Introduction to Scientific Computing

Jennifer Houchins

May 24, 2010

What is Scientific Computing?

In a nutshell . . .

- Develop algorithms that use numerical techniques to solve mathematical models
- Models are used to simulate natural phenomena or physical processes
- The quantities involved are *continuous* in nature
  - time, velocity, pressure . . .
- Approximations happen and their effects are considered

A Well-Posed Problem

Three conditions:

- Solution exists
- Solution is unique
- Solution depends continuously on the problem data

Without these three, problem is considered ill-posed

Remember!

- Problem may be well-posed, but solution could still be sensitive to input data
- Care should be taken so that algorithms don't negatively impact the sensitivity

Basic Strategy

The Approach:

- Replace difficult problems with easier ones having the same or closely related solutions
- For example,
  - infinite series $\longrightarrow$ finite sums
  - differential equations $\longrightarrow$ algebraic equations
  - non-linear problems $\longrightarrow$ linear problems
- Ideally, alternative problem solution would be identical to the original problem solution (This is rare!)

Approximations
Why Should We Be Concerned?

Approximations occuring BEFORE computation . . .

- Modeling
  - Simplified or omitted features of what is being modeled
- Empirical Measurements
  - Lab equipment, sample size, noise or bias
- Previous Computation
  - Input from a previous iteration that produces an approximation

Approximations
Why Should We Be Concerned?

Approximations occuring AFTER computation . . .

- Truncation or Discretization
  - Derivatives represented by finite differences
  - Infinite series represented by a finite sum

- Rounding
  - Limitations of expressing real numbers and arithmetic operations **(think PRECISION)**

Error and Its Significance

Significance of error . . .

is related to the magnitude of what is being measured!

For Example :

- An error of 1 when measuring the entire human population
- An error of 1 when counting the occupants of this room

Measuring Error

Absolute Error and Relative Error

Absolute Error

$$\text{approximate value} - \text{true value}$$

Relative Error

$$\frac{\text{absolute error}}{\text{true value}}$$

Normally, error is bound or estimated instead of computed exactly because the true value is unknown

Data Error and Computational Error

- Typical Problem: Compute the value of $f : \mathbb{R} \to \mathbb{R}$ for given argument where
  - x = true value of input
  - $f(x)$ = desired result
  - $\hat{x}$ = approximate input
  - $\hat{f}$ = approximate function actually computed
- Total Error

$$\hat{f}(\hat{x}) - f(x) =$$
$$\hat{f}(\hat{x}) - f(\hat{x}) + f(\hat{x}) - f(x)$$
$$\text{computational error} + \text{propagated data error}$$

- *Choice of algorithm has no effect on propagated data error!*

Truncation Error and Rounding Error

Truncation Error

- Difference between true result and result produced by given algorithm using exact arithmetic
- Results from approximations like truncating infinite series, replacing derivatives with finite differences or terminating iterative processes before convergence

Rounding Error

- Difference between result produced by given algorithm using exact arithmetic and result using the same algorithm using finite-precision or *rounded* arithmetic
- Results from the limitations in representation of real numbers and arithmetic operations performed on them

Forward Error and Backward Error

Want to compute $y = f(x)$ where $f : \mathbb{R} \to \mathbb{R}$ but instead obtain $\hat{y}$ (approximate value)

Forward Error

$$\Delta y = \hat{y} - y$$

Backward Error

$$\Delta x = \hat{x} - x, \text{ where } f(\hat{x}) = \hat{y}$$

Propagated Errors

The Effects of Perturbations

Sensitivity and Conditioning

- The qualitative notion of sensitivity and it's quantitative measure, conditioning, stem from propagated data error
- A problem is *sensitive* (or *ill-conditioned*) if small perturbations in input data cause much larger changes in the solution
- A problem is *insensitive* (or *well-conditioned*) if small perturbations in input data cause a similar relative change in the solution

Propagated Errors

The Effects of Perturbations

Sensitivity and Conditioning

- Condition Number

$$\text{cond} = \frac{|\text{relative change in solution}|}{|\text{relative change in input data}|} = \frac{\left|\frac{\Delta y}{y}\right|}{\left|\frac{\Delta x}{x}\right|}$$

