Lecture 6

Gaussian Elimination

T. Gambill

Department of Computer Science
University of Illinois at Urbana-Champaign

February ???, 2010

Gaussian Elimination

- Solving Triangular Systems
- Gaussian Elimination Without Pivoting
  - Hand Calculations
  - Cartoon Version
  - Algorithm
- Elementary Elimination Matrices And LU Factorization

Gaussian Elimination

Gaussian elimination is a mostly general method for solving square systems.

We will work with systems in their matrix form, such as

$$\begin{aligned} 4x_1 + 8x_2 + 12x_3 &= 4 \\ 2x_1 + 12x_2 + 16x_3 &= 6 \\ x_1 + 3x_2 + 6.25x_3 &= 1, \end{aligned}$$

in its equivalent matrix form,

$$\begin{bmatrix} 4 & 8 & 12 \\ 2 & 12 & 16 \\ 1 & 3 & 6.25 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 1 \end{bmatrix}$$

which can be compactly expresed in the form,

$$A * \mathbf{x} = \mathbf{b}.$$

Triangular Systems

If we can factor $A = L * U$ where,

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

and

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

Then solving $A * \mathbf{x} = \mathbf{b}$ involves solving the triangular systems

$$Ly = b \qquad Ux = y$$

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

Solving Triangular Systems

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

Listing 1: backward substitution

```
given A (upper △), b
x_n = b_n/a_nn
for i = n-1...1
   s = b_i
   for j = i+1...n
      s = s - a_{i,j} x_j
   end
   x_i = s/a_{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 2: backward substitution

```
given A (upper △), b
x_n = b_n / a_nn
for i = n-1...1
   s = b_i
   for j = i+1...n
      s = s - a_{i,j} x_j
   end
   x_i = s / a_{i,i}
end
```

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

Operations?

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 -k: about $2k - 1$ FLOPS

Total FLOPS? $\sum_{k=1}^{n} 2k - 1 = 2\frac{n(n+1)}{2} - n$ 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 backward substitution

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

```
>>> A = np.array([[4.,8.,12.],[2.,12.,16.],[1.,3.,6.25]])
>>> b = np.array([[4.],[6.],[1.]])
>>> LU = scipy.linalg.lu_factor(A)
>>> LU
(array([[  4.    ,   8.    ,  12.    ],
       [  0.5   ,   8.    ,  10.    ],
       [  0.25  ,   0.125,   2.    ]]), array([0, 1, 2], dtype=int32))
>>> x = scipy.linalg.lu_solve(LU,b)
>>> x
array([[ 0.125 ],
       [ 0.8125],
       [-0.25  ]])
```

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.

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

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

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

Solve by back substitution to get

$$x_3 = \frac{2}{-2} = -1$$
$$x_2 = \frac{1}{-2}\left(-9 - 5x_3\right) = 2$$
$$x_1 = \frac{1}{-3}\left(-1 - 2x_2 + x_3\right) = 2$$

Challenge: What would we do if we changed the value of the $\mathbf{b}$ vector? How would we solve the new system of equations?

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

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
- Example: $a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$
- If $a_1 \neq 0$, then

$$\begin{bmatrix} 1 & 0 \\ -a_2/a_1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} a_1 \\ 0 \end{bmatrix}$$

Gaussian Elimination Algorithm

Listing 3: 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 4: 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
```

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	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	$\frac{n(n-1)}{2}$
multipply-divides	$\frac{n(n+1)}{2}$

Naive Gaussian Elimination Cost

Combining the cost of forward elimination and backward substitution gives

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

LU factorization

Remember that vector-matrix multiplication $\mathbf{y}^T * B$ can be viewed as forming a linear combination of the rows in the matrix B. For example,

$$\mathbf{y}^T * B = \begin{bmatrix} -1 & -2 & -3 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$= (-1) * \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} + (-2) * \begin{bmatrix} 4 & 5 & 6 \end{bmatrix} + (-3) * \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}$$

