Numerical Linear Algebra I

Mohamed Amine Hamadi

February 9, 2026

UM6P University Mohammed VI Polytechnic

THE UM6P VANGUARD CENTER

#Empowering Minds.

Plan

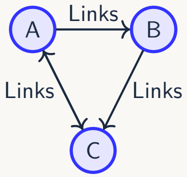
Linear Systems : Motivation and Examples

Example 1: How does Google rank websites?

The Challenge: There are over **1 Billion** websites. When you search "Linear Algebra", Google must decide which one to show first in 0.5 seconds.

The Logic (PageRank):

- A website is "Important" if other important websites link to it.
- It's a "Voting System".



A mini-internet

Example 1: The Equations of Importance

Let x_A, x_B, x_C be the "score" of each website.

The Rule: Your score is the sum of the scores of websites linking to you (shared equally).

For our mini-internet:

$$\begin{aligned} \text{Score of A } (x_A) &= \text{from C only} \to \mathbf{1 \cdot x_C} \\ \text{Score of B } (x_B) &= \text{from A only} \to \mathbf{\frac{1}{2} \cdot x_A} \\ \text{Score of C } (x_C) &= \text{from A (half)} + \text{from B (full)} \to \mathbf{\frac{1}{2} x_A + 1 x_B} \end{aligned}$$

Notice: The score of A depends on C, and C depends on A. It's a circular system!

Example 1: The Matrix Form of the Web

We can write this circular system as $Ax = x$ (or looking for specific vector states).

$$\begin{bmatrix} x_A \\ x_B \\ x_C \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0 \\ 0.5 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_A \\ x_B \\ x_C \end{bmatrix}$$

The Matrix on the right represents the "Structure of the Internet".

- Row 1 tells you who links to Page A.
- Column 1 tells you who Page A links to.

Example 1: The Explosion of Complexity

Our example had 3 websites.

$$\text{Matrix Size} = 3 \times 3$$

This is easy to solve on paper.

The Real Internet:

- 1 Billion+ websites ($N = 10^9$).
- Matrix Size $= 10^9 \times 10^9$.
- That is **1,000,000,000,000,000,000 (1 Quintillion)** numbers.

Why we need Numerical Linear Algebra

1. The matrix is too big to store in RAM completely. 2. Standard "hand" methods are too slow. 3. We need special algorithms (like Power Iteration) to find the solution without solving every single number explicitly.

The challenge: Google updates these scores every day. If the calculation takes 1 hour, the results are outdated (new links are added constantly).

Why do we need this?

Imagine you are organizing a show. You need to balance the budget.

The Problem:

- You sell **3** Standard tickets and **2** VIP tickets.
- Total revenue: **500 DH**.
- Later, you sell **1** Standard and **1** VIP ticket.
- Total revenue: **200 DH**.

Why do we need this?

Imagine you are organizing a show. You need to balance the budget.

The Problem:

- You sell **3** Standard tickets and **2** VIP tickets.
- Total revenue: **500 DH**.
- Later, you sell **1** Standard and **1** VIP ticket.
- Total revenue: **200 DH**.

Question: What is the price of a Standard ticket (x_1) and a VIP ticket (x_2)?

$$3x_1 + 2x_2 = 500$$
$$1x_1 + 1x_2 = 200$$

Why do we need this?

Imagine you are organizing a show. You need to balance the budget.

The Problem:

- You sell **3** Standard tickets and **2** VIP tickets.
- Total revenue: **500 DH**.
- Later, you sell **1** Standard and **1** VIP ticket.
- Total revenue: **200 DH**.

Question: What is the price of a Standard ticket (x_1) and a VIP ticket (x_2)?

$$3x_1 + 2x_2 = 500$$
$$1x_1 + 1x_2 = 200$$

Time Cost (directly by substitutions)

- 15-20 minutes for 3 equations
- High error probability
- For 10, 100 equations: **days of work!**

The Matrix: A Grid of Numbers

We store the coefficients in a **Matrix** (A) and the unknowns in a **Vector** (x).

$$\underbrace{\begin{bmatrix} 3 & 2 \\ 1 & 1 \end{bmatrix}}_{\text{Matrix } A \text{ (Coefficients)}} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\text{Vector } x \text{ (Unknowns)}} = \underbrace{\begin{bmatrix} 500 \\ 200 \end{bmatrix}}_{\text{Vector } b \text{ (Values)}}$$

