Floating Point Arithmetic

- On a computer, we can't store a number like

  $1/3 = 0.3333\ldots$

  exactly. The decimal expansion must be truncated.

  Note: in this context, we are not considering programs like maple, mathematica or other symbolic manipulators.

- Floating point arithmetic is the study of the effects of finite precision on arithmetic calculations

- This has important consequences in some of the programs we will write, but we will examine only of much of this as we need.

- When you add 2 numbers on a computer

  $a + b$

  you don't get $a+b$ exactly. Instead, you get

  $a + b +$ error

  where (hopefully) error is small.

- Let a = some real number (in the mathematical sense)

  $fl(a)$ = floating point representation of a (ie, the version of a that is stored on the computer).

  Hopefully $fl(a) = a +$ small error,

  but this is not always the case, especially if a is the result of several calculations.

- Storage error
  - This is the error made when storing a real number (a) on the computer
  - Suppose $a = \pi$ (which is irrational and has an infinite, non repeating decimal expansion).

$$a = 3.1415926535897.....$$

  - To store this on a computer, we need to truncate this to a finite number of decimal places.
  - There are 2 ways to do this
    - chop if a is chopped, we throw away all digits past some set number

chop a to 3 digits $\longrightarrow$ $fl(a) = 3.14$
chop a to 5 digits $\longrightarrow$ $fl(a) = 3.1415$

    - Round if a is rounded, we use the standard rounding rule

round a to 3 digits $\longrightarrow$ $fl(a) = 3.14$
round a to 5 digits $\longrightarrow$ $fl(a) = 3.1416$

Note
- The 5 digit representations are different between rounding and chopping
- There are many rounding rules that computers can use.

important →
- when we count digits, we start from the first non-zero digit

chop 0.0003472976 to 3 digits $\longrightarrow$ 0.000347
chop 41721163 to 4 digits $\longrightarrow$ 41720000

- Quantifying errors

There are two ways to quantify (or size) errors

- Absolute error

$$\text{abs error} = a_{\text{computed}} - a_{\text{exact}}$$

- Relative error

$$\text{rel. error} = \frac{a_{\text{computed}} - a_{\text{exact}}}{a_{\text{exact}}}$$

- most of the time we want relative error since this eliminates the magnitude effect of a

- Example 1 $a_{\text{computed}} = 1.347 \times 10^{10}$ $a_{\text{exact}} = 1.346 \times 10^{10}$

$$\text{abserror} = 0.001 \times 10^{10} = 10^{7}$$ (very large, looks bad)

$$\text{rel. error} = \frac{0.001 \times 10^{10}}{1.346 \times 10^{10}} = 7.42 \times 10^{-4}$$ (looks much better)

- Example 2 $a_{\text{computed}} = 2.347 \times 10^{-10}$ $a_{\text{exact}} = 1.462 \times 10^{-8}$

$$\text{abserror} = 2.347 \times 10^{-10} - 1.462 \times 10^{-8} = -1.4 \times 10^{-8}$$

(small, looks good)

$$\text{rel. error} = \frac{-1.4 \times 10^{-8}}{1.462 \times 10^{-8}} = -0.983$$ (nearly 100% error, very bad)

- Relative error tells you how many digits that a_{computed} and a_{exact} have in common

$|\text{rel. error}| \leq 5.0 \times 10^{-q}$ → a_{computed} and a_{exact} agree to q digits.

Example 3

$|\text{Rel. error}| = 3.65 \times 10^{-5}$ → the numbers agree to 5 digits

$|\text{Rel. error}| = 7.21 \times 10^{-5} = 0.721 \times 10^{-4}$ → the numbers agree to 4 digits

Floating Point Computation by Hand

when doing floating point computation by hand, the rules are the same as for standard calculations with the additional constraint that each intermmediate result needs to be rounded or chopped to the allowable number of digits

Example 4

- Suppose you are working on a computer that uses chopped storage with 3 digits and

$a = 1.34$

$b = 0.362$

$c = 7.99$

- compute the floating point version of $a+b+c$

$a+b = 1.34 + 0.362 = 1.702 \xrightarrow{\text{chop}} 1.70$

$(a+b)+c = 1.70 + 7.99 = 9.69 \xrightarrow{\text{chop}} \boxed{9.69}$

- compute $\dfrac{a \cdot b}{c}$

$a \cdot b = (1.34)\cdot(0.362) = 0.48508 \xrightarrow{\text{chop}} 0.485$

$\dfrac{a \cdot b}{c} = \dfrac{0.485}{7.99} = 0.060700876 \xrightarrow{\text{chop}} \boxed{0.0607}$

↑ remember that you start counting with the first non-zero digit.

- Though we are demonstrating the concepts using small number of digits, the ideas are present no matter how many digits are used.