APPROXIMATION IN NUMERICAL COMPUTATION

Numerical analysis is concerned with methods of computations which give solutions to mathematical problems. Numerical methods basically help us to solve mathematical problems where determining the solution is essential for further analysis, rather than the method of determining the solution. Such methods involve development of algorithms – a sequence of steps - which, using digital computers, can be programmed to attain fast solutions.

Types of Mathematical problems which can be solved:

- Definite Integrals
- Differentiation
- ODEs / Boundary Value Problems
- Solving system of linear equations
- Solving algebraic and transcendental equations

Sources of Errors:

The methods employed in numerical computations provide approximate solutions which give rise to errors. In general, the errors are of 2 types:

Truncation Errors

This arises due to the replacement of an infinite process by a finite one.
For example,

i. Computation of function using 1st few terms of Taylor's series
ii. In finding roots of equations by iterative process which converge when the no. of iterations tend to infinity
iii. Approximation of integrals by finite sum

Computational Errors

This arises during arithmetic computations due to finite representation of numbers.
For example, $\frac{1}{3} = 0.33333$

Significance Errors **Round-off Errors**

Significant Figures:

To compute results by finite precision arithmetic, the numbers are often rounded-off to a finite no. of digits. The general rule for ***rounding off a number to n significant figures*** is:

1. Discard all digits to the right of the nth. digit and if the discarded number is greater than half a unit in the nth place, add 1 to the nth digit, otherwise leave unchanged.
2. If the discarded number is exactly half a unit in the nth place, keep the digit unchanged if it is even otherwise, add 1.

Then the rounded-off number is said to be ***correct to n significant figures***.

Example: Round off the following correctly to 5 significant figures:

39.72431 79.87893 2.539951 22.73328

Zeroes on the left do not count as significant figure.

No. of decimal places is not the same as no. of significant figures.

Absolute Errors:

Absolute error (E_A) of an approximate result or of a measurement is the difference between the true value (x_T) of the quantity and the value obtained (x_A) by measurement, ie

$$\boldsymbol{E_A = |x_T - x_A|}$$

Since absolute error is related to no. of decimal places, it is not a very useful index of accuracy of approximate results

The absolute error of a result correct upto n sig. fig. cannot be greater than half a unit in the nth place.

Eg. $\frac{1}{3} = 0.3333\,, E_A = 0.00003333\ldots < \frac{1}{2} \times 0.0001 = 0.00005$

Relative Errors:

Relative error (E_r) is defined as the absolute error (E_A) divided by the true value (x_T), ie

$$\boldsymbol{E_r = \frac{E_A}{x_T} = \frac{|x_T - x_A|}{x_T}}$$

It is more meaningful if we express the error as **relative percentage error,**

$$\boldsymbol{E_p = \frac{E_A}{x_T} \times 100 = \frac{|x_T - x_A|}{x_T} \times 100}$$

The relative error of a result correct upto n sig. fig. is less than $(k \times 10^{n-1})^{-1}$ *where* k *is the first sig. fig.*

Eg. $N = 342.67392,\ n = 8,\ k = 3\,, E_A \leq \frac{1}{2} \times 10^{-5}\ \ ,\ E_r < (3 \times 10^7)^{-1}$

The following rules may be kept in mind while working with approximate numbers:

Rule 1: *The absolute error of an algebraic sum of several approximate numbers is equal to the sum of the absolute errors of the individual numbers*

E.g. Consider the sum of the approximate numbers 0.346, 0.1854, 345.2, 235.4, 11.53, 9.49, 0.0836, 0.0227, 0.00123, 0.000321 all their digits being correct upto the figures shown. Find the absolute error of the sum.

Maximum error is 0.05 from 345.2 and 235.4. Hence maximum error in the sum amounts to $2 \times 0.05 = 0.1$
To find the sum we round off the nos. to 2 decimal places and add which gives the sum as 602.26, rounding it up to 1 decimal place we get 602.3. So total rounding off error is 0.04 ± 0.1

Rule 2: *If the terms have one and the same sign, the relative error of their sum does not exceed the maximum relative error of any of the terms*

In the previous example, maximum relative error of the sum is $\frac{0.1}{602.26} \times 100 \simeq 0.017\%$

maximum relative error $\frac{0.05}{345.2} \times 100 = 0.014\%$, $\frac{0.05}{235.4} \times 100 = 0.021\%$

Rule 3: *The relative error of a difference of two approximate positive numbers is more than the relative error of these numbers*

Rule 4: *Avoid loss of significant digits that might occur while adding, subtracting or dividing by a small quantity, by using series representation for transcendental functions*

$$f(x) = \frac{e^x - 1}{x}$$

Generation and Propagation of Round off errors:

A round off error is introduced when a real number is converted to a machine number with finite size.

