M1: Solution of linear systems – systems of linear equations, matrices, solving systems of linear equations.

🔷 1. What is a System of Linear Equations?

A **system of linear equations** is a set of equations where each equation is linear, i.e., of the form:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

Example (2 equations, 2 variables):

$$\begin{aligned} 2x + 3y &= 8 \\ 4x - y &= 2 \end{aligned}$$

Goal: Find values of x and y that satisfy both equations **simultaneously**.

🔷 2. Representing the System Using Matrices

You can represent a system like this using **matrix notation**:

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

Where:

- A is the **coefficient matrix**
- $\mathbf{x}$ is the **variable vector**
- $\mathbf{b}$ is the **output vector** (RHS)

From the above example:

$$\begin{bmatrix} 2 & 3 \\ 4 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 8 \\ 2 \end{bmatrix}$$

🔷 3. **Methods to Solve Linear Systems**

🔹 a) **Substitution and Elimination** (manual, for small systems)

Good for simple cases by hand.

🔹 b) **Matrix Inversion Method** (when A is square and invertible)

$$\mathbf{x} = A^{-1}\mathbf{b}$$

- Fast, but **only works if A is square** (n×n) **and invertible** (non-singular).
- Not numerically stable for large systems.

🔹 c) **Gaussian Elimination / Row Reduction**

This method transforms the matrix A into **row echelon form** (or reduced row echelon form) via **elementary row operations**.

$$\begin{bmatrix} 2 & 3 & | & 8 \\ 4 & -1 & | & 2 \end{bmatrix} \xrightarrow{\text{row ops}} \text{Echelon form} \Rightarrow \text{Back-substitution}$$

This is the basis for most algorithms and calculators.

🔹 d) **LU Decomposition**

Factor the matrix A as:

$$A = LU$$

Where:

- L is lower triangular
- U is upper triangular

Solve using:

1. $L\mathbf{y} = \mathbf{b}$
2. $U\mathbf{x} = \mathbf{y}$

Efficient for solving multiple systems with the same A but different $\mathbf{b}$.

🔹 e) **Least Squares Solution** (if system is overdetermined, $m > n$)

When there are **more equations than variables** (common in data science):

$$A^T A\mathbf{x} = A^T\mathbf{b} \Rightarrow \mathbf{x} = (A^T A)^{-1} A^T\mathbf{b}$$

Minimizes the squared error $||A\mathbf{x} - \mathbf{b}||^2$

🔹 f) **Numerical Methods** (for very large systems)

Used when matrices are too large for exact algebraic solutions:

- **Iterative methods**:
  - Jacobi method
  - Gauss-Seidel method
  - Conjugate Gradient
- Used in simulations, optimization, machine learning

🔷 4. Types of Solutions

A system $A\mathbf{x} = \mathbf{b}$ may have:

Type	Description	Condition
Unique solution	One exact solution	$\det(A) \neq 0$ (full rank)
No solution	Inconsistent equations	Contradiction (e.g., $0 = 1$)
Infinite solutions	Underdetermined system, free variables	Matrix A is **not full rank**

🔷 5. Applications in Data Science

- **Linear regression**: Solving $X\beta = y$ using least squares
- **Optimization problems**: Constraints expressed as systems
- **Transformations**: Image, text, and feature transformation via matrices
- **Neural networks**: Forward pass as matrix equations

✅ Summary

Concept	Description
Linear system	Set of linear equations to solve together
Matrix form	$A\mathbf{x} = \mathbf{b}$
Inverse method	Fast but only works if A is square and invertible
Gaussian elimination	Step-by-step row simplification
LU decomposition	Fast for repeated solutions
Least squares	Best-fit solution for overdetermined systems
Applications	ML, data fitting, simulations, modeling

M2: Vectors Spaces - linear independence, basis and rank, affine spaces, Norms, inner products, Lengths and distances, Angles and orthogonality, Orthonormal basis

🔷 1. Vector Spaces

A **vector space** is a set of vectors where you can:

- Add any two vectors and stay in the space
- Multiply a vector by a scalar and stay in the space

> Formally: A set V is a vector space over a field (like $\mathbb{R}$ or $\mathbb{C}$) if it satisfies **8 axioms** (closure, associativity, identity, etc.)

Example:

$$\mathbb{R}^2 = \left\{ \begin{bmatrix} x \\ y \end{bmatrix} \mid x, y \in \mathbb{R} \right\}$$

is a 2D vector space.

🔷 2. Linear Independence

A set of vectors is **linearly independent** if **none** of them can be written as a combination of the others.

$$\text{Vectors } \{v_1, v_2, \ldots, v_k\} \text{ are independent if } c_1v_1 + c_2v_2 + \cdots + c_kv_k = 0 \Rightarrow c_1 = c_2 = \cdots = c_k = 0$$

Example:

- $\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ are independent
- $\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 4 \end{bmatrix}$ are **dependent** (second is 2× first)

🔷 3. **Basis and Rank**

✅ **Basis:**

A **basis** is a set of **linearly independent vectors** that **span** the vector space.

> Every vector in the space can be written **uniquely** as a combination of the basis vectors.

✅ **Rank:**

The **rank** of a matrix is the **dimension** of the vector space spanned by its rows or columns (i.e., number of linearly independent rows/columns).

Example:

For a 2×3 matrix with 2 independent rows, **rank = 2**.

🔷 4. **Affine Spaces**

An **affine space** is like a vector space, but without a fixed origin.

It consists of:

- A point (anchor) + directions (from a vector space)

Example:

A line not passing through the origin is **not** a vector space but is an **affine subspace**.

If x is a point and v is a direction, the line $x + tv$ for $t \in \mathbb{R}$ is affine.

