Jumat, 10 Mei 2024

Nama : Kadek Wisnu Parijata Putra

NIM : 21120122140036

Prodi : Teknik Komputer / 2022

Mata Kuliah : Metode Numerik / D

Github : https://github.com/wisnuprjt/Tugas-Implementasi-Sistem-Persamaan-Linear.git

Diinginkan aplikasi untuk mencari solusi sistem persamaan linear masing-masing menggunakan:

1. Metode Matriks Balikan
2. Metode Dekomposisi LU Gauss
3. Metode Dekomposisi Crout

Contoh kasus yang sama untuk mencari solusi menggunakan ketiga metode diatas:

$2x + 3y - z = 5$

$4x - y + 2z = 3$

$x + 2y - 3z = 1$

Maka :

$$A = \begin{pmatrix} 2 & 3 & -1 \\ 4 & -1 & 2 \\ 1 & 2 & -3 \end{pmatrix},$$

$$B = \begin{pmatrix} 5 \\ 3 \\ 1 \end{pmatrix}$$

I. Metode Matriks Balikan

```
import numpy as np

Matriks koefisien
A = np.array([[2, 3, -1],
              [4, -1, 2],
              [1, 2, -3]])

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

A_inv = np.linalg.inv(A)

X = np.dot(A_inv, B)

print("Solusi x, y, z:")
print(X)
```

Hasil :

```
Solusi x, y, z:
[0.67741935 1.51612903 0.90322581]
```

Analisa :

Metode Matriks Balikan invers dari matriks koefisien (matriks A) dan kemudian dikalikan dengan vektor konstanta (Matriks B) untuk mendapatkan vektor solutions. Kode numpy `np.linalg.inv(A)` adalah instruksi untuk mengembalikan nilai dari (matriks A). untuk mendapatkan solusi dengan Metode Invers, hasil balikan dari (matriks A) dikalikan dengan (matriks B). Perkalian ini menggunakan kode `np.dot(A_inv,b)`.

II. Metode Dekomposisi LU Gauss

```
import numpy as np

def lu_decomposition(A):

        n = len(A)
        L = np.eye(n)
        U = A.astype(float)

      for k in range(n-1):
          for i in range(k+1, n):
          factor = U[i, k] / U[k, k]
          L[i, k] = factor
          U[i, k:] -= factor * U[k, k:]
          return L, U

      def forward_substitution(L, B):

          n = len(L)
          Y = np.zeros(n)
          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 = len(U)
          X = np.zeros(n)
          for i in range(n-1, -1, -1):
              X[i] = (Y[i] - np.dot(U[i, i+1:], X[i+1:])) / U[i, i]
          return X

      def solve_linear_system(A, B):

          L, U = lu_decomposition(A)
          Y = forward_substitution(L, B)
          X = backward_substitution(U, Y)
          return X

      A = np.array([[2, 3, -1],
                    [4, -1, 2],
                    [1, 2, -3]])

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

      X = solve_linear_system(A, B)
      print("Solusi x, y, z:")
print(X)
```

Hasil :

```
Solusi x, y, z:
[0.67741935 1.51612903 0.90322581]
```

Analisa :

Metode dimulai dengan dekomposisi matriks koefisien A menjadi dua matriks segitiga, yaitu matriks segitiga bawah L dan matriks segitiga atas U. Tahap ini dilakukan melalui algoritma eliminasi Gauss. Proses ini terjadi di dalam fungsi lu_decomposition. Selama iterasi, faktor-faktor yang diperlukan untuk menghilangkan elemen di bawah diagonal utama dari U dan mengisi nilai-nilai L dihitung dan disimpan. Setelah mendapatkan matriks L dan U, langkah selanjutnya adalah menyelesaikan dua persamaan: $LY = B$ dan $U X = Y$, di mana Y adalah vektor hasil substitusi maju. Proses ini terjadi di dalam fungsi `forward_substitution` dan `backward_substitution`. Substitusi maju menyelesaikan persamaan $LY = B$ untuk mencari Y, sementara substitusi mundur menyelesaikan persamaan $U X = Y$ untuk mencari solusi X dari sistem persamaan linier. Kode kemudian menggunakan matriks koefisien A dan matriks hasil B yang telah diberikan untuk menemukan solusi X dari sistem persamaan linier tersebut dengan memanggil fungsi solve_linear_system.

III. Metode Dekomposisi Crout

```
import numpy as np

def crout_decomposition(A):

          n = len(A)
          L = np.zeros((n, n))
          U = np.zeros((n, n))

          for j in range(n):
              U[j, j] = 1  # Diagonal utama U adalah 1
              for i in range(j, n):
                  L[i, j] = A[i, j] - np.dot(L[i, :j], U[:j, j])
# Menghitung elemen L
          for i in range(j+1, n):
                  U[j+1:, j] = (A[j+1:, j] - np.dot(L[j+1:, :j],
      U[:j, j])) / L[j, j]  # Menghitung elemen U
          return L, U

def forward_substitution(L, B):

          n = len(L)
          Y = np.zeros(n)
          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 = len(U)
          X = np.zeros(n)
```

```
          for i in range(n-1, -1, -1):
          X[i] = (Y[i] - np.dot(U[i, i+1:], X[i+1:])) / U[i, i]
          return X

def solve_linear_system(A, B):

          L, U = crout_decomposition(A)
          Y = forward_substitution(L, B)
          X = backward_substitution(U, Y)
          return X

# Matriks koefisien
      A = np.array([[2, 3, -1],
                    [4, -1, 2],
                    [1, 2, -3]])

# Matriks hasil
      B = np.array([5, 3, 1])

# Menyelesaikan sistem persamaan linier
      X = solve_linear_system(A, B)
      print("Solusi x, y, z:")
      print(X)
```

Hasil :

```
Solusi x, y, z:
[2.5        7.         5.16666667]
```

Analisa :

Metode Dekomposisi Crout di atas berfungsi untuk menyelesaikan sistem persamaan linier dimulai dengan, fungsi `crout_decomposition` didefinisikan untuk memecah matriks koefisien A menjadi dua matriks segitiga, yakni ‘L’ dan ‘U’. Iterasi pertama pada fungsi ini menetapkan matriks identitas untuk U, kemudian melakukan iterasi melalui setiap elemen matriks L dan U untuk menghitung nilainya. Dalam iterasi kedua, elemen-elemen di bawah atau di atas diagonal utama dihitung menggunakan rumus dekomposisi Crout. Setelah dekomposisi, dua fungsi lainnya, yaitu `forward_substitution` dan `backward_substitution`, digunakan untuk menghitung vektor solusi. forward_substitution menghitung nilai vektor ‘Y’ dengan menggunakan matriks segitiga bawah ‘L’, sedangkan `backward_substitution` menghitung solusi akhir ‘X’ menggunakan matriks segitiga atas ‘U’. Jika semua perhitungan selesai, solusi sistem persamaan linier dicetak dalam bentuk vektor X.