E.g. $\mathrm{a} = 223.4 = \mathbf{0.2234} \times \mathbf{10^{3}}$, $\mathrm{b} = 0.05124 = \mathbf{0.5124} \times \mathbf{10^{-1}}$, $\mathrm{c} = 11.12 = \mathbf{0.1112} \times \mathbf{10^{2}}$

Let x_1^*, x_2^* be 2 real numbers such that $x_1^* = x_1 + \epsilon_1\,,\;\; x_2^* = x_2 + \epsilon_2$

Then the **sum**,

$$x_1^* + x_2^* = (x_1 + \epsilon_1) + (x_2 + \epsilon_2)$$

or, $$\boldsymbol{(x_1^* + x_2^*) - (x_1 + x_2) = \epsilon_1 + \epsilon_2}$$

Hence, **the propagated round-off error in the sum of the approximate number is the sum of the round-off errors in the individual numbers**.

For the **product**,

$$x_1^*\, x_2^* = (x_1 + \epsilon_1)(x_2 + \epsilon_2)$$

or, $$\boldsymbol{(x_1^* x_2^*) - (x_1 x_2) \simeq \epsilon_1 x_1 + \epsilon_2 x_2}$$

The **relative error** in the product is the sum of the relative errors of the individual numbers

$$\boldsymbol{\frac{(x_1^* x_2^*) - (x_1 x_2)}{x_1 x_2} \simeq \frac{\epsilon_1}{x_1} + \frac{\epsilon_2}{x_2}}$$

For the **division**,

$$\frac{x_1^*}{x_2^*} - \frac{x_1}{x_2} = \frac{(x_1 + \epsilon_1)}{(x_2 + \epsilon_2)} - \frac{x_1}{x_2} = \frac{(\epsilon_1 x_2 - \epsilon_2 x_1)}{x_2 (x_2 + \epsilon_2)} \Rightarrow \boldsymbol{\frac{\left(\frac{x_1^*}{x_2^*} - \frac{x_1}{x_2}\right)}{\frac{x_1}{x_2}} \simeq \frac{\epsilon_1}{x_1} - \frac{\epsilon_2}{x_2}}\,, \quad \boldsymbol{when\ \frac{\epsilon_2}{x_2} \ll 1}$$

The propagated error in evaluating a **function $\boldsymbol{f(x)}$** having an error ϵ can be determined by expanding $f(x)$ by Taylor's theorem as,

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + \frac{\epsilon^2}{2!} f''(x) + \cdots$$

Assuming the higher order terms are negligible, $f(x+\epsilon) - f(x) = \epsilon f'(x)$

OR

Absolute error = change in function value = change in $\boldsymbol{x}$-value $\times$ first derivative

Relative error = change in function value/actual function value = $\left(\frac{\boldsymbol{\epsilon f'(x)}}{\boldsymbol{f(x)}}\right)$

General Rules for minimizing round-off errors:

1. Input the real data with as many significant digits as possible
2. Try to keep the value of all intermediate calculations close to ± 1
3. Write the expression in a form which requires fewest arithmetic operations

Examples:

1. The following numbers are approximated and correct upto its last figure. Find their sum.

351.9, 537.91, 36.252, 3.9134

2. Round off the number 4.5126 to 4 sig. fig. and find the relative percentage error

3. How many sig. fig does $x_A = -0.054113$ has with respect to $x_T = -0.05418$

4. The no. of sig. fig in 1.8921 if $E_r = 0.1 \times 10^{-2}$ _____________

5. The no. of sig. fig in 0.3941 if $E_A = 0.25 \times 10^{-2}$ _____________

6. Compute the relative error in computing $y = x^3 + 3x^2 - x$ for $x = \sqrt{2}$, taking $\sqrt{2} \simeq 1.414$

7. Given $u = x_1x_2 + x_1x_3 + x_2x_3$ find the error in computation of u, at $x_1 = 2.104$, $x_2 = 1.935$, $x_3 = 0.845$

Significance error

This is caused due to loss of significant digits during computation, mainly due to finiteness of the size of machine number. There are 2 cases when this occurs – when 2 nearly equal numbers are subtracted, or when division is done with a very small divisor compared to the dividend.

