

Matrices

[Read this note online](#)

Matrices in linear algebra are rectangular arrays of numbers that encode linear transformations, systems of equations, datasets, graphs, and more. They're the central computational object of linear algebra and the workhorse of nearly every algorithm in data science, machine learning, computer graphics, and engineering.

A matrix is two things at once. Computationally, it's a grid of numbers you can add, multiply, and invert. Conceptually, it's a linear function, a rule that takes vectors as input and produces vectors as output. Holding both views simultaneously is the mental shift that turns matrix algebra from rote calculation into structural thinking.

This study note covers matrix notation, addition and scalar multiplication, matrix multiplication and its non-commutativity, the identity and inverse matrices, the transpose, **determinants** and their geometric meaning, the connection to systems of linear equations, applications, and common pitfalls.

Matrix dimensions determine which products exist. The order of multiplication matters because the operations can represent different successive transformations.

MATRICES

1. MATRIX

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad \begin{array}{l} A \text{ is a } 3 \times 3 \text{ matrix.} \\ a_{ij} \text{ is the entry in} \\ \text{row } i, \text{ column } j. \end{array}$$

3. IDENTITY MATRIX

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{array}{l} I \text{ is the } 3 \times 3 \text{ identity matrix.} \\ AI = IA = A \text{ for any } 3 \times 3 \\ \text{matrix } A. \end{array}$$

4. INVERSE MATRIX

If A is invertible, its inverse is denoted A^{-1} .
It satisfies:

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

A^{-1} exists only if A is nonsingular ($\det(A) \neq 0$).

2. MATRIX MULTIPLICATION

Let A, B be 3×3 matrices. Their product $C = AB$ is the 3×3 matrix with entries:

$$A \times B = C$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

Entry c_{ij} (row i of A with column j of B):

$$\begin{array}{c} \text{row } i \\ \text{of } A \rightarrow \end{array} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{i1} & a_{i2} & a_{i3} \\ \vdots & \vdots & \vdots \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{array}{c} \text{column } j \\ \text{of } B \downarrow \end{array} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = c_{ij}$$

$$c_{ij} = \sum_{k=1}^3 a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + a_{i3} b_{3j}$$

5. TRANSPOSE

The transpose of A , denoted A^T , is obtained by interchanging rows and columns.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

SUMMARY

- $A, B, C \in \mathbb{R}^{3 \times 3}$
- $(AB) \neq (BA)$ in general
- $(A^T)^T = A$
- $(AB)^T = B^T A^T$
- $(A^{-1})^T = (A^T)^{-1}$

Matrices in linear algebra multiplication identity inverse modern textbook illustration

Notation and Dimensions

A matrix is denoted with a capital letter and its entries with subscripts: A_{ij} is the entry in row i , column j . An $m \times n$ matrix has m rows and n columns:

Square matrices have $m = n$. Row **vectors** are $1 \times n$; column vectors are $m \times 1$. The dimensions matter for every operation, addition requires identical dimensions; multiplication requires the inner dimensions to match.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Matrix Addition and Scalar Multiplication

Matrices of the same dimension add entry-wise:

Scalar multiplication scales every entry:

Both operations are commutative and associative. The zero matrix (all entries zero) is the additive identity. Together with these operations, the set of $m \times n$ matrices forms a vector space.

$$(A + B)_{ij} = A_{ij} + B_{ij}$$

$$(cA)_{ij} = c \cdot A_{ij}$$

Matrix Multiplication

Matrix multiplication is the most important matrix operation. To multiply A ($m \times n$) by B ($n \times p$), the product AB is an $m \times p$ matrix with entries:

Row i of A dotted with column j of B gives entry (i, j) of AB . The inner dimensions must match (the number of columns of A equals the number of rows of B); the outer dimensions become the dimensions of the product.

Matrix multiplication is associative: $(AB)C = A(BC)$. It's distributive over addition: $A(B + C) = AB + AC$. It is **not commutative**: in general, $AB \neq BA$. This non-commutativity is the source of much of linear algebra's depth and most of its initial confusion for beginners.

$$(AB)_{ij} = \sum_{k=1}^n A_{ik}B_{kj}$$

Why Matrix Multiplication Is Defined That Way

The "row times column" rule looks arbitrary but encodes function composition. Each matrix represents a linear transformation. The product AB represents the composition: first apply B , then apply A . The strange-looking formula is exactly what makes $(AB)\mathbf{x} = A(B\mathbf{x})$ hold for all input vectors $\mathbf{x}$.