- When Condition Number $\gg 1$, problem is sensitive

Propagated Errors

The Effects of Perturbations

Stability is the relative insensitivity of an algorithm to perturbations during computation caused by approximations

Accuracy is how close the computed solution and the true solution are to one another

- Note that having a stable algorithm doesn't mean that we have an accurate algorithm.
- BOTH conditioning and stability affect the accuracy
- We can achieve an accurate solution by applying a stable algorithm to a well-conditioned model

Representations of $\mathbb{R}$

Floating-point number system $\mathbb{F}$

- Characterized by four integers

β	Base or radix
p	Precision
$[L, U]$	exponent range

- Number x is represented

$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_{p-1}}{\beta^{p-1}} \right) \beta^E$$

where $0 \leq d_i \leq \beta - 1$, $i = 0, \ldots, p-1$, and $L \leq E \leq U$

Floating-point number system $\mathbb{F}$

- Portion in parentheses represent by a string of p base-β digits is called the mantissa
- E is called the exponent or characteristic of x
- The portion $d_1 d_2 \ldots d_{p-1}$ of the mantissa is called the fraction
- The sign, exponent, and mantissa are stored in separate fixed-width fields of a floating-point word
- Most modern computers use binary ($\beta = 2$) arithmetic

Normalization

A floating-point number system is called *normalized* if the leading digit d_0 is always nonzero, unless the number represented is zero

Why Normalize $\mathbb{F}$?

- The representation of each number is unique
- Maximize precision because no digits wasted on leading zeros
- Gain one extra bit of precision for a given field width in the Binary system ($\beta = 2$) because the leading bit is always 1 and need not be stored

Properties of $\mathbb{F}$

- $\mathbb{F}$ is finite and discrete
- The number of normalized floating-point numbers is given by

$$2(\beta - 1)\beta^{p-1}(U - L + 1) + 1$$

- The smallest positive normalized floating-point number is called the Underflow Level

$$\text{Underflow Level} = \text{UFL} = \beta^{L}$$

- Similarly, the largest is called the Overflow Level

$$\text{Overflow Level} = \text{OFL} = \beta^{U+1}(1 - \beta^{-p})$$

Rounding

- When a real number is exactly representable in a given floating-point system, it is called a machine number
- When this isn't the case, the number has to be approximated a *nearby* floating-point number
- The process of choosing this *nearby* floating-point number is called rounding
- Naturally, the error produced from this approximation is called rounding error or roundoff error

Rounding

Chop

The base-β expansion is truncated after the $(p-1)$st digit

Round to nearest

Use the nearest floating-point number; if there is a tie, take the floating-point number whose last stored digit is even

Machine Precision

- The accuracy of $\mathbb{F}$ is characterized by a quantity known as machine precision (aka machine epsilon, aka unit roundoff)

$$\epsilon_{\text{mach}}$$

- Rounding by chopping gives

$$\epsilon_{\text{mach}} = \beta^{1-p}$$

- Rounding to nearest gives

$$\epsilon_{\text{mach}} = \frac{1}{2}\beta^{1-p}$$

Floating Point Arithmetic

- When adding or subtracting two floating-point numbers, the exponents must match before the mantissas can be added or subtracted
- If the exponents don't match initially, the mantissa of one the numbers needs to be shifted
- Problem! When shift is performed, trailing digits of smaller number are shifted off the mantissa field and then the correct result of the arithmetic operation cannot be represented exactly in the floating-point system
- Worst Case Scenario: The operand of smaller magnitude is lost completely!

Cancellation

And you thought rounding was bad!

- Subtraction between two p-digit numbers that have the same sign and similar magnitudes yields a result with fewer than p significant digits
- If the two numbers do not differ in magnitude by more than a factor of two, the result is always exactly representable
- Reason: the leading digits of the two numbers cancel!
- Despite exactness of the result, cancellation can imply serious loss of information

In Conclusion

One thing to remember

Don't panic!

For Further Reading

- M. Heath, *Scientific Computing: An Introductory Survey*, McGraw-Hill, 2002
- A. Shiflet and G. Shiflet, *Introduction to Computational Science: Modeling and Simulation for the Sciences*, Princeton University Press, 2006
- R. Landau and M. Páez, *Computational Physics: Problem Solving with Computers*, Wiley, 1997