Solving Linear System of Equations
Matrix Computations — CPSC 5006 E

Julien Dompierre

Department of Mathematics and Computer Science
Laurentian University

Sudbury, October 23, 2010

Outline

- Read Sections: 3.1 – 3.2
- Background on linear systems
- Gaussian elimination and the Gauss-Jordan algorithms
- The LU factorization
- Gaussian Elimination with pivoting

Background: Linear Systems

The problem: Suppose A is an $n \times n$ matrix, and b a vector of $\mathbb{R}^n$. Find x such that

$$Ax = b.$$

x is called the **unknown** vector, b the **right-hand side**, and A the **coefficient matrix**.

Example of a Linear System

$$\left\{\begin{array}{rcrcrcl} 2x_1 & + & 4x_2 & + & 4x_3 & = & 6 \\ x_1 & + & 5x_2 & + & 6x_3 & = & 4 \\ x_1 & + & 3x_2 & + & x_3 & = & 8 \end{array}\right. \quad \text{or} \quad \begin{bmatrix} 2 & 4 & 4 \\ 1 & 5 & 6 \\ 1 & 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 8 \end{bmatrix}$$

(Solution of above system ?)

The standard mathematical solution is given by the Cramer's rule:

$$x_i = \det(A_i)/\det(A)$$

where A_i is the matrix obtained by replacing the i-th column by b.

Note: This formula is useless in practice beyond $n = 3$ or $n = 4$.

The Three Cases

The are three cases for the solution of a system of equations:

1. The matrix A is non singular. There is a **unique solution** given by $x = A^{-1}b$.
2. The matrix A is singular and $b \in \text{range}(A)$. There are **infinitely many solutions**.
3. The matrix A is singular and $b \notin \text{range}(A)$. There are **no solutions**.

Example 1: Let

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 8 \end{bmatrix}.$$

Then A is non singular and there is a unique x given by

$$x = \begin{bmatrix} 1/2 \\ 2 \end{bmatrix}.$$

Example 2: Now let

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Then A is singular and $b \in \text{range}(A)$.
There are infinitely many solution given by

$$x(r) = \begin{bmatrix} 1/2 \\ r \end{bmatrix} \quad \forall r \in \mathbb{R}.$$

Example 3: Let A be the same as above, but define

$$b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The are no solutions because the second equation cannot be satisfied.

Lower Triangular System $Lx = b$ (p. 88)

$$\begin{bmatrix} 2 & 0 & 0 \\ 1 & 5 & 0 \\ 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \\ 9 \end{bmatrix}$$

One equation can be trivially solved, the first one

$$2x_1 = 6 \longrightarrow x_1 = 6/2 = 3.$$

Now that x_1 is known, we can solve the second equation

$$x_1 + 5x_2 = 8 \longrightarrow x_2 = (8 - x_1)/5 = (8 - 3)/5 = 1.$$

Now that x_1 and x_2 are known, we can solve the third equation

$$x_1 + 2x_2 + 2x_3 = 9 \longrightarrow x_3 = (9 - x_1 - 2x_2)/2 = (9 - 3 - 2)/2 = 2.$$

Forward Substitution (p. 89)

The algorithm used to solve an lower triangular system $Lx = b$ is known as **forward substitution**. The general procedure is obtained by solving the ith equation in $Lx = b$ for the ith variable x_i

$$x_i = \left(b_i - \sum_{j=1}^{i-1} \ell_{ij} x_j \right) \Bigg/ \ell_{ii}.$$

Note. The diagonal elements ℓ_{ii} must be nonzero.
The multiplication corresponds to a dot product of a row of L times the vector x.
Since b_i only is involved in the formula for x_i, the former may be overwritten by the latter.

Forward Substitution, Algorithm 3.1.1 (p. 89)

Algorithm 1 (Forward Substitution: Row Version) If $L \in \mathbb{R}^{n\times n}$ is lower triangular and $b \in \mathbb{R}^n$, then this algorithm overwrites b with the solution to $Lx = b$. L is assumed to be non singular.

