Linear Algebra Notes

Abror Maksudov

Created: November 8, 2025

Last Updated: 2025-11-15 23:17:40+05:00

Contents

1 Vectors

1.1 Definition of a Scalar

A scalar is a **single number**, denoted as $a, b, c, \alpha, \beta, \gamma, \lambda, \mu$.

Space Complexity: $O(1)$ — *Time Complexity*: *N/A*.

1.2 Definition of a Vector

A vector is an **ordered list** of numbers, usually denoted as $\mathbf{v}$ or $\vec{v}$.

- **Column Vector (Default/Standard):**

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}^T$$

- **Row Vector:**

$$\vec{v} = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \end{pmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}^T$$

Space Complexity: $O(n)$ — *Time Complexity*: *N/A*.

1.3 Vector Operations

For vectors $\vec{u}, \vec{v} \in \mathbb{R}^n$ and scalar $c \in \mathbb{R}$, the following rules hold:

1. **Scalar Multiplication:**
$$c\vec{v} = c \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} cv_1 \\ cv_2 \\ \vdots \\ cv_n \end{bmatrix}$$

2. **Vector Addition:**
$$\vec{u} + \vec{v} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \\ \vdots \\ u_n + v_n \end{bmatrix}$$
Requirement: Both vectors must have the same dimension.

3. **Vector Subtraction:**
$$\vec{u} - \vec{v} = \vec{u} + (-1)\vec{v} = \begin{bmatrix} u_1 - v_1 \\ u_2 - v_2 \\ \vdots \\ u_n - v_n \end{bmatrix}$$
Requirement: Both vectors must have the same dimension.

Space Complexity: $O(n)$ — *Time Complexity*: $O(n)$.

1.4 Properties of Vector

Let $\vec{u}, \vec{v}, \vec{w} \in \mathbb{R}^n$ and $c, d \in \mathbb{R}$. Then:

(1)	$\vec{u} + \vec{v} = \vec{v} + \vec{u}$	(Commutativity)
(2)	$(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w})$	(Associativity)
(3)	$\vec{v} + \vec{0} = \vec{v}$	(Identity)
(4)	$\vec{v} + (-\vec{v}) = \vec{0}$	(Inverse)
(5)	$c(d\vec{v}) = (cd)\vec{v}$	(Associativity of scalars)
(6)	$1 \cdot \vec{v} = \vec{v}$	(Multiplicative identity)
(7)	$c(\vec{u} + \vec{v}) = c\vec{u} + c\vec{v}$	(Distributivity over vectors)
(8)	$(c + d)\vec{v} = c\vec{v} + d\vec{v}$	(Distributivity over scalars)

These 8 axioms define a linear space (or vector space) in $\mathbb{R}^n$.

1.5 Linear Combinations

A vector $\vec{w} \in \mathbb{R}^n$ is a **linear combination** of vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k \in \mathbb{R}^n$ if there exist scalars $c_1, c_2, \ldots, c_k \in \mathbb{R}$ such that:

$$\vec{w} = c_1\vec{v}_1 + c_2\vec{v}_2 + \cdots + c_k\vec{v}_k = \sum_{i=1}^{k} c_i\vec{v}_i = \begin{bmatrix} | & | & & | \\ \vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_k \\ | & | & & | \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix} = \vec{w}$$

The scalars $c_1, c_2, \ldots, c_k$ are called **coefficients** or **weights** of the linear combination.

Space Complexity: $O(n)$ — *Time Complexity*: $O(nk)$.

1.6 Dot Product

The **dot product** of two vectors $\vec{u}, \vec{v} \in \mathbb{R}^n$ is a scalar defined as:

$$\vec{u} \cdot \vec{v} = u_1v_1 + u_2v_2 + \cdots + u_nv_n = \sum_{i=1}^{n} u_iv_i$$

Geometric View:

$$\vec{u} \cdot \vec{v} = \|\vec{u}\|\|\vec{v}\| \cos\theta$$

where θ is the angle between $\vec{u}$ and $\vec{v}$, and $\|\vec{u}\|$ is the **magnitude** (or **length** or **norm**) of $\vec{u}$.

Alternative notations:

$$\vec{u} \cdot \vec{v} = \langle \vec{u}, \vec{v} \rangle = \mathbf{u}^T\mathbf{v}$$

Special Cases:

$$\begin{aligned}
&(1) \quad \vec{u} \cdot \vec{v} = 0 \iff \theta = 90^\circ && \text{(Perpendicular)} \\
&(2) \quad \vec{u} \cdot \vec{v} > 0 \iff \theta < 90^\circ && \text{(Acute angle)} \\
&(3) \quad \vec{u} \cdot \vec{v} < 0 \iff \theta > 90^\circ && \text{(Obtuse angle)} \\
&(4) \quad \vec{u} \cdot \vec{u} = \|\vec{u}\|^2 \iff \theta = 0^\circ && \text{(Squared magnitude)}
\end{aligned}$$

Space Complexity: $O(1)$ — *Time Complexity*: $O(n)$.
Note: Highly optimized using SIMD (Single Instruction, Multiple Data) operations.

1.7 Properties of the Dot Product

Let $\vec{u}, \vec{v}, \vec{w} \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Then:

$$
\begin{aligned}
&(1) \quad \vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{u} && \text{(Commutativity)} \\
&(2) \quad \vec{u} \cdot (\vec{v} + \vec{w}) = \vec{u} \cdot \vec{v} + \vec{u} \cdot \vec{w} && \text{(Distributivity)} \\
&(3) \quad (c\vec{u}) \cdot \vec{v} = c(\vec{u} \cdot \vec{v}) = \vec{u} \cdot (c\vec{v}) && \text{(Scalar multiplication)} \\
&(4) \quad \vec{u} \cdot \vec{u} \geq 0 && \text{(Positive definiteness)} \\
&(5) \quad \vec{u} \cdot \vec{u} = 0 \iff \vec{u} = \vec{0} && \text{(Definiteness)} \\
&(6) \quad \vec{0} \cdot \vec{v} = 0 && \text{(Zero vector)}
\end{aligned}
$$

These properties define the dot product as an **inner product** on $\mathbb{R}^n$.

