Stability and Accuracy

Previously: Considered *problems or questions.*
Next: Considered *methods,* i.e. computational approaches to find solutions.
When is a method *accurate?*

Closeness of the method output to the true answer

When is a method *stable?*

- A method is stable if it produces the exact answer to a ‘nearby’ problem
- “backward stability” satisfies that the simple requirement of the numerical method
- Sensitivity of variation in input is no (or not much) greater than the conditioning of the underlying prob.
Getting into Trouble with Accuracy and Stability

How can I produce inaccurate results?

- apply a method that's inaccurate
- try to compute something ill-conditioned
- try to compute something well-conditioned but use an unstable method
In-Class Activity: Forward/Backward Error

In-class activity: Forward/Backward Error
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?

\[
23.625 = \ldots + 1 \cdot 2^0 + \overbrace{1 \cdot 2^{-1} + 0 \cdot 2^{-2}} + \overbrace{1 \cdot 2^{-3}}
\]

\[
= (10111.101)_2
\]

1 fixed point
Fixed-Point Numbers

Suppose we use units of 64 bits, with 32 bits for exponents $\geq 0$ 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?

rel. error of $2^{-64}$: about 19 digits
For smallest: 0
Floating Point Numbers

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

What pieces do you need to store an FP number?
Floating Point: Implementation, Normalization

Previously: Consider mathematical view of FP. (via example: \((1.101)_2\))
Next: Consider implementation of FP in hardware.

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

Leading bit is (almost) always one.

- not store that \(-1\) in 52 bits for everything but the leading one in the significand

\[
\text{true exponent} = -1023 + 1111111 \quad \text{11-bit biased integer}
\]

in double precision

-1 \begin{array}{cccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array} \quad \text{"two's complement"}
Unrepresentable numbers?

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

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

Demo: Picking apart a floating point number [cleared]
What is the smallest representable number in an FP system with 4 stored bits in the significand and a stored exponent range of \([-7, 8]\)?