[EEC3600-001] 수치해석		
소속: 전기전자공학부	학번: 12191529	이름: 장준영
Python Programming for Numerical Analysis		HW number: #3

3.1. (Matrix Condition) By evaluating the determinant, classify the following matrices as singular, ill conditione, or well conditioned:

a. 문제 내용

Problem 3.1 *(Matrix condition)* [Python only]

By evaluating the determinant, classify the following matrices as singular, ill conditioned, or well conditioned:

(a) $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$

(b) $\mathbf{A} = \begin{bmatrix} 2.11 & -0.80 & 1.72 \\ -1.84 & 3.03 & 1.29 \\ -1.57 & 5.25 & 4.30 \end{bmatrix}$

(c) $\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$

(d) $\mathbf{A} = \begin{bmatrix} 4 & 3 & -1 \\ 7 & -2 & 3 \\ 5 & -18 & 13 \end{bmatrix}$

b. 파이썬 코드 및 결과

Code

```python
def determinant(matrix):
    if len(matrix) == 1:
        return matrix[0][0]
    if len(matrix) == 2:
        return matrix[0][0]*matrix[1][1] - matrix[0][1]*matrix[1][0]

    det = 0
    for c in range(len(matrix)):
        minor = [row[:c] + row[c+1:] for row in matrix[1:]]
        det += ((-1) ** c) * matrix[0][c] * determinant(minor)
    return det

# 행렬 정의 (리스트로 직접 입력)
A_list = {
    'a': [[1, 2, 3], [2, 3, 4], [3, 4, 5]],
    'b': [[2.11, -0.80, 1.72], [-1.84, 3.03, 1.29], [-1.57, 5.25, 4.30]],
    'c': [[2, -1, 0], [-1, 2, -1], [0, -1, 2]],
    'd': [[4, 3, -1], [7, -2, 3], [5, -18, 13]]
}

threshold = 1e-1

# 계산 및 분류
for key, mat in A_list.items():
    det = determinant(mat)
    print(f"Matrix {key}: Determinant = {det:.6f}", end=' -> ')
    if abs(det) < 1e-10:
        print("Singular")
    elif abs(det) < threshold:
        print("Ill-conditioned")
    else:
        print("Well-conditioned")
```

Result
Matrix a: Determinant = 0.000000 -> Singular
Matrix b: Determinant = 0.058867 -> Ill-conditioned
Matrix c: Determinant = 4.000000 -> Well-conditioned
Matrix d: Determinant = 0.000000 -> Singular

c. 설명

: 라플라스 전개를 활용하여 각각의 행렬식을 구하였다(ill-contioned threshold 경계조건 = 0.1).

Problem 3.1 *(Matrix condition)* [Python only]

By evaluating the determinant, classify the following matrices as singular, ill conditioned, or well conditioned:

det = 0   det 매우 작음.   det 충분히 큼.

(a) $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$

(b) $\mathbf{A} = \begin{bmatrix} 2.11 & -0.80 & 1.72 \\ -1.84 & 3.03 & 1.29 \\ -1.57 & 5.25 & 4.30 \end{bmatrix}$

(c) $\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$

(d) $\mathbf{A} = \begin{bmatrix} 4 & 3 & -1 \\ 7 & -2 & 3 \\ 5 & -18 & 13 \end{bmatrix}$

(1) $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{pmatrix} = (1)\cdot(-1)^{1+1}\begin{vmatrix} 3 & 4 \\ 4 & 5 \end{vmatrix} + 2\cdot(-1)^{1+2}\begin{vmatrix} 2 & 4 \\ 3 & 5 \end{vmatrix} + 3\cdot(-1)^{1+3}\begin{vmatrix} 2 & 3 \\ 3 & 4 \end{vmatrix}$

$= \underbrace{(-1)}_{-1} - \underbrace{(2)(-2)}_{+4} + \underbrace{(3)(-1)}_{-3} = \boxed{0}$ (singular)

(2) $\begin{pmatrix} 2.11 & -0.80 & 1.72 \\ -1.84 & 3.03 & 1.29 \\ -1.57 & 5.25 & 4.30 \end{pmatrix}$

$= (2.11)(-1)^{1+1}\begin{vmatrix} 3.03 & 1.29 \\ 5.25 & 4.30 \end{vmatrix} + (-0.80)(-1)^{1+2}\begin{vmatrix} -1.84 & 1.29 \\ -1.57 & 4.30 \end{vmatrix} + (1.72)(-1)^{1+3}\begin{vmatrix} -1.84 & 3.03 \\ -1.57 & 5.25 \end{vmatrix}$

$= (2.11)(3.03\times4.30 - 1.29\cdot5.25) - (-0.80)(-1.84\cdot4.30 + 1.29\cdot1.57) + (1.72)(-1.84\cdot5.25 + 1.57\cdot3.03)$

$= 58.867\times10^{-3} = 0.058867$ (ill-conditioned)

(c) $\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$

$= (2)\cdot(-1)^{1+1}\begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} + (-1)(-1)^{1+2}\begin{vmatrix} -1 & -1 \\ 0 & 2 \end{vmatrix} + \cancel{(0)(-1)^{1+3}\begin{vmatrix} -1 & 2 \\ 0 & -1 \end{vmatrix}} \rightarrow 0$

$= (2)(3) - (-1)(-2) = 6 - 2 = \boxed{4}$

(d) $\begin{pmatrix} 4 & 3 & -1 \\ 7 & -2 & 3 \\ 5 & -18 & 13 \end{pmatrix}$

$= (4)\cdot(-1)^{1+1}\cdot\begin{vmatrix} -2 & 3 \\ -18 & 13 \end{vmatrix} + (3)\cdot(-1)^{1+2}\cdot\begin{vmatrix} 7 & 3 \\ 5 & 13 \end{vmatrix} + (-1)(-1)^{1+3}\begin{vmatrix} 7 & -2 \\ 5 & -18 \end{vmatrix}$

$= (4)\underbrace{(-26+54)}_{28} - (3)\underbrace{(91-15)}_{76} + (-1)\underbrace{(-126+10)}_{-116}$

$= 112 - 228 + 116 = \boxed{0}$ (singular)

3.2. (Using LU factorization to solve a linear system) Use the results of LU decomposition to solve Ax=b:

a. 문제 내용

Problem 3.2 *(Using LU factorization to solve a linear system)* **[Python & Hand-written]**

