DEPARTMENT OF COMPUTER ENGINEERING

SUBJECT: COMPUTER ORIENTED NUMERICAL TECHNIQUES

LAB MANUAL

Experiment No: 1 Case Study: Introduction to numerical techniques – Errors & Approximations

- What is Numerical Computing
- Numerical Data
- Process of Numerical Computing
- Characteristics of Numerical Computing
- Numerical Errors
- Error Estimation

Numerical Computing: Is an approach for solving complex mathematical problems using only simple arithmetic operations.

- The study of approximation techniques for numerically solving mathematical problems.
  Numerical Computation is necessary for problem solving in that not all the (in fact, very few) mathematical problems have closed-form solutions. We can only obtain numerical solutions.
  For example:
  1The root of a polynomial with degree 5 or more
  2 The eigenvalues of a matrix

Numerical computations play an indispensable role in solving real life mathematical, physical and engineering problems. They have been in use for centuries even before digital computers appeared on the scene. Great mathematicians like Gauss, Newton, Lagrange, Fourier and many others in the eighteenth and nineteenth centuries developed numerical techniques which are still widely used. The advent of digital computers has, however, enhanced the speed and accuracy of numerical computations.

Traditional Numerical Computing methods usually deal with the following topics:

- Finding roots of equations
- Solving systems of linear algebraic equations
- Interpolation and regression analysis
- Numerical integration
- Numerical differentiation
- Solution of differential equations
- Boundary value problem
- Solution of matrix problem

Numerical Data:

-Discrete: data obtained by counting (e.g number of items in box)

- Continuous: data obtained through measurement (speed of a car)

