Linear System Of Equations
Part 2

Calcoli di Processo dell' Ingegneria Chimica

POLITECNICO MILANO 1863
DEPARTMENT OF CHEMISTRY MATERIALS AND CHEMICAL ENGINEERING

Timoteo Dinelli

21st of October 2025

Department of Chemistry, Materials and Chemical Engineering, "Giulio Natta", Politecnico di Milano.

email: timoteo.dinelli@polimi.it

Linear System Of Equations

A system of linear equations consists of several linear equations that must all be satisfied simultaneously. A solution is a vector whose elements, when substituted for the unknowns, satisfy all equations.

From the classical representation to the **matrix** form:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = b_2 \\ \qquad \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = b_n \end{cases} \qquad \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \ldots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

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

Last practical

Gauss Elimination transforms the matrix **A** into an upper triangular matrix $\mathbf{A}^*$ through systematic row operations:

$$\mathbf{A}^* = \left[\begin{array}{cccc|c} a_{1,1}^{(0)} & \dots & \dots & a_{1,n}^{(0)} & b_1^{(0)} \\ 0 & a_{2,2}^{(1)} & \dots & a_{2,n}^{(1)} & b_2^{(1)} \\ \vdots & \dots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & a_{n,n}^{(n-1)} & b_n^{(n-1)} \end{array}\right]$$

Algorithm: At step k, eliminate column k below the diagonal:

$$m_{i,k} = \frac{a_{i,k}^{(k-1)}}{a_{k,k}^{(k-1)}}, \quad i = k+1, \dots, n$$

$$a_{i,j}^{(k)} = a_{i,j}^{(k-1)} - m_{i,k} \cdot a_{k,j}^{(k-1)}$$

Then solve by back substitution.

LU Decomposition factorizes **A** into a lower triangular matrix **L** and an upper triangular matrix **U** such that:

$$\mathbf{A} = \mathbf{LU}$$

Example for 3×3 **system:**

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ \ell_{21} & 1 & 0 \\ \ell_{31} & \ell_{32} & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Solution process:

1. Solve $\mathbf{Ly} = \mathbf{b}$ by forward substitution
2. Solve $\mathbf{Ux} = \mathbf{y}$ by back substitution

Advantage: Once computed, **L** and **U** can be reused for multiple right-hand sides **b**.

When Methods Can Fail

Singular matrices: If $\det(A) = 0$, the system has either:

- No solution (inconsistent)
- Infinitely many solutions (underdetermined)

Numerical issues during elimination:

- Zero pivot: If $a_{k,k}^{(k-1)} = 0$, division by zero occurs
- Small pivot: If $a_{k,k}^{(k-1)} \approx 0$, amplifies rounding errors

Solution: Partial pivoting

- At each step, swap rows to bring the largest element to the pivot position
- Improves numerical stability significantly
- MATLAB's `lu(A)` and `linsolve(A, b)` use pivoting by default

Partial Pivoting

Problem: Small or zero pivots cause numerical instability or failure.
Solution: At step k, swap rows to maximize $|a_{k,k}^{(k-1)}|$.
Algorithm:

1. At elimination step k, find the row $p \geq k$ with maximum $|a_{p,k}^{(k-1)}|$:

$$|a_{p,k}^{(k-1)}| = \max_{i=k,\dots,n} |a_{i,k}^{(k-1)}|$$

2. If $p \neq k$, swap rows p and k in both $\mathbf{A}^{(k-1)}$ and $\mathbf{b}^{(k-1)}$
3. Proceed with standard Gauss elimination using the new pivot

Benefits:

- Avoids division by zero (if matrix is non-singular)
- Minimizes rounding error propagation
- Guarantees $|m_{i,k}| \leq 1$ for all multipliers

Partial Pivoting: Example

Solve $\mathbf{Ax} = \mathbf{b}$ with partial pivoting:

Initial system:

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]$$

Step 1: Find max in column 1

- $|a_{1,1}| = 2$, $|a_{2,1}| = 3$, $|a_{3,1}| = 2$
- Swap rows 1 and 2

$$\left[\begin{array}{ccc|c} -3 & -1 & 2 & -11 \\ 2 & 1 & -1 & 8 \\ -2 & 1 & 2 & -3 \end{array}\right]$$

Eliminate column 1:

$$\left[\begin{array}{ccc|c} -3 & -1 & 2 & -11 \\ 0 & 1/3 & 1/3 & 2/3 \\ 0 & 1/3 & 10/3 & -35/3 \end{array}\right]$$

Step 2: Find max in column 2 (rows 2-3)

- $|a_{2,2}| = 1/3$, $|a_{3,2}| = 1/3$ (equal, no swap)

Eliminate column 2:

$$\left[\begin{array}{ccc|c} -3 & -1 & 2 & -11 \\ 0 & 1/3 & 1/3 & 2/3 \\ 0 & 0 & 3 & -37/3 \end{array}\right]$$

Back substitution: $\mathbf{x} = [3, 1, 2]^T$

Scaled Partial Pivoting (Balanced Pivoting)

Problem: Partial pivoting ignores the relative magnitude of coefficients in each row.

Idea: Choose pivot based on relative size compared to other elements in its row.

