x^2+2)dx = \frac{1}{2}\Big[3\times\Big(\frac{1+\frac{1}{\sqrt{3}}}{2}\Big)^2+2+3\times\Big(\frac{1-\frac{1}{\sqrt{3}}}{2}\Big)^2+2\Big]$$

$$\therefore \int_0^1 (3x^2+2)dx = 3.0.$$

Example 1.3

Use the Guass's Quadrature Formula to $\int_1^3 (x^2+5)dx$.

Solution

We have the integral is given by

$$\int_1^3 f(x)dx = \int_1^3 (x^2+5)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_1^3 (x^2+5)dx \approx \frac{3-1}{2}\Big[f\Big(\frac{3+1+\frac{3-1}{\sqrt{3}}}{2}\Big)+f\Big(\frac{3+1-\frac{(3-1)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{2}{2}\Big[f\Big(\frac{4+\frac{2}{\sqrt{3}}}{2}\Big)+f\Big(\frac{4-\frac{2}{\sqrt{3}}}{2}\Big)\Big]$$

$$i.e, \int_1^3 (x^2+5)dx = \Big(\frac{4+\frac{2}{\sqrt{3}}}{2}\Big)^2+5+\Big(\frac{4-\frac{2}{\sqrt{3}}}{2}\Big)^2+5$$

$$\therefore \int_1^3 (x^2+5)dx = 18.6667.$$

Example 1.4

Use the Guass's Quadrature Formula to $\int_{-1}^1 (4x^3+1)dx$.

Solution

We have the integral is given by

$$\int_{-1}^1 f(x)dx = \int_{-1}^1 (4x^3+1)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_{-1}^{1}(4x^3+1)dx \approx \frac{1--1}{2}\Big[f\Big(\frac{1+-1+\frac{1--1}{\sqrt{3}}}{2}\Big)+f\Big(\frac{1+-1-\frac{(1--1)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{2}{2}\Big[f\Big(\frac{0+\frac{2}{\sqrt{3}}}{2}\Big)+f\Big(\frac{0-\frac{2}{\sqrt{3}}}{2}\Big)\Big]$$

$$i.e,\ \int_{-1}^{1}(4x^3+1)dx = 4\times\Big(\frac{0+\frac{2}{\sqrt{3}}}{2}\Big)^3+1+4\times\Big(\frac{0-\frac{2}{\sqrt{3}}}{2}\Big)^3+1$$

$$\therefore \int_{-1}^{1}(4x^3+1)dx = 2.0000.$$

Example 1.5

Use the Guass's Quadrature Formula to $\int_0^{\pi/2}(sin(x))dx$.

Solution

We have the integral is given by

$$\int_0^{\pi/2} f(x)dx = \int_0^{\pi/2}(sin(x))dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_0^{\pi/2}(sin(x))dx \approx \frac{\pi/2-0}{2}\Big[f\Big(\frac{\pi/2+0+\frac{\pi/2-0}{\sqrt{3}}}{2}\Big)+f\Big(\frac{\pi/2+0-\frac{(\pi/2-0)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{\pi/2}{2}\Big[f\Big(\frac{\pi/2+\frac{\pi/2}{\sqrt{3}}}{2}\Big)+f\Big(\frac{\pi/2-\frac{\pi/2}{\sqrt{3}}}{2}\Big)\Big]$$

$$i.e,\ \int_0^{\pi/2}(sin(x))dx = \frac{\pi}{4}\Big[sin\Big(\frac{\pi/2+\frac{\pi/2}{\sqrt{3}}}{2}\Big)+sin\Big(\frac{\pi/2-\frac{\pi/2}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_0^{\pi/2}(sin(x))dx = 1.0.$$

Example 1.6

Use the Guass's Quadrature Formula to $\int_{-1}^{1}(x^2)dx$.

Solution

We have the integral is given by

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1}(x^2)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_{-1}^{1} x^2dx \approx \frac{1+1}{2}\Big[f\Big(\frac{1-1+\frac{1+1}{\sqrt{3}}}{2}\Big)+f\Big(\frac{1-1-\frac{(1+1)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{2}{2}\Big[f\Big(\frac{+\frac{2}{\sqrt{3}}}{2}\Big)+f\Big(\frac{-\frac{2}{\sqrt{3}}}{2}\Big)\Big]$$

$$i.e,\ \int_{-1}^{1}(x^2)dx = \Big[3\times\Big(\frac{+\frac{2}{\sqrt{3}}}{2}\Big)^2+2+3\times\Big(\frac{-\frac{2}{\sqrt{3}}}{2}\Big)^2+2\Big]$$

$$\therefore \int_{-1}^{1}(x^2)dx = 0.67.$$

Example 1.7

Use the Guass's Quadrature Formula to $\int_{-1}^{1}(3*x^2+1)dx$.

Solution

We have the integral is given by

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1}(3*x^2+1)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_{-1}^{1} 3*x^2+1dx \approx \frac{1+1}{2}\Big[f\Big(\frac{1-1+\frac{1+1}{\sqrt{3}}}{2}\Big)+f\Big(\frac{1-1-\frac{(1+1)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{2}{2}\Big[f\Big(\frac{+\frac{2}{\sqrt{3}}}{2}\Big)+f\Big(\frac{-\frac{2}{\sqrt{3}}}{2}\Big)\Big]$$

$$i.e,\ \int_{-1}^{1}(3*x^2+1)dx = -3*\sqrt(3)/3^2+3*\sqrt(3)/3^2+2$$

$$\therefore \int_{-1}^{1}(3*x^2+1)dx = 4.0.$$

Example 1.8

Use the Guass's Quadrature Formula to $\int_{-2}^{2}(3*x^2+1)dx$.

Solution

We have the integral is given by

$$\int_{-2}^{2} f(x)dx = \int_{-2}^{2}(3*x^2+1)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_{-2}^{2} 3*x^2+1dx \approx \frac{2+2}{2}\Big[f\Big(\frac{2-2+\frac{2+2}{\sqrt{3}}}{2}\Big)+f\Big(\frac{2-2-\frac{(2+2)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{4}{2}\Big[f\Big(\frac{+\frac{4}{\sqrt{3}}}{2}\Big)+f\Big(\frac{-\frac{4}{\sqrt{3}}}{2}\Big)\Big]$$

$$i.e,\ \int_{-2}^{2}(3*x^2+1)dx = \frac{4}{2}[(6*\frac{\sqrt{3}}{3})^2+(-6*\frac{\sqrt{3}}{3})^2+2]$$

$$\therefore \int_{-2}^{2}(3*x^2+1)dx = 20.0.$$

Example 1.9

Use the Guass's Quadrature Formula to $\int_{-1}^{1}(2*x^3+3*x^2)dx$.

