

CHAPTER ONE

Basic concepts In Error Estimation

1.1 Introduction

Numerical Analysis is a subject that is concerned with devising methods for approximating the solution of mathematically expressed problems. Such problems may each be formulated for example in terms of algebraic, transcendental equations or ordinary differential equations or partial or integral equations. More often the mathematical problems cannot be solving by exact methods. The mathematical models ordinarily do not solve the physical problems exactly, so it is often more appropriate to find an approximate solution. Generally numerical analysis does not give exact solution; instead it attempts to devise a method which will yield an approximation differing from exactness by less than a specified tolerance. The efficiency of the method used to solve the given problem depends both upon the accuracy required of the method and the ease with which it can be implemented.

Numerical Analysis is the subject that deals with describing an algorithm that approximates the solution of a physical problem.

i.e

physical problem



Change to Mathematical Expression (Set a mathematical model for the physical problem)



Describe an algorithm to solve the Mathematical Problem

NB: 1. Algorithm is a step by step procedure of solving problems.

2. or it is a finite, sequential (ordered) computational steps of solving a problem.

Numerical Analysis develops an algorithm to solve the problem. Hence it wills an approximate solution for the physical problem. As a result error will induce.

1.2 Sources of Errors

Analysis of errors is the central concern in the study of numerical analysis and therefore we will investigate the sources and types of errors that may occur in a given problem and the subsequent propagation of errors.

Errors in the solution of a problem are due to the following reasons:

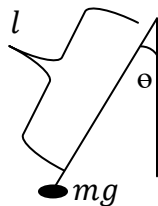
- To solve physical problems, mathematical models are formulated to describe them and these models do not describe the problems exactly and as a result errors are introduced. (Mathematical Models are not the exact copy or model or representation of the physical problem).
- The methods used to solve the mathematical models are often not exact and as a consequence errors are **introduced**. i.e. All mathematical expressions are not solved analytically or exactly.
- Computation for the solution is done using Computational tools that have limited space, like calculators, computer, etc. A computer has a finite word length and so only a fixed number of digits of a number are inserted and as a consequence errors are introduced.

1.3 Classification of Errors

The errors induced by the sources mentioned above are classified as:

- Inherent Errors:** These are errors that we cannot avoid; unless the mathematical models that are formulated to describe the physical problems are exact such errors will always be induced. Because of this such errors are called Inherent errors.
- Truncation Errors:** The mathematical models may be formulated in algebraic or transcendental or other type of equations. The solutions of such equations may not be solved analytically. Hence, we use numerical methods to obtain the solutions of such equations. In the process errors will be induced. Such errors are called Truncation error (errors due to the method).
- Computational Errors:** Computational tools have limited space to store digits, and a number with number of the digits greater than the tools accommodation capacity will be truncated and such errors are called computational errors.

To illustrate the ways in which the above errors arise, let us take oscillation of a pendulum as an example.



The differential equation that describes the oscillation of the pendulum is taken in the form

$$\lambda \frac{d^2\theta}{dt^2} + mg \sin\theta + \mu \frac{d\theta}{dt} = 0 \quad \dots\dots\dots (1.1)$$

λ = length of the pendulum

g = acceleration due to gravity

μ = coefficient of friction.

- The mathematical description of the pendulum's oscillation is not exact because air resistance and friction at the pivot are neglected and as a consequence the coefficient of friction is considered as linearly dependent on the speed. And these errors are inherent because they cannot be controlled in the process of numerical solution.
- The differential equation cannot be solved explicitly and will require some numerical methods and this generates the errors of the method.
- Determination of λ, g and m are not exactly correct and computation of these is not exact and hence computational error will be produced.

Using symbols we can see the classification of errors:

Let x denote the exact solution of the physical problem.

Let \tilde{x} be the solution corresponding to the given mathematical description (model)

Let \tilde{x}_n be the solution of the problem obtained from the numerical method on the assumptions that roundings are absent.