$b(1) = b(1)/L(1,1)$
for $i = 2 : n$
  for $j = 1 : i-1$
    $b(i) = b(i) - L(i,j)b(j)$
  end
  $b(i) = b(i)/L(i,i)$
end

Cost of the Forward Substitution

If we analyse the algorithm on the previous slide, the C in flops is given by

$$
\begin{aligned}
C &= \underbrace{1}_{1:} + \underbrace{\sum_{i=2}^{n}}_{2:} \left(\underbrace{1}_{6:} + \underbrace{\sum_{j=1}^{i-1}}_{3:} \underbrace{2}_{4:} \right) \\
&= 1 + \sum_{i=2}^{n} (1 + 2(i-1)) = 1 + \sum_{i=2}^{n} 2i - 1 \\
&= 1 - (n-1) + 2\left(\frac{n(n+1)}{2} - 1\right) = n^2
\end{aligned}
$$

This algorithm is $O(n^2)$ flops.

Forward Substitution, Algorithm 3.1.1 (p. 89)

In the previous algorithm, the inner loop is a scalar product and can be expressed using the colon notation:

Algorithm 2 (Forward Substitution: Row Version with Colon Notation) If $L \in \mathbb{R}^{n \times n}$ is lower triangular and $b \in \mathbb{R}^n$, then this algorithm overwrites b with the solution to $Lx = b$. L is assumed to be non singular.

$b(1) = b(1)/L(1,1)$
for $i = 2 : n$
  $b(i) = (b(i) - L(i, 1 : i-1)b(1 : i-1))/L(i,i)$
end

Forward Substitution. Column Version (p. 90)

$$\begin{bmatrix} 2 & 0 & 0 \\ 1 & 5 & 0 \\ 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \\ 9 \end{bmatrix}$$

From the first row, we find $x_1 = 3$ and then we deal with a 2×2 system

$$\begin{bmatrix} 5 & 0 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 9 \end{bmatrix} - 3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}.$$

From the first row, we find $x_2 = 1$ and then we deal with a "1×1" system

$$\begin{bmatrix} 2 \end{bmatrix} \begin{bmatrix} x_3 \end{bmatrix} = \begin{bmatrix} 6 \end{bmatrix} - 1 \begin{bmatrix} 2 \end{bmatrix} = \begin{bmatrix} 4 \end{bmatrix}$$

From the "first row," we find $x_3 = 2$.

Algorithm 3 (Forward Substitution: Column Version) If $L \in \mathbb{R}^{n\times n}$ is lower triangular and $b \in \mathbb{R}^n$, then this algorithm overwrites b with the solution to $Lx = b$. L is assumed to be non singular.

for $j = 1 : n-1$
  $b(j) = b(j)/L(j,j)$
  for $i = j+1 : n$
    $b(i) = b(i) - b(j)L(i,j)$
  end
end
$b(n) = b(n)/L(n,n)$

Note: The inner loop is now a saxpy in the column version of the forward substitution.

Forward Substitution, Algorithm 3.1.3 (p. 90)

Algorithm 4 (Forward Substitution: Column Version with Colon Notation) If $L \in \mathbb{R}^{n \times n}$ is lower triangular and $b \in \mathbb{R}^n$, then this algorithm overwrites b with the solution to $Lx = b$. L is assumed to be non singular.

for $j = 1 : n-1$
  $b(j) = b(j)/L(j,j)$
  $b(j+1:n) = b(j+1:n) - b(j)L(j+1:n,j)$
end
$b(n) = b(n)/L(n,n)$

Upper Triangular Linear Systems $Ux = b$

$$\begin{bmatrix} 2 & 4 & 4 \\ 0 & 5 & -2 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}$$

One equation can be trivially solved, the last one

$$2x_3 = 4 \longrightarrow x_3 = 2.$$

Now that x_3 is known, we can solve the second equation

$$5x_2 - 2x_3 = 1 \longrightarrow x_2 = (1 + 2x_3)/5 = (1 + 4)/5 = 1$$

Now that x_3 and x_2 are known, we can solve the first equation

$$2x_1 + 4x_2 + 4x_3 = 2 \longrightarrow x_1 = (2 - 4x_2 - 4x_3)/2 = (2 - 4 - 8)/2 = -5.$$

