solving systems

L. Olson

Department of Computer Science
University of Illinois at Urbana-Champaign
objectives

- Construct a linear system for a problem
- Solve a linear system
- Analyze the cost (and accuracy?) of a solve
- Develop an algorithm for solving systems
• Solving Triangular Systems
• Gaussian Elimination Without Pivoting
  • Hand Calculations
  • Cartoon Version
  • Algorithm
• Elementary Elimination Matrices And LU Factorization
Gaussian elimination is a mostly general method for solving square systems.

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

\[
\begin{align*}
    x_1 + 3x_2 + 5x_3 &= 4 \\
    9x_1 + 7x_2 + 8x_3 &= 6 \\
    3x_1 + 2x_2 + 7x_3 &= 1,
\end{align*}
\]

in its equivalent matrix form,

\[
\begin{bmatrix}
    1 & 3 & 5 \\
    9 & 7 & 8 \\
    3 & 2 & 7
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    x_3
\end{bmatrix}
=
\begin{bmatrix}
    4 \\
    6 \\
    1
\end{bmatrix}.
\]
The generic lower and upper triangular matrices are

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

and

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

The triangular systems

\[
Ly = b \quad Ux = c
\]

are easily solved by **forward substitution** and **backward substitution**, respectively.
Solving for $x_1, x_2, \ldots, x_n$ for an upper triangular system is called **backward substitution**.

Listing 1: backward substitution (page 270)

```plaintext
1 given A (upper △), b
2 $x_n = b_n/a_{nn}$
3 for $i = n - 1 \ldots 1$
4     $s = b_i$
5     for $j = i + 1 \ldots n$
6         $s = s - a_{i,j}x_j$
7     end
8 $x_i = s/a_{i,i}$
9 end
```
Solving for $x_1, x_2, \ldots, x_n$ for an upper triangular system is called **backward substitution**.

**Listing 2: backward substitution (page 270)**

```plaintext
given $A$ (upper $\triangle$), $b$

1. $x_n = b_n/a_{nn}$
2. **for** $i = n - 1 \ldots 1$
   3. $s = b_i$
   4. **for** $j = i + 1 \ldots n$
      5. $s = s - a_{i,j}x_j$
   6. **end**
7. $x_i = s/a_{i,i}$
8. **end**
```

Using forward or backward substitution is sometimes referred to as performing a **triangular solve**.
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
- ...
- row -j: about $2j - 1$ FLOPS

Total FLOPS? $\sum_{j=1}^{n} 2j - 1 = 2 \frac{n(n+1)}{2} - n$ or $\Theta(n^2)$ FLOPS
Triangular systems are easy to solve in $O(n^2)$ FLOPS. Goal is to transform an arbitrary, square system into an equivalent upper triangular system. Then easily solve with backward substitution.

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

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

\[ x_1 + 3x_2 = 5 \]
\[ 2x_1 + 4x_2 = 6 \]

Subtract 2 times the first equation from the second equation

\[ x_1 + 3x_2 = 5 \]
\[ -2x_2 = -4 \]

This equation is now in triangular form, and can be solved by backward substitution.
The elimination phase transforms the matrix and right hand side to an equivalent system

\[
\begin{align*}
x_1 + 3x_2 &= 5 \\
2x_1 + 4x_2 &= 6
\end{align*}
\quad \rightarrow \quad
\begin{align*}
x_1 + 3x_2 &= 5 \\
-2x_2 &= -4
\end{align*}
\]

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 \(x_1\).

\[
x_1 = 5 - (3)(2) = -1
\]
When performing Gaussian Elimination by hand, we can avoid copying the $x_i$ by using a shorthand notation.