This is also why the order matters. Composing transformations in the order "first B , then A " generally produces a different result from "first A , then B ." Even commuting transformations (rotations about a fixed axis, scalings) don't generalize: arbitrary linear maps don't commute. This non-commutativity is what makes linear algebra interesting.

The Identity Matrix

The identity matrix I_n is the $n \times n$ square matrix with 1s on the diagonal and 0s elsewhere:

The identity satisfies $IA = AI = A$ for any compatible matrix A . It's the multiplicative identity for

square matrices. Geometrically, the identity matrix represents the "do nothing" linear transformation: it leaves every vector unchanged.

The identity matrix appears constantly in proofs, inverse calculations, and matrix decompositions. It's also the natural starting point for many iterative algorithms (start with the identity and modify from there).

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

The Inverse Matrix

For a square matrix A , the inverse A^{-1} is the matrix satisfying $AA^{-1} = A^{-1}A = I$. Not every square matrix has an inverse, those that do are called *invertible* or *nonsingular*; those that don't are *singular*.

For a 2×2 matrix:

For larger matrices, computing the inverse uses Gauss-Jordan elimination, LU decomposition, or other numerical algorithms. Inverses solve the matrix equation $A\mathbf{x} = \mathbf{b}$: $\mathbf{x} = A^{-1}\mathbf{b}$, but in practice, never compute the inverse explicitly when you can solve the system directly via elimination, which is faster and more numerically stable.

$$A^{-1} = \frac{1}{\det A} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \text{ where } A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

The Transpose

The transpose A^T swaps rows and columns: $(A^T)_{ij} = A_{ji}$. An $m \times n$ matrix becomes $n \times m$ when transposed.

Properties: $(A^T)^T = A$, $(A + B)^T = A^T + B^T$, $(AB)^T = B^T A^T$ (note the order reversal), $(cA)^T = cA^T$.

A matrix is *symmetric* if $A^T = A$; it's *skew-symmetric* if $A^T = -A$. Symmetric matrices are central to many applications: covariance matrices in statistics, Hessian matrices in optimization, adjacency matrices of undirected graphs, stiffness matrices in finite element analysis. Symmetric matrices have especially clean spectral properties, all eigenvalues are real and orthonormal eigenvector bases always exist.

Determinants

The determinant $\det(A)$ of a square matrix is a scalar that measures how the matrix scales volumes (or signed volumes). For a 2×2 matrix:

Geometrically, this equals the signed area of the parallelogram with sides given by the matrix's column vectors. In 3D, the determinant equals the signed volume of the parallelepiped spanned by the columns.

Key properties: $\det(AB) = \det(A)\det(B)$, $\det(A^T) = \det(A)$, $\det(A^{-1}) = 1/\det(A)$. The matrix is invertible if and only if its determinant is nonzero. A determinant of zero means the matrix collapses some volume to zero, it has linearly dependent columns and is therefore singular.

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

Systems of Linear Equations

The matrix equation $A\mathbf{x} = \mathbf{b}$ represents a system of linear equations. The matrix A holds the coefficients; the vector $\mathbf{x}$ holds the unknowns; $\mathbf{b}$ holds the right-hand-side constants.

Three possibilities:

This trichotomy is the foundation of solving linear systems and underpins regression, optimization, and most of numerical linear algebra.

Unique solution: A is invertible (square and nonsingular). Then $\mathbf{x} = A^{-1}\mathbf{b}$, or compute via Gauss-Jordan elimination. **No solution:** the system is inconsistent (the vector $\mathbf{b}$ is not in the column space of A). **Infinitely many solutions:** the system is consistent but underdetermined; the solution set is a flat affine subspace.