Solution

We have the integral is given by

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1}(2*x^3+3*x^2)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_{-1}^{1} 2*x^3+3*x^2dx \approx \frac{1+1}{2}\Big[f\Big(\frac{1-1+\frac{1+1}{\sqrt{3}}}{2}\Big)+f\Big(\frac{1-1-\frac{(1+1)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{2}{2}\Big[f\Big(\frac{+\frac{2}{\sqrt{3}}}{2}\Big)+f\Big(\frac{-\frac{2}{\sqrt{3}}}{2}\Big)\Big]$$

$$= 2\Big[(\sqrt{(}3)/3)^3+(-\sqrt{(}3)/3)^3\Big]+3\Big[(\sqrt{(}3)/3)^2+(-\sqrt{(}3)/3)^2\Big]$$

$$\therefore \int_{-1}^{1}(2*x^3+3*x^2)dx = 2.0.$$

Example 1.10

Use the Guass's Quadrature Formula to $\int_{-1}^{1}(2x^3+3x^2+x)dx$.

Solution

We have the integral is given by

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1}(2x^3+3x^2+x)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_{-1}^{1} 2x^3+3x^2+xdx \approx \frac{1+1}{2}\Big[f\Big(\frac{1-1+\frac{1+1}{\sqrt{3}}}{2}\Big)+f\Big(\frac{1-1-\frac{(1+1)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{2}{2}\Big[f\Big(\frac{+\frac{2}{\sqrt{3}}}{2}\Big)+f\Big(\frac{-\frac{2}{\sqrt{3}}}{2}\Big)\Big]$$

$$= 2\Big[(\sqrt{3}/3)^3(-\sqrt{3}/3)^3\Big]+3\Big[(\sqrt{3}/3)^2(-sqrt3/3)^2\Big]+1\Big[(\sqrt{3}/3)^1(-\sqrt{3}/3)^1\Big]$$

$$\therefore \int_{-1}^{1}(2x^3+3x^2+x)dx = 2.0.$$

Example 1.11

Use the Guass's Quadrature Formula to $\int_{-2}^{2}(3*x^2+x)dx$.

Solution

We have the integral is given by

$$\int_{-2}^{2} f(x)dx = \int_{-2}^{2}(3*x^2+x)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_{-2}^{2}(3*x^2+x)dx \approx \frac{2+2}{2}\Big[f\Big(\frac{2-2+\frac{2+2}{\sqrt{3}}}{2}\Big)+f\Big(\frac{2-2-\frac{(2+2)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{4}{2}\Big[f\Big(\frac{+\frac{4}{\sqrt{3}}}{2}\Big)+f\Big(\frac{-\frac{4}{\sqrt{3}}}{2}\Big)\Big]$$

$$i.e,\ \int_{-2}^{2}(3*x^2+x)dx = \frac{4}{2}\times\Big\{3\Big[(2*\sqrt{3}/3)^2+(-2*\sqrt{3}/3)^2\Big]+\Big[(2*\sqrt{3}/3)+(-2*\sqrt{3}/3)\Big]\Big\}$$

$$\therefore \int_{-2}^{2}(3*x^2+x)dx = 16.0.$$

Example 1.12

Use the Guass's Quadrature Formula to $\int_{-2}^{2}(3*x^2-x)dx$.

Solution

We have the integral is given by

$$\int_{-2}^{2} f(x)dx = \int_{-2}^{2}(3*x^2-x)dx.$$

Applying the Guass's Quadrature formula for 2 points, we get

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f\Big(\frac{b+a+\frac{b-a}{\sqrt{3}}}{2}\Big)+f\Big(\frac{b+a-\frac{(b-a)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\therefore \int_{-2}^{2}(3x^2-x)dx \approx \frac{2+2}{2}\Big[f\Big(\frac{2-2+\frac{2+2}{\sqrt{3}}}{2}\Big)+f\Big(\frac{2-2-\frac{(2+2)}{\sqrt{3}}}{2}\Big)\Big]$$

$$\approx \frac{4}{2}\Big[f\Big(\frac{+\frac{4}{\sqrt{3}}}{2}\Big)+f\Big(\frac{-\frac{4}{\sqrt{3}}}{2}\Big)\Big]$$

$$i.e,\ \int_{-2}^{2}(3x^2-x)dx = \frac{4}{2}imes\Big\{3\Big[(2*\sqrt{3}/3)^2+(-2*\sqrt{3}/3)^2\Big]+\Big[(2*\sqrt{3}/3)+(-2*\sqrt{3}/3)\Big]\Big\}$$

$$\therefore \int_{-2}^{2}(3x^2-x)dx = 16.0.$$

12.8 The solution of Laplace's equation

12.8.1 The solution of Laplace's equation for two dimensions using finite difference method

T The Laplace's equation in two dimensions is of the form $\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$. From the finite difference the first and second derivatives are respectively given by

$$\frac{\partial u}{\partial x} = \frac{u(x+h,y)-u(x,y)}{h}$$
$$\frac{\partial u}{\partial y} = +\frac{u(x,y+k)-u(x,y)}{k}$$
$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+h,y)-2u(x,y)+u(x-h,y)}{h^2}$$
$$\frac{\partial^2 u}{\partial y^2} = \frac{u(x,y+k)-2u(x,y)+u(x,y-k)}{k^2}$$

Hence, the Laplace's equation can be written as,

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2} = \frac{u(x+h,y)-2u(x,y)+u(x-h,y)}{h^2}+\frac{u(x,y+k)-2u(x,y)+u(x,y-k)}{k^2}=0$$

$$u(x,y) = \frac{u(x,y-k)+u(x+h,y)+u(x,y+k)+u(x-h,y)}{4}$$

The step lengths $h=x_{i+1}-x_i,\ \ k=y_{j+1}-y_j\ \ \forall i=0,1,2,...,;j=0,1,2,...,$ or equivalently, $x_i=x_0+ih, y_j=y_0+jk\ \ \forall i=1,2,...,;j=1,2,...,$.

Hence by taking the node (x_0, y_0) as origin $(0, 0)$, we have any node (x, y) on the grid is $= (ih, jk)$. That gives $(x, y) = (i, j)$, if $h = k = 1$.

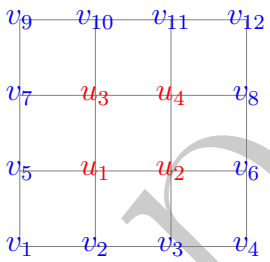
Correspondingly the Laplace's equation can be re written as

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = 0$$

$$u_{i,j} = \frac{u_{i,j-1} + u_{i+1,j} + u_{i,j+1} + u_{i-1,j}}{4}.$$

The equation $u_{i,j} = \frac{1}{4}[u_{i,j-1} + u_{i+1,j} + u_{i,j+1} + u_{i-1,j}]$ is known as the Standard Five Point Formula(SFPF).

We require the node values u_1, u_2, u_3 and u_4 , when the values $v_1, v_2, ..., v_{12}$ are provided as given below.



Since the solution of the Laplace's equation at the nodes (i, j) is given by $u_{i,j}$, the internal node values u_1, u_2, u_3 and u_4 can be computed when the values $v_1, v_2, ..., v_{12}$ are provided, using the standard five point formula.

