Scientific Computing

A practical introduction to computational problem solving – Methods/Software

Instructor: M. Shelley, WWH1105, shelley@cims.nyu.edu, 8-3284

OHs: Wednesday, 5-7, or by appointment.

Grader: Ken Ho.

Text: *Principles of Scientific Computing* by Jonathan Goodman and David Bindel. Posted on the course web-page.

Supplementary Text:

Supplementary Notes: See webpage of Aleks Donev, Spring 2011 Instructor.

This is not a programming course, but does require programming.

Some useful languages: Matlab (Courant labs, or get student editions), C, C++, Fortran, Python

Grades: homeworks approximately bi-weekly (6 or so, for 80%), plus larger final project (20%).

Topics:

- Some basic Numerical Analysis: Interpolation & Extrapolation, Approximating derivatives and integrals, local and global error, checking convergence.
- Numerical Linear Algebra: Direct methods for solving linear systems of equations. Conditioning. Matrix Eigenvalues, Matrix decomposition: SVD.
- Fourier series and the Fast Fourier Transform
- Solving nonlinear systems of equations, and optimization.
- Time-stepping ODEs (and PDEs, if time allows)
- Monte Carlo methods

Lecture 1:

First fact: Scientific computing is the art of approximate computing, understanding sources of error, its growth and control, and accuracy.

Sources of Error:

- round-off error – finite representations of real numbers, and the results of pair operations – $\times, +, -, /$ – between them.
- truncation or approximation error.

Examples:

1. **a**. The formula $\frac{f(x+h)-f(x)}{h}$ is an approximation for $f'(x)$, and improves as $h \to 0$. There are better ones.

b. $\sum_{k=1}^{N} \frac{x^k}{k!}$ is an approximation to e^x, and is especially good for x very small.

c. $\frac{b-a}{N} \sum_{k=0}^{N-1} f(a + k\frac{b-a}{N})$ is an approximation to $\int_a^b f(x)dx$ that improves as $N \to \infty$. There are better approximation formulae.

- Termination of iteration errors.
- Statistical errors.

Forms of errors: In approximation A by $\hat{A}$,

$$e = \hat{A} - A \Rightarrow \hat{A} = A + e$$

is the *absolute error* and is dimensional (has units: meters, seconds, etc).

$$\varepsilon = \frac{e}{A} = \frac{\hat{A} - A}{A} \Rightarrow \hat{A} = (1 + \varepsilon)A$$

is the *relative error* and is dimensionless, and hence more meaningful.

Side-note: $-\log_{10}|\varepsilon| \cong$ the number of digits of agreement between A and $\hat{A}$.

One important source of growth in relative error is *cancellation*.

Example: $A = 0.233999$ and $B = 0.233888$ are each known to 6 digits of accuracy, and agree in the first three. $C = A - B = 0.000111$ is known to only 3 digits of accuracy. Lost off significant digits is a real and present danger, especially through accumulation of such errors in many steps.

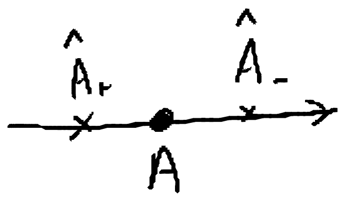
Floating Point Representation and Round-Off Error:

On modern computers, real numbers are represented approximately by *finite precision floating point numbers*: Consider *single precision* using 32 bit strings of 0's and 1's (binary).

$$\begin{array}{ccc} 1 & 10001010 & 01\ldots1 \\ \text{s} & \text{e} & \text{f} \\ = \pm & 2^{e-127} & \cdot\ (1.f)_2 \end{array}$$

So, 1 bit for sign, 8 bits for exponent in $[-127, 129]$ where e is chosen s.t. the 1^{st} bit of the mantissa is always 1 (meaning of floating point), with 23 bits for the remaining mantissa.

Let B and C be two finite precision floating point numbers.



$A = B/C$ will generally not be an FPFP number. **Rounding**: Find $\hat{A}$ closest to A.

Relative rounding error bounded by 1/2 the relative distance between two FP numbers:

$$\gamma = \frac{(1.f_+)_2 - (1.f_-)_2}{(1.f_-)_2} = \frac{2^{-23}}{(1.f_-)_2} \le 2^{-23}$$

smallest value of $(1.f_-)_2$ is 1.

