Approximations and Errors in numerical computing (1) (2)

Objective:

Approximations and errors are an indispensable piece of human life. They are all over the place and unavoidable. In numerical methods, we additionally can't disregard the presence of error. Blunders come in assortment of structures and sizes; some are avoidable, some are most certainly not. For instance, data conversion and round off mistakes can't be evaded, yet human error can be dispensed with. Although certain error can't be wiped out totally, we must tried to minimize these errors for our final solutions. It is accordingly basic to know how mistakes emerge, how they develop during the numerical procedure, and how they influence accuracy of an answer. The main objective of this lecture is to understand the major sources of errors in Numerical Method.

Numbers and Their Accuracy:

There are two kinds of numbers:

1. Exact Number
2. Approximate Numbers

Exact Numbers:

2, 1/3, 100 etc are exact numbers because there is no approximation or uncertainty associated with them. Π, √2 etc are also exact numbers when written in this form.

Approximate Numbers:

An approximate number is a number which is used as an approximation to an exact number and differs only slightly from the exact number for which it stands. For example, an approximate value of ∏ is 3.14 or if we desire a better approximation, it is 3.14159265. But we cannot write the exact value of ∏.

Significant Digits:

The digits that are used to express a number are called significant digits (or figures).

- ❑ A significant digit/figure is one of the digits 1,2, ... 9 and 0 is a significant figure except when it is used to fix the decimal point or to fill the places of unknown or disorder digits.
- ❑ The following notion describe how to count significant digits:

1. All non-zero digits are significant.
2. All zeros occurring between non-zero digits are significant digits.
3. Trailing zeros following a decimal point are significant. For example, 3.50, 65.0 and 0.230 have three significant digits each.
4. Zeros between the decimal point and preceding a non-zero digits are non significant. For example, the following numbers have four significant digits: 0.0001234, 0.001234, 0.01234
5. When the decimal point is not written, trailing zeros are ambiguous (generally. not considered to be significant). We can avoid the ambiguity by writing the number in the

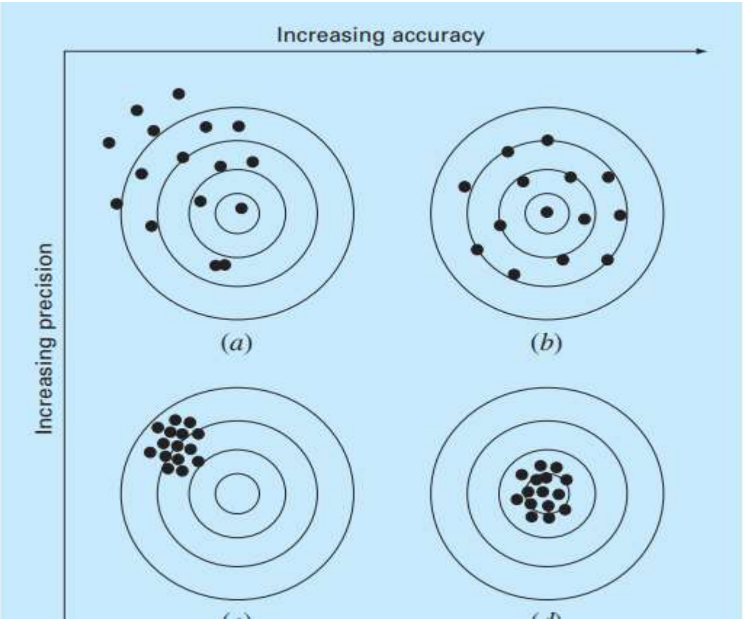
scientific notation. For example, 4500 may be written as 45 x 10^2 contains 2 significant digits. However, 4500.0 contains *five* significant digits. Further examples are:

7.56 x 10^4 has three significant digits
7.560 x 10^4 has four significant digits
7.5600 x 10^4 has five significant digits

Accuracy and Precision

In terms of their accuracy and precision, the errors associated with both calculations and measure ments can be described.The definition of precision and accuracy is closely linked to significant numbers:

- Accuracy refers to how closely a computed or measured value agrees with the true value. For example, the number of significant digits in a value. The number 57.345 is accurate up to five significant digits.
- Precision refers to how closely a computed or measured value agrees with each other. For example, the number of decimal position, i.e. the order of magnitude of last digit in a value. The number 57.396 have a precision of 0.001.



(2)

An example from marksmanship illustrating the concepts of accuracy and precision. (*a*) Inaccurate and imprecise; (*b*) accurate and imprecise; (*c*) inaccurate and precise; (*d*) accurate and precise.

Rules of Rounding off:

We come across numbers in numerical computation that have large numbers of digits and it will be necessary to cut them to manageable numbers of figures. This process is referred to as rounding off. The error caused by a large number cut-off into usable figure number is called a round-off error.

For example, 3.14159265... rounded to the nearest thousandth is 3.142. That is because the third number after the decimal point is the thousandths place, and because 3.14159265... is closer to 3.142 than 3.141.

Banker's Rounding Rule

In Banker's rounding rule (**also known as "Gaussian rounding"**) the value is rounded to the **nearest even number**. Banker's rounding is the default method for Delphi, VB.NET and VB6. It follows the specification of IEEE Standard 754. The rule is as follows:

To round off a number to n significant digits, discard all the digits to the right of the n^{th} significant digit and check the $(n+1)^{th}$ significant digit.

a) if it is less than 5, leave the n^{th} significant digit unaltered
b) if it is greater than 5, add 1 to the n^{th} significant digit
c) If it is exactly 5 then leave the n^{th} significant digit unaltered if it is an even number, but increase it by 1 if it is an odd number.

- See the following links:
  http://www.cs.umass.edu/~weems/CmpSci535/535lecture6.html

  http://www.rit.edu/~meseec/eecc250-winter99/IEEE-754references.html

  http://www.gotdotnet.com/Community/MessageBoard/Thread.aspx?id=260335

Example: The following numbers are rounded to 4 significant digits.

1.6583 → 1.658
30.0567 → 30.06
0.859**4**58 → 0.8594
3.14**1**59 → 3.142
3.14**3**58 → 3.144

Error in Numerical Methods:

Numerical errors arise from the use of approximations to represent exact mathematical operations and quantities.

True value = approximation + error
Error (E) = True Value – approximation

Sources of Errors

A number of different types of errors arise during the process of numerical computing. All these errors contribute to the total error in the final result.

1. Inherent Errors: Inherent errors are those present in the data provided to the model (also known as input errors). It contains two components, data errors and errors of conversion.

Data Errors:

Data errors (***also known as statistical errors***) occur when some experimental methods obtain dat a for a problem and are therefore of limited precision and reliability. This may be due to some limitation in instrumentation & reading and may therefore be inevitable.

Conversion Error:

Conversion errors (***also known as errors of representation***) occur due to the computer's limitation to store the data accurately. We know that there is only a specified number of digits in the floating point representation. The unretained digits are the round-up error.

Example: Represent the decimal number 0.1 and 0.4 in binary number form with an accuracy of 8 binary digits. Add them and then convert the result back to the decimal form.

Solution:

0.1_{10} = 0001 1001
0.4_{10} = 0110 0110
Sum = 0111 1111
$= 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7} + 2^{-8}$
$= 0.25 + 0.125 + 0.0625 + 0.03125 + 0.015625 + 0.0078125 + 0.00390625$
$= 0.49609375$

2. Numerical Errors:

Numerical errors were created during the process of implementing a numerical procedure (***also known as procedural errors)***. These come in two ways round off errors and and truncation errors.

a) **Round off error:**
Round-out errors occur when the exact numbers are represented by a fixed number of digits. Since the numbers are stored at every stage of computation, round off error is introduced at the end of every arithmetic operation.
Rounding a number can be dome in two ways-- chopping and symmetric rounding. Some systems use the chopping method while others use symmetric rounding.

i) **Chopping:** In chopping, the extra digits are dropped. This is called truncating the number. For example, 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.
$x = 42.7893$
$= 0.427893 \times 10^2$
$= (0.4278 + 0.000093) \times 10^2$
$= (0.4278 + 0.93\times 10^{-4}) \times 10^2$

This can be expressed in general form as:

True x $= (f_x + g_x \times 10^{-d}) \times 10^E$
$= f_x \times 10^E + g_x \times 10^{E-d}$
$=$ approximate x + error

- In chopping, Error $= g_x \times 10^{E-d}$; $0 \leq g_x < 1$, where d is the length of the mantissa, E is the exponent and g_x is the truncated part of the number in normalized form.
- In chopping, absolute error $\leq 10^E$.

ii) Symmetric Round off: In the symmetric Round off 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.

- In symmetric round off, when $g_x < 0.5$
  True x $= f_x \times 10^E + g_x \times 10^{E-d}$
  Approximate x $= f_x \times 10^E$
  Error $= g_x \times 10^{E-d}$
- In symmetric round off, when $g_x \geq 0.5$
  True x $= (f_x + 10^{-d}) \times 10^E = f_x \times 10^E + 10^{-d} \times 10^E$
  Error $= [f_x \times 10^E + g_x \times 10^{E-d}] - [f_x \times 10^E + 10^{-d} \times 10^E]$
  $= (g_x - 1) \times 10^{E-d}$
- In symmetric round off, absolute error $\leq 0.5 \times 10^{E-d}$.
- Sometimes banker's rounding rule is used for symmetric round off.

Example: Find the round off error in storing the number 752.6835 using a four digit mantissa.

Solution:

True x $= 752.6835$
$= 0.7526835 \times 10^3$
$= (0.\ 7526 + 0.0000835) \times 10^3$
$= (0.\ 7526 + 0.835 \times 10^{-4}) \times 10^3$
$= 0.\ 7526 \times 10^3 + 0.835 \times 10^{-1}$

Chopping method

Approximate x $= 0.\ 7526 \times 10^3$
Error $= 0.835 \times 10^{-1}$

Symmetric Round off

Approximate x $= 0.\ 7527 \times 10^3$
Error $= (g_x - 1) \times 10^{E-d} = (0.835 - 1) \times 10^{-1} = -\ 0.0165$

b) **Truncation Errors:** Truncation errors arise from using an approximation instead of a precise mathematical method. Usually, this is the error that occurs from the numerical system truncation. To approximate the sum of an infinite series, we often use a finite number of terms. For example, consider the following infinite series:

$$\sin(x) = x - x^3/ 3! + x^5 / 5! - x^7 / 7! + \dots \dots \dots \dots$$

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 term "truncated" introduces an error which is called truncation error.

Example: Find the truncation error in the result of the following function for x = 1/5 when we use first three terms.

$$e^x = 1 + x + x^2/ 2! + x^3 / 3! + x^4 / 4! + x^5 / 5! + x^6 / 6!$$

Solution:

$$\text{Truncation error} = x^3 / 3! + x^4 / 4! + x^5 / 5! + x^6 / 6! = 0.1402755 \times 10^{-2}$$

3. **Modeling Errors:**

Modeling errors occur in the formulation of mathematical models due to some simplifying assumption. For example, when designing a model to measure the force acting on a falling body, we may not be able to properly estimate the coefficient of air resistance or determine the direction and magnitude of wind force acting on the body, etc. 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 model which is called Modeling error.

Through adding more functionality, we can reduce modeling errors through improving or extending the models. But the improvement may take the model more difficult to solve or may take more time to implement the solution process. It is also not always true that an enhanced model will provide better results. On the other hand, an oversimplified model may produce a result that is unacceptable. It is, therefore, necessary to strike a balance between the level of accuracy and the complexity of the model.

4. Blunders: Blunders are errors that are cause due to human imperfections. 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. Some common types of errors are:

1. Lack of understanding the problem
2. Wrong assumption
3. Selecting a wrong numerical method for solving the mathematical model
4. Making mistakes in the computer program
5. Mistake in data input
6. Wrong guessing of initial values

Different Types of Error:

Absolute Error:

The absolute error is the absolute difference between the true value of a quantity and its approximate value as given or obtained by measurement or calculation. Thus, if X_t is the true value of a quantity and X_a is its approximate value, then the absolute error E_a is given by:

$$E_a = | X_t - X_a |.$$

Relative Error:

The relative error is nothing but the "normalized" absolute error. The relative error E_r is defined as follows:

$$Relative\ Error(\mathrm{Er}) = \left|\frac{Absolute\ Error\ (\mathrm{Ea})}{True\ Value\ (X)}\right|$$

More often, the quantity that is known to us is X_a and, therefore, we can modify the above relation as follows: $E_r = | X_t - X_a | / | X_t |$

Percent Relative Error:

The percent relative error is 100 times the relative error. It is denoted by Ep and defined by: $E_p = E_a \text{ x } 100$

$$Percentage\ Error(\mathrm{Ep}) = \left|\frac{Absolute\ Error\ (\mathrm{Ea})}{True\ Value\ (X)}\right| * 100\%$$

- ❑ Relative and Percent relative errors are independent of the unit of measurement, whereas absolute errors are expressed in terms of the unit uses.

Absolute and Relative Accuracy

Let ΔX is the number such that $| X_t - X_a | \leq \Delta X$ then ΔX is an upper limit on the magnitude of the absolute error is said to measure absolute accuracy.