Let \tilde{x}_n^* be the approximation to the solution obtained in the actual computation.

Then we have:

$$\text{Inherent error} = |\tilde{x} - x| = e_i$$

$$\text{Error due to the method} = |\tilde{x}_n - \tilde{x}| = e_m$$

$$\text{Computational error} = |\tilde{x}_n^* - \tilde{x}_n| = e_c$$

$$\text{Then actual error induced} = |\tilde{x}_n^* - x| = e$$

$$e = |\tilde{x}_n^* - x| = |\tilde{x}_n^* - \tilde{x}_n| + |\tilde{x}_n - \tilde{x}| + |\tilde{x}_n - x|$$

$$|\tilde{x}_n^* - x| \leq |\tilde{x}_n^* - \tilde{x}_2| + |\tilde{x}_n - \tilde{x}| + |\tilde{x}_n - x|$$

$$e \leq e_c + e_m + e_l \quad (1.2)$$

In many cases, the error is meant not the difference between the approximation and the exact but rather certain measures of distance between them.

Hence, If $e \leq e_c + e_m + e_l = \text{Tolerance}$, then the numerical solution is accepted.

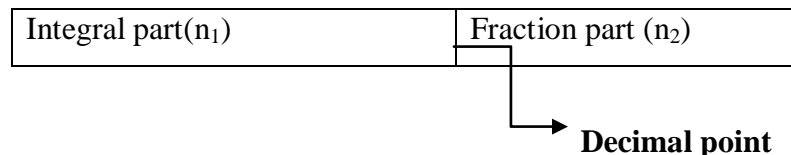
1.4 Computer Representation of Numbers.

Since there is a fixed space (word length) of memory in the digital computer, a given number in a certain base must be represented in a finite space in the memory of the computer. This means not all digits of the number can be represented in the memory. There are two conventional ways of representation of data in the word length or in a computer.

1.4.1 Fixed Point Representation

Suppose the number to be represented has n digits. In the fixed point representation system the n digits are subdivided into n_1 and n_2 where n_1 is reserved for the integral part and n_2 is reserved for the fractional part of the number. Here, whatever the value of the number is, the number n_1 and n_2 are fixed a priori .i.e. the decimal point is fixed.

The slot is



$$n_1 + n_2 = \text{word length}$$

Example:

1. Suppose $x = 13042$ is the number to be represented. If $n_1 = 3 = n_2$ then x will be represented as $\tilde{x} = 130.420$ in the machine on the other hand if $n_1 = 1$ and $n_2 = 4$ the $\tilde{x} = 1.3042$ and for $n_1 = 5$ and $n_2 = 0$, we get $\tilde{x} = 13042.0$
2. Let $x = 1,345.267$ using a fixed point representation insert x on a $2 + 4$ word length computer.

$n_1 = 13$	$n_2 = 4526$
------------	--------------

The computer understands the given number as $x' = 13.4526$

1.4.2 Floating Point Representation

Computers represent an n-digit floating point number in base β , in general form as:

$$X = \pm f \times \beta^E, \text{ where } f \in [1, \beta)$$

f is called the mantissa / the fraction part.

E is called the exponent

β is called the base the number base that is used

$$X = \pm (.d_1 d_2 d_3 \dots d_n) \times \beta^E$$

Where the d_i 's are digits or bits with values from zero to (β) and the whole d 's are called mantissa.

Example: $30.421 = \underbrace{0.30421}_{\text{mantissa}} \times \underbrace{10^2}_{\text{base}} \rightarrow \text{exponent}.$

A floating point representation is normalized if the first digit of the mantissa is different from zero.

i.e. If $0 \leq d_i \leq 9$ and $d_1 \neq 0$.

