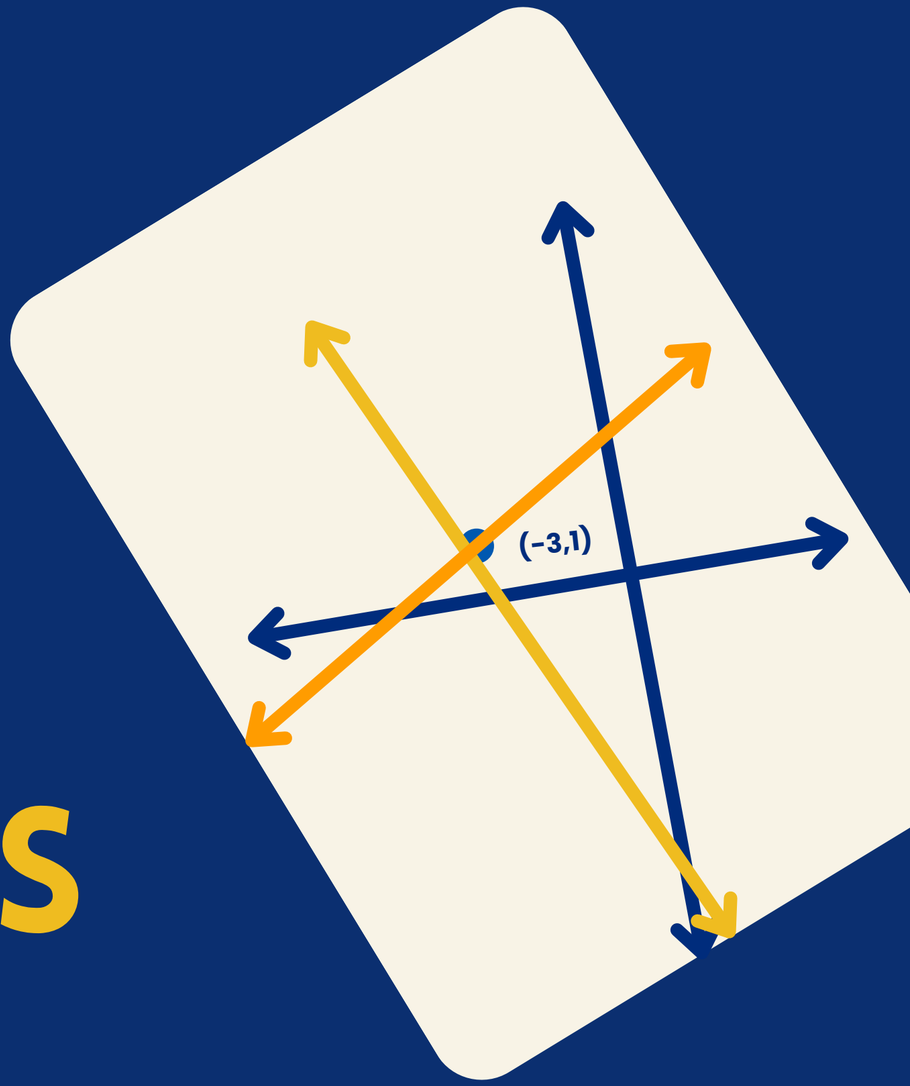
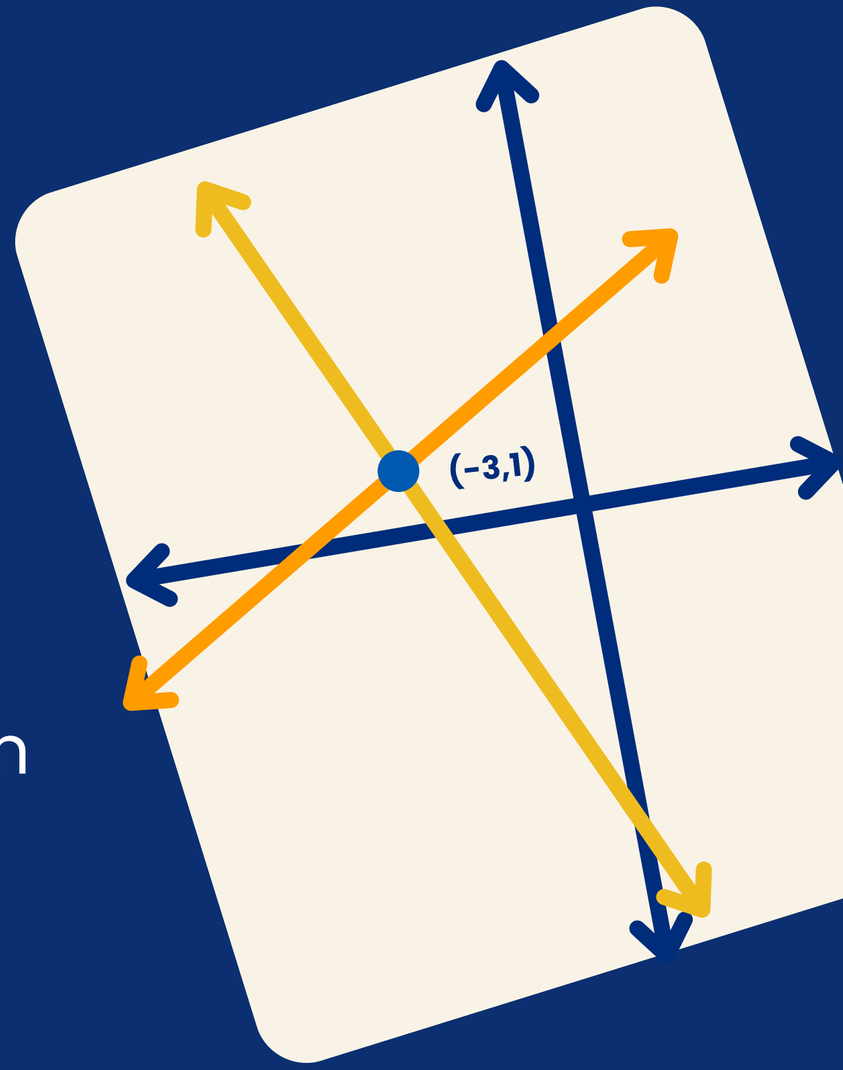
CHAPTER 14.