Special Types of Matrices

Diagonal: nonzero entries only on the main diagonal. Easy to multiply, invert, and exponentiate. **Triangular:** upper or lower triangular. Solving systems with triangular matrices is fast (forward or back substitution), the basis for LU decomposition. **Symmetric:** $A^T = A$. Real eigenvalues, orthonormal eigenvectors. Central in statistics and optimization. **Orthogonal:** $Q^T Q = I$. Preserves lengths and angles. Rotations and reflections are orthogonal matrices. **Sparse:** mostly zero entries. Storage and computation use special algorithms; central in scientific computing and machine learning at scale. **Identity, Zero, Permutation, Hermitian (complex symmetric), Unitary (complex orthogonal), Positive definite:** each carries useful structure for specific applications.

Matrix Decompositions

Many matrix algorithms work by decomposing a matrix into simpler factors. Common decompositions:

Decompositions reveal structure and enable efficient algorithms. Modern numerical linear algebra is largely the engineering of robust decomposition algorithms for huge matrices.

LU: $A = LU$ where L is lower triangular and U is upper triangular. Used for solving linear systems efficiently. **QR:** $A = QR$ where Q is orthogonal and R is upper triangular. Used for least-squares regression. **Cholesky:** $A = LL^T$ for symmetric positive-definite matrices. Faster than LU for these matrices. **Eigendecomposition:** $A = V\Lambda V^{-1}$ where columns of V are eigenvectors and Λ is diagonal of eigenvalues. **Singular Value Decomposition (SVD):** $A = U\Sigma V^T$, the most general decomposition; underpins PCA, low-rank approximation, and recommender systems.

Worked Example: Matrix Multiplication

Multiply $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ and $B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$:

Now BA :

$AB \neq BA$. The non-commutativity is concrete: same matrices, different order, completely different products.

$$AB = \begin{pmatrix} 1 \cdot 5 + 2 \cdot 7 & 1 \cdot 6 + 2 \cdot 8 \\ 3 \cdot 5 + 4 \cdot 7 & 3 \cdot 6 + 4 \cdot 8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

$$BA = \begin{pmatrix} 5 \cdot 1 + 6 \cdot 3 & 5 \cdot 2 + 6 \cdot 4 \\ 7 \cdot 1 + 8 \cdot 3 & 7 \cdot 2 + 8 \cdot 4 \end{pmatrix} = \begin{pmatrix} 23 & 34 \\ 31 & 46 \end{pmatrix}$$

Where Matrices Show Up in Practice

Machine learning: data is stored as matrices (rows = examples, columns = features). Neural network weights are matrices. Convolution operations multiply matrices behind the scenes. Read more in [best machine learning courses](#). **Computer graphics:** 3D transformations (translation, rotation, scaling) are 4×4 matrix operations applied to homogeneous coordinates. **Statistics:** covariance matrices, regression coefficient matrices, correlation matrices, Hat matrices in linear regression. **Engineering:** stiffness matrices in finite element analysis, transfer matrices in control systems, scattering matrices in optics and electromagnetics. **Network analysis:** adjacency ma-

trices of graphs, Laplacian matrices for spectral methods, transition matrices for Markov chains. **Quantum mechanics:** state vectors transformed by unitary matrices; observables represented by Hermitian matrices.

Common Mistakes with Matrices

Assuming $AB = BA$. Matrix multiplication is generally not commutative. Always preserve order. **Mismatched dimensions.** The product AB requires the number of columns of A to equal the number of rows of B . Software gives explicit errors; mental computation often skips this check. **Computing the inverse when you don't need to.** Solving $A\mathbf{x} = \mathbf{b}$ by computing A^{-1} and multiplying is slower and less stable than direct elimination. **Confusing inverse with transpose.** A^T and A^{-1} are different operations. They coincide only for orthogonal matrices. **Treating singular matrices as if they were invertible.** A determinant of zero means no inverse exists. Always check determinant or rank before assuming invertibility.

A Brief History of Matrices

The Chinese mathematical text *The Nine Chapters on the Mathematical Art* (~200 BCE) describes solving systems of linear equations using array manipulations equivalent to Gaussian elimination, over two thousand years before the West formalized the technique.

The term "matrix" was coined by James Joseph Sylvester in 1850. Arthur Cayley developed matrix algebra in the 1850s and 1860s, including matrix multiplication, the inverse, and the characteristic polynomial. By the early 20th century, matrices had become central to physics (quantum mechanics is matrix mechanics) and engineering.

The modern era of matrix algorithms began with John von Neumann and Herman Goldstine in the 1940s as they developed numerical methods for early computers. LAPACK (Linear Algebra PACKage), released in 1992 and still maintained, is the canonical reference implementation for dense matrix algorithms and underpins NumPy, MATLAB, R, Julia, and most scientific computing software in use today.

Matrix Conditioning and Numerical Stability

The condition number of a matrix measures how sensitive its solutions are to perturbations in the input. A well-conditioned matrix produces stable solutions; an ill-conditioned matrix amplifies tiny input errors into huge output errors.

Formally, the condition number is $\kappa(A) = \|A\| \cdot \|A^{-1}\|$. Large condition numbers (above 10^6 or so) signal numerical trouble. Solutions to ill-conditioned systems may be wildly wrong even with small

floating-point errors.

Diagnosing and remedying ill-conditioning is a daily concern in scientific computing, machine learning (poorly scaled features create ill-conditioned design matrices), and engineering simulation. Regularization, preconditioning, and feature scaling all address this.

Sparse Matrices and Their Algorithms

A sparse matrix has mostly zero entries. Storing only the nonzero entries (and their positions) saves enormous memory. Algorithms designed for sparse matrices skip the zero entries entirely, achieving massive speedups for the same problem.

Compressed sparse row (CSR) and compressed sparse column (CSC) are the standard storage formats. Iterative solvers (conjugate gradient, GMRES) and sparse factorization methods (sparse LU, sparse Cholesky) handle large sparse systems that would be impossible to solve with dense algorithms.

Sparse matrices appear constantly in finite element analysis, large-scale optimization, web graphs (PageRank's link matrix is sparse), and recommender systems (user-item matrices are mostly empty). Modern scientific computing leans heavily on sparse linear algebra.

Tensors as Generalizations of Matrices

Tensors generalize matrices to multi-index arrays. A scalar is a rank-0 tensor; a vector is rank-1; a matrix is rank-2; a 3D array of numbers is rank-3; and so on. Tensors are the natural data structure for physical quantities (stress, strain, electromagnetic field), multi-dimensional signal processing, and modern deep learning (input batches are 4D tensors: batch $\times$ channels $\times$ height $\times$ width).

Tensor operations generalize matrix operations. Tensor contraction generalizes matrix multiplication. Einstein notation provides a compact way to express complex tensor operations. PyTorch, TensorFlow, and JAX are essentially tensor manipulation libraries; the "matrix" in matrix algorithms is just a convenient 2D special case of the underlying tensor framework.

Block Matrices and Block Operations

Large matrices can be partitioned into blocks (sub-matrices), and many operations work block-wise. Block multiplication mirrors scalar multiplication:

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

when block dimensions match.

Block matrices appear constantly in numerical algorithms. Schur complements decompose block-

structured matrices into smaller solves. Strassen's algorithm uses block matrix tricks to multiply matrices faster than the naive $O(n^3)$ bound. Modern dense matrix libraries (BLAS Level 3) work primarily on blocks rather than individual entries.

The related [OpenStax mathematics textbooks](#) provide additional explanations and exercises for the mathematical definitions used here.

A Product with Dimensions

Let $A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$ and $B = \begin{pmatrix} 2 & 0 \\ 3 & 1 \end{pmatrix}$. Multiplication gives $AB = \begin{pmatrix} 8 & 2 \\ 3 & 1 \end{pmatrix}$, while $BA = \begin{pmatrix} 2 & 4 \\ 3 & 7 \end{pmatrix}$. They are different.

Using column vectors, the transformation AB applies B first and A afterward. The rightmost matrix acts first. For a general $m \times n$ matrix multiplying an $n \times p$ matrix, the result has $m \times p$ dimensions. A shape check catches an invalid operation before its entries are calculated.

The points to check in this example are:

- Matching inner dimensions allow a matrix product.
- Matching whole dimensions allow matrix addition.
- An inverse exists only for a square matrix with nonzero determinant.