By employing the standard five point formula with the known grid values, now we can find the unknown values at interior nodes of the mesh using the following equations.

$$u(2,1) = \frac{u(2,0) + u(3,1) + u(2,2) + u(1,1)}{4}$$

$$u(2,2) = \frac{u(2,1) + u(3,2) + u(2,3) + u(1,2)}{4}$$

$$u(1,1) = \frac{u(1,0) + u(2,1) + u(1,2) + u(0,1)}{4}$$

$$u(1,2) = \frac{u(1,1) + u(2,2) + u(1,3) + u(0,2)}{4}$$

Rearranging the above equations,

$$-4 \times u(2,1) + u(2,2) + u(1,1) = u(2,0) + u(3,1)$$
$$-4 \times u(2,2) + u(2,1) + u(1,2) = u(3,2) + u(2,3)$$
$$-4 \times u(1,1) + u(2,1) + u(1,2) = u(1,0) + u(0,1)$$
$$-4 \times u(1,2) + u(1,1) + u(2,2) = u(1,3) + u(0,2)$$

Now by substituting node values in the above equations, we have,

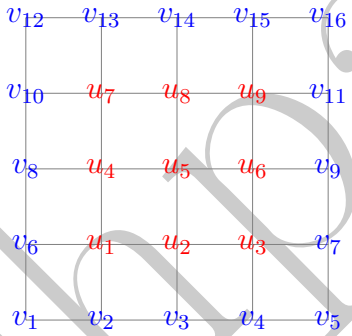
$$\begin{aligned}
-4 \times u_1 + u_2 + u_3 + 0 \times u_4 &= v_5 + v_2 \\
u_1 - 4 \times u_2 + 0 \times u_3 + u_4 &= v_3 + v_6 \\
u_1 + 0 \times u_2 - 4 \times u_3 + u_4 &= v_7 + v_{10} \\
0 \times u_1 + u_2 + u_3 - 4 \times u_4 &= v_8 + v_{11}.
\end{aligned}$$

Solving the above equations for u_1, u_2, u_3 and u_4 completes the process.

Another method for solving the Laplace's equation is using the Diagonal Five Point Formula(DFPF) which is detailed below.

$$u_{i,j} = \frac{u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1}}{4}.$$

For example, consider the following mesh.



We can use DFPF to compute the grid values $u_{1,1} = u_1, u_{1,3} = u_3, u_{3,1} = u_7$ and $u_{3,3} = u_9$, once the grid value $u_{2,2} = u_5$ is known and which can be found out using SFPF as shown below.

$$u_{2,2} = \frac{u_{2,0} + u_{4,2} + u_{2,4} + u_{0,2}}{4} = u_5.$$

and

$$\begin{aligned}
u_{1,1} &= \frac{u_{0,0} + u_{0,2} + u_{2,0} + u_{2,2}}{4} = u_1 \\
u_{1,3} &= \frac{u_{0,2} + u_{0,4} + u_{2,2} + u_{2,4}}{4} = u_3 \\
u_{3,1} &= \frac{u_{2,0} + u_{2,2} + u_{4,0} + u_{4,2}}{4} = u_7 \\
u_{3,3} &= \frac{u_{2,2} + u_{2,4} + u_{4,2} + u_{4,4}}{4} = u_9.
\end{aligned}$$

Finally we can employ the following SFPF to find the remaining grid values $u_{1,2} = u_2, u_{2,1} = u_4, u_{2,3} = u_6$ and $u_{3,2} = u_8$. Hence according to the availabilty of the grid values we will use

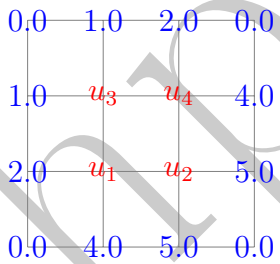
the SFPF and/ or DFPF for solving the Laplace's equation.

$$
\begin{aligned}
u_{1,2} &= \frac{u_{1,1} + u_{2,2} + u_{1,3} + u_{0,2}}{4} = u_2 \\
u_{2,1} &= \frac{u_{2,0} + u_{3,1} + u_{2,2} + u_{1,1}}{4} = u_4 \\
u_{2,3} &= \frac{u_{2,2} + u_{3,3} + u_{2,4} + u_{1,3}}{4} = u_6 \\
u_{3,2} &= \frac{u_{3,1} + u_{4,2} + u_{3,3} + u_{2,2}}{4} = u_8.
\end{aligned}
$$

12.8.2 Illustrative Example

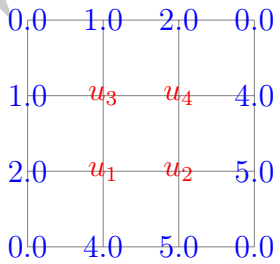
Example 1.1

Use the standard five point formula to solve the Laplace's Equation of the form $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ from the grid points given below.



Solution

We have the grid points are given by



and the standard five point formula $u_{i,j} = \frac{1}{4}[u_{i,j-1} + u_{i+1,j} + u_{i,j+1} + u_{i-1,j}]$ yields the following system of equations.

$$
\begin{aligned}
-4 \times u_1 + u_2 + u_3 + 0 \times u_4 &= v_5 + v_2 \\
u_1 - 4 \times u_2 + 0 \times u_3 + u_4 &= v_3 + v_6 \\
u_1 + 0 \times u_2 - 4 \times u_3 + u_4 &= v_7 + v_{10} \\
0 \times u_1 + u_2 + u_3 - 4 \times u_4 &= v_8 + v_{11}.
\end{aligned}
$$

Now by substituting the grid values in above equations, we get

$$\begin{aligned}
-4.0u_1 + 1.0u_2 + 1.0u_3 + 0.0u_4 &= -6.0 \\
1.0u_1 + -4.0u_2 + 0.0u_3 + 1.0u_4 &= -10.0 \\
1.0u_1 + 0.0u_2 + -4.0u_3 + 1.0u_4 &= -2.0 \\
0.0u_1 + 1.0u_2 + 1.0u_3 + -4.0u_4 &= -6.0
\end{aligned}$$

Finally to solve the above system of equations, we will represent in the form $AX = b$ as given below.

$$\begin{bmatrix} -4.0 & 1.0 & 1.0 & 0.0 \\ 1.0 & -4.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & -4.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & -4.0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} -6.0 \\ -10.0 \\ -2.0 \\ -6.0 \end{bmatrix}$$

And use the Gauss Elimination Method for solving them.

For,

First construct the augmented matrix A:B as given below,

$$A : B = \begin{bmatrix} -4.0 & 1.0 & 1.0 & 0.0 & -6.0 \\ 1.0 & -4.0 & 0.0 & 1.0 & -10.0 \\ 1.0 & 0.0 & -4.0 & 1.0 & -2.0 \\ 0.0 & 1.0 & 1.0 & -4.0 & -6.0 \end{bmatrix}$$

By employing the relevent elementary row operations as indicated below, convert the matrix A:B into upper triangular matrix.

