Nguyễn Minh Đức - ITITIU21045 - Lab 3 - TMCLab

Q1. a) $10x_1 + 2x_2 - x_3 = 27$ (1)

$-3x_1 - 6x_2 + 2x_3 = -61.5$ (2)

$x_1 + x_2 + 5x_3 = -21.5$ (3)

Times $\frac{-3}{10}$ to (1) and subtract from (2)

$10x_1 + 2x_2 - x_3 = 27$ (1)

$-5.4x_2 + 1.7x_3 = -53.4$ (2)

$x_1 + x_2 + 5x_3 = -21.5$ (3)

Times $\frac{1}{10}$ to (1) and subtract from (3)

$10x_1 + 2x_2 - x_3 = 27$ (1)

$-5.4x_2 + 1.7x_3 = -53.4$ (2)

$0.8x_2 + 5.1x_3 = -24.2$ (3)

Times $\frac{-0.8}{5.4}$ to (2) and subtract from (3)

$10x_1 + 2x_2 - x_3 = 27$ (1)

$-5.4x_2 + 1.7x_3 = -53.4$ (2)

$5.35185x_3 = \cancel{16.28889}$ $-32.1$ (3)

$\Rightarrow x_3 = \cancel{3.0436}$ $-6$

$x_2 = \cancel{10.84706}$ $8$

$x_1 = \cancel{0.83495}$ $0.5$

b) Substitute ~~the~~ $x_1, x_2, x_3$ into (1), (2), (3) we get answer ~~resp~~ respectively:

(1) = 27~~.00002~~

(2) = -61.5~~0001~~

(3) = - 21.5

(3)/ = 16

Q2a) $8x_1 + 2x_2 - 2x_3 = 8$ (1)

$10x_1 + 2x_2 + 4x_3 = 16$ (2)

$12x_1 + 2x_2 + 2x_3 = 16$ (3)

Transform into matrix form

$$\left[\begin{array}{ccc:c} 8 & 2 & -2 & 8 \\ 10 & 2 & 4 & 16 \\ 12 & 2 & 2 & 16 \end{array}\right] \rightarrow \left[\begin{array}{ccc:c} 12 & 2 & 2 & 16 \\ 10 & 2 & 4 & 16 \\ 8 & 2 & -2 & 8 \end{array}\right]$$

$$\begin{array}{c} \xrightarrow{R_2 + \frac{-10}{12} R_1} \\ \xrightarrow{R_3 + \frac{-8}{12} R_1} \end{array} \left[\begin{array}{ccc:c} 12 & 2 & 2 & 16 \\ 0 & 1/3 & 7/3 & 8/3 \\ 0 & 2/3 & -10/3 & -8/3 \end{array}\right] \xrightarrow{R_3 - 2R_2} \left[\begin{array}{ccc:c} 12 & 2 & 2 & 16 \\ 0 & 1/3 & 7/3 & 8/3 \\ 0 & 0 & -8 & -8 \end{array}\right]$$

We have : $12x_1 + 2x_2 + 2x_3 = 16$

$$\left.\begin{aligned} 12x_1 + 2x_2 + 2x_3 &= 16 \\ \frac{1}{3}x_2 + \frac{7}{3}x_3 &= \frac{8}{3} \\ -8x_3 &= -8 \end{aligned}\right\} \Rightarrow \begin{cases} x_1 = 1 \\ x_2 = 1 \\ x_3 = 1 \end{cases}$$

Substituting into original equations give the same values.

Q3. $2x_1 + x_2 - x_3 = 2$

$5x_1 + 2x_2 + 2x_3 = 9$

$3x_1 + x_2 + x_3 = 5$

$$\xrightarrow{\text{Matrix}} \left[\begin{array}{ccc:c} 2 & 1 & -1 & 2 \\ 5 & 2 & 2 & 9 \\ 3 & 1 & 1 & 5 \end{array}\right]$$

$$\xrightarrow[R_3 - \frac{3}{2}R_1]{R_2 - \frac{5}{2}R_1} \left[\begin{array}{ccc:c} 2 & 1 & -1 & 2 \\ 0 & -0.5 & 4.5 & 4 \\ 0 & -0.5 & 2.5 & 2 \end{array}\right] \xrightarrow{R_1/2} \xrightarrow{R_3 - R_2} \left[\begin{array}{ccc:c} 1 & 0.5 & -0.5 & 1 \\ 0 & -0.5 & 4.5 & 4 \\ 0 & 0 & -2 & -2 \end{array}\right]$$

$$\xrightarrow[R_3/-2]{R_2/-0.5} \left[\begin{array}{ccc:c} 1 & 0.5 & -0.5 & 1 \\ 0 & 1 & -9 & -8 \\ 0 & 0 & 1 & 1 \end{array}\right] \xrightarrow[R_2 + 9R_3]{R_1 - 0.5R_2} \left[\begin{array}{ccc:c} 1 & 0 & 4 & 5 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right]$$

~~$R_2 - 19R_3$~~ ~~$R_2 - 13.5R_3$~~

$$\xrightarrow{R_1 - 4R_3} \left[\begin{array}{ccc:c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right] \Rightarrow \begin{aligned} x_1 &= 1 \\ x_2 &= 1 \\ x_3 &= 1 \end{aligned}$$

(True when substitute back into original equations)

Q4: $15c_1 - 3c_2 - c_3 = 3300$

$-3c_1 + 18c_2 - 6c_3 = 1200 \Rightarrow$

$-4c_1 - c_2 + 12c_3 = 2400$

$$\Rightarrow c_1 = \frac{3300 + 3c_2 + c_3}{15};\ c_2 = \frac{1200 + 3c_1 + 6c_3}{18};\ c_3 = \frac{2400 + 4c_1 + c_2}{12}$$

| Ite | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $c_1$ | 0 | 220 | 259.46 | 281.01 | 283.7 | 284.408 |
| $c_2$ | 0 | 103.(3) | 203.89 | 214.66 | 217.8 | 212.308 |
| $c_3$ | 0 | 281.94 | 303.48 | 311.56 | ~~3217.8~~<br>312.72 | 312.495 |

→ After 6 iterations, $\varepsilon_s = 5\%$, we can conclude $c_1 = 284.408$; $c_2 = 212.308$; $c_3 = 312.495$

% diff ① = 1.15% ; ② = 3.65% ; ③ = 0%

```
#Nguyễn Minh Đức - ITITIU21045_TMCLab03_Q1
def gauss_elimination(A, B):
    a = len(B)
    for pivot in range(a):

        # Partial pivoting
        max_row = pivot
        for i in range(pivot + 1, a):
            if abs(A[i][pivot]) > abs(A[max_row][pivot]):
                max_row = i
        A[pivot], A[max_row] = A[max_row], A[pivot]
        B[pivot], B[max_row] = B[max_row], B[pivot]

        # Make the pivot element 1
        pivot_element = A[pivot][pivot]
        for j in range(pivot, a):
            A[pivot][j] /= pivot_element
        B[pivot] /= pivot_element

        # Eliminate other rows
        for i in range(a):
            if i != pivot:
                factor = A[i][pivot]
                for j in range(pivot, a):
                    A[i][j] -= factor * A[pivot][j]
                B[i] -= factor * B[pivot]
    return B


# Coefficients of the linear equations
A = [[10, 2, -1], [-3, -6, 2], [1, 1, 5]]
# Constants on the right-hand side
B = [27, -61.5, -21.5]
# Solve the system of linear equations
solution = gauss_elimination(A, B)
# Display the solution
for i, sol in enumerate(solution):
    print(f'x_{i+1} = {sol}')
```

```
x_1 = 0.5000000000000002
x_2 = 7.999999999999999
x_3 = -6.0
```

