กฎของคราเมอร์ (Cramer’s Rule)

กำหนดให้

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n
\end{aligned}
$$

เป็นระบบสมการซึ่งมี n สมการ และ n ตัวแปร และให้ $A = [a_{ij}]$ เป็นเมทริกซ์สัมประสิทธิ์ ซึ่งสามารถเขียนระบบสมการข้างต้นได้ดังสมการต่อไปนี้

$$AX = B$$

เมื่อ $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$, $X = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}$, $B = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix}$ ถ้า $|A| \neq 0$

แล้วจะได้ว่า ระบบสมการมีผลเฉลยดังนี้

$$x_1 = \frac{|A_1|}{|A|}, x_2 = \frac{|A_2|}{|A|}, \ldots, x_n = \frac{|A_n|}{|A|}$$

โดยที่ A_i เป็นเมทริกซ์ซึ่งได้จากเมทริกซ์ A โดยการแทนที่หลักที่ i ของ A ด้วยเมทริกซ์ B

เพราะฉะนั้นการหาผลเฉลยของระบบสมการโดยใช้กฎของคราเมอร์มีสูตรดังนี้

รูปทั่วไปกฎของคราเมอร์ $x_i = \dfrac{|A_i|}{|A|}; i = 1,2,3,\ldots,n$

เช่น เมื่อ $n = 2;$ $\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{aligned}$ จะได้ $x_1 = \dfrac{\begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}}$ และ $x_2 = \dfrac{\begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}}$

ตัวอย่าง จงหาผลเฉลยด้วยกฎของคราเมอร์ของระบบสมการต่อไปนี้

$$\begin{aligned} x_1 &+ 2x_2 + x_3 = 0 \\ 2x_1 &+ 2x_2 + 3x_3 = 3 \\ -x_1 &- 3x_2 = 2 \end{aligned}$$

วิธีทำ จากสมการจะได้ $\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = \begin{vmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ -1 & -3 & 0 \end{vmatrix} = -1$

จะได้

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}} = \frac{\begin{vmatrix} 0 & 2 & 1 \\ 3 & 2 & 3 \\ 2 & -3 & 0 \end{vmatrix}}{-1} = \frac{-1}{-1} = 1$$

$$x_2 = \frac{\begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}} = \frac{\begin{vmatrix} 1 & 0 & 1 \\ 2 & 3 & 3 \\ -1 & 2 & 0 \end{vmatrix}}{-1} = \frac{1}{-1} = -1$$

$$x_3 = \frac{\begin{vmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ a_{31} & a_{32} & b_3 \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}} = \frac{\begin{vmatrix} 1 & 2 & 0 \\ 2 & 2 & 3 \\ -1 & -3 & 2 \end{vmatrix}}{-1} = \frac{-1}{-1} = 1$$

ดังนั้นคำตอบของสมการคือ $x_1 = 1$, $x_2 = -1$ และ $x_3 = 1$

$$Programming\ (Python)$$

Add Two Matrices Using Numpy

- import numpy as np

- X = [[1, 2, 3],
-     [4, 5, 6],
-     [7, 8, 9]]

- Y = [[9, 8, 7],
-     [6, 5, 4],
-     [3, 2, 1]]

- result = np.array(X) + np.array(Y)

- print(result)

Matrix Multiplication in NumPy

- **Example 1 :** Matrix multiplication of 2 square matrices.

```
# importing the module
import numpy as np

# creating two matrices
p = [[1, 2], [2, 3]]
q = [[4, 5], [6, 7]]
print("Matrix p :")
print(p)
print("Matrix q :")
print(q)

# computing product
result = np.dot(p, q)

# printing the result
print("The matrix multiplication is :")
print(result)
```

- **Example 2 :** Matrix multiplication of 2 rectangular matrices.

```
# importing the module
import numpy as np

# creating two matrices
p = [[1, 2], [2, 3], [4, 5]]
q = [[4, 5, 1], [6, 7, 2]]
print("Matrix p :")
print(p)
print("Matrix q :")
print(q)

# computing product
result = np.dot(p, q)

# printing the result
print("The matrix multiplication is :")
print(result)
```

Calculate the determinant of a matrix using NumPy

- **Example 1:** Calculating Determinant of a **2X2 Numpy matrix** using numpy.linalg.det() function

```
# importing Numpy package
import numpy as np

# creating a 2X2 Numpy matrix
n_array = np.array([[50, 29], [30, 44]])

# Displaying the Matrix
print("Numpy Matrix is:")
print(n_array)

# calculating the determinant of matrix
det = np.linalg.det(n_array)

print("\nDeterminant of given 2X2 matrix:")
print(int(det))
```