Hence,

$\gamma_{\max} = 2^{-23}$ and so maximal rounding error is

$\frac{1}{2}\gamma_{\max} = 2^{-24} \cong 6 \cdot 10^{-8}$. Called *machine precision* $\varepsilon_{\text{mach}}$

Double precision: Nearly all computing is done in 64 bit arithmetic:
1 bit for sign
11 bits for exponent
52 bits for *f*.
$\Rightarrow \varepsilon_{\text{mach}} = 2^{-53} \cong 10^{-16}$. Often said: "16 digits of precision".

Nearly all computing is done in 64 bit double precision. Gives broader range of exponents and more available fractions. There is demand for 128 bit and higher.

Truncation or approximation errors: The error in analytical approximations.

$$D_h f(x) = \frac{f(x+h) - f(x)}{h}$$

is an approximation to $f'(x)$. Taylor series with remainder:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \cdots \Rightarrow$$

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \boxed{\tfrac{1}{2}hf''(x)} + \cdots$$

dominant absolute
approximation error

$$\Rightarrow \varepsilon_h(x) = \frac{D_h f(x) - f'(x)}{f'(x)} \cong \frac{1}{2} h \frac{f''(x)}{f'(x)}, \text{ assuming } f'(x) \neq 0$$

So, approximation errors – either absolute or relative – should decrease roughly as $Const \times h$.

Let's check. Consider $x = 1$ and let $\varepsilon_h(1) = \frac{\frac{\sin(1+h)-\sin(1)}{h} - \cos 1}{\cos 1}$ (relative error). Compute in 16 digit arithmetic.

h	ε_h	
10^{-2}	7.8×10^{-3}	decreases
10^{-4}	7.8×10^{-5}	linearly with h
10^{-6}	7.8×10^{-7}	
10^{-8}	5.5×10^{-9}	loss of accuracy
10^{-10}	1.1×10^{-7}	cancellation
10^{-12}	8.0×10^{-5}	

At first *approximation error* dominates and error decreases. Then, loss of accuracy through cancellation begins to dominate.

Quick analysis: In forming the divided difference we are using FPFP numbers:

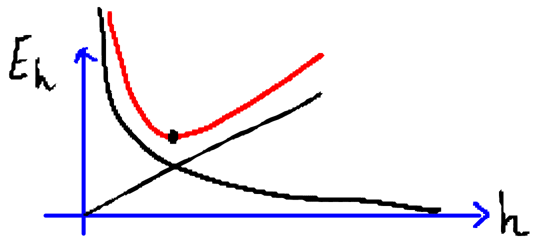
$$D_h f \cong \frac{\hat{f}_1 - \hat{f}_2}{h}$$

where $\hat{f}_{1,2} = f_{1,2} + e^h_{1,2}\varepsilon_{\mathrm{m}}$, where $e^h_{1,2}\varepsilon_{\mathrm{m}}$ are the rounding errors with $e^h_{1,2}$ being order one coefficients. Then

$$\frac{\hat{f}_1 - \hat{f}_2}{h} = D_h f + \Delta e^h \frac{\varepsilon_{\mathrm{m}}}{h}$$
$$\cong f' + \frac{1}{2} h f'' + \Delta e^h \frac{\varepsilon_{\mathrm{m}}}{h}$$

Hence the error is

$$E_h = \frac{1}{2} h f'' + \Delta e^h \frac{\varepsilon_{\mathrm{m}}}{h}$$



which is plotted above. Assuming that Δe is a (roughly) a constant, then the minimum error occurs when $h \sim \varepsilon_{\mathrm{m}}^{1/2}$ which is consistent with the numerical results.

Lesson: The approximation of derivatives is sensitve to loss of significance through cancellation errors. h cannot be taken too small.

Note: Here I might write $\varepsilon_h(x) = O(h)$ meaning $\frac{|\varepsilon_h(x)|}{h}$ is bounded as $h \to 0$.

