Lecture 9

Triangular Systems, Naive Gaussian Elimination, Less Naive (maybe)

Owen L. Lewis

Department of Mathematics and Statistics
University of New Mexico

Sept. 17, 2024

Goals:

- Review Diagonal Systems
- Solving Triangular Systems
- Gaussian Elimination Without Pivoting
    - Hand Calculations
    - Cartoon Version
    - Algorithm
    - Cost
- Pivoting (maybe)

Solving Diagonal Systems

The system defined by

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ 6 \\ -15 \end{bmatrix}$$

Solving Diagonal Systems

The system defined by

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ 6 \\ -15 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rrrcr} x_1 & & & = & -1 \\ & 3x_2 & & = & 6 \\ & & 5x_3 & = & -15 \end{array}$$

Solving Diagonal Systems

The system defined by

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ 6 \\ -15 \end{bmatrix}$$

is equivalent to

$$\begin{aligned} x_1 \qquad\qquad\qquad &= -1 \\ 3x_2 \qquad\quad &= 6 \\ 5x_3 &= -15 \end{aligned}$$

The solution is

$$x_1 = -1 \qquad\qquad x_2 = \frac{6}{3} = 2 \qquad\qquad x_3 = \frac{-15}{5} = -3$$

Solving Diagonal Systems

Listing 1: Diagonal System Solution

```
given A, b
for i = 1...n
   x_i = b_i/a_{i,i}
end
```

In Matlab:

```
>> A = ...          % A is a diagonal matrix
>> b = ...
>> x = b./diag(A)
```

This is the *only* place where element-by-element division (`./`) has anything to do with solving linear systems of equations.

Example

Try this in Matlab using `A = diag(rand(5,1));`

Operations?

Try...

Sketch out an operation count to solve a diagonal system of equations...

Operations?

Try...

Sketch out an operation count to solve a diagonal system of equations...

cheap!

one division n times $\longrightarrow \mathcal{O}(n)$ FLOPS

This is the best we can *ever* do. Why?

Triangular Systems

The generic lower and upper triangular matrices are

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & & 0 \\ \vdots & & \ddots & \vdots \\ l_{n1} & & \cdots & l_{nn} \end{bmatrix}$$

and

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & & u_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & u_{nn} \end{bmatrix}$$

The triangular systems

$$Ly = b \qquad Ux = c$$

are easily solved by **forward substitution** and **backward substitution**, respectively

Solving Triangular Systems

$$A = \begin{bmatrix} 4 & 0 & 0 \\ -2 & 3 & 0 \\ 2 & 1 & -2 \end{bmatrix} \qquad b = \begin{bmatrix} 8 \\ -1 \\ 9 \end{bmatrix}$$

Solving Triangular Systems

$$A = \begin{bmatrix} 4 & 0 & 0 \\ -2 & 3 & 0 \\ 2 & 1 & -2 \end{bmatrix} \qquad b = \begin{bmatrix} 8 \\ -1 \\ 9 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} 4x_1 & & & & & = & 8 \\ -2x_1 & + & 3x_2 & & & = & -1 \\ 2x_1 & + & x_2 & + & -2x_3 & = & 9 \end{array}$$

Solving Triangular Systems

$$A = \begin{bmatrix} 4 & 0 & 0 \\ -2 & 3 & 0 \\ 2 & 1 & -2 \end{bmatrix} \qquad b = \begin{bmatrix} 8 \\ -1 \\ 9 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} 4x_1 & & & & & = & 8 \\ -2x_1 & + & 3x_2 & & & = & -1 \\ 2x_1 & + & x_2 & + & -2x_3 & = & 9 \end{array}$$

Solve in forward order (first equation is solved first, etc)

$$x_1 = \frac{8}{4} = 2$$

$$x_2 = \frac{1}{3}\left(-1 + 2x_1\right) = \frac{3}{3} = 1$$

$$x_3 = \frac{1}{-2}\left(9 - x_2 - 2x_1\right) = \frac{4}{-2} = -2$$

Notes

To the board!

Solving Triangular Systems

Solving for $x_1, x_2, \ldots, x_n$ for a lower triangular system is called **forward substitution**.

```
given L (lower △), b
x_1 = b_1/ℓ_11
for i = 2...n
   s = b_i
   for j = 1...i-1
      s = s - ℓ_{i,j} x_j
   end
   x_i = s/ℓ_{i,i}
end
```

Note: We've effectively calculated $\vec{x} = L^{-1}\vec{b}$, but we never calculated L^{-1} or did a mat-vec multiplication.