Space Complexity: $O(1)$ — *Time Complexity*: $O(n)$.

1.8 Cross Product

The **cross product** of two vectors $\vec{u}, \vec{v} \in \mathbb{R}^3$ is a vector $\vec{u} \times \vec{v} \in \mathbb{R}^3$ that is **perpendicular** to both $\vec{u}$ and $\vec{v}$ as is defined as:

$$\vec{u} \times \vec{v} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

Note: Cross product is only defined in $\mathbb{R}^3$ (and $\mathbb{R}^7$, but rarely used).
Space Complexity: $O(1)$ — *Time Complexity*: $O(1)$

1.9 Properties of the Cross Product

Let $\vec{u}, \vec{v}, \vec{w} \in \mathbb{R}^3$, $c \in \mathbb{R}$ and θ be the angle between $\vec{u}$ and $\vec{v}$. Then:

$$
\begin{array}{lll}
(1) & \vec{u} \times \vec{v} = -(\vec{v} \times \vec{u}) & \text{(Anti-commutativity)} \\
(2) & \vec{u} \times (\vec{v} + \vec{w}) = \vec{u} \times \vec{v} + \vec{u} \times \vec{w} & \text{(Distributivity)} \\
(3) & (c\vec{u}) \times \vec{v} = c(\vec{u} \times \vec{v}) = \vec{u} \times (c\vec{v}) & \text{(Scalar multiplication)} \\
(4) & \vec{u} \times \vec{u} = \vec{0} & \text{(Self cross product)} \\
(5) & \vec{u} \times \vec{0} = \vec{0} \times \vec{u} = \vec{0} & \text{(Zero vector)} \\
(6) & \vec{u} \times \vec{v} = \vec{0} \iff \vec{u} \parallel \vec{v} & \text{(Parallel vectors)} \\
(7) & \vec{u} \cdot (\vec{u} \times \vec{v}) = 0, \quad \vec{v} \cdot (\vec{u} \times \vec{v}) = 0 & \text{(Orthogonality)} \\
(8) & \|\vec{u} \times \vec{v}\| = \|\vec{u}\|\|\vec{v}\| \sin\theta & \text{(Magnitude)}
\end{array}
$$

Warning: The cross product is **not associative** in general: $\vec{u} \times (\vec{v} \times \vec{w}) \neq (\vec{u} \times \vec{v}) \times \vec{w}$.

Space Complexity: $O(1)$ — *Time Complexity*: $O(1)$

1.10 Scalar Projection

Let $\vec{u}, \vec{v} \in \mathbb{R}^n$. Then, the **scalar (component) projection** of $\vec{u}$ onto $\vec{v}$ is the signed length of the projection:

$$\text{comp}_{\vec{v}}\vec{u} = \frac{\vec{u} \cdot \vec{v}}{\|\vec{v}\|}$$

Space Complexity: $O(1)$ — *Time Complexity*: $O(n)$.

1.11 Vector Projection

Let $\vec{u}, \vec{v} \in \mathbb{R}^n$. Then, the **vector projection** of $\vec{u}$ onto $\vec{v}$ is a vector in the direction of $\vec{v}$:

$$\text{proj}_{\vec{v}}\vec{u} = \frac{\vec{u} \cdot \vec{v}}{\|\vec{v}\|^2}\vec{v} = \frac{\vec{u} \cdot \vec{v}}{\vec{v} \cdot \vec{v}}\vec{v} = (\text{comp}_{\vec{v}}\vec{u})\,\frac{\vec{v}}{\|\vec{v}\|}$$

Orthogonal Component:
The component of $\vec{u}$ orthogonal to $\vec{v}$ is:

$$\vec{u}_\perp = \vec{u} - \text{proj}_{\vec{v}}\vec{u}$$

Note that $\vec{u} = \text{proj}_{\vec{v}}\vec{u} + \vec{u}_\perp$ (orthogonal decomposition).

Space Complexity: $O(n)$ — *Time Complexity*: $O(n)$.

1.12 Angle Between Vectors

The **angle** θ between two non-zero vectors $\vec{u}, \vec{v} \in \mathbb{R}^n$ is given by:

$$\cos\theta = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|} \iff \theta = \arccos\left(\frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|}\right), \quad 0 \leq \theta \leq \pi$$

Special Cases:

$$\begin{aligned}
&(1) \quad \theta = 0 \iff \vec{u} \text{ and } \vec{v} \text{ point in the same direction} && (\cos\theta = 1) \\
&(2) \quad \theta = \frac{\pi}{2} \iff \vec{u} \perp \vec{v} && (\cos\theta = 0) \\
&(3) \quad \theta = \pi \iff \vec{u} \text{ and } \vec{v} \text{ point in opposite directions} && (\cos\theta = -1)
\end{aligned}$$

Note: The formula $\vec{u} \cdot \vec{v} = \|\vec{u}\|\|\vec{v}\| \cos\theta$ is the geometric definition of the dot product.
Space Complexity: $O(1)$ — *Time Complexity*: $O(n)$

1.13 Cauchy-Schwarz Inequality

Let $\vec{u}, \vec{v} \in \mathbb{R}^n$. Then:

$$|\langle \vec{u}, \vec{v} \rangle|^2 \leq \langle \vec{u}, \vec{u} \rangle \cdot \langle \vec{v}, \vec{v} \rangle,$$

or,

$$|\langle \vec{u}, \vec{v} \rangle|^2 \leq \|\vec{u}\|^2 \|\vec{v}\|^2 \iff |\langle \vec{u}, \vec{v} \rangle| = \|\vec{u}\| \|\vec{v}\| |\cos\theta| \leq \|\vec{u}\| \|\vec{v}\|,$$

or in component form,

$$\left(\sum_{i=1}^{n} u_i v_i\right)^2 \leq \left(\sum_{i=1}^{n} u_i^2\right)\left(\sum_{i=1}^{n} v_i^2\right).$$

Equality holds if and only if $\vec{u}$ and $\vec{v}$ are linearly dependent.