Therefore matrix-matrix multiplication $C = A * B$ can be viewed as forming the product of the rows of A with B. That is, if $C = A * B$ and $\mathbf{a_i}$,$\mathbf{c_i}$ denote the rows of A and C respectively then we have,

$$\mathbf{c_i} = \mathbf{a_i} * B$$

Example matrix-matrix multiplication

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

$$= \begin{bmatrix} (-1) * \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} + (-2) * \begin{bmatrix} 4 & 5 & 6 \end{bmatrix} + (-3) * \begin{bmatrix} 7 & 8 & 9 \end{bmatrix} \\ (-4) * \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} + (-5) * \begin{bmatrix} 4 & 5 & 6 \end{bmatrix} + (-6) * \begin{bmatrix} 7 & 8 & 9 \end{bmatrix} \\ (-7) * \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} + (-8) * \begin{bmatrix} 4 & 5 & 6 \end{bmatrix} + (-9) * \begin{bmatrix} 7 & 8 & 9 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} -30 & -36 & -42 \\ -66 & -81 & -96 \\ -102 & -126 & -150 \end{bmatrix}$$

How is this related to Gaussian Elimination?

For example, to solve:

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

using LU factorization we will want to first perform a ”Gaussian elimination” on the first column of A (NOT the augmented matrix). This can be performed by using matrix multiplication, for example,

$$M_1 * A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -3 & 2 & -1 \\ 6 & -6 & 7 \\ 3 & -4 & 4 \end{bmatrix} = \begin{bmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & -2 & 3 \end{bmatrix}$$

and next on the second column,

$$M_2 * (M_1 * A) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} * \begin{bmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & -2 & 3 \end{bmatrix} = \begin{bmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{bmatrix}$$

Elimination Matrices

- Another way to zero out entries in a column of A
- Annihilate entries below k^{th} element in $\mathbf{a}$ with matrix, M_k:

$$M_k\mathbf{a} = \begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & -m_{k+1} & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -m_n & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ a_{k+1} \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

  where $m_i = a_i/a_k$, $i = k+1, \dots, n$.
- The divisor a_k is the "pivot" (and needs to be nonzero)

Elimination Matrices

- Matrix M_k is an "elementary elimination matrix"
  - Adds a multiple of row k to each subsequent row, with "multipliers" m_i
  - Result is zeros in the k^{th} column for rows $i > k$.
- M_k is unit lower triangular and nonsingular
- $M_k = I - \mathbf{m_k}\mathbf{e_k}^T$ where $\mathbf{m_k} = [0, \dots, 0, m_{k+1}, \dots, m_n]^T$ and $\mathbf{e_k}$ is the k^{th} column of the identity matrix I.
- $M_k^{-1} = I + \mathbf{m_k}\mathbf{e_k}^T$, which means M_k^{-1} is also lower triangular, and we will denote $M_k^{-1} = L_k$.

Can you prove $M_k^{-1} = I + \mathbf{m_k}\mathbf{e_k}^T$?

Elimination Matrices

- Suppose M_j and M_k are elementary elimination matrices with $j > k$, then

$$\begin{aligned} M_k M_j &= I - \mathbf{m_k} \mathbf{e_k}^T - \mathbf{m_j} \mathbf{e_j}^T + \mathbf{m_k} \mathbf{e_k}^T \mathbf{m_j} \mathbf{e_j}^T \\ &= I - \mathbf{m_k} \mathbf{e_k}^T - \mathbf{m_j} \mathbf{e_j}^T + \mathbf{m_k} (\mathbf{e_k}^T \mathbf{m_j}) \mathbf{e_j}^T \\ &= I - \mathbf{m_k} \mathbf{e_k}^T - \mathbf{m_j} \mathbf{e_j}^T \end{aligned}$$

  because the k^{th} entry of vector $\mathbf{m_j}$ is zero (since $j > k$)
- Thus $M_k M_j$ is essentially a union of their columns.
- Note this is also true for $M_k^{-1} M_j^{-1}$.

Example continued...

We showed that for the matrix A,

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

$$M_2 * (M_1 * A) = \begin{bmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{bmatrix} = U$$

and since,

$$L_1 = M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, L_2 = M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

we can write,

$$A = L_1 L_2 U = LU = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{bmatrix}$$

Gaussian Elimination

- To reduce $Ax = b$ to upper triangular form, first construct M_1 with a_{11} as the pivot (eliminating the first column of A below the diagonal.)
- Then $M_1 A x = M_1 b$ still has the same solution.
- Next construct M_2 with pivot a_{22} to eliminate the second column below the diagonal.
- Then $M_2 M_1 A x = M_2 M_1 b$ still has the same solution
- $M_{n-1} \ldots M_1 A x = M_{n-1} \ldots M_1 b$
- Let $M = M_{n-1} \ldots M_1$. Then $MAx = Mb$, with MA upper triangular.
- Do back substitution on $MAx = Mb$.