For example, to solve:

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

Form the **augmented** system

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

The vertical bar inside the augmented matrix is just a reminder that the last column is the $b$ vector.
Add 2 times row 1 to row 2, and add (1 times) row 1 to row 3

\[ \tilde{A}_{(1)} = \begin{bmatrix} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & -2 & 3 & -7 \end{bmatrix} \]

Subtract (1 times) row 2 from row 3

\[ \tilde{A}_{(2)} = \begin{bmatrix} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & 0 & -2 & 2 \end{bmatrix} \]
The transformed system is now in upper triangular form

\[ \tilde{A}_{(2)} = \begin{bmatrix} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & 0 & -2 & 2 \end{bmatrix} \]

Solve by back substitution to get

\[ x_3 = \frac{2}{-2} = -1 \]

\[ x_2 = \frac{1}{-2} (-9 - 5x_3) = 2 \]

\[ x_1 = \frac{1}{-3} (-1 - 2x_2 + x_3) = 2 \]
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

\[
\begin{bmatrix}
  x & x & x & x & x \\
  x & x & x & x & x \\
  x & x & x & x & x \\
  x & x & x & x & x \\
\end{bmatrix}
\]
gaussian elimination — cartoon version

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

\[
\begin{bmatrix}
X & x & x & x & x \\
x & x & x & x & x \\
x & x & x & x & x \\
x & x & x & x & x \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
X & x & x & x & x \\
0 & x' & x' & x' & x' \\
x & x & x & x & x \\
x & x & x & x & x \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
X & x & x & x & x \\
0 & x' & x' & x' & x' \\
0 & x' & x' & x' & x' \\
x & x & x & x & x \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
X & x & x & x & x \\
0 & x' & x' & x' & x' \\
0 & x' & x' & x' & x' \\
0 & x' & x' & x' & x' \\
\end{bmatrix}
\]
• 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.
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.
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
  \[
  \tilde{a}_{ij} = \tilde{a}_{ij} - \left( \frac{\tilde{a}_{ik}}{\tilde{a}_{kk}} \right) \tilde{a}_{kj}, \quad i > k, \quad j \geq k
  \]
- Elimination requires three nested loops.
- The result of the elimination phase is represented by the image below.
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 $k = 1 \ldots n - 1$
  for $i = k + 1 \ldots n$
    for $j = k \ldots n$
      $a_{ij} = a_{ij} - (a_{ik} / a_{kk}) a_{kj}$
    end
  
  $b_i = b_i - (a_{ik} / a_{kk}) b_k$
end

either

- 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
Listing 4: Forward Elimination

given $A$, $b$

for $k = 1 \ldots n - 1$

for $i = k + 1 \ldots n$

\[
x_{\text{mult}} = \frac{a_{ik}}{a_{kk}}
\]

$a_{ik} = 0$

for $j = k + 1 \ldots n$

\[
a_{ij} = a_{ij} - (x_{\text{mult}})a_{kj}
\]

end

\[
b_i = b_i - (x_{\text{mult}})b_k
\]

end

end
naive gaussian elimination algorithm

- Forward Elimination
- + Backward substitution
- = Naive Gaussian Elimination
What is the cost in converting from $A$ to $U$?

<table>
<thead>
<tr>
<th>Step</th>
<th>Add</th>
<th>Multiply</th>
<th>Divide</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(n - 1)^2$</td>
<td>$(n - 1)^2$</td>
<td>$n - 1$</td>
</tr>
<tr>
<td>2</td>
<td>$(n - 2)^2$</td>
<td>$(n - 2)^2$</td>
<td>$n - 2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

or

\[
\begin{align*}
\text{add} & \quad \sum_{j=1}^{n-1} j^2 \\
\text{multiply} & \quad \sum_{j=1}^{n-1} j^2 \\
\text{divide} & \quad \sum_{j=1}^{n-1} j
\end{align*}
\]
forward elimination cost?

<table>
<thead>
<tr>
<th>Operation</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>( \sum_{j=1}^{n-1} j^2 )</td>
</tr>
<tr>
<td>Multiply</td>
<td>( \sum_{j=1}^{n-1} j^2 )</td>
</tr>
<tr>
<td>Divide</td>
<td>( \sum_{j=1}^{n-1} j )</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add-subtracts</td>
<td>( \frac{n(n-1)(2n-1)}{6} )</td>
</tr>
<tr>
<td>Multiply-divides</td>
<td>( \frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)}{2} = \frac{n(n^2-1)}{3} )</td>
</tr>
</tbody>
</table>
forward elimination cost?