+ Code + Text

```
def gauss_elimination_partial_pivoting(A, B):
    a = len(B)
    for pivot in range(a):

        # Partial pivoting
        max_row = pivot
        for i in range(pivot + 1, a):
            if abs(A[i][pivot]) > abs(A[max_row][pivot]):
                max_row = i
        A[pivot], A[max_row] = A[max_row], A[pivot]
        B[pivot], B[max_row] = B[max_row], B[pivot]

        # Make the pivot element 1
        pivot_elem = A[pivot][pivot]
        for j in range(pivot, a):
            A[pivot][j] /= pivot_elem
        B[pivot] /= pivot_elem

        # Eliminate other rows
        for i in range(a):
            if i != pivot:
                factor = A[i][pivot]
                for j in range(pivot, a):
                    A[i][j] -= factor * A[pivot][j]
                B[i] -= factor * B[pivot]
    return B


# Coefficients of the linear equations
A = [[8, -2, 2], [10, 2, 4], [12, 2, 2]]
# Constants on the right-hand side
B = [8, 16, 16]
# Solve the system of linear equations
solution = gauss_elimination_partial_pivoting(A, B)
# Display the solution
for i, sol in enumerate(solution):
    print(f'x_{i+1} = {sol}')
```

```
x_1 = 0.9999999999999999
x_2 = 1.0
x_3 = 1.0000000000000004
```

```
#Nguyễn Minh Đức - ITITIU21045_TMCLab03_Q3
import numpy as np

# Coefficient matrix A
A = np.array([[2, 1, -1],
              [5, 2, 2],
              [3, 1, 1]])

# Right-hand side vector B
B = np.array([2, 9, 5])

# Augment the coefficient matrix with the right-hand side vector
augmented_matrix = np.column_stack((A, B))

# Perform Gauss-Jordan elimination
a = len(augmented_matrix)
for i in range(a):

    # Find the pivot row
    pivot_row = i
    for j in range(i, a):
        if abs(augmented_matrix[j, i]) > abs(augmented_matrix[pivot_row, i]):
            pivot_row = j

    # Check if the pivot element is close to zero
    if abs(augmented_matrix[pivot_row, i]) < 1e-10:
        break
    augmented_matrix[i], augmented_matrix[pivot_row] = augmented_matrix[pivot_row], augmented_matrix[i]

    # Make the pivot element 1
    pivot_elem = augmented_matrix[i, i]
    augmented_matrix[i] = (augmented_matrix[i] / pivot_elem)

    # Eliminate other rows
    for j in range(a):
        if j != i:
            factor = augmented_matrix[j, i]
            augmented_matrix[j] -= factor * augmented_matrix[i]

# Extract the solution
solution = augmented_matrix[:, -1]

# Display the solution
for i, sol in enumerate(solution):
    print(f'x_{i+1} = {sol}')
```

```
x_1 = 1
x_2 = 2
x_3 = 0
```

```
#Nguyễn Minh Đức - ITITIU21045_TMCLab03_Q4
def gauss_seidel(A, B, initial_guess, max_iterations, tolerance):
    a = len(B)
    x = initial_guess.copy()
    for _ in range(max_iterations):
        x_new = x.copy()
        for i in range(a):
            s1 = sum(A[i][j] * x_new[j] for j in range(i))
            s2 = sum(A[i][j] * x[j] for j in range(i + 1, a))
            x_new[i] = (B[i] - s1 - s2) / A[i][i]
        max_diff = max(abs((x_new[i] - x[i]) / x_new[i]) for i in range(a))
        x = x_new
        if max_diff < tolerance:
            return x
    return x
```

```
# Coefficients of the linear equations
A = [[15, -3, -1],
     [-3, 18, -6],
     [-4, -1, 12]]

# Constants on the right-hand side
B = [3300, 1200, 2400]

# Initial guess
initial_guess = [0, 0, 0]

# Maximum number of iterations
max_iterations = 1000

# Tolerance (5%)
tolerance = 0.05

# Solve the system of linear equations using Gauss-Seidel
solution = gauss_seidel(A, B, initial_guess, max_iterations, tolerance)

# Display the solution
for i, sol in enumerate(solution):
    print(f'c_{i+1} = {sol}')
```

```
c_1 = 283.7028230420921
c_2 = 217.8033174275813
c_3 = 312.71788413299583
```