Space Complexity: $O(1)$ — *Time Complexity*: $O(n)$

1.14 Triangle Inequality

Let $\vec{u}, \vec{v} \in \mathbb{R}^n$. Then:

$$\|\vec{u} + \vec{v}\| \leq \|\vec{u}\| + \|\vec{v}\|$$

Equality holds if and only if $\vec{u} = c\vec{v}$ with $c \geq 0$.
Reverse Triangle Inequality:

$$\big|\|\vec{u}\| - \|\vec{v}\|\big| \leq \|\vec{u} - \vec{v}\|$$

Space Complexity: $O(1)$ — *Time Complexity*: $O(n)$

1.15 Vectorization for Computational Efficiency

Vectorization is the process of replacing explicit loops with vector/matrix operations to achieve dramatic performance improvements by leveraging optimized, low-level linear algebra libraries and hardware parallelism.

Performance Example:

Operation	Complexity	Loop (Python)	Vectorized (NumPy)
Dot Product ($n = 10^6$)	$O(n)$	~ 500 ms	~ 1 ms
Matrix-Vector ($n \times n$)	$O(n^2)$	~ 2000 ms	~ 10 ms
Matrix-Matrix ($n \times n$)	$O(n^3)$	$\sim$ hours	$\sim$ seconds

Why Vectorization is Faster:

1. **Hardware (SIMD):** Modern CPUs use **S**ingle **I**nstruction, **M**ultiple **D**ata (SIMD) instructions. This allows a single CPU instruction to perform the same operation (e.g., multiplication) on multiple data elements (e.g., 8 or 16 numbers) at the same time. Vectorized code uses these instructions automatically; Python loops do not.

2. **Cache Efficiency:** Vectorized operations access memory in large, contiguous blocks. This is very "cache-friendly" and avoids the high cost of fetching data from main memory (RAM) for each small step of a loop.

3. **Optimized Libraries:** NumPy, PyTorch, and TensorFlow operations are thin wrappers around highly optimized libraries (like BLAS/LAPACK, MKL, cuBLAS) written in C or Fortran. These libraries have been fine-tuned for decades.

Space-Time Trade-off:

- Loops: $O(1)$ extra space, slow
- Vectorized: May need $O(n)$ temporary arrays, fast
- *Rule:* Memory is cheap, time is expensive

2 Matrices

2.1 Definition of a Matrix

A matrix is a **rectangular array** of numbers arranged in rows and columns, denoted as A, B, C or $\mathbf{A}, \mathbf{B}, \mathbf{C}$.

An $m \times n$ matrix A has m rows and n columns:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = [a_{ij}]_{m\times n}$$

where a_{ij} is the element in the i-th row and j-th column also denoted as A_{ij}.

Notation:

- $A \in \mathbb{R}^{m \times n}$ means A is an $m \times n$ matrix with real entries
- $(A)_{ij} = a_{ij}$ denotes the (i, j)-entry of matrix A
- $m \times n$ is the **dimension** or **size** of the matrix

Special Cases:

$$
\begin{aligned}
&(1) \quad m = n \implies A \text{ is a } \textbf{square matrix} \quad (n \times n) \\
&(2) \quad m = 1 \implies A \text{ is a } \textbf{row vector} \quad (1 \times n) \\
&(3) \quad n = 1 \implies A \text{ is a } \textbf{column vector} \quad (m \times 1) \\
&(4) \quad m = n = 1 \implies A \text{ is a } \textbf{scalar} \quad (1 \times 1)
\end{aligned}
$$

Interpretations: Unlike vectors, matrices have multiple common interpretations:

1. **As Data:** A way to store data in a grid.
2. **As a Collection of Vectors:**

$$
\begin{bmatrix} | & | & & | \\ \vec{c}_1 & \vec{c}_2 & \cdots & \vec{c}_n \\ | & | & & | \end{bmatrix}
\qquad\qquad
\begin{bmatrix} - & \vec{r}_1 & - \\ - & \vec{r}_2 & - \\ & \vdots & \\ - & \vec{r}_m & - \end{bmatrix}
$$

Set of column vectors — Set of row vectors

3. **As a Linear Transformation:** A function that "transforms" a vector $\vec{x}$ into a new vector $\vec{y}$ via matrix-vector multiplication ($\vec{y} = A\vec{x}$). The matrix can scale, rotate, or shear space.