Back Substitution (p. 89)

The algorithm used to solve an upper triangular system $Ux = b$ is known as **back substitution**. The general procedure is obtained by solving the ith equation in $Ux = b$ for the ith variable x_i

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) \Big/ u_{ii}.$$

Note. The diagonal elements u_{ii} must be nonzero.
The multiplication corresponds to a dot product of a row of U times the vector x.
Since b_i only is involved in the formula for x_i, the former may be overwritten by the latter.

Back Substitution, Algorithm 3.1.2 (p. 89)

Algorithm 5 (Back Substitution: Row Version) If $U \in \mathbb{R}^{n\times n}$ is upper triangular and $b \in \mathbb{R}^n$, then this algorithm overwrites b with the solution to $Ux = b$. U is assumed to be non singular.

$b(n) = b(n)/U(n,n)$
for $i = n-1 : -1 : 1$
  for $j = i+1 : n$
    $b(i) = b(i) - U(i,j)b(j)$
  end
  $b(i) = b(i)/U(i,i)$
end

Back Substitution, Algorithm 3.1.2 (p. 89)

In the previous algorithm, the inner loop is a scalar product and can be expressed using the colon notation:

Algorithm 6 (Back Substitution: Row Version with Colon Notation) If $U \in \mathbb{R}^{n\times n}$ is upper triangular and $b \in \mathbb{R}^n$, then this algorithm overwrites b with the solution to $Ux = b$. U is assumed to be non singular.

$b(n) = b(n)/U(n,n)$
for $i = n-1 : -1 : 1$
  $b(i) = (b(i) - U(i, i+1:n)b(i+1:n))/U(i,i)$
end

This algorithm requires n^2 flops.

Back Substitution. Column Version (p. 90)

$$\begin{bmatrix} 2 & 4 & 4 \\ 0 & 5 & -2 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}$$

From the last row, we find $x_3 = 2$ and then we deal with a 2×2 system

$$\begin{bmatrix} 2 & 4 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} - 2 \begin{bmatrix} 4 \\ -2 \end{bmatrix} = \begin{bmatrix} -6 \\ 5 \end{bmatrix}.$$

From the last row, we find $x_2 = 1$ and then we deal with a "1×1" system

$$\begin{bmatrix} 2 \end{bmatrix} \begin{bmatrix} x_1 \end{bmatrix} = \begin{bmatrix} -6 \end{bmatrix} - 1 \begin{bmatrix} 4 \end{bmatrix} = \begin{bmatrix} -10 \end{bmatrix}$$

From the "last row," we find $x_1 = -5$.

Back Substitution, Algorithm 3.1.4 (p. 90)

Algorithm 7 (Back Substitution: Column Version) If $U \in \mathbb{R}^{n\times n}$ is upper triangular and $b \in \mathbb{R}^n$, then this algorithm overwrites b with the solution to $Ux = b$. U is assumed to be non singular.

---

for $j = n : -1 : 2$
  $b(j) = b(j)/U(j,j)$
  for $i = 1 : j-1$
    $b(i) = b(i) - b(j)U(i,j)$
  end
end
$b(1) = b(1)/U(1,1)$

Note: The inner loop is now a saxpy in the column version of the back substitution.

Back Substitution, Algorithm 3.1.4 (p. 90)

Algorithm 8 (Back Substitution: Column Version with Colon Notation) If $U \in \mathbb{R}^{n \times n}$ is upper triangular and $b \in \mathbb{R}^n$, then this algorithm overwrites b with the solution to $Ux = b$. U is assumed to be non singular.

for $j = n : -1 : 2$
  $b(j) = b(j)/U(j,j)$
  $b(1 : j-1) = b(1 : j-1) - b(j)U(1 : j-1, j)$
end
$b(1) = b(1)/U(1,1)$

Backward Error Analysis for the Triangular Solve

The computed solution $\hat{x}$ of the triangular system $Ux = b$ computed by the previous algorithm satisfies:

$$(U + E)\hat{x} = b$$

with

$$|E| \leq n\, u\, |U| + O(u^2)$$

Backward error analysis. Computed x solves a slightly perturbed system.