Example: Let $x = 426.357$, then the normalized floating point representation of x is 4.26357×10^2

In computers, floating-point numbers have three parts: the sign, the fraction part often called the mantissa and the exponent part. The three parts of the number have a fixed total length that is often 32 or 64 bits (some times even more). The mantissa part uses most of these bits (23 – 52 bits) and this determines the precision, the exponent part uses (7 – 11) bits and this number determines the range of the values.

$$\begin{array}{ccccc}
 & & n \text{ bits} & & \\
 \text{Mantissa} & \text{sign} & & \text{Exponent} & \\
 & \pm & & & \\
 t\text{-bits} & & 1\text{-bit} & & r\text{-bits} \\
 n = t + r + 1
 \end{array}$$

Note: A number cannot be represented exactly if it contains more than t bits in the mantissa

1.4.3 The Rounding Off Errors

In a floating point representation system with t digits for the mantissa, a number with mantissa greater than t digits cannot be represented exactly. Thus, such a number must somehow be rounded off to t digits, and there are two ways of reducing the number of digits of a given number.

1.4.3.1 Chopping

In a t digit computation, all digits of the mantissa to the right are dropped off.

Example: Let $x = \text{sign } a \times 10^b$

where $a = \text{sign}(0, d_1 d_2 \dots d_t d_{t+1} \dots d_n)$

Then in a 't' digit computer the mantissa a is chopped as,

$\tilde{a} = \text{sign } 0, d_1 d_2 \dots d_t$ all digits starting from d_{t+1} to the right are dropped.

The $\tilde{x} = \text{sign } \tilde{a} \times 10^b = 0.d_1 d_2 \dots d_t \times 10^b$

Then the error induces $= |x - \tilde{x}| = 0.d_{t+1} d_{t+2} \dots d_n \times 10^{-t+b}$.

$$e = |x - \tilde{x}| = d_{t+1}.d_{t+2} \dots d_n \times 10^{b-(t+1)} \dots (1.3)$$

Since each $d_i \leq 9$ then $e = |x - \tilde{x}| \leq 10^{b-t}$

1.4.3.2 Rounding

In this method if $d_{t+1} \geq 5$, we add one to d_t i.e. we round up. If $d_{t+1} < 5$ we merely chop off all after the first t + 1 digits.

Example: Let $x = a \times 10^b$

where $a = 0, d_1 d_2 \dots d_t d_{t+1} \dots d_n$

Then in a t digit computation, the mantissa a will be round up as:

$$\tilde{a} = \begin{cases} 0.d_1 d_2 \dots d_t & \text{if } d_{t+1} < 5 \\ 0.d_1 d_2 \dots d_t + 10^{-t} & \text{if } d_{t+1} \geq 5 \end{cases} \dots (1.4)$$

Then $\tilde{x} = \text{sign } \tilde{a} \times 10^b = \text{sign } (0.d_1 d_2 \dots d_t + 10^{-t}) \times 10^b$

$$\begin{aligned}
\text{error } e &= |x - \tilde{x}| = |a - \tilde{a}_1| \times 10^b \\
&= a_{t+1}.a_{t+2} \dots a_n \times 10^{b-(t+s)} \\
&\leq 5 \times 10^{b-(t+1)} \text{ since } d_{t+1} < 5 \\
&= 0.5 \times 10^{b-t}
\end{aligned}$$

Example:

$$x = 0.142862 \times 10^{02}$$

Let $t = 4$ and $b = 2$

Chopping Form

$$\tilde{x} = 0.1428 \times 10^{02}$$

$$e = |x - \tilde{x}| = 6.2 \times 10^{02-5} < 10^{02-4}$$

Rounding Form

$$x = 0.142862 \times 10^{02} \text{ and } t = 4 \text{ and } b = 2$$

$$\text{the } \tilde{x} = 0.1429 \times 10^{02} \text{ since } 6 > 5$$

$$e = |x - \tilde{x}| = 3.8 \times 10^{02-5} < 5 \times 10^{02-5} = 0.5 \times 10^{02-4}$$

1.4.3.3 Absolute and Relative Errors.

The error that results from replacing a number with its floating-point form is called round off error and we have two methods for measuring approximation of errors.

