**TUGAS METODE NUMERIK SISTEM PERSAMAAN LINEAR**

Irfan Maulana Manaf

21120122140097

**DEKOMPOSISI CROUT**

Berdasarkan *Jurnal Analisis Kinerja Dekomposisi Crout Sebagai Penyelesaian Sistem Persamaan Linier Berukuran Besar dari Universitas Islam Indonesia, 18 Juni 2005* menjelaskan algoritma untuk penyelesaian masalah dengan metode dekomposisi crout akan seperti :

Do j = 2 to n

a(i,j) = a(i,j) / a(i,1)

Enddo

Do j = 2 to (n – 1)

Do I = j to n

Sum = 0

Do k = 1 to (j – 1)

Sum = Sum + a(i,k) \* a(k,j)

Enddo

a(I,j) = a(i,j) – Sum

Enddo

Do k = (j + 1) to n

Sum = 0

Do i = 1 to (j – 1)

Sum = Sum + a(j,i) \* a(i,k)

Enddo

a(j,k) = (a(j,k) – Sum) / a(j,j)

Enddo

Enddo

Sum = 0

Do k = 1 to (n – 1)

Sum = Sum + (a(n,k) \* a(k,n))

Enddo

a(n,n) = a(n,n) - Sum

langkah-langkahnya secara detail:

1. **Inisialisasi:**

* Algoritma dimulai dengan mengasumsikan kita memiliki matriks koefisien (A) berukuran (n \times n).
* (n) adalah ukuran matriks (jumlah baris dan kolom).

1. **Langkah 1:**

* Iterasi pertama dimulai dengan (j = 2) hingga (n).
* Setiap elemen (a(i,j)) dihitung dengan membagi (a(i,j)) dengan (a(i,1)).

1. **Langkah 2:**
   * Iterasi kedua dimulai dengan (j = 2) hingga (n-1).
   * Iterasi dalam (i) dimulai dari (j) hingga (n).
   * Menghitung (Sum) dengan menjumlahkan produk (a(i,k) \times a(k,j)) untuk (k = 1) hingga (j-1).
   * Menghitung (a(i,j)) dengan mengurangkan (Sum) dari (a(i,j)).
2. **Langkah 3:**
   * Iterasi ketiga dimulai dengan (k = j+1) hingga (n).
   * Iterasi dalam (i) dimulai dari (1) hingga (j-1).
   * Menghitung (Sum) dengan menjumlahkan produk (a(j,i) \times a(i,k)) untuk (i = 1) hingga (j-1).
   * Menghitung (a(j,k)) dengan mengurangkan (Sum) dari (a(j,k)) dan membaginya dengan (a(j,j)).
3. **Langkah 4:**

* Setelah semua iterasi selesai, kita menghitung (Sum) dengan menjumlahkan produk (a(n,k) \times a(k,n)) untuk (k = 1) hingga (n-1).
* Mengurangkan (Sum) dari (a(n,n)) untuk mendapatkan elemen diagonal utama terakhir.

1. **Hasil:**

Setelah algoritma selesai, kita memiliki matriks segitiga bawah (L) dan matriks segitiga atas (U).

Matriks (L) dan (U) dapat digunakan untuk menyelesaikan sistem persamaan linier (Ax = b).

Adapun apabila algoritma tersebut diimplementasikan kedalam bahasa pemrograman dengan menggunakan bahasa python, maka akan menjadi :

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)

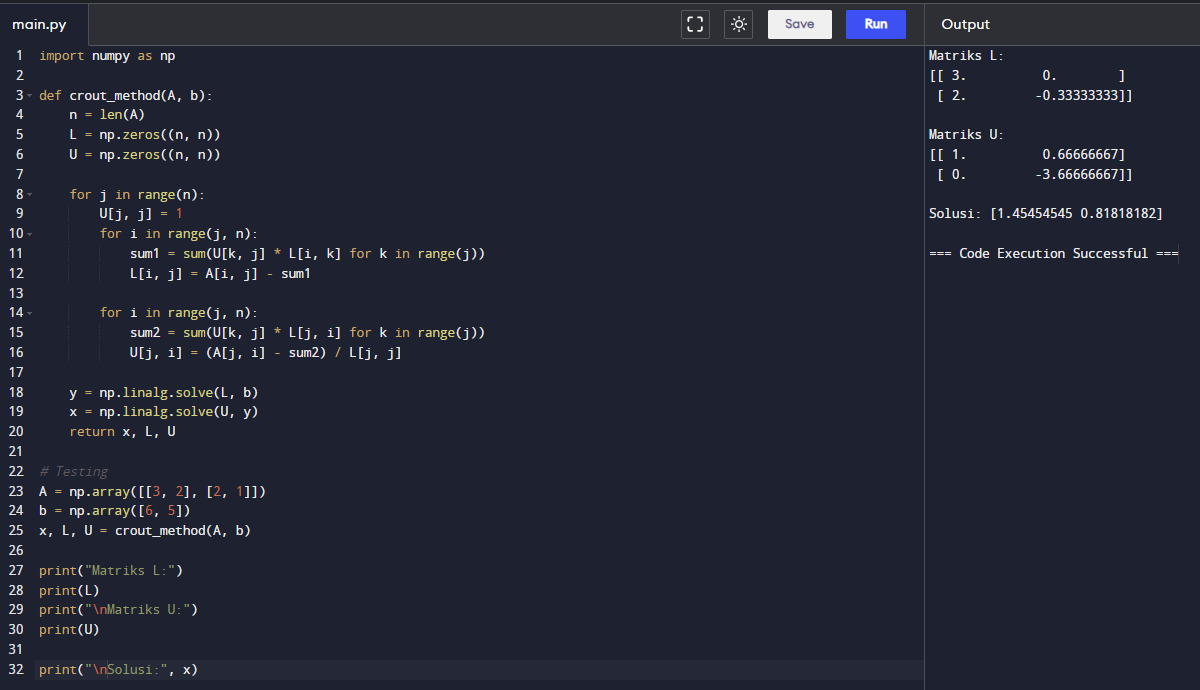
print("\nMatriks U:")

print(U)

print("\nSolusi:", x)

Metode dekomposisi Crout adalah salah satu teknik yang digunakan untuk memecah matriks koefisien **A** dalam sistem persamaan linear **Ax = b** menjadi dua matriks: matriks segitiga bawah **L** dan matriks segitiga atas **U**. Adapun langkah bagaimana kode penyelesaian dengan metode dekomposisi Crout ini berjalan adalah sebagai berikut:

1. **Inisialisasi**: inisialisasi matriks **L** dan **U** dengan nol. Matriks **U** juga diinisialisasi dengan diagonal utama bernilai 1.
2. **Dekomposisi Crout**:
   * Untuk setiap kolom **j** dari matriks **A**, kita hitung elemen-elemen matriks **L** dan **U**:
     + **L[i, j]** dihitung dengan mengurangkan hasil dari produk **U[k, j] \* L[i, k]** untuk semua **k** dari 0 hingga **j** dari elemen **A[i, j]**.
     + **U[j, i]** dihitung dengan mengurangkan hasil dari produk **U[k, j] \* L[j, i]** untuk semua **k** dari 0 hingga **j** dari elemen **A[j, i]**. Kemudian hasilnya dibagi dengan **L[j, j]**.
   * Proses ini menghasilkan faktorisasi **A = LU**.
3. **Penyelesaian Sistem Persamaan Linear**:
   * Setelah mendapatkan matriks **L** dan **U**, kita dapat menyelesaikan sistem persamaan linear   
     **Ax = b** dengan langkah berikut:
     + Pertama, kita selesaikan **Ly = b** untuk mendapatkan vektor **y** menggunakan metode substitusi maju.
     + Kemudian, kita selesaikan **Ux = y** untuk mendapatkan vektor solusi **x** menggunakan metode substitusi mundur.
4. **Hasil**:
   * Hasil dari kode ini adalah vektor solusi **x** dan matriks **L** serta **U** yang digunakan dalam proses dekomposisi.



**MATRIKS BALIKAN**

import numpy as np

def solve\_linear\_system(A, B):

"""

Menyelesaikan SPL Ax = B menggunakan metode matriks balikan.

A: Matriks koefisien (n x n)

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.")

# Kode testing

# Verifikasi solusi dengan mengalikan matriks A dengan solusi yang ditemukan

if np.allclose(np.dot(A, solution), B):

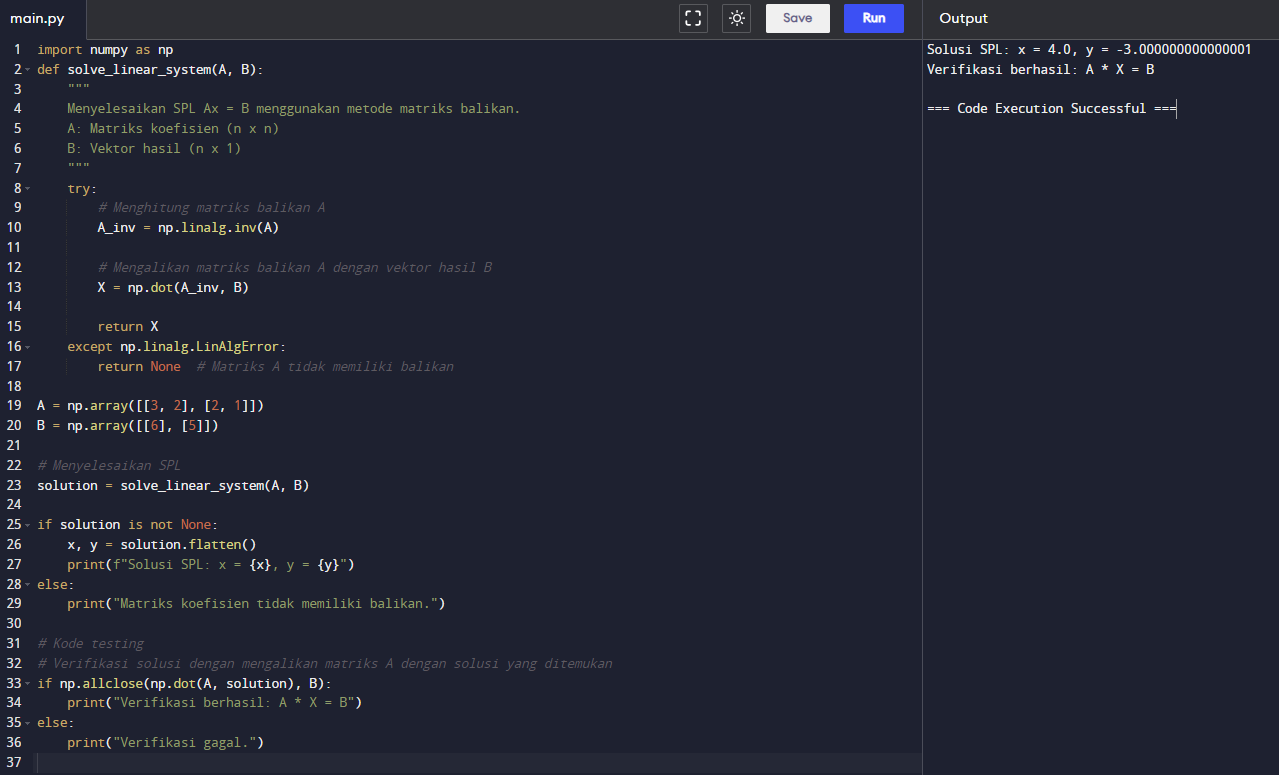
print("Verifikasi berhasil: A \* X = B")

else:

print("Verifikasi gagal.")

