Introduction to Numerical Computing

What is Numerical Methods?
Numerical method:

- An approach to find an approximate solution to complex mathematical problems using only simple arithmetic operations.
- Employed for problem for which an analytical result either cannot be obtained or not practical.
- Techniques of solving non-analytical problems.
- Analysis error for true value
- Involve computing a large number of arithmetic calculations and, therefore, requiring fast and efficient computing devices.

❑ The traditional numerical computing methods usually deals with the following topics:

- Finding roots of equations.
- Solving system of linear algebraic equations
- Interpolation & Regression Analysis
- Numerical Integration
- Numerical Differentiation
- Solving of Differential Equation
- Boundary value Problems
- Solution of matrix problems

Numeric Data
Numerical computing may involve two types of data:

- *discrete data* and *continuous data*.

❑ Data that are obtained by counting are called discrete data. Examples of discrete data are total number of students in a class, total number of items in a box etc.

❑ Data that are obtained through measurement are called continuous data. Example of continuous data is the temperature of a patient as measured by a thermometer.

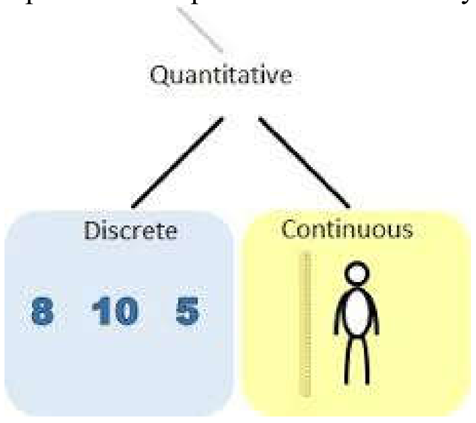


Figure 1.1: Discrete Vs continuous

Process of Numerical Computing

Numerical computing involves formulation of mathematical models of physical problems that can be solved using basic arithmetic operations. The process of numerical computing can be roughly divided into the following four phases which are illustrated in Fig-1.1

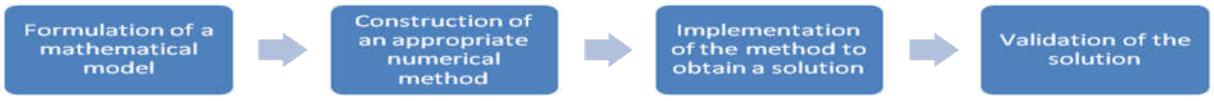

Formulation of a mathematical model
Construction of an appropriate numerical method
Implementation of the method to obtain a solution
Validation of the solution

Figure 1.2: Processes of Numerical Computings

Characteristics of Numerical Computing
Numerical techniques show certain computational characteristics during their implementation. It is imperative to think about these attributes while picking a specific strategy for execution. The characteristics that are critical to the success of implementation are:

1. Accuracy
2. Rate of convergence
3. Numerical stability
4. Efficiency

Accuracy:
There is computational error for every numerical method such as truncation errors or round off errors. These errors affect the accuracy of the result**.** The results we obtain must be sufficiently accurate to serve the purpose for which the mathematical model was built.

Rate of convergence:
Many numerical methods involved in an iterative process. Results in a solution moves progressively closer to the true solution at successive iterations, is known as **convergence**. A numerical method is not always guaranteed to produce converging results. It is, therefore, important to test for convergence before a method is used.

Numerical Stability:
In some cases error grows exponentially, which causes disaster computational results for specific problem. If a process shows such growth is said to be numerically unstable. We must choose methods that are not only fast but also stable. Numerical instability may also arise due to ill-conditioned problems.

Efficiency:
We can consider a numerical method is efficient if it requires less of computing time, less of programming effort and yet achieves the desired accuracy by both human and computer.