Definition: If \tilde{x} is an approximation to x then

- the Absolute error is given by $e_A = |x - \tilde{x}|$ where e_A means absolute error.
- the relative error is given by

$$e_R = \frac{|x - \tilde{x}|}{|x|} \text{ if } x \neq 0 \quad \dots \quad (1.5)$$

Example: Let $x = 0.3000 \times 10^1$ and $\tilde{x} = 0.3100 \times 10^1$

$$\text{Absolute error} = e_A = |x - \tilde{x}| = |0.3 - 0.31| \times 10^1 = 0.01 \times 10^1 = 0.1$$

$$\text{Relative error} = e_R = \frac{|x - \tilde{x}|}{|x|} = 0.3333 \times 10^{-1}$$

- Let $x = 0.3000 \times 10^{-3}$ and $\tilde{x} = 0.3100 \times 10^{-3}$

$$\text{Absolute error} = e_A = |x - \tilde{x}| = 0.1 \times 10^{-4}.$$

$$\text{relative error} = e_R = \frac{|x - \tilde{x}|}{|x|} = 0.3333 \times 10^{-1}$$

- Let $x = 0.3000 \times 10^4$ and $\tilde{x} = 0.3100 \times 10^4$

$$\text{Absolute error } e_A = |x - \tilde{x}| = 0.1 \times 10^3$$

$$\text{Relative error} = e_R = \frac{|x - \tilde{x}|}{|x|} = 0.333 \times 10^{-1}$$

These examples show that the same relative error 0.3333×10^{-1} , occurs for widely varying absolute errors. Therefore, as a measure of accuracy the absolute error may be misleading leading and the relative error is more meaningful.

Note: A number correct to n decimal digits

$$\text{i.e. } x = 0.d_1d_2d_3 \dots d_nd_{n+1} \dots$$

$$\tilde{x} = 0.d_1d_2d_3 \dots d_n.$$

$$\text{has an absolute error} = e_A = |x - \tilde{x}|$$

$$= d_{n+1}.d_{n+2} \dots \times 10^{-(n+1)}$$

$$< 5 \times 10^{-(n+1)} \text{ (Rounding)}$$

$$= 0.5 \times 10^{-n}$$

$$\text{has a relative error} = e_R = \frac{|x - \tilde{x}|}{|x|} < \frac{0.5 \times 10^{-n}}{|x|}$$

$$< 5 \times 10^{-n} \text{ (rounding)}$$

$$\text{since } |x| \geq 0.1$$

1.4.3.4 Relation between the relative error

In addition to inaccurate representation of numbers during the entry in the computer the arithmetic performed inside it is not exact.

Assume that 't' and y real numbers are given and let +, -, \times , \div are Computer arithmetic operations.

Let $x = \frac{1}{3}$ and $y = \frac{5}{7}$ if inserted in a 5 digit Computer it will be $\tilde{x} = 0.33333$ and $\tilde{y} = 0.71428$.

If we apply the Computer arithmetic operations we get:

Operations	Result	Exact value	Absolute error	Relative error
$\tilde{x} + \tilde{y}$	0.10476×10^{01}	0.1046619×10^{01}	0.190×10^{-04}	0.182×10^{-4}
$\tilde{x} - \tilde{y}$	0.38085	0.38095238	0.238×10^{-05}	0.625×10^{-05}
$\tilde{x} \times \tilde{y}$	0.23809	0.238095238	0.524×10^{-05}	0.220×10^{-04}
$\tilde{x} \div \tilde{y}$	$0.21428571 \times 10^{01}$	$0.21428571 \times 10^{01}$	0.571×10^{-04}	0.267×10^{-04}

1.5 Propagation of Errors

Definition 1.1 An algorithm is a procedure that describes a finite sequence of steps to be performed in a specified order to solve a given problem.