Check Your Understanding

Try the questions before reading the answers.

Question 1. What is the size of a 2×3 matrix times a 3×4 matrix?

Answer. 2×4 .

Question 2. Must AB equal BA for square matrices?

Answer. No. The worked example is a counterexample.

FAQs

What is a matrix?

A rectangular array of numbers arranged in rows and columns. Matrices encode linear transformations, systems of equations, datasets, graphs, and more. The dimensions are written as rows $\times$ columns; a 3×4 matrix has 3 rows and 4 columns.

Why isn't matrix multiplication commutative?

Because matrix multiplication represents function composition: AB means 'first apply B, then A.' Composing transformations in different orders generally gives different results. Multiplying a rotation by a translation, for example, is not the same as multiplying the translation by the rotation.

When does a matrix have an inverse?

A square matrix is invertible if and only if its determinant is nonzero (equivalently, its rank equals its dimension, or its rows/columns are linearly independent). Non-square matrices don't have inverses in the standard sense, though pseudoinverses generalize the concept.

What is the identity matrix?

The square matrix with 1s on the main diagonal and 0s elsewhere. It satisfies $IA = AI = A$ for any compatible matrix A . Geometrically, it represents the 'do nothing' transformation that leaves every vector unchanged.

How do I compute the inverse of a matrix?

For 2×2 : swap diagonal entries, negate off-diagonal entries, divide by determinant. For larger matrices: Gauss-Jordan elimination, LU decomposition, or numerical libraries (NumPy `linalg.inv`, MATLAB `inv`). In practice, never compute the explicit inverse to solve $Ax = b$, use direct elimination instead, which is faster and more numerically stable.

What does the determinant tell me?

It measures how the matrix scales volumes (or signed volumes) under the linear transformation. A determinant of zero means the matrix collapses some volume to zero, the columns are linearly dependent and the matrix is singular. Negative determinants indicate orientation reversal.

What's the difference between a row vector and a column vector?

Row vectors are $1 \times n$ matrices; column vectors are $n \times 1$. They behave differently under matrix multiplication: row vectors multiply matrices on the left; column vectors on the right. By convention, vectors in linear algebra default to columns unless context says otherwise.

What is the transpose of a matrix?

The matrix you get by swapping rows and columns: $(A^T)_{ij} = A_{ji}$. An $m \times n$ matrix becomes $n \times m$ when transposed. The transpose appears in dot products, normal equations of regression, and the definition of symmetric and orthogonal matrices.

How are matrices used in machine learning?

Almost everywhere. Datasets are stored as matrices (rows = examples, columns = features). Neural network weights are matrices. Forward passes are matrix multiplications. Backpropagation involves matrix derivatives. Optimization updates are matrix arithmetic. Without matrix algebra, modern ML is impossible.

What is matrix decomposition and why is it useful?

Decomposition factors a matrix into simpler matrices (LU, QR, Cholesky, eigendecomposition, SVD). Each decomposition reveals structure and enables efficient algorithms: LU for solving systems, QR for least-squares, eigendecomposition for spectral analysis, SVD for PCA and low-rank approximation.

What is the rank of a matrix?

The dimension of the column space (equivalently, the row space). It equals the number of linearly independent columns, or the number of nonzero rows after row-reduction. Rank-deficient matrices are singular; full-rank square matrices are invertible.

How do matrices represent linear transformations?

Each matrix encodes a linear function from one vector space to another. Multiplying the matrix by an input vector produces the output vector. The columns of the matrix are the images of the standard basis vectors. Matrix multiplication corresponds to function composition.

What is a singular matrix?

A square matrix whose determinant is zero. Equivalently, its columns (or rows) are linearly dependent, its rank is less than its dimension, and it has no inverse. Singular matrices arise in degenerate systems of equations, near-collinear regression, and ill-conditioned numerical problems.

What's the difference between matrix multiplication and element-wise multiplication?

Matrix multiplication uses the row-times-column rule and represents function composition. Element-wise multiplication (Hadamard product) multiplies corresponding entries. They're different operations with different applications: matrix multiplication for transformations and systems; element-wise for masking, scaling, and per-feature operations in ML.