Nama : Naufal Izzuddin Taufik

NIM : 21120122140102

Mata Kuliah : Metode Numerik B

TUGAS IMPLEMENTASI SISTEM PERSAMAAN LINEAR

Link Github:

https://github.com/naufalizzuddin/Tugas-Implementasi-Sistem-Persamaan-Linear

A. MATRIKS BALIKAN

Matriks balikan adalah konsep dalam aljabar linear yang merujuk pada matriks yang dapat "membalik" operasi perkalian. Dalam konteks matematika, jika kita memiliki sebuah matriks A, matriks balikan dari A (biasanya disimbolkan sebagai A^{-1} adalah matriks yang ketika dikalikan dengan matriks A akan menghasilkan matriks identitas.

Implementasi *Source Code*:

```
def transpose(matrix):
    return [[matrix[j][i] for j in range(len(matrix))] for i in
range(len(matrix[0]))]

def matrix_multiplication(matrix1, matrix2):
    return [[sum(a*b for a,b in zip(X_row,Y_col)) for Y_col in
transpose(matrix2)] for X_row in matrix1]

def matrix_inverse(matrix):
    n = len(matrix)
    identity = [[1 if i == j else 0 for j in range(n)] for i in
range(n)]

    for i in range(n):
        factor = matrix[i][i]
        for j in range(n):
            matrix[i][j] /= factor
            identity[i][j] /= factor
        for k in range(i+1, n):
            factor = matrix[k][i]
            for j in range(n):
                matrix[k][j] -= factor * matrix[i][j]
                identity[k][j] -= factor * identity[i][j]

    for i in range(n-1, -1, -1):
        for k in range(i-1, -1, -1):
            factor = matrix[k][i]
            for j in range(n):
                matrix[k][j] -= factor * matrix[i][j]
                identity[k][j] -= factor * identity[i][j]

    return identity
```

```
def solve_linear_equations(A, b):
    A_inv = matrix_inverse(A)
    x = matrix_multiplication(A_inv, [[elem] for elem in b])
    return [elem[0] for elem in x]

# Masukkan koefisien matriks A dan vektor b
A = [[2, 1, -1],
     [1, 1, -1],
     [-1, -1, 2]]

b = [1, 1, -2]

# Hitung solusi x
x = solve_linear_equations(A, b)
```

Kode ini saya implementasikan menggunakan bahasa Python yang dimana *source code* tersebut dirancang dan akan di-*testing* menggunakan aplikasi Visual Studio Code. Untuk alur jalannya kode adalah sebagai berikut:

1) Fungsi Transpose (`transpose`)

   Fungsi ini akan menmbalikkan baris dan kolom dari matriks (*transpose*). Fungsi ini menggunakan *list comprehension* untuk mengulang setiap baris "j" dan setiap kolom "i" dari matriks yang akan dibalik menjadi baris "i" dan kolom "j".

2) Fungsi Perkalian Matriks (`matrix_multiplication`)

   Fungsi ini akan melakukan perkalian dua matriks. Menggunakan *list comprehension* untuk mengulangi setiap baris dari matriks pertama (matrix1) dan setiap kolom matriks kedua ditransposisikan (matrix2).

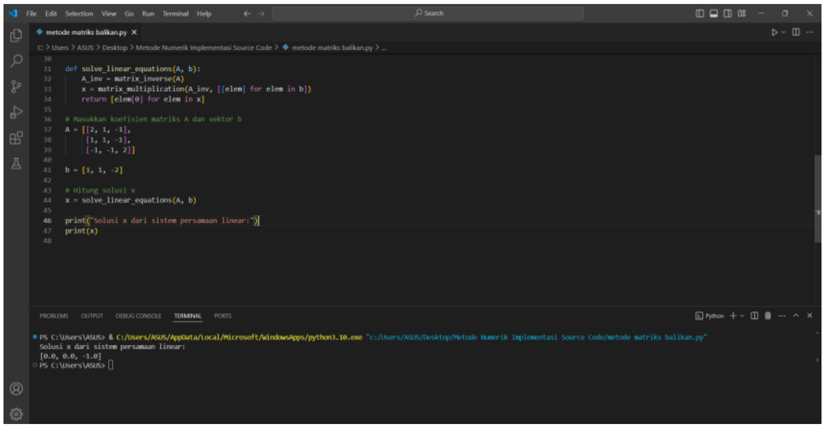
3) Fungsi Invers Matriks (`matrix_inverse`)

   Fungsi ini menghitung invers dari Matriks A menggunakan metode eliminasi Gauss. Menginialisasi matriks identitas dengan dimensi yang sama dengan matriks input. Kemudian dilakukan eliminasi hingga membentuk matriks segitiga atas dan akan disubstitusi mundur untuk menemukan nilai masing-masing variabel.

