ePGPathshala

Subject: Information Technology
Paper: Numerical Methods
Module 2: Sources and Types of Errors

1. Introduction

In previous module, we learned about different characteristics of numerical methods. One important characteristic of numerical methods is that solutions are supposed to be approximate in nature. We also learned about different measures of error like True Error, approximate absolute error, relative error, percentage error etc. In this module our main focus is on different sources of errors and types of errors which occur during numerical computations

2. Significant Digit

Many times, especially in determination of scientific constants, for example, Elasticity constants, gravitational constants, Heat constants, approximate solutions obtained are needed to be correct to certain number of significant digits. Before we understand, why so, let us understand the concept of significant digit. When is a digit said to be significant?

(i) Every nonzero digit is significant. If a number does not contain any zero, all digits of it are significant. Only when a number contains zeroes, number of significant digits may be different from number of decimal digits in a number. Thus, 9.5763 has 5 significant digits, 492 has 3 significant digits.

(ii) Zero may be significant or may not be significant. Zeros between non-zero digits are always significant. So, 2047 has 4 significant digits , 5<u>0</u>.<u>0</u>32 has 5 significant digits

(iii) Leading zeros, that is, zeros before non-zero digits are **<u>not</u>** significant; for example, 0.0123 has 3 significant digits namely 1, 2 and 3. Similarly, 0.0000000123 also has 3 significant digits only.

(iv) Trailing zeros, that is, zeros behind non-zero digits are **sometimes** significant; zeros after decimal point are significant. 3.000 has 4 significant digits but 3000 has only 1 significant digit as trailing zeroes are not occurring after decimal point.

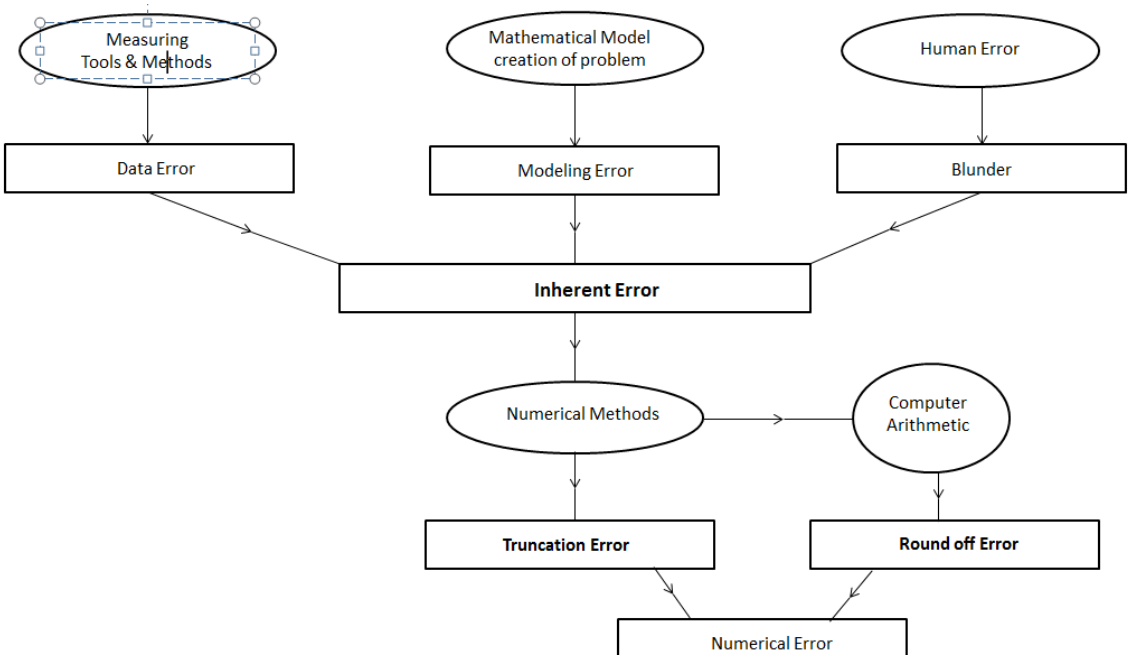
Illustration 1:

- 6.4320 has 5 significant digits (case (iv))
- 0.06432 has 4 significant digits (case (iii))
- 64 has 2 significant digits (case (i))
- 64.0 has 3 significant digits (case (iv))
- $6.432*10^4$ has 4 significant digits (case (iv))
- 6.43200 has 6 significant digits (case (iv))
- 2000 has 1 significant digits (case (iv))

Having understood, concept of significant digit, let us have a look at the situation, where answer correct to certain number of significant digits , rather than certain number of decimal digits becomes a must. When an answer falls under case (iii), answer being numerically less than 1, it would have leading zeroes. When such a constant is to be determined, one is required to give first few non-zero numbers. That time, answer correct to fixed number of decimal places may yield the answer as 0. Let us suppose the answer to be determined has value 0.0000000011457648. Answer correct to six decimal places is 0.000000 whereas answer correct to six significant digits is 0.00000000114576. On the other hand, if value 1.00636; then answer correct to three decimal places is 1.006 and answer correct to three significant places is also 1.006. If non-significant zeros are there in a number then two answers differ.

3. Sources and Types of Errors

When a mathematically formulated problem comes to numerical analyst, it is likely that it contains many different errors. They may have occurred due to Mathematical Modeling of the problem, limitations of measurement tools and methods or/and errors committed by humans unintentionally or due to their non-commitment. As such nothing can be done to reduce or remove them. These errors being inherited by the numerical analyst are called **Inherent Errors**. Two types of errors occur in implementation of numerical methods on computer, classified as **Truncation Error** and **Round off Error**.



4. Truncation Error

Truncation Error is an error in implementation of numerical approximation method occurring due to truncating a process involving infinite number of steps to finite number of steps, like:

(i) Limiting infinite series to finite number of terms

(ii) Limiting infinite number of iterations to finite number of iterations (f(x) = 0)

(iii) Taking finite step size instead of infinitesimal step size (Numerical Differentiation and Numerical Integration)

a. Limiting infinite series to finite number of terms

Consider Maclaurine series of $e^x = 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \ldots$ converges ∀ x.

Let us use the terms on the right side to determine the value of e^x . We cannot use infinite terms, cannot go on adding terms upto infinity, say we use only first four terms. Thus, approximate e^x by $e^x \approx 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!}$

Truncation Error $= e^x - (1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!})$

$= \frac{x^4}{4!} + \frac{x^5}{5!} + \ldots$ (if 4 terms are used)

Illustration 1:

Estimate $e^{0.5}$ for different no of terms and calculate relative % approximate error

(Exact value of $e^{0.5}$ upto 5 decimal places is 1.64872)

Number of Terms	Estimate of $e^{0.5}$	Absolute Approximate Error $\lvert E_a \rvert$	Relative % Approximate Error $\lvert e_a \rvert$ %	Absolute Error
1	1	----	-----	0.64872
2	1.5	0.5	33%	0.14872
3	1.625	0.125	7.69%	0.02372
4	1.645832	0.020832	1.27%	0.00289
5	1.64843617	0.00260417	0.16%	0.00028

The Taylor series expansion of $f(x)$ about a:

$$f(a)+f'(a)(x-a)+\frac{f^{(2)}(a)}{2!}(x-a)^2+\frac{f^{(3)}(a)}{3!}(x-a)^3+\ldots$$

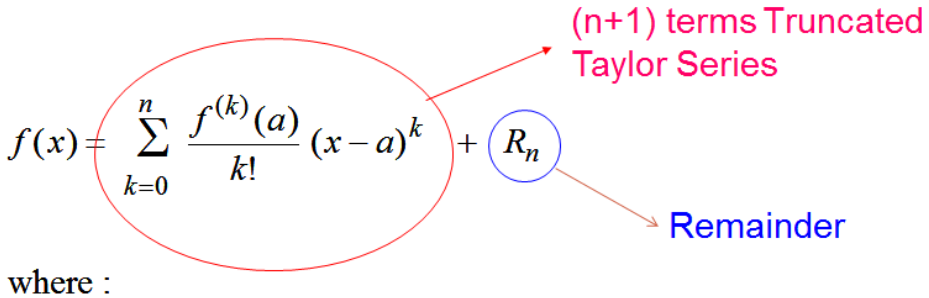
or

$$Taylor\ \ Series=\sum_{k=0}^{\infty}\frac{1}{k!}f^{(k)}(a)\ (x-a)^k$$

If the series converge, we can write :

$$f(x)=\sum_{k=0}^{\infty}\frac{1}{k!}f^{(k)}(a)\ (x-a)^k$$

If a function $f(x)$ possesses derivatives of orders $1, 2, \ldots, (n+1)$ on an interval containing a and x then the value of $f(x)$ is given by :