Storage: Matrices are typically stored in row-major or column-major order in memory.
Space Complexity: $O(mn)$ — *Time Complexity*: $O(1)$

2.2 Types of Matrices

1. **Square Matrix:** $A \in \mathbb{R}^{n \times n}$ (number of rows = number of columns)

$$
A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}_{3 \times 3}
$$

2. **Zero Matrix:** All entries are zero, denoted O or $0_{m \times n}$

$$
O = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{2 \times 3}
$$

3. **Identity Matrix:** Square matrix with ones on the main diagonal and zeros else-

where, denoted I or I_n

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Formally: $(I_n)_{ij} = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$ (Kronecker delta function)

4. **Diagonal Matrix:** Square matrix with non-zero entries only on the main diagonal

$$D = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} = \text{diag}(d_1, d_2, d_3)$$

5. **Upper Triangular Matrix:** All entries below the main diagonal are zero

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Formally: $u_{ij} = 0$ for $i > j$

6. **Lower Triangular Matrix:** All entries above the main diagonal are zero

$$L = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$$

Formally: $l_{ij} = 0$ for $i < j$

7. **Symmetric Matrix:** Square matrix where $A = A^T$ (equal to its transpose)

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

Formally: $a_{ij} = a_{ji}$ for all i, j

8. **Skew-Symmetric Matrix:** Square matrix where $A = -A^T$

$$A = \begin{bmatrix} 0 & 2 & -3 \\ -2 & 0 & 5 \\ 3 & -5 & 0 \end{bmatrix}$$

Formally: $a_{ij} = -a_{ji}$ for all i, j (diagonal entries must be zero)

9. **Orthogonal Matrix:** Square matrix where $A^T A = AA^T = I$ (columns and rows are orthonormal)

10. **Row Echelon Form (REF):** Matrix where:
    - All nonzero rows are above any rows of all zeros
    - Leading entry (pivot) of each nonzero row is to the right of the leading entry of the row above it

11. **Reduced Row Echelon Form (RREF):** REF where:
    - Each leading entry is 1
    - Each leading 1 is the only nonzero entry in its column

Note: Special matrix types often have computational advantages (e.g., diagonal matrix multiplication is $O(n)$ instead of $O(n^3)$).

2.3 Matrix Operations

For matrices $A, B \in \mathbb{R}^{m\times n}$, $\vec{x} \in \mathbb{R}^n$ and scalar $c \in \mathbb{R}$, the following rules hold::

1. **Scalar Multiplication:**
$$(cA)_{ij} = c(a_{ij})$$
Space Complexity: $O(mn)$ — *Time Complexity*: $O(mn)$.

2. **Matrix Addition:**
$$(A + B)_{ij} = a_{ij} + b_{ij}$$
Requirement: Both matrices must have the same dimension.
Space Complexity: $O(mn)$ — *Time Complexity*: $O(mn)$.

3. **Matrix Subtraction:**
$$(A - B)_{ij} = a_{ij} - b_{ij}$$
Requirement: Both matrices must have the same dimension.
Space Complexity: $O(mn)$ — *Time Complexity*: $O(mn)$.

4. **Matrix Transpose:**
$$(A^T)_{ij} = a_{ji}$$
Space Complexity: $O(mn)$ — *Time Complexity*: $O(mn)$.

5. **Matrix-Vector Multiplication:**
$$\vec{y} = A\vec{x} \quad (A \in \mathbb{R}^{m\times n} : \vec{x} \in \mathbb{R}^n \mapsto \vec{y} \in \mathbb{R}^m)$$
Space Complexity: $O(m)$ — *Time Complexity*: $O(mn)$.

6. **Matrix-Matrix Multiplication:**
$$C = AB \quad (A \in \mathbb{R}^{m\times k}, B \in \mathbb{R}^{k\times n} \implies C \in \mathbb{R}^{m\times n})$$
Space Complexity: $O(mn)$ — *Time Complexity*: $O(mnk)$.
Note: For two $n \times n$ matrices, the time complexity is $O(n^3)$.

2.4 Matrix-Vector Multiplication

The product of a matrix $A \in \mathbb{R}^{m\times n}$ and a column vector $\vec{x} \in \mathbb{R}^{n\times 1}$ results in a column vector $\vec{y} \in \mathbb{R}^{m\times 1}$ and is defined as:

$$A\vec{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix} = \vec{y}.$$

Requirement: The number of columns in A must equal the dimension of $\vec{x}$.
This operation can be interpreted in two essential ways:

1. **The "Row Picture" (Dot Product Method):**
   Each entry of the output vector $\vec{y}$ is the **dot product** of the corresponding **row of** A with the vector $\vec{x}$.

   $$\begin{bmatrix} - & \vec{r}_1 & - \\ - & \vec{r}_2 & - \\ & \vdots & \\ - & \vec{r}_m & - \end{bmatrix} \begin{bmatrix} | \\ \vec{x} \\ | \end{bmatrix} = \begin{bmatrix} \vec{r}_1 \cdot \vec{x} \\ \vec{r}_2 \cdot \vec{x} \\ \vdots \\ \vec{r}_m \cdot \vec{x} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

   This is the standard method used for manual computation.

2. **The "Column Picture" (Linear Combination Method):**
   The output vector $\vec{y}$ is a **linear combination** of the **columns of** A, where the **entries of** $\vec{x}$ **are the weights**.

   $$\begin{bmatrix} | & | & & | \\ \vec{c}_1 & \vec{c}_2 & \cdots & \vec{c}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} | \\ \vec{c}_1 \\ | \end{bmatrix} + x_2 \begin{bmatrix} | \\ \vec{c}_2 \\ | \end{bmatrix} + \cdots + x_n \begin{bmatrix} | \\ \vec{c}_n \\ | \end{bmatrix}$$

   This interpretation is more conceptual and is fundamental to understanding linear systems, column space, and rank.

