ME 261: Numerical Analysis

Lecture-2: Approximation & Error

Md. Tanver Hossain
Department of Mechanical Engineering,
BUET
http://tantusher.buet.ac.bd

Floating Point Arithmetic

- fractional quantities are typically represented in computers using floating point format
- this approach is very much similar to scientific notation
- for example, fixed point number 17.542 is the same as the floating point number $.17542*10^2$ which is often displayed as .17542e2
- another example, -.004428 is same as $-.4428*10^{-2}$
- General form of floating-point number in Computers:

$$\pm .d_1 d_2 d_3 \ldots . d_p * B^e$$

where, d_j's are digits or bits with values from 0 to B-1
B=the number base that is used, e.g. 2, 10,16,8
P=number of significand bits (digits), that is, the precision.
e=an integer exponent, ranging from E_{min} to E_{max}

- $d_1 d_2 d_3 \ldots d_p$ constitute the fractional part of the number

Floating Point Arithmetic

Normalization:

- the fractional digits are shifted and the exponents are adjusted so that d_1 is NONZERO

  e.g.

  $-.004428 \cong -0.4428*10^{-2}$ is the normalized form

- In base-2(binary) system normalization means *the first bit is always 1*

IEEE Standard for the floating-point numbers:

- consists of a set of binary representations of real numbers
- normalization gives the advantage that the first bit does not need to be stored
- Hence, nonzero binary floating point numbers can be expressed as,

$$\pm (1+f)*2^e$$

where f =the mantissa (or the fraction)

e= exponent

Floating Point Arithmetic

- e.g. the decimal number 9, which is 1001 in binary, would be stored as,

$$+1.001 * 2^3$$

although the original number has 4 significant digits, we only have to store the 3 fractional bits: .001

- There are three commonly used levels of precision for floating point numbers:
  - single precision (32 bits)
  - double precision (64 bits)
  - long double (80 bits)

The bits are divided as follows:

precision	sign	exponent	mantissa
single	1	8	23
double	1	11	52
long double	1	15	64

Floating Point Arithmetic

- By default, MATLAB has adopted IEEE double precision format

Figure: The manner in which a floating point number is stored in IEEE double precision format

Note: because of normalization, 53 bits can be stored in mantissa.

- Sign bit is 0 for positive numbers and 1 for negative numbers

The decimal number $9.4 = (1001.\overline{0110})_2 = (1.00\overline{10110})_2 * 2^3$

Normalized

It will be stored in exponent

+1. 00101100110011001100110011001100110011001100110011001100 110...

f (52 bit)

53rd bit

How do we fit the infinite binary number representing 9.4 in a finite number of bits?

Rounding:

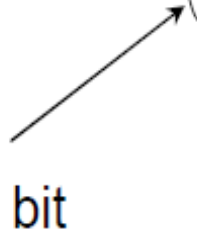
- In base 10, numbers are customarily rounded up if the next digit is 5 or higher, and rounded down otherwise
- In binary, this corresponds to rounding up if the bit is 1
- Specifically, the important bit in the double precision format is the 53rd bit

IEEE Rounding to Nearest Rule:

if bit 53 is 0,then truncate after the 52nd bit (round down)
if bit 53 is 1, add 1 to 52nd bit (round up)
exception: if the bits 53rd and beyond are like 10000000...,
then add 1 to 52nd bit if and only if 52nd bit is 1.

For the number (9.4) discussed previously, the 53rd bit in mantissa is 1

+1. 0010110011001100110011001100110011001100110011001100 110…



53rd bit

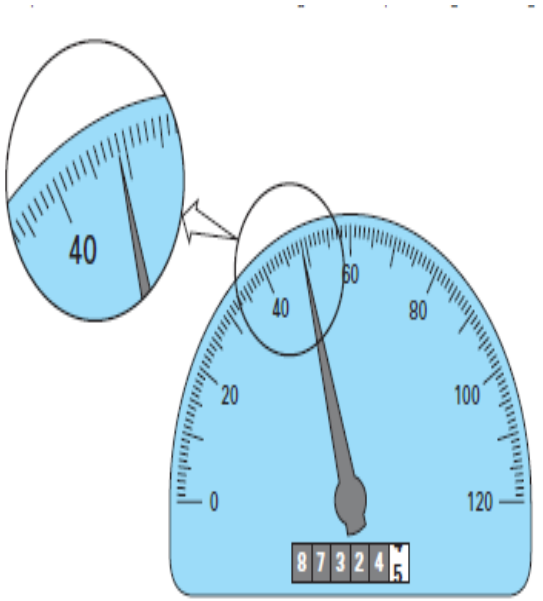
So, the Rounding to the Nearest Rule says to round up, i.e. to add 1 to bit 52