Backward error not large in general. It is said that triangular solve is "backward stable".

Elementary Matrices

Definition

An **elementary matrix** is one that can be obtained from the identity matrix I_n through a single elementary row operation.

$$R3 \leftarrow R3 - 4R1 \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix} = E_1$$

$$R1 \leftrightarrow R2 \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = E_2$$

$$R3 \leftarrow 5R3 \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix} = E_3$$

Elementary Matrices

After the application of an elementary row operation on a matrix A of size $m \times n$, the resulting matrix can be written as EA where E is a square $m \times m$ matrix created by applying the same elementary row operation on the identity matrix I_m.

$R3 \leftarrow R3 - 4R1$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \approx \begin{bmatrix} a & b & c \\ d & e & f \\ g-4a & h-4b & i-4c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$L1 \leftrightarrow L2 \quad \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \approx \begin{bmatrix} d & e & f \\ a & b & c \\ g & h & i \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

Elementary Matrices

After the application of an elementary row operation on a matrix A of size $m \times n$, the resulting matrix can be written as EA where E is a square $m \times m$ matrix created by applying the same elementary row operation on the identity matrix I_m.

$$
\begin{aligned}
R3 \leftarrow 5R3 \quad \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} &\approx \begin{bmatrix} a & b & c \\ d & e & f \\ 5g & 5h & 5i \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}
\end{aligned}
$$

Inverse of Elementary Matrices

Theorem

Any elementary matrix E is invertible. The inverse of E is the elementary matrix that transforms back E into I.

$$R3 \leftarrow R3 + 4R1 \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix} \approx I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix}$$

$$L2 \leftrightarrow L1 \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx I_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R3 \leftarrow R3/5 \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix} \approx I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

Theorem of Inversibility

Theorem

An $n \times n$ matrix A is invertible if and only if A is row equivalent to I_n and, in this case, any sequence of elementary row operations that transforms A into I_n will transform also I_n into A^{-1}.

Proof

Proof.

There exists a sequence of elementary row operations E_1, E_2, ..., E_p such that

$$A \approx E_1 A \approx E_2(E_1 A) \approx E_p(E_{p-1} \cdots E_2 E_1 A) = I_n.$$

In other words $E_p E_{p-1} \cdots E_2 E_1 A = I_n$. The product $E_p E_{p-1} \cdots E_2 E_1$ of invertible matrices being invertible, we get

$$\begin{aligned} (E_p \cdots E_2 E_1)^{-1}(E_p \cdots E_2 E_1)A &= (E_p \cdots E_2 E_1)^{-1} I_n \\ A &= (E_p \cdots E_2 E_1)^{-1} \end{aligned}$$

Then

$$A^{-1} = ((E_p \cdots E_2 E_1)^{-1})^{-1} = (E_p \cdots E_2 E_1) = (E_p \cdots E_2 E_1) I_n.$$

A^{-1} is then the result of the successive application of E_1, E_2, ..., E_p to the matrix I_n.

Algorithm to Compute A^{-1}

Let A be an $n \times n$ matrix.

1. Adjoin the identity $n \times n$ matrix I_n to A to form the augmented system $\left[\begin{array}{c|c} A & I_n \end{array}\right]$.
2. Compute the reduced row echelon form of $\left[\begin{array}{c|c} A & I_n \end{array}\right]$.
3. If the reduced row echelon form is of the type $\left[\begin{array}{c|c} I_n & B \end{array}\right]$, then B is the inverse of A.
   If the reduced row echelon form is not of the type $\left[\begin{array}{c|c} I_n & B \end{array}\right]$, in that the first $n \times n$ submatrix is not I_n, the A has no inverse.

Linear Systems of Equations: Gaussian Elimination

Back to arbitrary linear systems.