Process of Numerical Computing

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

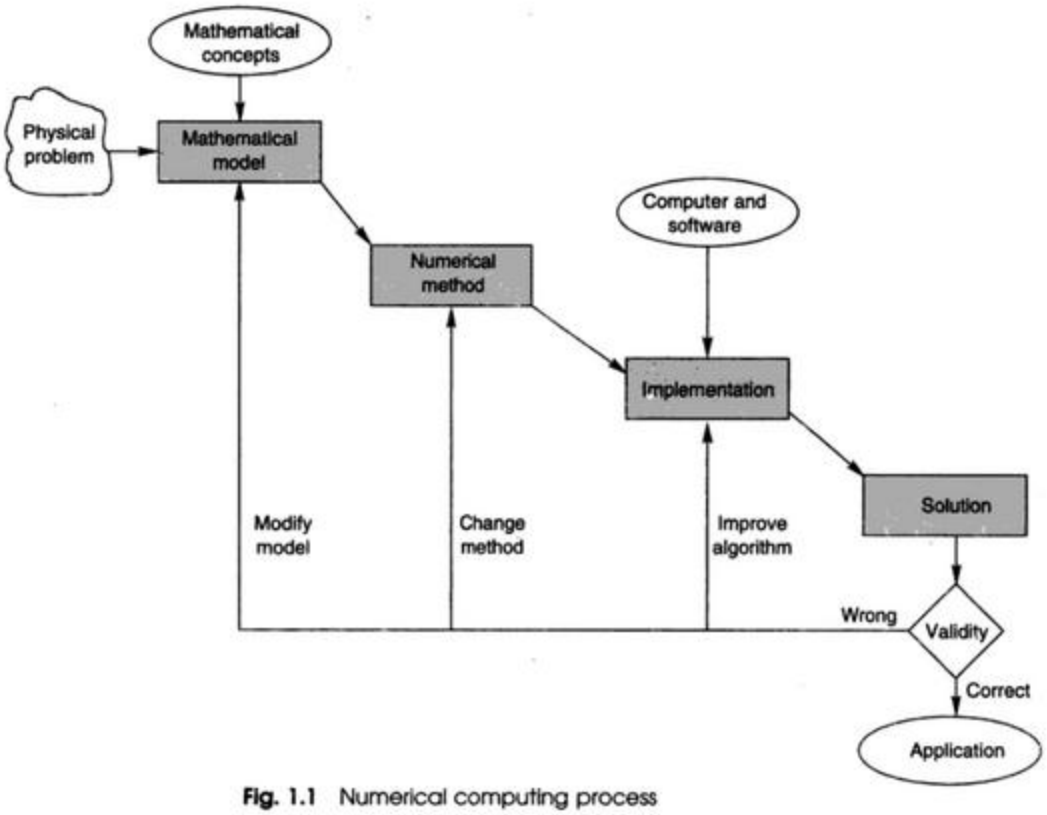


Fig. 1.1 Numerical computing process

Types of Numerical Equations

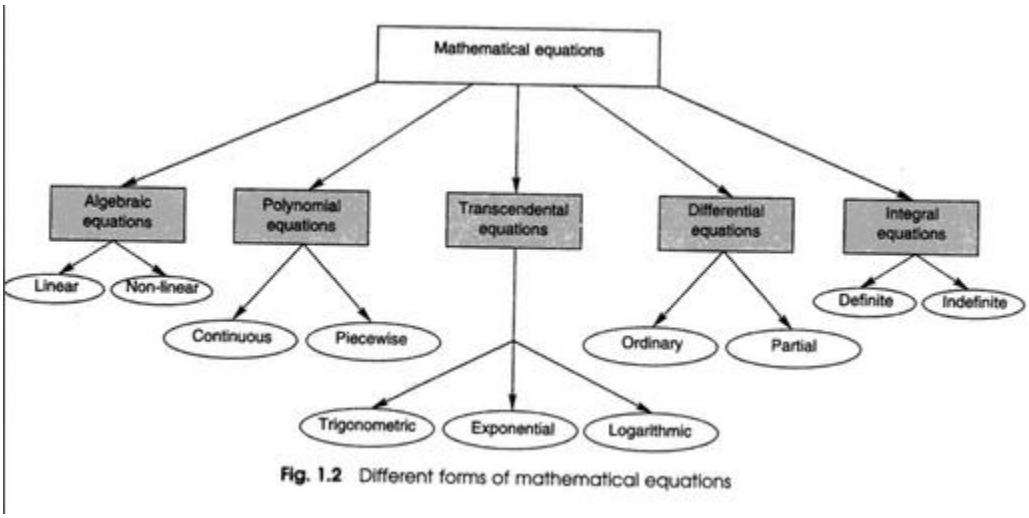


Fig. 1.2 Different forms of mathematical equations

Characteristics of Numerical Computing

- Accuracy
- Rate of convergence
- Numerical stability
- Efficiency

1.6 CHARACTERISTICS OF NUMERICAL COMPUTING

Numerical methods exhibit certain computational characteristics during their implementation. It is important to consider these characteristics while choosing a particular method for implementation. The characteristics that are critical to the success of implementation are: accuracy, rate of convergence, numerical stability, and efficiency.

Accuracy

Every method of numerical computing introduces errors. They may be either due to using an approximation in place of an exact mathematical procedure (known as *truncation errors*) or due to inexact representation and manipulation of numbers in the computer (known as *roundoff errors*). These errors affect the accuracy of the results. The results we obtain must be sufficiently accurate to serve the purpose for which the mathematical model was built. Choice of a method is, therefore, very much dependent on the particular problem. The general nature of these errors will be discussed in detail in Chapter 4.

Rate of Convergence

Many numerical methods are based on the idea of an *iterative process*. This process involves generation of a sequence of approximations with the hope that the process will converge to the required solution. Certain methods converge faster than others. Some methods may not converge at all. It is, therefore, important to test for convergence before a method is used. Rapid convergence takes less execution time on the computer. There are several techniques for accelerating the rate of convergence of certain methods. The concepts of convergence and divergence are discussed in Chapter 4. They are also discussed in various places where specific methods are analysed for convergence.

Numerical Stability

Another problem introduced by some numerical computing methods is that of *numerical instability*. Errors introduced into a computation, from whatever source, propagate in different ways. In some cases, these errors tend to grow exponentially, with disastrous computational results. A computing process that exhibits such exponential error 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. There are many problems which are inherently sensitive to round off errors and other uncertainties. Thus, we must distinguish between sensitivity of methods and sensitivity inherent in problems.

When the problem is ill-conditioned, there is nothing we can do to make a method to become numerically stable.

Numerical Errors:

In software engineering and mathematics, numerical error is the combined effect of two kinds of error in a calculation. The first is caused by the finite precision of computations involving floating-point or integer values. The second usually called truncation error is the difference between the exact mathematical solution and the approximate solution obtained when simplifications are made to the mathematical equations to make them more amenable to calculation. The term

truncation comes from the fact that either these simplifications usually involve the truncation of an infinite series expansion so as to make the computation possible and practical, or because the least significant bits of an arithmetic operation are thrown away.