Use the results of LU decomposition

$$\mathbf{A} = \mathbf{LU} = \begin{bmatrix} 1 & 0 & 0 \\ 3/2 & 1 & 0 \\ 1/2 & 11/13 & 1 \end{bmatrix} \begin{bmatrix} 2 & -3 & -1 \\ 0 & 13/2 & -7/2 \\ 0 & 0 & 32/13 \end{bmatrix}$$

to solve

$$\mathbf{Ax} = \mathbf{b}$$

where

$$\mathbf{b} = [1 \quad -1 \quad 2]^{\top}.$$

b. Python 코드 및 결과

Code

```python
from fractions import Fraction

# L과 U 행렬 (Fraction 사용)
L = [
    [Fraction(1), Fraction(0), Fraction(0)],
    [Fraction(3, 2), Fraction(1), Fraction(0)],
    [Fraction(1, 2), Fraction(11, 13), Fraction(1)]
]

U = [
    [Fraction(2), Fraction(-3), Fraction(-1)],
    [Fraction(0), Fraction(13, 2), Fraction(-7, 2)],
    [Fraction(0), Fraction(0), Fraction(32, 13)]
]

# b 벡터
b = [Fraction(1), Fraction(-1), Fraction(2)]

# Step 1: Ly = b => forward substitution
y = [Fraction(0) for _ in range(3)]
y[0] = b[0]
y[1] = b[1] - L[1][0] * y[0]
y[2] = b[2] - L[2][0] * y[0] - L[2][1] * y[1]
print("Solution y =", y)

# Step 2: Ux = y => back substitution
x = [Fraction(0) for _ in range(3)]
x[2] = y[2] / U[2][2]
x[1] = (y[1] - U[1][2] * x[2]) / U[1][1]
x[0] = (y[0] - U[0][1] * x[1] - U[0][2] * x[2]) / U[0][0]

print("Solution x =", x)
```

Result

```
Solution y = [Fraction(1, 1), Fraction(-5, 2), Fraction(47, 13)]
Solution x = [Fraction(59, 32), Fraction(13, 32), Fraction(47, 32)]
```

c. 설명

sol)

① Forward substitution

$$\therefore Ly=b \longrightarrow \begin{pmatrix} 1 & 0 & 0 \\ 3/2 & 1 & 0 \\ 1/2 & 11/13 & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix} \qquad \therefore y_1 = 1, \quad y_2 = -\frac{5}{2}, \quad y_3 = \frac{47}{13}$$

② Backward substitution

$$\therefore Ux = y \longrightarrow \begin{pmatrix} 2 & -3 & -1 \\ 0 & 13/2 & -7/2 \\ 0 & 0 & 32/13 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -\frac{5}{2} \\ \frac{47}{13} \end{pmatrix}$$

$$\therefore \frac{32}{13}x_3 = \frac{47}{13} \ ; \ x_3 = \frac{47}{32} \qquad \therefore x_1 = \frac{59}{32}, \ x_2 = \frac{13}{32}, \ x_3 = \frac{47}{32}$$

$$\frac{13}{2}x_2 - \frac{7}{2} \times \frac{47}{32} = -\frac{5}{2} \ ; \ x_2 = \frac{13}{32}$$

$$2x_1 - 3x_2 - x_3 = 1$$

$$; \ 2x_1 - \frac{39}{32} - \frac{47}{32} = 1 \ ; \ 2x_1 = 1 + \frac{86}{32} = \frac{118}{32}$$

$$x_1 = \frac{59}{32}$$

3.3. (Gauss elimination to solve a linear system) Solve the equation AX=B by Gauss elimination:

a. 문제 내용

Problem 3.3 *(Gauss elimination to solve a linear system)* [Python only]

Solve the equations $\mathbf{AX} = \mathbf{B}$ by Gauss elimination, where

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 1 & 2 & 0 \\ -1 & 2 & 0 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

b. Python 코드 및 결과

```python
from fractions import Fraction

def gauss_elimination_fraction(A, B):
    n = len(A)
    # Deep copy to avoid modifying input
    A = [[Fraction(A[i][j]) for j in range(n)] for i in range(n)]
    B = [[Fraction(B[i][j]) for j in range(len(B[0]))] for i in range(n)]

    # Forward elimination
    for i in range(n):
        # Pivoting if needed (simple row swap)
        if A[i][i] == 0:
            for k in range(i + 1, n):
                if A[k][i] != 0:
                    A[i], A[k] = A[k], A[i]
                    B[i], B[k] = B[k], B[i]
                    break

        for j in range(i + 1, n):
            factor = A[j][i] / A[i][i]
            for k in range(i, n):
                A[j][k] -= factor * A[i][k]
            for k in range(len(B[0])):
                B[j][k] -= factor * B[i][k]

    # Back substitution
    X = [[Fraction(0) for _ in range(len(B[0]))] for _ in range(n)]
    for i in reversed(range(n)):
        for j in range(len(B[0])):
            total = sum(A[i][k] * X[k][j] for k in range(i + 1, n))
            X[i][j] = (B[i][j] - total) / A[i][i]

    return X

# A, B 정의 (리스트로)
A = [
    [2, 0, -1, 0],
    [0, 1, 2, 0],
    [-1, 2, 0, 1],
    [0, 0, 1, -2]
]

B = [
    [1, 0],
    [0, 0],
    [0, 1],
    [0, 0]
]

# 계산
X = gauss_elimination_fraction(A, B)

# 결과 출력
print("Solution X (as fractions):")
for row in X:
    print([str(val) for val in row])
```

Code

```
Solution X (as fractions):
['7/16', '-1/8']
['1/4', '1/2']
['-1/8', '-1/4']
['-1/16', '-1/8']
```

Result

c. 설명

sol)

$$[A|B] = \left(\begin{array}{cccc|cc} 2 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ -1 & 2 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -2 & 0 & 0 \end{array}\right) \rightarrow \left(\begin{array}{cccc|cc} 1 & 0 & -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ -1 & 2 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -2 & 0 & 0 \end{array}\right)$$

$$\rightarrow \left(\begin{array}{cccc|cc} 1 & 0 & -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 2 & -0.5 & 1 & 0.5 & 1 \\ 0 & 0 & 1 & -2 & 0 & 0 \end{array}\right) \xrightarrow[(2행)\times(-2)+(3행)]{} \left(\begin{array}{cccc|cc} 1 & 0 & -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & -4.5 & 1 & 0.5 & 1 \\ 0 & 0 & 1 & -2 & 0 & 0 \end{array}\right)$$

$$\rightarrow \left(\begin{array}{cccc|cc} 1 & 0 & -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 0 & 0 \\ 0 & 0 & -4.5 & 1 & 0.5 & 1 \end{array}\right) \rightarrow \begin{array}{c} \begin{array}{cccccc} & & & & b_1 & b_2 \end{array} \\ \left(\begin{array}{cccc|cc} 1 & 0 & -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 0 & 0 \\ 0 & 0 & 0 & -8 & 0.5 & 1 \end{array}\right) \end{array}$$

