Floating Point – Error and Stability

CS370 - May 6, 2016

Who are you?

1. What area(s) of CS are you most interested in?
2. What aspect/topic of the course are you most interested in?
3. What are your career plans/hopes for after graduation?

Floating Point – Quick Recap

We approximated the real numbers with finite storage by limiting precision (# of digits, t) and range (exponent, p):

$$0.d_1 d_2 \ldots d_t \times \beta^p$$

An FP system F guarantees that:

- $fl(x) = x(1 + \delta)$ for $x \in \mathbb{R}$ with $|\delta| \leq E$.
- $a \oplus b = fl(a + b) = (a + b)(1 + \delta)$ for $a, b \in F$ and $|\delta| \leq E$.

These rules let us bound error in our calculations.

Error Bounds and Condition Number

Last time, we said that $(a \oplus b) \oplus c$ satisfies:

$$E_{rel} \leq \frac{|a| + |b| + |c|}{|a + b + c|}(2E + E^2)$$

The term $\frac{|a|+|b|+|c|}{|a+b+c|}$ …

- describes how the error *E* is magnified along the way.
- may be called a *condition number* for this calculation.

Cancellation Errors

When is $\frac{|a|+|b|+|c|}{|a+b+c|}$ large? When is it small?

Worst case magnification when denominator is small.

i.e., when $|a+b+c| \ll |a|+|b|+|c|$.

This occurs when quantities have differing signs and similar magnitudes, leading to cancellation.

Error Bound Example #1

$a = 2000$
$b = -3.234$
$c = -2000$
$F \ = \{10, 4, -10, 10\}$

Consider $(a \oplus b) \oplus c$:

True result: -3.234 FP result: -3.000

E^2 term is very small, so we drop it.

Error bound:

$$E_{rel} \leq \frac{|a|+|b|+|c|}{|a+b+c|}(2\mathrm{E} + \mathrm{E}^2) \approx \frac{4003.234}{3.234} \cdot 2\left(\frac{1}{2}10^{-3}\right) \approx 1.238.$$

A rather weak bound; large potential relative error.

Error Bound Example #2

$a = 2000$
$b = -3.234$
$c = -2000$
$F \ = \{10, 4, -10, 10\}$

Consider $(a \oplus c) \oplus b$:

True result: -3.234 FP result: -3.234

Observation #1: Re-ordering operations often changes the results.

Observation #2: Our error bound is the *same as before.*

The condition number only gives a *worst-case* bound.

Error Bound Example #3

$F = \{10, 4, -10, 10\}$

Consider $(a \oplus b) \oplus c$ for $a = -2000, b = -3.234, c = -2000$:

(i.e. all matching signs now).

True: -4003.234 Computed: -4003

Condition number: $\frac{|a|+|b|+|c|}{|a+b+c|} = \frac{4003.234}{4003.234} = 1.$

Error bound: $E_{rel} \leq 1 \cdot (2E + E^2) \approx 2E = 10^{-3}$.

Observation: Avoiding cancellation gives better bounds on error.

Catastrophic Cancellation Error

Catastrophic cancellation occurs when subtracting numbers of the same magnitude, *and* the input numbers contain error.

e.g.,

$$\underbrace{173.00}_{\text{Correct digits}}\underbrace{063}_{\text{Erroneous digits}} - \underbrace{173.00}_{\text{Correct digits}}\underbrace{017}_{\text{Erroneous digits}} = 0.00\underbrace{046}_{\text{Erroneous digits}}$$

All *significant* digits cancelled out; the result might have no correct digits whatsoever!

Benign Cancellation

However, if input quantities are known to be *exact*, we have our usual guarantee on floating point ops (addition, subtraction, etc.):

$$w \oplus z = fl(w + z) = (w + z)(1 + \delta).$$

Round-off Errors We've Seen

Adding large and small numbers (very different magnitudes).

