Fundamental Concepts in Computational and Applied Mathematics

Juan Meza [1]

Math 298 Fall 2014

Course Goals

- Introduce fundamental concepts in computational and applied mathematics.
- Highlight some of the classic NA papers and algorithms.
- Demonstrate use of these concepts in real world applications.

Disclaimer

Please note that we will not be able to go into any one subject in much depth. This course is a whirlwind tour of the fundamentals.

Learning Outcomes

- Be familiar with key mathematical concepts and computational frameworks used in developing numerical algorithms.
- Understand some of the basic skills and resources necessary to start research in computational and applied mathematics.
- Be aware of basic communications skills needed to present mathematics clearly to a broad audience in writing and in speech.

Ground Rules/Grading

50% Class participation – critical to getting the most out of this course

25% Some assigned reading and writing assignments

25% Presentations at the end of the semester

Theory vs. Practice

- In theory there is no difference between theory and practice.
- In practice there is.

Nick Trefethen's 13 Classic Numerical Analysis Papers.

- Cover some of the most important NA papers in the last 60 years
- Cover most of the areas mentioned in the next 3 slides
- It is your responsibility to be familiar with all of the papers
- Note: Copies of all of these have been uploaded into CROPS

Phil Colella's 7 Dwarfs

1. Dense linear algebra
2. Sparse line algebra
3. Spectral methods (Fast Fourier transform)
4. N-body (particle) methods
5. Structured grids
6. Unstructured grids
7. Monte Carlo

Defining Software Requirements for Scientific Computing, P. Colella, DARPA presentation, 2004.

13 Motifs – Patterson et al.

1. First 6 Dwarfs (Dense linear algebra, Sparse line algebra, Spectral methods, N-body methods, Structured grids, Unstructured grids) $+$
2. MapReduce
3. Combinational Logic
4. Graph Traversal
5. Dynamic Programming
6. Backtrack and Branch-and-Bound
7. Graphical Models
8. Finite State Machines

The Landscape of Parallel Computing View from Berkeley

Top 10 Algorithms of the Century

1946 Monte Carlo

1947 Simplex method for LP

1950 Conjugate gradient (Krylov subspace iteration methods)

1951 Decompositional approach to matrix computations

1957 Fortran compiler

1959 QR for computing eigenvalues

1962 Quicksort

1965 Fast Fourier transform

1977 PSLQ (integer relation detection algorithm)

1987 Fast Multipole

Computing in Science and Engineering, January/February 2000, Vol. 2, No. 1

Finite Precision

- Difference between infinite precision and computer arithmetic
- You will almost always make an error when approximating a mathematical object on a computer
- One of the major types of errors is discretization(truncation), which is the difference between the solution of the discrete problem and the exact solution (representation) of the mathematical object

Finite Precision

Example

Suppose you have a function defined by an infinite series and you "truncate" it at some point, e.g.

$$\begin{aligned} \exp(x) &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \\ &\approx 1 + x \end{aligned}$$

Discussion:

How many terms in the series should we use? Are all values of x the same?

