

Lecture # 2

Floating Point Arithmetic

Announce MATLAB session.

Finish work on quadratic equation from last lecture.

Floating Point Number System

x is a floating point number if it has the form

$$x = \pm d \cdot \beta^e \quad \beta \in \{2, 10\}$$

Base 2 is for general purpose computers, base 10 is for pocket calculators.

e is the exponent and satisfies

$$e_{\min} \leq e \leq e_{\max} \quad , e_{\min} < 0 < e_{\max}$$

We will assume that arithmetic is in base 2, but will usually give examples in base 10.

Mantissa d has the form

$$d = 0.d_1 \dots d_t = d_1\beta^{-1} + d_2\beta^{-2} + \dots + d_t\beta^{-t}$$

$$d \in \{0, 1\}$$

$$d_1 = 1 \quad \text{normalized}$$

$$d_1 = 0 \quad \text{unnormalized}$$

Standard form for floating point numbers is normalized except at the bottom of the exponent range.

During input and output numbers are converted from binary to decimal and back.

Computer arithmetic is standardized, by the IEEE 754 standard for binary arithmetic. All but a few modern computers follow this standard.

Machine unit

$$\epsilon_M = \max_{[\log_2 |x|] \in [e_{\min}, e_{\max}]} \frac{|x - fl(x)|}{|x|} = 2^{-t}$$

All real numbers in the normalized range.

$fl(x)$ is the floating point round of x .

IEEE Single Precision $\beta = 2, t = 24$

$$\epsilon_M = 2^{-24} \approx 5.9605 \times 10^{-8}$$

$$e_{\min} = -126, e_{\max} = 128$$

IEEE Double Precision $\beta = 2, t = 53$

$$\epsilon_M = 2^{-53} = 1.1102 \times 10^{-16}$$

MATLAB “eps” command gives $\epsilon_m = 2.2204 \times 10^{-16}$ which is the maximum relative spacing between two floating point numbers.

Floating Point Operations

$$op \in \{+, -, *, /\}$$

$$fl(x \ op \ y) = (x \ op \ y) (1 + \xi), \quad |\xi| \leq \epsilon_M$$

For division, we assume $y \neq 0$.

Any IEEE standard computer will follow this rule.

Number Ranges

Largest Computer Number

$$\Omega = (1 - 2^{-t}) \cdot 2^{e_{\max}}$$

IEEE Single

$$e_{\max} = 128, \quad \Omega \approx 10^{38}$$

IEEE Double

$$e_{\max} = 1024, \quad \Omega = 1.79777 \times 10^{308}$$

The MATLAB “REALMAX” command displays this number.

When numbers exceed Ω , they are stored as “Inf” or “-Inf.”

Smallest Computer Number

You may find this a bit confusing? No pun intended.

IEEE Standard

$$\omega = 2^{-t}2^{e_{\min}}$$

IEEE Single

$$2^{-24-126} = 2^{-150} \approx 7.0056 \times 10^{-46}.$$

IEEE Double

$$2^{-1022-53} = 2^{-1075} \approx 10^{-323.6072}$$

Confusing point — Numbers at bottom of exponent range are not normalized.

MATLAB function REALMIN yields

$$\underline{\omega \approx 2.2251 \times 10^{-308}}$$

Some people call this the smallest USEFUL floating point number, Since

$$1/\omega \leq \Omega$$

and it is normalized.

Smallest floating point number is of the form

$$0.0 \dots 01 \times 2^{e_{\min}} \quad \dots \quad \underline{\text{Gradual Underflow}}$$

Before the IEEE standard most computers had the smallest floating point number as

$$0.10 \dots 0 \times 2^{e_{\min}} \quad \dots \quad \text{normalized}$$

Earlier computers, (pre-1985) set numbers below this smallest “useful” floating point number to zero. This change was one of the more controversial features of the IEEE standard.

Here is an example (in decimal arithmetic) which shows why this feature made it into the IEEE standard.

Example $\beta = 10, -5 \leq e \leq 5$

$$x = 0.1957 \times 10^{-5}$$

$$y = 0.1942 \times 10^{-5}$$

Do $fl(x - y)$ — What happens?

$$0.1957 \times 10^{-5} - 0.1942 \times 10^{-5} = 0.0015 \times 10^{-5}$$

Old Philosophy — sets $x - y$ to zero

Gradual Underflow stores $x - y$ as 0.0015×10^{-5}

In gradual underflow

$$fl(x - y) = 0 \text{ if and only if } x = y.$$

Interesting floating point computations

Example 1

$$f(x) = \sqrt{1 + x^2} - 1 \quad x \text{ near zero.}$$

$f(10^{-12}) = 0$ using the formula in IEEE double.

Better

$$\begin{aligned} f(x) &= (\sqrt{1 + x^2} - 1) \left(\frac{\sqrt{1 + x^2} + 1}{\sqrt{1 + x^2} + 1} \right) \\ &= \frac{x^2}{\sqrt{1 + x^2} + 1} \end{aligned}$$

$$\frac{f(10^{-12})}{\text{Very good answer.}} = 0.5 \cdot 10^{-24}$$

Very good answer.

This trick is used quite often.

Example 2 (Quadratic Equation– Revisited)

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

Roots

$$x_1 = \frac{-b - \text{sign}(b)\sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b + \text{sign}(b)\sqrt{b^2 - 4ac}}{2a}$$

when $b^2 - 4ac \geq 0$.

$$\begin{aligned} x_2 &\cdot \left(\frac{b + \text{sign}(b)\sqrt{b^2 - 4ac}}{b + \text{sign}(b)\sqrt{b^2 - 4ac}} \right) \\ &= \frac{-b^2 + b^2 - 4ac}{2c \cdot (b + \text{sign}(b)\sqrt{b^2 - 4ac})} \\ &= \frac{-4ac}{2a \cdot (b + \text{sign}(b)\sqrt{b^2 - 4ac})} = \frac{-2c}{b + \text{sign}(b)\sqrt{b^2 - 4ac}} \end{aligned}$$