- **Example 2:** Calculating Determinant of a **3X3 Numpy matrix** using numpy.linalg.det() function

```python
# importing Numpy package
import numpy as np

# creating a 3X3 Numpy matrix
n_array = np.array([[55, 25, 15],
                    [30, 44, 2],
                    [11, 45, 77]])

# Displaying the Matrix
print("Numpy Matrix is:")
print(n_array)

# calculating the determinant of matrix
det = np.linalg.det(n_array)

print("\nDeterminant of given 3X3 square matrix:")
print(int(det))
```

System of Linear Equations in three variables using Cramer's Rule

```python
import numpy as np

def cramers_rule(A, b):
    """
    Solve a system of linear equations using Cramer's Rule.

    Parameters:
        A (numpy.ndarray): Coefficient matrix.
        b (numpy.ndarray): Constant vector.

    Returns:
        numpy.ndarray: Solution vector.
    """
    det_A = np.linalg.det(A)

    if det_A == 0:
        raise ValueError("The determinant of the matrix is zero. Cramer's Rule cannot be applied.")

    num_variables = A.shape[1]
    solutions = np.zeros(num_variables)

    for i in range(num_variables):
        # Create a modified matrix by replacing the i-th column with b
        A_i = A.copy()
        A_i[:, i] = b
        # Calculate the determinant of the modified matrix
        det_A_i = np.linalg.det(A_i)
        # Solve for x_i
        solutions[i] = det_A_i / det_A

    return solutions
```

```python
# Example Usage
if __name__ == "__main__":
    # Coefficient matrix
    A = np.array([
        [2, -1, 3],
        [1,  3, 2],
        [3,  1, 2]
    ], dtype=float)

    # Constants vector
    b = np.array([5, 10, 8], dtype=float)

    # Solve using Cramer's Rule
    try:
        solution = cramers_rule(A, b)
        print("Solution:", solution)
    except ValueError as e:
        print(e)
```

Excel

จงหา x_1, x_2 และ x_3 จากเมทริกซ์ต่อไปนี้ด้วยเทคนิคของ crimer's rule, $2x_1 + 3x_2 - 2x_3 = 3$, $-x_1 + 2x_2 - x_3 = 1$, $4x_1 + x_2 = 0$

เราต้องแปลงโจทย์เป็น Array ก่อน

จะได้

$$\begin{bmatrix} 2 & 3 & -2 \\ -1 & 2 & -1 \\ 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$$

หลักการคิดของกฎ Cramer คือ

ขั้นตอนที่ 1: หา Det(A) (ตัวหารหลัก)

เราจะหาค่า D หรือ $\det(A)$ โดยการนำเมทริกซ์สัมประสิทธิ์มาคิด:

$$A = \begin{bmatrix} 2 & 3 & -2 \\ -1 & 2 & -1 \\ 4 & 1 & 0 \end{bmatrix}$$

วิธีคิด (คูณลง - คูณขึ้น):

1. **คูณลง (เส้นสีแดง):**
   - $(2)(2)(0) = 0$
   - $(3)(-1)(4) = -12$
   - $(-2)(-1)(1) = 2$
   - **รวมผลคูณลง:** $0 + (-12) + 2 = \mathbf{-10}$
2. **คูณขึ้น (เส้นสีน้ำเงิน):**
   - $(4)(2)(-2) = -16$
   - $(1)(-1)(2) = -2$
   - $(0)(-1)(3) = 0$
   - **รวมผลคูณขึ้น:** $(-16) + (-2) + 0 = \mathbf{-18}$
3. **หา Det:** (ผลคูณลง) - (ผลคูณขึ้น)

$$\det(A) = (-10) - (-18) = -10 + 18 = \mathbf{8}$$

ขั้นตอนที่ 2: หา Det(x_1) (D_1)

นำคำตอบ $[3, 1, 0]$ ไปแทนที่ **หลักที่ 1** (หลักของ x_1):

$$A_1 = \begin{bmatrix} \mathbf{3} & 3 & -2 \\ \mathbf{1} & 2 & -1 \\ \mathbf{0} & 1 & 0 \end{bmatrix}$$

วิธีคิด:

- **คูณลง:** $(3)(2)(0) + (3)(-1)(0) + (-2)(1)(1) = 0 + 0 - 2 = \mathbf{-2}$
- **คูณขึ้น:** $(0)(2)(-2) + (1)(-1)(3) + (0)(1)(3) = 0 - 3 + 0 = \mathbf{-3}$
- **หา Det:** $(-2) - (-3) = -2 + 3 = \mathbf{1}$

ขั้นตอนที่ 3: หา Det(x_2) (D_2)

นำคำตอบ $[3, 1, 0]$ ไปแทนที่ **หลักที่ 2** (หลักของ x_2):

$$A_2 = \begin{bmatrix} 2 & \mathbf{3} & -2 \\ -1 & \mathbf{1} & -1 \\ 4 & \mathbf{0} & 0 \end{bmatrix}$$