Notation

We write this system simply as:

$$Ax = b$$

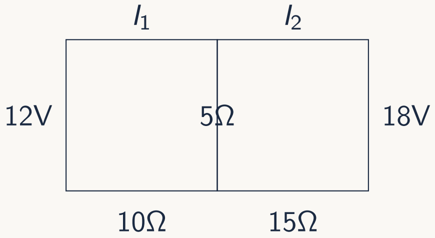
Linear Systems in Circuit Analysis

Example: Solving for Currents

Using Kirchhoff's Voltage Law, each loop in a circuit gives one linear equation.

$$\begin{cases} 10I_1 + 5I_2 = 12 \\ 5I_1 + 15I_2 = 18 \end{cases}$$

Solving this system gives the electric currents in the circuit.



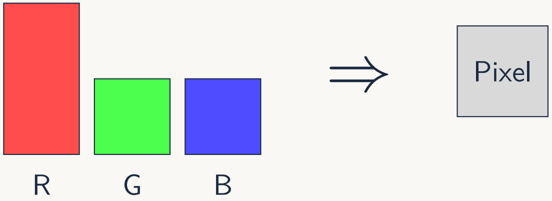
Linear Systems in Computer Graphics

Example: RGB Color Mixing

A pixel color is computed by mixing Red, Green, and Blue intensities.

$$\begin{cases} R + G + B = 1 \\ R = 2G \\ B = G \end{cases}$$

Solving the system gives the exact color of the pixel.



General Form

Every linear systems of equations can be presented as :

$$A\mathbf{x} = \mathbf{b},$$

where:

- $A \in \mathbb{R}^{n\times n}$ is the coefficient matrix
- $\mathbf{x} \in \mathbb{R}^n$ is the unknown vector
- $\mathbf{b} \in \mathbb{R}^n$ is the right-hand side vector

Gaussian Elimination

Gaussian Elimination

The Basic Idea

Goal

Transform the system to **upper triangular form** using elementary row operations:

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}$$

Elementary Operations

1. Swap two rows
2. Multiply a row by a nonzero scalar
3. Add a multiple of one row to another

Gaussian Elimination: 3 by 3 Matrix

Given linear system

$$\begin{cases} x + y + z = 6 \\ 2x + 3y + z = 11 \\ -x + y + 2z = 5 \end{cases}$$

Gaussian Elimination: 3 by 3 Matrix

Given linear system

$$\begin{cases} x + y + z = 6 \\ 2x + 3y + z = 11 \\ -x + y + 2z = 5 \end{cases}$$

Augmented matrix

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 2 & 3 & 1 & 11 \\ -1 & 1 & 2 & 5 \end{array}\right]$$

Gaussian Elimination: 3 by 3 Matrix (continued)

Forward elimination

Rq: 1 is the pivot in position $(1, 1)$. We eliminate 1st entry in Row 2 and 3.

Operation 1: $R_2 \leftarrow R_2 - 2R_1$
$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & 1 & -1 & -1 \\ -1 & 1 & 2 & 5 \end{array}\right]$$

Operation 2: $R_3 \leftarrow R_3 + R_1$
$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & 1 & -1 & -1 \\ 0 & 2 & 3 & 11 \end{array}\right]$$

Rq: 1 is the pivot in position $(2, 2)$. Now, we eliminate 2nd entry in Row 3.

Operation 3: $R_3 \leftarrow R_3 - 2R_2$
$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 5 & 13 \end{array}\right]$$

Back Substitution

We need to solve

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 5 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 6 \\ -1 \\ 13 \end{bmatrix}$$

The solution using Back substitution

$$z = \frac{13}{5},$$
$$y = \frac{8}{5},$$
$$x = \frac{9}{5}$$

General view

Algorithm 1: Gaussian Elimination

```
for $k = 1$ to $n - 1$ do
    for $i = k + 1$ to $n$ do
        $m_{ik} = a_{ik} / a_{kk}$ (multiplier)
        for $j = k$ to $n$ do
            $a_{ij} = a_{ij} - m_{ik} \cdot a_{kj}$
        $b_i = b_i - m_{ik} \cdot b_k$
```

Back Substitution

Once in upper triangular form $U\mathbf{x} = \mathbf{y}$:

$$x_n = \frac{y_n}{u_{nn}}, \quad x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j \right)$$

LU factorization

What if b changes?

In real engineering :

- The structure A: stiffness matrix of a bridge
  depends on geometry and materials (**doesn't change**)
- b: applied forces
  cars, wind... (**change every second.**)

Do we really have to do Gaussian Elimination from scratch every time?