What about Upper Triangular?

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

What about Upper Triangular?

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} -2x_1 & + & x_2 & + & 2x_3 & = & 9 \\ & & 3x_2 & + & -2x_3 & = & -1 \\ & & & & 4x_3 & = & 8 \end{array}$$

What about Upper Triangular?

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} -2x_1 & + & x_2 & + & 2x_3 & = & 9 \\ & & 3x_2 & + & -2x_3 & = & -1 \\ & & & & 4x_3 & = & 8 \end{array}$$

Solve in backwards order (last equation is solved first, etc)

$$x_3 = \frac{8}{4} = 2 \qquad \begin{aligned} x_2 &= \frac{1}{3}\left(-1 + 2x_3\right) = \frac{3}{3} = 1 \\ x_1 &= \frac{1}{-2}\left(9 - x_2 - 2x_3\right) = \frac{4}{-2} = -2 \end{aligned}$$

Solving Triangular Systems

Solving for $x_1, x_2, \ldots, x_n$ for an upper triangular system is called **backward substitution**.

Listing 2: backward substitution

```
given U (upper △), b
x_n = b_n / u_nn
for i = n−1 ... 1
   s = b_i
   for j = i+1 ... n
      s = s − u_{i,j} x_j
   end
   x_i = s / u_{i,i}
end
```

Solving Triangular Systems

Solving for $x_1, x_2, \ldots, x_n$ for an upper triangular system is called **backward substitution**.

Listing 3: backward substitution

```
given U (upper △), b
x_n = b_n/u_nn
for i = n−1 ... 1
   s = b_i
   for j = i+1 ... n
      s = s − u_{i,j} x_j
   end
   x_i = s/u_{i,i}
end
```

Using forward or backward substitution is sometimes referred to as performing a **triangular solve**.

Operations?

Try...

Sketch out an operation count to solve a triangular system of equations...

Operations?

Try...

Sketch out an operation count to solve a triangular system of equations...

cheap!

- begin in the bottom corner: 1 div
- row 2: 1 mult, 1 add, 1 div, or 3 FLOPS
- row 3: 2 mult, 2 add, 1 div, or 5 FLOPS
- row 4: 3 mult, 3 add, 1 div, or 7 FLOPS
- $\vdots$
- row j: about $2j$ FLOPS

Total FLOPS? $\sum_{j=1}^{n} 2j = 2\frac{n(n+1)}{2}$ or $\mathcal{O}(n^2)$ FLOPS

Gaussian Elimination

- Triangular systems are easy to solve in $\mathcal{O}(n^2)$ FLOPS
- Goal is to transform an arbitrary, square system into an equivalent upper triangular system
- Then easily solve with forward/backward substitution

This process is equivalent to the *formal solution* of $Ax = b$, where A is an $n \times n$ matrix.

$$x = A^{-1}b$$

In MATLAB:

```
>> A = ...
>> b = ...
>> x = A\b
```

Gaussian Elimination — Hand Calculations

Solve

$$\begin{aligned} x_1 + 3x_2 &= 5 \\ 2x_1 + 4x_2 &= 6 \end{aligned}$$

Subtract 2 times the first equation from the second equation

$$\begin{aligned} x_1 + 3x_2 &= 5 \\ -2x_2 &= -4 \end{aligned}$$

This equation is now in triangular form, and can be solved by backward substitution.

Gaussian Elimination — Hand Calculations

The elimination phase transforms the matrix and right hand side to an equivalent system

$$\begin{aligned} x_1 + 3x_2 &= 5 \\ 2x_1 + 4x_2 &= 6 \end{aligned} \qquad \longrightarrow \qquad \begin{aligned} x_1 + 3x_2 &= 5 \\ -2x_2 &= -4 \end{aligned}$$

The two systems have the same solution. The right hand system is upper triangular.
Solve the second equation for x_2

$$x_2 = \frac{-4}{-2} = 2$$

Substitute the newly found value of x_2 into the first equation and solve for x_1.

$$x_1 = 5 - (3)(2) = -1$$

Notes

Gaussian Elimination — Hand Calculations

When performing Gaussian Elimination by hand, we can avoid copying the x_i by using a shorthand notation.
For example, to solve:

$$A = \begin{bmatrix} -3 & 2 & -1 \\ 6 & -6 & 7 \\ 3 & -4 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ -7 \\ -6 \end{bmatrix}$$

Form the *augmented* system

$$\tilde{A} = [A \;\; b] = \left[\begin{array}{ccc|c} -3 & 2 & -1 & -1 \\ 6 & -6 & 7 & -7 \\ 3 & -4 & 4 & -6 \end{array}\right]$$

The vertical bar inside the augmented matrix is just a reminder that the last column is the b vector.

