Lecture 5b: Introduction to Floating Point Arithmetic

1 Floating Point Arithmetic – Chapter 5

1.1 Overview

- Digital representation of numbers
  - ▷ Size limits
  - ▷ Resolution limits
  - ▷ The floating point number line
- Floating point arithmetic
  - ▷ roundoff
  - ▷ machine precision
- Implications for routine computation
  - ▷ Use "close enough" instead of "equals"
  - ▷ loss of significance for addition
  - ▷ catastrophic cancellation for subtraction
- Truncation error
  - ▷ Demonstrate with Taylor series
  - ▷ Order Notation

1.2 What's going on here?

Spontaneous generation of an insignificant digit:

```
>> format long e  % display lots of digits
>> 2.6 + 0.2
ans =
        2.800000000000000e+00
>> ans + 0.2
ans =
        3.000000000000000e+00
>> ans + 0.2
ans =
        3.200000000000001e+00     Why does the least significant digit appear?
>> 2.6 + 0.6
ans =
        3.200000000000000e+00     Why does the small error not show up here?
```

1.3 Digital Storage of Floating Point Numbers

Numeric values with non-zero fractional parts are stored as **floating point numbers**.

All floating point values are represented with a normalized scientific notation[1].

Example:

$$12.2792 = \underbrace{0.123792}_{\text{Mantissa}} \times 10^{2} \quad \text{Exponent}$$

1.4 Digital Storage of Floating Point Numbers

Floating point values have a fixed number of bits allocated for storage of the mantissa and a fixed number of bits allocated for storage of the exponent.

Two common precisions are provided in numeric computing languages

Precision	Bits for mantissa	Bits for exponent
Single	23	8
Double	53	11

1.5 Digital Storage of Floating Point Numbers

A double precision (64 bit) floating point number can be schematically represented as

$$\overbrace{\underbrace{b}_{\text{sign}}\ \underbrace{bb\ldots\ldots bbb}_{\substack{\text{52 bit value}\\ \text{of mantissa}}}\ \underbrace{bbbbbbbbbbb}_{\substack{\text{11 bit exponent,}\\ \text{including sign}}}}^{\text{64 bits}}$$

The finite number of bits in the exponent limits the magnitude or *range* of the floating point numbers.

The finite number of bits in the mantissa limits the number of significant digits or the *precision* of the floating point numbers.

[1] The IEEE Standard on Floating Point arithmetic defines a normalized *binary* format. Here we use a simplified *decimal* (base ten) format that, while abusing the standard notation, expresses the essential ideas behind the decimal to binary conversion.

1.6 Digital Storage of Floating Point Numbers

The floating point mantissa is expressed in powers of $\frac{1}{2}$

$$\left(\frac{1}{2}\right)^0 = 1 \quad \text{is not used}$$

$$\left(\frac{1}{2}\right)^1 = 0.5 \qquad \left(\frac{1}{2}\right)^2 = 0.25 \qquad \left(\frac{1}{2}\right)^3 = 0.125 \qquad \cdots$$

1.7 Consequences of Finite Storage

Limiting the number of bits allocated for storage of the exponent	$\Longrightarrow$	Upper and lower limits on the **range** (or magnitude) of floating point numbers
Limiting the number of bits allocated for storage of the mantissa	$\Longrightarrow$	Limit on the**precision** (or number of significant digits) for any floating point number.

1.8 Floating Point Number Line

Compare floating point numbers to real numbers.

	Real numbers	Floating point numbers
Range	**Infinite**: arbitrarily large and arbitrarily small real numbers exist.	**Finite**: the number of bits allocated to the exponent limit the magnitude of floating point values.
Precision	**Infinite**: There is an infinite set of real numbers between any two real numbers.	**Finite**: there is a finite number (perhaps zero) of floating point values between any two floating point values.

Note: In other words The floating point number line is a subset of the real number line.

1.9 Floating Point Number Line