DOKUMENTASI FUNGSIONAL PROGRAM

Kelas : B

Anggota Kelompok:

- Fathin Achmad Ashari (L0122062)
- Ferdy Rizkiawan (L0122064)
- Ikhsan Ari Novianto (L0122077)

Pengantar

Program ini adalah Aplikasi Aljabar Linear dalam Python yang memungkinkan untuk melakukan berbagai operasi pada matriks. Program ini mencakup fungsionalitas seperti menyelesaikan sistem persamaan linear, mencari karakteristik polinomial, nilai eigen, vektor eigen, dan diagonalisasi matriks menggunakan metode seperti Metode Normal, Metode Kuadrat Terkecil, Metode Eliminasi Gauss, SVD (Singular Value Decomposition), dan SVD Complex.

Kelas: Matrix

Kelas ini merepresentasikan sebuah matriks dan berisi metode-metode untuk melakukan berbagai operasi pada matriks.

1. **Konstruktor: __init__(self, n, m, A, b)**
   - Parameter:
     - n: Bilangan bulat yang mewakili jumlah baris dalam matriks.
     - m: Bilangan bulat yang mewakili jumlah kolom dalam matriks.
     - A: Matriks numpy 2D yang merepresentasikan koefisien matriks.
     - b: Vektor numpy 1D yang merepresentasikan vektor konstanta dalam sistem persamaan linear.
   - Menginisialisasi objek matriks dengan parameter yang diberikan.

2. **Metode: svd(self)**
   - Menyelesaikan matriks menggunakan SVD (Singular Value Decomposition).

- Menghitung dekomposisi nilai singular (Singular Value Decomposition) dari matriks A menggunakan np.linalg.svd.
- Mencetak matriks U, SVD, dan V.
- Menyimpan hasil dalam file no4.txt.

3. **Metode: svd_complex(self)**
   - Menyelesaikan persamaan menggunakan SVD Complex.
   - Menghitung vektor solusi x menggunakan pendekatan SVD Complex.
   - Mencetak solusi-solusi x.
   - Menyimpan hasil dalam file no5.txt.

4. **Metode: poly_eigen_diag(self)**
   - Mencari karakteristik polinomial, nilai eigen, vektor eigen, dan matriks diagonal.
   - Menghitung karakteristik polinomial menggunakan np.poly.
   - Menghitung nilai eigen dan vektor eigen menggunakan np.linalg.eig.
   - Menghitung diagonalisasi matriks menggunakan dekomposisi eigen.
   - Mencetak karakteristik polinomial, nilai eigen, vektor eigen, matriks P, matriks D, dan matriks diagonal.
   - Menyimpan hasil dalam file no3.txt.

5. **Metode: spl(self, sk)**
   - Menyelesaikan sistem persamaan linear menggunakan metode yang berbeda berdasarkan pilihan pengguna.
   - Menampilkan pilihan metode (Metode Normal, Metode Kuadrat Terkecil, atau Metode Eliminasi Gauss).
   - Berdasarkan pilihan pengguna, melakukan metode yang sesuai dan mencetak solusi-solusi.
   - Menyimpan hasil dalam file no{}.txt, di mana {} diganti dengan nilai sk yang diberikan.

Fungsi: find_characteristic_poly(matrix)

- Parameter:

    - matrix: Matriks numpy 2D yang merepresentasikan sebuah matriks.
- Menghitung karakteristik polinomial matriks menggunakan np.poly.

Fungsi: find_eigen(matrix)

- Parameter:
    - matrix: Matriks numpy 2D yang merepresentasikan sebuah matriks.
- Menghitung nilai eigen dan vektor eigen matriks menggunakan np.linalg.eig.
- Mengembalikan nilai eigen dan vektor eigen.

Fungsi: find_diagonalize(matrix)

- Parameter:
    - matrix: Matriks numpy 2D yang merepresentasikan sebuah matriks.
- Menghitung matriks diagonal dan matriks diagonalisasi menggunakan dekomposisi eigen.
- Mengembalikan matriks P, matriks D, dan matriks diagonalisasi.

Fungsi: gaussian_elimination(matrix, vector)

- Parameter:
    - matrix: Matriks numpy 2D yang merepresentasikan koefisien matriks dalam sistem persamaan linear.
    - vector: Vektor numpy 1D yang merepresentasikan vektor konstanta dalam sistem persamaan linear.
- Menyelesaikan sistem persamaan linear menggunakan metode eliminasi Gauss.
- Mengembalikan vektor solusi jika solusi tunggal ada.
- Mengembalikan "Infinite solutions!" jika sistem memiliki solusi tak terhingga.
- Mengembalikan "No solution!" jika sistem tidak memiliki solusi.

Fungsi: limit(matrix)

- Parameter:

    - matrix: Matriks numpy yang akan dibulatkan.
- Membulatkan elemen-elemen matriks menjadi 5 angka desimal.
- Mengembalikan matriks yang telah dibulatkan.

Fungsi: matrix_to_txt(title, matrix, file_name)

- Parameter:
    - title: String yang mewakili judul matriks.
    - matrix: Matriks numpy yang akan disimpan.
    - file_name: String yang mewakili nama file untuk menyimpan matriks.
- Menyimpan matriks dan judulnya ke dalam file yang ditentukan.
- Jika nama file berakhiran '1', '2', atau '5', matriks akan disimpan dengan nama variabel x1, x2, dst.
- Jika tidak, matriks akan disimpan sebagai matriks 2D.

Main Program

- Menanyakan kepada pengguna untuk memasukkan nomor studi kasus, jumlah baris, dan jumlah kolom.
- Menginisialisasi matriks A dan vektor b berdasarkan dimensi yang diberikan.
- Menanyakan kepada pengguna untuk memasukkan koefisien matriks A.
- Menanyakan kepada pengguna untuk memasukkan vektor konstanta b.
- Memanggil fungsi matrix_to_txt untuk menyimpan matriks A dan vektor b ke dalam file no{}.txt, di mana {} diganti dengan nomor studi kasus.
- Membuat objek Matrix dengan parameter yang diberikan.
- Berdasarkan nomor studi kasus, memanggil metode yang sesuai (spl, poly_eigen_diag, svd, atau svd_complex) dari objek Matrix.
- Menampilkan hasil dan menyimpannya ke file-file yang sesuai.

Catatan: Program ini mengasumsikan bahwa pengguna memberikan masukan yang valid dan menangani pengecualian untuk kasus-kasus tertentu (misalnya, matriks singular atau tidak ada solusi dalam sistem persamaan linear).

Panduan Pengguna

1. Contoh 1

a. Input case study

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study :
```

b. Input row dan column

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 1

Enter the number of rows :
```

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 1

Enter the number of rows : 4
Enter the number of columns :
```

c. Input matrix A

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 1

Enter the number of rows : 4
Enter the number of columns : 4

Enter the matrix A (or coefficients) :
1 1 -1 -1
2 5 -7 -5
2 -1 1 3
5 2 -4 2
```

d. Input vector b

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 1

Enter the number of rows : 4
Enter the number of columns : 4

Enter the matrix A (or coefficients) :
1 1 -1 -1
2 5 -7 -5
2 -1 1 3
5 2 -4 2

Enter the matrix b (constants) :
1 -2 4 6
```

e. Input menu

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 1

Enter the number of rows : 4
Enter the number of columns : 4

Enter the matrix A (or coefficients) :
1 1 -1 -1
2 5 -7 -5
2 -1 1 3
5 2 -4 2

Enter the matrix b (constants) :
1 -2 4 6

Menu :
a. Normal Method
b. Least-squares Method
c. Gaussian Elimination Method
Pilih :
```

f. Output

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 1

Enter the number of rows : 4
Enter the number of columns : 4

Enter the matrix A (or coefficients) :
1 1 -1 -1
2 5 -7 -5
2 -1 1 3
5 2 -4 2

Enter the matrix b (constants) :
1 -2 4 6

Menu :
a. Normal Method
b. Least-squares Method
c. Gaussian Elimination Method
Pilih : a

Normal Method :
No solution!

The results have been saved in no1.txt !
```

2. Contoh 2

a. Input case study

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 5
```

b. Input row dan column

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 5

Enter the number of rows : 6
Enter the number of columns : 5
```

c. Input matrix A

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 5

Enter the number of rows : 6
Enter the number of columns : 5

Enter the matrix A (or coefficients) :
0+1j 0+0j -1+0j 0+0j 1+0j
1+0j 1+0j 0-1j 0+0j 0+0j
2+0j 0+0j 0+5j 0+0j 0+0j
0+0j 0+2j 2+0j 0+0j 0+0j
0+0j 0+0j 0+0j 1+0j 1+0j
0+1j 0+1j 0+0j 0+0j 0+0j
```

d. Input vector b

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 5

Enter the number of rows : 6
Enter the number of columns : 5

Enter the matrix A (or coefficients) :
0+1j 0+0j -1+0j 0+0j 1+0j
1+0j 1+0j 0-1j 0+0j 0+0j
2+0j 0+0j 0+5j 0+0j 0+0j
0+0j 0+2j 2+0j 0+0j 0+0j
0+0j 0+0j 0+0j 1+0j 1+0j
0+1j 0+1j 0+0j 0+0j 0+0j

Enter the matrix b (constants) :
3+0j 0+4j 0-3j -5+0j 5+0j -3+0j
```

e. Output

```
PS E:\Code> python -u "e:\Code\pypypy\linear_algebra\final_project\src\main.py"
Linear Algebra Application in Python
Case study : 5

Enter the number of rows : 6
Enter the number of columns : 5

Enter the matrix A (or coefficients) :
0+1j 0+0j -1+0j 0+0j 1+0j
1+0j 1+0j 0-1j 0+0j 0+0j
2+0j 0+0j 0+5j 0+0j 0+0j
0+0j 0+2j 2+0j 0+0j 0+0j
0+0j 0+0j 0+0j 1+0j 1+0j
0+1j 0+1j 0+0j 0+0j 0+0j

Enter the matrix b (constants) :
3+0j 0+4j 0-3j -5+0j 5+0j -3+0j

SVD Complex :
x1 = (-1.7285262260207235e-16+5.180555555555554j)
x2 = (7.496584076010932e-16-5.541666666666666j)
x3 = (2.7777777777777777+1.1441323316087654e-17j)
x4 = (4.402777777777778+9.661796276856013e-16j)
x5 = (0.5972222222222205-1.828130584126575e-15j)

The results have been saved in no5.txt !
```