Space Complexity: $O(m)$ — *Time Complexity*: $O(mn)$

2.5 Matrix-Matrix Multiplication

For matrices $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$, the product $C = AB \in \mathbb{R}^{m\times n}$ is defined as:

$$C = AB = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mk} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{k1} & b_{k2} & \cdots & b_{kn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix}$$

where each entry is computed as:

$$c_{ij} = \sum_{r=1}^{k} a_{ir}b_{rj} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{ik}b_{kj}$$

Requirement: The number of columns in A must equal the number of rows in B.
This operation can be interpreted in multiple ways:

1. **The "Entry-wise" View (Dot Product Method):**
   Each entry c_{ij} of the product matrix C is the **dot product** of the i-th **row of** A with the j-th **column of** B.

$$c_{ij} = (\text{row } i \text{ of } A) \cdot (\text{column } j \text{ of } B) = \sum_{r=1}^{k} a_{ir} b_{rj}$$

$$\begin{bmatrix} \cdots & \vec{r}_i & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ \vec{c}_j \\ \vdots \end{bmatrix} = \begin{bmatrix} \ddots & \vdots & \\ \cdots & C_{ij} & \cdots \\ & \vdots & \ddots \end{bmatrix}$$

   This is the standard computational method.

2. **The "Column Picture":**
   Each **column of** AB is a **linear combination of the columns of** A, with weights taken from the corresponding **column of** B.

$$\text{col}_j(C) = A \cdot (\text{col}_j(B))$$

$$AB = A \begin{bmatrix} | & | & & | \\ \vec{b}_1 & \vec{b}_2 & \cdots & \vec{b}_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ A\vec{b}_1 & A\vec{b}_2 & \cdots & A\vec{b}_n \\ | & | & & | \end{bmatrix}$$

   This view shows that every column of AB must be in the column space of A.

3. **The "Row Picture":**
   Symmetrically, each **row of** AB is a **linear combination of the rows of** B, with weights taken from the corresponding **row of** A.

$$\text{row}_i(C) = (\text{row}_i(A)) \cdot B$$

$$AB = \begin{bmatrix} - & \vec{a}_1 & - \\ - & \vec{a}_2 & - \\ & \vdots & \\ - & \vec{a}_m & - \end{bmatrix} B = \begin{bmatrix} - & \vec{a}_1 B & - \\ - & \vec{a}_2 B & - \\ & \vdots & \\ - & \vec{a}_m B & - \end{bmatrix}$$

   This view shows that every row of C must be in the row space of B.

Identity Property:

$$AI_n = A, \quad I_m A = A$$

where I is the identity matrix of appropriate size.

Note: Optimized algorithms like Strassen's can achieve $O(n^{2.807})$ for square matrices. Modern libraries use cache-optimized blocking and GPU parallelization for practical speedups.
Space Complexity: $O(mn)$ — *Time Complexity*: $O(mnk)$ (naive algorithm), $O(n^3)$ (for $n \times n$ matrices).

2.6 Properties of Matrix Operations

Addition and Scalar Multiplication Properties: Let $A, B, C \in \mathbb{R}^{m\times n}$ and $c, d \in \mathbb{R}$. Then:

$$
\begin{aligned}
&(1) \quad A + B = B + A && \text{(Commutativity)} \\
&(2) \quad (A + B) + C = A + (B + C) && \text{(Associativity)} \\
&(3) \quad A + O = A && \text{(Additive Identity)} \\
&(4) \quad A + (-A) = O && \text{(Additive Inverse)} \\
&(5) \quad c(A + B) = cA + cB && \text{(Distributivity over Matrices)} \\
&(6) \quad (c + d)A = cA + dA && \text{(Distributivity over Scalars)} \\
&(7) \quad c(dA) = (cd)A && \text{(Associativity)} \\
&(8) \quad 1 \cdot A = A && \text{(Scalar Identity)}
\end{aligned}
$$

Multiplication Properties: Let $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{p\times q}$, and $c \in \mathbb{R}$. Then:

$$
\begin{aligned}
&(1) \quad (AB)C = A(BC) && \text{(Associativity)} \\
&(2) \quad A(B + C) = AB + AC && \text{(Left Distributivity)} \\
&(3) \quad (A + B)C = AC + BC && \text{(Right DistributDivity)} \\
&(4) \quad c(AB) = (cA)B = A(cB) && \text{(Scalar Associativity)} \\
&(5) \quad AI_n = A \text{ and } I_m A = A && \text{(Multiplicative Identity)} \\
&(6) \quad AO = O = OA && \text{(Zero Product)}
\end{aligned}
$$

Warning: Matrix multiplication is **not commutative** in general.

$$AB \neq BA$$

Time: $O(mn)$ for addition/scalar multiplication, $O(mnp)$ for multiplication — Space: Result size.

Computational Note:

- **Associativity of Multiplication:** $O(mnk+mkl)$ vs $O(nkl+mnl)$. While $(AB)C$ and $A(BC)$ are mathematically equal, their computational cost can be vastly different. Choosing the optimal order is a key optimization problem (e.g., in deep learning backpropagation).
- **Complexity:** Verifying additive properties is $O(mn)$. Verifying multiplicative properties is $O(n^3)$ (for $n \times n$ matrices).

2.7 Matrix Transpose

The **transpose** of a matrix $A \in \mathbb{R}^{m\times n}$ is the matrix $A^T \in \mathbb{R}^{n\times m}$ obtained by interchanging rows and columns:

$$(A^T)_{ij} = a_{ji}$$

$$A = \begin{bmatrix} - & \vec{r}_1 & - \\ - & \vec{r}_2 & - \\ & \vdots & \end{bmatrix} \implies A^T = \begin{bmatrix} | & | & & | \\ \vec{r}_1^T & \vec{r}_2^T & \cdots & \\ | & | & & | \end{bmatrix}.$$

That is, the i-th row of A becomes the i-th column of A^T.