$\boxed{1}$ b_1에 대하여:

- $-8x_4 = 0.5$; $x_4 = -\frac{1}{16}$
- $x_3 - 2x_4 = 0$; $x_3 = -\frac{1}{8}$
- $x_2 + 2x_3 = 0$; $x_2 = \frac{1}{4}$
- $x_1 - \frac{1}{2}x_3 = \frac{1}{2}$; $x_1 = \frac{7}{16}$

$\boxed{2}$ b_2 에 대하여:

- $-8x_4 = 1$; $x_4 = -\frac{1}{8}$
- $x_3 - 2x_4 = 0$; $x_3 = -\frac{1}{4}$
- $x_2 + 2x_3 = 0$; $x_2 = \frac{1}{2}$
- $x_1 - \frac{1}{2}x_3 = 0$; $x_1 = -\frac{1}{8}$

답: $X = \begin{bmatrix} \frac{7}{16} & -\frac{1}{8} \\ \frac{1}{4} & \frac{1}{2} \\ -\frac{1}{8} & -\frac{1}{4} \\ -\frac{1}{16} & -\frac{1}{8} \end{bmatrix}$

3.4. (LU and Choleski's factorization) Find L and U so that using (a) Doolittle's decomposition; (b) Choleski's decomposition:

a. 문제 내용

> **Problem 3.4** *(LU and Choleski's factorization)* **[Python & Hand-written]**
>
> Find $\mathbf{L}$ and $\mathbf{U}$ so that
>
> $$\mathbf{A} = \mathbf{LU} = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}$$
>
> using (a) Doolittle's decomposition; (b) Choleski's decomposition.

b. Python 코드 및 결과

(a)

Code

```python
##############################
# (a) Doolittle's LU decomposition (Fraction)
##############################
def doolittle_fraction(A):
    n = A.shape[0]
    L = np.array([[Fraction(0) for _ in range(n)] for _ in range(n)])
    U = np.array([[Fraction(0) for _ in range(n)] for _ in range(n)])

    for i in range(n):
        L[i][i] = Fraction(1)
        for j in range(i, n):
            U[i][j] = A[i][j] - sum(L[i][k]*U[k][j] for k in range(i))
        for j in range(i+1, n):
            L[j][i] = (A[j][i] - sum(L[j][k]*U[k][i] for k in range(i))) / U[i][i]

    return L, U
```

Result

```
=== Doolittle LU Decomposition (Fraction) ===
L =
[[Fraction(1, 1) Fraction(0, 1) Fraction(0, 1)]
 [Fraction(-1, 4) Fraction(1, 1) Fraction(0, 1)]
 [Fraction(0, 1) Fraction(-4, 15) Fraction(1, 1)]]
U =
[[Fraction(4, 1) Fraction(-1, 1) Fraction(0, 1)]
 [Fraction(0, 1) Fraction(15, 4) Fraction(-1, 1)]
 [Fraction(0, 1) Fraction(0, 1) Fraction(56, 15)]]
LU =
[[Fraction(4, 1) Fraction(-1, 1) Fraction(0, 1)]
 [Fraction(-1, 1) Fraction(4, 1) Fraction(-1, 1)]
 [Fraction(0, 1) Fraction(-1, 1) Fraction(4, 1)]]
```

(b)

Code

```python
################################
# (b) Choleski decomposition (float version)
################################
def choleski(A_float):
    A = A_float.copy()
    n = len(A)
    for k in range(n):
        try:
            A[k, k] = math.sqrt(A[k, k] - np.dot(A[k, :k], A[k, :k]))
        except ValueError:
            raise ValueError("Matrix is not positive definite")

        for i in range(k+1, n):
            A[i, k] = (A[i, k] - np.dot(A[i, :k], A[k, :k])) / A[k, k]

    for k in range(1, n):
        A[:k, k] = 0.0

    return A

def choleskiSol(L, b):
    n = len(b)
    # [L]{y} = {b}
    for k in range(n):
        b[k] = (b[k] - np.dot(L[k, :k], b[:k])) / L[k, k]
    # [L.T]{x} = {y}
    for k in range(n-1, -1, -1):
        b[k] = (b[k] - np.dot(L[k+1:n, k], b[k+1:n])) / L[k, k]
    return b
```

Result

```
=== Choleski Decomposition (Float) ===
L =
[[ 2.          0.          0.        ]
 [-0.5         1.93649167  0.        ]
 [ 0.         -0.51639778  1.93218357]]
L @ L.T =
[[ 4. -1.  0.]
 [-1.  4. -1.]
 [ 0. -1.  4.]]
```

c. 설명

(a)

(a) Doolittle

$$\therefore \left(\begin{array}{ccc|ccc} 4 & -1 & 0 & 1 & 0 & 0 \\ -1 & 4 & -1 & \square & 1 & 0 \\ 0 & -1 & 4 & \square & \square & 1 \end{array}\right) \rightarrow \left(\begin{array}{ccc|ccc} 4 & -1 & 0 & 1 & 0 & 0 \\ 0 & \frac{15}{4} & -1 & -\frac{1}{4} & 1 & 0 \\ 0 & -1 & 4 & 0 & \square & 1 \end{array}\right) \rightarrow \left(\begin{array}{ccc|ccc} 4 & -1 & 0 & 1 & 0 & 0 \\ 0 & \frac{15}{4} & -1 & -\frac{1}{4} & 1 & 0 \\ 0 & 0 & \frac{56}{15} & 0 & -\frac{4}{15} & 1 \end{array}\right)$$

답 : $A = LU = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{4} & 1 & 0 \\ 0 & -\frac{4}{15} & 1 \end{bmatrix} \begin{bmatrix} 4 & -1 & 0 \\ 0 & \frac{15}{4} & -1 \\ 0 & 0 & \frac{56}{15} \end{bmatrix}$

(b)

(b) Choleski

$$\therefore A = LL^T = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix} = \begin{bmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32} \\ l_{11}l_{31} & l_{21}l_{31} + l_{22}l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix}$$

$$\therefore l_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}, \quad \underset{(i>j)}{l_{ij}} = \frac{A_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}}{l_{jj}}$$

① $l_{11} = \sqrt{A_{11}} = 2 \rightarrow l_{21} = \frac{A_{21}}{l_{11}} = -\frac{1}{2}, \quad l_{31} = \frac{A_{31}}{l_{11}} = 0$

② $l_{22} = \sqrt{A_{22} - l_{21}^2} = \frac{\sqrt{15}}{2} \rightarrow l_{32} = \frac{A_{32} - l_{31}l_{21}}{l_{22}} = \frac{(-1) - 0}{\sqrt{15}/2} = -\frac{2}{\sqrt{15}}$

③ $l_{33} = \sqrt{A_{33} - (l_{31}^2 + l_{32}^2)} = \sqrt{4 - (0 + \frac{4}{15})} = \frac{\sqrt{56}}{\sqrt{15}}$

$$\Rightarrow L = \begin{bmatrix} 2 & 0 & 0 \\ -\frac{1}{2} & \frac{\sqrt{15}}{2} & 0 \\ 0 & -\frac{2}{\sqrt{15}} & \frac{\sqrt{56}}{\sqrt{15}} \end{bmatrix}$$

3.5. (Interpolation as a linear system solution) Determine the coefficients of the polynomial $y = a_0 + a_1x + a_2x^2 + a_3x^3$ that passes through the points (0,10), (1,35), (3,31), and (4,2).

a. 문제 내용

Problem 3.5 *(Interpolation as a linear system solution)* **[Python & Hand-written]**

Determine the coefficients of the polynomial $y = a_0 + a_1x + a_2x^2 + a_3x^3$ that passes through the points $(0, 10)$, $(1, 35)$, $(3, 31)$, and $(4, 2)$.

b. Python 코드 및 결과

Code

```
import numpy as np
from fractions import Fraction

# 점들
points = [(0, 10), (1, 35), (3, 31), (4, 2)]

# A 행렬과 b 벡터 구성 (Fraction 사용)
A = []
b = []

for x, y in points:
    A.append([Fraction(1), Fraction(x), Fraction(x**2), Fraction(x**3)])
    b.append(Fraction(y))

A = np.array(A)
b = np.array(b)

# 연립방정식 Ax = b 풀기
coefficients = np.linalg.solve(A.astype(np.float64), b.astype(np.float64))

# 결과 출력 (Fraction 근사 출력)
print("=== Coefficients of the polynomial ===")
for i, coef in enumerate(coefficients):
    print(f"a_{i} = {coef:.5f}")

# 다항식 형태 출력
print("\nPolynomial:")
poly = " + ".join([f"{coef:.3f}*x^{i}" if i > 0 else f"{coef:.3f}" for i, coef in enumerate(coefficients)])
print(f"y = {poly}")
```

Result

```
=== Coefficients of the polynomial ===
a_0 = 10.00000
a_1 = 34.00000
a_2 = -9.00000
a_3 = 0.00000

Polynomial:
y = 10.000 + 34.000*x^1 + -9.000*x^2 + 0.000*x^3
```

c. 설명

sol) 조건 대입 $\Rightarrow$
$$\begin{cases} a_0 + 0\cdot a_1 + 0^2\cdot a_2 + 0^3\cdot a_3 = 10 \\ a_0 + 1\cdot a_1 + 1^2\cdot a_2 + 1^3\cdot a_3 = 35 \\ a_0 + 3\cdot a_1 + 3^2\cdot a_2 + 3^3\cdot a_3 = 31 \\ a_0 + 4\cdot a_1 + 4^2\cdot a_2 + 4^3\cdot a_3 = 2 \end{cases} \rightarrow \overset{A}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{bmatrix}} \overset{x}{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}} = \overset{b}{\begin{bmatrix} 10 \\ 35 \\ 31 \\ 2 \end{bmatrix}}$$

$$\therefore [A|b] = \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 10 \\ 1 & 1 & 1 & 1 & 35 \\ 1 & 3 & 9 & 27 & 31 \\ 1 & 4 & 16 & 64 & 2 \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 10 \\ 0 & 1 & 1 & 1 & 25 \\ 0 & 3 & 9 & 27 & 21 \\ 0 & 4 & 16 & 64 & -8 \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 10 \\ 0 & 1 & 1 & 1 & 25 \\ 0 & 0 & 6 & 24 & -54 \\ 0 & 0 & 12 & 60 & -108 \end{array}\right)$$

$$\rightarrow \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 10 \\ 0 & 1 & 1 & 1 & 25 \\ 0 & 0 & 6 & 24 & -54 \\ 0 & 0 & 0 & 12 & 0 \end{array}\right)$$

① $a_3 = 0$

② $6a_2 = -54$; $a_2 = -9$

③ $a_1 + a_2 = 25$; $a_1 = 34$

④ $a_0 = 10$

$$\boxed{답: \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 34 \\ -9 \\ 0 \end{bmatrix}}$$

3.6. (Gauss elimination for a complex linear system) What we learned also works with complex numbers. Use it (any one of what we studied in class) to solve Ax = b, where:

a. 문제 내용

Problem 3.6 *(Gauss elimination for a complex linear system)* [Python only]

What we learned also works with complex numbers. Use it (any one of what we studied in class) to solve $\mathbf{Ax} = \mathbf{b}$, where

$$\mathbf{A} = \begin{bmatrix} 5+i & 5+2i & -5+3i & 6-3i \\ 5+2i & 7-2i & 8-i & -1+3i \\ -5+3i & 8-i & -3-3i & 2+2i \\ 6-3i & -1+3i & 2+2i & 8+14i \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 15-35i & 2+10i & -2-34i & 8+14i \end{bmatrix}^\top$$

b. Python 코드 및 결과

Code

```python
import numpy as np

def gauss_elimination_complex(A, b):
    A = A.astype(complex)  # ensure complex type
    b = b.astype(complex)
    n = len(b)

    # Forward elimination
    for k in range(n):
        # Pivoting (partial, complex-compatible)
        max_row = np.argmax(np.abs(A[k:, k])) + k
        if max_row != k:
            A[[k, max_row]] = A[[max_row, k]]
            b[[k, max_row]] = b[[max_row, k]]

        for i in range(k + 1, n):
            factor = A[i, k] / A[k, k]
            A[i, k:] -= factor * A[k, k:]
            b[i] -= factor * b[k]

    # Back substitution
    x = np.zeros(n, dtype=complex)
    for i in reversed(range(n)):
        x[i] = (b[i] - np.dot(A[i, i + 1:], x[i + 1:])) / A[i, i]

    return x

# Define A and b
A = np.array([
    [5+1j, 5+2j, -5+3j, 6-3j],
    [5+2j, 7-2j, 8-1j, -1+3j],
    [-5+3j, 8-1j, -3-3j, 2+2j],
    [6-3j, -1+3j, 2+2j, 8+14j]
], dtype=complex)
```

```python
b = np.array([15 - 35j, 2 + 10j, -2 - 34j, 8 + 14j], dtype=complex)

# Solve the system
x = gauss_elimination_complex(A, b)

# 출력
for i, val in enumerate(x, 1):
    print(f"x{i} = {val:.6f}")
```

Result

```
x1 = 2.401493+0.085197j
x2 = 0.174940-4.004007j
x3 = -0.576107+4.089776j
x4 = 0.209850+0.774904j
```

c. 설명

: 복소수 가우스 소거법을 사용한 후, Back Substitution을 사용하여 해를 구하였다.