Algorithm:

1. Compute the scaling factors for each row (done once at the beginning):

$$s_i = \max_{j=1,\ldots,n} |a_{i,j}|, \quad i = 1, \ldots, n$$

2. At elimination step k, find the row $p \geq k$ that maximizes the scaled pivot:

$$\frac{|a_{p,k}^{(k-1)}|}{s_p} = \max_{i=k,\ldots,n} \frac{|a_{i,k}^{(k-1)}|}{s_i}$$

3. If $p \neq k$, swap rows p and k (and their scaling factors)
4. Proceed with standard Gauss elimination

Note: Scaling factors remain constant after row swaps, not recomputed.

Scaled Partial Pivoting: Example

Consider the system where row magnitudes differ significantly:

Initial system:

$$\left[\begin{array}{cc|c} 2 & 100000 & 100000 \\ 1 & 1 & 2 \end{array}\right]$$

Standard partial pivoting:

- $|a_{1,1}| = 2 > |a_{2,1}| = 1$
- No swap! Use pivot $a_{1,1} = 2$
- Multiplier: $m_{2,1} = 1/2$
- Result: Poor numerical behavior

Scaled partial pivoting:

- Compute scales:

$$s_1 = 100000, \quad s_2 = 1$$

- Compare scaled pivots:

$$\frac{|a_{1,1}|}{s_1} = \frac{2}{100000} = 0.00002$$

$$\frac{|a_{2,1}|}{s_2} = \frac{1}{1} = 1$$

- Swap rows! Better numerical stability

Conclusion: Scaled pivoting accounts for different row magnitudes, providing better stability for ill-conditioned systems.

Exercises

MATLAB Implementation

Function signature:

```
function [A, b] = gauss_elimination_scaled_pivoting(A, b)
```

Key steps in the implementation:

1. Compute scaling factors once at the beginning:
   ```
   s = max(abs(A), [], 2);  % Maximum absolute value per row
   ```
2. For each column k, find the best pivot:
   ```
   [~, p] = max(abs(A(k:n, k)) ./ s(k:n));
   ```
3. Swap rows in both **A**, **b**, and scaling vector **s**
4. Eliminate below the pivot using standard Gauss elimination

Exercise 1: Why Scaling Matters

System with vastly different row magnitudes:

$$\begin{bmatrix} 2 & 100000 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 100000 \\ 2 \end{bmatrix}$$

Without scaling:

- Choose pivot $a_{1,1} = 2$ (larger)
- Multiplier: $m_{2,1} = 0.5$
- Row 2 becomes: $[0, -49999]$
- Numerical instability!

With scaled pivoting:

- $s_1 = 100000$, $s_2 = 1$
- Scaled: $2/100000$ vs $1/1$
- Swap rows! Use pivot $a_{2,1} = 1$
- Better stability

```
[A_scaled, b_scaled] = gauss_elimination_scaled_pivoting(A, b);
x = back_substitution(A_scaled, b_scaled);
```

Exercise 2: Complete 3×3 System

Solve the system from lecture slides:

$$\begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -11 \\ -3 \end{bmatrix}$$

```
A = [2, 1, -1; -3, -1, 2; -2, 1, 2];
b = [8; -11; -3];

% Apply Gauss elimination with scaled pivoting
[A_upper, b_upper] = gauss_elimination_scaled_pivoting(A, b);

% Solve by back substitution
x = back_substitution(A_upper, b_upper);
```

Result: $\mathbf{x} = [3, 1, 2]^T$, **Verification:** $||\mathbf{Ax} - \mathbf{b}|| \approx 0$

Exercise 3: Ill-Conditioned Systems

The Hilbert matrix is notoriously difficult to solve numerically:

$$H_{ij} = \frac{1}{i+j-1}, \quad \text{e.g., } H_4 = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix}$$

Condition number: $\kappa(\mathbf{H}_4) \approx 1.55 \times 10^4$ (very ill-conditioned!)

```
A = hilb(4);
b = sum(A, 2);  % Ensures solution is x = [1, 1, 1, 1]'

[A_upper, b_upper] = gauss_elimination_scaled_pivoting(A, b);
x = back_substitution(A_upper, b_upper);
```

Key insight: Even with scaled pivoting, ill-conditioned systems require careful numerical handling!

Exercise 4: Chemical Engineering Application

Material balance for a multi-component system: Three components (A, B, C) flowing through three units. Find flow rates x_1, x_2, x_3 (kmol/h):

$$
\begin{aligned}
2x_1 + 3x_2 + x_3 &= 100 \quad \text{(Component A balance)} \\
x_1 + 2x_2 + 3x_3 &= 150 \quad \text{(Component B balance)} \\
3x_1 + x_2 + 2x_3 &= 120 \quad \text{(Component C balance)}
\end{aligned}
$$

```
A = [2, 3, 1; 1, 2, 3; 3, 1, 2];
b = [100; 150; 120];

[A_upper, b_upper] = gauss_elimination_scaled_pivoting(A, b);
x = back_substitution(A_upper, b_upper);

fprintf('Flow rates: x1=%.2f, x2=%.2f, x3=%.2f kmol/h\n', x);
```

Solution: $x_1 = 10$ kmol/h, $x_2 = 20$ kmol/h, $x_3 = 30$ kmol/h

Best Practices and Tips

When to use scaled partial pivoting:

- ✓ Systems with coefficients of vastly different magnitudes
- ✓ Ill-conditioned matrices (high condition number)
- ✓ When numerical stability is critical
- ✓ Material/energy balance problems with different units

Implementation tips:

- Always check for singular matrices: $\det(\mathbf{A}) \approx 0$
- Verify your solution: compute residual $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|$
- For very large systems, consider iterative methods
- MATLAB's built-in `linsolve` uses pivoting by default

Thank you for your attention!