Lecture 1
Introduction to Numerical Computing

J. B. Schroder

Department of Mathematics and Statistics
University of New Mexico
Numerical Computing?

Numerical Analysis?
Numerical Computation vs. Symbolic Computation

- Numerical Computation: involve numbers directly
  - manipulate numbers to produce a **numerical** result
- Symbolic Computation: symbols represent numbers
  - manipulate symbols according to mathematical rules to produce a symbolic result

**Example (numerical)**

\[
\frac{(17.36)^2 - 1}{17.36 + 1} = 16.36
\]

**Example (symbolic)**

\[
\frac{x^2 - 1}{x + 1} = x - 1
\]
Analytic Solution vs. Numerical Solution

- **Analytic Solution (a.k.a. symbolic):** The exact numerical or symbolic representation of the solution
  - may use special characters such as $\pi$, $e$, or $\tan(83)$
- **Numerical Solution:** The computational representation of the solution
  - entirely numerical

---

**Example (analytic)**

\[
\frac{1}{4} \\
\frac{1}{3} \\
\pi \\
\tan(83)
\]

**Example (numerical)**

0.25

0.33333\ldots (?)

3.14159\ldots (?)

0.88472\ldots (?)
Numerical Computation and Approximation

- Numerical Approximation is needed to carry out the steps in the numerical calculation. The overall process is a numerical computation.

Example (symbolic computation, numerical solution)

\[
\frac{1}{2} + \frac{1}{3} + \frac{1}{4} - 1 = \frac{1}{12} = 0.0833333333\ldots
\]

Example (numerical computation, numerical approximation)

\[
0.500 + 0.333 + 0.250 - 1.000 = 0.083
\]
Method vs. Algorithm vs. Implementation

- Method: a general (mathematical) framework describing the solution process
- Algorithm: a detailed description of executing the method
- Implementation: a particular instantiation of the algorithm

- Is it a “good” method?
- Is it a robust algorithm?
- Is it a fast implementation?
The Big Theme

Accuracy

Cost
History: Numerical Algorithms

date to 1650 BC: The Rhind Papyrus of ancient Egypt contains 85 problems; many use numerical algorithms (T. Chartier, Davidson)

Approximates $\pi$ with $\left(\frac{8}{9}\right)^2 \times 4 \approx 3.1605$
287-212BC developed the "Method of Exhaustion"

Method for determining $\pi$

- find the length of the perimeter of a polygon inscribed inside a circle of radius $1/2$
- find the perimeter of a polygon circumscribed outside a circle of radius $1/2$
- the value of $\pi$ is between these two lengths
A circle is not a polygon

A circle is a polygon with an infinite number of sides

\[ C_n = \text{circumference of an n-sided polygon inscribed in a circle of radius } \frac{1}{2} \]

\[ \lim_{n \to \infty} = \pi \]

Archimedes determined

\[ \frac{223}{71} < \pi < \frac{22}{7} \]

\[ 3.1408 < \pi < 3.1429 \]

two places of accuracy....

http://www.pbs.org/wgbh/nova/archimedes/pi.html
Around 1700, John Machin discovered the trig identity

\[ \pi = 16 \arctan \left( \frac{1}{5} \right) - 4 \arctan \left( \frac{1}{239} \right) \]

- Led to calculation of the first 100 digits of \( \pi \)
- Uses the Taylor series of \( \arctan \) in the algorithm

\[ \arctan (x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} \ldots \]

- Used until 1973 to find the first Million digits
**Definition (Trefethen)**

Study of algorithms for the problems of continuous mathematics

We’ve been doing this since Calculus (and before!)

- Riemann sum for calculating a definite integral
- Newton’s Method
- Taylor’s Series expansion + truncation
Big Questions

- How algorithms work and how they fail
- Why algorithms work and why they fail

- Connects mathematics and computer science
- Need mathematical theory, computer programming, and scientific inquiry
A Numerical Analyst needs

- computational knowledge (e.g. programming skills)
- understanding of the application (physical intuition for validation)
- mathematical ability to construct a meaningful algorithm
Numerical Analysis

Numerical focus:

**Approximation**  An approximate solution is sought. How close is this to the desired solution?

**Efficiency**  How fast and cheap (memory) can we compute a solution?

**Stability**  Is the solution sensitive to small variations in the problem setup?

**Error**  What is the role of finite precision of our computers?
Numerical Analysis

Why?
- Numerical methods improve scientific simulation
- Some disasters attributable to bad numerical computing (Douglas Arnold)

▶ The Patriot Missile failure, in Dhara, Saudi Arabia, on February 25, 1991 which resulted in 28 deaths, is ultimately attributable to poor handling of rounding errors.
▶ The explosion of the Ariane 5 rocket just after lift-off on its maiden voyage off French Guiana, on June 4, 1996, was ultimately the consequence of a simple overflow.
▶ The sinking of the Sleipner A offshore platform in Gandsfjorden near Stavanger, Norway, on August 23, 1991, resulted in a loss of nearly one billion dollars. It was found to be the result of inaccurate finite element analysis.
I thought we were studying “Numerical Computing”?! 

Numerical analysis is the study of numerical computing

- Numerical Analysis: understanding general behavior of numerical computing
- Numerical Computing: understanding how to apply certain methods to solve specific tasks
- As computational scientists, we need to understand the concepts of numerical analysis and implementation aspects of the numerical computing

We thus focus on

- Matlab implementation
  - fast learning curve
  - quick time-to-production: low development times
  - a major development environment in scientific computing
  - integrated graphics
- Errors in computation
- Specific methods for solving linear and nonlinear systems, root finding, integrating, interpolation, etc.
Applications: Mathematics, Engineering, Computer Science...

In Engineering

Biomolecular Systems

Rocket Simulations

Crack propagation
In Computer Science?

**AI**: transitions, state systems, patterns (eigenvalues, linear algebra)

**Informatics**: Google matrix, Amazon recommendations (eigenvalues, linear algebra, sparse matrices, iterative methods)

**Graphics**: image compression, representation of curves/surfaces/lighting (interpolation, differentiation, etc)

**Security**: ssh (random numbers)

**Economics/Finance**: modeling/simulation of financial data (monte carlo)

**Scientific computing**: design of algorithms for high performance/parallel computing.
All course materials are at
http://math.unm.edu/~schroder/2020_Spring_375/index.html

**Book:**
OR
Numerical Analysis by T. Sauer, 3rd Edition

*You will need one of these books as a reference, to buttress the in-course slides and derivations. Sauer is likely a better general reference book, but some course material will more closely follow Cheney and Kincaid.*

Jacob’s contact:

- **Office:** SMLC 332
- **Office Hours:** Wednesday 3:30pm-5:00pm and Thursday 2:00pm-3:30pm
- **email:** jbschroder@unm.edu
All course materials are at http://math.unm.edu/~schroder/2020_Spring_375/index.html

Grades:
- Homework 45%
- Midterm Exam 25%
- Final Exam 30%

Homework:
- Assigned approximately every 1 or 2 weeks
- Submitted as a report
- Source code included upon request
- No graphics by hand
- Accepted for 50% credit up to one week late
- No dropped scores
- Collaborate, groups of 2
- Submit one report, but cite collaborators at the top
Finally...

All course materials are at
http://math.unm.edu/~schroder/2020_Spring_375/index.html

Schedule and Notes:
- Course structure is in syllabus (Let’s review it)
- Course materials on the web (Let’s review website)
- Look at the MATLAB tutorial on website
- Everything is **tentative** including midterm exam date
- Final exam is fixed
- Grades posted on UNM Learn

Questions?
Preliminaries...

- A computational experiment...
- relative vs absolute error
- Matlab

Goals

- To think like a numerical analyst
- Understanding error in numerical computing
- Utilizing Matlab
Typical Process

Goal
Find $f'(1.0)$ for function $f(x) = e^x$.

Problem
Don’t (want to need to) know $f'(x)$ explicitly.

Method
Use definition.

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

Method

Use definition.

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

Listing 1: Algorithm (pseudocode)

1. \( h = 1 \)
2. \( x = 1 \)
3. \( \textbf{for} \ i = 1 \ \textbf{to} \ 20 \ \textbf{do} \)
4. \( h = h/2 \)
5. \( y = (f(x + h) - f(x))/h \)
6. \( \text{err} = |(f(x) - y)| \)
7. \( \textbf{end} \)

Listing 2: Implementation (Matlab)

1. \( f = @(x) \exp(x); \)
2. \( h = 1; \)
3. \( x = 1; \)
4. \( \textbf{for} \ i = 1:20 \)
5. \( h = h/2; \)
6. \( y = (f(x+h) - f(x))/h; \)
7. \( \text{err}(i) = \text{abs}(f(x) - y); \)
8. \( \textbf{end} \)
Measuring Error

- Here we used the absolute error:

\[ \text{Error}(x_a) = |x_t - x_a| \]

\[ = |\text{true value} - \text{approximate value}| \]

- This doesn’t tell the whole story. For example, if the values are large, like billions, then an Error of 100 is small. If the values are smaller, say around 10, then an Error of 100 is large. We need the relative error:

\[ \text{Rel}(x_a) = \left| \frac{x_t - x_a}{x_t} \right| \]

\[ = \left| \frac{\text{true value} - \text{approximate value}}{\text{true value}} \right| \]

- More on types of errors in future...
What is MATLAB?

- both a computing environment and a language
- initially developed as an easy interface to LAPACK: Linear Algebra Package (FORTRAN libraries)
- MATLAB
- Written in C. For matrix computations, it calls C/Fortran libraries == Fast
- Matlab + “Toolboxes”
  - Symbolic Math Toolbox: mathematical manipulation of symbols
  - Partial Differential Equation Toolbox: tools for solving PDEs in 2-D
  - Statistics Toolbox: statistical data analysis
  - Image processing toolbox: visualization and image analysis
  - Bioinformatics toolbox: computational molecular biology
  - Compiler: application development
  - many many more.
- http://www.mathworks.com
Why Matlab?

The Good:
- Fast development times
- no compiling, easy debugging
- accessible syntax and language constructs
- in-house graphics capabilities
- tons of basic "libraries" or functions available
- many more complicated "toolboxes" can be added

The Bad:
- small coding mistakes can result in slow code
- loops are computationally intensive
- language is limited: no templates etc.

The Ugly:
- proprietary (but the language format is open)
- expensive (but free for UNM students/faculty?)
- the open source substitute, Octave, is not fully compatible

Now, you can fire up Matlab and start the tutorial...
What is Python?

Full fledged, easy to use programming language

The Good:

- All the good stuff that Matlab has!
- Large range of libraries
- Free to use forever!
- Language has full object oriented support
- Jupyter notebooks are great for displaying text, code and figures in the same file

The Bad:

- Installing it and the associated libraries can be a pain (although many good installers now available)
- Slightly steeper learning curve compared to Matlab