Similarly, the quantity, $\Delta X / | X_t | \approx \Delta X / | X_a |$ measures the relative accuracy.

$$Relative\ Accuracy\ (R_A) = \frac{Absolute\ Error\ (\mathrm{Ea})}{|True\ Value\ (X)|} \approx \frac{Absolute\ Error\ (\mathrm{Ea})}{|Approximate\ Value\ (X_A)|}$$

- ❑ **If the number X is rounded to N decimal places, then**

$$\Delta X = ½ * 10^{-N}$$

- ***Example***: If X = 0.51 and is correct to 2 decimal places then $\Delta X = 0.005$. The relative accuracy is given by $0.005/0.51 \approx 0.98 = 98$ %

Example:

Suppose that you have the task of measuring the lengths of a bridge and a rivet and come up with 9999 and 9 cm, respectively. If the true values are 10,000 and 10 cm, respectively, compute:
(*a*) the true absolute error and
(*b*) the true percent relative error for each case.

Solution:

(a) The Error for measuring the bridge is: Et = 10,000-9999 = 1 cm.
The Error for measuring the bridge is: Et = 10-9 = 1 cm.
(b) The percent relative error for the bridge is: €t = 1/1000 *100% = 0.01%
For the rivet it is: €t = 1/10*100% = 10%

Thus, although both measurements have an error of 1 cm, the relative error for the rivet is much greater. We would conclude that we have done an adequate job of measuring the bridge, whereas our estimate for the rivet leaves something to be desired.

Machine Epsilon:

The round off error introduced in a number when it is represented in floating point form is given by, Chopping error = $\boldsymbol{g * 10^{E-d}, 0 \le g < 1}$

Where,

g: truncated part of the number in normalized form,

d: is the number of digits permitted in the mantissa, and

E: is the exponent.

The absolute relative error due to chopping is then given by, $\mathbf{E_r = |(g * 10^{E-d}) / (f * 10^E)|}$

The relative error is maximum when *g* is maximum and *f* is minimum. We know that the maximum possible value of *g* is less than 1.0 and minimum possible value of *f* is 0.1. The absolute value of the relative error therefore satisfies:

$$\mathbf{E_r \le |(1.0 * 10^{E-d}) / (0.1 * 10^E)| = 10^{-d+1}}$$

The maximum relative error given above is known as *machine epsilon*. The name “machine” indicates that this value is machine dependent. This is true because the length of mantissa *d* is machine dependent.

For a decimal, machine that use chopping, **Machine epsilon** $\boldsymbol{\epsilon = 10^{-d+1}}$

Similarly, for a machine which uses symmetric round off,

$$\mathbf{E_r \le |(0.5 * 10^{E-d})/(0.1 * 10^E)| = ½ * 10^{-d+1}}$$

And therefore **Machine epsilon $\epsilon = ½ * 10^{-d+1}$**

It is important to note that the machine epsilon represents upper bound for the round off error due to floating point representation. It also suggests that data can be represented in the machine with *d* significant decimal digits and the relative error does not depend in any way on the size of the number.

More generally, for a number x represented in a computer,

Absolute error bound = $|x| * \epsilon$

For a computer system with binary representation, the machine epsilon is given by

Chopping: Machine epsilon $\epsilon = 2^{-d+1}$

Symmetric rounding: Machine epsilon $\epsilon = 2^{-d}$

Here we have simply replaced the base 10 by base 2, where *d* indicates the length of binary mantissa in bits.

We may generalize the expression for machine epsilon for a machine, which uses base *b* with d-digit mantissa as follows:

Chopping: Machine epsilon $\epsilon = b * b^{-d}$

Symmetric rounding: Machine epsilon $\epsilon = b/2 * b^{-d}$

Error propagation

Numerical computing involves a series of computations consisting of basic arithmetic operations. Therefore, it is not the individual round off errors that are important but the final error on the result. Our major concern is how an error at one point in the process propagates and how it effects the final total error.

Addition and Subtraction:

Consider addition of two numbers, say, x and y.

$$x_t + y_t = x_a + e_x + y_a + e_y = (x_a + y_a) + (e_x + e_y)$$

Therefore,

Total error = $e_{x+y} = e_x + e_y$

Similarly, for subtraction

Total error = $e_{x-y} = e_x - e_y$

The addition $e_x + e_y$ does not mean that error will increase in all cases. It depends on the sign of individual errors. Similar is the case with subtractions.

Since we do not normally know the sign of errors, we can only estimate error bounds. That is, we can say that $|e_{x\pm y}| \leq |e_x| + |e_y|$

Therefore, the rule for addition and subtraction is: *the magnitude of the absolute error of a sum (or difference of the absolute errors of the operands.*

- This inequality is called *triangle inequality*.

