Understanding Errors in Computing

Storage space in a computer or any device is finite. These have the following effects :-

- a new number line called
  - floating point number line.
- a floating point arithmetic which features
  - largest and smallest number
  - Round off mechanism
  - machine precision.
- need to take care (during computations)
  - loss of significance in +
  - close enough versus =
  - cancellation occurring in −

- introduction of Truncation error when transferring symbolic analysis for computation.

- need to keep track of number of computations; i.e Order of computations

Computer storage :-

Bit - a short form for binary digit. Smallest unit of storage. These can be thought of as on/off switches to store information.

Byte - 8 bits make one byte. original intent was that 8 bits was enough to assign a unique storage value for language characters.

26 - Capital A-Z
26 lower case a-z
10 0-9
32 punctuation
...

. standardise
American
Standard
Code
for
Information
Interchange

Around 127 were standardised.

Word - "A machine is 32-bit or 64-bit"
# of bits a particular Computer's CPU can deal with in one go.

Storing and working with numbers.

Integers: are stored in R using 32-bit integer format.

$\underset{\uparrow}{\_}$ $\underbrace{\_ \; \_ \; \_ \; \_}$

one bit for sign

31 bits for number in basic binary

$\Rightarrow$ The largest positive integer $= 2^{31} - 1$
That can be stored (integer storage)

E.g. largest 3-bit integer

111 $\Leftrightarrow$ $2^2 + 2^1 + 2^0 = 2^3 - 1$

most significant bit

least significant bit.

Thus the number of bits limit the size of integer that can be represented.

- There are some advantages :-
  - Convenient format of entering binary #.
  - Hexadecimal code [Colours]

Floating point numbers (default R uses to store numbers)

I-nstitute for
E-lectrical
E-lectronic
E-ngineers

double precision storage.

Any number say 12.3456 is re written as

$$\underbrace{+}_{\text{sign}} \underbrace{0.123456}_{\text{mantissa}} \times 10^{\overset{\text{Exponent}}{2}}$$

Sign - 1 bit
mantissa - 52 bits
Exponent - 11 bits [1 bit for sign]

64

Again finite storage space will result in range - largest and smallest [Exponent]

Precision - limits on # of significant digits [mantissa]

Range :-

largest number ~ 1.797693×10^{308}
Smallest number ~ $2.225074 \times 10^{-308}$
[Positive]

0.5
0.25
0.0
0.0625
$\overline{0.8125}$

Precision: $0.8125 \leftrightarrow 1101$ $= 1\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + 1\cdot 2^{-4}$

mantissa binary

[Exact] – no loss in storage

$0.1 \leftrightarrow$ loose some precision.

– Cannot be represented by a finite number of binary digits.

– we will need to approximate such numbers by a round off mechanism

– most real numbers cannot be stored exactly.

Machine precision :- (ε)

– captures the spacing of the floating point line.

machine ε_m :: is the smallest $\varepsilon > 0$ such that $1 + \varepsilon > 1$

– this is not as small as the "smallest number"

$$\varepsilon_m = 2.220446 \times 10^{-16}$$

This is a real limit. Also this changes with x. ie smallest $\varepsilon > 0$

$$x + \varepsilon_x > x$$

directly related to and proportional to value of x. | Grows with x

Thus many numbers cannot be represented by machine. Such numbers are represented by approximations. Such induced errors are called round-off errors.

Accuracy :-

floating point comparison "close enough" instead of equality

Absolute difference : $|x - y|$

Relative difference : $\frac{|x-y|}{|x|}$

We must address this in any calculation as:

if $x == y$ [not to be done]

if $abs(x-y) < tol$ [✓]

Absolute Error

α - exact value

$\hat{\alpha}$ - Some computed value

$$E(\hat{\alpha}) = |\alpha - \hat{\alpha}|$$

Relative error

$$E_{rel}(\hat{\alpha}) = \frac{|\alpha - \hat{\alpha}|}{|\alpha|}$$

$$E_{rel}(\hat{\alpha}) = \frac{|\alpha - \hat{\alpha}|}{\alpha_{ref}}$$

Could be used to determine convergence

(Absolute)

$x = \cdots$

$x_{old} = \cdots$

while $\text{abs}(x - x_{old}) > tol$

    $x_{old} = x$

    update x

end

(Relative)

$x = \cdots$

$x_{old} = \cdots$

while $\text{abs}\left(\dfrac{x - x_{old}}{x_{old}}\right) > tol$

    $x_{old} = x$

    update x

end

Cancellation due to Round off ERRORS

Quadratic Equation

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$a = 1, \quad b = 54.32, \quad c = 0.1$

Solution (11- digits)

$$x_1 = 54.32\ 181\ 58995$$

$$x_2 = 0.001\ 841\ 6004\ 9576$$

Note: $b^2 = 2950.7 \ggg 4ac = 0.4.$

Floating point 4-digit [Decimal] Arithmetic :-

$$\sqrt{b^2 - 4ac} = \sqrt{2951 - 0.4000}$$

$$= \sqrt{2951} = 54.32$$

$$x_{1,4} = \frac{108.6}{2.000} = 54.30$$

$$x_{2,4} = \frac{54.32 - 54.32}{2.000} = 0.$$

Relative Error : 1 → 0.4 %

2 → 100 %

Alternate: sec due to cancellation of 2 numbers close in 4 digit

Rationalize!

$$x_{2,4} = \frac{2c}{-b + \sqrt{b^2 - 4ac}} = \frac{0.2}{108.6} = 0.0018$$
4c

Relative Error reduced $\equiv$ 0.05 percent.

- Alas alteration if used for $x_{1,4}$ would imply division by zero.

  – Catastrophic Error

Robust Solution :

Define : $q = -\frac{1}{2}\left[b + \text{Sign}(b)\sqrt{b^2 - 4ac}\right]$

$(b \neq 0)$

$$\text{Sign}(b) = \begin{cases} 1 & b > 0 \\ -1 & b < 0 \end{cases}$$

Then the two solutions are :-

$$x_1 = \frac{q}{a} \qquad x_2 = \frac{c}{q}$$

- Finite precision causes round off in individual Calculations
- Effect of round off accumulates

- Subtracting nearly equal numbers or very different in magnitude numbers lead to large loss of precision

- $\alpha >> \beta$ or $\beta >> \alpha$ then $\alpha + \beta$ or $\alpha - \beta$ will have Errors
- $\alpha \simeq \beta$ then $\alpha - \beta$ Cancellation

- Error caused in one operation can itself lead to a huge loss..