3.7. (Random matrices and Gauss elimination) Test the function gaussElimin by solving Ax = b, where A is a $n \times n$ random matrix and $b_{ij} = \sum_{j=1}^{n} A_{ij}$ (sum of the elements in the i^{th} row of A). A random matrix can be generated with the rand function in the numpy.random module:

a. 문제 내용

Problem 3.7 *(Random matrices and Gauss elimination)* [Python only]

Test the function `gaussElimin` by solving $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a $n \times n$ random matrix and $b_i = \sum_{j=1}^{n} A_{ij}$ (sum of the elements in the i th row of $\mathbf{A}$). A random matrix can be generated with the rand function in the numpy.random module:

```
from numpy.random import ran
a = rand(n,n)
```

The solution should be $x = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^{\top}$. Run the program with $n = 200$ or bigger.

b. Python 코드 및 결과

Code

```python
import numpy as np

def gaussElimin(A, b):
    A = A.astype(float)
    b = b.astype(float)
    n = len(b)

    # Forward elimination
    for k in range(n):
        # Pivoting
        max_row = np.argmax(np.abs(A[k:, k])) + k
        if max_row != k:
            A[[k, max_row]] = A[[max_row, k]]
            b[[k, max_row]] = b[[max_row, k]]
        for i in range(k+1, n):
            factor = A[i, k] / A[k, k]
            A[i, k:] -= factor * A[k, k:]
            b[i] -= factor * b[k]

    # Back substitution
    x = np.zeros(n)
    for i in reversed(range(n)):
        x[i] = (b[i] - np.dot(A[i, i+1:], x[i+1:])) / A[i, i]

    return x

```

```python
# 테스트용 코드
def test_gaussElimin(n=200, seed=42):
    np.random.seed(seed)  # 재현성을 위해 시드 고정
    A = np.random.rand(n, n)
    b = np.sum(A, axis=1)  # 각 행의 합 -> x = [1, 1, ..., 1]이 해가 되게 함
    x = gaussElimin(A.copy(), b.copy())

    # 결과 비교
    expected = np.ones(n)
    error = np.linalg.norm(x - expected)
    print(f"오차 = {error:.6e}")
    if error < 1e-6:
        print("정답과 거의 일치합니다.")
    else:
        print("오차가 큽니다. 수치 안정성 확인 필요.")

# 실행
test_gaussElimin(n=200)
```

Result

```
오차 = 5.290811e-13
정답과 거의 일치합니다.
```

c. 설명

: 이 테스트는 b를 "특별하게" 만들어서 해가 모두 1인 벡터가 되도록 설계함.

1) 랜덤 행렬 생성 (테스트 입력 구성)

   : A = np.random.rand(n, n), b = np.sum(A, axis=1)

2) 가우스 소거법 수행 (gaussElimin)

3) 해 비교 및 오차 출력

   : 정답 벡터(expected = np.ones(n)), 실제 해와의 차이 계산 (error = np.linalg.norm(x - expected))

3.8. **(Matrix condition number) Write a function that returns the condition number of a matrix based on the euclidean norm. Test the function by computing the condition number of the ill-conditioned matrix. Use the function inv(A) in numpy.linalg to determine the inverse of A.**

a. 문제 내용

Problem 3.8 *(Matrix condition number)* [Python only]

Write a function that returns the condition number of a matrix based on the euclidean norm. Test the function by computing the condition number of the ill-conditioned matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 9 & 16 \\ 4 & 9 & 16 & 25 \\ 9 & 16 & 25 & 36 \\ 16 & 25 & 36 & 49 \end{bmatrix}$$

Use the function `inv(A)` in `numpy. linalg` to determine the inverse of $\mathbf{A}$.

b. Python 코드 및 결과

Code

```python
import numpy as np

# 주어진 ill-conditioned matrix
A = np.array([
    [1, 4, 9, 16],
    [4, 9, 16, 25],
    [9, 16, 25, 36],
    [16, 25, 36, 49]
], dtype=float)

# (1) Condition number 함수 직접 구현
def condition_number(A):
    A_norm = np.linalg.norm(A, 2)
    A_inv = np.linalg.inv(A)
    A_inv_norm = np.linalg.norm(A_inv, 2)
    cond = A_norm * A_inv_norm
    return A_norm, A_inv, A_inv_norm, cond

# (2) 함수 호출
A_norm, A_inv, A_inv_norm, cond_custom = condition_number(A)
cond_numpy = np.linalg.cond(A, 2)
```

Result

```
=== Condition Number Analysis ===
Matrix A:
[[ 1.  4.  9. 16.]
 [ 4.  9. 16. 25.]
 [ 9. 16. 25. 36.]
 [16. 25. 36. 49.]]

2-Norm of A:
9.01480e+01

Inverse of A:
[[ 4.69124961e+13 -1.40737488e+14  1.40737488e+14 -4.69124961e+13]
 [-1.40737488e+14  4.22212465e+14 -4.22212465e+14  1.40737488e+14]
 [ 1.40737488e+14 -4.22212465e+14  4.22212465e+14 -1.40737488e+14]
 [-4.69124961e+13  1.40737488e+14 -1.40737488e+14  4.69124961e+13]]

2-Norm of A-inverse:
9.38250e+14

Condition Number (Custom):
8.45813e+16

Condition Number (NumPy built-in):
cond(A) = 5.34183e+16
```

c. 코드 설명

- $\kappa(A) = \|A\|_2 \cdot \|A^{-1}\|_2$
- $\|A\|_2 = \sigma_{max}(A) = \sqrt{\lambda_{max}(A^T A)},\ \|A^{-1}\|_2 = \frac{1}{\sigma_{min}(A)} = \frac{1}{\sqrt{\lambda_{min}(A^T A)}}$ 방법을 통해 연산 복잡도를 더 간단히 할 수 있으나, 이번 과제에서는 역행렬을 직접 구하여 보는 쪽으로 코드를 작성하였다.