Other examples again:

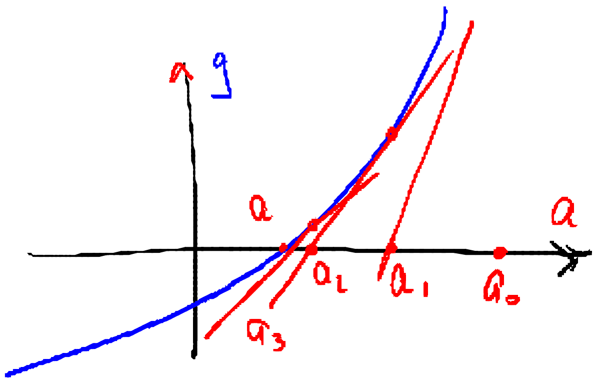
- $\sum_{k=1}^{N} \frac{x^k}{k!}$ is an approximation to e^x. The approximation error is given by $\frac{\zeta^{N+1}}{(N+1)!}$ for some $\zeta \in [0,x]$.
- $h\sum_{k=0}^{N-1} f(a+h)$ with $h = (b-a)/N$ approximates $\int_a^b f(x)dx$. Approximation error can be expressed through the Euler-MacLaurin formulae. *Quadrature* does not typicaly suffer from cancellation errors.

Iterative Methods and Termination Errors

Task: Solve $g(a) = 0$ for a. Typically this cannot be solved in closed form and instead a is sought through convergence of a sequence $a_k \to a$ as $k \to \infty$.

Example: *Newton's method*. If g is diffentiable we could try

$$a_{k+1} = a_k - g(a_k)/g'(a_k)$$



This process may or may not converge, but even if it does, it must eventually be terminated. Good iterative methods can produce and approximation to a that are essentially as accurate as the finite precision allows.

Stopping Criteria:

$$\frac{|a_k - a_{k-1}|}{|a_k|} < \text{ tolerance, or}$$

$$\frac{|g(a_{k-1})|}{|g(a_0)|} < \text{ tolerance, etc.}$$

which will give a relative termination error of $\frac{|a-a_k|}{|a|}$.

Statistical Errors:

Approximate $A = E[X]$, where X is a random variable, by

$$A_N = \frac{1}{N}\sum_{k=1}^{N} X_k$$

where $X_1,\ldots,X_N$ are independent samples of X.

Basic Theorem: $A_N \to A$ as $N \to \infty$ (almost assuredly). But, errors for finite N are large and convergence is very slow.

Conditioning and the Condition Number: Input errors can arise from many sources, such as errors in data accuracy, finite sample size, or because of rounding errors. The condition number K measures, in a non-dimensional way, the sensitivity of output (the "solution") to small changes in the input.

Smallest input error is rounding error $\varepsilon_{\text{mach}}$;
Error in output is $K\varepsilon_{\text{mach}}$.

Computations with $K \sim O(1)$ are call well-conditioned. There is little if any amplification of input errors.

Condition numbers can easily be large, and large K arise commonly in the solution of large systems of linear equations. Computations with $K >> 1$ are called ill-conditioned, and algorithms should be re-designed if possible to avoid this difficulty. Solving large least-squares problems via the Normal Equations, or solving 1^{st}-kind integral equations, can be ill-conditioned. If $K = 10^8$ and input error is $\varepsilon_{\text{mach}} = 10^{-16}$, then automatic error of 10^{-8}, or loss of 1/2 of the digits!

Simplest case: Compute $A(x)$ with input $x + \Delta x$.

$$\Delta A = A(x + \Delta x) - A(x)$$

Define K by relating relative output error to relative input error as

$$\left|\frac{\Delta A}{A}\right| \cong K\left|\frac{\Delta x}{x}\right| \text{ where } \left|\frac{\Delta x}{x}\right| \cong \varepsilon_{\text{mach}}$$

$$\left|\frac{\Delta A}{A}\right| \cong K\left|\frac{\Delta x}{x}\right| \Leftrightarrow \left|\frac{A'\Delta x}{A}\right| \cong K\left|\frac{\Delta x}{x}\right|$$

$$\Leftrightarrow K \cong \left|\frac{A'(x)x}{A(x)}\right|$$

Most simple binary operations are well-conditioned. Rather it results from algorithms that have many, as in solving linear equations.