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -6.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -11.5000 \\ 1.0000 & 0.0000 & -4.0000 & 1.0000 & -2.0000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -6.0000 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{1.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -6.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -11.5000 \\ 0.0000 & 0.2500 & -3.7500 & 1.0000 & -3.5000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -6.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{1.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -6.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -11.5000 \\ 0.0000 & 0.2500 & -3.7500 & 1.0000 & -3.5000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -6.0000 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{0.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -6.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -11.5000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -4.2667 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -6.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{0.25}{-3.75} \times R_2.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -6.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -11.5000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -4.2667 \\ 0.0000 & 0.0000 & 1.0667 & -3.7333 & -9.0667 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{1.00}{-3.75} \times R_2.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -6.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -11.5000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -4.2667 \\ 0.0000 & 0.0000 & 0.0000 & -3.4286 & -10.2857 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{1.07}{-3.73} \times R_3.$

After performing the back substitution, we get the solution as,

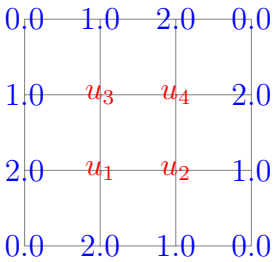
$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 3.0000 \\ 4.0000 \\ 2.0000 \\ 3.0000 \end{bmatrix}$$

$\therefore$ solutions are

$$\begin{aligned} u_1 &= 3.0000; u_2 = 4.0000; \\ u_3 &= 2.0000; u_4 = 3.0000. \end{aligned}$$

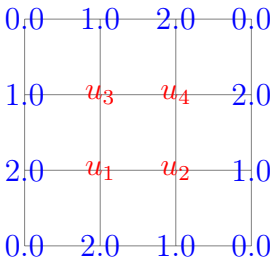
Example 1.2

Use the standard five point formula to solve the Laplace's Equation of the form $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ from the grid points given below.



Solution

We have the grid points are given by



and the standard five point formula $u_{i,j} = \frac{1}{4}[u_{i,j-1}+u_{i+1,j}+u_{i,j+1}+u_{i-1,j}]$ yields the following system of equations.

$$\begin{aligned}
-4 \times u_1 + u_2 + u_3 + 0 \times u_4 &= v_5 + v_2 \\
u_1 - 4 \times u_2 + 0 \times u_3 + u_4 &= v_3 + v_6 \\
u_1 + 0 \times u_2 - 4 \times u_3 + u_4 &= v_7 + v_{10} \\
0 \times u_1 + u_2 + u_3 - 4 \times u_4 &= v_8 + v_{11}.
\end{aligned}$$

Now by substituting the grid values in above equations, we get

$$\begin{aligned}
-4.0u_1 + 1.0u_2 + 1.0u_3 + 0.0u_4 &= -4.0 \\
1.0u_1 + -4.0u_2 + 0.0u_3 + 1.0u_4 &= -2.0 \\
1.0u_1 + 0.0u_2 + -4.0u_3 + 1.0u_4 &= -2.0 \\
0.0u_1 + 1.0u_2 + 1.0u_3 + -4.0u_4 &= -4.0
\end{aligned}$$

Finally to solve the above system of equations, we will represent in the form $AX = b$ as given below.

$$\begin{bmatrix} -4.0 & 1.0 & 1.0 & 0.0 \\ 1.0 & -4.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & -4.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & -4.0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} -4.0 \\ -2.0 \\ -2.0 \\ -4.0 \end{bmatrix}$$

And use the Gauss Elimination Method for solving them.

For,

First construct the augmented matrix A:B as given below,

$$A : B = \begin{bmatrix} -4.0 & 1.0 & 1.0 & 0.0 & -4.0 \\ 1.0 & -4.0 & 0.0 & 1.0 & -2.0 \\ 1.0 & 0.0 & -4.0 & 1.0 & -2.0 \\ 0.0 & 1.0 & 1.0 & -4.0 & -4.0 \end{bmatrix}$$

By employing the relevent elementary row operations as indicated below, convert the matrix A:B into upper triangular matrix.

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -4.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -3.0000 \\ 1.0000 & 0.0000 & -4.0000 & 1.0000 & -2.0000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -4.0000 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{1.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -4.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -3.0000 \\ 0.0000 & 0.2500 & -3.7500 & 1.0000 & -3.0000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -4.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{1.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -4.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -3.0000 \\ 0.0000 & 0.2500 & -3.7500 & 1.0000 & -3.0000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -4.0000 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{0.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -4.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -3.0000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -3.2000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -4.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{0.25}{-3.75} \times R_2.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -4.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -3.0000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -3.2000 \\ 0.0000 & 0.0000 & 1.0667 & -3.7333 & -4.8000 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{1.00}{-3.75} \times R_2.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -4.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -3.0000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -3.2000 \\ 0.0000 & 0.0000 & 0.0000 & -3.4286 & -5.7143 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{1.07}{-3.73} \times R_3.$

After performing the back substitution, we get the solution as,

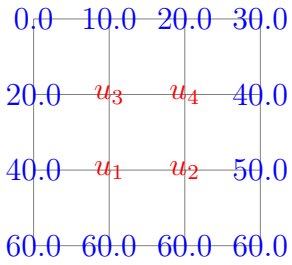
$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1.6667 \\ 1.3333 \\ 1.3333 \\ 1.6667 \end{bmatrix}$$

$\therefore$ solutions are

$$\begin{aligned} u_1 &= 1.6667; u_2 &= 1.3333; \\ u_3 &= 1.3333; u_4 &= 1.6667. \end{aligned}$$

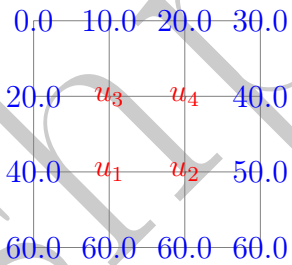
Example 1.3

Use the standard five point formula to solve the Laplace's Equation of the form $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ from the grid points given below.



Solution

We have the grid points are given by



and the standard five point formula $u_{i,j} = \frac{1}{4}[u_{i,j-1} + u_{i+1,j} + u_{i,j+1} + u_{i-1,j}]$ yields the following system of equations.

$$\begin{aligned} -4 \times u_1 + u_2 + u_3 + 0 \times u_4 &= v_5 + v_2 \\ u_1 - 4 \times u_2 + 0 \times u_3 + u_4 &= v_3 + v_6 \\ u_1 + 0 \times u_2 - 4 \times u_3 + u_4 &= v_7 + v_{10} \\ 0 \times u_1 + u_2 + u_3 - 4 \times u_4 &= v_8 + v_{11}. \end{aligned}$$

Now by substituting the grid values in above equations, we get

$$\begin{aligned} -4.0u_1 + 1.0u_2 + 1.0u_3 + 0.0u_4 &= -100.0 \\ 1.0u_1 + -4.0u_2 + 0.0u_3 + 1.0u_4 &= -110.0 \\ 1.0u_1 + 0.0u_2 + -4.0u_3 + 1.0u_4 &= -30.0 \\ 0.0u_1 + 1.0u_2 + 1.0u_3 + -4.0u_4 &= -60.0 \end{aligned}$$

Finally to solve the above system of equations, we will represent in the form $AX = b$ as given below.

$$\begin{bmatrix} -4.0 & 1.0 & 1.0 & 0.0 \\ 1.0 & -4.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & -4.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & -4.0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} -100.0 \\ -110.0 \\ -30.0 \\ -60.0 \end{bmatrix}$$

And use the Gauss Elimination Method for solving them.