Taxonomy of Errors

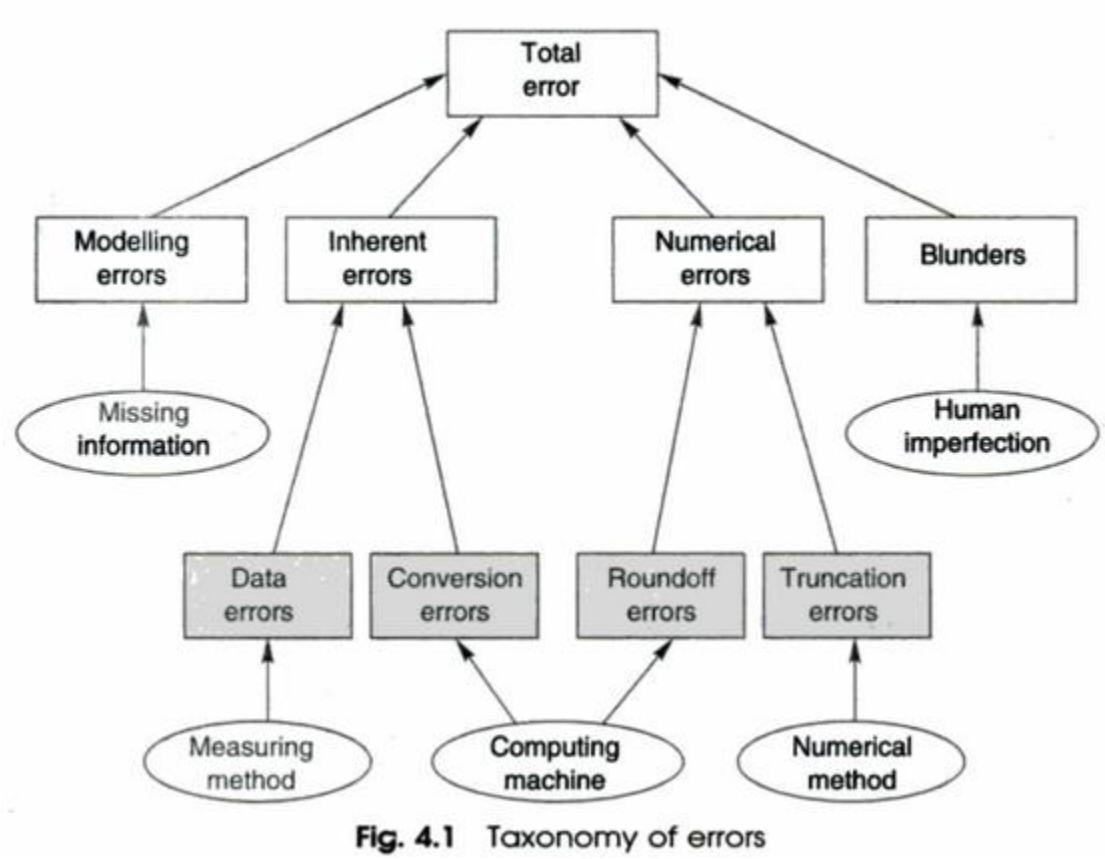


Fig. 4.1 Taxonomy of errors

4.3 INHERENT ERRORS

Inherent errors are those that are present in the data supplied to the model. Inherent errors (also known as *input errors*) contain two components, namely, *data errors* and *conversion errors*.

Data Errors

Data error (also known as *empirical error*) arises when data for a problem are obtained by some experimental means and are, therefore, of limited accuracy and precision. This may be due to some limitations in instrumentation and reading, and therefore may be unavoidable. A physical measurement, such as a distance, a voltage, or a time period, cannot be exact. It is, therefore, important to remember that there is no use in performing arithmetic operations to, say, four decimal places when the original data themselves are only correct to two decimal places. For instance, the scale reading in a weighing machine may be accurate to only one decimal place.

Conversion Errors

Conversion errors (also known as *representation errors*) arise due to the limitations of the computer to store the data exactly. We know that the floating point representation retains only a specified number of digits. The digits that are not retained constitute the roundoff error.

As we have already seen, many numbers cannot be represented exactly in a given number of decimal digits. In some cases a decimal number cannot be represented exactly in binary form. For example, the decimal number 0.1 has a non-terminating binary form like 0.00011001100110011.... but the computer retains only a specified number of bits. Thus, if we add 10 such numbers in a computer, the result will not be exactly 1.0 because of roundoff error during the conversion of 0.1 to binary form.