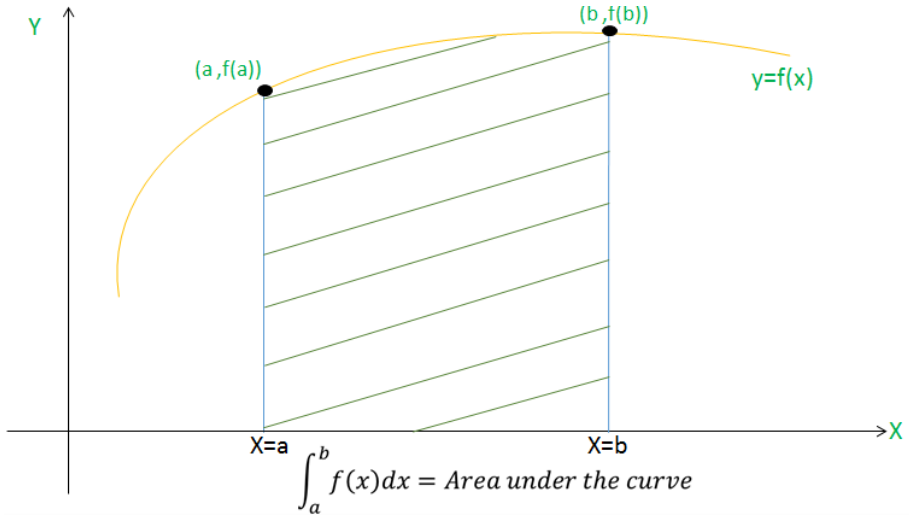
where :

$$R_n=\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1}\ \ and\ \xi\ \text{is between}\ a\ \text{and}\ x.$$

Remainder Term R_n is the Truncation error term.

b. Truncation Error in Numerical Integration

We know, numerical integration $\int_a^b f(x)\,dx$ gives area under the curve $y = f(x)$ from $x = a$ to $x = b$.

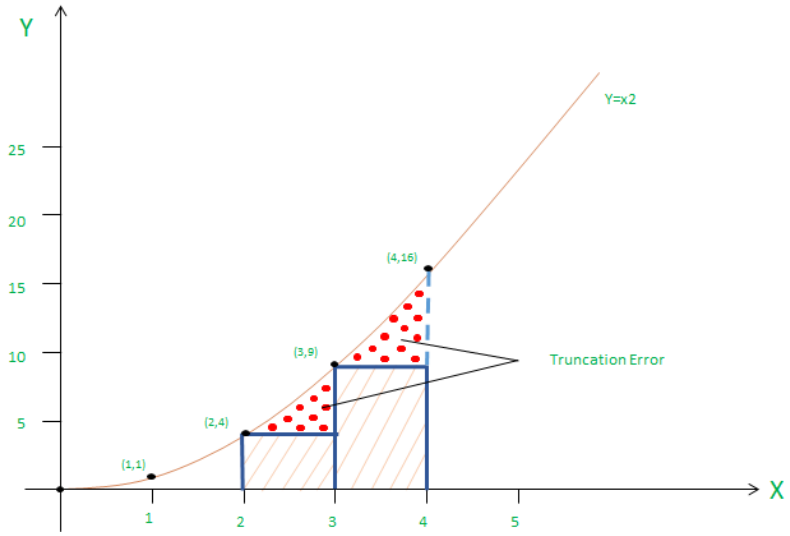


Let us calculate $\int_2^4 x^2 dx = \left[\frac{x^3}{3}\right]$, 2 to 4

$$= \frac{4^3 - 2^3}{3}$$

$$= \frac{64-8}{3} = 18.67;$$

18.67 is the exact value of $\int_2^4 x^2 dx$ rounded to 2 decimal places. Now, Let us apply numerical approximation method. For simplicity, ease, convenience and to avoid round off errors, let us take rectangles of width 1 from 2 to 4 and sum up their areas as an estimate of the integral $\int_2^4 x^2 dx$.