3.9. (Solving a linear system with a tridiagonal coefficient matrix) Solve the tridiagonal equations Ax = b by Doolittle's decomposition method, where:

a. 문제 내용

Problem 3.9 *(Solving a linear system with a tridiagonal coefficient matrix)*

Solve the tridiagonal equations $\mathbf{Ax} = \mathbf{b}$ by Doolittle's decomposition method, where

$$\mathbf{A} = \begin{bmatrix} 6 & 2 & 0 & 0 & 0 \\ -1 & 7 & 2 & 0 & 0 \\ 0 & -2 & 8 & 2 & 0 \\ 0 & 0 & 3 & 7 & -2 \\ 0 & 0 & 0 & 3 & 5 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 2 \\ -3 \\ 4 \\ -3 \\ 1 \end{bmatrix}$$

b. Python 코드 및 결과

Code

```python
from fractions import Fraction
import numpy as np

# LU 분해 for tridiagonal matrix (in-place + L/U 분리 출력)
def LUdecomp3_full(c_in, d_in, e_in):
    n = len(d_in)

    # 원본 보존을 위해 복사
    c = c_in.copy()
    d = d_in.copy()
    e = e_in.copy()

    # LU 병합 저장용
    L = np.eye(n, dtype=object)
    U = np.zeros((n, n), dtype=object)

    for k in range(n):
        if k == 0:
            U[k][k] = d[k]
            U[k][k+1] = e[k]
        else:
            L[k][k-1] = c[k-1] / d[k-1]
            d[k] = d[k] - L[k][k-1] * e[k-1]
            U[k-1][k] = e[k-1]
            U[k][k] = d[k]

    return L, U
```

```python
# 간단한 버전 (값만 변경)
def LUdecomp3(c, d, e):
    n = len(d)
    for k in range(1, n):
        lam = c[k-1] / d[k-1]
        d[k] = d[k] - lam * e[k-1]
        c[k-1] = lam
    return c, d, e

# LU 해법
def LUsolve3(c, d, e, b):
    n = len(d)
    for k in range(1, n):
        b[k] -= c[k-1] * b[k-1]
    b[n-1] = b[n-1] / d[n-1]
    for k in range(n-2, -1, -1):
        b[k] = (b[k] - e[k] * b[k+1]) / d[k]
    return b

# 입력 (분수로 표현)
c = [Fraction(-1), Fraction(-2), Fraction(3), Fraction(3)]  # sub-diagonal
d = [Fraction(6), Fraction(7), Fraction(8), Fraction(7), Fraction(5)]  # main
e = [Fraction(2), Fraction(2), Fraction(2), Fraction(-2)]  # super
b = [Fraction(2), Fraction(-3), Fraction(4), Fraction(-3), Fraction(1)]  # RHS

# L, U 행렬 분해
L, U = LUdecomp3_full(c.copy(), d.copy(), e.copy())
```

```python
# LU 분해 (값 반영)
c, d, e = LUdecomp3(c, d, e)
x = LUsolve3(c, d, e, b.copy())

# 출력
print("=== L 행렬 ===")
for row in L:
    print([str(val) for val in row])

print("\n=== U 행렬 ===")
for row in U:
    print([str(val) for val in row])

print("\n=== 해 x (분수 기반) ===")
for i, xi in enumerate(x):
    print(f"x[{i}] = {xi}  (approx. {float(xi):.4f})")
```

Result

```
=== L 행렬 ===
['1', '0', '0', '0', '0']
['-1/6', '1', '0', '0', '0']
['0', '-3/11', '1', '0', '0']
['0', '0', '33/94', '1', '0']
['0', '0', '0', '141/296', '1']

=== U 행렬 ===
['6', '2', '0', '0', '0']
['0', '22/3', '2', '0', '0']
['0', '0', '94/11', '2', '0']
['0', '0', '0', '296/47', '-2']
['0', '0', '0', '0', '881/148']

=== 해 x (분수 기반) ===
x[0] = 1/2  (approx. 0.5000)
x[1] = -1/2  (approx. -0.5000)
x[2] = 1/2  (approx. 0.5000)
x[3] = -1/2  (approx. -0.5000)
x[4] = 1/2  (approx. 0.5000)
```

c. 설명

sol)

$$[A|b] = \left(\begin{array}{ccccc|ccccc} 6 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 7 & 2 & 0 & 0 & \square & 1 & 0 & 0 & 0 \\ 0 & -2 & 8 & 2 & 0 & \square & \square & 1 & 0 & 0 \\ 0 & 0 & 3 & 7 & -2 & \square & \square & \square & 1 & 0 \\ 0 & 0 & 0 & 3 & 5 & \square & \square & \square & \square & 1 \end{array}\right) \rightarrow \left(\begin{array}{ccccc|ccccc} 6 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{22}{3} & 2 & 0 & 0 & -1/6 & 1 & 0 & 0 & 0 \\ 0 & -2 & 8 & 2 & 0 & 0 & \square & 1 & 0 & 0 \\ 0 & 0 & 3 & 7 & -2 & 0 & \square & \square & 1 & 0 \\ 0 & 0 & 0 & 3 & 5 & 0 & \square & \square & \square & 1 \end{array}\right)$$

$$\rightarrow \left(\begin{array}{ccccc|ccccc} 6 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{22}{3} & 2 & 0 & 0 & -\frac{1}{6} & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{94}{11} & 2 & 0 & 0 & -\frac{3}{11} & 1 & 0 & 0 \\ 0 & 0 & 3 & 7 & -2 & 0 & 0 & \square & 1 & 0 \\ 0 & 0 & 0 & 3 & 5 & 0 & 0 & \square & \square & 1 \end{array}\right) \rightarrow \left(\begin{array}{ccccc|ccccc} 6 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{22}{3} & 2 & 0 & 0 & -\frac{1}{6} & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{94}{11} & 2 & 0 & 0 & -\frac{3}{11} & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{296}{47} & -2 & 0 & 0 & \frac{33}{94} & 1 & 0 \\ 0 & 0 & 0 & 3 & 5 & 0 & 0 & 0 & \square & 1 \end{array}\right)$$

$$\rightarrow \left(\begin{array}{ccccc|ccccc} 6 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{22}{3} & 2 & 0 & 0 & -\frac{1}{6} & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{94}{11} & 2 & 0 & 0 & -\frac{3}{11} & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{296}{47} & -2 & 0 & 0 & \frac{33}{94} & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{881}{148} & 0 & 0 & 0 & \frac{141}{296} & 1 \end{array}\right)$$

: 위 과정은 Doolittle’s Method를 Tridiagonal Matrix LU decomposition 방법에 최적화하여 LU 분해를 한 과정이다. 이후는 Forward Substitution, Backward Substitution을 통해 해를 구했다.