For,

First construct the augmented matrix A:B as given below,

$$A : B = \begin{bmatrix} -4.0 & 1.0 & 1.0 & 0.0 & -100.0 \\ 1.0 & -4.0 & 0.0 & 1.0 & -110.0 \\ 1.0 & 0.0 & -4.0 & 1.0 & -30.0 \\ 0.0 & 1.0 & 1.0 & -4.0 & -60.0 \end{bmatrix}$$

By employing the relevent elementary row operations as indicated below, convert the matrix A:B into upper triangular matrix.

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -100.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -135.0000 \\ 1.0000 & 0.0000 & -4.0000 & 1.0000 & -30.0000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -60.0000 \end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{1.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -100.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -135.0000 \\ 0.0000 & 0.2500 & -3.7500 & 1.0000 & -55.0000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -60.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{1.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -100.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -135.0000 \\ 0.0000 & 0.2500 & -3.7500 & 1.0000 & -55.0000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -60.0000 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{0.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -100.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -135.0000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -64.0000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -60.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{0.25}{-3.75} \times R_2.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -100.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -135.0000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -64.0000 \\ 0.0000 & 0.0000 & 1.0667 & -3.7333 & -96.0000 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{1.00}{-3.75} \times R_2.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -100.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -135.0000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -64.0000 \\ 0.0000 & 0.0000 & 0.0000 & -3.4286 & -114.2857 \end{bmatrix}$$

because enspace $\because \quad R_4 \leftarrow R_4 - \frac{1.07}{-3.73} \times R_3.$

After performing the back substitution, we get the solution as,

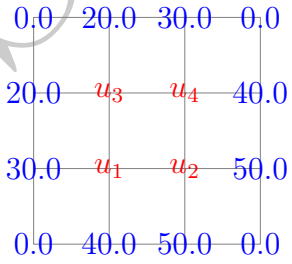
$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 43.3333 \\ 46.6667 \\ 26.6667 \\ 33.3333 \end{bmatrix}$$

$\therefore$ solutions are

$$\begin{aligned} u_1 &= 43.3333; u_2 = 46.6667; \\ u_3 &= 26.6667; u_4 = 33.3333. \end{aligned}$$

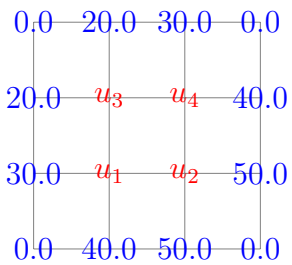
Example 1.4

Use the standard five point formula to solve the Laplace's Equation of the form $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ from the grid points given below.



Solution

We have the grid points are given by



and the standard five point formula $u_{i,j} = \frac{1}{4}[u_{i,j-1}+u_{i+1,j}+u_{i,j+1}+u_{i-1,j}]$ yields the following system of equations.

$$\begin{aligned}
-4 \times u_1 + u_2 + u_3 + 0 \times u_4 &= v_5 + v_2 \\
u_1 - 4 \times u_2 + 0 \times u_3 + u_4 &= v_3 + v_6 \\
u_1 + 0 \times u_2 - 4 \times u_3 + u_4 &= v_7 + v_{10} \\
0 \times u_1 + u_2 + u_3 - 4 \times u_4 &= v_8 + v_{11}.
\end{aligned}$$

Now by substituting the grid values in above equations, we get

$$\begin{aligned}
-4.0u_1 + 1.0u_2 + 1.0u_3 + 0.0u_4 &= -70.0 \\
1.0u_1 + -4.0u_2 + 0.0u_3 + 1.0u_4 &= -100.0 \\
1.0u_1 + 0.0u_2 + -4.0u_3 + 1.0u_4 &= -40.0 \\
0.0u_1 + 1.0u_2 + 1.0u_3 + -4.0u_4 &= -70.0
\end{aligned}$$

Finally to solve the above system of equations, we will represent in the form $AX = b$ as given below.

$$\begin{bmatrix} -4.0 & 1.0 & 1.0 & 0.0 \\ 1.0 & -4.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & -4.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & -4.0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} -70.0 \\ -100.0 \\ -40.0 \\ -70.0 \end{bmatrix}$$

And use the Gauss Elimination Method for solving them.

For,

First construct the augmented matrix A:B as given below,

$$A : B = \begin{bmatrix} -4.0 & 1.0 & 1.0 & 0.0 & -70.0 \\ 1.0 & -4.0 & 0.0 & 1.0 & -100.0 \\ 1.0 & 0.0 & -4.0 & 1.0 & -40.0 \\ 0.0 & 1.0 & 1.0 & -4.0 & -70.0 \end{bmatrix}$$

By employing the relevent elementary row operations as indicated below, convert the matrix A:B into upper triangular matrix.

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -70.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -117.5000 \\ 1.0000 & 0.0000 & -4.0000 & 1.0000 & -40.0000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -70.0000 \end{bmatrix}$$

$\because \; R_2 \leftarrow R_2 - \frac{1.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -70.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -117.5000 \\ 0.0000 & 0.2500 & -3.7500 & 1.0000 & -57.5000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -70.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{1.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -70.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -117.5000 \\ 0.0000 & 0.2500 & -3.7500 & 1.0000 & -57.5000 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -70.0000 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{0.00}{-4.00} \times R_1.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -70.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -117.5000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -65.3333 \\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -70.0000 \end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{0.25}{-3.75} \times R_2.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -70.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -117.5000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -65.3333 \\ 0.0000 & 0.0000 & 1.0667 & -3.7333 & -101.3333 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{1.00}{-3.75} \times R_2.$

$$\begin{bmatrix} -4.0000 & 1.0000 & 1.0000 & 0.0000 & -70.0000 \\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -117.5000 \\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -65.3333 \\ 0.0000 & 0.0000 & 0.0000 & -3.4286 & -120.0000 \end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{1.07}{-3.73} \times R_3.$

After performing the back substitution, we get the solution as,

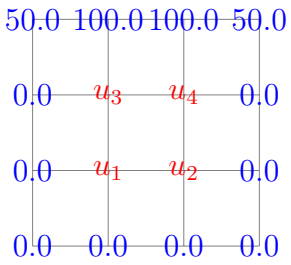
$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 35.0000 \\ 42.5000 \\ 27.5000 \\ 35.0000 \end{bmatrix}$$

$\therefore$ solutions are

$$\begin{aligned} u_1 &= 35.0000; u_2 = 42.5000; \\ u_3 &= 27.5000; u_4 = 35.0000. \end{aligned}$$

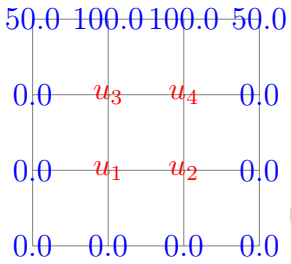
Example 1.5

Use the standard five point formula to solve the Laplace's Equation of the form $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ from the grid points given below.