Kode yang saya berikan adalah implementasi dalam bahasa Python untuk menyelesaikan sistem persamaan linear (SPL) menggunakan metode matriks balikan. Adapun alur dan langkah langkahnya adalah sebagai berikut :

1. **Fungsi**solve\_linear\_system(A, B):
   * Fungsi ini menerima dua parameter:
     + A: Matriks koefisien (n x n) dari SPL.
     + B: Vektor hasil (n x 1) dari SPL.
   * Langkah-langkah yang dijalankan oleh fungsi ini:
     + Menghitung matriks balikan dari matriks koefisien A menggunakan np.linalg.inv(A).
     + Mengalikan matriks balikan A\_inv dengan vektor hasil B menggunakan np.dot(A\_inv, B).
     + Mengembalikan vektor solusi X.
     + Jika matriks A tidak memiliki balikan (misalnya, determinannya nol), fungsi mengembalikan None.
2. **Inisialisasi Matriks Koefisien dan Vektor Hasil:**
   * + Matriks koefisien A diberikan sebagai [[3, 2], [2, 1]].
     + Vektor hasil B diberikan sebagai [[6], [5]].
3. **Pemanggilan Fungsi** 
   * + Fungsi ini digunakan untuk menyelesaikan SPL dengan matriks koefisien A dan vektor hasil B.
     + Jika solusi ditemukan, nilai x dan y dari vektor solusi dicetak.
     + Jika matriks A tidak memiliki balikan, pesan “Matriks koefisien tidak memiliki balikan.” dicetak.
4. **Verifikasi Solusi:**
   * + Dilakukan verifikasi dengan mengalikan matriks A dengan solusi yang ditemukan (A \* X) dan membandingkannya dengan vektor hasil B.
     + Jika hasil verifikasi berhasil, pesan “Verifikasi berhasil: A \* X = B” dicetak.
     + Jika verifikasi gagal, pesan “Verifikasi gagal.” dicetak.



**DEKOMPOSISI GAUSS LU**

import numpy as np

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

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

# Example usage

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

b = np.array([6, 5])

solution = solve\_system(A, b)

print("Solution:", solution)

**Dekomposisi LU** (Lower-Upper) adalah metode yang digunakan untuk memecahkan sistem persamaan linear dengan menguraikan matriks koefisien menjadi dua matriks: matriks segitiga bawah (L) dan matriks segitiga atas (U). Adapun langkah-langkahnya adalah sebagai berikut:

1. **Dekomposisi LU**:

* Pertama, hitung matriks **L** dan **U** dari matriks koefisien **A**.
* Matriks **L** adalah matriks segitiga bawah dengan elemen diagonal utama bernilai 1. Elemen di atas diagonal utama adalah hasil dari eliminasi Gauss pada matriks **A**.
* Matriks **U** adalah matriks segitiga atas yang juga diperoleh dari eliminasi Gauss pada matriks **A**.
* Proses tersebut membagi matriks **A** menjadi **L** dan **U** sehingga **A = LU**.

1. **Penyelesaian Sistem Persamaan Linear**:

* Setelah memiliki **L** dan **U**, kita dapat menyelesaikan sistem persamaan linear **Ax = b**.
* Pertama, selesaikan **Ly = b** dengan menghitung vektor **y** menggunakan metode *forward substitution*.
* Kemudian, selesaikan **Ux = y** dengan menghitung vektor **x** menggunakan metode *backward substitution*.
* Hasil akhir adalah solusi sistem persamaan linear.

1. **Contoh Penggunaan**:

* Pada contoh di kode, memiliki matriks koefisien **A** dan vektor hasil **b**.
* terlebih dahulu menghitung **L** dan **U** dari **A**.
* Kemudian, selesaikan sistem persamaan linear menggunakan **L** dan **U**.