Computational Mathematics

Lecture 1

Floating Point:

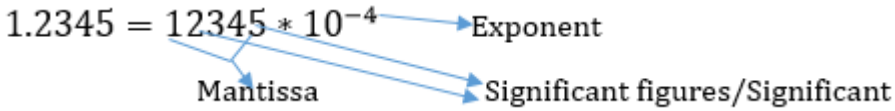
Floating point describes a method of representing an approximation of a real number in a way that can support a wide range of values. The numbers are, in general, represented approximately to a fixed numbers of significant digits (the mantissa) and scaled by exponent. The base for the scaling is normally 2, 10 or 16. The typical number that can be represented exactly is of the form:

$$Number = Significant\ digit * base^{exponent}.$$

$$\Rightarrow x = f * 10^{E}$$

The number f is called Mantissa, and E is the Exponent.

Example:



Ex-1: Convert the following numbers to floating point notation:

(i) 0.00596
(ii) 65.7452
(iii) – 486.8

$i.\ 0.00596\ is\ expressed\ as\ 0.596 * 10^{-2}$

$ii.\ 65.7452\ is\ expressed\ as\ 0.657452 * 10^{2}$

$iii.\ -486.8\ is\ expressed\ as\ -0.4868 * 10^{3}$

Consider the number 123, it can be written using exponential notation as:

1. $1.23 * 10^{2}$
2. $12.3 * 10^{1}$
3. $123 * 10^{0}$
4. $.123 * 10^{3}$
5. $1230 * 10^{-1}$ etc.

All of these representations of the number 123 are numerically equivalent; but differ only for their form of normalization, by using the point (.) in different places. That means point is floating for respective purpose.

Floating Point number:

A real number (That is, a number that can contain a fractional point). The following are floating-point numbers:

$$3.0, \quad -111.5, \quad \frac{1}{2}, \quad 3E-5, \quad 0.596 * 10^{-2}, \quad .596E-2, \quad -0.4868E3 \quad etc.$$

In the previous example, notice how the decimal point "floats" along the number as the exponent is changed. This phenomenon gives floating point numbers their name.

Normalization/Normalized floating point Numbers:

In the previous example the shifting of the decimal point to the left of the most significant digit is called normalization and the numbers represented in the normalized form are known as **Normalized Floating Point** numbers.

Mantissa should satisfy the following conditions:

For +ve numbers : <1.0 but ≥0.1

For –ve numbers : > -1.0 but ≤ -0.1

However, in scientific notation, The Mantissa/Significant is always: <10 but ≥1

$[e.g. \quad 1.23 * 10^2]$

And,

Standard computer normalization for floating point numbers (mantissa) is always:

<1 but ≥0.1 $\quad [e.g. \quad .123 * 10^3]$

Example:

The normalized floating point numbers are written using the following notation:

$0.596 * 10^{-2}$ *written as* $.596\text{E}-2$

$-0.4868 * 10^3$ *written as* $-.4868\text{E}3$

Floating Point Arithmetic:

1. Addition: Consider the highest of Exponent (E).

a. Add the numbers: 0.964572E2 and 0.586351E5

Here, E=5, So, 0.96457E2 ⇒ 0.000964E5 (modified denormalized)

And 0.586351⇒ 0.586351E5

+0.587315E5

Note:

1. Both the Mantissa and Exponent of the numbers with smaller exponent are modified.
2. Modified Mantissa is truncated to six-digits.

Add the numbers 0.735816E4 and 0.635742E4

Both Exponent (E) here = 4;

∴ 0.735816 + 0.635742 = 1.371558 ⇒ 1.371558E4 ⇒ 0.1371558E5 (Normalized)

Note: Mantissa of the result is truncated.

2. Subtraction:

Subtract 0.994576E-3 from 0.999658E-3

= 0.005082E-3

=0.508200E-5 (normalized)

3. Multiplication:
4. Division:

[Multiplication and Division Changes/modifies in both the Mantissa and/or exponent – and also truncates the values. That means, During Arithmetic operation. There happen some changes, according to necessity. – These are called errors.]

1. Truncation Error: When denormalize the smaller number before operation (Example (a)).
2. Overflow Error: When the result is larger than the maximum limit, it is reffered to as Overflow.
3. Underflow Error: When it is less than lower limit, it is reffered to as Underflow.