4.4 NUMERICAL ERRORS

Numerical errors (also known as *procedural errors*) are introduced during the process of implementation of a numerical method. They come in two forms, *roundoff errors* and *truncation errors*. The total numerical error is the summation of these two errors. The total error can be reduced by devising suitable techniques for implementing the solution. We shall see in this section the magnitude of these errors.

Roundoff Errors

Roundoff errors occur when a fixed number of digits are used to represent exact numbers. Since the numbers are stored at every stage of computation, roundoff error is introduced at the end of every arithmetic operation. Consequently, eventhough an individual roundoff error could be very small, the cumulative effect of a series of computations can be very significant.

Rounding a number can be done in two ways. One is known as *chopping* and the other is known as *symmetric rounding*. Some systems use the chopping method while others use symmetric rounding.

Chopping

In chopping, the extra digits are dropped. This is called *truncating* the number. Suppose we are using a computer with a fixed word length of four digits. Then a number like 42.7893 will be stored as 42.78, and the digits 93 will be dropped. We can express the number 42.7893 in floating point form as

$$\begin{aligned} x &= 0.427893 \times 10^2 \\ &= (0.4278 + 0.000093) \times 10^2 \\ &= [0.4278 + (0.93 \times 10^{-4})] \times 10^2 \end{aligned}$$

This can be expressed in general form as

$$\begin{aligned} \text{True } x &= (f_x + g_x \times 10^{-d})10^E \\ &= f_x \times 10^E + g_x \times 10^{E-d} \\ &= \text{approximate } x + \text{error.} \end{aligned}$$

Symmetric Roundoff

In the symmetric roundoff method, the last retained significant digit is "rounded up" by 1 if the first discarded digit is larger or equal to 5; otherwise, the last retained digit is unchanged. For example, the number 42.7893 would become 42.79 and the number 76.5432 would become 76.54.

Truncation Errors

Truncation errors arise from using an approximation in place of an exact mathematical procedure. Typically, it is the error resulting from the truncation of the numerical process. We often use some finite number of terms to estimate the sum of an infinite series. For example,

$$S = \sum_{i=0}^{\infty} a_i x^i \text{ is replaced by the finite sum } \sum_{i=0}^{n} a_i x^i$$

The series has been truncated.

Another example is the use of a number of discrete steps in the solution of a differential equation. The error introduced by such discrete approximations is also called *discretisation error*. Consider the following infinite series:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

When we calculate the sine of an angle using this series, we cannot use all the terms in the series for computation. We usually terminate the process after a certain term is calculated. The terms "truncated" introduce an error which is called *truncation error*.

Find the truncation error in the result of the following function for $x = 1/5$ when we use (a) first three terms, (b) first four terms, and (c) first five terms.

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!}$$

(a) Truncation error when first three terms are added

$$\text{Truncation error} = + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!}$$

$$= + \frac{0.2^3}{6} + \frac{0.2^4}{24} + \frac{0.2^5}{120} + \frac{0.2^6}{720}$$

$$= 0.1402755 \times 10^{-2}$$

(b) Truncation error when first four terms are added

$$\text{Truncation error} = 0.694222 \times 10^{-4}$$

(c) Truncation error when first five terms are added

$$\text{Truncation error} = 0.275555 \times 10^{-5}$$

4.5 MODELLING ERRORS

Mathematical models are the basis for numerical solutions. They are formulated to represent physical processes using certain parameters involved in the situations. In many situations, it is impractical or impossible to include all of the real problem and, therefore, certain simplifying assumptions are made. For example, while developing a model for calculating the force acting on a falling body, we may not be able to estimate the air resistance coefficient (*drag coefficient*) properly or determine the direction and magnitude of wind force acting on the body, and so on. To simplify the model, we may assume that the force due to air resistance is linearly proportional to the velocity of the falling body or we may assume that there is no wind force acting on the body. All such simplifications certainly result in errors in the output from such models.

4.6 BLUNDERS

Blunders are errors that are caused due to human imperfection. As the name indicates, such errors may cause a very serious disaster in the result. Since these errors are due to human mistakes, it should be possible to avoid them to a large extent by acquiring a sound knowledge of all aspects of the problem as well as the numerical process.