Principle of the method: Since triangular systems are easy to solve, we will transform a linear system into one that is upper triangular. Main operation: combine rows so that zeros appear in the required locations to make the system triangular.

$$\left\{\begin{array}{rcrcrcr} 2x_1 & + & 4x_2 & + & 4x_3 & = & 2 \\ x_1 & + & 3x_2 & + & 1x_3 & = & 1 \\ x_1 & + & 5x_2 & + & 6x_3 & = & -6 \end{array}\right. \quad \text{Notation:} \left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 1 & 3 & 1 & 1 \\ 1 & 5 & 6 & -6 \end{array}\right]$$

Main Operation Used: Scaling and Adding Rows

Example: Replace row2 by row2 $-\frac{1}{2}\times$row1

$$\left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 1 & 3 & 1 & 1 \\ 1 & 5 & 6 & -6 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 1 & 5 & 6 & -6 \end{array}\right].$$

This is equivalent to

$$\begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 1 & 3 & 1 & 1 \\ 1 & 5 & 6 & -6 \end{array}\right] = \left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 1 & 5 & 6 & -6 \end{array}\right].$$

The left-hand matrix is of the form

$$M = I - ve_1^T \text{ with } v = \begin{bmatrix} 0 \\ \frac{1}{2} \\ 0 \end{bmatrix}.$$

Go back to original system. Step 1 must transform

$$\left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 1 & 3 & 1 & 1 \\ 1 & 5 & 6 & -6 \end{array}\right] \text{ into } \left[\begin{array}{ccc|c} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{array}\right].$$

$\text{row}_2 = \text{row}_2 - \frac{1}{2}\times\text{row}_1$:

$$\left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 1 & 5 & 6 & -6 \end{array}\right]$$

$\text{row}_3 = \text{row}_3 - \frac{1}{2}\times\text{row}_1$:

$$\left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 0 & 3 & 4 & -7 \end{array}\right]$$

Continued

Equivalent to

$$\begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{bmatrix} \times \left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 1 & 3 & 1 & 1 \\ 1 & 5 & 6 & -6 \end{array}\right] = \left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 0 & 3 & 4 & -7 \end{array}\right],$$

$$\left[\begin{array}{c|c} A & b \end{array}\right] \approx M_1 \left[\begin{array}{c|c} A & b \end{array}\right] \text{ where } M_1 = I - v_1 e_1^T \text{ with } v_1 = \begin{bmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}.$$

New system is $M_1 \left[\begin{array}{c|c} A & b \end{array}\right] = \left[\begin{array}{c|c} A_1 & b_1 \end{array}\right]$. Step 2 must transform

$$\left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 0 & 3 & 4 & -7 \end{array}\right] \text{ into } \left[\begin{array}{ccc|c} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{array}\right].$$

$$\text{row}_3 = \text{row}_3 - 3 \times \text{row}_2 : \rightarrow \left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 7 & -7 \end{array}\right]$$

Equivalent to

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix} \times \left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 0 & 3 & 4 & -7 \end{array}\right] = \left[\begin{array}{ccc|c} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 7 & -7 \end{array}\right].$$

This is a upper triangular system that we can solve with back substitution.

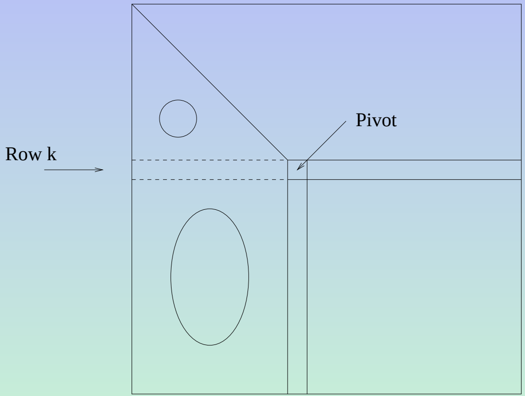
The second transformation is as follows:

$$\left[\begin{array}{c|c} A_1 & b_1 \end{array}\right] \approx M_2 \left[\begin{array}{c|c} A_1 & b_1 \end{array}\right] = \left[\begin{array}{c|c} A_2 & b_2 \end{array}\right]$$

$$\text{where } M_2 = I - v_2 e_2^T \text{ with } v_2 = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}.$$