LINEAR ALGEBRA AND SYSTEMS OF LINEAR EQUATIONS



CHAPTER OUTLINE

1. Basics of Linear Algebra
2. Linear Transformations
3. Systems of Linear Equations
4. Solutions to Systems of Linear Equations
5. Solve Systems of Linear Equations in Python
6. Matrix Inversion
7. Summary and Problems



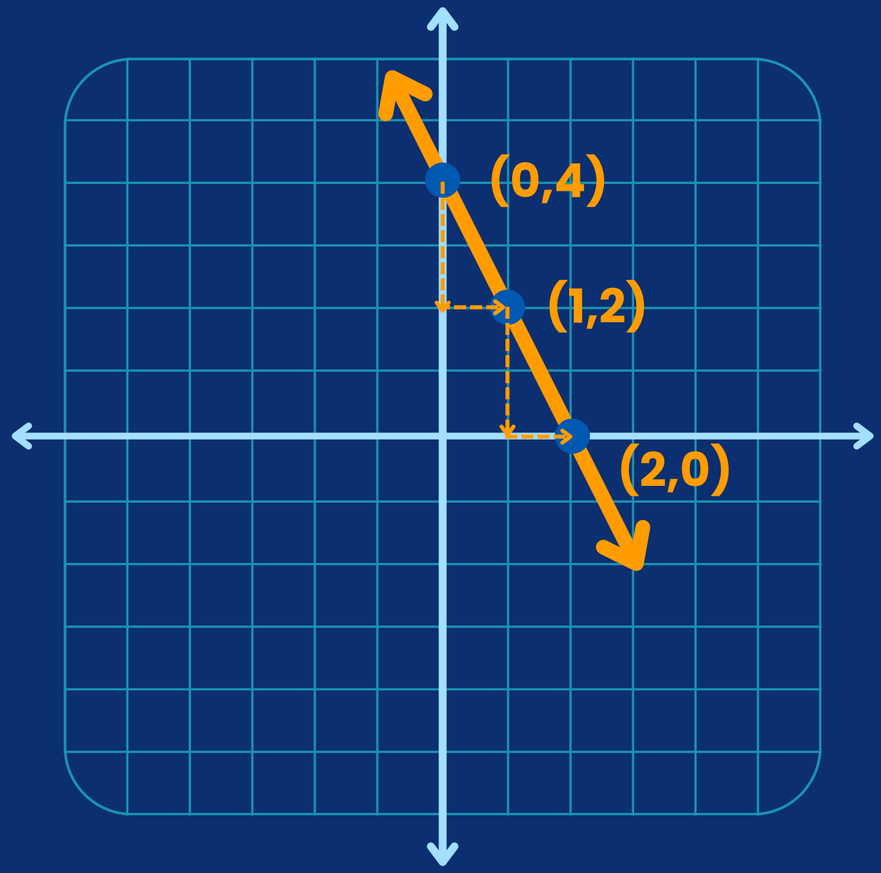
BASICS OF LINEAR ALGEBRA

What is Linear Algebra?

The study of vectors, matrices, and linear transformations between them.

Foundation of many fields: data analysis, engineering, machine learning, physics.

In numerical computation, linear algebra helps solve systems of equations and model transformations efficiently.



BASICS OF LINEAR ALGEBRA

Sets

Definition:

A set is a collection of distinct objects written in braces {}.
Example: S = {orange, apple, banana}

The empty set: {} or ∅.

Operations:

Union: A ∪ B → elements in either A or B.

Intersection: A ∩ B → elements common to A and B.

Set difference: A \ B → elements in A not in B.

Common sets of numbers:

Name	Symbol	Example / Meaning
Naturals	$\mathbb{N}$	{1, 2, 3, ...}
Integers	$\mathbb{Z}$	{..., -2, -1, 0, 1, 2, ...}
Rationals	$\mathbb{Q}$	fractions p/q
Reals	$\mathbb{R}$	all rational + irrational
Complex	$\mathbb{C}$	a + bi

BASICS OF LINEAR ALGEBRA

Vectors

Definition:

A vector is an ordered list of numbers, representing magnitude & direction.
Example:

In $\mathbb{R}^2 \rightarrow$ (x, y)

In $\mathbb{R}^3 \rightarrow$ (x, y, z)

Notation:

Column vector:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Row vector: [1, 2, 3]

Addition:

$$u + v = [u_1 + v_1, u_2 + v_2, \ldots]$$

Scalar Multiplication:

$$a \cdot v = [a \cdot v_1, a \cdot v_2, \ldots]$$

Dot Product:

$$v \cdot w = \Sigma(v_i w_i)$$

Angle Formula:

$$v \cdot w = \|v\| \|w\| cos\theta$$

```
from numpy import arccos, dot
norm = np.linalg.norm
v = np.array([[10, 9, 3]])
w = np.array([[2, 5, 12]])
theta = \
    arccos(dot(v, w.T)/(norm(v)*norm(w)))
print(theta)
```

✓ 0.0s

```
[[0.97992471]]
```

Cross Product:

$\vec{A} =$ [a, b, c] $\quad \vec{B} =$ [d, e, f]

x y z x y
a b c a b
d e f d e

$\vec{A} \times \vec{B} = [(bf - ce), (cd - af), (ae - bd)]$

```
v = np.array([[0, 2, 0]])
w = np.array([[3, 0, 0]])
print(np.cross(v, w))
```

✓ 0.0s

```
[[ 0  0 -6]]
```

BASICS OF LINEAR ALGEBRA

Matrices

Determinant (square matrices only):

2×2: |M| = ad - bc

3×3: aei + bfg + cdh - ceg - bdi - afh

Identity Matrix (I):

Diagonal ones, zeros elsewhere

```
from numpy.linalg import det

M = np.array([[0,2,1,3],
              [3,2,8,1],
              [1,0,0,3],
              [0,3,2,1]])
print('M:\n', M)

print('Determinant: %.1f'%det(M))
I = np.eye(4)
print('I:\n', I)
print('M*I:\n', np.dot(M, I))
```

