Nama = Alia Niswah

NIM = 21120122130063

Metode Matriks Balikan

```
def solve_linear_system(A, B):
    print("Menyelesaikan SPL Ax = B menggunakan metode matriks balikan.")
    print("A: Matriks koefisien (n x n)")
    print("B: Vektor hasil (n x 1)")
    try:
        # Menghitung matriks balikan A
        A_inv = np.linalg.inv(A)

        # Mengalikan matriks balikan A dengan vektor hasil B
        X = np.dot(A_inv, B)
        return X
    except np.linalg.LinAlgError:
        return None  # Matriks A tidak memiliki balikan

A = np.array([[3, 2], [2, 1]])
B = np.array([[6], [5]])

# Menyelesaikan SPL
solution = solve_linear_system(A, B)

if solution is not None:
    x, y = solution.flatten()
    print(f"Solusi SPL: x = {x}, y = {y}")
else:
    print("Matriks koefisien tidak memiliki balikan.")

# Verifikasi solusi dengan mengalikan matriks A dengan solusi yang ditemukan
if np.allclose(np.dot(A, solution), B):
    print("Verifikasi berhasil: A * X = B")
else:
    print("Verifikasi gagal.")
```

Alur Kode Metode Matriks Balikan

1. Fungsi '`solve_linear_system`' didefinisikan untuk menyelesaikan sistem persamaan linear Ax = B.
2. Terdapat blok '`try-except`' untuk menangani ketika matriks A tidak memiliki invers (balikan). Jika matriks A tidak memiliki balikan, function mengembalikan '`none`'
3. Kemudian matriks A akan dihitung menggunakan '`np.linalg.inv(A)`' dan disimpan dalam variabel '`A_inv`'.
4. Hasil perkalian matriks A dengan vektor B dihitung menggunakan '`np.dot(A_inv, B)`' dan disimpan dalam variabel '`X`'.

5. Kemudian nilai variabel ’`x`’ dikembalikan.
6. Inisialisasi matriks A dan vektor b.
7. Fungsi ’`solve_linear_system`’ dipanggil dengan matriks A dan vektor b sebagai argumen.
8. Verifikasi apakah solusi ditemukan. Jika ditemukan, nilai solusi diekstrak ke variabel x dan y.

Metode Dekomposisi Gauss

Alur Kode Metode Dekomposisi LU Gauss

1. Fungsi '`dekomposisi_lu_gauss`'

```
def dekomposisi_lu_gauss(A):
   n = len(A)
   L = np.zeros((n, n))
   U = np.zeros((n, n))

   for i in range(n):
      L[i, i] = 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
```

- Inisialisasi matriks L (segitiga bawah) dan U (segitiga atas) dengan ukuran yang sama dengan matriks A.
- Function ini digunakan untuk melakukan dekomposisi LU dengan metode eliminasi Gauss pada matriks A.
- Input: matriks A (numpy array) – matriks yang didekomposisi
- Output: matriks L dan U – hasil dekomposisi dari matriks A

2. Fungsi '`solve_system`'

```
def solve_system(A, b):
   L, U = dekomposisi_lu_gauss(A)
   y = np.linalg.solve(L, b)
   x = np.linalg.solve(U, y)
   return x
```

- Menyelesaikan sistem persamaan linear.
- Function ini digunakan untuk menyelesaikan sistem persamaan linear Ax = b dengan dekomposisi LU.
- Input: A (numpy array) – matriks, dan b (numpy array) – vektor
- Output: Solusi x dari sistem persamaan linear Ax = b

Dekomposisi Crout

```
import numpy as np

def crout_method(A, b):
    n = len(A)
    L = np.zeros((n, n))
    U = np.zeros((n, n))

    for j in range(n):
        U[j, j] = 1
        for i in range(j, n):
            sum1 = sum(U[k, j] * L[i, k] for k in range(j))
            L[i, j] = A[i, j] - sum1

        for i in range(j, n):
            sum2 = sum(U[k, j] * L[j, i] for k in range(j))
            U[j, i] = (A[j, i] - sum2) / L[j, j]

    y = np.linalg.solve(L, b)
    x = np.linalg.solve(U, y)
    return x, L, U

# Testing
A = np.array([[3, 2], [2, 1]])
b = np.array([6, 5])
x, L, U = crout_method(A, b)

print("Matriks L:")
print(L)
print("\nMatriks U:")
print(U)

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

Alur Kode Metode Dekomposisi Crout

1. Variabel n didefinisikan sebagai dimensi matriks A.
2. Matriks L dan U diinisialisasi sebagai matriks nol dengan ukuran yang sesuai.
3. Iterasi dilakukan pada kolom variabel j (dari 0 hingga n-1).
4. Menyelesaikan sistem persamaan linear dengan function '`np.linalg.solve(L, b)`' untuk mendapatkan vektor y.
5. Menyelesaikan sistem persamaan linear dengan function '`np.linalg.solve(U, y)`' untuk mendapatkan vektor x.
6. Mengembalikan vektor solusi x dan matriks L dan U.