Representation of Numbers

All modern computers understand only binary digits to represent numbers and other information. The memory is usually organized into strings of bits called *words* and the largest number a computer can store depends on its word length. For example, the largest binary number a 16 bit word can hold is 16 bits of 1 and equivalent to a decimal value of 65535. The following relation gives the largest decimal number that can be stores in a computer:

$$\textit{Largest number} = 2^n - 1$$

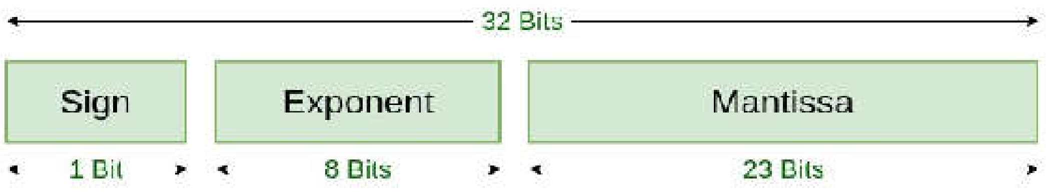
where n is the word length in bits. Thus, we see that the greater number of bits, the larger the number that may be stored.
Decimal numbers are first converted into the binary equivalent and then represented in either integer or floating-point form.

Integer representation

- For integers, the decimal or binary point is always fixed to the right of the least significant digit and therefore, fraction is not included.
- The magnitude of the number is restricted to $2^n - 1$ where n is the word length in bits.
- Negative numbers are stored by using the 2's complement. This is achieved by taking the 1's complement of the binary representation of the positive number and then adding 1 to it.
- If we reserve one bit to represent the sign of the number, called *sign bit*, we have only n-1 bits to represent the number. Thus a 16bit word can contain numbers -2^{15} to $2^{15} - 1$ (i,e. –32768 to 32767).
- Generally, if the sign bit is 1, the number is negative and if the sign bit is 0, the number is positive.

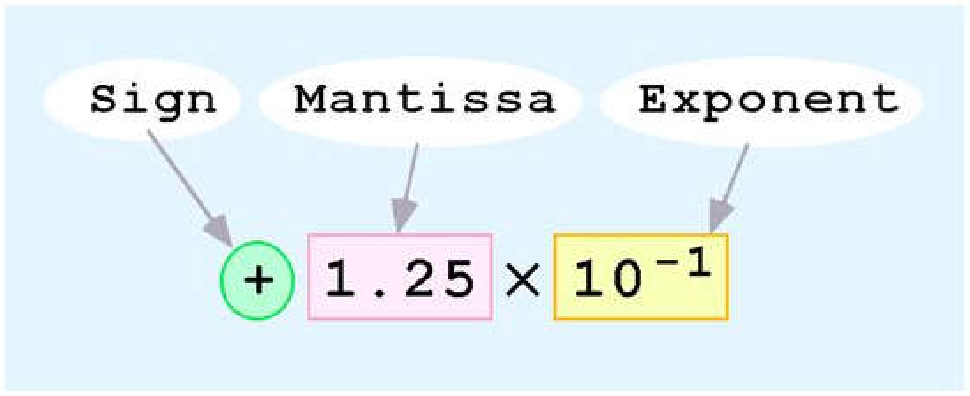
Floating point Representation

- Use scientific notation to represent floating point and have three components.



Single Precision
IEEE 754 Floating-Point Standard

Example



- 35.7812 → $0.357812 * 10^2$.
- 987654321 → $0.987654 * 10^9$.
- By writing a large number in exponential form, we may lose some digits. If x is a real number, its floating-point form representation is $x = f * 10^E$ where the number f is called mantissa and E is the exponent.
- The shifting of decimal points to the left of the most significant digit is called normalization and the numbers represented in normalized form are known as normalized floating point numbers.
- Mantissa should satisfy the following conditions:
  For positive numbers: less than 1.0 but greater than or equal to 0.1.
  For negative numbers: greater than -1.0 but less than or equal to -0.1.
  That is $0.1 \leq |f| < 1$.