✓ 0.2s

```
M:
 [[0 2 1 3]
 [3 2 8 1]
 [1 0 0 3]
 [0 3 2 1]]
Determinant: -38.0
I:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
M*I:
 [[0. 2. 1. 3.]
 [3. 2. 8. 1.]
 [1. 0. 0. 3.]
 [0. 3. 2. 1.]]
```

Invers Matrix:

$$M^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{|M|} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

```
from numpy.linalg import inv

print('Inv M:\n', inv(M))
P = np.array([[0,1,0],
              [0,0,0],
              [1,0,1]])
print('det(p):\n', det(P))
```

✓ 0.0s

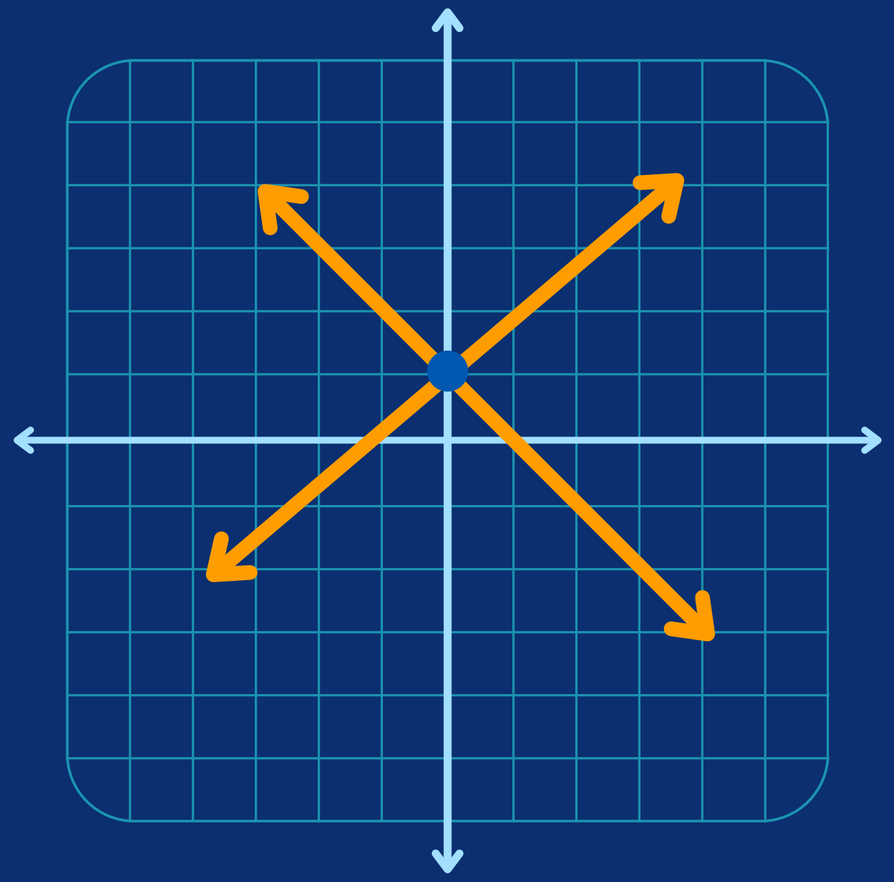
```
Inv M:
 [[-1.57894737 -0.07894737  1.23684211  1.10526316]
 [-0.63157895 -0.13157895  0.39473684  0.84210526]
 [ 0.68421053  0.18421053 -0.55263158 -0.57894737]
 [ 0.52631579  0.02631579 -0.07894737 -0.36842105]]
det(p):
 0.0
```

LINEAR TRANSFORMATIONS

A linear transformation is a function F satisfying:

$$F(ax+by)=aF(x)+bF(y)$$

for all scalars a, b and vectors x, y.



LINEAR TRANSFORMATIONS

Matrix as Linear Function

Matrix multiplication itself is a linear transformation:

$$F(x)=A \cdot x$$

A: m×n matrix

x: n×1 vector

Result F(x): m×1 vector

Domain, Range, and Null Space

Concept	Definition	Notation
Domain	All input vectors x	$\mathbb{R}^n$
Range	All outputs Ax	R(A)
Null Space	Set of x such that Ax = 0	N(A)

Let

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

Domain: $\mathbb{R}^3$

Range: x–y plane (z=0)

Null space: z-axis (all [0,0,z])

SYSTEMS OF LINEAR EQUATIONS

A linear equation is of the form:

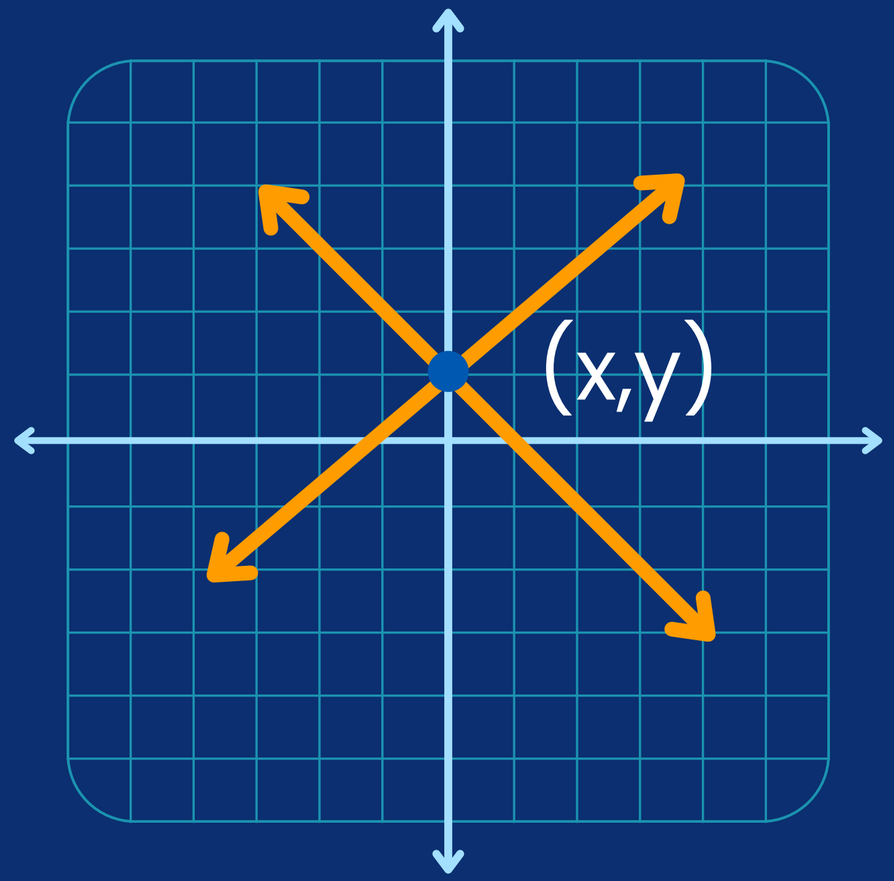
$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = y$$

a_i → coefficients

x_i → unknowns

y → constant (scalar)

Example: 3x + 4y − 3z = −5 → linear.