Human errors can occur at any stage of the numerical processing cycle. Some common types of errors are:

1. lack of understanding of the problem
2. wrong assumptions
3. overlooking of some basic assumptions required for formulating the model
4. errors in deriving the mathematical equation or using a model that does not describe adequately the physical system under study

5. selecting a wrong numerical method for solving the mathematical model
6. selecting a wrong algorithm for implementing the numerical method
7. making mistakes in the computer program, such as testing a real number for zero and using < symbol in place of > symbol
8. mistakes in data input, such as misprints, giving values column-wise instead of row-wise to a matrix, forgetting a negative sign, etc.
9. wrong guessing of initial values

As mentioned earlier, all these mistakes can be avoided through a reasonable understanding of the problem and the numerical solution methods, and use of good programming techniques and tools.

4.7 ABSOLUTE AND RELATIVE ERRORS

Let us now consider some fundamental definitions of error analysis. Regardless of its source, an error is usually quantified in two different but related ways. One is known as *absolute error* and the other is called *relative error*.

Let us suppose that the *true value* of a data item is denoted by x_t and its *approximate value* is denoted by x_a. Then, they are related as follows:

$$\text{True value } x_t = \text{Approximate value } x_a + \text{Error.}$$

The error is then given by

$$\text{Error} = x_t - x_a$$

The error may be negative or positive depending on the values of x_t and x_a. In error analysis, what is important is the magnitude of the error and not the sign and, therefore, we normally consider what is known as *absolute error* which is denoted by

$$e_a = |x_t - x_a|$$

In many cases, absolute error may not reflect its influence correctly as it does not take into account the order of magnitude of the value under study. For example, an error of 1 gram is much more significant in the weight of a 10 gram gold chain than in the weight of a bag of rice. In view of this, we introduce the concept of *relative error* which is nothing but the "normalised" absolute error. The relative error is defined as follows:

$$e_r = \frac{\text{absolute error}}{|\text{true value}|}$$

$$= \frac{|x_t - x_a|}{|x_t|} = \left|1 - \frac{x_a}{x_t}\right|$$

4.12 ERROR ESTIMATION

It is now clear that it is almost impossible to know the exact error in a computed result. Nevertheless, it is possible at least to have some estimate of the error in the final result. There are three approaches that are popularly used in error estimation:

1. forward error analysis
2. backward error analysis
3. experimental error analysis

In *forward error analysis*, we try to estimate error bounds in the computed result using information such as uncertainties in the input data and the nature and number of arithmetic operations involved in the computing process. We can estimate the contribution due to

1. errors in the input data
2. roundoff errors in arithmetic operations
3. truncation of the iterative process
4. errors in formulation of the model

In *backward error analysis*, we try to show that the computed results satisfy the problem within the given bounds. For example, we can put back the roots computed in the equation and see to what extent they satisfy the original equation. By comparison, we can then decide on how much confidence we can place in the computed results. Backward analysis is usually easier to perform than forward error analysis.

Experimental error analysis involves a series of experiments by using different methods and step sizes and then comparing the results. We may also perform *sensitivity analysis* to see how any change in parameters affects the result.

When the application is very critical in nature (such as space and defence applications) the problem may be solved by more than two independent specialists groups and the results can be compared.

4.13 MINIMISING THE TOTAL ERROR

Assuming that the mathematical model has been properly formulated and the input data are accurate, the total numerical error primarily consists of two components, namely, truncation and roundoff errors. Any effort to minimise the total error should, therefore, be concentrated on the ways to reduce these two types of errors. The steps may include:

1. increasing the significant figures of the computer
2. minimising the number of arithmetic operations
3. avoiding subtractive cancellations
4. choosing proper initial parameters

In many iterative processes such as numerical integration, it is possible to minimise the truncation error by decreasing the step size. But this would necessarily increase the number of iterations and thereby, arithmetic operations. This would certainly increase the roundoff error. This phenomenon is illustrated in Fig. 4.4. We must, therefore, judiciously choose a step size that would minimise the sum of these errors.

