CS542G - Breadth in Scientific Computing

Web

- www.cs.ubc.ca/~rbridson/courses/542g
- Course schedule
  - Slides online, but you need to take notes too!
- Reading
  - No text, but if you really want one, try Heath…
  - Relevant papers as we go
- Assignments + Final Exam information
  - Look for Assignment 1
- Resources

Contacting Me

- Robert Bridson
  - X663 (new wing of CS building)
  - Drop by, or make an appointment (safer)
  - 604-822-1993 (or just 21993)
  - email rbridson@cs.ubc.ca
- I always like feedback!
  - Ask questions if I go too fast…

Evaluation

- ~4 assignments (40%)
- Final exam (60%)

MATLAB

- Tutorial Sessions at UBC
- Aimed at students who have not previously used Matlab.
- Wed. Sept. 12, 5 - 7pm, DMP 110.
  www.cs.ubc.ca/~mitchell/matlabResources.html

Floating Point

Numbers

- Many ways of representing real numbers
- Apart from some specialized applications and/or hardware, floating point is pervasive
- **Speed:** while not as simple as fixed point from a hardware point of view, not too bad
  - CPU's, GPU's now tend to have a lot of FP resources
- **Safety:** designed to do as good a job as reasonably possible with fixed size
  - Arbitrary precision numbers can be much more costly (though see J. Shewchuk's work on leveraging FPU's to compute extended precision)
  - Interval arithmetic tends to be overly pessimistic

Floating Point Basics

- Sign, Mantissa, Exponent
- Epsilon
- Rounding
- Absolute Error vs. Relative Error

IEEE Floating Point

- 32-bit and 64-bit versions defined (and more on the way)
- Most modern hardware implements the standard
  - Though it may not be possible to access all capabilities from a given language
  - GPU's etc. often simplify for speed
- Designed to be as safe/accurate/controlled as possible
  - Also allows some neat bit tricks…

IEEE Special Numbers

- +/- infinity
  - When you divide 1/0 for example, or log(0)
  - Can handle some operations consistently
  - Instantly slows down your code
- NaN (Not a Number)
  - The result of an undefined operation e.g. 0/0
  - Any operation with a NaN gives a NaN
    - Clear traceable failure deemed better than silent "graceful" failure!
  - Nan != NaN

Exact numbers in fp

- Integers (up to the range of the mantissa) are exact
- Those integers times a power of two (up to the range of the exponent) are exact
- Other numbers are rounded
  - Simple fractions 1/3, 1/5, 0.1, etc.
  - Very large integers

Floating point gotchas

- Floating point arithmetic is commutative:
  $a+b=b+a$ and $ab=ba$
- But not associative in general:
  $(a+b)+c \approx a+(b+c)$
- Not distributive in general:
  $a(b+c) \approx ab+ac$
- Results may change based on platform, compiler settings, presence of debugging print statements, …
- See required reading on web

Cancellation

- The single biggest issue in fp arithmetic
- Example:
  - Exact arithmetic:
    1.489106 - 1.488463 = 0.000643
  - 4 significant digits in operation:
    1.489 - 1.488 = 0.001
  - Result only has one significant digit (if that)
- When close numbers are subtracted, significant digits cancel, left with bad relative error
- Absolute error is still fine…

Cancellation Example 1

- Can sometimes be easily cured
- For example, solving quadratic
  $ax^2+bx+c=0$
  with real roots

Cancellation Example 2

- Sometimes not obvious to cure
- Estimate the derivative of an unknown function

Accumulation

- 2+eps=2
- (2+eps)+eps=2
- ((2+eps)+eps)+eps=2
- …
- Add any number of eps to 2, always get 2
- But if we add all the eps first, then add to 2, we get a more accurate result

Stability and Well-Posedness

- A problem is well-posed if small perturbations/errors in the “data” lead to small perturbations in solution
  (and solution exists and is unique)
- A numerical method for a well-posed problem might not be well-posed itself: **unstable** method
- Floating-point operations introduce error, even if all else is exact

Performance

- Vectorization, ILP
- Separate fp / int pipelines
- Caches, prefetch
- Page faults
- Multi-core, multi-processors
- Use good libraries when you can!