We are going to look at the algorithm for Gaussian Elimination as a sequence of matrix operations (multiplies).

Not really how you want to implement it, but gives a better framework for the theory, and our next topic:

LU-factorization.

Permutations

- A permutation matrix $P$ is a re-ordering of the identity matrix $I$. It can be used to:
  - Interchange the order of the equations
    - Interchange the rows of $A$ and $b$
  - Interchange the order of the variables
    - This technique changes the order of the solution variables.
    - Hence a reordering is required after the solution is found.

Permutation Matrix

- Properties of a Permutation matrix:
  - $|P| = 1 \Rightarrow $ non-singular
  - $P^{-1} = P$
  - $P^T = P$

\[
P = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Switches equations 1 and 3.

\[
PA = \begin{bmatrix}
a_{11} & a_{12} & a_{13} & a_{14} \\
a_{21} & a_{22} & a_{23} & a_{24} \\
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
= \begin{bmatrix}
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{21} & a_{22} & a_{23} & a_{24} \\
a_{11} & a_{12} & a_{13} & a_{14} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{bmatrix}
\]
**Permutation Matrix**

- Order is important!

\[ AP = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{13} & a_{12} & a_{11} & a_{14} \\ a_{23} & a_{22} & a_{21} & a_{24} \\ a_{33} & a_{32} & a_{31} & a_{34} \\ a_{43} & a_{42} & a_{41} & a_{44} \end{bmatrix} \]

Switches variables 1 and 3.

**Permutation Matrix**

- Apply a permutation to a linear system:

\[ (PA)x = Pb \]

- Changes the order of the equations (need to include \( b \)), whereas:

\[ (AP)x' = b, \text{ where } x = Px' \]

- Permutes the order of the variables (\( b \)'s stay the same).

**Adding Two Equations**

- What matrix operation allows us to add two rows together?
- Consider \( MA \), where:

\[ M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

Leaves this equation alone
Leaves this equation alone
Adds equations 2 and 3
Leaves this equation alone

**Undoing the Operation**

- Note that the inverse of this operation is to simply subtract the unchanged equation 2 from the new equation 3.

\[ M^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
Gaussian Elimination

- The first set of multiply and add operations in Gaussian Elimination can thus be represented as:

\[ M_1 A x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & 0 & 0 \\ -\frac{a_{31}}{a_{11}} & 0 & 1 & 0 \\ -\frac{a_{41}}{a_{11}} & 0 & 0 & 1 \end{bmatrix} \]

\[ Ax = M_1 b \]

\[ M_1 M_2 A x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -\frac{a'_{22}}{a'_{22}} & 1 & 0 \\ 0 & -\frac{a'_{12}}{a'_{22}} & 0 & 1 \end{bmatrix} \]

\[ Ax = M_2 M_1 b \]

Gaussian Elimination

- Note, the scale factors in the second step use the new set of equations (a')!

\[ M_1^{-1} = \begin{bmatrix} \frac{a_{21}}{a_{11}} & 1 & 0 & 0 \\ \frac{a_{31}}{a_{11}} & 0 & 1 & 0 \\ \frac{a_{41}}{a_{11}} & 0 & 0 & 1 \end{bmatrix} \]

\[ M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \frac{a'_{12}}{a'_{22}} & 1 & 0 \\ 0 & \frac{a'_{12}}{a'_{22}} & 0 & 1 \end{bmatrix} \]

Gaussian Elimination

- The composite of all of these matrices reduce A to a triangular form:

\[ M A x = M_{n-1} \cdots M_1 A x = M_{n-1} \cdots M_1 b = M b \]

\[ \text{upper triangular} \]

- Can rewrite this:
  - \( U x = y \) where \( U = MA \)
  - \( M b = y \) or \( M^T y = b \)

- What is \( M^{-1} \)?
  - Just add the scaled row back in!
Gaussian Elimination

- These are all lower triangular matrices.
- The product of lower triangular matrices is another lower triangular matrix.
- These are even simpler!

\[
M_1^{-1}M_2^{-1} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
\frac{a_{21}}{a_{11}} & 1 & 0 & 0 \\
\frac{a_{31}}{a_{11}} & \frac{a_{32}'}{a_{22}'} & 1 & 0 \\
\frac{a_{41}}{a_{11}} & \frac{a_{42}'}{a_{22}'} & 0 & 1 \\
\end{bmatrix}
\]
- Just keep track of the scale factors!!

LU Factorization

- Let \( L = M^{-1} \) and \( U = MA \)
  - \( L \) is a lower triangular matrix with 1’s on the diagonal.
  - \( U \) is an upper triangular matrix:

\[
\begin{align*}
    Ly &= b \quad \text{forward substitution} \\
    Ux &= y \quad \text{backward substitution}
\end{align*}
\]

LU Factorization

- Note, \( L \) and \( U \) are only dependent on \( A \).
  - \( A = LU \) – a factorization of \( A \)

Hence, \( Ax = b \) implies

- \( LUx = b \) or
- \( Ly = b \) where \( Ux = y \)

Find \( y \) and then we can solve for \( x \).
- Both operations in \( O(n^2) \) time.

LU Factorization

- Problem: How do we compute the LU factorization?
- Answer: Gaussian Elimination
  - Which is \( O(n^3) \) time, so no free lunch!
In many cases, the matrix $A$ defines the structure of the problem, while the vector $b$ defines the current state or initial conditions. The structure remains fixed! Hence, we need to solve a set or sequence of problems:

$$Ax_k = b_k \text{ or } Ax(t_k) = b(t_k)$$

LU Factorization works great for these problems:

$$Ly_k = b_k$$
$$Ux_k = y_k$$

If we have $M$ problems or time steps, we have $O(n^3 + Mn^2)$ versus $O(Mn^3)$ time complexity. In many situations, $M > n$.

C# Implementation

```csharp
// Factor A into LU in-place A->LU
for (int k = 0; k < n - 1; k++) {
    try {
        for (int i = k + 1; i < n; i++) {
            double s = a[i,k] / a[k,k];
            for (int j = k + 1; j < n; j++)
                a[i,j] -= a[k,j] * s;
        }
    }
    catch (DivideByZeroException e) {
        Console.WriteLine(e.Message);
    }
}
```