September 5, 2024

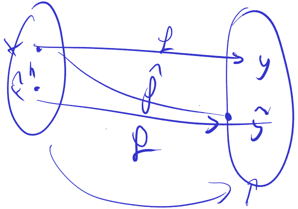
Announcements

HW1 due
HW2 out later
Exam 1
cutoff: end of Lec 5

Goals

- demo on bw stab
- FP

Review



$$\leq \kappa \cdot \frac{|x - \hat{x}|}{|x|}$$

$A \hat{x} = \tilde{b}$

out. in

backward

$A \tilde{x} = \hat{b}$

LHS = RHS

LHS − RHS = residual

Relevance of Backward Error

What do we gain from a bound on backward error like

$$\frac{\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|} \leq \epsilon?$$

Demo: Backward Stability by Example [cleared]

Getting into Trouble with Accuracy and Stability

How can I produce inaccurate results?

- Apply an inaccurate method
- Apply an unstable method to a well-conditioned problem
- Apply any type of method to an ill-conditioned problem

Wanted: Real Numbers. . . in a computer

Computers can represent *integers*, using bits:

$$23 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = (10111)_2$$

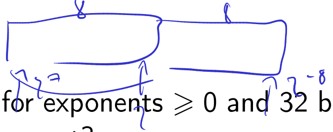
How would we represent fractions?

$23.625 = (10111.10 \quad)$

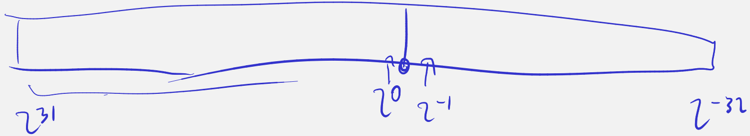
$2^{-1} \quad 2^{-2} \quad 2^{-3}$

$.625 - .5 = .125$

Fixed-Point Numbers



Suppose we use units of 64 bits, with 32 bits for exponents $\geqslant 0$ and 32 bits for exponents < 0. What numbers can we represent?



How many 'digits' of relative accuracy (think relative rounding error) are available for the smallest vs. the largest number?

Floating Point Numbers

Convert $13 = (1101)_2$ into floating point representation.

$$13 = \pm(1.101)_2 \cdot 2^{3}$$

What pieces do you need to store an FP number?

sign

significand

$1.\underbrace{101}_{\text{fraction}}$

exponent

store

"normalization"

Floating Point: Implementation, Normalization

Previously: Consider *mathematical* view of FP. (via example: $(1101)_2$)

Next: Consider *implementation* of FP in hardware.

Do you notice a source of inefficiency in our number representation?

- leading of sig always 1 : throw away
- to store exponent:

$$3 = \underbrace{-1023}_{\text{"implicit offset"}} + \underbrace{1026}_{\substack{\text{stored} \\ \geq 0}}$$

Implementing Arithmetic

How is floating point addition implemented?

Consider adding $a = (1.101)_2 \cdot 2^1$ and $b = (1.001)_2 \cdot 2^{-1}$ in a system with three stored bits (four total) in the significand.

$$a = (1.101)_2 \cdot 2^1$$

$$+\ b = (0.01001)_2 \cdot 2^1$$

$$(1.11101)_2 \cdot 2^1$$

1.

1.

10

Implementing Arithmetic

How is floating point addition implemented?
Consider adding $a = (1.101)_2 \cdot 2^1$ and $b = (1.001)_2 \cdot 2^{-1}$ in a system with three stored bits (four total) in the significand.

Rough algorithm:

1. Bring both numbers onto a common exponent
2. Do grade-school addition from the front, until you run out of digits in your system.
3. Round result.

$$\begin{aligned} a &= 1.\quad 101 \cdot 2^1 \\ b &= 0.\quad 01001 \cdot 2^1 \\ a + b &\approx 1.\quad 111 \cdot 2^1 \end{aligned}$$

Unrepresentable numbers?

Can you think of a somewhat central number that we cannot represent as

$$x = (1.\_\_\_\_\_\_\_\_\_\_\_)_2 \cdot 2^{-p}?$$

(done via demo)

Demo: Picking apart a floating point number [cleared]

Subnormal Numbers

What is the smallest representable number in an FP system with 4 stored bits (5 total) in the significand and a stored exponent range of $[-7, 8]$?

(done via demo)

Subnormal Numbers

What is the smallest representable number in an FP system with 4 stored bits (5 total) in the significand and a stored exponent range of $[-7, 8]$?

First attempt:

- Significand as small as possible $\rightarrow$ all zeros after the implicit leading one
- Exponent as small as possible: -7

So:

$$(1.0000)_2 \cdot 2^{-7}.$$

Unfortunately: wrong.

Subnormal Numbers, Attempt 2

What is the smallest representable number in an FP system with 4 stored bits in the significand and a (stored) exponent range of $[-7, 8]$?

(done via demo)

Why learn about subnormals?

Subnormal Numbers, Attempt 2

What is the smallest representable number in an FP system with 4 stored bits in the significand and a (stored) exponent range of $[-7, 8]$?

- Can go way smaller using the *special exponent* (turns off the leading one)
- Assume that the special exponent is -7; interpreted as -6.
- So: $(0.0001)_2 \cdot 2^{-6}$ (with four digits after the point stored).

Numbers with the special exponent are called *subnormal* (or *denormal*) FP numbers. Technically, zero is also a subnormal.

Why learn about subnormals?

- Subnormal FP is often slow: not implemented in hardware.
- Many compilers support options to 'flush subnormals to zero'.