For vectors: A column vector becomes a row vector, and vice versa:

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \quad \implies \quad \vec{v}^T = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$$

This is why $\vec{u} \cdot \vec{v} = \vec{u}^T \vec{v}$ (dot product as matrix multiplication).
Special Cases:

$$\begin{aligned}
&(1) \quad A = A^T \iff A && \text{(Symmetric)} \\
&(2) \quad A = -A^T \iff A && \text{(Skew-symmetric)} \\
&(3) \quad (A^T)^T = A && \text{(Double transpose)} \\
&(4) \quad \vec{u} \cdot \vec{v} = \vec{u}^T \vec{v} && \text{(Dot Product as Matrix Multiplication)}
\end{aligned}$$

Space Complexity: $O(mn)$ — *Time Complexity*: $O(mn)$.

2.8 Properties of Transpose

Let A, B be matrices of compatible dimensions and $c \in \mathbb{R}$. Then:

$$\begin{aligned}
&(1) \quad (A^T)^T = A && \text{(Double Transpose/Involution)} \\
&(2) \quad (A+B)^T = A^T + B^T && \text{(Additivity)} \\
&(3) \quad (cA)^T = cA^T && \text{(Scalar Multiplication)} \\
&(4) \quad (AB)^T = B^T A^T && \text{(Reversal Rule)} \\
&(5) \quad (A_1 A_2 \cdots A_k)^T = A_k^T \cdots A_2^T A_1^T && \text{(Extended Reversal)} \\
&(6) \quad (\vec{u}^T \vec{v})^T = \vec{v}^T \vec{u} = \vec{u}^T \vec{v} && \text{(Dot Product Symmetry)}
\end{aligned}$$

Transpose of the Identity Matrix:

$$I^T = I$$

Transpose Preserves Rank:

$$\operatorname{rank}(A^T) = \operatorname{rank}(A)$$

Transpose and Inverse: If A is invertible, then A^T is also invertible, and:

$$(A^T)^{-1} = (A^{-1})^T$$

2.9 Symmetric and Skew-Symmetric Matrices

These are special types of square matrices defined by their relationship with their own transpose.

Symmetric Matrix: A square matrix $A \in \mathbb{R}^{n\times n}$ is **symmetric** if it is equal to its transpose.

$$A \text{ is symmetric} \iff A = A^T \iff a_{ij} = a_{ji} \quad \forall i, j$$

Skew-Symmetric Matrix: A square matrix $A \in \mathbb{R}^{n\times n}$ is **skew-symmetric** (or **anti-symmetric**) if it is equal to the negative of its transpose

$$A \text{ is skew-symmetric} \iff A = -A^T \iff a_{ij} = -a_{ji} \quad \forall i, j$$

Note: This implies $a_{ii} = 0$ (diagonal entries must be zero).

Decomposition Theorem: Any square matrix $A \in \mathbb{R}^{n\times n}$ can be **uniquely** decomposed as:

$$A = \underbrace{\frac{A + A^T}{2}}_{\text{Symmetric part } S} + \underbrace{\frac{A - A^T}{2}}_{\text{Skew-symmetric part } K}$$

Key Properties:

1. A symmetric matrix must be square
2. A skew-symmetric matrix must have zeros on the diagonal
3. $A^T A$ is always symmetric (Gram matrix)
4. AA^T is always symmetric (Covariance structure)
5. $\vec{x}^T A\vec{x} = \vec{x}^T A^T \vec{x}$ for symmetric A (Quadratic form)
6. Any square matrix A can be uniquely expressed as the sum of a symmetric matrix S and a skew-symmetric matrix K:

$$A = S + K$$

7. If A and B are symmetric and commute ($AB = BA$), then AB is symmetric
8. The eigenvalues of a real symmetric matrix are real
9. The eigenvalues of a real skew-symmetric matrix are purely imaginary or zero

Tip: Symmetric = Scaling, Skew-symmetric = Rotation.

Special Property for Skew-Symmetric:

$$\det(K) = 0 \text{ if } n \text{ is odd}, \quad \det(K) \geq 0 \text{ if } n \text{ is even}$$

Space Complexity: $\frac{n(n+1)}{2}$ — *Time Complexity*: $O(n^2)$.

2.10 Trace of a Matrix

The **trace** of a square matrix $A \in \mathbb{R}^{n\times n}$, denoted $\mathrm{tr}(A)$, is the **sum of its diagonal entries**:

$$\mathrm{tr}(A) = \sum_{i=1}^{n} a_{ii} = a_{11} + a_{22} + \cdots + a_{nn}$$

Example:

$$A = \begin{bmatrix} \mathbf{1} & 2 & 3 \\ 4 & \mathbf{5} & 6 \\ 7 & 8 & \mathbf{9} \end{bmatrix} \implies \mathrm{tr}(A) = 1 + 5 + 9 = 15$$

Important Note: Trace is **only defined for square matrices**. For non-square matrices, the concept is undefined.

Properties: Let $A, B \in \mathbb{R}^{n\times n}$ and $c \in \mathbb{R}$

(1)	$\mathrm{tr}(A+B) = \mathrm{tr}(A) + \mathrm{tr}(B)$	(Additivity)
(2)	$\mathrm{tr}(cA) = c \cdot \mathrm{tr}(A)$	(Homogeneity)
(3)	$\mathrm{tr}(A^T) = \mathrm{tr}(A)$	(Transpose Invariance)
(4)	$\mathrm{tr}(AB) = \mathrm{tr}(BA)$	(Cyclic Property)
(5)	$\mathrm{tr}(ABC) = \mathrm{tr}(CAB) = \mathrm{tr}(BCA)$	(Extended Cyclic)
(6)	$\mathrm{tr}(I_n) = n$	(Identity Trace)
(7)	$\mathrm{tr}(\vec{u}\vec{v}^T) = \vec{u}^T\vec{v} = \vec{v}^T\vec{u}$	(Outer Product)

Properties (1) and (2) show that trace is a **linear functional** (or linear operator) on the space of matrices.
Note on (4): Even though $AB \neq BA$ in general, their traces are equal.