🔷 5. **Norms**

A **norm** is a function that assigns a **length** or **size** to a vector.

Common norms:

- **L2 norm (Euclidean):**

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

- **L1 norm (Manhattan):**

$$\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$

- **Infinity norm:**

$$\|x\|_\infty = \max |x_i|$$

🔷 6. **Inner Products**

An **inner product** generalizes the **dot product** and measures how aligned two vectors are.

$$\langle x, y \rangle = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

It's used to:

- Define **length**: $\|x\| = \sqrt{\langle x, x \rangle}$
- Compute **angles and projections**

🔷 7. **Lengths and Distances**

- **Length** (or norm) of vector x:

$$\|x\| = \sqrt{x \cdot x}$$

- **Distance** between x and y:

$$d(x, y) = \|x - y\|$$

Used in:

- Measuring errors
- Nearest neighbor algorithms
- Clustering

🔷 8. **Angles and Orthogonality**

- The **angle** θ between vectors x and y is:

$$\cos(\theta) = \frac{\langle x, y \rangle}{\|x\| \|y\|}$$

- Vectors are **orthogonal** (i.e., at 90°) if:

$$\langle x, y \rangle = 0$$

🔷 9. **Orthonormal Basis**

A **basis** is **orthonormal** if:

- All vectors have unit length: $\|v_i\| = 1$
- All pairs are orthogonal: $\langle v_i, v_j \rangle = 0$ for $i \neq j$

Benefits:

- Simplifies projections and decomposition
- Used in PCA (Principal Component Analysis), Fourier analysis

✅ Summary Table

Concept	Key Idea
Vector Space	A set closed under addition & scalar mult.
Linear Independence	No vector in the set is a combo of others
Basis	Minimal set of independent vectors that span
Rank	Dimension of space spanned by matrix rows/cols
Affine Space	Shifted vector space (no origin required)
Norm	Measures size or length of a vector
Inner Product	Generalization of dot product (alignment)
Length & Distance	Vector length and difference between vectors
Angle & Orthogonality	Relationship between vectors' directions
Orthonormal Basis	Basis with perpendicular, unit-length vectors

M3: Matrix Decomposition methods - Determinant and Trace, Eigenvalues and Eigenvectors, Cholesky decomposition, Eigen-decomposition and diagonalization, singular value decomposition, matrix approximation

🔷 1. Determinant and Trace

✅ Determinant ($\det(A)$)

A scalar that describes **volume scaling** and **invertibility** of a matrix.

- For 2×2:

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

- If $\det(A) = 0$, the matrix is **singular** (not invertible)

Geometric Meaning:

- In 2D/3D: how much the matrix **scales space**
- If negative, it also **flips orientation**

✅ **Trace** ($\mathrm{tr}(A)$)

Sum of the diagonal elements:

$$\mathrm{tr}(A) = \sum_{i=1}^{n} a_{ii}$$

Used in:

- Sum of eigenvalues: $\mathrm{tr}(A) = \sum \lambda_i$
- Optimization (e.g., regularization penalties)

🔷 2. **Eigenvalues and Eigenvectors**

Given a square matrix A, an **eigenvector** v satisfies:

$$Av = \lambda v$$

- λ: eigenvalue
- v: eigenvector (non-zero)

Meaning:

- The matrix scales the vector without changing its direction
- Found by solving:

$$\det(A - \lambda I) = 0$$

Why Important?

- PCA: top eigenvectors = principal components
- Stability of systems (differential equations)
- Spectral clustering, quantum mechanics, and more

🔷 3. **Cholesky Decomposition**

For **symmetric, positive definite** matrices A:

$$A = LL^T$$

- L: lower triangular matrix
- More efficient and stable than LU decomposition in some cases
- Used in:
  - Solving linear systems
  - Bayesian statistics
  - Gaussian Processes

🔷 4. **Eigen-Decomposition and Diagonalization**

If a matrix A is **diagonalizable**, it can be written as:

$$A = PDP^{-1}$$

Where:

- D is a diagonal matrix of eigenvalues
- P contains eigenvectors as columns

Meaning:

- Converts matrix multiplication to **scalar multiplication**
- Speeds up computations (especially powers of matrices: $A^k = PD^kP^{-1}$)

🔷 5. **Singular Value Decomposition (SVD)**

For any $m \times n$ matrix A:

$$A = U\Sigma V^T$$

- U: orthogonal matrix (left singular vectors)
- Σ: diagonal matrix of **singular values**
- V^T: transpose of orthogonal matrix (right singular vectors)

Applications:

- **Dimensionality reduction** (PCA)
  $A \to A_k \approx U_k \Sigma_k V_k^T$
- **Noise filtering**
- **Latent semantic analysis** (text mining)
- **Recommender systems** (matrix completion)

🔷 6. **Matrix Approximation**

✅ Low-Rank Approximation:

Use only top k singular values in SVD:

$$A_k = U_k \Sigma_k V_k^T \quad \text{(rank-}k\text{ approx)}$$

Benefit:

- Reduces storage and computation
- Captures main structure of data (like in image compression)

✅ Frobenius Norm Error:

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

Shows how much information you lose.

✅ Summary Table:

Concept	Description
Determinant	Scalar: volume scale, invertibility
Trace	Sum of diagonal elements; sum of eigenvalues
Eigenvalues/vectors	Directions scaled by matrix; foundational in PCA
Cholesky Decomposition	$A = LL^T$; fast for symmetric positive definite matrices
Eigen-Decomposition	$A = PDP^{-1}$; simplifies power & exponential
SVD	Works for any matrix; powerful for data approximation
Matrix Approximation	Keep top singular values to reduce data while preserving structure