Solution

We have the grid points are given by



and the standard five point formula $u_{i,j} = \frac{1}{4}[u_{i,j-1} + u_{i+1,j} + u_{i,j+1} + u_{i-1,j}]$ yields the following system of equations.

$$\begin{aligned} -4 \times u_1 + u_2 + u_3 + 0 \times u_4 &= v_5 + v_2 \\ u_1 - 4 \times u_2 + 0 \times u_3 + u_4 &= v_3 + v_6 \\ u_1 + 0 \times u_2 - 4 \times u_3 + u_4 &= v_7 + v_{10} \\ 0 \times u_1 + u_2 + u_3 - 4 \times u_4 &= v_8 + v_{11}. \end{aligned}$$

Now by substituting the grid values in above equations, we get

$$\begin{aligned} -4.0u_1 + 1.0u_2 + 1.0u_3 + 0.0u_4 &= -0.0 \\ 1.0u_1 + -4.0u_2 + 0.0u_3 + 1.0u_4 &= -0.0 \\ 1.0u_1 + 0.0u_2 + -4.0u_3 + 1.0u_4 &= -100.0 \\ 0.0u_1 + 1.0u_2 + 1.0u_3 + -4.0u_4 &= -100.0 \end{aligned}$$

Finally to solve the above system of equations, we will represent in the form $AX = b$ as given below.

$$\begin{bmatrix} -4.0 & 1.0 & 1.0 & 0.0 \\ 1.0 & -4.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & -4.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & -4.0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} -0.0 \\ -0.0 \\ -100.0 \\ -100.0 \end{bmatrix}$$

And use the Gauss Elimination Method for solving them.

For,

First construct the augmented matrix A:B as given below,

$$A:B=\begin{bmatrix}-4.0 & 1.0 & 1.0 & 0.0 & -0.0\\ 1.0 & -4.0 & 0.0 & 1.0 & -0.0\\ 1.0 & 0.0 & -4.0 & 1.0 & -100.0\\ 0.0 & 1.0 & 1.0 & -4.0 & -100.0\end{bmatrix}$$

By employing the relevent elementary row operations as indicated below, convert the matrix A:B into upper triangular matrix.

$$\begin{bmatrix}-4.0000 & 1.0000 & 1.0000 & 0.0000 & -0.0000\\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -0.0000\\ 1.0000 & 0.0000 & -4.0000 & 1.0000 & -100.0000\\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -100.0000\end{bmatrix}$$

$\because \quad R_2 \leftarrow R_2 - \frac{1.00}{-4.00} \times R_1.$

$$\begin{bmatrix}-4.0000 & 1.0000 & 1.0000 & 0.0000 & -0.0000\\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -0.0000\\ 0.0000 & 0.2500 & -3.7500 & 1.0000 & -100.0000\\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -100.0000\end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{1.00}{-4.00} \times R_1.$

$$\begin{bmatrix}-4.0000 & 1.0000 & 1.0000 & 0.0000 & -0.0000\\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -0.0000\\ 0.0000 & 0.2500 & -3.7500 & 1.0000 & -100.0000\\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -100.0000\end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{0.00}{-4.00} \times R_1.$

$$\begin{bmatrix}-4.0000 & 1.0000 & 1.0000 & 0.0000 & -0.0000\\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -0.0000\\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -100.0000\\ 0.0000 & 1.0000 & 1.0000 & -4.0000 & -100.0000\end{bmatrix}$$

$\because \quad R_3 \leftarrow R_3 - \frac{0.25}{-3.75} \times R_2.$

$$\begin{bmatrix}-4.0000 & 1.0000 & 1.0000 & 0.0000 & -0.0000\\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -0.0000\\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -100.0000\\ 0.0000 & 0.0000 & 1.0667 & -3.7333 & -100.0000\end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{1.00}{-3.75} \times R_2.$

$$\begin{bmatrix}-4.0000 & 1.0000 & 1.0000 & 0.0000 & -0.0000\\ 0.0000 & -3.7500 & 0.2500 & 1.0000 & -0.0000\\ 0.0000 & 0.0000 & -3.7333 & 1.0667 & -100.0000\\ 0.0000 & 0.0000 & 0.0000 & -3.4286 & -128.5714\end{bmatrix}$$

$\because \quad R_4 \leftarrow R_4 - \frac{1.07}{-3.73} \times R_3.$

After performing the back substitution, we get the solution as,

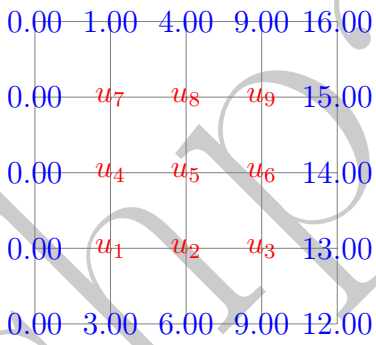
$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 12.5000 \\ 12.5000 \\ 37.5000 \\ 37.5000 \end{bmatrix}$$

$\therefore$ solutions are

$$\begin{aligned} u_1 &= 12.5000; u_2 &= 12.5000; \\ u_3 &= 37.5000; u_4 &= 37.5000. \end{aligned}$$

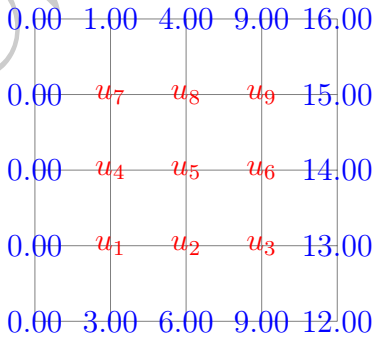
Example 1.6

Solve the Laplace's Equation of the form $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ using the grid points given below.



Solution

We have the grid points are given by



From the grid points it is clear that,the unknown points are given by the coordinate points as given below.

$$\begin{aligned} u_{1,1} = u_1, \quad u_{1,2} = u_2, \quad u_{1,3} = u_3 \\ u_{2,1} = u_4, \quad u_{2,2} = u_5, \quad u_{2,3} = u_6 \\ u_{3,1} = u_7, \quad u_{3,2} = u_8, \quad u_{3,3} = u_9 \end{aligned}$$

Now we will compute the grid value at $u_5 = u_{i,j}$ using the SFPF, so we have,

$$\begin{aligned} u_{2,2} &= \frac{u_{2,0} + u_{4,2} + u_{2,4} + u_{0,2}}{4} \\ &= \frac{0.00 + 6.00 + 14.00 + 4.00}{4} = 6.00 \end{aligned}$$

Using this grid value u_5 , now we can compute u_1, u_3, u_7 and u_9 , using the DFPF as below.

