ECE 4960
Spring 2017

Lecture 2

Numerical Representation and Precision

Edwin C. Kan
School of Electrical and Computer Engineering
Cornell University

Suggested Platform

- Language: C++
- OS: Linux (Debian or Ubuntu); generic platform or Oracle virtual machine
- Integrated development platform (IDE): code::block or generic Make with vim or emacs
- Tradeoffs between generic vs. customized platforms
  - Long-term evolution
  - Cost to the company
  - Porting among various platforms

Discussion

- What is your most comfortable platform?
  - Language
  - OS
  - Development environment

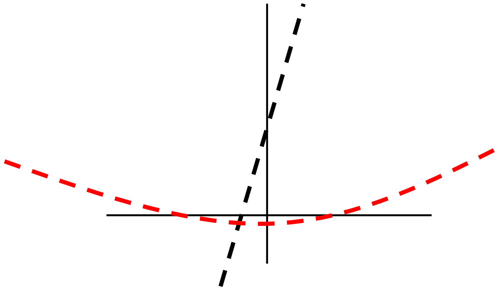
Integers and Floating Points in IEEE 754

Data type (C++ declaration)	No. of bits	Attributes	Exception handling	Smallest possible[1]	Largest possible
Integer (`short`)	16	Seldom used now	None	-2^{15}	$2^{15}-1$
Integer (`long`)	32	1 sign bit;	None	-2^{31}	$2^{31}-1$
Single precision floating point (`real`)	32	Seldom used now; 1 sign bit (s); 23 mantissa bits (f); 8 exponent bits (e). Normalized: $x=(-1)^s \cdot (1.f) \cdot 2^{e-127}$ 127 is the "bias".	***NaN***: $e=255; f \neq 0$ ***INF***: $e=255; f=0; s=0$ ***NINF***: $e=255; f=0; s=1$	Only by e: 2^{-126} Soft landing: $2^{-23} \cdot 2^{-126} = 2^{-149} \cong 1.4\times10^{-45}$	$(2-2^{-23}) \cdot 2^{127} \cong 3.4\times10^{38}$
Double precision floating point (`double`)	64	1 sign bit (s); 52 mantissa bits (f); 11 exponent bits (e). Normalized: $x=(-1)^s \cdot (1.f) \cdot 2^{e-1023}$ 1023 is the "bias".	***NaN***: $e=2047; f \neq 0$ ***INF***: $e=2047; f=0; s=0$ ***NINF***: $e=2047; f=0; s=1$	Only by e: 2^{-1022} Soft landing $2^{-52} \cdot 2^{-1022} = 2^{-1074} \cong 4.9\times10^{-324}$	$(2-2^{-52}) \cdot 2^{1023} \cong 1.8\times10^{308}$

[1]Some bit combinations are used for exception handling. Also, very small number has underflow controls.

Why Precision Matters?

$$y = f(x) = ax^2 + bx + c = 0$$



$$a = 10^{-5};\ b = 10^{3};\ c = 10^{3}$$

With 9 digits of precision

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$x_1 = -1.0$ and $x_2 = -3.0518\times10^7$

$$x_{1,2} = \frac{-2c}{-b \pm \sqrt{b^2 - 4ac}}$$

$x_1 = -1.0$ and $x_2 = -3.2768\times10^7$

Group Discussion

- What does your calculator say by using these two equations?
- How can you make double precision calculation to show such problems?

Function Conditioning?

We can define a function condition number κ by the sensitivity to perturbation:

$$\left|\frac{\Delta f}{f}\right| \cong \kappa \left|\frac{\Delta x}{x}\right| \qquad \text{or} \qquad \kappa = \left|\frac{\Delta f}{\Delta x} \cdot \frac{x}{f}\right|$$

- Only one of the possible error sources!

$$\left|\frac{f(-1.01) - f(-1)}{0.01 \times \left(\frac{f(-1.01) + f(-1)}{2}\right)}\right| = 2.01$$

$$\left|\frac{f\left(-3.0518 \times 10^{7}\right) - f\left(-3.0823 \times 10^{7}\right)}{0.01 \times \left(\frac{f\left(-3.0518 \times 10^{7}\right) + f\left(-3.0518 \times 10^{7}\right)}{2}\right)}\right| = 0.55$$

Precision Improvement by Perturbation

The precision error comes from $-b \pm \sqrt{b^2 - 4ac}$

If we have **existing knowledge** (or by numerical tests) for this precision issue, we can use the perturbation solution for $4ac << b^2$,

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm b\sqrt{1 - \frac{4ac}{b^2}}}{2a} \cong \frac{-b \pm b\left(1 - \frac{2ac}{b^2}\right)}{2a}$$

$$= -\frac{c}{b} \quad or \quad -\frac{b}{a} + \frac{c}{b}$$

We now obtain the two roots of -1 and $\mathbf{1 - 10^8}$!
Check back substitution, $f(1 - 10^8) = 10^{-5}$ and $f(-10^8) = 10^3$

Much better than $f(-3.0518\times10^7) = -2.12\times10^{10}$ or $f(-3.2768\times10^7) = -2.20\times10^{10}$ in terms of residual of $f(x)$!

Ground Truth, Asymptote and Perturbation

- Do we know the "symbolic" truth? (related to **formal verification**, then we have 100% proof that some answers are correct)
- Can we **back substitute** the answer for validation?
- Do we know the answer at special points or **asymptote**? (such as 0, INF and NINF)
- Can we or do we need to check the sensitivity by **perturbation**?

Hacker Practice

- Solve the quadratic equation for $a = 10^{-20}$; $b = 10^{3}$; $c = 10^{3}$ in your platform.

- Where is the potential problem in precision?

- What are the possible ways to detect and compensate the nearly degenerate condition?