Another Way to Look at A

We've mentioned L and U today. Why?
Consider this

$$
\begin{aligned}
A &= A \\
A &= (M^{-1}M)A \\
A &= (M_1^{-1}M_2^{-1}\ldots M_{n-1}^{-1})(M_{n-1}\ldots M_1)A \\
A &= (M_1^{-1}M_2^{-1}\ldots M_{n-1}^{-1})((M_{n-1}\ldots M_1)A) \\
A &= \qquad\qquad L \qquad\qquad\qquad\qquad U
\end{aligned}
$$

But MA is upper triangular, and we've seen that $M_1^{-1}\ldots M_{n-1}^{-1}$ is lower triangular. Thus, we have an algorithm that factors A into two matrices L and U.

Why is this "naive"?

Example

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

Example

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

Matrix Inverse — Hand Calculations

One way to obtain the inverse of a matrix A is to augment the original matrix with the identity matrix $[A|I]$and then perform Gaussian Elimination until the matrix A becomes the identity matrix. We then have $[I|A^{-1}]$. Consider the following example:
Find the inverse of the following matrix.

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

We augment the matrix A with the $3x3$ identity matrix.

$$\tilde{A} = [A \mid I] = \left[\begin{array}{ccc|ccc} -3 & 2 & -1 & 1 & 0 & 0 \\ 6 & -6 & 7 & 0 & 1 & 0 \\ 3 & -4 & 4 & 0 & 0 & 1 \end{array}\right]$$

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

Matrix Inverse — Hand Calculations

Perform Gaussian elimination on the first column to get...

$$\tilde{A} = [A \mid I] = \left[\begin{array}{ccc|ccc} -3 & 2 & -1 & 1 & 0 & 0 \\ 0 & -2 & 5 & 2 & 1 & 0 \\ 0 & -2 & 3 & 1 & 0 & 1 \end{array}\right]$$

and next on the second column...

$$\tilde{A} = [A \mid I] = \left[\begin{array}{ccc|ccc} -3 & 2 & -1 & 1 & 0 & 0 \\ 0 & -2 & 5 & 2 & 1 & 0 \\ 0 & 0 & -2 & -1 & -1 & 1 \end{array}\right]$$

Now, zero out the values off the main diagonal starting with the second column.

$$\tilde{A} = [A \mid I] = \left[\begin{array}{ccc|ccc} -3 & 0 & 4 & 3 & 1 & 0 \\ 0 & -2 & 5 & 2 & 1 & 0 \\ 0 & 0 & -2 & -1 & -1 & 1 \end{array}\right]$$

Matrix Inverse — Hand Calculations

Continue to zero out the values off the main diagonal with the third column.

$$\tilde{A} = [A \mid I] = \left[\begin{array}{ccc|ccc} -3 & 0 & 0 & 1 & -1 & 2 \\ 0 & -2 & 0 & -1/2 & -3/2 & 5/2 \\ 0 & 0 & -2 & -1 & -1 & 1 \end{array}\right]$$

Finally, multiply row one by -1/3, row two by -1/2 and row three by -1/2 to get the following.

$$\tilde{A} = [A \mid I] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -1/3 & 1/3 & -2/3 \\ 0 & 1 & 0 & 1/4 & 3/4 & -5/4 \\ 0 & 0 & 1 & 1/2 & 1/2 & -1/2 \end{array}\right]$$

and the inverse is,

$$I = \begin{bmatrix} -1/3 & 1/3 & -2/3 \\ 1/4 & 3/4 & -5/4 \\ 1/2 & 1/2 & -1/2 \end{bmatrix}$$

Matrix Inverse Algorithm

Listing 5: Matrix Inversion

```
  given A
n = length(a); b = eye(n);
for k=1:n % k-th pivot
    for i = [(k+1): n, (k-1):-1:1] %i-th row, all except k-th
      xmult = a(i,k)./a(k,k);
      for j = k:n % j-th column
       a(i,j) = a(i,j) - xmult.*a(k,j);
      end
      for j = 1:n %j-th column
        b(i,j) = b(i,j) - xmult.*b(k,j);
      end
    end
end
for k=1:n % k-th diagonal value
      for j =1:n
       b(k,j)= b(k,j)./a(k,k);
      end
end
```