$$\begin{aligned} u_{1,1} &= \frac{u_{0,0} + u_{0,2} + u_{2,0} + u_{2,2}}{4} \\ &= \frac{0.00 + 4.00 + 0.00 + 6.00}{4} = 2.50 \\ u_{1,3} &= \frac{u_{0,2} + u_{0,4} + u_{2,2} + u_{2,4}}{4} \\ &= \frac{4.00 + 16.00 + 6.00 + 14.00}{4} = 10.00 \\ u_{3,1} &= \frac{u_{2,0} + u_{2,2} + u_{4,0} + u_{4,2}}{4} \\ &= \frac{0.00 + 6.00 + 0.00 + 6.00}{4} = 3.00 \\ u_{3,3} &= \frac{u_{2,2} + u_{2,4} + u_{4,2} + u_{4,4}}{4} \\ &= \frac{6.00 + 14.00 + 6.00 + 12.00}{4} = 9.50 \end{aligned}$$

To compute the remaining unknown grid values, now we can employ SFPF. Hence we get,

$$\begin{aligned} u_{1,2} &= \frac{u_{1,1} + u_{2,2} + u_{1,3} + u_{0,2}}{4} \\ &= \frac{2.50 + 6.00 + 10.00 + 4.00}{4} = 5.62 \\ u_{2,1} &= \frac{u_{2,0} + u_{3,1} + u_{2,2} + u_{1,1}}{4} \\ &= \frac{0.00 + 3.00 + 6.00 + 2.50}{4} = 2.88 \\ u_{2,3} &= \frac{u_{2,2} + u_{3,3} + u_{2,4} + u_{1,3}}{4} \\ &= \frac{6.00 + 9.50 + 14.00 + 10.00}{4} = 9.88 \\ u_{3,2} &= \frac{u_{3,1} + u_{4,2} + u_{3,3} + u_{2,2}}{4} \\ &= \frac{3.00 + 6.00 + 9.50 + 6.00}{4} = 6.12 \end{aligned}$$

Hence the solution is given by,

0.00	1.00	4.00	9.00	16.00
0.00	2.50	5.62	10.00	15.00
0.00	2.88	6.00	9.88	14.00
0.00	3.00	6.12	9.50	13.00
0.00	3.00	6.00	9.00	12.00

Example 1.7

Solve the Laplace's Equation of the form $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ using the grid points given below.

0	11.1	17	19.7	18.6
0	u_7	u_8	u_9	21.9
0	u_4	u_5	u_6	21
0	u_1	u_2	u_3	17
0	8.7	12.1	12.8	9

Solution

We have the grid points are given by

0	11.1	17	19.7	18.6
0	u_7	u_8	u_9	21.9
0	u_4	u_5	u_6	21
0	u_1	u_2	u_3	17
0	8.7	12.1	12.8	9

From the grid points it is clear that,the unknown points are given by the coordinate points as given below.

$$\begin{aligned} u_{1,1} &= u_1, \quad u_{1,2} = u_2, \quad u_{1,3} = u_3 \\ u_{2,1} &= u_4, \quad u_{2,2} = u_5, \quad u_{2,3} = u_6 \\ u_{3,1} &= u_7, \quad u_{3,2} = u_8, \quad u_{3,3} = u_9 \end{aligned}$$

Now we will compute the grid value at $u_5 = u_{i,j}$ using the SFPF, so we have,

$$\begin{aligned} u_{2,2} &= \frac{u_{2,0} + u_{4,2} + u_{2,4} + u_{0,2}}{4} \\ &= \frac{0.00 + 12.10 + 21.00 + 17.00}{4} = 12.52 \end{aligned}$$

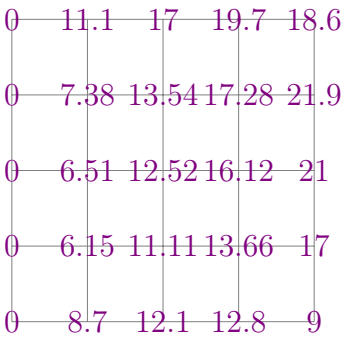
Using this grid value u_5 , now we can compute u_1, u_3, u_7 and u_9 , using the DFPF as below.

$$\begin{aligned} u_{1,1} &= \frac{u_{0,0} + u_{0,2} + u_{2,0} + u_{2,2}}{4} \\ &= \frac{0.00 + 17.00 + 0.00 + 12.52}{4} = 7.38 \\ u_{1,3} &= \frac{u_{0,2} + u_{0,4} + u_{2,2} + u_{2,4}}{4} \\ &= \frac{17.00 + 18.60 + 12.52 + 21.00}{4} = 17.28 \\ u_{3,1} &= \frac{u_{2,0} + u_{2,2} + u_{4,0} + u_{4,2}}{4} \\ &= \frac{0.00 + 12.52 + 0.00 + 12.10}{4} = 6.15 \\ u_{3,3} &= \frac{u_{2,2} + u_{2,4} + u_{4,2} + u_{4,4}}{4} \\ &= \frac{12.52 + 21.00 + 12.10 + 9.00}{4} = 13.66 \end{aligned}$$

To compute the remaining unknown grid values, now we can employ SFPF. Hence we get,

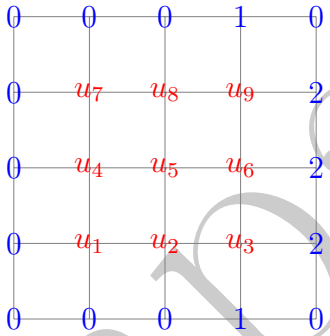
$$\begin{aligned} u_{1,2} &= \frac{u_{1,1} + u_{2,2} + u_{1,3} + u_{0,2}}{4} \\ &= \frac{7.38 + 12.52 + 17.28 + 17.00}{4} = 13.54 \\ u_{2,1} &= \frac{u_{2,0} + u_{3,1} + u_{2,2} + u_{1,1}}{4} \\ &= \frac{0.00 + 6.15 + 12.52 + 7.38}{4} = 6.51 \\ u_{2,3} &= \frac{u_{2,2} + u_{3,3} + u_{2,4} + u_{1,3}}{4} \\ &= \frac{12.52 + 13.66 + 21.00 + 17.28}{4} = 16.12 \\ u_{3,2} &= \frac{u_{3,1} + u_{4,2} + u_{3,3} + u_{2,2}}{4} \\ &= \frac{6.15 + 12.10 + 13.66 + 12.52}{4} = 11.11 \end{aligned}$$

Hence the solution is given by,



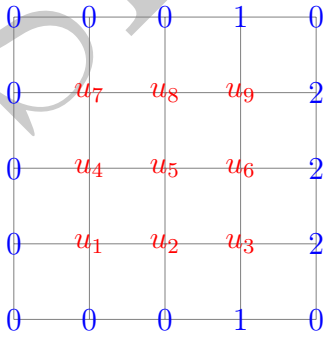
Example 1.8

Solve the Laplace's Equation of the form $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ using the grid points given below.