- Smaller digits get lost or "swamped"!
- Rule of thumb: Try to sum numbers of approximately same size.

Subtracting nearby numbers *that contain error.*

- Loss of accuracy due to catastrophic cancellation.
- Rule of thumb: Try to reformulate computations to avoid cancellation.

Taylor series example, revisited

So what's the reason we observed that

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

performs so much worse than

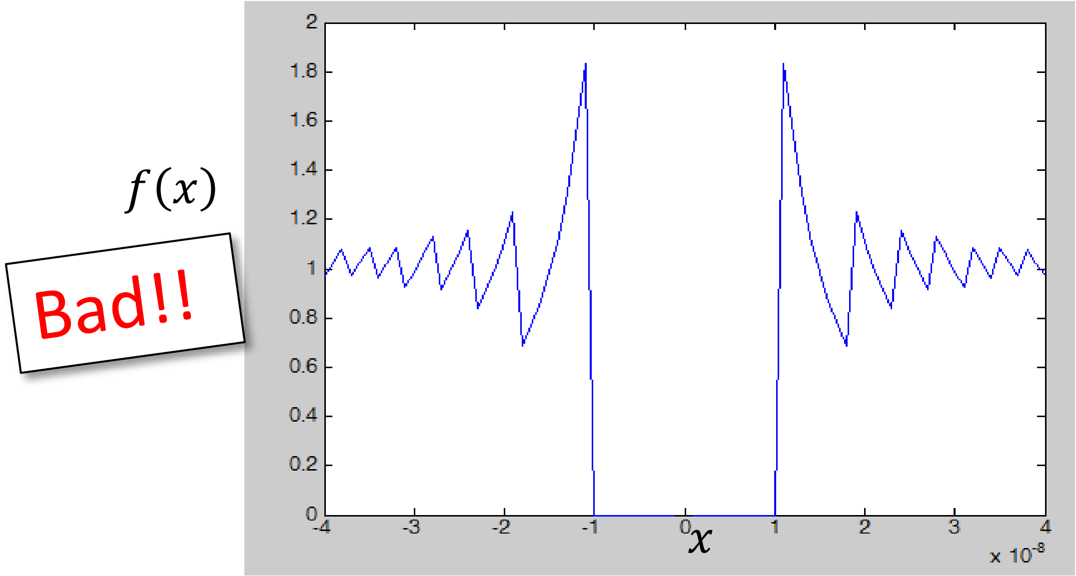
$$e^{-x} = \frac{1}{e^x} = \frac{1}{1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots}$$

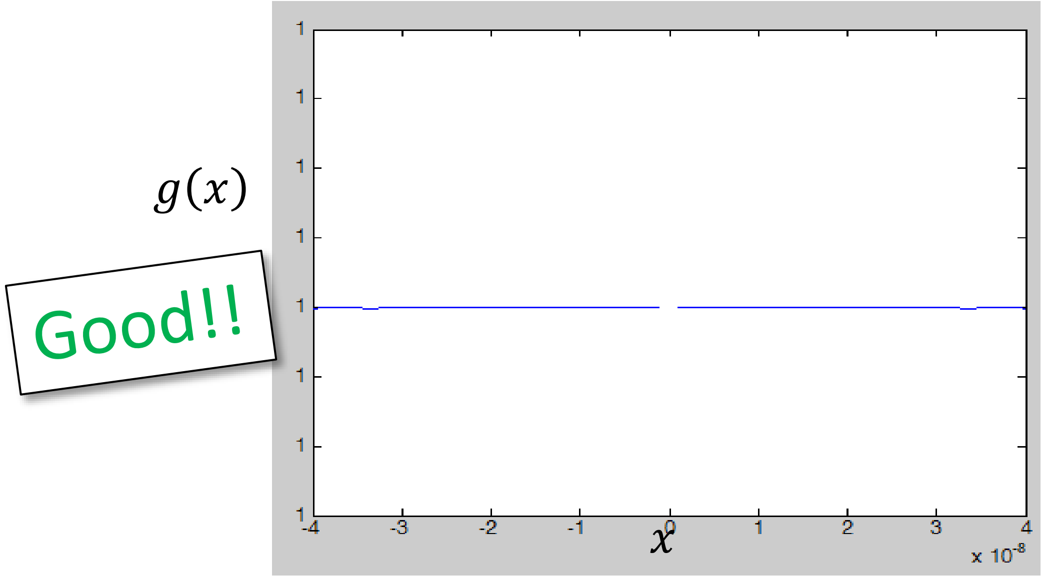
as an algorithm to evaluate $e^{-5.5}$ in FP?

Latter *avoids alternating signs* in the sums which lead to cancellation.

A Visualized Example of Cancellation Error

Compare: $f(x) = \frac{1-\cos^2 x}{x^2}$ and $g(x) = \frac{\sin^2 x}{x^2}$ near $x = 0$. True solution approaches 1 as $x \rightarrow 0$.





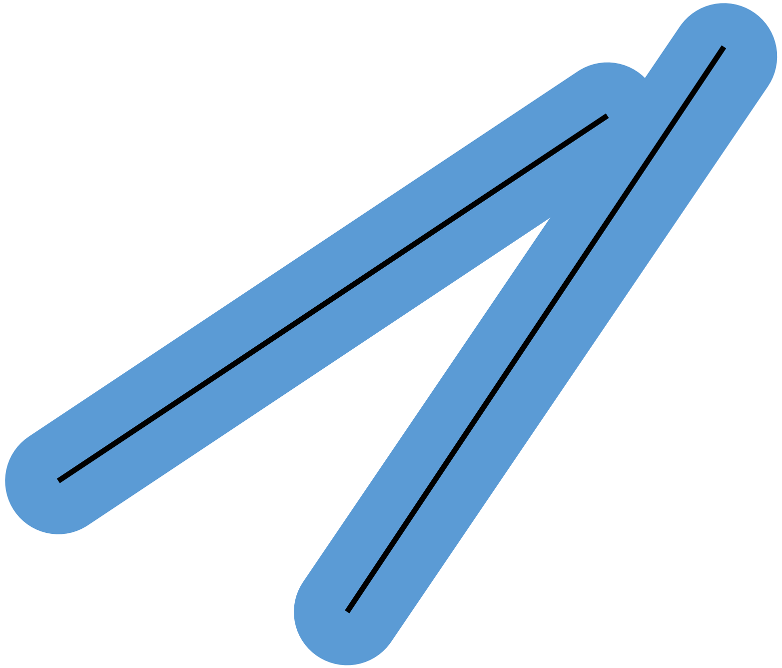
MATLAB plot with IEEE double precision (i.e., $\approx 15 - 17$ decimal digits).

FP Error: A Geometric Application Example

Do two line segments intersect?

A useful ***robust*** test for this must consider round-off error!

Both false positives or false negatives can occur.



e.g. “Epsilon geometry” [Guibas et al. 1989]

Sources of Error

When solving a problem numerically, there are many possible error sources:

- Round-off error – due to FP representation and arithmetic.
- Truncation error – eg. truncating a Taylor series after n terms.
- Uncertainty/error in the input itself (e.g. from measurements).
- Error/approximation in our mathematical model of the “real” problem.

We will (continue to) focus largely on the first two.

Conditioning of Problems

Problems may be ***ill-conditioned*** or ***well-conditioned****.*

For problem P, with input I and output O, if a change to the input, ΔI, gives a "small" change in the output ΔO, P is *well-conditioned*.

Otherwise, P is *ill-conditioned*.

This is a property of the problem, *independent* of any specific implementation (algorithm or hardware).

Conditioning is relative; there's a "sliding scale".