Special Identities:

1. $\mathrm{tr}(A^TA) = \mathrm{tr}(AA^T) = \sum_{i,j} a_{ij}^2 = \|A\|_F^2$ (Frobenius norm squared/Matrix magnitude)
2. $\mathrm{tr}(\vec{x}\vec{x}^T) = \vec{x}^T\vec{x} = \|\vec{x}\|^2$ (Vector norm)

Relationship to Eigenvalues: For a matrix A with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$:

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

This is a fundamental result connecting diagonal entries to spectral properties (covered in detail in eigenvalue section).

Space Complexity: $O(1)$ — *Time Complexity*: $O(n)$.

2.11 Relationship Between Trace and Eigenvalues

For any square matrix $A \in \mathbb{R}^{n\times n}$, the **trace** of A is equal to the **sum of all its eigenvalues** $(\lambda_1, \lambda_2, \ldots, \lambda_n)$:

$$\operatorname{tr}(A) = \sum_{i=1}^{n} \lambda_i = \lambda_1 + \lambda_2 + \cdots + \lambda_n$$

This holds **regardless of whether A is diagonalizable**.
Notes:

- The eigenvalues λ_i are the roots of the characteristic polynomial of A.
- This property holds even if the eigenvalues are repeated (counted with their algebraic multiplicity) or are complex numbers (for $A \in \mathbb{C}^{n\times n}$).

Companion Result (Determinant):

$$\det(A) = \prod_{i=1}^{n} \lambda_i = \lambda_1 \cdot \lambda_2 \cdots \lambda_n$$

Together, these provide powerful invariants of a matrix under similarity transformations.
Proof via Characteristic Polynomial:
The characteristic polynomial of A is:

$$p(\lambda) = \det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0$$

By the fundamental theorem of algebra, this factors as:

$$p(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2)\cdots(\lambda - \lambda_n)$$

Expanding and comparing coefficients:

$$\text{Coefficient of } \lambda^{n-1}: \quad c_{n-1} = -(\lambda_1 + \lambda_2 + \cdots + \lambda_n)$$

$$\text{Constant term}: \quad c_0 = (-1)^n \lambda_1\lambda_2\cdots\lambda_n = (-1)^n \det(A)$$

From the definition of characteristic polynomial, $c_{n-1} = -\operatorname{tr}(A)$, thus:

$$\operatorname{tr}(A) = \sum_{i=1}^{n} \lambda_i$$

Key Properties:

$$\begin{aligned}
&(1) \quad \mathrm{tr}(A^k) = \sum_{i=1}^{n} \lambda_i^k && \text{(Powers of eigenvalues)} \\
&(2) \quad \mathrm{tr}(P^{-1}AP) = \mathrm{tr}(A) && \text{(Similarity invariance)} \\
&(3) \quad \mathrm{tr}(e^A) = \sum_{i=1}^{n} e^{\lambda_i} && \text{(Matrix exponential)} \\
&(4) \quad \mathrm{tr}(A^{-1}) = \sum_{i=1}^{n} \lambda_i^{-1} && \text{(Inverse, if exists)} \\
&(5) \quad \mathrm{tr}(A) = 0 \not\Rightarrow \lambda_i = 0 && \text{(Zero trace doesn't imply zero eigenvalues)}
\end{aligned}$$

Property (2) is crucial: similar matrices (representing the same linear transformation in different bases) have the same trace and eigenvalues.
Example:

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Eigenvalues: $\lambda_1 = 3, \lambda_2 = 1$ (verify: $\det(\lambda I - A) = (\lambda - 3)(\lambda - 1) = 0$)

$$\mathrm{tr}(A) = 2 + 2 = 4 = 3 + 1 = \lambda_1 + \lambda_2 \quad \checkmark$$

$$\det(A) = 4 - 1 = 3 = 3 \cdot 1 = \lambda_1 \lambda_2 \quad \checkmark$$

Geometric Interpretation:

- **Trace:** Measures total "stretching" along all eigendirections. Sum of scaling factors.
- **Determinant:** Measures volume distortion. Product of scaling factors.
- For rotation matrices (orthogonal), eigenvalues lie on unit circle: $|\lambda_i| = 1$, so $\mathrm{tr}(Q) \leq n$ with equality for identity.

Special Cases:

$$\begin{aligned}
&\text{Diagonal matrix:} \quad D = \mathrm{diag}(d_1, \ldots, d_n) \implies \lambda_i = d_i, \quad \mathrm{tr}(D) = \sum d_i \\
&\text{Triangular matrix:} \quad \text{Eigenvalues are diagonal entries} \\
&\text{Symmetric matrix:} \quad \text{All eigenvalues real, orthogonal eigenvectors} \\
&\text{Orthogonal matrix:} \quad |\lambda_i| = 1, \quad \mathrm{tr}(Q) = \sum \lambda_i \text{ (complex sum)}
\end{aligned}$$

2.12 Powers of Matrices

For a square matrix $A \in \mathbb{R}^{n \times n}$ and a non-negative integer k, the **power** A^k is defined as the matrix A multiplied by itself k times:

Definition:

$$A^k = \underbrace{A \cdot A \cdot \ldots \cdot A}_{k \text{ times}}$$

Recursive Definition:

$$A^k = A \cdot A^{k-1} \quad \text{for } k \geq 1$$

Requirement: Powers are only defined for **square matrices**, as the dimensions must be compatible for repeated multiplication ($n \times n$ times $n \times n$).

