

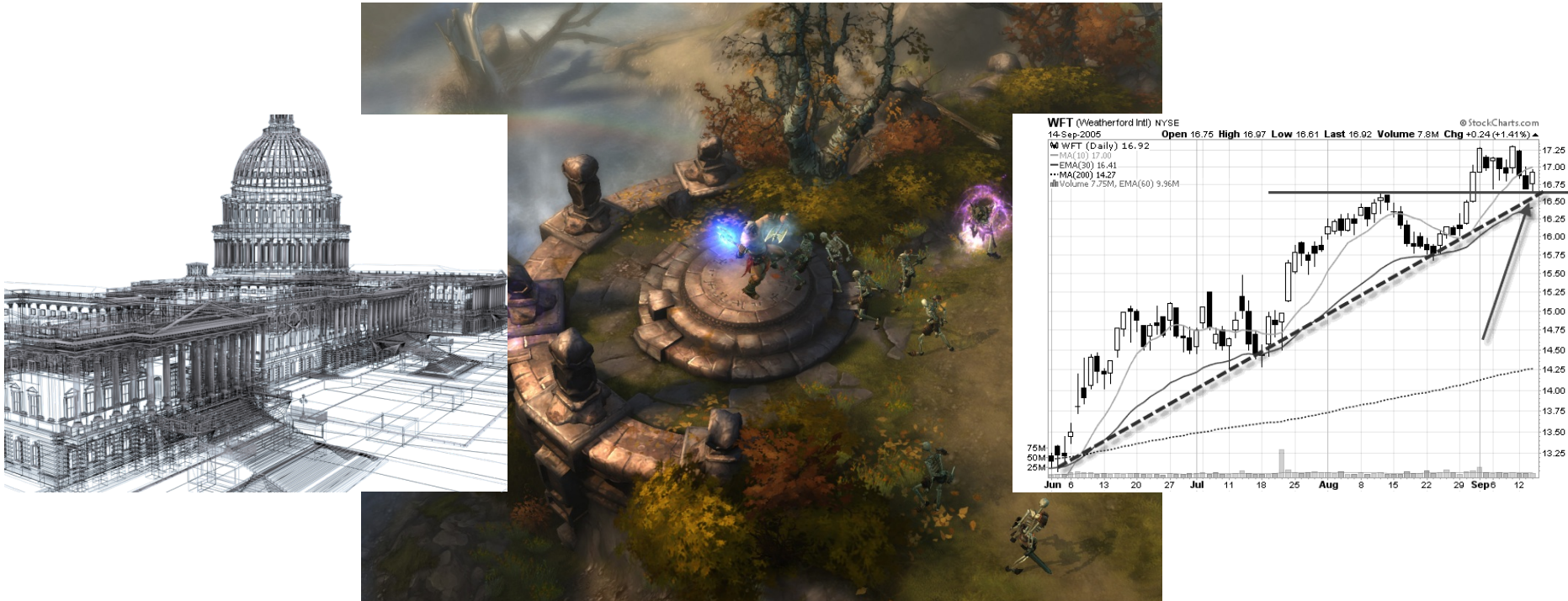
Numerical Analysis

*The purpose of computing is
insight, not numbers.*

-- R.W. Hamming,
Numerical Methods for Scientists and Engineers

Software makes arithmetic seem easy

- Every day, we use software that makes millions of arithmetic calculations
- We trust these calculations
- Ensuring accuracy is extremely difficult



Four kinds of calculation error

1. An inexact mathematical model

- “Projectiles fly in straight lines!”

2. Computing with bad input

- “I am 8.3 feet tall”

3. Round-off errors in the arithmetic

- “My GPA is 4”

4. Approximations of a mathematical system

- “Two men enter, one man leaves”

Four kinds of calculation error

1. An inexact mathematical model

- “Projectiles fly in straight lines!”

2. Computing with bad input

- “I am 8.3 feet tall”

**These aren't
software
problems**

3. Round-off errors in the arithmetic

- “My GPA is 4”

4. Approximations of a mathematical system

- “Two men enter, one man leaves”

Let's define calculation error

Let x be a real number, and let x^* be a number that approximates x .

- x^* approximates x is written as:

$$x^* \approx x$$

- The **absolute error** in the approximation is

$$|x^* - x|$$

- The **relative error** in the approximation is

$$(|x^* - x|) / x$$

- We say that x^* has k **significant decimal digits** if its relative error is less than $5 \cdot 10^{-k-1}$

We all have ideas where error comes from

- **Digital representations** are all in bits.
 - So we can't represent infinitely many numbers
 - Must spread 2^{64} numbers out as usefully as possible
- We basically have two number representations
 - Fixed point representations
 - Uniformly spread over the real numbers: ints
 - Floating point representations
 - Numbers closer to zero get closer together: doubles
 - Floating point numbers **get twice as far apart!**
 - Numbers near $0.1111 \cdot 2^n$ are twice as close together compared to numbers near $0.1111 \cdot 2^{n+1}$

Digital representations are the root of error

$$x = \frac{301}{2000} \approx .15050000 \quad y = \frac{301}{2001} \approx .150424787..$$

- Exact difference:

$$x - y = \frac{301}{4002000} \approx .000075212...$$

- This is what we get with 4 digit approximation

$$x = \frac{301}{2000} \approx .1505 \quad y = \frac{301}{2001} \approx .1504 \quad \rightarrow \quad x - y \approx .0001$$

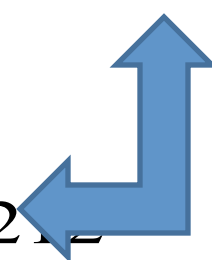
- None of the four digits of the approximation has anything to do with the actual difference!

But perhaps we can re-arrange things ...

$$\begin{aligned}x &= \frac{301}{2000} \approx .15050000 & y &= \frac{301}{2001} \approx .150424787.. \\x - y &= \frac{301 \times 2001 - 301 \times 2000}{4002000} = \frac{602301 - 602000}{4002000} \\&\approx \frac{6.023 \times 10^5 - 6.020 \times 10^5}{4.002 \times 10^6} = \frac{.003 \times 10^5}{4.002 \times 10^6} \approx .00007496\end{aligned}$$

- Exact difference:

$$x - y = \frac{301}{4002000} \approx .00007521$$



We got
2 digits
closer
here

- The result of the floating point calculation depends on the order of operations!
 - Associativity and distributivity are **NOT VALID** in floating point arithmetic!
 - Goodbye safe mathematical universe

Okay, mathematical drama is not dramatic.

- Here's another example

$$p(x) = (x-1)(x-2)(x-3)(x-4)(x-5)(x-6)(x-7)(x-8)(x-9)(x-10)$$

$$q(x) = (x-1)(x-2)(x-3)(x-4)(x-5)(x-6)(x-7)(x-8)(x-9)(x-10) + x^5$$

- If you expand these into 10th order polynomials, they only differ in their 5th term:

$$-902055x^5 \text{ in } p(x) \quad -902054x^5 \text{ in } q(x)$$

Tiny coefficient
change here

- So what are the roots of these equations?

$$- p(x): 1, 2, 3, 4, \dots 10$$

$$- q(x): 1.0000027558, 1.99921, 3.02591, 3.82275, \\ 5.24 \pm .751485i, 7.57 \pm 1.11728i, \dots$$

Crazy change in the
roots here!

Sensitive dependence on initial conditions

- Lots of numerical code will sometimes work – it just might work because it gets the right doubles. With the wrong values, watch out!
 - We must develop code that gives us the kinds of solutions that we expect, even when this kind of crazy error might happen
- Some problems always change dramatically with small differences in input
 - We call these *ill conditioned problems*.
- Numerical analysis helps us to step back from what we type into the computer and remember that the **whole thing is an approximation**.

Major points of numerical error

- Division by tiny numbers

$$\frac{z}{(x - y)} \quad \text{where } x \approx y$$

- Small differences in the denominator lead the huge differences in the outcome, including sign changes.

- Arithmetic on floating point numbers with vastly different magnitudes

$$(2.534 \times 10^{72}) + (3.293 \times 10^{-5})$$

- The floating point number cannot hold 77 zeroes between the first and the second number, so the less significant figures get ignored.

Linear algebra is a huge source of error

- Matrix operations lead to many multiplications and divisions
- We need to review some matrix algebra before we discuss the numerical issues
- A linear system of equations looks like this:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1,$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2,$$

$$\vdots \qquad \qquad \qquad \vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m$$

Matrices are shorthand for systems of eq's

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \vdots \qquad \qquad \qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{array} \quad \longleftrightarrow \quad \left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & b_3 \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_4 \end{array} \right)$$

- Solving systems of equations like this is bread and butter for many pieces of software
 - Signal processing
 - Constraint satisfaction

Gaussian Elimination

- Gaussian elimination is a simple, explicit algorithm for solving systems of linear equations.
- In order to explain Gaussian elimination, we must first describe several operations on linear systems
 1. Add multiples of one equation onto another
 2. Interchange two equations
 3. Multiply an equation by a nonzero constant

Operation 1: Adding equation multiples

$$x + 2y + z = 2$$

$$2x + 6y + z = 7$$

$$x + y + 4z = 3$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 2 & 6 & 1 & 7 \\ 1 & 1 & 4 & 3 \end{array} \right)$$

- Here we will subtract twice the first equation from the second equation, to eliminate the first term from the second equation.
 - This does not change the system of equations

$$x + 2y + z = 2$$

$$2y - z = 3$$

$$x + y + 4z = 3$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & 2 & -1 & 3 \\ 1 & 1 & 4 & 3 \end{array} \right)$$

- The result simplifies the system of equations by eliminating x from the second equation

Operation 2: Interchange two equations

$$x + 2y + z = 2$$

$$2x + 6y + z = 7$$

$$x + y + 4z = 3$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 2 & 6 & 1 & 7 \\ 1 & 1 & 4 & 3 \end{array} \right)$$

- Clearly this operation does not affect the system of linear equations
 - Changing the order does not affect the constraints
 - Here we swap the first two lines

$$2x + 6y + z = 7$$

$$x + 2y + z = 2$$

$$x + y + 4z = 3$$

$$\left(\begin{array}{ccc|c} 2 & 6 & 1 & 7 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 4 & 3 \end{array} \right)$$

Operation #3 Multiply by nonzero constants

$$x + 2y + z = 2$$

$$2x + 6y + z = 7$$

$$x + y + 4z = 3$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 2 & 6 & 1 & 7 \\ 1 & 1 & 4 & 3 \end{array} \right)$$

- Multiplying any equation by a nonzero constant only proportionately scales the constraints
 - Thus the solutions for x , y , and z remain the same
 - Here we will multiply the bottom row by 2

$$x + 2y + z = 2$$

$$2x + 6y + z = 7$$

$$2x + 2y + 8z = 6$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 2 & 6 & 1 & 7 \\ 2 & 2 & 8 & 6 \end{array} \right)$$

Gaussian Elimination

- We want to solve for all variables. A solved system of equations looks like this:

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & b_1 \\ 0 & 1 & 0 & b_2 \\ 0 & 0 & 1 & b_3 \end{array} \right)$$

- This matrix means that x_1 , x_2 , and x_3 are equal to b_1 , b_2 and b_3 .
- We get to this state by applying the matrix operations we just discussed.

The Gaussian Elimination Algorithm

- Begin with row 1. Multiply everything in row 1 by a_{21}/a_{11} . Then subtract row 1 from row 2.
- Repeat this process for all rows other than row 1. This causes all values in column 1 to be equal to zero, except a_{11} .
- Divide all values in row 1 by a_{11}
- Repeat the whole process for a_{22} and continue. The result will be an identity matrix on the left and values on the right, as

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & b_3 \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_4 \end{array} \right)$$

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}' & \cdots & a_{2n}' & b_2' \\ \vdots & \vdots & \ddots & \vdots & b_3' \\ 0 & a_{m2}' & \cdots & a_{mn}' & b_4' \end{array} \right)$$

$$\left(\begin{array}{cccc|c} a_{11} & 0 & \cdots & 0 & b_1' \\ 0 & a_{22}' & \cdots & 0 & b_2' \\ \vdots & \vdots & \ddots & \vdots & b_3' \\ 0 & 0 & \cdots & a_{mn}' & b_4' \end{array} \right)$$

What you will do for the homework

- You will use Gaussian elimination to generate a matrix inverse.
- You will also use Gaussian Elimination with partial pivoting to generate a matrix inverse
- How do you use Gaussian elimination to generate a matrix inverse?

Gaussian Elimination and Inverses

$$\left(\begin{array}{ccc|ccc} 1 & 2 & 1 & 1 & 0 & 0 \\ 2 & 6 & 1 & 0 & 1 & 0 \\ 1 & 1 & 4 & 0 & 0 & 1 \end{array} \right)$$

- In order to generate the matrix inverse, the gaussian elimination approach generates an identity matrix on the left and the inverse matrix on the right
- Apply the same approach as in solving it, except apply it to all six columns

Points of numerical sensitivity

$$\left(\begin{array}{ccc|ccc} 1 & 2 & 1 & 1 & 0 & 0 \\ 2 & 6 & 1 & 0 & 1 & 0 \\ 1 & 1 & 4 & 0 & 0 & 1 \end{array} \right)$$

- Interchanging two equations has no influence on the precision of the floating point numbers
- Adding a multiple of an equation could have precision problems if we have to divide by a very small number to find a multiple
 - That results in multiplying all the numbers in that row by a large and inaccurate number
 - The errors propagate to another row, column, etc.

Another problem with our algorithm

- We cannot simply start with each row/column along the diagonal and multiply/subtract the current position by the target row
 - Some rows will have a zero
- If a target row has a zero, then skip it – we don't have to eliminate the current variable in that equation
- If the originating row has a zero in a diagonal, then we cannot use it (division by zero)
 - Swap it with an unused target row.

How you can verify the inverse matrix

- Matrix multiplication
- Right multiply the original matrix by the inverse matrix. If you get the identity matrix, then you have an inverse.
 - You can use this to evaluate how accurately you calculated the inverse too

Next time:

- We will discuss how to use partial pivoting to increase precision in generating the inverse
- Use the Gaussian Elimination approach to generate simple inverses for your homework assignments (out now)