LINEAR TRANSFORMATIONS

Linear vs Nonlinear

Equation	Linear?	Why
3x + 4y - 3z = -5	✅	first power only
x^2 + 2y = 5	❌	squared term
e^x + y = 2	❌	exponential term

Systems of Equations

A system = multiple linear equations with same variables.

$$\begin{cases} 4x + 3y - 5z = 2 \\ -2x - 4y + 5z = 5 \\ 7x + 8y = -3 \\ x + 2z = 1 \\ 9 + y - 6z = 6 \end{cases}$$

Matrix Representation

System → matrix form: A·x = y

$$\begin{bmatrix} 4 & 3 & -5 \\ -2 & -4 & 5 \\ 7 & 8 & 0 \\ 1 & 0 & 2 \\ 9 & 1 & -6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \\ -3 \\ 1 \\ 6 \end{bmatrix}$$

```python
import numpy as np
A = np.array([[4,3,-5],[-2,-4,5],[7,8,0],[1,0,2],[9,1,-6]])
y = np.array([[2],[5],[-3],[1],[6]])
x = np.linalg.lstsq(A, y, rcond=None)[0]
print(x)
```

✓ 0.0s

```
[[ 0.94120512]
 [-1.19876734]
 [ 0.10032621]]
```

LINEAR TRANSFORMATIONS

RECALL

Graph y= −2x + 4.

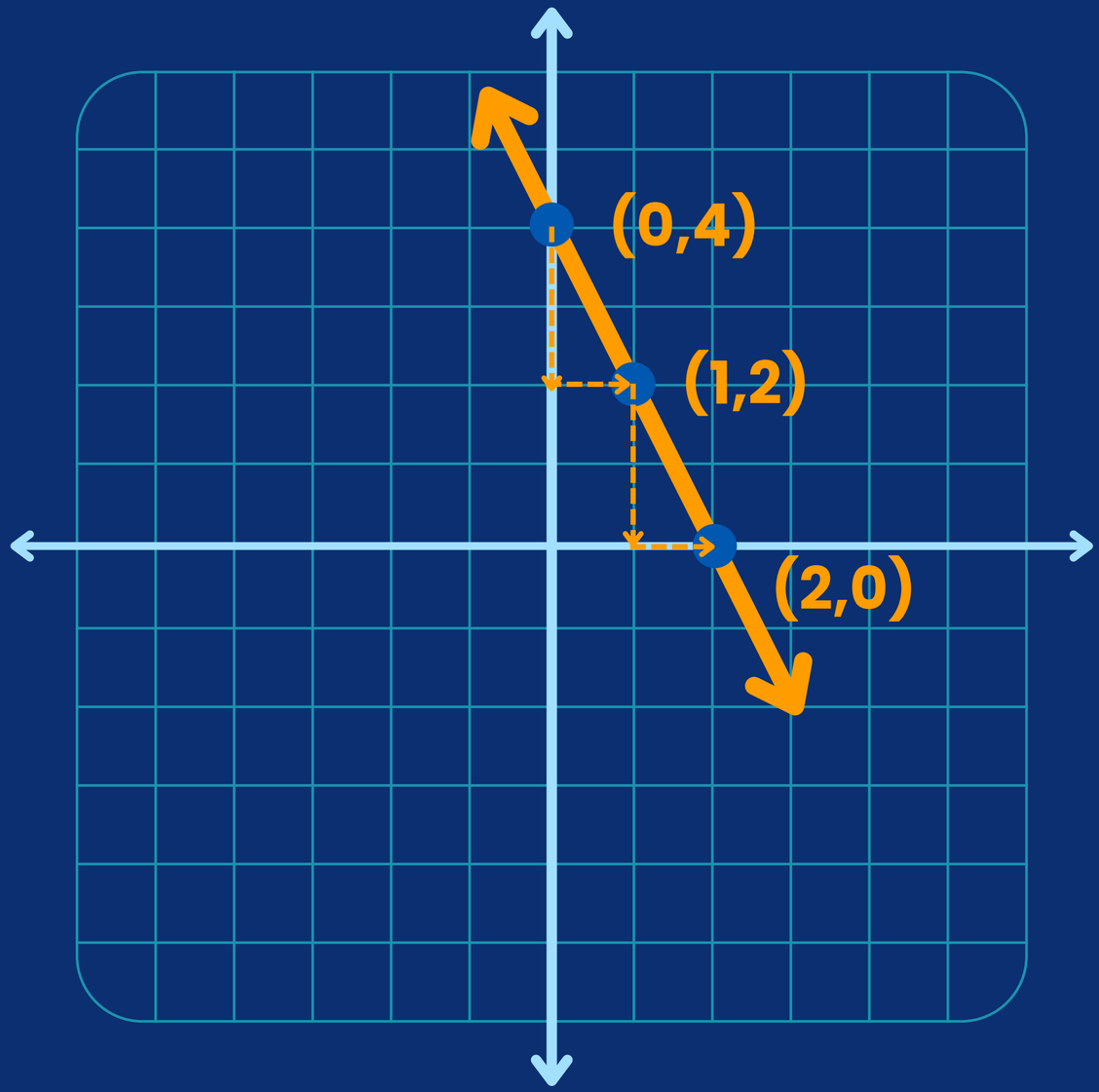
1. Plot the y-intercept.