Layout of the Gaussian Elimination

$A_k =$



Gaussian Elimination without Scaling the Pivot

$$[A|b] = C \approx \begin{array}{cl}
\left[\begin{array}{cc|ccccc}
c_{11} & X & c_{1k} & \cdots & c_{1j} & \cdots & c_{1n} \\
0 & c_{22} & \vdots & \cdots & \vdots & \cdots & \vdots \\ \hline
0 & 0 & \mathbf{c_{kk}} & \cdots & c_j & \cdots & c_{kn} \\
\vdots & \vdots & \vdots & & \vdots & & \vdots \\
0 & 0 & c_{ik} & \cdots & c_{ij} & \cdots & c_{in} \\
\vdots & \vdots & \vdots & & \vdots & & \vdots \\
0 & 0 & c_{mk} & \cdots & c_{mj} & \cdots & c_{mn}
\end{array}\right] & \begin{array}{l} \\ \\ \leftarrow \text{row } k \\ \\ \leftarrow \text{row } i \\ \\ \\ \end{array} \\
\begin{array}{ccccccc} \phantom{c_{11}} & \phantom{c_{22}} & \text{col } k & & \text{col } j & & \phantom{c_{mn}} \end{array} &
\end{array}$$

$$R_i \leftarrow R_i - (c_{ik}/c_{kk})R_k, \quad k+1 \le i \le m$$

$$c_{ij} \leftarrow c_{ij} - (c_{ik}/c_{kk})c_{kj}, \quad k+1 \le i \le m, \quad k \le j \le n$$

$$C(i,k{:}n) = C(i,k{:}n) - (C(i,k)/C(k,k))C(k,k{:}n) \quad k+1 \le i \le m$$

Algorithm of Gaussian Elimination

Algorithm 9 Gaussian Elimination. If $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, then this algorithm overwrites the augmented system $C = [A|b] \in \mathbb{R}^{n\times n+1}$ by an upper triangular one that can be solved with a back substitution. The pivots are assume to be non zero.

```
for k = 1 : n − 1
    for i = k + 1 : n
        multiplier = C(i, k)/C(k, k)
        for j = k + 1 : n + 1
            C(i, j) = C(i, j) − multiplier × C(k, j)
        end
    end
end
```

The pivot $A(k, k)$ must be checked to avoid a zero divide.

Complexity of Gaussian Elimination

Operation count in flops:

$$
\begin{aligned}
C &= \underbrace{\sum_{k=1}^{n-1}}_{1:} \underbrace{\sum_{i=k+1}^{n}}_{2:} \left(\underbrace{1}_{3:} + \underbrace{\sum_{j=k+1}^{n+1}}_{4:} \underbrace{2}_{5:} \right) = \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} (2(n-k)+3) \\
&= \sum_{k=1}^{n-1} (2(n-k)+3)(n-k) \\
&= \ldots \\
&= \frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n
\end{aligned}
$$

(Complete the above calculation...)

The LU Factorization

Now ignore the right-hand side from the transformations.

Observation: Gaussian elimination is equivalent to $n-1$ successive **Gaussian transformations**, i.e., multiplications with matrices of the form $M_k = I - v_k e_k^T$, where the first k components of v_k equal zero.

Set $A_0 \equiv A$. Then

$$
\begin{aligned}
A &\approx M_1 A_0 = A_1 \\
A_1 &\approx M_2 A_1 = A_2 = M_2 M_1 A_0 \\
A_2 &\approx M_3 A_2 = A_3 = M_3 M_2 M_1 A_0 \\
&\vdots \\
A_{n-1} &\approx M_{n-1} A_{n-2} = A_{n-1} \equiv U
\end{aligned}
$$

Last $A_k \equiv U$ is an upper triangular matrix.