Gaussian Elimination — Hand Calculations

Add 2 times row 1 to row 2, and add (1 times) row 1 to row 3

$$\tilde{A}_{(1)} = \left[\begin{array}{ccc|c} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & -2 & 3 & -7 \end{array}\right]$$

Subtract (1 times) row 2 from row 3

$$\tilde{A}_{(2)} = \left[\begin{array}{ccc|c} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & 0 & -2 & 2 \end{array}\right]$$

Gaussian Elimination — Hand Calculations

The transformed system is now in upper triangular form

$$\tilde{A}_{(2)} = \left[\begin{array}{rrr|r} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & 0 & -2 & 2 \end{array}\right]$$

Solve by back substitution to get

$$x_3 = \frac{2}{-2} = -1$$

$$x_2 = \frac{1}{-2}\left(-9 - 5x_3\right) = 2$$

$$x_1 = \frac{1}{-3}\left(-1 - 2x_2 + x_3\right) = 2$$

Gaussian Elimination — Cartoon Version

Start with the augmented system

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}$$

The x's represent numbers, they are generally *not* the same values.

Begin elimination using the first row as the *pivot row* and the first element of the first row as the pivot element

$$\begin{bmatrix} \boxed{x} & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}$$

Gaussian Elimination — Cartoon Version

- Eliminate elements under the pivot element in the first column.
- x' indicates a value that has been changed once.

$$\begin{bmatrix} \boxed{x} & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \longrightarrow \begin{bmatrix} \boxed{x} & x & x & x & x \\ 0 & x' & x' & x' & x' \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \boxed{x} & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & x' & x' & x' & x' \\ x & x & x & x & x \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \boxed{x} & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & x' & x' & x' & x' \\ 0 & x' & x' & x' & x' \end{bmatrix}$$

Gaussian Elimination — Cartoon Version

- The pivot element is now the diagonal element in the second row.
- Eliminate elements under the pivot element in the second column.
- x'' indicates a value that has been changed twice.

$$\begin{bmatrix} x & x & x & x & x \\ 0 & \boxed{x'} & x' & x' & x' \\ 0 & x' & x' & x' & x' \\ 0 & x' & x' & x' & x' \end{bmatrix} \longrightarrow \begin{bmatrix} x & x & x & x & x \\ 0 & \boxed{x'} & x' & x' & x' \\ 0 & 0 & x'' & x'' & x'' \\ 0 & x' & x' & x' & x' \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} x & x & x & x & x \\ 0 & \boxed{x'} & x' & x' & x' \\ 0 & 0 & x'' & x'' & x'' \\ 0 & 0 & x'' & x'' & x'' \end{bmatrix}$$

Gaussian Elimination — Cartoon Version

- The pivot element is now the diagonal element in the third row.
- Eliminate elements under the pivot element in the third column.
- x''' indicates a value that has been changed three times.

$$\begin{bmatrix} x & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & 0 & \boxed{x''} & x'' & x'' \\ 0 & 0 & x'' & x'' & x'' \end{bmatrix} \longrightarrow \begin{bmatrix} x & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & 0 & \boxed{x''} & x'' & x'' \\ 0 & 0 & 0 & x''' & x''' \end{bmatrix}$$

Gaussian Elimination — Cartoon Version

- The pivot element is now the diagonal element in the third row.
- Eliminate elements under the pivot element in the third column.
- x''' indicates a value that has been changed three times.

$$\begin{bmatrix} x & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & 0 & \boxed{x''} & x'' & x'' \\ 0 & 0 & x'' & x'' & x'' \end{bmatrix} \longrightarrow \begin{bmatrix} x & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & 0 & \boxed{x''} & x'' & x'' \\ 0 & 0 & 0 & x''' & x''' \end{bmatrix}$$

Notes

Gaussian Elimination — Cartoon Version

Summary

- Gaussian Elimination is an orderly process for transforming an augmented matrix into an equivalent upper triangular form.
- The elimination operation at the k^{th} step is

$$\tilde{a}_{ij} = \tilde{a}_{ij} - (\tilde{a}_{ik}/\tilde{a}_{kk})\tilde{a}_{kj}, \quad i > k, \quad j \geqslant k$$

- Elimination requires three nested loops.
- The result of the elimination phase is represented by the image below.