Multiplication:

Here, we have

$$x_t * y_t = (x_a + e_x) * (y_a + e_y) = x_a y_a + y_a e_x + x_a e_y + e_x e_y$$

Errors are normally small and their products will be much smaller. Therefore, if we neglect the product of the errors, we get

$$x_t * y_t = x_a y_a + x_a e_y + y_a e_x = x_a y_a + x_a y_a (e_x/x_a + e_y/y_a)$$

Then,

$$\text{Total error} = e_{xy} = x_a y_a (e_x/x_a + e_y/y_a)$$

Division:

We have

$$x_t / y_t = (x_a + e_x) / (y_a + e_y)$$

Multiplying both numerator and denominator by $y_a - e_y$ and rearranging the terms, we get

$$x_t / y_t = (x_a y_a + y_a e_x - x_a e_y - e_x e_y) / (y_a^2 - e_y^2)$$

Dropping all terms that involve only product of errors, we have

$$x_t / y_t = (x_a y_a + y_a e_x - x_a e_y) / y_a^2 = x_a/y_a + x_a/y_a (e_x/x_a – e_y/y_a)$$

Thus,

$$\text{Total error} = e_{x/y} = x_a/y_a (e_x/x_a – e_y/y_a)$$

- Applying the triangle inequality theorem, we have

$$e_{x/y} \leq |x_a/y_a| (|e_x/x_a| + |e_y/y_a|)$$

$$e_{xy} \leq |x_a y_a| (|e_x/x_a| + |e_y/y_a|)$$

- Note: The final errors (after arithmetic operations) e_{x+y}, e_{x-y}, e_{xy}, and $e_{x/y}$ are expressed in terms of only e_x and e_y and do not contain the round off errors introduced by the operations themselves. This results from the final error in floating point representation. Therefore, we must add the round off error in doing the operation in each case. For example, $e_{x+y} = e_x + e_y + e_0$

- **Now, we can have relative errors for all the four operations as follows:**

Addition & Subtraction:

$e_{r,\, x\pm y} \leq (|e_x| + |e_y|) / (|\, x_a \pm y_a \,|)$

Multiplication & Division:

$e_{r,\, xy} = |\, e_{r,\, x} \,| + |\, e_{r,\, y} \,|$

$e_{r,\, x/y} = |\, e_{r,\, x} \,| + |\, e_{r,\, y} \,|$

Example: Estimate the relative error in z = x – y when x = $0.1234*10^4$ and y = $0.1232*10^4$ as stored in a system with four–digit mantissa.

Solution: We know

$e_{r,z} \leq (|e_x| + |e_y|) / (|\, x_a - y_a \,|)$

Since, the number x and y are stored in a four-digit mantissa system they are properly rounded off and therefore,

$e_{r,x} \leq ½ * 10^{-3} = 0.05\%$

$e_{r,y} \leq ½ * 10^{-3} = 0.05\%$

Then

$e_x \leq 0.1234 * 10^4 * 0.5 * 10^{-3} = 0.617$

$e_y \leq 0.1232 * 10^4 * 0.5 * 10^{-3} = 0.616$

Therefore

$|e_z| \leq |e_x| + |e_y| = 1.233$

$|e_{r,z}| \leq (1.233 * 10^{-4}) / |\, 0.1234 – 0.1232 \,| = 0.6165 = \mathit{61.65\%}$

Example: If ΔX = 0.005 and ΔY = 0.001 be the absolute error in X = 2.11 and Y = 4.15. Find the relative error in the computation of x + y.

Solution: Here, X = 2.11   Y = 4.15

$\therefore\; x + y = 2.11 + 4.15 = 6.26$

and ΔX = 0.005   ΔY = 0.001

$\therefore \Delta X + \Delta Y = 0.005 + 0.001 = 0.006$

Relative error is $E_R = 0.006 / 6.26 = 0.000958 \approx 0.001$

CONVERGENCE

- Numerical computing is based on the idea of iterative process.
- Iterative processes involve generation of sequence of approximation with the hope that, the process will end of the required solution.
- Certain methods convert faster than others.
- It is necessary to know that convergence rate of any method to get the required solution.
- Rapid convergent take less execution time.

References:

1. **BalaGurushamy, E.** *Numerical Methods.* New Delhi : Tata McGraw-Hill, 2000.

2. **Steven C.Chapra, Raymon P. Cannale.** *Numerical Methods for Engineers.* New Delhi : Tata McGRAW-HILL, 2003. ISMN 0-07-047437-0.