Definition 1.2 Propagated Errors is an error in the succeeding steps of an algorithm due to an error at the initial step.

Propagated error is of critical importance.

- If errors are magnified continuously as the algorithm continues eventually they will overshadow the true value and hence destroying the validity of the algorithm and we call such algorithm as **unstable**.

If errors made at initial stage die out as the algorithm continues, the algorithm is said to be stable. Usually the initial error induced will not die out to the last stage of the algorithm. But to consider the stability of an algorithm we consider two cases:

Suppose that an error e_0 is introduced at initial stage of the algorithm and after 'n' subsequence operations of the algorithm, E_n error is resulted which is the propagated error:

- If $|E_n| \leq k n e_0$ where k is a constant the propagated error is said to be linear growth and the algorithm is stable.

Linear growth of error is usually unavoidable and is acceptable and Algorithms that show linear growth are considered to be stable.

- If $|E_n| \geq k^n e_0$ for some $k > 1$, the propagated error is exponential. Exponential growth of error should be avoided.
- And algorithms that show exponential growth are said to be unstable.

Example:

- The sequence $p_n = \left(\frac{1}{3}\right)^n$, $n > 0$ can be generated using.

1. Let $p_0 = 1$ and defining $p_n = \left(\frac{1}{3}\right)p_{n-1} \approx (0.33333)p_{n-1}$ for $n > 1$

If the sequence is generated using five digit rounding arithmetic, the results are:

n	Computed p_n	Exact l_n
0	0.10000×10^0	0.10000×10^1
1	0.33333×10^0	0.33333×10^0
2	0.11111×10^0	0.11111×10^0
3	0.37036×10^{-1}	0.37037×10^{-1}
4	0.12345×10^{-1}	0.12346×10^{-1}

The rounding error introduced by replacing $\frac{1}{3}$ by 0.33333 produces a propagated error of $(0.33333)^n \times 10^{-5}$ in the n th term of the sequence. Hence, the method of generating the sequence is stable.

2. Let us use another algorithm to generate the sequence

$$p_n = \left(\frac{1}{3}\right)^n, n > 0, \text{ define } p_0 = 1, p_1 = \frac{1}{3} \text{ and compute } p_n = \left(\frac{10}{3}\right)p_{n-1} - p_{n-2} \text{ for } n \geq 2$$

Using five-digit rounding using this algorithm the results are:

n	Computed p_n	Exact l_n
0	0.10000×10^0	0.10000×10^1
1	0.33333×10^0	0.33333×10^0
2	0.11110×10^0	0.11111×10^0
3	0.37000×10^{-1}	0.37037×10^{-1}
4	0.12230×10^{-1}	

5	0.37660×10^{-2}	0.12346×10^{-1}
		0.41152×10^{-2}

The same rounding error introduced by replacing $\frac{1}{3}$ by (0.33333) generates propagated error of $3^n (0.12500 \times 10^{-5})$ to produce p_n . Hence, the error is exponential growth and the algorithm is unstable.

1.6 Over Flow and Under Flow.

Rounding Form

$$a = 0.142862 \quad 6 = \alpha_{t+1} > 5$$

$$a^1 = 0.1429$$

$$1a^1 - a = 28 \times 10^{-5} < 5 \times 10^{-5} = 0.5 \times 10^{-5}$$

since only a finite number of places are available to express the exponent in a floating – point representation, over flow and under flow of exponent can occur.

Example:

$$t = 4, e = 2$$

$$a) \quad x = 0.31794_{10} 110$$

$$x = a \times 10^b$$

$$a^1 = 0.3179$$

$$\tilde{x} = a^1 \times 10^b \text{ but } b \text{ has digits } > 2$$

This is an example of over flow i.e. the exponent is greatly positive to fit the allotted spaces.

$$b) \quad x = 0.012345_{10} - 99$$

$$x = 0.12345_{10} - 100$$

$$a^1 = 0.1235$$

$$\tilde{x} = a^1 \times 10^b \text{ where } b \text{ has digits } > 2.$$

This is an example of under flow i.e. the exponent is too greatly negative to fit the allotted space and the value is treated as zero.