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \longrightarrow \begin{bmatrix} x & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & 0 & x'' & x'' & x'' \\ 0 & 0 & 0 & x''' & x''' \end{bmatrix}$$

Gaussian Elimination

Summary

- Transform a linear system into (upper) triangular form. i.e. transform lower triangular part to zero
- Transformation is done by taking linear combinations of rows
  - This means at every step we are actually multiplying (on the left) by *some* matrix!

Gaussian Elimination Algorithm

Listing 4: Forward Elimination beta

```
given A, b

for k = 1 ... n-1
   for i = k+1 ... n
      for j = k ... n
         a_ij = a_ij - (a_ik / a_kk) a_kj
      end
      b_i = b_i - (a_ik / a_kk) b_k
   end
end
```

- the multiplier can be moved outside the j-loop
- no reason to actually compute 0

Challenge: The loops over i and j may be exchanged—why would one be preferable?

Gaussian Elimination Algorithm

Listing 5: Forward Elimination

```
given A, b

for k = 1...n-1
   for i = k+1...n
      xmult = a_ik / a_kk
      a_ik = 0
      for j = k+1...n
         a_ij = a_ij - (xmult) a_kj
      end
      b_i = b_i - (xmult) b_k
   end
end
```

Gaussian Elimination Algorithm: Storing Multipliers

Listing 6: Forward Elimination

```
given A, b

for k = 1 ... n-1
   for i = k+1 ... n
      xmult = a_ik / a_kk
      a_ik = xmult
      for j = k+1 ... n
         a_ij = a_ij - (xmult) a_kj
      end
      b_i = b_i - (xmult) b_k
   end
end
```

We are storing the multipliers in the below diagonal entries (just being efficient).
Those entries will never be accessed during back-substitution!

Naive Gaussian Elimination Algorithm

- Forward Elimination
- + Backward substitution
- = Naive Gaussian Elimination

Example

GE_naive.m GE_naive_test.m

Forward Elimination Cost?

What is the cost in converting from A to U?

Step k	Add	Multiply	Divide
1	$(n-1)^2$	$(n-1)^2$	$n-1$
2	$(n-2)^2$	$(n-2)^2$	$n-2$
$\vdots$			
n-1	1	1	1

or

add	$\sum_{j=1}^{n-1} j^2$
multiply	$\sum_{j=1}^{n-1} j^2$
divide	$\sum_{j=1}^{n-1} j$

Forward Elimination Cost?

add	$\sum_{j=1}^{n-1} j^2$
multiply	$\sum_{j=1}^{n-1} j^2$
divide	$\sum_{j=1}^{n-1} j$

We know $\sum_{j=1}^{p} j = \frac{p(p+1)}{2}$ and $\sum_{j=1}^{p} j^2 = \frac{p(p+1)(2p+1)}{6}$, so

add-subtracts	$\frac{n(n-1)(2n-1)}{6}$
multiply-divides	$\frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)}{2} = \frac{n(n^2-1)}{3}$

Forward Elimination Cost?

add-subtracts	$\frac{n(n-1)(2n-1)}{6}$
multiply-divides	$\frac{n(n^2-1)}{3}$
add-subtract for *b*	$\frac{n(n-1)}{2}$
multiply-divides for *b*	$\frac{n(n-1)}{2}$

Back Substitution Cost

As before

add-subtract	$\frac{n(n-1)}{2}$
multiply-divides	$\frac{n(n+1)}{2}$

Naive Gaussian Elimination Cost

Combining the cost of forward elimination, updating b, and backward substitution gives

add-subtracts	$\frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)}{2} + \frac{n(n-1)}{2} = \frac{n(n-1)(2n+5)}{3}$
multiply-divides	$\frac{n(n^2-1)}{3} + \frac{n(n-1)}{2} + \frac{n(n+1)}{2} = \frac{n(n^2+3n-1)}{3}$

So the total cost of add-subtract-multiply-divide is about

$$\frac{2}{3}n^3$$

$\Rightarrow$ double n results in a cost increase of a factor of 8

Why is this "naive"?

Example

$$A = \begin{bmatrix} 0 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Example

$$A = \begin{bmatrix} 1e-10 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$