Solution

We have the grid points are given by



From the grid points it is clear that,the unknown points are given by the coordinate points as given below.

$$\begin{aligned} u_{1,1} &= u_1, \quad u_{1,2} = u_2, \quad u_{1,3} = u_3 \\ u_{2,1} &= u_4, \quad u_{2,2} = u_5, \quad u_{2,3} = u_6 \\ u_{3,1} &= u_7, \quad u_{3,2} = u_8, \quad u_{3,3} = u_9 \end{aligned}$$

Now we will compute the grid value at $u_5 = u_{i,j}$ using the SFPF, so we have,

$$\begin{aligned} u_{2,2} &= \frac{u_{2,0} + u_{4,2} + u_{2,4} + u_{0,2}}{4} \\ &= \frac{0.00 + 0.00 + 2.00 + 0.00}{4} = 0.50 \end{aligned}$$

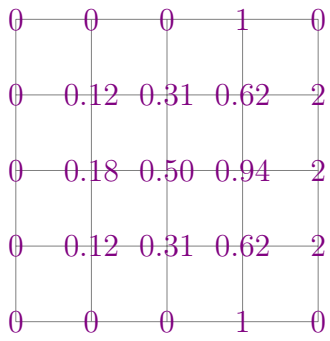
Using this grid value u_5 , now we can compute u_1, u_3, u_7 and u_9 , using the DFPF as below.

$$\begin{aligned} u_{1,1} &= \frac{u_{0,0} + u_{0,2} + u_{2,0} + u_{2,2}}{4} \\ &= \frac{0.00 + 0.00 + 0.00 + 0.50}{4} = 0.12 \\ u_{1,3} &= \frac{u_{0,2} + u_{0,4} + u_{2,2} + u_{2,4}}{4} \\ &= \frac{0.00 + 0.00 + 0.50 + 2.00}{4} = 0.62 \\ u_{3,1} &= \frac{u_{2,0} + u_{2,2} + u_{4,0} + u_{4,2}}{4} \\ &= \frac{0.00 + 0.50 + 0.00 + 0.00}{4} = 0.12 \\ u_{3,3} &= \frac{u_{2,2} + u_{2,4} + u_{4,2} + u_{4,4}}{4} \\ &= \frac{0.50 + 2.00 + 0.00 + 0.00}{4} = 0.62 \end{aligned}$$

To compute the remaining unknown grid values, now we can employ SFPF. Hence we get,

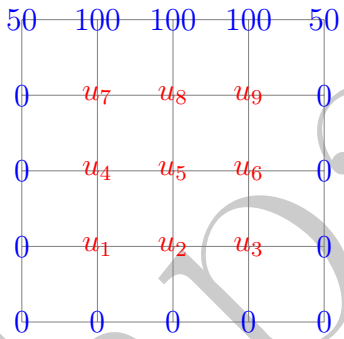
$$\begin{aligned} u_{1,2} &= \frac{u_{1,1} + u_{2,2} + u_{1,3} + u_{0,2}}{4} \\ &= \frac{0.12 + 0.50 + 0.62 + 0.00}{4} = 0.31 \\ u_{2,1} &= \frac{u_{2,0} + u_{3,1} + u_{2,2} + u_{1,1}}{4} \\ &= \frac{0.00 + 0.12 + 0.50 + 0.12}{4} = 0.18 \\ u_{2,3} &= \frac{u_{2,2} + u_{3,3} + u_{2,4} + u_{1,3}}{4} \\ &= \frac{0.50 + 0.62 + 2.00 + 0.62}{4} = 0.94 \\ u_{3,2} &= \frac{u_{3,1} + u_{4,2} + u_{3,3} + u_{2,2}}{4} \\ &= \frac{0.12 + 0.00 + 0.62 + 0.50}{4} = 0.31 \end{aligned}$$

Hence the solution is given by,



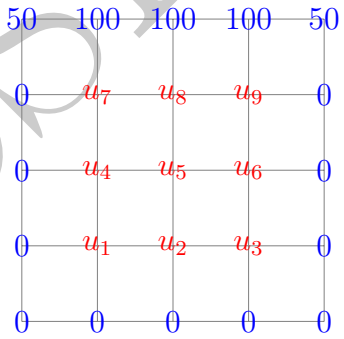
Example 1.9

Solve the Laplace's Equation of the form $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ using the grid points given below.



Solution

We have the grid points are given by



From the grid points it is clear that,the unknown points are given by the coordinate points as given below.

$$\begin{aligned} u_{1,1} &= u_1, \quad u_{1,2} = u_2, \quad u_{1,3} = u_3 \\ u_{2,1} &= u_4, \quad u_{2,2} = u_5, \quad u_{2,3} = u_6 \\ u_{3,1} &= u_7, \quad u_{3,2} = u_8, \quad u_{3,3} = u_9 \end{aligned}$$

Now we will compute the grid value at $u_5 = u_{i,j}$ using the SFPF, so we have,

$$\begin{aligned} u_{2,2} &= \frac{u_{2,0} + u_{4,2} + u_{2,4} + u_{0,2}}{4} \\ &= \frac{0.00 + 0.00 + 0.00 + 100.00}{4} = 25.00 \end{aligned}$$

Using this grid value u_5 , now we can compute u_1, u_3, u_7 and u_9 , using the DFPF as below.

$$\begin{aligned} u_{1,1} &= \frac{u_{0,0} + u_{0,2} + u_{2,0} + u_{2,2}}{4} \\ &= \frac{50.00 + 100.00 + 0.00 + 25.00}{4} = 43.75 \\ u_{1,3} &= \frac{u_{0,2} + u_{0,4} + u_{2,2} + u_{2,4}}{4} \\ &= \frac{100.00 + 50.00 + 25.00 + 0.00}{4} = 43.75 \\ u_{3,1} &= \frac{u_{2,0} + u_{2,2} + u_{4,0} + u_{4,2}}{4} \\ &= \frac{0.00 + 25.00 + 0.00 + 0.00}{4} = 6.25 \\ u_{3,3} &= \frac{u_{2,2} + u_{2,4} + u_{4,2} + u_{4,4}}{4} \\ &= \frac{25.00 + 0.00 + 0.00 + 0.00}{4} = 6.25 \end{aligned}$$

To compute the remaining unknown grid values, now we can employ SFPF. Hence we get,

$$\begin{aligned} u_{1,2} &= \frac{u_{1,1} + u_{2,2} + u_{1,3} + u_{0,2}}{4} \\ &= \frac{43.75 + 25.00 + 43.75 + 100.00}{4} = 53.12 \\ u_{2,1} &= \frac{u_{2,0} + u_{3,1} + u_{2,2} + u_{1,1}}{4} \\ &= \frac{0.00 + 6.25 + 25.00 + 43.75}{4} = 18.75 \\ u_{2,3} &= \frac{u_{2,2} + u_{3,3} + u_{2,4} + u_{1,3}}{4} \\ &= \frac{25.00 + 6.25 + 0.00 + 43.75}{4} = 18.75 \\ u_{3,2} &= \frac{u_{3,1} + u_{4,2} + u_{3,3} + u_{2,2}}{4} \\ &= \frac{6.25 + 0.00 + 6.25 + 25.00}{4} = 9.38 \end{aligned}$$

Hence the solution is given by,