[Underflow/Overflow happens during normalization.]

Example:

1. **Truncation Error:** By adding **X** = 0.500000E1 and **Y** = 0.100000E-7

Here, Exp. Max=1

X ⇒ 0.500000E1 → 0.5000000E1

Y ⇒ 0.100000E-7 → 0.0000001E1 (denormalized)

Z ⇒ 0.5000001E1 → 0.500000E1

2. **Overflow Error:** Multiply the number : 0.350000E40 by 0.500000

   ⇒0.175000E1110

   Note: If we assume that the exponent can have a maximum value of 99, then this result **Overflows.**

3. **Underflow Error:** Divide the number 0.875000E-18 by 0.200000E95

   $here\ \mathrm{E}_z = -18 - 95 = -113$

   ⇒ 4.375000E-113

   ⇒ 0.437500E-114 (Normalized)

   Note: If we assume that the exponent can have a minimum value of -99, then this result **Underflows.**

Laws of Arithmetic:

Due to errors introduced in floating point arithmetic, the **Associative** and **Distributive** laws of arithmetic are not always satisfied. That is:

X+(Y+Z) ≠ (X+Y) + Z → Associative laws

X * (Y * Z) ≠ (X * Y) * Z → Associative laws

X * (Y + Z) ≠ (X * Y) + (X * Z) → Distributive laws

Proof:

a. **Associative law for Addition:**

$let\ x = 0.456732 * 10^2;$

$y = 0.243451;$

$z = -0.248000$

$x + y = 0.004567 + 0.243451 = 0.248018$

$(x + y) + z = 0.248018 - 0.248000 = 0.000018 = 0.180000 * 10^{-4}$

(Normalized)

Again $y + z = 0.243451 - 0.248000 = -0.004549 = -0.4549 * 10^{-2}$

And $x + (y + z) = (0.456732 - 0.454900) * 10^{-2} = 0.183200 * 10^{-2}$

Thus, $(x + y) + z \neq x + (y + z)$.

b. Associative law for Multiplication:

$let, x = 0.400000 * 10^{40}$;

$y = 0.500000 * 10^{70}$;

$z = 0.300000 * 10^{-30}$;

$(x * y) * z = (0.200000 * 10^{110})(0.300000 * 10^{-30}) = 0.600000 * 10^{80}$

But (x*y) may cause overflow and so result will be error.

Whereas: $x * (y * z) = (0.400000 * 10^{40}) * (0.1500000 * 10^{40}) = 0.060000 * 10^{80} = 0.600000 * 10^{79}$ $(normalized)$

Thus (x * (y * z)) ≠ (x * y) * z

c. Distributive law:

$let\ x = 0.400000 * 10^{1}$

$y = 0.200001 * 10^{0}$

$z = 0.200000 * 10^{0}$

$x * (y - z) = (0.400000 * 10^{1}) * (0.000001 * 10^{0})$
$= 0.400000 * 10^{1} * 0.100000 * 10^{-5} = 0.400000 * 10^{-5}$

$where\ as: (x * y) - (x * z) = 0.800000 * 10^{0} - 0.800000 * 10^{0} = 0$ $[not\ same]$

Exercise for Home Work:

P(60): Ex. 6-10;

6) Write the following numbers in normalized exponential form and E-form:

a) 12.34
b) -654.321
c) 0.001234
d) -0.009876
e) 0.0
f) 12345

7) Assuming that the mantissa are truncated to 4 decimal digits, Show how the computer performs the following floating point operations:

a) $0.5678 * 10^4 + 0.6666 * 10^4$
b) $0.1234 * 10^4 + 0.4455 * 10^{-2}$
c) $0.3366 * 10^{-2} - 0.2244 * 10^{-1}$
d) $(0.6789 * 10^2) * (0.2233 * 10^{-1})$
e) $(0.6789 * 10^2) + (0.2233 * 10^{-1})$

8) Assuming that the Mantissa are truncated to 4 decimal digits, compute the error in following computations:

(a) $5.6789 - 1.2345$

(b) $5.6789 + 9.2345$

9) Illustrate the concept of **Overflow** and **Underflow** with examples.

10) Discuss an example to show that the distributive law of arithmetic is not always satisfied in computational mathematics.