The Solution: LU Factorization We split A into two special matrices:

$$A = L \cdot U$$

- **U**: The Upper Triangular matrix (result of elimination).
- **L**: The Lower Triangular matrix (the "memory" of what we did).

Visualizing L and U

$$\underbrace{\begin{bmatrix} 3 & 2 & -1 \\ 1 & 5 & 4 \\ 2 & -1 & 2 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ \dots & \dots & 1 \end{bmatrix}}_{L \text{ (The "Multipliers")}} \cdot \underbrace{\begin{bmatrix} 3 & 2 & -1 \\ 0 & 4.3 & \dots \\ 0 & 0 & \star \end{bmatrix}}_{U \text{ (The Result)}}$$

Why is this cool?

1. Solve $Lz = b$ (This is easy, go *forward*).
2. Solve $Ux = z$ (This is easy, go *backward*).

We only factorize A once!

Why pivoting ?

The Problem

Without pivoting:

$$\begin{bmatrix} 0.0001 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Multiplier: $m = 1/0.0001 = 10000$

- Large multipliers amplify round-off errors
- Division by zero possible

Solution : Partial Pivoting

At step k, find p such that:

$$|a_{pk}| = \max_{i=k,k+1,\dots,n} |a_{ik}|.$$

Then swap rows k and p.

Gaussian Elimination with Partial Pivoting

Given linear system

$$A = \begin{pmatrix} 0.001 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

Gaussian Elimination with Partial Pivoting (Continued)

Step $k = 1$: Choose the pivot

We select p such that

$$|a_{p1}| = \max_{i=1,2,3} |a_{i1}|$$

$$|0.001|,\ |2|,\ |1| \quad \Rightarrow \quad p = 2$$

Swap rows 1 and 2

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 0.001 & 1 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

Proceed with elimination

Now the pivot is $a_{11} = 2$, which is large and numerically stable.

What if we hit a zero?

Consider this system:

$$\left[\begin{array}{cc|c} \mathbf{0} & 1 & 2 \\ 2 & 3 & 4 \end{array}\right]$$

To eliminate the 2 in the bottom left, we calculate: $Row_2 \leftarrow Row_2 - \frac{2}{0} Row_1$.

CRASH! Division by zero.

What if we hit a zero?

Consider this system:

$$\left[\begin{array}{cc|c} \mathbf{0} & 1 & 2 \\ 2 & 3 & 4 \end{array}\right]$$

To eliminate the 2 in the bottom left, we calculate: $Row_2 \leftarrow Row_2 - \frac{2}{0} Row_1$.

CRASH! Division by zero.

Pivoting Strategy: Simply **swap** the rows so the non-zero number is on top!

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

Now we have a triangle. Problem solved.

Computational cost and Stability

Computational Cost (Big-O Notation)

Why don't we just do elimination by hand? Because n can be huge (e.g., $n = 100,000$ for a 3D simulation).

Let's compare the number of multiplications needed:

Method	**Operations (Approx)**	**If** n=1000
Gaussian Elimination / LU	$\frac{2}{3}n^3$	$\approx 6.6 \times 10^8$ (0.6 billion)
Forward/Back Substitution	n^2	10^6 (1 million)

Utility: Finding the LU decomposition is expensive (n^3). Solving $Lz = b$ and $Ux = z$ is cheap (n^2).
So we do the hard work once, solve cheaply for the rest of the day.

Numerical Stability

Computers store numbers with limited precision (e.g., 3.14159...). They drop digits.

Sometimes, bad pivoting leads to huge errors due to rounding.

Example:

$$0.0001x + y = 1$$

$$x + y = 2$$

If we don't swap rows (don't pivot), we might divide by a tiny number (0.0001), multiplying the round-off error by $10,000$. The answer becomes garbage.

Conclusion: Always pick the largest number as the "Pivot" to minimize rounding errors. This is called **Partial Pivoting**.

A modern Intel i9 laptop can perform roughly:

$$10^{10} \text{ to } 10^{11} \text{arithmetic operations per second.}$$

Matrix size n	**Operations** $\sim n^3$	**Time (i9 laptop)**
100	10^6	Instant
1,000	10^9	$<$ 1 second
5,000	1.25×10^{11}	$\sim$ 10 seconds
10,000	10^{12}	Minutes
50,000	1.25×10^{14}	Hours

Table: Growth of computational cost for cubic algorithms