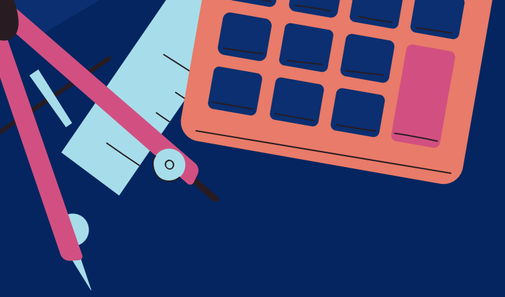
   b 4 (0,4)

2. Use the slope to find at least two other points.

   m m = −2 rise = −2, run = 1

3. Draw a straight line through the points.





EXAMPLE 1

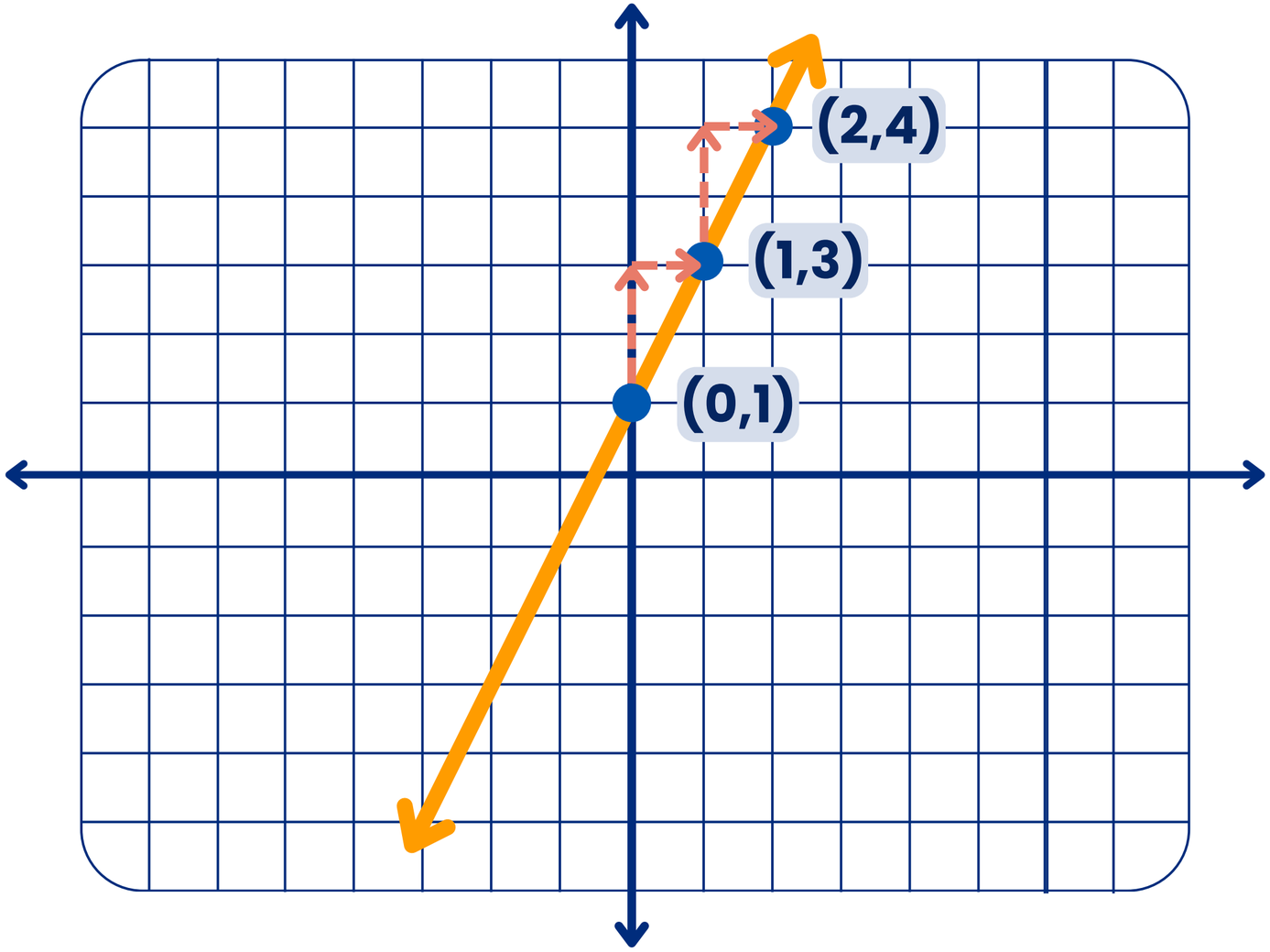
Solve the system of linear equations by graphing:

$$\begin{cases} y = 2x + 1 \\ y = -x - 2 \end{cases}$$

Graph $y = 2x + 1$.

m $m = 2$ rise = 2, run = 1

b 1 (0,1)