E.g. find the root of the equation $x^2 + 100.00001x + 0.01 = 0$

In Digital Signal Processor(DSP), data representation is of 3 types: **integer, fixed-point, floating point**

Fixed – Point Arithmetic

Fixed-point representation is a generalisation of integer representation. In a computer, the numbers are represented in binary(0 and 1) format.

Eg.
$$\mathbf{110101_2} = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$= 32 + 16 + 0 + 4 + 0 + 1 = \mathbf{53_{10}}$$

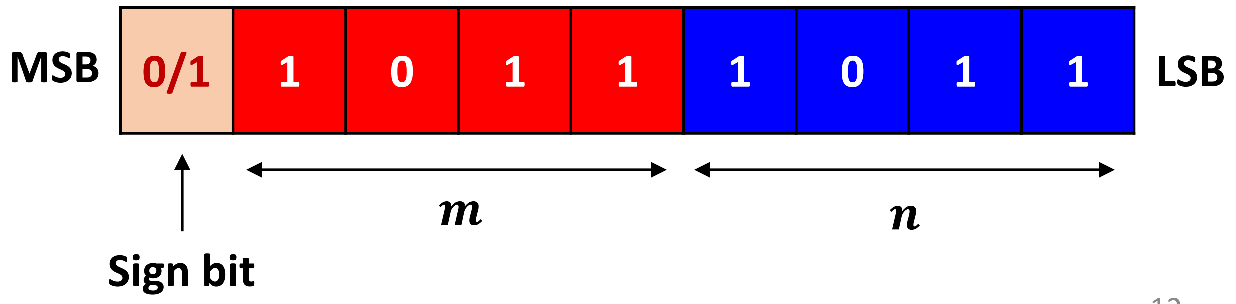
Eg.
$$\mathbf{1011.1011_2}$$
$$= \mathbf{1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0} + \mathbf{1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}}$$
$$= \mathbf{8 + 0 + 2 + 1} + \mathbf{0.5 + 0 + 0.125 + 0.0625}$$
$$= \mathbf{11.6875}$$

In a fixed-point representation, a number has a fixed number of digits before and after the decimal point

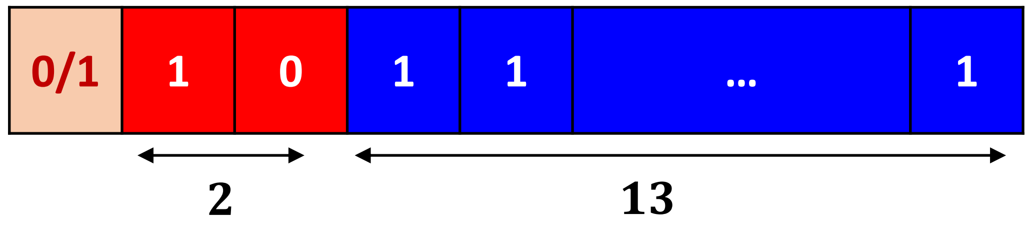
A real number, x is represented by an integer X with $N = m + n + 1$ bits where,

- N is the word length
- m is the no. of integer bits
- n is the no. of fractional bits

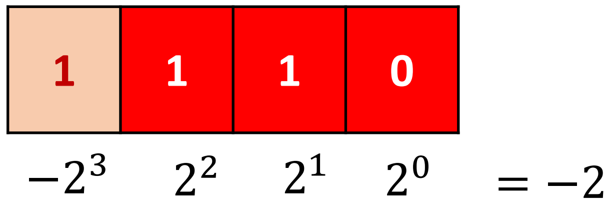
X is said to be in $\boldsymbol{Qm.n}$ notation



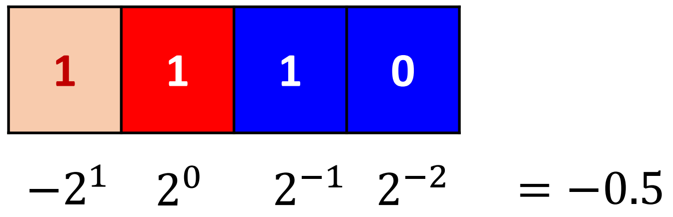
Eg. A 16-bit (N = 15+1) number in **Q2.13** format would have