<table>
<thead>
<tr>
<th>Add-Subtracts</th>
<th>$\frac{n(n-1)(2n-1)}{6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiply-Divides</td>
<td>$\frac{n(n^2-1)}{3}$</td>
</tr>
<tr>
<td>Add-Subtract for $b$</td>
<td>$\frac{n(n-1)}{2}$</td>
</tr>
<tr>
<td>Multiply-Divides for $b$</td>
<td>$\frac{n(n-1)}{2}$</td>
</tr>
</tbody>
</table>
back substitution cost

As before

\[
\begin{align*}
\text{add-subtract} & \quad \frac{n(n-1)}{2} \\
\text{multiply-divides} & \quad \frac{n(n+1)}{2}
\end{align*}
\]
naive gaussian elimination cost

Combining the cost of forward elimination and backward substitution gives

\[
\begin{align*}
\text{add-subtracts} & \quad \frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)}{2} + \frac{n(n-1)}{2} \\
& = \frac{n(n-1)(2n+5)}{3}
\end{align*}
\]

\[
\begin{align*}
\text{multiply-divides} & \quad \frac{n(n^2-1)}{3} + \frac{n(n-1)}{2} + \frac{n(n+1)}{2} \\
& = \frac{n(n^2+3n-1)}{3}
\end{align*}
\]

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

\[
\frac{2}{3}n^3
\]

⇒ double \( n \) results in a cost increase of a factor of 8
Another way to zero out entries in a column of $A$

- Annihilate entries below $k^{th}$ element in $a$ with matrix, $M_k$:

\[
M_k a = \begin{bmatrix}
1 & \ldots & 0 & 0 & \ldots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \ldots & 1 & 0 & \ldots & 0 \\
0 & \ldots & -m_{k+1} & 1 & \ldots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \ldots & -m_n & 0 & \ldots & 1
\end{bmatrix}
\begin{bmatrix}
a_1 \\
\vdots \\
a_k \\
a_{k+1} \\
\vdots \\
a_n
\end{bmatrix}
= \begin{bmatrix}
a_1 \\
\vdots \\
a_k \\
0
\end{bmatrix}
\]

where $m_i = a_i / a_k$, $i = k + 1, \ldots, n$.

- 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
- $M_k = I - m_k e_k^T$ where $m_k = [0, \ldots, 0, m_{k+1}, \ldots, m_n]^T$ and $e_k$ is the $k^{th}$ column of the identity matrix $I$.
- $M_k^{-1} = I + m_k e_k^T$, which means $M_k^{-1}$ is also lower triangular, and we will denote $M_k^{-1} = L_k$.

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

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

$$M_k M_j = I - m_k e_k^T - m_j e_j^T + m_k e_k^T m_j e_j^T$$

$$= I - m_k e_k^T - m_j e_j^T + m_k (e_k^T m_j) e_j^T$$

$$= I - m_k e_k^T - m_j e_j^T$$

because the $k^{th}$ entry of vector $m_j$ is zero (since $j > k$)

- Thus $M_k M_j$ is essentially a union of their columns.

- Note this is also true for $M_k^{-1} M_j^{-1}$. 
Let \( a = \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} \).

\[
M_1 a = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}
\]

and

\[
M_2 a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix}
\]
So

\[
L_1 = M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad L_2 = M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1/2 & 1 \end{bmatrix}
\]

which means

\[
M_1 M_2 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 1/2 & 1 \end{bmatrix}, \quad L_1 L_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -1/2 & 1 \end{bmatrix}
\]
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 $M_1Ax = M_1b$ still has the same solution.
- Next construct $M_2$ with pivot $a_{22}$ to eliminate the second column below the diagonal.
- Then $M_2M_1Ax = M_2M_1b$ still has the same solution
- $M_{n-1} \ldots M_1Ax = M_{n-1} \ldots M_1b$
- Let $M = M_nM_{n-1} \ldots M_1$. Then $MAx = Mb$, with $MA$ upper triangular.
- Do back substitution on $MAx = Mb$. 

We’ve mentioned $L$ and $U$ today. Why?

Consider this

$$
A = A \\
A = (M^{-1}M)A \\
A = (M_1^{-1}M_2^{-1} \ldots M_n^{-1})(M_nM_{n-1} \ldots M_1)A \\
A = (M_1^{-1}M_2^{-1} \ldots M_n^{-1})((M_nM_{n-1} \ldots M_1)A) \\
A = L U
$$

But $MA$ is upper triangular, and we’ve seen that $M_1^{-1} \ldots M_n^{-1}$ 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

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