3.10. (Solving a linear system with a symmetric and tridiagonal coefficient matrix) Solve the symmetric, tridiagonal equations with n = 10:

a. 문제 내용

Problem 3.10 *(Solving a linear system with a symmetric and tridiagonal coefficient matrix)*

Solve the symmetric, tridiagonal equations

$$
\begin{aligned}
4x_1 - x_2 &= 9 \\
-x_{i-1} + 4x_i - x_{i+1} &= 5, \quad i = 2, \ldots, n-1 \\
-x_{n-1} + 4x_n &= 5
\end{aligned}
$$

with $n = 10$.

b. Python 코드 및 결과

Code

```python
from fractions import Fraction

# 삼대각 LU 분해 함수 (Doolittle 방식)
def LUdecomp3(c, d, e):
    n = len(d)
    for k in range(1, n):
        lam = c[k-1] / d[k-1]
        d[k] = d[k] - lam * e[k-1]
        c[k-1] = lam
    return c, d, e

# 전방 및 후방 대입 함수
def LUsolve3(c, d, e, b):
    n = len(d)

    # Forward substitution: Ly = b
    for k in range(1, n):
        b[k] -= c[k-1] * b[k-1]

    # Backward substitution: Ux = y
    b[n-1] = b[n-1] / d[n-1]
    for k in range(n-2, -1, -1):
        b[k] = (b[k] - e[k] * b[k+1]) / d[k]

    return b

# 문제 3.10 설정 (n = 10)
n = 10
c = [Fraction(-1)] * (n - 1)  # 하삼각 부분
d = [Fraction(4)] * n          # 주대각
e = [Fraction(-1)] * (n - 1)  # 상삼각 부분
b = [Fraction(9)] + [Fraction(5)] * (n - 1)  # 우변 벡터

# LU 분해 및 풀이
c, d, e = LUdecomp3(c, d, e)
x = LUsolve3(c, d, e, b)

# 결과 출력
print("=== 문제 3.10 해 (분수 기반) ===")
for i, xi in enumerate(x):
    print(f"x[{i+1}] = {xi}  (approx. {float(xi):.6f})")
```

Result

```
=== 문제 3.10 해 (분수 기반) ===
x[1] = 1638769/564719  (approx. 2.901919)
x[2] = 1472605/564719  (approx. 2.607677)
x[3] = 1428056/564719  (approx. 2.528790)
x[4] = 1416024/564719  (approx. 2.507484)
x[3] = 1428056/564719  (approx. 2.528790)
x[4] = 1416024/564719  (approx. 2.507484)
x[4] = 1416024/564719  (approx. 2.507484)
x[5] = 1412445/564719  (approx. 2.501147)
x[6] = 1410161/564719  (approx. 2.497102)
x[7] = 1404604/564719  (approx. 2.487262)
x[8] = 1384660/564719  (approx. 2.451945)
x[9] = 1310441/564719  (approx. 2.320519)
x[10] = 1033509/564719  (approx. 1.830130)
```

c. 설명

: Tridiagonal Matrix LU Decomposition 알고리즘 최적화를 위해 Hard coding을 하였다.

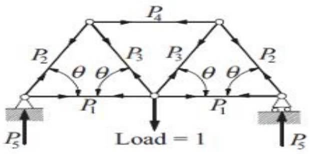
$$d_k = d_k - (c_{k-1}/d_{k-1})e_{k-1}$$

$$c_{k-1} = c_{k-1}/d_{k-1}$$

3.11. (Analysis of truss deformation) $P_i(i = 1, 2, \ldots, 5)$를 구하는 프로그램 코드를 짜서 P_i를 구하여라:

a. Python 코드

Problem 3.11 *(Analysis of truss deformation)*



The force formulation of the symmetric truss shown results in the joint equilibrium equations

$$\begin{bmatrix} c & 1 & 0 & 0 & 0 \\ 0 & s & 0 & 0 & 1 \\ 0 & 0 & 2s & 0 & 0 \\ 0 & -c & c & 1 & 0 \\ 0 & s & s & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

where $s = \sin\theta$, $c = \cos\theta$, and P_i are the unknown forces. Write a program that computes the forces, given the angle θ. Run the program with $\theta = 53°$.

b. Python 코드 및 결과

Code

```python
import numpy as np
import math

# 각도 설정
theta_deg = 53
theta_rad = math.radians(theta_deg)

s = math.sin(theta_rad)
c = math.cos(theta_rad)

# 계수 행렬 A (5x5)
A = np.array([
    [ c,  1,   0,  0,  0],
    [ 0,  s,   0,  0,  1],
    [ 0,  0, 2*s,  0,  0],
    [ 0, -c,   c, 1,  0],
    [ 0,  s,   s, 0,  0]
], dtype=float)

# 우변 벡터 b
b = np.array([0, 0, 1, 0, 0], dtype=float)

# 선형 시스템 Ax = b 풀기
P = np.linalg.solve(A, b)

# 결과 출력
print("=== 문제 3.11: Truss 구조 해석 결과 ===")
for i, p in enumerate(P, start=1):
    print(f"P{i} = {p:.6f}")
```

Result
`` === 문제 3.11: Truss 구조 해석 결과 === P1 = 1.040299 P2 = -0.626068 P3 = 0.626068 P4 = -0.753554 P5 = 0.500000 ```

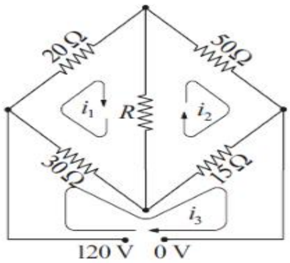
c. 설명

: 이번 문제에서는 이미 계수행렬이 정해졌으므로, numpy의 linalg.solve 메서드를 사용하여 해를 구하였다. 만약 Hard-coding을 수행한다면, 이전 문제처럼 Doolittle's Method의 Tridiagonal Matrix LU decomposition 최적화 알고리즘을 사용하여 구현할 수 있을 것이다.