Normally over flow and under flow do not happen very frequently. Further they can be avoided to some extent by suitable scaling of the input data by incorporating special checks and rescaling during computations.

Example: 1

$$\begin{array}{rcl} & 0.000128 & \\ \text{scale} & & \\ 1 & 100 & \\ 0.000128 & 0.0128 = 0.128 \times 10^{-1} & \end{array}$$

Relation between the relative error of an approximate number to its first significant digit

The following theorem relates to the magnitude of the relative error of an approximate number to the number of correct digits.

Theorem: If a positive approximate number \tilde{x} has n correct digits the relative error e_R of this number does not exceed $\left(\frac{1}{10}\right)^{n-1}$ divided by the first significant digit of the given number.

Symbolically

$$e_R \leq \frac{1}{d_m} \left(\frac{1}{10}\right)^{n-1},$$

where d_m is the first significant digit of \tilde{x}

Proof: Let the number \tilde{x} be represented as $x = \tilde{x} = (d_m d_{m-1} d_{m-2} \dots d_1 d_0)$ base 10.

$$\tilde{x} = d_m 10^m + d_{m-1} 10^{m-1} + \dots d_{m-n+1} 10^{m-n+1} \text{ where } (d_m \geq 1)$$

be an approximate value of the exact number x which is correct to n digits.

By definition we have (using rounding)

$$e_n = |x - \tilde{x}| \leq \frac{1}{2} 10^{m-n+1} \quad \dots \quad (1.6)$$

$$\therefore -\frac{1}{2} 10^{m-n+1} \leq x - \tilde{x} \leq \frac{1}{2} 10^{m-n+1}$$

$$\therefore \tilde{x} - \frac{1}{2} 10^{m-n+1} \leq x \leq \tilde{x} + \frac{1}{2} 10^{m-n+1}$$

$$\therefore x \geq \tilde{x} - \frac{1}{2} 10^{m-n+1}$$

This inequality is further strengthened if the number x is replaced by a definitely smaller number $d_m 10^m$

$$\text{Thus } x \geq d_m 10^m - \frac{1}{2} \cdot 10^{m-n+1}$$

$$= \frac{1}{2} 10^m (2d_m - \frac{1}{10^{n-1}}).$$

The right hand side of above is a minimum for $n = 1$

$$\text{Thus } x > \frac{1}{2} 10^m (2d_m - 1)$$

$$\text{since } 2d_m - 1 = d_m + (d_m - 1) \geq d_m$$

$$\text{Hence } x > \frac{1}{2} 10^m d_m$$

$$\text{Hence } e_R = \frac{e_A}{x} \leq \frac{\frac{1}{2} 10^{m-n+1}}{\frac{1}{2} d_m 10^m}$$

$$\therefore e_R \leq \frac{1}{dm} \left(\frac{1}{10} \right)^{n-1} \quad \dots \quad (1.7)$$

Hence proved.

Exercise

1. An approximate value π is given by $x_1 = \frac{22}{7} = 3.142871$ and its true value is $x = 3.1415926$.

Find the absolute and relative errors.

2. The three approximate values of the number $\frac{1}{3}$ are given as 0.30, 0.33 and 0.34, which of the three is the best approximation.
3. Find the relative error of the number 8.6 if both of its digits are correct.
4. Evaluate the sum $= \sqrt{3} + \sqrt{5} + \sqrt{7}$ to four significant digits and find its absolute and relative errors.

5. Determine the relative error and the number of correct digits in the product

$$u = (17.63) \times (14.289)$$