Stability of an *Algorithm*

If any initial error in the data is magnified by an *algorithm*, the algorithm is considered numerically *unstable*.

Can lead to meaningless results, for *seemingly* reasonable methods on reasonable problems.

Conditioning v.s. Stability

Conditioning of a problem:

- How sensitive is the *problem* itself to errors/changes in input?

Stability of a numerical algorithm:

- How sensitive is the *algorithm* to errors/changes in input?

Note that:

1. An *algorithm* can be unstable even for a well-conditioned problem!
2. An ill-conditioned problem limits how well we can expect an algorithm to perform.

Stability Analysis of an *Algorithm*

Consider the integration problem

$$I_n = \int_0^1 \frac{x^n}{x+\alpha} dx$$

for a given n where α is some fixed parameter.

The course notes gives a recursive algorithm, for $n \geq 0$:

$$I_0 = \log \frac{1+\alpha}{\alpha}, \qquad I_n = \frac{1}{n} - \alpha I_{n-1}.$$

Stability Analysis of an *Algorithm*

Hence that recurrence algorithm is *unstable* for solving $I_n = \int_0^1 \frac{x^n}{x+\alpha} dx$ for values of $|\alpha| > 1$.

e.g. if you code up this recurrence in C / Matlab / etc.:

$$I_{100} = 6.64 \times 10^{-3} \text{ for } \alpha = 0.5$$ Correct!

$$I_{100} = 2.1 \times 10^{22} \text{ for } \alpha = 2.0$$ Wrong!

Summary of FP

- F is not $\mathbb{R}$!
- We can use **round-off error analysis** to bound the errors incurred by floating point operations.
- We can analyze whether initial errors grow or shrink to determine the **stability** of algorithms.

Further reading on FP [Optional]

“What Every Computer Scientist Should Know About Floating-Point Arithmetic”, by David Goldberg, 1991.

Appendix D

What Every Computer Scientist Should Know About Floating-Point Arithmetic

Note – This appendix is an edited reprint of the paper *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, by David Goldberg, published in the March, 1991 issue of Computing Surveys. Copyright 1991, Association for Computing Machinery, Inc., reprinted by permission.

Abstract

Floating-point arithmetic is considered an esoteric subject by many people. This is rather surprising because floating-point is ubiquitous in computer systems. Almost every language has a floating-point datatype; computers from PCs to supercomputers have floating-point accelerators; most compilers will be called upon to compile floating-point algorithms from time to time; and virtually every operating system must respond to floating-point exceptions such as overflow. This paper presents a tutorial on those aspects of floating-point that have a direct impact on designers of computer systems. It begins with background on floating-point representation and rounding error, continues with a discussion of the IEEE floating-point standard, and concludes with numerous examples of how computer builders can better support floating-point.

Categories and Subject Descriptors: (Primary) C.0 [Computer Systems Organization]: General -- *instruction set design*; D.3.4 [Programming Languages]: Processors -- *compilers, optimization*; G.1.0 [Numerical Analysis]: General -- *computer arithmetic, error analysis, numerical algorithms* (Secondary)

D.2.1 [Software Engineering]: Requirements/Specifications -- *languages*; D.3.4 Programming Languages]: Formal Definitions and Theory -- *semantics*; D.4.1 Operating Systems]: Process Management -- *synchronization*.

General Terms: Algorithms, Design, Languages

Additional Key Words and Phrases: Denormalized number, exception, floating-point, floating-point standard, gradual underflow, guard digit, NaN, overflow, relative error, rounding error, rounding mode, ulp, underflow.

Neat Tool: “Herbie”

Some researchers developed a tool to automatically rearrange expressions to reduce FP error: http://herbie.uwplse.org/demo/

http://herbie.uwplse.org/demo/

Herbie web demo

See the main page for more info on Herbie.

Enter a formula below, hit `Enter`, and Herbie will try to improve it.

sqrt(x + 1) - sqrt(x)