Continued

At each step we have $A_k = M_{k+1}^{-1} A_{k+1}$. Therefore

$$\begin{aligned} A_0 &= M_1^{-1} A_1 \\ &= M_1^{-1} M_2^{-1} A_2 \\ &= M_1^{-1} M_2^{-1} M_3^{-1} A_3 \\ &= \dots \\ &= M_1^{-1} M_2^{-1} M_3^{-1} \cdots M_{n-1}^{-1} A_{n-1} \end{aligned}$$

$$L = M_1^{-1} M_2^{-1} M_3^{-1} \cdots M_{n-1}^{-1}$$

Note: L is **Lower triangular**, A_{n-1} is **Upper triangular**

LU decomposition $A = LU$

How To Get L?

$$L = M_1^{-1} M_2^{-1} M_3^{-1} \cdots M_{n-1}^{-1}$$

Consider only the first 2 matrices in this product.

Note $M_k^{-1} = (I - v_k e_k^T)^{-1} = (I + v_k e_k^T)$. So

$$M_1^{-1} M_2^{-1} = (I + v_1 e_1^T)(I + v_2 e_2^T) = I + v_1 e_1^T + v_2 e_2^T$$

Generally,

$$M_1^{-1} M_2^{-1} \cdots M_k^{-1} = I + v_1 e_1^T + v_2 e_2^T + \cdots + v_k e_k^T$$

The L factor is a lower triangular matrix with ones on the diagonal. Column k of L, contains the multipliers ℓ_{ik} used in the k-th step of Gaussian elimination.

Example of an LU Decomposition

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 3 & 6 & 10 \end{bmatrix} \quad R_2 - 2R_1$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 0 & -6 & -11 \end{bmatrix} \quad R_3 - 3R_1$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 0 & 0 & 1 \end{bmatrix} \quad R_3 - 2R_2$$

Suppose that $A = LU$. The system $Ax = LUx = b$ can be solved in two steps. First, use a forward substitution to solve $Ly = b$. Second, use a backward substitution to solve $Ux = y$. For ex., if

$$b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \text{ and } A = LU = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 0 & 0 & 1 \end{bmatrix},$$

then $Ly = b$, i.e.

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

implies $y = (1, -1, 0)^T$. Second, $Ux = y$, i.e.

$$\begin{bmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$$

implies $x = (-1/3, 1/3, 0)^T$.

LU Decomposition. Algorithm 3.2.1 (p. 98)

Algorithm 10 LU Decomposition. Suppose $A \in \mathbb{R}^{n \times n}$ has the property that $A(1:k, 1:k)$ is non singular for $k = 1:n-1$. This algorithm computes the factorization $M_{n-1} \cdots M_1 A = U$ where U is upper triangular and each M_k is a Gauss transform. U is stored in the upper triangle of A. The multipliers associated with M_k are stored in $A(k+1:n, k)$, i.e., $A(k+1:n, k) = -M_k(k+1:n, k)$.

```
for k = 1 : n − 1
    for i = k + 1 : n
        multiplier = A(i, k)/A(k, k)
        A(i, k) = multiplier
        for j = k + 1 : n
            A(i, j) = A(i, j) − multiplier × A(k, j)
        end
    end
end
```

Determinant from the LU Decomposition

A matrix A has an LU decomposition if

$$\det(A(1:k, 1:k)) \neq 0 \quad \text{for} \quad k = 1, \cdots, n-1.$$

In this case, the determinant of A satisfies:

$$\det A = \det(U) = \prod_{i=1}^{n} u_{ii}$$

If, in addition, A is non singular, then the LU factorization is unique.

Questions

Show how to obtain L directly from the "multipliers" [Sec. 3.2.7]

Practical use: Show how to use the LU factorization to solve linear systems with the same matrix A and different b's.

LU factorization of the matrix $A = \begin{bmatrix} 2 & 4 & 4 \\ 1 & 5 & 6 \\ 1 & 3 & 1 \end{bmatrix}$?

Determinant of A?

True or false: "Computing the LU factorization of matrix A involves more arithmetic operations than solving a linear system $Ax = b$ by Gaussian elimination".