3.12. (Electrical circuit analysis) $i_j(j = 1, 2, 3)$을 구하는 프로그램 코드를 짜서 i_j를 구하여라:

a. 문제 내용

Problem 3.12 *(Electrical circuit analysis)*



The electrical network shown can be viewed as consisting of three loops. Applying Kirchoff's law ($\sum$voltage drops $= \sum$voltage sources) to each loop yields the following equations for the loop currents i_1, i_2 and i_3:

$$(50 + R)i_1 - Ri_2 - 30i_3 = 0$$

$$-Ri_1 + (65 + R)i_2 - 15i_3 = 0$$

$$-30i_2 - 15i_2 + 45i_3 = 120$$

1. Determine the loop currents i_1, i_2 and i_3 for $R = 5$ Ohm.
2. Determine the loop currents i_1, i_2 and i_3 for $R = 10$ Ohm.
3. Determine the loop currents i_1, i_2 and i_3 for $R = 20$ Ohm.

b. Python 코드 및 결과

Code

```python
import numpy as np

def solve_using_numpy_cholesky(R):
    A = np.array([
        [50 + R, -R, -30],
        [-R, 65 + R, -15],
        [0, -45, 45]
    ], dtype=float)

    b = np.array([0, 0, 120], dtype=float)

    # Cholesky 분해
    L = np.linalg.cholesky(A)

    # Forward & Backward Substitution
    y = np.linalg.solve(L, b)
    x = np.linalg.solve(L.T, y)

    return x

# 테스트
for R in [5, 10, 20]:
    x = solve_using_numpy_cholesky(R)
    print(f"R = {R} Ohm -> i1 = {x[0]:.4f} A, i2 = {x[1]:.4f} A, i3 = {x[2]:.4f} A")
```

Result
R = 5 Ohm -> i1 = 3.0526 A, i2 = 1.3684 A, i3 = 5.3684 A R = 10 Ohm -> i1 = 3.0000 A, i2 = 1.5000 A, i3 = 5.5000 A R = 20 Ohm -> i1 = 2.9231 A, i2 = 1.6923 A, i3 = 5.6923 A

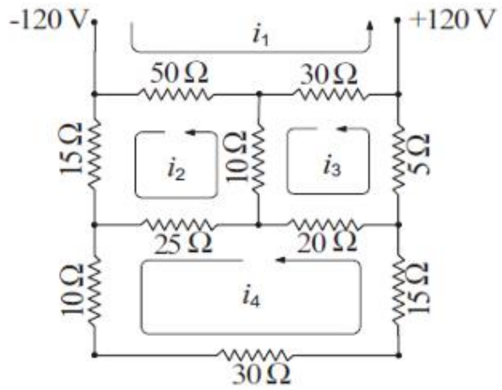
c. 설명

1) Symmetric 하지 않으므로, Doolittle 방식을 사용.
2) 이 문제에서는 np.linalg 에서 제공하는 solve 메서드를 사용.

3.13. (Electrical circuit analysis) $i_j (j = 1, 2, 3, 4)$을 구하는 프로그램 코드를 짜서 i_j를 구하여라:

a. 문제 내용

Problem 3.13 *(Electrical circuit analysis)*



Determine the loop currents i_1 to i_4 in the electrical network shown.

b. Python 코드 및 결과

Code

```python
import numpy as np

# 주어진 행렬 A
A = np.array([
    [80, -50, -30,   0],
    [-50, 100, -10, -25],
    [-30, -10, 65, -20],
    [0, -25, -20, 100]
], dtype=float)

# 우변 벡터 b
b = np.array([120, 0, 0, 0], dtype=float)

def is_positive_definite(A):
    # Leading principal minors 양수인지 확인
    for k in range(1, A.shape[0] + 1):
        if np.linalg.det(A[:k, :k]) <= 0:
            return False
    return True

def cholesky_decomposition(A):
    n = A.shape[0]
    L = np.zeros_like(A)

    for i in range(n):
        for j in range(i+1):
            sum_val = sum(L[i][k] * L[j][k] for k in range(j))
            if i == j:
                L[i][j] = (A[i][i] - sum_val) ** 0.5
            else:
                L[i][j] = (A[i][j] - sum_val) / L[j][j]
    return L
```

```python
def forward_substitution(L, b):
    n = L.shape[0]
    y = np.zeros_like(b)
    for i in range(n):
        y[i] = (b[i] - np.dot(L[i,:i], y[:i])) / L[i,i]
    return y

def backward_substitution(U, y):
    n = U.shape[0]
    x = np.zeros_like(y)
    for i in reversed(range(n)):
        x[i] = (y[i] - np.dot(U[i,i+1:], x[i+1:])) / U[i,i]
    return x

# 1. Positive definite 확인
if is_positive_definite(A):
    print("Positive definite -> Cholesky decomposition 사용")
    L = cholesky_decomposition(A)
    y = forward_substitution(L, b)
    x = backward_substitution(L.T, y)
else:
    print("Not positive definite -> Doolittle decomposition 사용")
    L, U = doolittle_decomposition(A)
    y = forward_substitution(L, b)
    x = backward_substitution(U, y)

# 결과 출력
for i, val in enumerate(x, 1):
    print(f"i{i} = {val:.6f}")
```

Result
<pre>Positive definite -> Cholesky decomposition 사용 i1 = 4.182395 i2 = 2.664552 i3 = 2.712133 i4 = 1.208565</pre>

c. 설명

① $-120 + 50(i_1 - i_2) + 30(i_1 - i_3) = 120$

$\rightarrow 80 i_1 - 50 i_2 - 30 i_3 = 240$

② $50(i_2 - i_1) + 15 i_2 + 25(i_2 - i_4) + 10(i_2 - i_3) = 0$

$\rightarrow -50 i_1 + 100 i_2 - 10 i_3 - 25 i_4 = 0$

③ $30(i_3 - i_1) + 10(i_3 - i_2) + 20(i_3 - i_4) + 5 i_3 = 0$

$\rightarrow -30 i_1 - 10 i_2 + 65 i_3 - 20 i_4 = 0$

④ $25(i_4 - i_2) + 55 i_4 + 20(i_4 - i_3) = 0$

$\rightarrow \quad -25 i_2 - 20 i_3 + 100 i_4 = 0$

$$\Rightarrow \begin{pmatrix} 80 & -50 & -30 & 0 \\ -50 & 100 & -10 & -25 \\ -30 & -10 & 65 & -20 \\ 0 & -25 & -20 & 100 \end{pmatrix} \begin{pmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{pmatrix} = \begin{pmatrix} 120 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

이후:

1) Symmetric 여부 & Positive Definite 판정 -> (O)
2) Choleski Method 사용하여 LU 분해 -> Forward&Backward Substitution 사용하여 해를 도출.