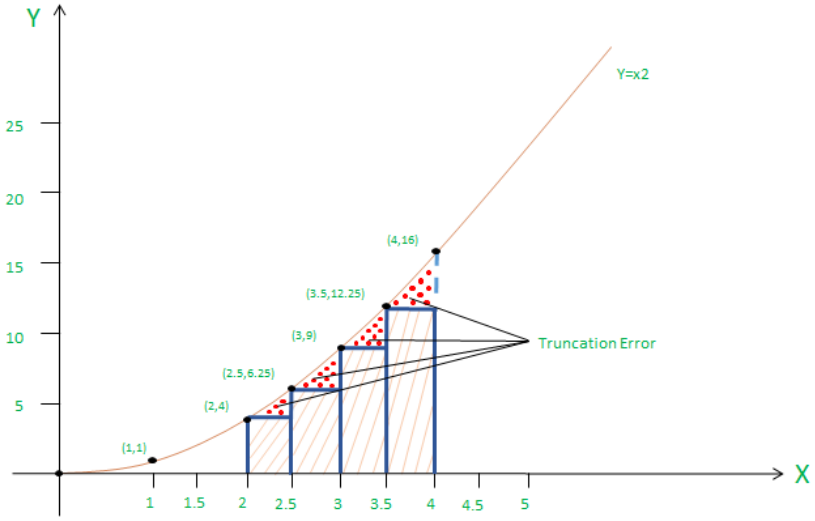


This gives us:

Estimated value I_a = 1*4 + 1* 9 = 13 (Height2^2 and 4^2)

Truncation Error = 18.67 – 13 = 5.67

Let us ask ourselves this question: What would happen, if we rework the same example with smaller width? (Smaller step size and increase the number of steps). So, let number of subintervals now in [2, 4] be 4 instead of just 2. As a result, width of each rectangle would be now 0.5. Heights would be 2^2, 2.5^2, 3^2, 3.5^2 giving 4, 6.25, 9, and 12.25 respectively.



Thus,

I_a = 0.5 (4 + 6.25 + 9 + 12.25)

= 15.75

Truncation Error = 18.67 – 15.75 = 2.92 (reduction from 5.67 to 2.92)

Let us reduce the size of subintervals still further: say now 8 subintervals of length 0.25. Then,

I_a = 0.25($2^2 + 2.25^2 + 2.5^2 + 2.75^2 + 3^2 + 3.25^2 + 3.5^2 + 3.75^2$)

= 0.25* (4+5.0625 + 6.25 + 7.5625 +9 +10.5625 +12.25 +14.0625 +17.1875)

= 0.25 * 68.75

= 17.1875

Truncation Error = 18.6667 – 17.1875 = 1.4792

So, reduction of step size (increasing number of steps) makes the method more nearer to reality and truncation error reduces.

c. Truncation Error in Iterative steps

Similarly, if we take the method of finding roots of an equation $f(x) = 0$, in Bisection method the truncation error is bounded by ; $|E| \leq \frac{|b-a|}{2^k}$, where k is the iteration step number and *a* and *b* are the end points of the interval, within which the root lies. As k $\to \infty$, $|E| \to 0$. But the method can not go on, the iteration process would be stopped after a fixed number of iterations (say *n*), leading to Truncation error $\leq \frac{|b-a|}{2^n}$; again confirming that truncation error reduces with increased number of iterations.

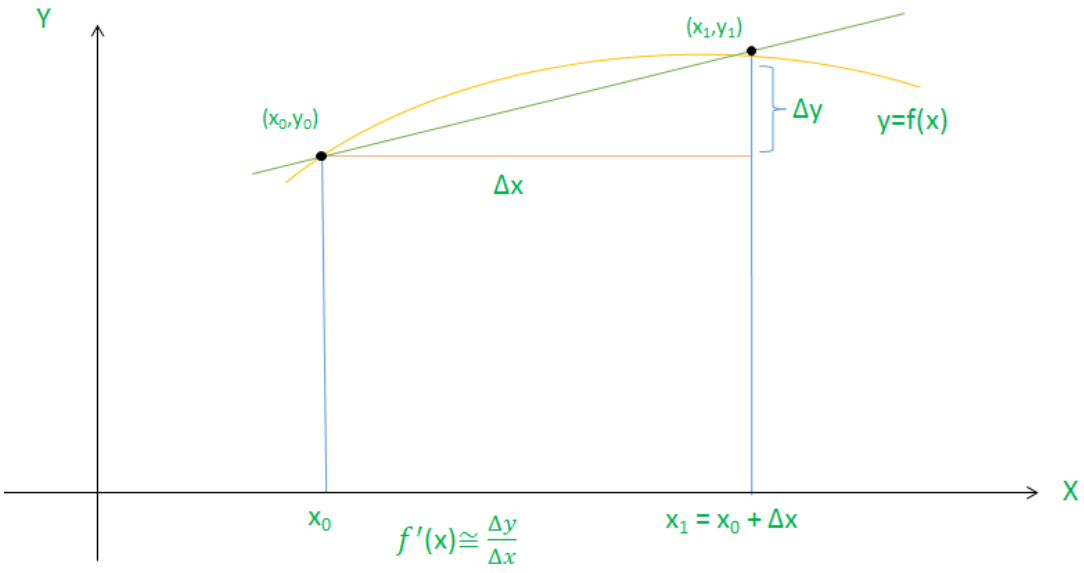
d. Truncation error in Numerical Differentiation

Again for simplicity and no round off error take $f(x) = x^3$ and let us estimate derivative at x = 2. $f'(x) = 3x^2$; $f'(2) = 3 * 4 = 12$ (True Value)

We know, derivative is defined as $f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$

Let us estimate derivative at 2 by taking *h* = 0.1

$f'(2) \cong \frac{f(2.1)-f(2)}{0.1} = \frac{2.1^3-2^3}{0.1} = \frac{1.261}{0.1}$ = 12. 61; giving $|E_a|$ = 0.61 and relative absolute error as $\frac{0.61}{12}$* 100 = 5.08%



If we reduce *h* to 0.05 and repeat the same exercise, estimate obtained is

$f'(2) \cong \frac{f(2.05)-f(2)}{0.05} = \frac{2.05^3-2^3}{0.1} = \frac{8.615125-8}{0.05}$ = 12. 3025 giving $|E_a|$ = 0.3025

and relative absolute error as $\frac{0.3025}{12}$* 100 = 2.52%

e. Observations

Truncation Error arises due to numerical approximation method being applied to solve the problem and is basically due to truncating the process to finite number of steps. As step size is reduced, Truncation Error decreases. Step size and number of steps are related by; *Reduction in step size ↔ Increase in number of steps.*

5. Round off Errors

Round off errors occur due to finite precision in a computer. A number may not always have a finite representation, e.g.

$\frac{1}{3}$ = 0.3333. . .

$\sqrt{2}$ = 1.4142135623. . .

e = 2.71828182845. . .. ;

Whereas, 2, 10, 9.32 have finite representation in decimal system. Moreover, a number having finite representation in one number system may not have finite representation in another number system, for example

$(1.1)_{10}$ = $(1.000110011001100 \ldots)_2$

One can never represent 1.1 exactly in binary system. So, let us understand first how a number is stored in a computer.

a. Floating Point representation

Let us use decimal system for illustration for our convenience. Suppose we have five boxes in addition to decimal sign for storing a number , position of decimal sign is fixed , so 562.36 is represented as

If the number to be represented is 458.9875, it has two extra digits, for which no space is available, then there are two ways of representing it. Either chop the number and store it as 458.98, or round to the nearest digit and represent it as

458.99 (rounding). Error due to rounding 458.99-458.9875 = .0025 and relative error is .0025/458.9875 = 0.000545%

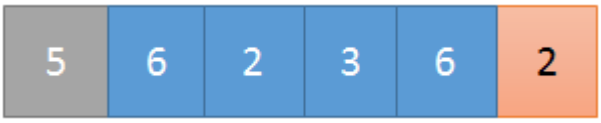
Now, let us take the other case:

If the number to be represented is 5.9875, it also has two extra digits, for which no space is available; then again, there are two ways of representing it. Either chop the number and store it as 5.98, or round to the nearest digit and represent it as 5.99 (rounding).