Therefore the mantissa of floating point number that represents 9.4 is

+1. 0010110011001100110011001100110011001100110011001101

SIGNIFICANT FIGURES

- Whenever we employ a number in a computation, we must have assurance that it can be used with confidence.
- The concept of a significant figure, or digit, has been developed to formally designate the reliability of a numerical value.
- The *significant digits of a number are those that can be* used with confidence.
- If we say we know the value is 48 with two significant digits. The value is 48.5 with three significant digits and vice versa.
- Although quantities such as π, e, or $\sqrt{7}$ represent specific quantities, they cannot be expressed exactly by a limited number of digits.



SIGNIFICANT FIGURES

- Although it is usually a straightforward procedure to ascertain the significant figures of a number, some cases can lead to confusion.
- For example, zeros are not always significant figures because they may be necessary just to locate a decimal point.
- The numbers 0.00001845, 0.0001845, and 0.001845 all have four significant figures.
- Similarly, when trailing zeros are used in large numbers, it is not clear how many, if any, of the zeros are significant.
- For example, at face value the number 45,300 may have three, four, or five significant digits, depending on whether the zeros are known with confidence. Such uncertainty can be resolved by using scientific notation, where 4.53×10^4, 4.530×10^4, 4.5300×10^4 designate that the number is known to three, four, and five significant figures, respectively.

ERROR DEFINITIONS

- True value = approximation + error
- E_t = *true value – approximation*

  E_t *is used to designate the exact value.*

Let X_{comp} be a computed version of the exact quantity X_{exact}

$$\text{Absolute error} = |X_{comp} - X_{exact}|$$

$$\text{Relative error} = \frac{|X_{comp} - X_{exact}|}{|X_{exact}|}$$

- For example, we want to measure the length of bridge and a rivet.
- We measured the length of bridge is 9999 cm and the length of the rivet is 9 cm.
- The exact values are 10000 and 10 cm, respectively.
- The absolute error for both cases is 1 cm.
- The percentage relative errors are 0.01% and 10%.

Sources of Error in Numerical Computations

1. Errors in Mathematical Modeling
2. Errors in Numerical Input
3. Machine Error

   The floating point representation (in binary digits) of numbers involves rounding and chopping errors since each computing machine (computer, calculator etc.) can hold a finite number of digits. These errors are introduced at each arithmetic operation during the computations. It Depends on the bit of computer operating system and associated software packages-16 Bit/32 Bit/64 Bit.

Inherent Error = Errors in Mathematical Model + Errors in numerical input+ Machine Error

4. Blunders

Blunders are errors that are caused due to human imperfection. *Such errors may* cause a very serious disaster in the result. Some common sources of such errors are-

(i) Selecting a wrong numerical method for solving the mathematical model.

(ii) Selecting a wrong algorithm for implementing the numerical method.

(iii) Making mistakes in the computer programming (coding). Thus when a large program is written, it is a good practice to divide it into smaller sub-programs and test each sub-program separately for accuracy.

5. Computational Errors

Computational errors are introduced during the process of implementation of a numerical method. They come in two forms

Rounding a number can be done in two ways-

Chopping- Extra digits are dropped. This is called truncating the number. Example- a exact number 43.92638 can be approximated to 43.92 (2 digit chopping), 43.926 (3 digit chopping), 43.9263 (4 digit chopping)

Symmetric Roundoff- 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. Example- 42.7893 can be approximated to 42.79 (correct to 4 significant digits/2 decimal places) and 76.5432 can be approximated to 76.54.- round off errors.

6. Truncation Errors

Truncation errors are defined as those errors that result from using an approximation in place of an exact mathematical procedure. Truncation error results from terminating after a finite number of terms known as *formula truncation error or simply truncation error. The Taylor series*

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots + \frac{f^{(n)}(x_i)}{n!}h^n + R_n$$