EXAMPLE 1

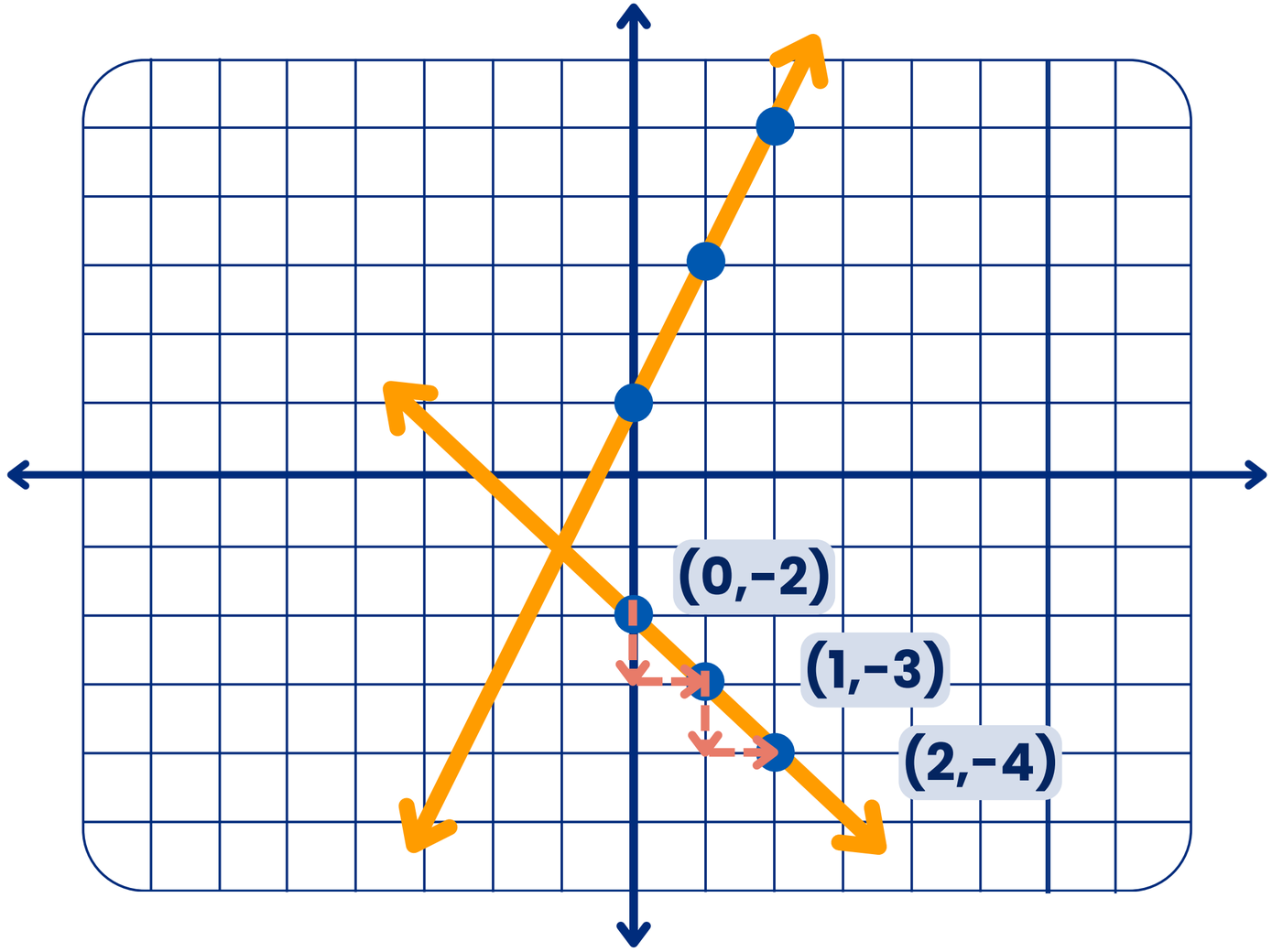
Solve the system of linear equations by graphing:

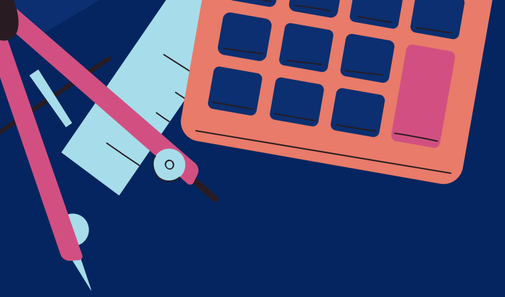
$$\begin{cases} y = 2x + 1 \\ y = -x - 2 \end{cases}$$

Graph $y = -x - 2$

m $m = -1$ $\text{rise} = -1$, $\text{run} = 1$

b -2 $(0,-2)$





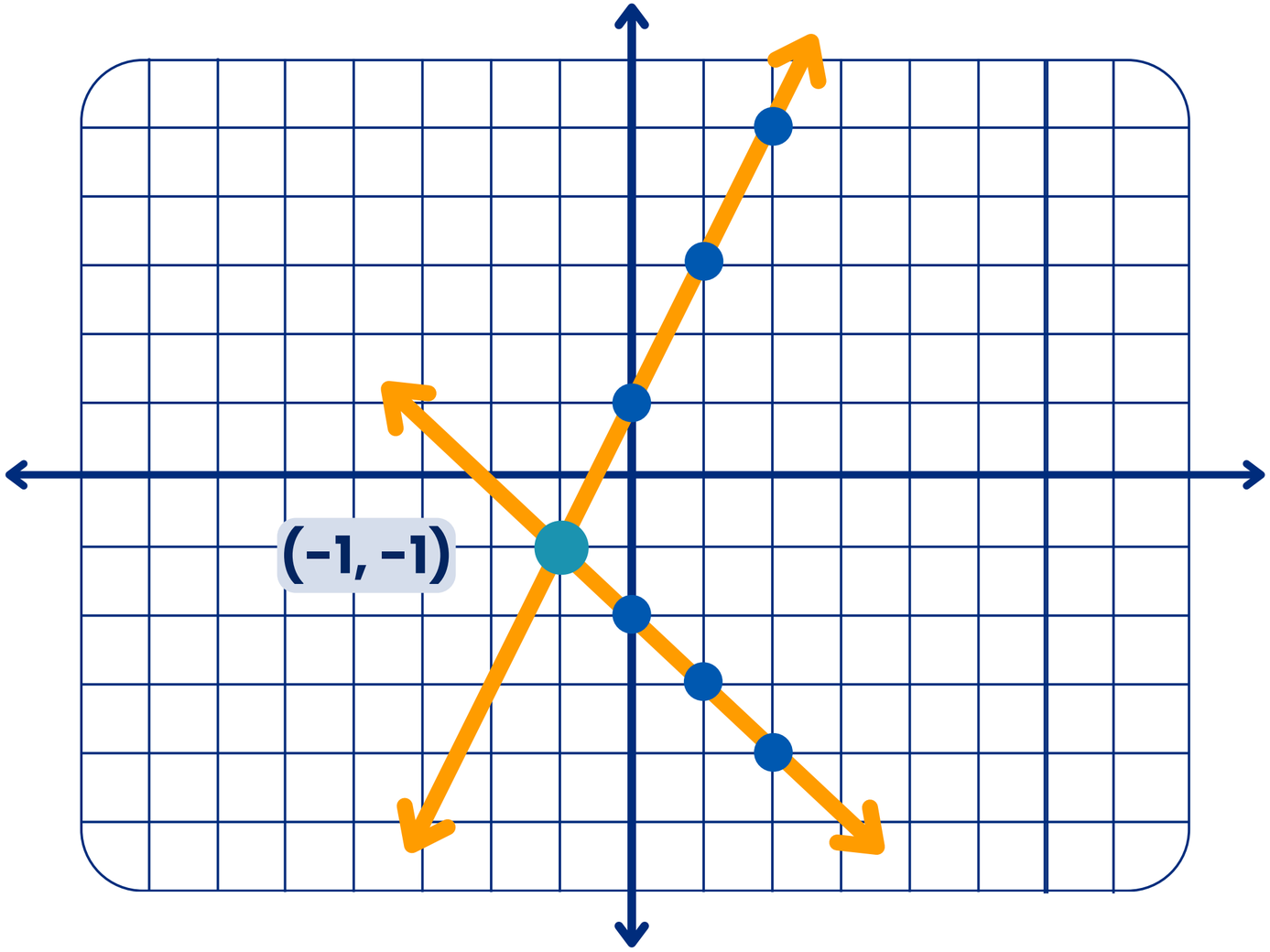
EXAMPLE 1

Solve the system of linear equations by graphing:

$$\begin{cases} y = 2x + 1 \\ y = -x - 2 \end{cases}$$

Find the intersection.

The solution is (–1,–1).



SOLUTIONS TO SYSTEMS OF LINEAR EQUATIONS

Definition:

A system of linear equations consists of multiple equations that can be written in matrix form as Ax=b.

Objective:

Find the vector xxx that satisfies all equations simultaneously.

SOLUTIONS TO SYSTEMS OF LINEAR EQUATIONS

Methods for Solving

Direct Methods
Compute the exact solution.

1. Gaussian Elimination: transforms the matrix into an upper triangular form.

By turning the matrix form into this, we can see the equations turn into:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} \\ 0 & 0 & a'_{3,3} & a'_{3,4} \\ 0 & 0 & 0 & a'_{4,4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix}$$

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + a_{1,4}x_4 &= y_1, \\ a'_{2,2}x_2 + a'_{2,3}x_3 + a'_{2,4}x_4 &= y'_2 \\ a'_{3,3}x_3 + a'_{3,4}x_4 &= y'_3, \\ a'_{4,4}x_4 &= y'_4. \end{aligned}$$

2. Gauss-Jordan Elimination: reduces the matrix further to diagonal form.

Essentially, the equations become:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix}$$

$$\begin{aligned} x_1 + 0 + 0 + 0 &= y'_1, \\ 0 + x_2 + 0 + 0 &= y'_2 \\ 0 + 0 + x_3 + 0 &= y'_3, \\ 0 + 0 + 0 + x_4 &= y'_4. \end{aligned}$$

SOLUTIONS TO SYSTEMS OF LINEAR EQUATIONS

Methods for Solving

3. LU Decomposition: factors A into L (lower) and U (upper) matrices for efficient solving.

$$LUx = y \rightarrow \begin{bmatrix} l_{1,1} & 0 & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} & 0 \\ l_{4,1} & l_{4,2} & l_{4,3} & l_{4,4} \end{bmatrix} \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ 0 & u_{2,2} & u_{2,3} & u_{2,4} \\ 0 & 0 & u_{3,3} & u_{3,4} \\ 0 & 0 & 0 & u_{4,4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

If we define $Ux = M$, then the above equations become:

$$\begin{bmatrix} l_{1,1} & 0 & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} & 0 \\ l_{4,1} & l_{4,2} & l_{4,3} & l_{4,4} \end{bmatrix} M = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

$$\begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ 0 & u_{2,2} & u_{2,3} & u_{2,4} \\ 0 & 0 & u_{3,3} & u_{3,4} \\ 0 & 0 & 0 & u_{4,4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix}$$

SOLUTIONS TO SYSTEMS OF LINEAR EQUATIONS

Methods for Solving

Example:

$$\begin{aligned} -x+2y&=-6 \\ 3x-4y&=14 \end{aligned} \longrightarrow \left[\begin{array}{cc|c} -1 & 2 & -6 \\ 3 & -4 & 14 \end{array}\right] \text{ B1.-1} \longrightarrow \left[\begin{array}{cc|c} 1 & -2 & 6 \\ 3 & -4 & 14 \end{array}\right] \text{ B2-3B1}$$

$$\longrightarrow \left[\begin{array}{cc|c} 1 & -2 & 6 \\ 0 & 2 & -4 \end{array}\right] \text{ B2:2} \longrightarrow \left[\begin{array}{cc|c} 1 & -2 & 6 \\ 0 & 1 & -2 \end{array}\right] \text{ B1+2B2} \longrightarrow \left[\begin{array}{cc|c} 1 & 0 & 2 \\ 0 & 1 & -2 \end{array}\right]$$

So,

$$\begin{aligned} x+0y&=20 \\ x+y&=-2 \end{aligned} \longrightarrow \begin{aligned} x&=2 \\ y&=-2 \end{aligned}$$

SOLUTIONS TO SYSTEMS OF LINEAR EQUATIONS

Methods for Solving

Iterative Methods
Approximate the solution.

1. Gauss-Seidel Method: uses the most recent updated values during iteration.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{bmatrix}$$

we can write its explicit form as:

$$x_i = \frac{1}{a_{i,i}} \left[y_i - \sum_{j=1, j \neq i}^{j=n} a_{i,j} x_j \right]$$

Matrix Properties

- The system has a unique solution if the determinant of A is nonzero.
- If det(A)=0\text{det}(A) = 0det(A)=0, the system may be inconsistent or have infinite solutions.