Error due to rounding 5.99-5.9875 = .0025 and relative error is .0025/5.9875 = 0.041754%. The point to be noted is, though rounding off error is same, but relative error has increased.

b. Normalized Floating Point Representation

Normalized Floating Point Representation is designed to keep the relative errors of the same order for small and large numbers and also to be able to store higher range in the same space. We can express 562.36 as + 5.6236 * 10^2 and 0.0056236 as 5.6236 * 10^{-3}.



This is the base of Scientific notation. In this notation, exactly one non-zero digit appears before decimal point. Its advantage is, its efficiency in representing very small or very large numbers and relative error in representation of large and small numbers are of the same order.

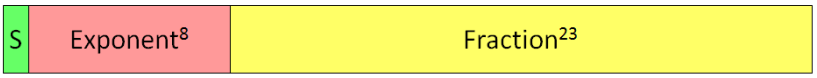
$$\underset{sign}{\pm} \quad \underbrace{d \;.\; f_1 \; f_2 \; f_3 \; f_4}_{mantissa} \quad \times \quad \underset{exponent}{10^{\pm n}}$$

$$d \neq 0 \,, \quad \pm n : \text{signed exponent}$$

c. IEEE 754 Floating-Point Standards

Single Precision (32-bit representation) is as follows:

1-bit Sign + 8-bit Exponent + 23-bit Fraction (Mantissa)



Double Precision (64-bit representation) is as follows:

1-bit Sign + 11-bit Exponent + 52-bit Fraction

6. Error Propagation

Every arithmetic computation in digital arithmetic leads to compounding of rounding off errors as follows:

$E_{a+b} \cong E_a + E_b + r_a$

$RE_{a*b} \cong RE_a * RE_b + r_m$

Rounding off Error increase with increase in computations, the higher the number of steps, the more shall be the rounding error.

For example, suppose we wish to compute: 3.578 * 2.139, using a calculator with two-digit fractions. Then

3.57	*	2.13	=	7.60

True answer: 7.653342

7. Total Numerical error (Computational)

Total Numerical Error = Truncation error + Round-off error

The only advisable way to reduce round-off errors is to increase number of significant digits. The round-off error increases due to either subtractive cancellation or due to increase in the number of computations in an analysis (smaller step size). The truncation errors can be reduced by decreasing step size. So, determining appropriate step size is essential in order to balance truncation and round-off errors to minimize the total error.

8. Control of Numerical errors

In most practical cases, we do not know the exact error associated with numerical methods. Also, there are no systematic and general approaches to evaluate numerical errors for all problems. However, there are several practical programming guidelines for controlling numerical errors.

(i) Avoid subtracting two nearly equal numbers. Try to rearrange the formula if possible to avoid subtractive cancellation.

(ii) When adding or subtracting numbers, it is best to sort the numbers and work with smallest numbers first. This avoids loss of significance.

(iii) Attempt to predict total numerical errors by theoretical formulations. This may get quite complicated and therefore can be attempted for only small scale tasks.

(iv) Estimate the accuracy of your results by seeing if the results obtained satisfy some condition or equation as a check.

(v) One should be prepared to perform numerical experiments to increase one's awareness of computational errors and possible ill-conditioned problems. (If the problem is ill conditioned, a small error in initial data creates large errors in the answer. So, inherent errors in ill conditioned problems can create havoc.) Such experiments may involve repeating the computation with a different step-size, or method and comparing the results.

(vi) As learned by now, any Numerical computation is susceptible to different types of errors. The errors may be related to required initial data. And hence it is essential to carry out sensitivity analysis for such methods. (By what amount the answer changes as a result of change in inputs). If making small changes in the input data leads to large change in the solution then such a computation is called numerically unstable. One needs to take appropriate measures to ensure that the methods that we use are relatively stable.

****