Finite Precision

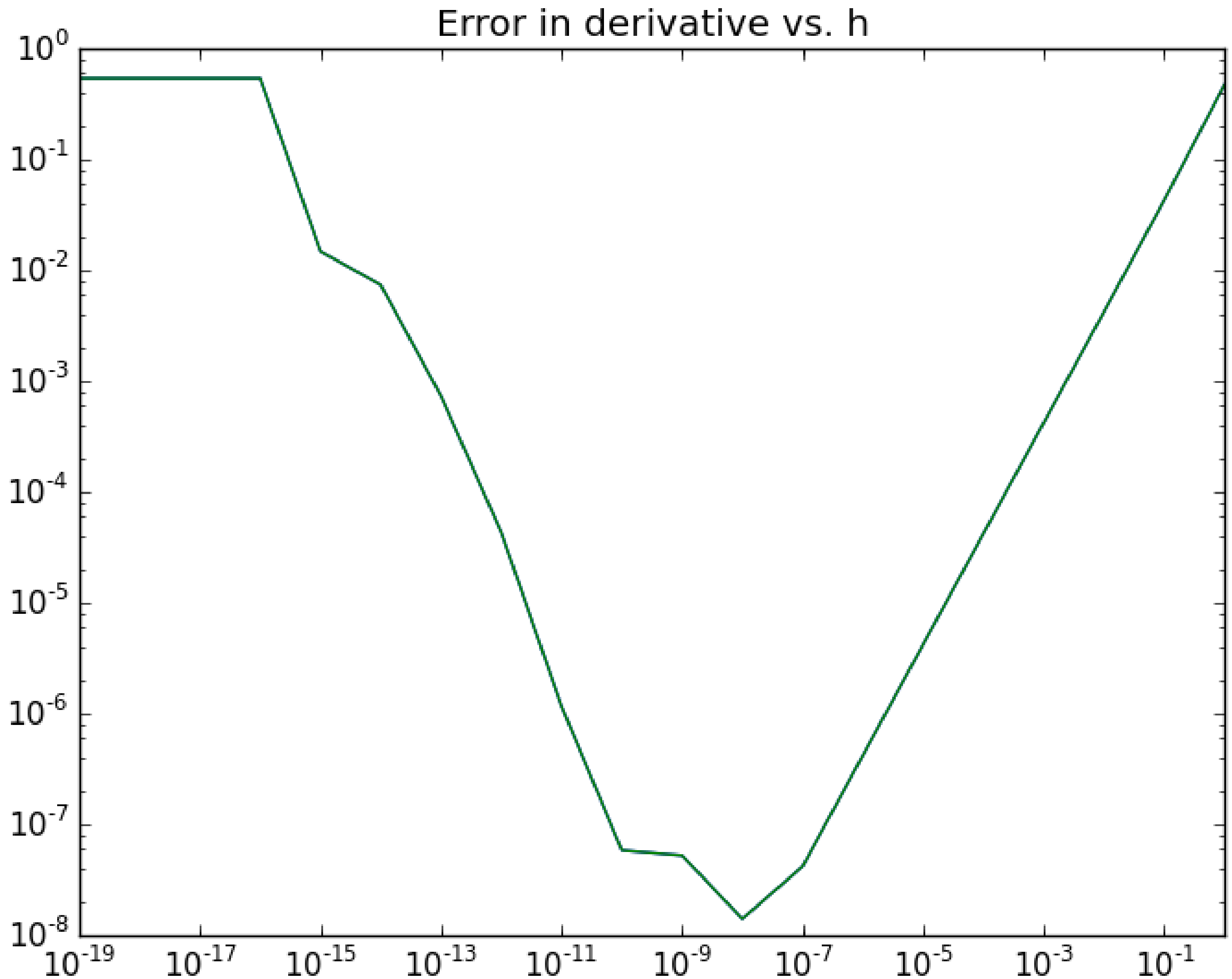
Example

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Discussion:

How do we pick h to get the best approximation? What value of h should we choose?

Finite Precision Example: $f(x) = \sin(x), x = 1.0$



Python Code

```
import numpy as np
from numpy import *
import matplotlib.pyplot as plt

n = 20
x = np.ones(n) # create an array of ones

h0   = 0.1*np.ones(n)
h    = h0**np.arange(n)     # create an array of increasingly
                            # smaller h values
yhat = (sin(x+h) - sin(x))/h # compute finite difference
err  = abs(yhat - cos(x))    # error in numerical approx.

plt.loglog(h,err) # Plot the results
plt.title('Error in derivative vs. h')
plt.show
```

Finite Precision

Important Lesson:

Don't solve an approximate problem too exactly!

Rule of Thumb

To approximate

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

choose h to perturb about half of the digits in x.

Finite Precision Real World Examples

Applications requiring extra precision

- Climate Modeling
- Supernova Simulations
- Coulomb N-Body Atomic System Simulations
- Computational Geometry and Grid Generation
- Many, many more, . . .

Reference: **High-Precision Floating-Point Arithmetic in Scientific Computation**, David H. Bailey, Computing in Science & Engineering, IEEE, May/June 2005.

Conditioning

- Suppose we would like to know how perturbing x will affect the value of $y = f(x)$

Definition (Condition Number)

$$\kappa_f(x) = \frac{|x| \cdot |f'(x)|}{|f(x)|}$$

- Then
$$\frac{|\hat{y} - y|}{|y|} \approx \kappa_f(x) \cdot \frac{|\hat{x} - x|}{|x|}$$
- Fact: $\#$ decimal digits to which two numbers agree $\approx -\log_{10}$(relative error)

Rule of Thumb

You will lose approximately $\log_{10}(\kappa_f(x))$ decimal digits when computing anything

Stability

- Informal definition of stability. An algorithm for computing $\hat{y} = f(x)$ is **stable** if it returns a $\hat{y}$ that satisfies:

$$\frac{|\hat{y} - y|}{|y|} \approx \kappa_f(x) \cdot \frac{|\hat{x} - x|}{|x|}.$$

Discussion:

What do you think of the following algorithm for computing $\exp(x)$?

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

Summary

- Beware catastrophic cancellation - know limits of precision
- Learn about the conditioning of a problem
- Choose stable algorithms so as to not introduce any more loss of precision than is necessary
- Always remember
    - Conditioning is fundamentally a characteristic of the problem while
    - Stability is related to algorithms

Further Reading I

M.L. Overton.
Numerical Computing with IEEE Floating Point Arithmetic.
SIAM, 2001.

D. Goldberg.
What Every Computer Scientist Should Know About Floating Point Arithmetic
Computing Surveys, ACM, Vol. 23, No. 1, pp. 5–48, 1991.

D.H. Bailey.
High-Precision Floating-Point Arithmetic in Scientific Computation
Computing in Science & Engineering, IEEE, May/June 2005.