COS 323: Computing for the Physical and Social Sciences

Szymon Rusinkiewicz
COS 323

- Course webpage
  
  http://www.cs.princeton.edu/~cos323/

- Instructor:
  
  Szymon Rusinkiewicz (smr@cs)

- TAs:
  
  Connelly Barnes (csbarnes@cs), Cynthia Lu (jingwanl@cs), Dmitry Drutskoy (drutskoy@cs)
What’s This Course About?

- Numerical Algorithms
- Analysis of Data
- Simulation
  - Learn through applications
Scientific Computing

Computers through the 70s/80s were used mostly to solve problems

– Before “personal” computers (!)

– Users were scientists: producers of numerical “codes” rather than consumers of “applications”
Stanisław Ulam with MANIAC I --- about $10^4$ ops/sec
Numerical Analysis

- Algorithms for solving numerical problems
  - Calculus, algebra, data analysis, etc.
  - Applications in all scientific and engineering fields

- Analyze/design algorithms based on:
  - Running time, memory usage
    (both asymptotic and constant factors)
  - Applicability, stability, and accuracy for different classes of problems
Why Is This Hard/Interesting?

• “Numbers” in computers $\neq$ numbers in math
  – Limited precision and range

• Algorithms sometimes don’t give right answer
  – Iterative, randomized, approximate
  – Unstable

• Running time / accuracy / stability tradeoffs
Numbers in Computers

- **“Integers”**
  - Implemented in hardware: fast
  - Mostly sane, except for limited range

- **Floating point**
  - Implemented in most hardware
  - Much larger range
    (e.g. $-2^{31}..2^{31}$ for integers, vs. $-2^{127}..2^{127}$ for FP)
  - Lower precision (e.g. 7 digits vs. 9)
  - “Relative” precision: actual accuracy depends on size
Floating Point Numbers

- Like scientific notation: e.g., $c$ is $2.99792458 \times 10^8 \text{ m/s}$
- This has the form $(\text{multiplier}) \times (\text{base})^{(\text{power})}$
- In the computer,
  - Multiplier is called mantissa
  - Base is almost always 2
  - Power is called exponent
Modern Floating Point Formats

- Almost all computers use IEEE 754 standard
- “Single precision”:
  - 24-bit mantissa, base = 2, 8-bit exponent, 1 bit sign
  - All fits into 32 bits (!)
- “Double precision”:
  - 53-bit mantissa, base = 2, 11-bit exponent, 1 bit sign
  - All fits into 64 bits
- Sometimes also have “extended formats”
Other Number Representations

• **Fixed point**
  - *Absolute* accuracy doesn’t vary with magnitude
  - Represent fractions to a fixed precision
  - Not supported directly in hardware, but can hack it

• “*Infinite precision*”
  - Integers or rationals allocated dynamically
  - Can grow up to available memory
  - No direct support in hardware, but libraries available
Consequences of Floating Point

• “Machine epsilon”: smallest positive number you can add to 1.0 and get something other than 1.0

• For single precision: $\varepsilon \approx 10^{-7}$
  – No such number as 1.000000001
  – Rule of thumb: “almost 7 digits of precision”

• For double: $\varepsilon \approx 2 \times 10^{-16}$
  – Rule of thumb: “not quite 16 digits of precision”

• These are all relative numbers
So What?

- Simple example: add $\frac{1}{10}$ to itself 10 times
Yikes!

- Result: \( \frac{1}{10} + \frac{1}{10} + \ldots \neq 1 \)
- Reason: 0.1 can’t be represented exactly in binary floating point
  - Like \( \frac{1}{3} \) in decimal

- Rule of thumb: comparing floating point numbers for equality is always wrong
More Subtle Problem

• Using quadratic formula

\[ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

to solve \( x^2 - 9999x + 1 = 0 \)

– Only 4 digits: single precision should be OK, right?

• Correct answers: 0.0001… and 9998.999…

• Actual answers in single precision: 0 and 9999

– First answer is 100% off!
– Total cancellation in numerator because \( b^2 >> 4ac \)
Catalog of Errors

- Roundoff error – caused by limitations of floating-point numbers
- Truncation error – caused by stopping an approximate technique early
  - e.g., too few terms of Taylor series for \( \sin(\theta) \)
- Inherent error – limitation on data available
  - GIGO
- Statistical error – too few random samples
Error Tradeoff

[Heath]
Well-Posedness and Sensitivity

• **Problem is well-posed if solution**
  – exists
  – is unique
  – depends continuously on problem data

Otherwise, **problem is ill-posed**

• **Solution may still be sensitive to input data**
  – **Ill-conditioned**: relative change in solution much larger than that in input data

[Heath]
Running Time

- Depending on algorithm, we’ll look at:
  - **Asymptotic analysis** for noniterative algorithms
    (e.g., inverting an $n \times n$ matrix requires time proportional to $n^3$)
  - **Convergence order** for iterative approximate algorithms
    (e.g., an answer to precision $\delta$ might require iterations proportional to $1/\delta$ or $1/\delta^2$)
Simulation and Modeling

• Purposes: quantitative or qualitative prediction, development of intuition, theory testing

• Often requires changing the problem (modeling)
  – Continuous $\rightarrow$ discrete
  – Infinite $\rightarrow$ finite
  – Omitting effects, variables, dimensions
  – Q: how accurate is the resulting approximation?
Example Applications

• Population genetics
• Digital signal processing (including images/audio)
• Simulation of markets
• Classical mechanics
• Weather prediction
This Course

• **Basic techniques**: root finding, optimization, linear systems
• **Data analysis and modeling**: least squares, dimensionality reduction, visualization, statistics
• **Signal processing**: sampling, filtering
• **Integration and differential equations**
• **Data analysis, fitting, and modeling**
• **Simulation**
Mechanics

• Programming assignments
  – Typically more thought than coding
  – Some in MATLAB, some in Java
  – Analysis, writeup counts a lot!

• Final project