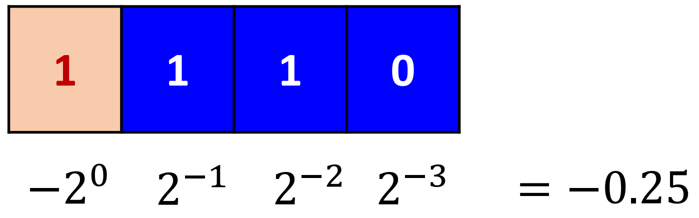
Eg. 1110 in **Q3.0** format :



Eg. 11.10 in **Q1.2** format :



Eg. 1.110 in **Q0.3** format :



Range of a signed fixed – point number is the difference between the largest positive number and the least negative number = $-2^m \leq x \leq 2^m - 2^{-n}$

Range of a unsigned fixed – point number = $0 \leq x \leq 2^m - 2^{-n}$

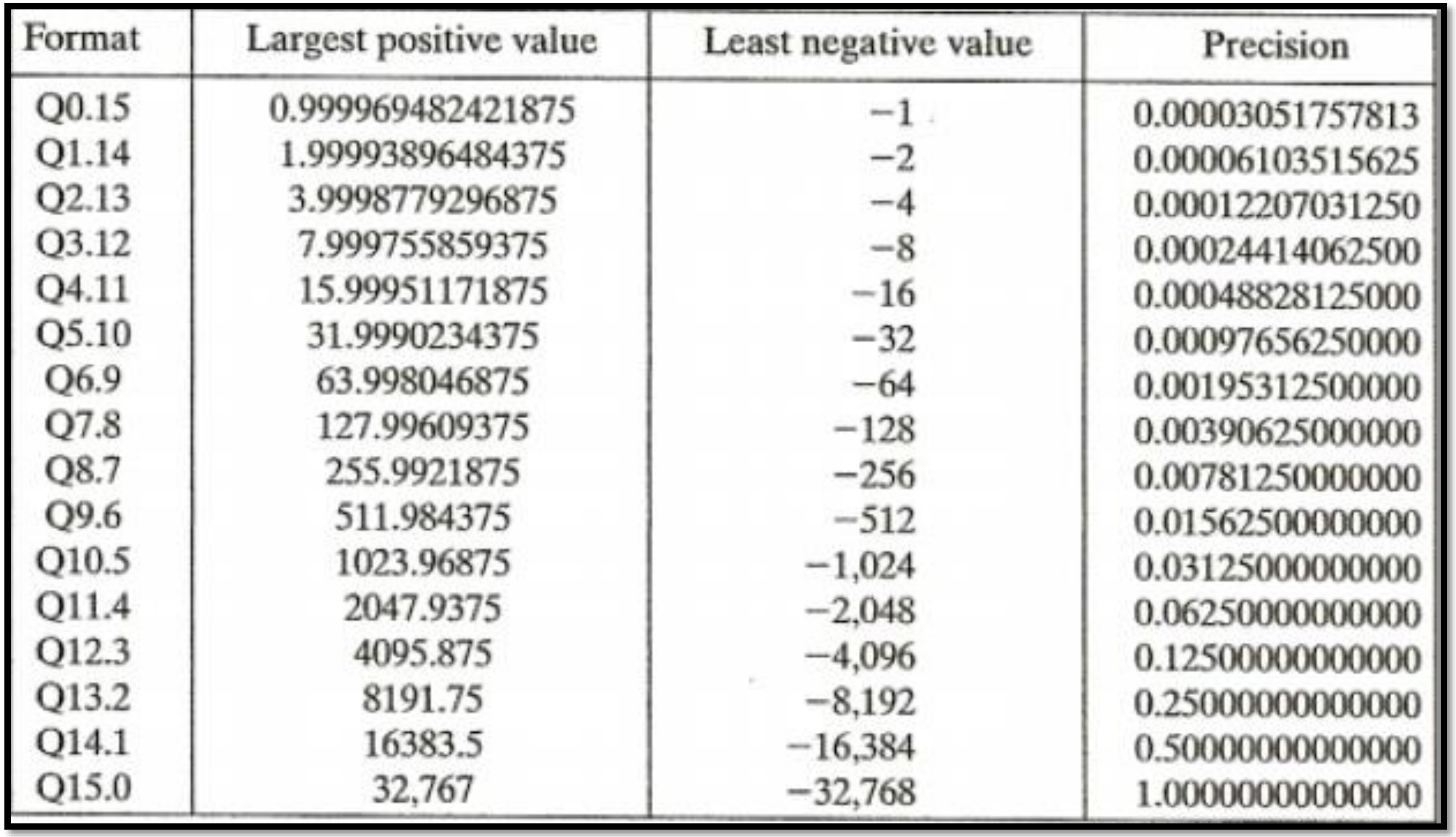
Overflow can happen if the result of an operation is larger or smaller than the numbers in the range

Resolution is the smallest non-zero number representable = 2^{-n}

Given a fixed range, if **n is too small then numbers have poor resolution** and if **n is too large there is a risk of overflow**

Precision is the difference between successive values representable, which is equal to the value of the least significant bit.