4) Fungsi Penyelesaian Persamaan Linier (`solve_linear_equations`)

   Fungsi ini akan mengalikan antara hasil A^{-1} dengan B yang dimana akan menghasilkan nilai x dari metode matriks balikan. Matriks A dan B dapat diubah pada kode *testing* untuk melakukan percobaan pada *source code*.

Hasil *Running* dari kode *testing*:



B. DEKOMPOSISI LU GAUSS

Metode Dekomposisi LU Gauss adalah teknik dalam aljabar linear yang memecah matriks koefisien A dari sistem persamaan linear menjadi dua matriks: matriks segitiga bawah L (lower triangular matrix) dan matriks segitiga atas U (upper triangular matrix). Dan akan dikombinasikan dengan metode Eliminasi Gauss.

Implementasi *Source Code*:

```
def lu_decomposition(A):
    """
    Menghitung dekomposisi LU untuk matriks A.
    Asumsi: A adalah matriks persegi dan memiliki dekomposisi LU yang
unik.
    """
    n = len(A)
    L = [[0.0] * n for _ in range(n)]
    U = [[0.0] * n for _ in range(n)]

    # Inisialisasi matriks L dengan diagonal 1
    for i in range(n):
        L[i][i] = 1.0

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

    return L, U
```

```
def solve_lu(L, U, b):
    """
    Menyelesaikan SPL Ax = b menggunakan dekomposisi LU.
    """
    n = len(L)
    y = [0.0] * n
    x = [0.0] * n

    # Forward substitution (Ly = b)
    for i in range(n):
        y[i] = b[i]
        for j in range(i):
            y[i] -= L[i][j] * y[j]

    # Backward substitution (Ux = y)
    for i in range(n - 1, -1, -1):
        x[i] = y[i]
        for j in range(i + 1, n):
            x[i] -= U[i][j] * x[j]
        x[i] /= U[i][i]

    return x

# Contoh penggunaan
A = [[2, -1, 0],
     [-1, 2, -1],
     [0, -1, 2]]
b = [1, 2, 3]

L, U = lu_decomposition(A)
x = solve_lu(L, U, b)

print("Matriks L:")
for row in L:
    print(row)

print("\nMatriks U:")
for row in U:
    print(row)

print("\nSolusi SPL:")
print("x =", x)
```

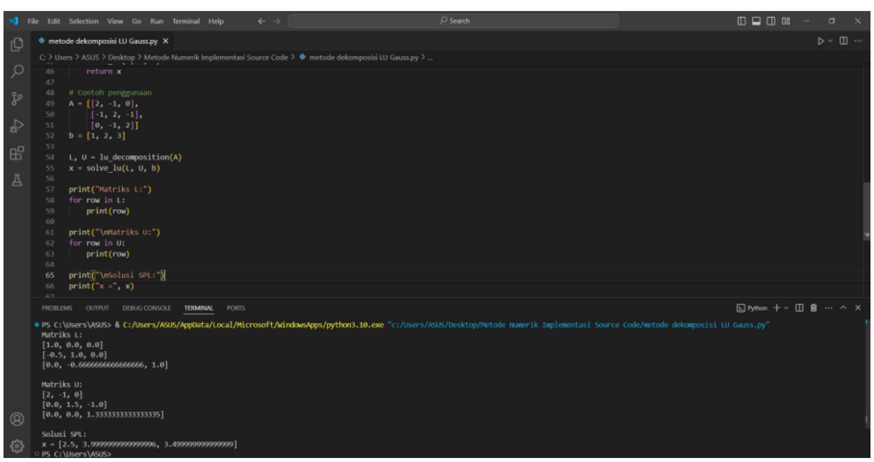
Kode ini saya implementasikan menggunakan bahasa Python yang dimana source code tersebut dirancang dan akan di-testing menggunakan aplikasi Visual Studio Code. Untuk alur jalannya kode adalah sebagai berikut:

1) Fungsi `lu_decomposition`
    - Fungsi ini mengambil matriks A sebagai input.
    - Membuat matriks nol L dan U yang memiliki ukuran sama dengan matriks A.
    - Inisialisasi matriks L dengan diagonal utama 1.