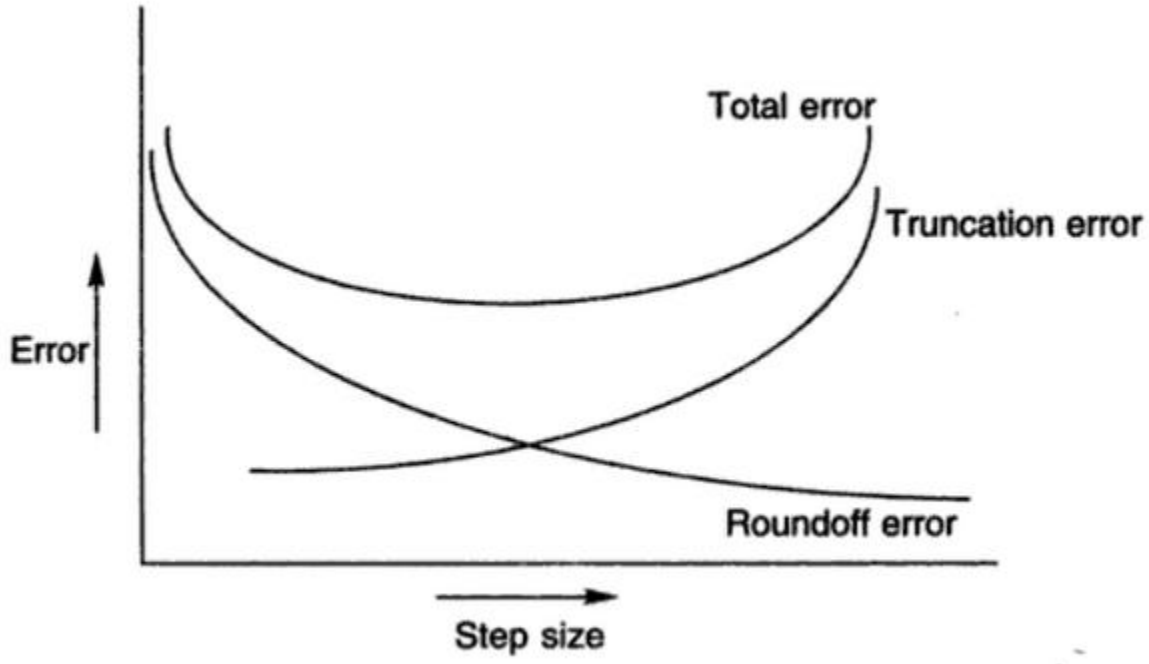


Fig. 4.4 Dependence of error on step size

Conclusion:

Experiment No: 2: Implementation of Bisection Method

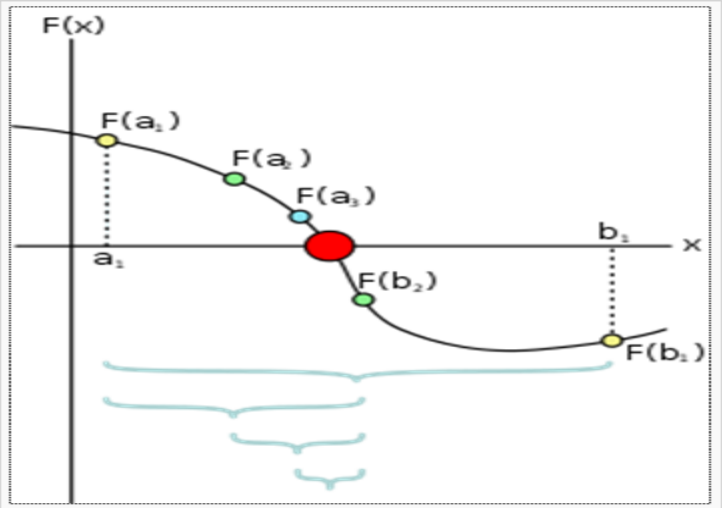
Theory:

The bisection method in mathematics is a root-finding method which repeatedly bisects an interval and then selects a subinterval in which a root must lie for further processing. It is a very simple and robust method, but it is also relatively slow. Because of this, it is often used to obtain a rough approximation to a solution which is then used as a starting point for more rapidly converging methods.[1] The method is also called the binary search method

The method is applicable when we wish to solve the equation f(x) = 0 for the real variable x, where f is a continuous function defined on an interval [a, b] and f(a) and f(b) have opposite signs. In this case a and b are said to bracket a root since, by the intermediate value theorem, the f must have at least one root in the interval (a, b).

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

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



Algorithm:

1. Decide initial values for x1 & x2 & stopping criterion E.
2. Compute f1=f(x1), f2=f(x2)
3. If f1 * f2 > 0, x1 & x2 do not bracket any root & goto step 7
   Otherwise continue
4. Compute x0=(x1+x2)/2 and compute f0=f(x0)
5. If f1*f0<0 then

   Set x2=x0

   Else

   Set x1=x0

   Set f1 = f0

6. If absolute value of (x2-x1)/x2 is less then error E then

   Root = (x1+x2)/2

   Write the value of root

   Goto step 7

   Else

   Goto step 4

7. Stop.

Conclusion: