

Lecture 00

Introduction to Numerical Computing

T. Takahashi

Department of Mathematics and Statistics
University of New Mexico



Scope



What is Numerical Computing?

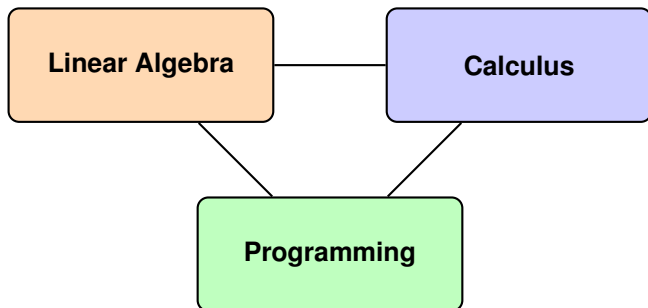
- Numerical: Study of real numbers, and problems of continuous mathematics, e.g. $y = f(x), x \in \mathbb{R}$.
- Computing: Study of algorithms and computer architecture.

Numerical Computing

Numerical computing is the study of digital representation of numbers, and algorithms to solve problems of continuous mathematics.



Three Major Components



Some Algorithms you will Learn

Derivatives

- Forward Difference
- Central Difference
- Richardson Extrapolation

Integrals

- Riemann Sum
- Trapezoidal Rule
- Gaussian Quadrature

Linear Algebra

- Gaussian Elimination
- LU Decomposition
- Power Method

Numerical vs. Symbolic Computation

This course is on **Numerical Computation**, not **Symbolic Computation**

Numerical Computation

- Operate on numbers to get a numerical result
- Example:

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

Symbolic Computation

- Operate on symbols to get an expression
- Example:

$$\frac{x^2 - 1}{x + 1} = x - 1$$



Analytic vs. Numerical Solution

Problem: What number x satisfies:

$$e^x = 3\pi$$

Analytical Solution

- The exact numerical or symbolic representation of the solution.
- May use special characters such as π , e , or even variables
- Analytical Solution:

$$x = \ln(3\pi)$$

Numerical Solution

- Numerical Solution: The computational representation of the solution
- Entirely numerical. It is usually an approximation.
- Numerical Solution:

$$x \approx 2.24334217$$

Levels of Abstraction

Method

A **method** is a general framework describing the solution process.

Algorithm

An **algorithm** is a detailed description of executing the method. Think of pseudocode.

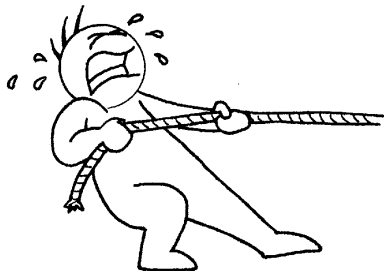
Implementation

An **implementation** is a particular instantiation of an algorithm. Think of software.

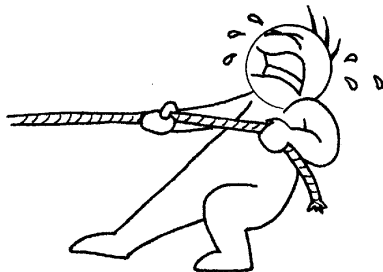
divide-and-conquer sort → heap sort → the actual code



Major Theme



Accuracy



Cost

- There is no “best” algorithm
- You must to sacrifice accuracy for speed, and vice versa



History of Pi



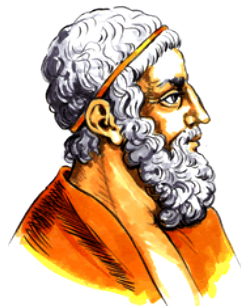
Ancient Egypt

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



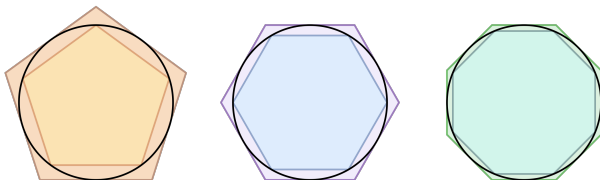
- Approximates π with $(8/9)^2 * 4 \approx 3.1605$
- More of a formula than an algorithm

- 287-212BC - **"Method of Exhaustion"**
- Method for determining π
 - draw an n -sided polygon inscribing the circle
 - draw an n -sided polygon circumscribing the circle
 - estimate the circumference
 - calculate pi: $\pi \approx C/D$
- Two digits of precision



Importance: Method of Exhaustion

- C_n = circumference of an n -sided polygon
- $\lim_{n \rightarrow \infty} C_n = \pi$



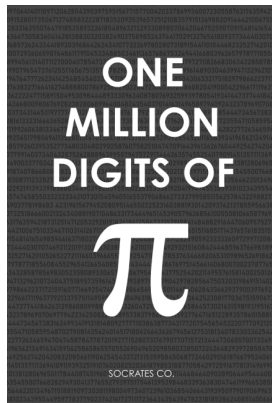
Key Idea

- The number n is a parameter for the algorithm.
- As n grows, algorithm gets **more expensive** but also **more accurate**.



- 1700s - John Machin discovered used identity to calculate 100 digits

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



- 1970s - π computed to million digits using **Taylor series**

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} \dots$$

- Let computers do all of the leg work

Why Study Computing?



Scientific Computing

- **Scientific Computing** is a field where real-world problems are simplified to mathematical models and solved using computational algorithms.
- Must know math and programming
- Must also know the context: physics, biology, statistics, etc.

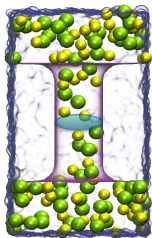
Scientists enable **simulation** which plays a vital role in the development of technology

- Safety
- Optimization
- Analysis

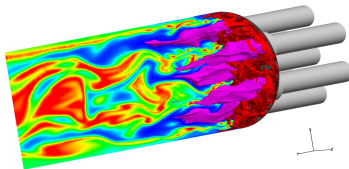


Applications

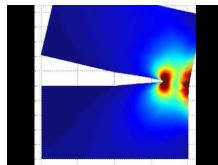
Data produced by computational scientists:



Biomolecular
Systems



Rocket
Simulations



Crack
propagation

Numerical computing is vital in engineering



More Applications

Outside of engineering:

- **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)
- **High performance computing:** design of algorithms for rapid/parallel computing.



Analysis of Algorithms



Numerical Analysis

Numerical analysis is the study of algorithms for the problems of continuous mathematics.

Question

- How do these algorithms work and how do they fail? (math theory)
- To what degree can we trust these algorithms? (error analysis)
- How expensive are these algorithms and will it scale well? (algorithmic efficiency)



The Big Three

For any algorithm we study, we want to know:

Accuracy How accurate is the approximated solution? How reliable are the algorithmic outputs?

Efficiency How fast and cheap (in terms of operations and memory) can we compute a solution with respect to problem size?

Stability Is the computed solution sensitive to changes in the problem parameters? If there's a small offset in the in the conditions, will I get a completely different answer?



Analysis is Important!

Some disasters attributable to bad numerical computing:

- The Patriot Missile failure, in Dharan, 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 a discretization model of the finite element system.



Course Info



All information about the course can be found on the syllabus or Canvas home page

Topics

- All class meetings and Drop-in Hours
- HW Policy
- Dates Dates
- Grading rubric
- Matlab tutorial
- Weekly schedule
- Online access to textbook



Assessment

	Homework	15%
Grades:	Midterms	60%
	Final	25%

Homework:

- Weekly assignment/report
- PDF must include:
 - source code
 - output
 - high-quality graphs
 - answers to any questions
- 15% penalty for each day late
- Lowest score dropped
- Encouraged to work in groups of two



Schedule and Notes:

- Lecture involves slides + hand calculations
- Attendance is mandatory
- Final exam is fixed

- You may ask for accommodations - HW extensions
- Read the syllabus
- Course materials on Canvas
- Look at the MATLAB tutorial on website
⇒ **We will do the tutorial next class**



What to Expect



Typical Process

1. Approach a problem using theory

Example

Find $f'(1.0)$ for function $f(x) = e^x$.
(Assume you do not know the solution.)

Theory

The definition of the derivative

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

Method

Choose a small parameter $h > 0$

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

Typical Process

2. Design an algorithm and implement it

Implementation

Write out your algorithm by hand. Then write your method in Matlab.

```
1  $f(x) = e^x$   
2  $x_0 = 1$   
3  $h = 10^{-2}$   
4  $f'(x_0) \approx (f(x+h) - f(x))/h$ 
```

Listing 1: Algorithm (by hand)

```
1 f = @(x) (exp(x));  
2 x0 = 1;  
3 h = 1e-2;  
4 fp = (f(x+h) - f(x))/h;
```

Listing 2: Implementation (Matlab)

Translate the mathematical algorithm into code.



Typical Process

3. Analyze the error

Error Analysis

For this problem, we know the real answer, $f'(1.0) = e^{1.0} = e$. We can compute the error as a function of h .

- Method

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

- Error

$$E(h) = \left| \underbrace{f'(x)}_{\text{true}} - \underbrace{\frac{f(x+h) - f(x)}{h}}_{\text{approximate}} \right|$$

Be investigative

- How quickly does the error decrease with respect to h ?
- Does the convergence depend on $f(x)$?

Colored Blocks in Lecture Slides

Definition: Terminology

Definitions are in yellow blocks

Method: Name of algorithm

- **Given:** Algorithm inputs
- **Output:** Algorithm output
- **Idea:** What makes the method special

Example

Problem you may see on an exam

Key Idea

Important conceptual idea

Question

Usually a segue question



What to Expect for Exams

Types of Questions:

- **Short calculations**

- ▶ Example: Approximate $\sqrt{2}$ using a Taylor series
- ▶ Focus on setup and method, not long arithmetic

- **Short answers**

- ▶ Example: Explain why a method converges or fails
- ▶ Clear reasoning in words matters

- **Derivations**

- ▶ Reproduce key steps shown in lecture
- ▶ You will not be asked to invent proofs from scratch

- **Graph and table interpretation**

- ▶ Read plots, trends, stability, or behavior from visuals

- **Concept checks**

- ▶ Multiple choice
- ▶ Fill-in-the-blank
- ▶ True/False

Use the review sheets to prepare for exams.



● Schedule Page - One lecture slide per day

Schedule ^{A†}

Week	Reading	Topics	Assignments
1: Jan 19-23	• Homework Guide	• (holiday) • Lecture 1 - Introduction (29) ↓ • Matlab Review ↓	• Homework 1 due Jan 28th
2: Jan 26-30	• Sec 1.1, 1.2, 1.3, 1.4 • Appendix B, C • Review 1 ↓ • Review 2 ↓	• Lecture 2 - Taylor Series & Convergence (28) ↓ • Lecture 3a - Floating Point Numbers (27) ↓ • Lecture 3b - Error with Floats (34) ↓	• Homework 2 due Feb 4th
3: Feb 2-6	• Sec 4.3 • Appendix D • Review 3 ↓	• Lecture 4a - Numerical Differentiation (29) ↓ • Lecture 4b - Extrapolation (24) ↓ • Lecture 5 - Linear Algebra Fundamentals (35) ↓	• Homework 3 due Feb 11th

● Modules - New module available every week

• Week 1	✓ + ...
Lectures	✓ ...
Matlab tutorial.pdf	🔄 ✓ ...
lecture01.pdf	🔄 ✓ ...
Homework	✓ ...
Homework 1 Jan 28 100 pts	✓ ...
hw01.pdf	🔄 ✓ ...
hw01_key.pdf	🔄 ⓧ ...



MATLAB



- **MAT**rix **LAB**oratory
- Both a computing environment and a language
- 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>

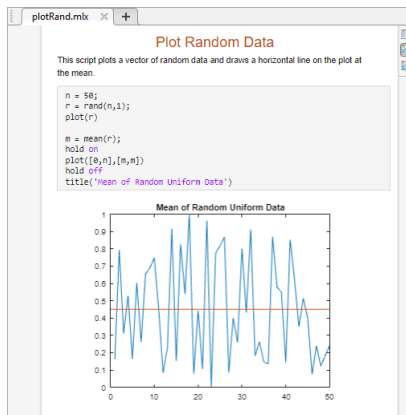


- | | |
|----------|---|
| The Good | <ul style="list-style-type: none">● Beginner friendly● No compiling● Easy debugging → fast development● Designed for linear algebra● Great for plotting● Great for generating reports● Online editor + synced collaboration |
| The Bad | <ul style="list-style-type: none">● Computational overhead● Very limited customization |
| The Ugly | <ul style="list-style-type: none">● Proprietary and expensive (but free for UNM students)● ~~Can't update version???~~ Now you can |

We will use Matlab for homework assignments.



Matlab Live Script



- Combine code, output, and text all in one document
- Friendly UI
- Mathematical equations
- Can include pictures
- Easy to debug



Installation

Matlab is covered by your tuition.
There are versions for Linux, Windows, and Mac.

- 1 Go to <https://www.mathworks.com/>
- 2 Click on Get MATLAB top right
- 3 Create an account with UNM email (or sign in)
- 4 Download and install latest version with your account

Matlab Online

You can also access Matlab on your browser at matlab.mathworks.com



Todo List

- 1 Read the syllabus
- 2 Note the exam dates
- 3 Skim through homework 1
- 4 Read the homework tutorial
- 5 Create a Mathworks account
- 6 (Optional) Find a partner for the homework