- Melakukan dekomposisi LU dengan iterasi melalui setiap elemen matriks. Pada setiap iterasi:
    - Menentukan elemen diagonal utama U dari matriks A.
    - Mengisi kolom di bawah diagonal utama U dengan elemen matriks A dibagi dengan elemen diagonal utama U untuk mendapatkan elemen matriks L.
    - Mengisi baris di sebelah kanan diagonal utama U dengan elemen matriks A.
    - Memperbarui matriks A dengan memodifikasi elemen-elemennya menggunakan elemen-elemen dari matriks L dan U.
- Mengembalikan matriks L dan U yang telah dihitung.

2) Fungsi `solve_lu`
    - Fungsi ini mengambil matriks L, U, dan vektor b sebagai input.
    - Membuat vektor nol y dan x dengan ukuran yang sama dengan panjang vektor b.
    - Melakukan substitusi maju (forward substitution) untuk menyelesaikan sistem $Ly=b$. Iterasi melalui setiap elemen vektor b dan menghitung nilai-nilai y berdasarkan matriks L dan vektor b.
    - Melakukan substitusi mundur (backward substitution) untuk menyelesaikan sistem $Ux=y$. Iterasi mundur melalui setiap elemen vektor y dan menghitung nilai-nilai x berdasarkan matriks U dan vektor y.

Hasil *running* dari kode *testing*:



C. DEKOMPOSISI CROUT

Dekomposisi Crout adalah salah satu metode untuk melakukan dekomposisi matriks persegi A menjadi dua matriks segitiga bawah L dan segitiga atas U, di mana $A=LU$. Perbedaan utama antara dekomposisi Crout dan dekomposisi LU tradisional adalah pada urutan pengisian elemen-elemen matriks L dan U.

Implementasi *Source Code*:

```
def dekomposisi_crout(A, b):
    n = len(A)
    L = [[0.0] * n for _ in range(n)]
    U = [[0.0] * n for _ in range(n)]

    # Faktorisasi matriks A menjadi L dan U
    for i in range(n):
        L[i][i] = 1.0
        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]

    # Forward substitution
    t = [0.0] * n
    for i in range(n):
        t[i] = b[i] - sum(L[i][j] * t[j] for j in range(i))

    # Backward substitution
    x = [0.0] * n
    for i in range(n - 1, -1, -1):
        x[i] = (t[i] - sum(U[i][j] * x[j] for j in range(i + 1, n))) /
U[i][i]

    return x

# Contoh penggunaan
A = [[2, -1, 0], [-1, 2, -1], [0, -1, 2]]
b = [1, 0, 1]
solusi = dekomposisi_crout(A, b)
print("Solusi SPL:")
for i, x_i in enumerate(solusi):
    print(f"x_{i+1} =", x_i)
```

Kode ini saya implementasikan menggunakan bahasa Python yang dimana source code tersebut dirancang dan akan di-testing menggunakan aplikasi Visual Studio Code. Untuk alur jalannya kode adalah sebagai berikut:

- Inisialisasi: Matriks A dan vektor b diberikan. n adalah ukuran dari matriks A.

- Inisialisasi Matriks L dan U: Dua matriks segitiga bawah L dan segitiga atas U dengan ukuran $n \times n$ diinisialisasi dengan nilai nol.
- Faktorisasi Matriks A menjadi L dan U:
  - Setiap elemen diagonal Lii diatur ke 1.
  - Setiap elemen diatas atau pada diagonal untuk matriks U dihitung dengan mengurangkan hasil perkalian titik dalam baris terkait matriks L dan kolom terkait matriks U dari elemen yang sesuai dalam matriks A.
  - Setiap elemen di bawah diagonal untuk matriks L dihitung dengan mengurangkan hasil perkalian titik dalam baris terkait matriks L dan kolom terkait matriks U dari elemen yang sesuai dalam matriks A, lalu dibagi dengan elemen diagonal terkait dari matriks U.
- Forward Substitution: Solusi sementara tt diperoleh dengan melakukan substitusi maju menggunakan matriks L dan vektor b.
- Backward Substitution: Solusi akhir x diperoleh dengan melakukan substitusi mundur menggunakan matriks U dan solusi sementara t.

Hasil *running* dari kode *testing*: