Approximations and Errors in numerical computing

Approximations and errors are an integral part of human life. They are everywhere and unavoidable. In numerical methods, we also cannot ignore the existence of errors.

Errors come in variety of forms and sizes; some are avoidable, some are not. For example, data conversion and round off errors cannot be avoided, but human error can be eliminated. Although certain errors cannot be eliminated completely, we must at least know the bounds of these errors to make use our final solution. It is therefore essential to know how errors arise, how they grow during the numerical process, and how they affect the accuracy of a solution.

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 / Significant figures
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.

How to count significant digits
The following rules describe the notion of 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 45×10^2 contains 2 significant digits. However, 4500.0 contains *five* significant digits. Further examples are:
   7.56×10^4 has three significant digits
   7.560×10^4 has four significant digits
   7.5600×10^4 has five significant digits

- A useful link: http://mathforum.org/library/drmath/view/57160.html
- The concept of accuracy and precision are closely related to significant digits.
  1. *Accuracy* refers to the number of significant digits in a value. For example, the number 57.396 is accurate to five significant digits (sd).

2. *Precisions* refers to the number of decimal positions i.e. the order of magnitude of the last digit in a value. The number 57.396 has a precision of 0.001 or 10^{-3}.

Example- 4.1 & 4.2: Page- 63, Balagurusamy.

Rounding a Number

A method of approximating a number using a nearby number at a given degree of accuracy. 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 nth significant digit and check the (n+1) th significant digit.

a) if it is less than 5, leave the nth significant digit unaltered
b) if it is greater than 5, add 1 to the nth significant digit
c) If it is exactly 5 then leave the nth 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.

$1.6583 \rightarrow 1.658$
$30.0567 \rightarrow 30.06$
$0.859458 \rightarrow 0.8594$
$3.14159 \rightarrow 3.142$

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. A taxonomy of errors encountered in a numerical process is shown in Fig-4.1 [page-61, Balagurusamy] which shows that every stage of the numerical computing cycle contributes to the total error.

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

a) **Data Errors:** Data errors (also known as *empirical errors*) 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 limitation in instrumentation & reading, and therefore may be unavoidable

b) **Conversion Error:** Conversion Errors (also known as *representation errors*) arise due to the limitation 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.

Example 4.3: Page-64, Balagurusamy.

2. 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.

a) **Roundoff error:** 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.
   Rounding a number can be dome 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.
   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.
      - In chopping, Error = g_x x 10^{E-d} 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 $\leq 10^{E-d}$.
   ii) **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.
      - In symmetric roundoff, error = g_x x 10^{E-d} when $g_x < 0.5$
        error = (g_x-1) x 10^{E-d} when $g_x \geq 0.5$
      - In symmetric roundoff, absolute error ≤ 0.5 x 10^{E-d}.
      - ❑ Sometimes banker's rounding rule is used for symmetric roundoff.
      - ❑ Example 4.4: Page-66, Balagurusamy.

b) **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 numerical process.
   We often use some finite number of term to estimate the sum of an infinite series. 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 4.5: Page – 68, Balagurusamy.

3. Modelling Errors: Modelling errors arise due to certain simplifying assumption in the formulation of mathematical models. 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 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 model which is called Modelling error.
We can reduce modelling errors by refining or enlarging the models by incorporating more features. But the enhancement 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

Absolute and Relative Errors
An error is usually quantified in two different but related ways. One is known as absolute error and the other is called relative 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:

$$E_r = \frac{\text{Absolute Error}}{|\text{ True Value }|} = \frac{| X_t - X_a |}{| X_t |}$$

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

Percent Relative Error: The percent relative error is 100 times the relative error. It is denoted by Ep and defined by: $E_p = E_a \times 100$

Example 4.7: Page-71, Balagurusamy.

- ❑ 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 $| X_t – X_a | \le \Delta X$ 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.

- If the number X is rounded to N decimal places, then
  $\Delta X = \frac{1}{2} \times 10^{-N}$
- Example: If X = 0.51 and is correct to 2 decimal places then $\Delta X = 0.005$. The relative accuracy is given by $0.005/0.51 \approx 0.98 = 98\ \%$

Machine Epsilon:

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

$$E_r = \left| (g * 10^{E-d}) / (f * 10^{E}) \right|$$

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

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

- 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 $\epsilon = 10^{-d+1}$

Similarly, for a machine which uses symmetric round off,

$$E_r \leq \left| (0.5 * 10^{E-d})/(0.1 * 10^{E}) \right| = \frac{1}{2} * 10^{-d+1}$$

And therefore Machine epsilon $\epsilon = \frac{1}{2} * 10^{-d+1}$

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 $\epsilon = b/2 * b^{-d}$

Example 4.8: Page –72, Balagurusamy.