Special Cases:

$$\begin{aligned}
&(1) \quad A^0 = I && \text{(Identity, by convention)} \\
&(2) \quad A^1 = A && \text{(First power)} \\
&(3) \quad A^{-k} = (A^{-1})^k = (A^k)^{-1} && \text{(Negative powers, if } A \text{ invertible)} \\
&(4) \quad (cA)^k = c^k A^k && \text{(Scalar multiplication)}
\end{aligned}$$

Fundamental Properties:

$$\begin{aligned}
&(1) \quad A^m A^n = A^{m+n} && \text{(Exponent addition)} \\
&(2) \quad (A^m)^n = A^{mn} && \text{(Exponent multiplication)} \\
&(3) \quad (A^T)^k = (A^k)^T && \text{(Transpose commutes)} \\
&(4) \quad (A^{-1})^k = (A^k)^{-1} && \text{(Inverse commutes)} \\
&(5) \quad AB = BA \implies (AB)^k = A^k B^k && \text{(Commuting matrices only)} \\
&(6) \quad \text{tr}(A^k) = \sum_{i=1}^{n} \lambda_i^k && \text{(Trace of power)}
\end{aligned}$$

Warning: In general, $(AB)^k \neq A^k B^k$ unless $AB = BA$.

Computational Methods:

1. **Naive Repeated Multiplication:**

$$A^k = A \cdot A \cdot A \cdots A \quad (k - 1 \text{ multiplications}) - O(kn^3)$$

2. **Binary Exponentiation (Repeated Squaring):**

$$A^{13} = A^8 \cdot A^4 \cdot A^1 - O(n^3 \log k)$$

3. **Diagonalization Method (Best for multiple powers):**

$$A^k = (PDP^{-1})^k = PD^k P^{-1} - O(n^3)$$

Space Complexity: $O(n^2)$ — *Time Complexity*: $O(kn^3)$ | $O(n^3 \log k)$ | $O(n^3)$.

2.13 Elementary Matrices

An **elementary matrix** (E) is a matrix is obtained by performing a **single elementary row operation** on the identity matrix (I).

Three Types of Elementary Matrices:

1. **Type I — Row Swap:** E_{ij} swaps rows i and j

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \xrightarrow{R_1 \leftrightarrow R_2} E_{12} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{(swaps rows 1 and 2)}$$

2. **Type II — Row Scaling:** $E_i(c)$ multiplies row i by non-zero scalar c

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \xrightarrow{5R_1 \to R_1} E_2(5) = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix} \quad \text{(scales row 2 by 5)}$$

3. **Type III — Row Addition:** $E_{ij}(c)$ adds c times row j to row i

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \xrightarrow{R_2 + 4R_1 \to R_2} E_{21}(4) = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix} \quad \text{(adds 4 times row 1 to row 2)}$$

Key Principle: Performing elementary row operation on matrix A is equivalent to **left-multiplying** A by the corresponding elementary matrix:

$$\text{Row operation on } A \iff EA$$

where E is the elementary matrix representing that operation.
Fundamental Properties:

$$\begin{aligned}
&(1) \quad \text{Every elementary matrix is } \textbf{invertible} \\
&(2) \quad \text{The inverse of an elementary matrix is also elementary} \\
&(3) \quad \det(E) \neq 0 \text{ for all elementary matrices} && \text{(Always invertible)} \\
&(4) \quad (EA)^T = A^T E^T && \text{(Transpose relation)}
\end{aligned}$$

Inverses of Elementary Matrices:

$$\begin{aligned}
&\text{Type I:} \quad E_{ij}^{-1} = E_{ij} && \text{(Self-inverse: swap twice returns original)} \\
&\text{Type II:} \quad E_i(c)^{-1} = E_i(1/c) && \text{(Scale by reciprocal)} \\
&\text{Type III:} \quad E_{ij}(c)^{-1} = E_{ij}(-c) && \text{(Negate the coefficient)}
\end{aligned}$$

Determinants of Elementary Matrices:

$$\begin{aligned}
&\text{Type I:} \quad \det(E_{ij}) = -1 && \text{(Row swap changes sign)} \\
&\text{Type II:} \quad \det(E_i(c)) = c && \text{(Scaling factor)} \\
&\text{Type III:} \quad \det(E_{ij}(c)) = 1 && \text{(No change)}
\end{aligned}$$

These correspond exactly to how elementary row operations affect determinants!

Geometric Interpretation:

- **Type I:** Reflection across a hyperplane (swaps two basis vectors)
- **Type II:** Scaling along one coordinate axis
- **Type III:** Shear transformation (preserves volume, hence det = 1)

Every invertible linear transformation can be decomposed into these three basic operations!

2.14 Block Matrices

A **block matrix** (or **partitioned matrix**) is a matrix divided into submatrices called **blocks**:

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

where $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times r}$, $C \in \mathbb{R}^{s\times q}$, $D \in \mathbb{R}^{s\times r}$, and $M \in \mathbb{R}^{(p+s)\times(q+r)}$.
General Form:

$$M = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix}$$

where each A_{ij} is a submatrix (block).
Block Matrix Operations:
Operations on block matrices follow the same rules as scalar matrices, **provided dimensions are compatible**.

1. **Block Addition:**

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} + \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} A+E & B+F \\ C+G & D+H \end{bmatrix}$$

   Requirement: Corresponding blocks must have same dimensions.

2. **Block Scalar Multiplication:**

$$c\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} cA & cB \\ cC & cD \end{bmatrix}$$

3. **Block Matrix Multiplication:**

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}\begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{bmatrix}$$

   Requirement: Inner dimensions must match ($A \in \mathbb{R}^{p\times q}$, $E \in \mathbb{R}^{q\times r}$, etc.).

   This follows the same "row-times-column" pattern as scalar matrix multiplication!

4. **Block Transpose:**

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^T = \begin{bmatrix} A^T & C^T \\ B^T & D^T \end{bmatrix}$$

   Blocks are transposed **and** positions are transposed.