วิธีคิด:

- **คูณลง:** $(2)(1)(0) + (3)(-1)(4) + (-2)(-1)(0) = 0 - 12 + 0 = \mathbf{-12}$
- **คูณขึ้น:** $(4)(1)(-2) + (0)(-1)(2) + (0)(-1)(3) = -8 + 0 + 0 = \mathbf{-8}$
- **หา Det:** $(-12) - (-8) = -12 + 8 = \mathbf{-4}$

ขั้นตอนที่ 4: หา Det(x_3) (D_3)

นำคำตอบ $[3, 1, 0]$ ไปแทนที่ **หลักที่ 3** (หลักของ x_3):

$$A_3 = \begin{bmatrix} 2 & 3 & \mathbf{3} \\ -1 & 2 & \mathbf{1} \\ 4 & 1 & \mathbf{0} \end{bmatrix}$$

วิธีคิด:

- **คูณลง:** $(2)(2)(0) + (3)(1)(4) + (3)(-1)(1) = 0 + 12 - 3 = \mathbf{9}$
- **คูณขึ้น:** $(4)(2)(3) + (1)(1)(2) + (0)(-1)(3) = 24 + 2 + 0 = \mathbf{26}$
- **หา Det:** $9 - 26 = \mathbf{-17}$

ขั้นตอนสุดท้าย: สรุปคำตอบ

จับ Det ย่อย หารด้วย Det หลัก ($D = 8$)

$$x_1 = \frac{\det(A_1)}{\det(A)} = \frac{1}{8} = \mathbf{0.125}$$
$$x_2 = \frac{\det(A_2)}{\det(A)} = \frac{-4}{8} = \mathbf{-0.5}$$
$$x_3 = \frac{\det(A_3)}{\det(A)} = \frac{-17}{8} = \mathbf{-2.125}$$

ต่อไปถ้าทำใน Excel ละ คตง่าย

ตัวแปลได้แก่

		A		X		b		
	2	3	-2	x1		3		
	-1	2	-1	x2		1		
	4	1	0	x3		0		
det(A)	8							
det(x1)					det(x2)			
	3	3	-2			2	3	-2
	1	2	-1			-1	1	-1
	0	1	0			4	0	0
	1							
x1 = det(x1)/det(A)						-4		
					x2=det(x2)/det(A)			
	8							
						-2		

หา det(A) ยังไง ใช้ฟังก์ชันนี้ได้เลย

=MDETERM((ตำแหน่ง0,0) : (ตำแหน่ง2,2))

$$Code\ Python$$

```
import numpy as np

# ฟังก์ชัน Cramer's Rule ที่คุณเขียนมา (ถูกต้องแล้วครับ)
def cr(A, b):
    det_a = np.linalg.det(A)
    # เช็คว่า Det A เป็น 0 หรือไม่ (ถ้าเป็น 0 จะหาคำตอบไม่ได้)
    if np.isclose(det_a, 0):
        raise ValueError("Cannot divided by zero (Det A = 0)!!!")

    num_variable = A.shape[1]
    solution = np.zeros(num_variable)

    for i in range(num_variable):
        A_i = A.copy()
        A_i[:, i] = b  # เอาคำตอบ (b) ไปแทนในหลักที่ i
        det_A_i = np.linalg.det(A_i)
        solution[i] = det_A_i / det_a

    return solution

# --- ส่วนที่แก้ไข: ใส่ค่าจากโจทย์ในรูปภาพ ---

# เมทริกซ์สัมประสิทธิ์ (Coefficient Matrix)
# บรรทัด 3: 4x1 + 1x2 + 0x3 = 0
A = np.array([
    [2, 3, -2],
```

```
    [-1, 2, -1],
    [4, 1, 0]
], dtype=float)

# เมทริกซ์คำตอบ (Constant Matrix)
b = np.array([3, 1, 0], dtype=float)

print("--- โจทย์ Cramer's Rule ---")
print("Matrix A:\n", A)
print("Vector b:\n", b)
print("-" * 30)

try:
    solution = cr(A, b)
    print("คำตอบที่ได้ (x1, x2, x3):")
    # แสดงผลทศนิยม 6 ตำแหน่ง
    print(f"x1 = {solution[0]:.6f}")
    print(f"x2 = {solution[1]:.6f}")
    print(f"x3 = {solution[2]:.6f}")

    # เช็คคำตอบด้วยไลบรารี numpy โดยตรง (เพื่อความชัวร์)
    # x_check = np.linalg.solve(A, b)
    # print("Check with numpy:", x_check)

except ValueError as e:
    print(e)
```