Dynamic Range and precision of 16-bit Numbers in different $Qm.n$ formats



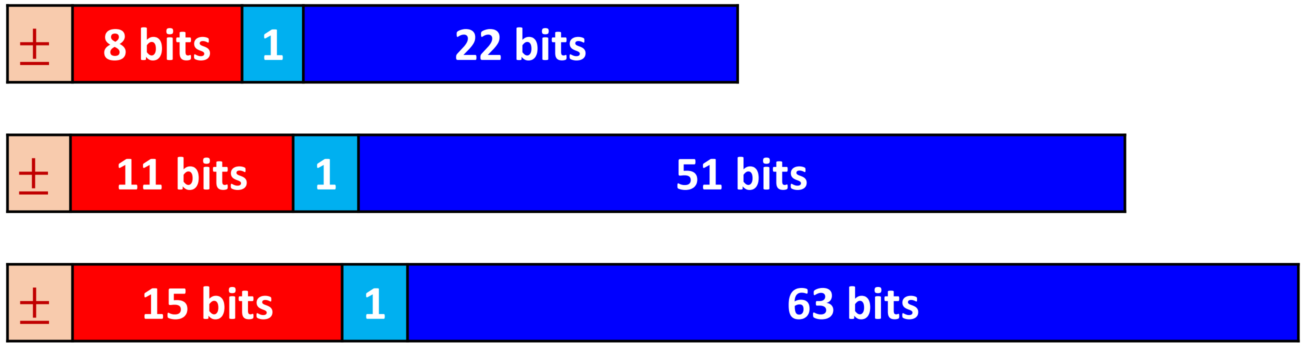
Format	Largest positive value	Least negative value	Precision
Q0.15	0.999969482421875	−1	0.00003051757813
Q1.14	1.99993896484375	−2	0.00006103515625
Q2.13	3.9998779296875	−4	0.00012207031250
Q3.12	7.999755859375	−8	0.00024414062500
Q4.11	15.99951171875	−16	0.00048828125000
Q5.10	31.9990234375	−32	0.00097656250000
Q6.9	63.998046875	−64	0.00195312500000
Q7.8	127.99609375	−128	0.00390625000000
Q8.7	255.9921875	−256	0.00781250000000
Q9.6	511.984375	−512	0.01562500000000
Q10.5	1023.96875	−1,024	0.03125000000000
Q11.4	2047.9375	−2,048	0.06250000000000
Q12.3	4095.875	−4,096	0.12500000000000
Q13.2	8191.75	−8,192	0.25000000000000
Q14.1	16383.5	−16,384	0.50000000000000
Q15.0	32,767	−32,768	1.00000000000000

Floating – Point Arithmetic

Floating-point numbers are used to represent fractions and allow more precision.

IEEE formats:

- 32 bits (**float** in C, C++, JAVA) (single)
- 64 bits (**double** in C, C++, JAVA) (double)
- 80 bits (**long double** in C, C++) (extended)



$$\boldsymbol{N} = (-\mathbf{1})^{\boldsymbol{S}} \times (\mathbf{1} + \boldsymbol{F}) \times \mathbf{2}^{\boldsymbol{E}}$$

Exponent field

Sign field
0 : positive
1 : negative

Mantissa
(Significand)

Base

Floating – Point Range and Precision

Type	Storage size	Value range	Precision
float	32 bits	1.2×10^{-38} to 3.4×10^{38}	6 decimal places
double	64 bits	2.3×10^{-308} to 1.7×10^{308}	15 decimal places
long double	80 bits	3.4×10^{-4932} to 1.1×10^{4932}	19 decimal places

- The **addition/subtraction** of 2 normalized floating point numbers is made by first making the exponents equal, by Shifting the digits of mantissa with lower value of exponent and then adding/subtracting the mantissa
- **Multiplication** of 2 normalized FP numbers is made by multiplying the mantissa and adding the exponents; the result is normalized and rounded off
- **Division** is done by dividing the mantissa and subtracting the exponent; the result is normalized and rounded off.