SOLVE SYSTEMS OF LINEAR EQUATIONS IN PYTHON

Solving a 3×3 Linear System with NumPy

```
import numpy as np

A = np.array([[4, 3, -5],
              [-2, -4, 5],
              [8, 8, 0]])
y = np.array([2, 5, -3])

x = np.linalg.solve(A, y)
print(x)
```

```
[ 2.20833333 -2.58333333 -0.18333333]
```

Output gives the values of x1, x2, x3

Alternative Method: Matrix Inversion Approach

```
A_inv = np.linalg.inv(A)

x = np.dot(A_inv, y)
print(x)
```

```
[ 2.20833333 -2.58333333 -0.18333333]
```

Matrix inversion gives the same solution, but it’s less efficient

SOLVE SYSTEMS OF LINEAR EQUATIONS IN PYTHON

Getting LU (PA) using SciPy

```
from scipy.linalg import lu

P, L, U = lu(A)
print('P:\n', P)
print('L:\n', L)
print('U:\n', U)
print('LU:\n',np.dot(L, U))
```

```
P:
 [[0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]]
L:
 [[ 1.    0.    0.  ]
 [-0.25  1.    0.  ]
 [ 0.5   0.5   1.  ]]
U:
 [[ 8.   8.   0. ]
 [ 0.  -2.   5. ]
 [ 0.   0.  -7.5]]
LU:
 [[ 8.  8.  0.]
 [-2. -4.  5.]
 [ 4.  3. -5.]]
```

P = Permutation matrix
L = Lower triangular matrix
U = Upper triangular matrix
LU = It confirms that the LU decomposition was performed correctly

SOLVE SYSTEMS OF LINEAR EQUATIONS IN PYTHON

Checking the Effect of the Permutation Matrix

```
print(np.dot(P, A))
```

```
[[ 8.  8.  0.]
 [-2. -4.  5.]
 [ 4.  3. -5.]]
```

The reordered matrix PA matches the product LU

MATRIX INVERSION

Inversion and Identity

A matrix inverse M^-1 is defined only for square matrices and satisfies:

$$M \cdot M^{-1} = M^{-1} \cdot M = I$$

For larger matrices, directly computing the inverse using algebraic formulas becomes impractical — so numerical methods are needed.

Case for 4x4 matriks:
One approach: treat each column of the inverse as the solution to a linear system.

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix}$$ and the inverse of M is. $$X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{bmatrix}$$

therefore, we will have:

$$M \cdot X = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix} \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

MATRIX INVERSION

Inversion and Identity

Mx1=e1,Mx2=e2,Mx3=e3,Mx4=e4

where each ei is a column of the identity matrix. The solutions x1,x2,x3,x4 form the columns of M^-1

Therefore, if we solve the above four system of equations, we will get the inverse of the matrix. We can use any method we introduced previously to solve these equations, such as Gauss Elimination, Gauss-Jordan, and LU decomposition. Here, we will just show an example of matrix inversion using Gauss-Jordan method.

Recall that, in Gauss-Jordan method, we convert our problem from

1. Recall that, in Gauss-Jordan method, we convert our problem from

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

To:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix}$$

2. and get the solution. Essentially, we are converting

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} & y_1 \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} & y_2 \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} & y_3 \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} & y_4 \end{bmatrix}$$

To:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & y'_1 \\ 0 & 1 & 0 & 0 & y'_2 \\ 0 & 0 & 1 & 0 & y'_3 \\ 0 & 0 & 0 & 1 & y'_4 \end{bmatrix}$$

MATRIX INVERSION

Inversion and Identity

3. Let us generalize it here, all we need to do is to convert, Create an augmented matrix [M | I]

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} & 1 & 0 & 0 & 0 \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} & 0 & 1 & 0 & 0 \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} & 0 & 0 & 1 & 0 \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} & 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Perform row operations until the left side becomes I → the right side becomes M^-1

$$\begin{bmatrix} 1 & 0 & 0 & 0 & m'_{1,1} & m'_{1,2} & m'_{1,3} & m'_{1,4} \\ 0 & 1 & 0 & 0 & m'_{2,1} & m'_{2,2} & m'_{2,3} & m'_{2,4} \\ 0 & 0 & 1 & 0 & m'_{3,1} & m'_{3,2} & m'_{3,3} & m'_{1,4} \\ 0 & 0 & 0 & 1 & m'_{4,1} & m'_{4,2} & m'_{4,3} & m'_{1,4} \end{bmatrix}$$

Then the matrix $\begin{bmatrix} m'_{1,1} & m'_{1,2} & m'_{1,3} & m'_{1,4} \\ m'_{2,1} & m'_{2,2} & m'_{2,3} & m'_{2,4} \\ m'_{3,1} & m'_{3,2} & m'_{3,3} & m'_{1,4} \\ m'_{4,1} & m'_{4,2} & m'_{4,3} & m'_{1,4} \end{bmatrix}$ is the inverse of M we are looking for.

SUMMARY AND PROBLEMS

Summary

1. Linear algebra is the foundation of many engineering fields
2. Vectors can be considered as points in R^3; addition and multiplication are defined on them, although not necessarily the same as for scalars.
3. A set of vectors is linearly independent if none of the vectors can be written as a linear combination of the others.
4. Matrices are tables of numbers. They have several important properties including the determinant, rank, and inverse.
5. system of linear equations can be represented by the matrix equation Ax=y.
6. The number of solutions to a system of linear equations is related to the rank(A) and the rank([A,y]). It can be zero, one, or infinity.
7. We can solve the equations using Gauss Elimination, Gauss-Jordan Elimination, LU decomposition and Gauss-Seidel method.
8. We introduced methods to find matrix inversion.

SUMMARY AND PROBLEMS

DIKTISAINTEK BERDAMPAK

Problems that might exist, xixixi

1. Show that matrix multiplication distributes over matrix addition: show A(B+C)=AB+AC assuming that A,B, and C are matrices of compatible size.

2. Recognize orthogonality using dot products and norms.
   - 1 if the angle between vectors is close to 90° (within tolerance)
   - 0 otherwise

   Math used:

$$\theta = \arccos\left(\frac{v_1 \cdot v_2}{||v_1||\,||v_2||}\right)$$

Code:

```python
import numpy as np
def my_is_orthogonal(a, b, tol):
    # Compute dot product
    dot_product = np.dot(a.T, b)[0][0]

    # Check if absolute dot product < tolerance
    if abs(dot_product) < tol:
        return 1
    else:
        return 0
# Test cases for problem 2
a = np.array([[1], [0.001]])
b = np.array([[0.001], [1]])
# output: 1
my_is_orthogonal(a,b, 0.01)

# output: 0
my_is_orthogonal(a,b, 0.001)

# output: 0
a = np.array([[1], [0.001]])
b = np.array([[1], [1]])
my_is_orthogonal(a,b, 0.01)

# output: 1
a = np.array([[1], [1]])
b = np.array([[-1], [1]])
my_is_orthogonal(a,b, 1e-10)
```

Output:

```
... 1
```

And 8 more to go (not enough time)...

TERIMA KASIH.