David F. Gleich

Call me …
“Prof Gleich”
“Dr. Gleich”

Please not
“Hey matrix guy!”
What is numerical analysis?
Numerical analysis

The goal of real or mathematical analysis
Rigorously formulate and study everything you’ve learned so far in calculus, etc.

The goal of numerical analysis
Rigorously formulate and study everything you’ve learned so far in calculus, etc. ...
with a numerical approximation to the real numbers and a focus on computer algorithms!
Your textbook

“Numerical analysis is the branch of mathematics that provides tools and methods for solving mathematical problems in numerical form. The objective is to develop detailed computational procedures, capable of being implemented on electronic computers, and to study their performance characteristics”

–Gautschi, page xix
“Numerical analysis is the study of algorithms for the problems of continuous mathematics.”

- Lloyd N. Trefethen, The definition of numerical analysis.

“Numerical analysis is the study of algorithms that use numerical approximation (as opposed to general symbolic manipulations) for the problems of mathematical analysis (as distinguished from discrete mathematics).”

-Wikipedia
Numerical analysis is not ... 

... the study of rounding errors
... the study of computer arithmetic
... the study of Matlab programming
... the study of Taylor expansion
Recurring key elements

- Algorithms
- Problems
- Continuous mathematics
- Mathematical
What will we cover in this course?
Your introduction to mathematics

Arithmetic – (1+1, 3 x 5, \( \frac{3}{4} + \frac{1}{2} \) ...)
Geometry – mathematics of shapes
Algebra – Functions and Trig \( f(x) = x^2+5 \)
Calculus – Derivatives, Integrals
Linear Algebra – Large systems of equations
ODEs – Implicit functions

We’ll revisit this sequence!
Chapter 1

• Properties of computer arithmetic
• Types of errors on a computer

By the end of this chapter, you should understand how to study errors on a computer and where they can arise. (you’ll “relearn” arithmetic!)
Chapter 2

• Computer representations of functions
  – Polynomials
  – Splines
• Best approximation of functions

By the end of this chapter, you will understand how to represent a function in an easy-to-manipulate way on a computer and understand the trade-offs in different representations. (You’ll “relearn” functions from algebra.)
Runge phenomenon in polynomial approximation
Chapter 3

• Numerical algorithms for integration and differentiation (quadrature)
  – Heavily depends on functional representation!

\[ g(x) = f'(x) \quad F = \int_{a}^{b} f(x) \, dx \]

By the end of this chapter, you will understand how to evaluate integrals numerically and be able to evaluate different techniques to evaluate them. (You’ll “relearn” calculus.)
Chapter 4

• Solving systems of equations on a computer find $x$ such that $f(x) = 0$
  – Algorithms, convergence, etc. e.g. Newton’s method

By the end of this chapter, you’ll understand how to use various algorithms to solve systems of nonlinear equations and how this helps us evaluate many fundamental quantities on a computer (back to “algebra” again!)
Chapter 5

• Numerical evaluation of ODEs (initial value problems). Euler, Runge-Kutta, etc.

\[
\frac{dx}{dt} = f(x, t) \quad x(0) = c
\]

By the end of this chapter, you’ll understand how to evolve an ODE on the computer and some trade-offs that arise.
Your introduction to mathematics

Arithmetic – Chapter 1
Geometry – Graphics
Algebra – Chapter 2, 4
Calculus – Chapter 3
Linear Algebra – CS 515
ODEs – Chapter 5
Isn’t symbolic computation good enough?
Floating point

• $1/3 = 0.3333333333333333$
• $2/3 = 0.6666666666666666$
• $+ = 1.0000000000000000$

• $3/10 = 0.2999999999999999$
• $4/10 = 0.40000000000000002$
• $+ = 0.69999999999999996$
Alternatives to floating point

• (From Nick Trefethen) Exact (rational) arithmetic, we want the roots of

\[ p(x) = x^5 - 2x^4 - 3x^3 + 3x^2 - 2x - 1. \]

• No “analytical” formula. (Galois 1820s)
Alternatives to floating point

Using Newton’s Method to solve a polynomial

\[ x^{(0)} = 0 , \]
\[ x^{(1)} = -\frac{1}{2} , \]
\[ x^{(2)} = -\frac{22}{95} , \]
\[ x^{(3)} = -\frac{11414146527}{36151783550} , \]
Alternatives to floating point

Using Newton’s Method to solve a polynomial

\[ x^{(4)} = -\frac{43711566319307638440325676490949986758792998960085536}{138634332790087616118408127558389003321268966090918625}, \]

\[ x^{(5)} = -\frac{7243914791768201761290013818789259730350038836047543931178041194343579260105802744696299}{22974602373157587333399081666432000351477598472080210886006687478324948875098845198224797}
\]

\[ \approx \frac{22882064184585670017703551996316651611596343634562735299921308664663139405767412052875538}{5822898447180467981536221568972260935865495325922571792991768547894449519518216876316931}
\]

\[ \approx \frac{201240642484300698212354536105198706894715223176068754569028985198376505043454529677921}{56837046590814400249541967480411666750181397522783471619066874148005355642107851077541250}
\]

20 iterations produce a 16 terabyte file
Numerical analysis in practice

\[ f(x) \approx x^{-1/2} \]

```c
float Q_rsqrt( float number ) {
    long i; float x2, y; float threehalves = 1.5F;
    x2 = number * 0.5F;
    y = number;
    // evil floating point bit level hacking
    i = * ( long * ) &y;
    i = 0x5f3759df - ( i >> 1 );
    y = * ( float * ) &i;
    // 1st iteration
    y = y * ( threehalves - ( x2 * y * y ) );
    // 2nd iteration, this can be removed
    // y = y * ( threehalves - ( x2 * y * y ) );
    return y;
}
```

Floating point bit-level representation, Newton’s method, no division!
Numerical computing software
MATLAB
Why I picked Julia this term

This is an experiment. We may revert to Matlab.

• Free, free, free! (Run it anywhere.)

• Close enough to Matlab
  – A(5,6) vs. A[5,6]
  – [V,D] = eig(A) vs. V,d = eig(A)
  – “my string” vs. ‘my string’

• For-loops are “more efficient” (used a lot here!)

• Many helpful tools for numerical analysis
  – BigFloat has enhanced precision
What I’m worried about

- Julia is still under active development
- There are many rough edges
- Graphics / plotting are still very rough
How to use Julia

• Online (easiest!)
  – Juliabox.org
  – But there is a critical bug at the moment 😞

• On your own computer
  – Julia command line
  – Jupyter notebook (Julia-Python-R)
  – June (not recommended)
  – Atom/Electron (not recommended)
How to fix Juliabox

1. Create a new text file. Usually called “untitled.txt”
2. Paste in the following line
   
   ```
   VERSION >= v"0.4" && splice!(Base.LOAD_CACHE_PATH, 3)
   ```

3. Close the edit window.
4. Select the file “untitled.txt” (check-box at left)
5. A “rename” button should appear.
6. Click rename and enter “.juliarc”
7. Make sure it’s correct, visit
   
   [https://www.juliabox.org/edit/.juliarc](https://www.juliabox.org/edit/.juliarc) to see your line
Stuff that’s annoying

Quick debugging
• Run Pkg.update()
• Restart Julia
• Run Pkg.update()