

LAPORAN
APLIKASI ALJABAR LINEAR PADA PERSAMAAN LINEAR
BILANGAN KOMPLEKS



Disusun oleh :

Fathin Achmad Ashari

L0122062

Ferdy Rizkiawan

L0122064

Ikhsan Ari Novianto

L0122077

MATA KULIAH ALJABAR LINEAR
PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA
UNIVERSITAS SEBELAS MARET
2023

KATA PENGANTAR

Puji dan syukur kami panjatkan kepada Tuhan Yang Maha Esa atas segala petunjuk, rahmat, dan karunia-Nya sehingga kami dapat menyelesaikan laporan ini tepat pada waktunya. Adapun judul dari laporan kami adalah “Aplikasi Aljabar Linear pada Persamaan Linear Bilangan Kompleks”.

Dalam laporan ini, kami akan memperkenalkan sebuah program yang dikembangkan menggunakan bahasa pemrograman Python. Program ini dirancang untuk menghitung solusi dari studi kasus persamaan linear bilangan kompleks yang telah diberikan. Kami menggunakan library NumPy dalam Python untuk melakukan operasi aljabar linear secara efisien.

Kami menyadari bahwa banyak pihak yang memberikan dukungan dan bantuan selama menyelesaikan makalah ini. Oleh karena itu, kami mengucapkan terima kasih dan mendoakan semoga Tuhan Yang Maha Esa memberikan balasan terbaik kepada segenap pihak, terutama kepada Prof. Drs. Bambang Harjito, M.App.Sc., Ph.D selaku dosen pengampu dan telah membimbing proses pelaksanaan proyek hingga penyelesaian makalah ini dengan sangat baik.

Akhir kata, kami menyadari bahwa makalah ini masih jauh dari kata sempurna. Oleh karena itu, kami memohon maaf atas ketidaktepatan kami dalam pembuatan laporan ini. Kami berharap semoga laporan ini dapat bermanfaat bagi kami dan juga pembaca sehingga dapat menjadi referensi demi pengembangan ke arah yang lebih baik.

Surakarta, 25 Juni 2023

Penyusun

DAFTAR ISI

KATA PENGANTAR	2
DAFTAR ISI.....	3
BAB I DESKRIPSI MASALAH	4
A. DESKRIPSI UMUM TUGAS PROJECT PEMBUATAN APLIKASI.....	4
B. BAHASA PEMROGRAMAN.....	4
C. PENILAIAN	4
D. SPESIFIKASI UMUM.....	5
E. STUDI KASUS.....	5
BAB II TEORI.....	7
A. SISTEM PERSAMAAN LINEAR	7
B. MATRIKS ESELON TEREDUKSI	10
C. POLYNOMIAL KARAKTERISTIK	11
D. NILAI EIGEN.....	11
E. VEKTOR EIGEN	12
F. DIAGONALISASI.....	12
G. SINGULAR VALUE DECOMPOSITION	13
BAB III IMPLEMENTASI PROGRAM.....	15
A. MAIN PROGRAM	15
B. CLASS “MATRIX”	18
C. FUNCTIONS	23
BAB IV EKSPERIMEN	27
BAB V KESIMPULAN.....	36
A. KESIMPULAN.....	36
B. SARAN PENGEMBANGAN.....	36
C. REFLEKSI.....	37
DAFTAR PUSTAKA	38

BAB I

DESKRIPSI MASALAH

A. DESKRIPSI UMUM TUGAS PROJECT PEMBUATAN APLIKASI

Tugas Project ini adalah membuat program penyelesaian Sistem Persamaan Linier (SPL), membuktikan matriks terdiagonalisasi, serta mencari Singular Value Decomposition (SVD) dalam bahasa pemrograman Python dengan menggunakan metode eliminasi Gauss dan/atau Gauss-Jordan. SPL dapat memiliki penyelesaian tunggal, banyak penyelesaian, atau penyelesaian tidak ada. Mencari nilai Eigen value dan Eigen Vektor untuk membuktikan diagonalisasi matrik serta menyelesaikan system persamaan linear kompleks dengan SVD.

Untuk menyelesaikan Sistem Persamaan Linear (SPL) dengan n peubah (*variable*) dan m persamaan: SPL dapat diselesaikan secara numerik dengan metode eliminasi Gauss dan metode eliminasi Gauss-Jordan. Program harus dapat menangani kasus-kasus sebagai berikut:

1. SPL memiliki solusi unik, tampilkan solusinya
2. SPL memiliki solusi tak terbatas, tampilkan solusinya dalam bentuk parameter
3. SPL tidak memiliki solusi, tuliskan tidak ada solusinya

B. BAHASA PEMROGRAMAN

1. Bahasa program yang digunakan adalah Python dengan menggunakan fungsi-fungsi yang sudah ada maupun dapat membuat fungsi sendiri.
2. Program tidak harus berbasis GUI, cukup text-based saja.

C. PENILAIAN

Komposisi penilaian umum adalah sebagai berikut :

1. Program: 80 %
2. Laporan : 20 %

D. SPESIFIKASI UMUM

1. Program harus dapat menerima input data dari
 - Papan ketik
2. Keluaran program harus dapat ditampilkan ke:
 - Layar monitor
 - Simpan ke dalam arsip

Format luaran (misalnya dalam bentuk tabel) didefinisikan sendiri. Luaran harus mudah dibaca dan informatif.

E. STUDI KASUS

1. Carilah Penyelesaian dari sistem Persamaan Linear berikut ini :

a.

$$A = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 5 & -7 & -5 \\ 2 & -1 & 1 & 3 \\ 5 & 2 & -4 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -2 \\ 4 \\ 6 \end{bmatrix}$$

b.

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & -3 & 0 \\ 2 & -1 & 0 & 1 & -1 \\ -1 & 2 & 0 & -2 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 6 \\ 5 \\ -1 \end{bmatrix}$$

2. Sistem Persamaan berbentuk

a.

$$\begin{aligned} 3x_1 + 8x_2 - 3x_3 - 14x_4 &= 2 \\ 2x_1 + 3x_2 - x_3 - 2x_4 &= 1 \\ x_1 - 2x_2 + x_3 + 10x_4 &= 0 \\ x_1 + 5x_2 - 2x_3 - 12x_4 &= 1 \end{aligned}$$

b.

$$\begin{aligned} x_1 - x_2 + x_3 - x_4 &= 0 \\ -x_1 + x_2 + x_3 + x_4 &= 0 \\ x_1 + x_2 - x_3 + x_4 &= 0 \\ x_1 + x_2 + x_3 + x_4 &= 0 \end{aligned}$$

3. Carilah polynomial characteristic, eigenvalues, eigenvector dan (jika mungkin) carilah matrik P yang mempunyai invers sehingga $P^{-1}AP$ adalah diagonal

$$\text{g. } A = \begin{bmatrix} 3 & 1 & 1 \\ -4 & -2 & -5 \\ 2 & 2 & 5 \end{bmatrix} \quad \text{h. } A = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & -1 & 2 \end{bmatrix}$$

4. Carilah a Sebuah SVD dari matrik berikut ini

$$\text{a. } A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{b. } \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & -2 \\ 1 & 2 & 0 \end{bmatrix}$$

5. Selesaikan sistem persamaan linear kompleks dengan menggunakan SVD. Diberikan sistem persamaan linear kompleks dengan 6 persamaan dan 5 variabel sebagai berikut :

$$\begin{aligned} ix_1 - x_3 + x_5 &= 3 \\ x_1 + x_2 - ix_3 &= 4i \\ 2x_1 + 5ix_3 &= -3i \\ 2ix_2 + 2x_3 &= -5 \\ x_4 + x_5 &= 5 \\ ix_1 + ix_2 &= -3 \end{aligned}$$

BAB II

TEORI

A. SISTEM PERSAMAAN LINEAR

1. Pengertian Sistem Persamaan Linear

Sistem Persamaan Linear atau SPL merupakan sebuah sistem yang terdiri dari sekumpulan persamaan linear yang berisikan variabel-variabel. Nilai variabel tersebut harus ditentukan sehingga semua persamaan tersebut dapat dipenuhi secara bersamaan. Tujuan dari sistem persamaan linear adalah untuk menemukan solusi yang memenuhi semua persamaan dalam sistem tersebut.

Sistem persamaan linear secara umum dapat dituliskan dalam bentuk matriks sebagai berikut :

$$AX = B.$$

A merupakan matriks koefisien yang berisi koefisien-koefisien dari persamaan linear, X merupakan vektor variabel yang memuat variabel-variabel yang perlu dihitung, dan B adalah vektor konstanta yang berisi konstanta-konstanta di sisi kanan persamaan linear.

2. Metode Penyelesaian Sistem Persamaan Linear

Dalam sebuah Sistem Persamaan Linear (SPL) terdapat beberapa metode yang dapat digunakan untuk menyelesaikan Sistem Persamaan Linear (SPL) diantaranya, Eliminasi Gauss, Eliminasi Gauss-Jordan, Aturan Cramer, dan invers matriks koefisien, dimana solusi dari metode ini adalah :

$$X = A^{-1}B.$$

Metode aturan Cramer dan invers matriks tidak dapat digunakan apabila matriks A bukan merupakan matriks persegi atau . Keempat metode tersebut juga memiliki kelemahan lain yaitu mempunyai pemecahan yang tidak konsisten. Akan tetapi pada laporan ini yang perlu kita sorot lebih lanjut adalah metode Eliminasi Gauss dan Eliminasi Gauss-Jordan. Berikut adalah penjelasan kedua metode tersebut :

a. Eliminasi Gauss

Eliminasi Gauss adalah teknik yang digunakan untuk memanipulasi nilai-nilai dalam matriks dengan tujuan menghasilkan matriks yang lebih sederhana. Metode ini melibatkan operasi baris pada matriks sehingga matriks tersebut

berada dalam bentuk baris. Eliminasi Gauss merupakan salah satu metode penyelesaian persamaan linear dengan menggunakan matriks. Prosesnya melibatkan pengubahan persamaan linear menjadi matriks teraugmentasi dan melakukan operasi pada matriks tersebut. Setelah matriks berada dalam bentuk baris, langkah selanjutnya adalah melakukan substitusi balik untuk mendapatkan nilai variabel dalam persamaan tersebut.

Metode eliminasi Gauss memiliki ciri-ciri berikut:

- 1) Tahap Eliminasi: Metode eliminasi Gauss melibatkan tahap eliminasi, di mana operasi baris dilakukan pada matriks untuk mengurangi atau menghilangkan koefisien variabel dalam persamaan. Tujuan tahap ini adalah mengubah matriks menjadi bentuk segitiga atas.
- 2) Matriks Segitiga Atas: Hasil dari tahap eliminasi adalah matriks segitiga atas. Dalam matriks segitiga atas, semua elemen di bawah diagonal utama adalah nol. Hal ini memudahkan langkah selanjutnya dalam menentukan solusi sistem persamaan linear.
- 3) Penggunaan Faktor Pengali: Metode eliminasi Gauss menggunakan faktor pengali untuk menghilangkan koefisien variabel. Faktor pengali diperoleh dengan membagi koefisien variabel pada baris di bawahnya dengan koefisien variabel pada baris yang sedang diproses.
- 4) Substitusi Mundur: Setelah tahap eliminasi selesai dan matriks segitiga atas terbentuk, langkah selanjutnya adalah melakukan substitusi mundur. Substitusi ini dilakukan dengan memulai dari persamaan terakhir dan menghitung nilai variabel secara berurutan dengan membagi konstanta pada sisi kanan oleh koefisien variabel yang bersangkutan.
- 5) Penyelesaian Sistem Persamaan Linear: Metode eliminasi Gauss memberikan solusi sistem persamaan linear dengan menghasilkan nilai variabel yang memenuhi semua persamaan dalam sistem tersebut. Solusi dapat berupa solusi unik, solusi tak terhingga, atau mungkin tidak ada solusi, tergantung pada karakteristik matriks koefisien.

Ciri-ciri ini merupakan elemen utama dalam metode eliminasi Gauss yang membedakannya dari metode penyelesaian SPL lainnya. Metode ini efektif dalam menyelesaikan sistem persamaan linear dengan menggunakan langkah-langkah eliminasi dan substitusi mundur.

b. Eliminasi Gauss-Jordan :

Eliminasi Gauss-Jordan merupakan perkembangan dari metode eliminasi Gauss yang menghasilkan bentuk matriks yang lebih sederhana. Metode ini melibatkan kelanjutan dari operasi baris yang dilakukan dalam eliminasi Gauss, sehingga menghasilkan matriks dalam bentuk Eselon-baris. Metode eliminasi Gauss-Jordan juga dapat digunakan sebagai salah satu metode penyelesaian persamaan linear dengan menggunakan matriks. Berikut adalah karakteristik metode ini :

Ada beberapa karakteristik metode eliminasi Gauss-Jordan:

- 1) Tahap Eliminasi Lanjutan: Metode eliminasi Gauss-Jordan memasukkan tahap eliminasi tambahan. Setelah matriks mencapai bentuk eselon-baris tereduksi, operasi baris elemen terus dilakukan padanya. Tujuan dari tahap ini adalah untuk menyederhanakan matriks menjadi bentuk yang paling sederhana mungkin.
- 2) Matriks Eselon-Baris Tereduksi: Matriks eselon-baris tereduksi adalah hasil dari tahap eliminasi metode ini. Matriks ini memiliki satu baris dengan elemen bukan nol yang memiliki satu leading 1 (1 utama) di antara semua elemen bukan nol. Selain itu, setiap leading 1 ini adalah satu-satunya elemen bukan nol di kolomnya.
- 3) Penggunaan Faktor Skala: Seperti metode eliminasi Gauss-Jordan, metode ini juga menggunakan faktor skala. Koefisien variabel dihilangkan dengan faktor skala ini, yang membawa matriks ke bentuk tereduksi.
- 4) Solusi yang Lebih Lengkap: Metode eliminasi Gauss-Jordan tidak hanya memberikan solusi yang lebih lengkap dibandingkan dengan eliminasi Gauss, tetapi juga dapat menghasilkan solusi tak terhingga jika ada variabel bebas dan menemukan ketidaksesuaian (*inconsistency*) dalam kasus di mana tidak ada solusi yang memenuhi persamaan linear.
- 5) Langkah Substitusi: Substitusi balik dilakukan dengan memulai dari persamaan terakhir dan menghitung nilai variabel secara berurutan dengan membagi konstanta pada sisi kanan oleh koefisien variabel yang bersangkutan. Langkah ini dilakukan setelah tahap eliminasi selesai dan matriks eselon-baris tereduksi dibentuk.
- 6) Ciri-ciri ini membedakan metode eliminasi Gauss-Jordan dari metode eliminasi Gauss biasa. Metode ini menghasilkan matriks Eselon-baris

tereduksi, yang memberikan informasi lebih rinci tentang solusi sistem persamaan linear.

B. MATRIKS ESELON TEREDUKSI

Matriks eselon tereduksi, juga dikenal sebagai matriks baris tereduksi atau matriks echelon tereduksi, adalah bentuk istimewa dari matriks eselon yang memiliki sifat-sifat khusus untuk analisis dan aplikasi matematika.

$$\begin{bmatrix} 1 & * & * \\ 0 & 1 & * \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & * & * & * \\ 0 & 0 & 1 & * \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & * & * & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Keterangan: * adalah sembarang nilai

Untuk membentuk matriks eselon tereduksi, langkah-langkah berikut diikuti:

1. Baris-baris yang terdiri dari nol ditempatkan di bagian bawah matriks.
2. Pada setiap kolom yang berisi elemen nol di luar baris terakhir, elemen pertama yang bukan nol (elemen terkiri atas) diubah menjadi 1 (pemimpin).
3. Setiap elemen di bawah dan di atas pemimpin dalam kolom yang sama diubah menjadi 0.
4. Pada setiap baris yang berisi pemimpin, semua elemen lain dalam kolom yang sama juga diubah menjadi 0.

Dengan menerapkan langkah-langkah ini, matriks awal dapat diubah menjadi matriks eselon tereduksi.

Keuntungan utama dari matriks eselon tereduksi adalah memberikan informasi yang lebih jelas dan kompak tentang hubungan linier antara baris dan kolom matriks. Dalam bentuk ini, variabel utama (*leading variables*) dan variabel bebas (*free variables*) dalam sistem persamaan linear yang mewakili matriks dapat diidentifikasi dengan mudah. Selain itu, matriks eselon tereduksi mempermudah perhitungan dan analisis lebih lanjut, seperti perhitungan ruang kolom dan ruang nol matriks.

Dalam praktiknya, matriks eselon tereduksi sering digunakan dalam penyelesaian sistem persamaan linear, inversi matriks, dan berbagai aplikasi matematika lainnya di mana manipulasi matriks diperlukan.

C. POLYNOMIAL KARAKTERISTIK

Polynomial karakteristik adalah polynomial dengan koefisien di dalam lapangan yang memiliki sifat-sifat khusus terkait matriks. Dalam teori matriks, polynomial ini memberikan informasi penting tentang matriks tersebut.

Polynomial karakteristik didefinisikan sebagai polynomial monik dengan koefisien di dalam lapangan yang memenuhi persamaan $p(A) = \det(A - \lambda I) = 0$. Dalam persamaan ini, A adalah matriks persegi, $\det(A - \lambda I)$ adalah determinan dari A dikurangi dengan λ dikali dengan matriks identitas I , dan λ adalah variabel polynomial.

Polynomial karakteristik memiliki beberapa sifat dan memberikan informasi yang berharga tentang matriks, seperti nilai eigen (eigenvalues), multiplicitas eigen, diagonalisasi matriks, dan sifat-sifat lainnya. Akar-akar dari polynomial karakteristik merupakan nilai eigen matriks, dan multiplicitas eigen memberikan informasi tentang jumlah eigenvektor yang terkait dengan nilai eigen tersebut. Selain itu, matriks persegi dapat diagonalisasi jika semua nilai eigen memiliki multiplicitas 1 dalam polynomial karakteristik. Polynomial karakteristik juga membantu dalam pemecahan masalah eigen, diagonalisasi matriks, dan analisis struktur matriks.

Dengan memahami polynomial karakteristik matriks, kita dapat mendapatkan pemahaman yang lebih mendalam tentang sifat-sifat matematika yang terkait dengan matriks tersebut.

D. NILAI EIGEN

Nilai eigen (*eigenvalues*) adalah nilai-nilai khusus terkait dengan matriks. Mereka adalah solusi dari persamaan eigen untuk matriks persegi.

Persamaan eigen adalah $Av = \lambda v$, di mana A adalah matriks persegi, v adalah vektor eigen, dan λ adalah nilai eigen yang bisa berupa bilangan riil atau kompleks.

Nilai eigen dapat ditemukan dengan mencari akar-akar dari polynomial karakteristik, yaitu $\det(A - \lambda I) = 0$, di mana I adalah matriks identitas.

Nilai eigen memiliki beberapa sifat penting. Multiplicitas eigen menunjukkan jumlah *eigenvector* yang terkait dengan nilai eigen tersebut. Nilai eigen juga memberikan informasi tentang sifat-sifat matriks, seperti diagonalisasi, kestabilan, dan determinan matriks. Selain itu, nilai eigen dan eigenvektor memainkan peran penting dalam merepresentasikan transformasi linear yang dilambangkan oleh matriks.

Nilai eigen memiliki berbagai aplikasi di bidang matematika dan ilmu terapan. Mereka digunakan dalam pemecahan sistem persamaan linear, analisis spektral, pengolahan citra, pengembangan algoritma, dan optimisasi. Dalam konteks matriks, nilai eigen memberikan wawasan penting tentang sifat dan perilaku sistem linier yang direpresentasikan oleh matriks tersebut.

E. VEKTOR EIGEN

Vektor eigen (*eigenvector*) adalah vektor non-nol yang terkait dengan nilai eigen dari sebuah matriks persegi. Vektor eigen adalah vektor yang tidak berubah arah, kecuali mungkin dengan skalar, ketika dikalikan dengan matriks.

Untuk matriks persegi A dan nilai eigen λ , vektor v yang bukan nol disebut vektor eigen dari A jika memenuhi persamaan $Av = \lambda v$, di mana Av adalah hasil perkalian matriks A dengan v .

Dalam mencari vektor eigen, kita mencari vektor yang memenuhi persamaan $(A - \lambda I)v = 0$, di mana I adalah matriks identitas.

Vektor eigen memiliki beberapa sifat penting. Untuk setiap nilai eigen, ada banyak vektor eigen yang mungkin terkait, tetapi vektor eigen hanya ditentukan hingga faktor skalar yang berbeda. Kumpulan semua vektor eigen yang terkait dengan suatu nilai eigen membentuk ruang eigen terkait. Vektor eigen dan nilai eigen terkait memainkan peran penting dalam merepresentasikan transformasi linear yang dilambangkan oleh matriks.

Vektor eigen memiliki banyak aplikasi dalam matematika dan ilmu terapan, seperti pemecahan masalah sistem persamaan linear, diagonalisasi matriks, analisis spektral, kompresi data, pengolahan citra, dan pengembangan algoritma. Dalam konteks matriks, vektor eigen memberikan wawasan berharga tentang perubahan yang terjadi pada vektor-vektor khusus yang terkait dengan matriks tersebut.

F. DIAGONALISASI

Diagonalisasi matriks melibatkan penggunaan polynomial karakteristik, yaitu sebuah polynomial dengan koefisien dalam suatu lapangan yang memenuhi persamaan $p(A) = \det(A - \lambda I) = 0$. Diagonalisasi matriks adalah proses mengubah matriks menjadi bentuk diagonal yang lebih sederhana dengan menggunakan vektor eigen.

Untuk melakukan diagonalisasi, kita perlu menemukan nilai eigen dan vektor eigen terkait dari matriks menggunakan polynomial karakteristik. Jika matriks memiliki n vektor

eigen yang linear independen, kita dapat membentuk matriks P dengan kolom-kolomnya sebagai vektor eigen tersebut. Selanjutnya, matriks diagonal D dapat ditemukan menggunakan invers dari matriks P , yaitu $D = P^{-1}AP$.

Matriks diagonal memiliki elemen-elemen diagonal yang merupakan nilai eigen, sedangkan elemen-elemen di luar diagonal adalah nol. Diagonalisasi matriks memungkinkan kita untuk menganalisis dan memanipulasi matriks dengan lebih mudah karena sifat-sifat khusus dari matriks diagonal.

Proses diagonalisasi dan penggunaan polynomial karakteristik sangat penting dalam berbagai aplikasi matematika, termasuk dalam pemecahan masalah sistem persamaan linear, perhitungan nilai pangkat tinggi matriks, dan analisis struktur matriks secara umum.

G. SINGULAR VALUE DECOMPOSITION

Singular Value Decomposition (SVD) adalah sebuah metode dalam aljabar linear yang penting untuk memecah suatu matriks menjadi tiga bagian utama: matriks singular value, matriks kolom singular, dan matriks baris singular.

Dalam SVD, matriks awal A dengan ukuran $(m \times n)$ dapat dipisahkan menjadi tiga matriks sebagai berikut:

$$A = U\Sigma V^T$$

- Matriks U adalah matriks kolom singular $(m \times m)$, yang berisi vektor kolom ortogonal yang disebut vektor singular kiri.
- Matriks Σ adalah matriks diagonal singular value $(m \times n)$, yang berisi nilai singular dari matriks A yang diurutkan secara menurun.
- Matriks V^T adalah matriks baris singular $(n \times n)$, yang berisi vektor baris ortogonal yang disebut vektor singular kanan.

Singular Value Decomposition memiliki beberapa karakteristik penting:

1. Matriks singular value (Σ) adalah matriks diagonal dengan nilai singular non-negatif diatas diagonal utama, dan bernilai nol di tempat lainnya.
2. Matriks kolom singular (U) dan matriks baris singular (V) terdiri dari vektor-vektor ortogonal yang membentuk basis untuk ruang kolom dan ruang baris matriks A .
3. Vektor-vektor singular kiri (kolom-kolom U) dan vektor-vektor singular kanan (baris-baris) saling tegak lurus.

4. SVD memberikan representasi terbaik untuk matriks A dalam bentuk aproksimasi rang rendah. Nilai singular terbesar memberikan kontribusi terbesar dalam aproksimasi matriks A .

SVD digunakan dalam berbagai bidang seperti kompresi data, pengolahan citra, analisis faktor, dan pemrosesan sinyal. Metode ini memungkinkan analisis yang mendalam terhadap struktur matriks dan membantu dalam pemahaman dan manipulasi data yang kompleks.

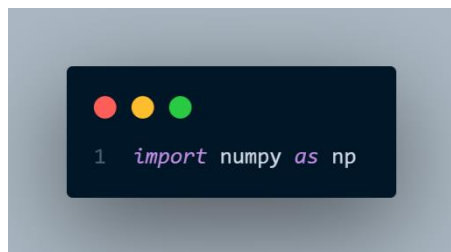
BAB III

IMPLEMENTASI PROGRAM

Pada bab ini, akan dijelaskan bagian-bagian dari program. Main Program akan mengatur alur utama program dan menerima input dari pengguna. Class "Matrix" akan mengimplementasikan fungsi-fungsi (method) untuk melakukan operasi matriks seperti SVD, diagonalisasi, dan lain-lain. Sementara itu, fungsi-fungsi di luar class akan memberikan dukungan bagi operasi-operasi utama.

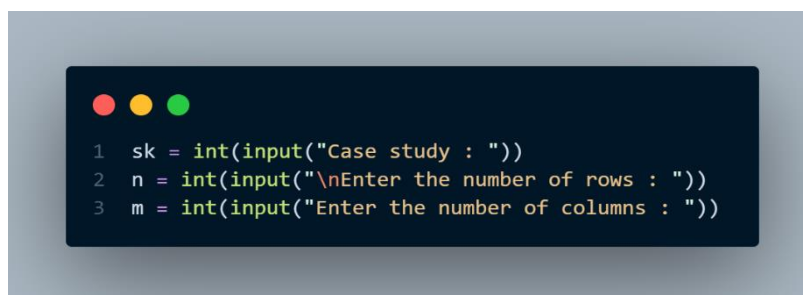
A. MAIN PROGRAM

1. Import NumPy



Kode di atas merupakan pernyataan impor yang memungkinkan kita menggunakan fungsi dan fitur yang disediakan oleh NumPy dalam program kita. NumPy adalah modul Python yang kuat untuk komputasi numerik, termasuk operasi matriks, aljabar linear, dan banyak lagi.

2. Mengambil input dari pengguna



Kode di atas bertujuan untuk meminta pengguna untuk memasukkan studi kasus, jumlah baris, dan jumlah kolom matriks yang ingin dioperasikan.

3. Membuat matriks A dan vektor b

```
1  if sk == 5:
2      A = np.zeros((n, m), dtype=np.complex_)
3      b = np.zeros(n, dtype=np.complex_)
4  else:
5      A = np.zeros((n, m))
6      b = np.zeros(n)
```

Kode di atas bertujuan untuk menginisialisasi matriks A dan vektor b dengan nilai nol menggunakan fungsi np.zeros. Jika studi kasus adalah 5, maka tipe data kompleks digunakan untuk matriks dan vektor. Sedangkan jika tidak, maka tipe data float yang digunakan.

4. Memasukkan matriks A dari user

```
1  print("\nEnter the matrix A (or coefficients) :")
2  for i in range(n):
3      if sk != 5:
4          A[i] = list(map(float, input().split()))
5      else:
6          row_i = input()
7          A[i] = [complex(x.strip()) for x in row_i.split()]
8
9  matrix_to_txt('A', A, 'no{}.txt'.format(sk))
```

Kode di atas bertujuan untuk memasukkan elemen-elemen matriks A baris per baris. Jika studi kasus bukan 5, elemen-elemen dimasukkan sebagai float. Jika studi kasus adalah 5, elemen-elemen diinterpretasikan sebagai kompleks dan diubah menjadi tipe data kompleks menggunakan fungsi complex(). Selanjutnya, kode akan memanggil fungsi matrix_to_txt untuk menyimpan matriks A ke file teks dengan nama file yang sesuai.

5. Memasukkan vektor b dari user

```
1 if sk < 3:
2     b = list(map(float, input("\nEnter the matrix b (constants) :\n").split()))
3     matrix_to_txt('b', b, 'no{}.txt'.format(sk))
4 elif sk == 5:
5     y = input("\nEnter the matrix b (constants) :\n")
6     b = [complex(x.strip()) for x in y.split()]
7     matrix_to_txt('b', b, 'no{}.txt'.format(sk))
```

Jika studi kasus kurang dari 3, pengguna diminta memasukkan elemen-elemen vektor b sebagai float. Jika studi kasus adalah 5, pengguna diminta memasukkan elemen-elemen vektor b sebagai kompleks.

6. Membuat objek Matrix dan menjalankan operasi sesuai studi kasus

```
1 matrix = Matrix(n, m, A, b)
2 if sk == 1 or sk == 2:
3     matrix.spl(sk)
4 elif sk == 3:
5     matrix.poly_eigen_diag()
6 elif sk == 4:
7     matrix.svd()
8 else:
9     matrix.svd_complex()
```

Membuat objek Matrix dengan menggunakan nilai n, m, A, dan b yang telah dimasukkan sebelumnya. Kemudian, berdasarkan studi kasus yang dipilih, metode yang sesuai pada objek Matrix dipanggil untuk menjalankan operasi yang diperlukan.

B. CLASS “MATRIX”

1. Constructor

```
1 def __init__(self, n, m, A, b):
2     self.n = n
3     self.m = m
4     self.A = A
5     self.b = b
```

Metode ini merupakan constructor (pembuat objek) untuk class Matrix. Menerima argumen n dan m sebagai jumlah baris dan kolom matriks, serta argumen A dan b sebagai matriks dan vektor yang akan digunakan sebagai attribute dalam class Matrix.

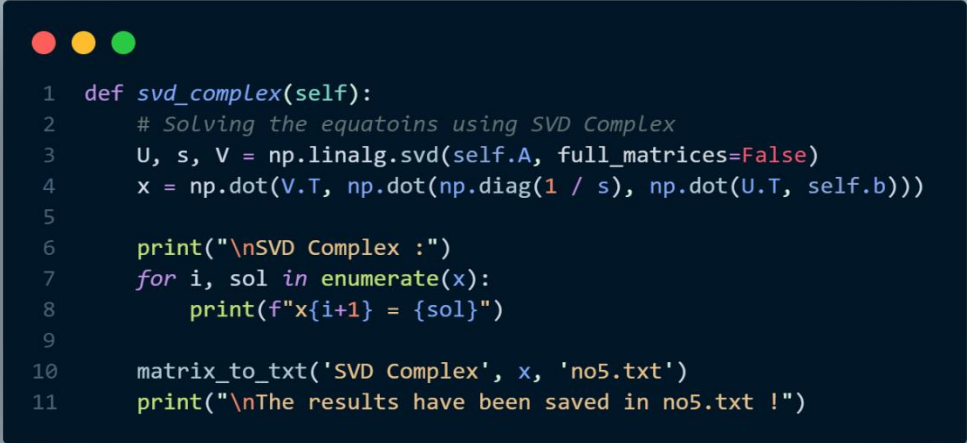
2. Method svd(self)

```
1 def svd(self):
2     # Solving the matrix using SVD
3     U, x, V = np.linalg.svd(self.A)
4     U = limit(U)
5     x = limit(x)
6     V = limit(V)
7
8     print("\nU :")
9     print(U)
10    matrix_to_txt('U', U, 'no4.txt')
11
12    print("\nSVD :")
13    print(x)
14    matrix_to_txt('SVD', x, 'no4.txt')
15
16    print("\nV :")
17    print(V)
18    matrix_to_txt('V', V, 'no4.txt')
19
20    print("\nThe results have been saved in no4.txt !")
```

Method ini digunakan untuk menyelesaikan matriks dengan menggunakan Singular Value Decomposition (SVD). Matriks A dipecah menjadi U, x, dan V menggunakan fungsi np.linalg.svd(). Hasilnya kemudian dibatasi dengan fungsi

limit() dan dicetak ke layar. Selain itu, matriks-matriks tersebut juga disimpan dalam file teks menggunakan fungsi matrix_to_txt().

3. Method svd_complex(self)



```
1 def svd_complex(self):
2     # Solving the equatoins using SVD Complex
3     U, s, V = np.linalg.svd(self.A, full_matrices=False)
4     x = np.dot(V.T, np.dot(np.diag(1 / s), np.dot(U.T, self.b)))
5
6     print("\nSVD Complex :")
7     for i, sol in enumerate(x):
8         print(f"x{i+1} = {sol}")
9
10    matrix_to_txt('SVD Complex', x, 'no5.txt')
11    print("\nThe results have been saved in no5.txt !")
```

Method ini digunakan untuk menyelesaikan persamaan matriks menggunakan Singular Value Decomposition Complex (SVD Complex). Matriks A dipecah menjadi U, s, dan V menggunakan fungsi np.linalg.svd() dengan opsi full_matrices=False. Selanjutnya, vektor solusi x dihitung menggunakan operasi matriks. Hasilnya dicetak ke layar dan disimpan dalam file teks menggunakan fungsi matrix_to_txt().

4. Method `poly_eigen_diag(self)`

```
1 def poly_eigen_diag(self):
2     # Find polynomial characteristic, eigenval, eigenvecs, and diagonalize
3     characteristic_poly = find_characteristic_poly(self.A)
4     eigenvalues, eigenvectors = find_eigen(self.A)
5     P, D, diagonalize = find_diagonalize(self.A)
6
7     eigenvalues = limit(eigenvalues)
8     eigenvectors = limit(eigenvectors)
9
10    print("\nPolynomial Characteristic:")
11    print(characteristic_poly)
12    matrix_to_txt('Polynomial Characteristic', characteristic_poly, 'no3.txt')
13
14    print("\nEigenvalues:")
15    print(eigenvalues)
16    matrix_to_txt('Eigenvalues', eigenvalues, 'no3.txt')
17
18    print("\nEigenvectors:")
19    print(eigenvectors)
20    matrix_to_txt('Eigenvectors', eigenvectors, 'no3.txt')
21
22    print("\nP:")
23    print(P)
24    matrix_to_txt('P', P, 'no3.txt')
25
26    print("\nD:")
27    print(D)
28    matrix_to_txt('D', D, 'no3.txt')
29
30    print("\nDiagonalized A:")
31    print(diagonalize)
32    matrix_to_txt("Diagonalized A", diagonalize, 'no3.txt')
33
34    print("\nThe results have been saved in no3.txt !")
```

Method ini digunakan untuk mencari polinomial karakteristik, eigenvalue, eigenvector, dan matriks diagonal. Polinomial karakteristik ditemukan menggunakan fungsi `find_characteristic_poly()`, eigenvalue dan eigenvector ditemukan menggunakan fungsi `find_eigen()`, dan matriks diagonal ditemukan menggunakan fungsi `find_diagonalize()`. Hasilnya dicetak ke layar dan disimpan dalam file teks menggunakan fungsi `matrix_to_txt()`.

5. Method `spl(self, sk)`

```
1 def spl(self, sk):
2
3     print("\nMenu :")
4     print("a. Normal Method")
5     print("b. Least-squares Method")
6     print("c. Gaussian Elimination Method")
7
8     menu = input("Pilih : ")
```

Method ini digunakan untuk melakukan pemecahan sistem persamaan linear. Bergantung pada nilai `sk`, pengguna akan diminta untuk memilih metode pemecahan antara metode Normal, Least-squares, atau Gaussian Elimination. Metode yang dipilih kemudian dijalankan, solusi dicetak ke layar, dan disimpan dalam file teks menggunakan fungsi `matrix_to_txt()`.

a. Normal

```
1 if menu == 'a':
2     print("\nNormal Method :")
3     try:
4         solutions = limit(np.linalg.solve(self.A, self.b))
5         for i, sol in enumerate(solutions):
6             print(f"x{i+1} = {sol}")
7         matrix_to_txt('Normal Method', solutions, 'no{}.txt'.format(sk))
8     except np.linalg.LinAlgError:
9         print("No solution!")
10    with open('no{}.txt'.format(sk), 'a') as file:
11        file.write("\nNormal Method :\n")
12        file.write("No solution!\n")
```

Jika pengguna memilih menu "a", metode akan menjalankan metode "Normal Method". Metode ini menggunakan fungsi `np.linalg.solve()` untuk mencari solusi persamaan linear menggunakan metode eliminasi Gauss. Hasil kemudian dicetak ke layar dan disimpan dalam file "no{}.txt" sesuai dengan kasus studi yang dipilih. Jika sistem tidak memiliki solusi, pesan "No solution!" akan dicetak ke layar dan ditulis ke dalam file teks.

b. Least-squares

```
1 elif menu == 'b':
2     print("\nLeast-squares Method :")
3     try:
4         solutions, residuals, rank, s = np.linalg.lstsq(self.A, self.b, rcond=None)
5         solutions = limit(solutions)
6         for i, sol in enumerate(solutions):
7             print(f"x{i+1} = {sol}")
8         matrix_to_txt('Least-squares Method', solutions, 'no{}.txt'.format(sk))
9     except np.linalg.LinAlgError:
10        print("No solution!")
11        with open('no{}.txt'.format(sk), 'a') as file:
12            file.write("\nLeast-squares Method :\n")
13            file.write("No solution!\n")
```

Jika pengguna memilih menu "b", metode akan menjalankan metode "Least-squares Method". Metode ini menggunakan fungsi `np.linalg.lstsq()` untuk mencari solusi persamaan linear menggunakan metode kuadrat terkecil (*least-squares*). Hasil kemudian dicetak ke layar dan disimpan dalam file "no{}.txt" sesuai dengan kasus studi yang dipilih. Jika sistem tidak memiliki solusi, pesan "No solution!" akan dicetak ke layar dan ditulis ke dalam file teks.

c. Gaussian

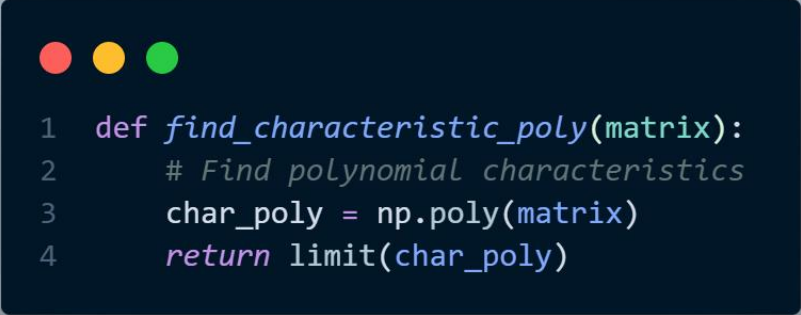
```
1 else:
2     solutions = gaussian_elimination(self.A, self.b)
3
4     print("\nGaussian Elimination :")
5     if type(solutions) == str:
6         print(solutions)
7         with open('no{}.txt'.format(sk), 'a') as file:
8             file.write("\nGaussian Elimination :\n")
9             file.write(f"{solutions}\n")
10    else:
11        solutions = limit(solutions)
12        for i, sol in enumerate(solutions):
13            print(f"x{i+1} = {sol}")
14        matrix_to_txt('Gaussian Elimination Method', solutions, 'no{}.txt'.format(sk))
15
16 print("\nThe results have been saved in no{}.txt !".format(sk))
```

Jika pengguna memilih menu "c", metode akan menjalankan metode "Gaussian Elimination". Metode ini menggunakan fungsi `gaussian_elimination()` untuk mencari solusi persamaan linear menggunakan metode eliminasi Gauss. Solusi-solusi ini kemudian dicetak ke layar dan disimpan dalam file "no{}.txt" sesuai dengan kasus studi yang dipilih. Jika sistem tidak memiliki solusi atau memiliki

solusi tak terhingga, pesan yang sesuai akan dicetak ke layar dan ditulis ke dalam file teks.

C. FUNCTIONS

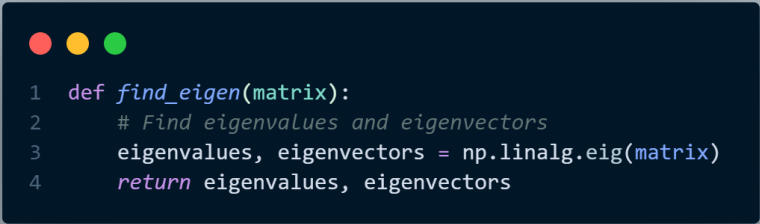
1. Karakteristik Polinomial



```
1 def find_characteristic_poly(matrix):  
2     # Find polynomial characteristics  
3     char_poly = np.poly(matrix)  
4     return limit(char_poly)
```

Fungsi ini digunakan untuk mencari polinomial karakteristik dari suatu matriks. Menggunakan fungsi `np.poly()`, fungsi ini menghitung polinomial karakteristik matriks dan mengembalikan hasilnya setelah dibatasi dengan fungsi `limit()`.

2. Eigenvalue dan Eigenvector



```
1 def find_eigen(matrix):  
2     # Find eigenvalues and eigenvectors  
3     eigenvalues, eigenvectors = np.linalg.eig(matrix)  
4     return eigenvalues, eigenvectors
```

Fungsi ini digunakan untuk mencari eigenvalue dan eigenvector dari suatu matriks. Menggunakan fungsi `np.linalg.eig()`, fungsi ini menghitung eigenvalue dan eigenvector matriks dan mengembalikan hasilnya.

3. Diagonalisasi

```
1 def find_diagonalize(matrix):  
2     # Find diagonalize matrix using eigen decomposition  
3     eigenvalues, eigenvectors = find_eigen(matrix)  
4     D = np.diag(eigenvalues)  
5     P = eigenvectors  
6     P_inv = np.linalg.inv(P)  
7  
8     # Compute  $P^{-1}AP$   
9     diagonalized_A = P_inv @ matrix @ P  
10  
11     return limit(P), limit(D), limit(diagonalized_A)
```

Fungsi ini digunakan untuk mencari matriks diagonal hasil dari diagonalisasi suatu matriks menggunakan eigen decomposition. Fungsi ini memanggil fungsi `find_eigen()` untuk mendapatkan eigenvalue dan eigenvector, kemudian menghitung matriks diagonal D dan matriks P menggunakan nilai tersebut. Selanjutnya, matriks diagonalized_A dihitung dengan melakukan operasi perkalian matriks antara P^{-1} , A , dan P . Hasilnya kemudian dikembalikan setelah dibatasi dengan fungsi `limit()`.

4. Eliminasi Gauss

```
1 def gaussian_elimination(matrix, vector):
2     n = len(matrix)
3     m = len(matrix[0])
4
5     for i in range(n):
6         if matrix[i][i] == 0:
7             for j in range(i+1, n):
8                 if matrix[j][i] != 0:
9                     matrix[i], matrix[j] = matrix[j], matrix[i]
10                    vector[i], vector[j] = vector[j], vector[i]
11                    break
12
13            if matrix[i][i] == 0:
14                if vector[i] == 0:
15                    return "Infinite solutions!"
16                else:
17                    return "No solution!"
18
19            pivot = matrix[i][i]
20            for j in range(i, m):
21                matrix[i][j] /= pivot
22            vector[i] /= pivot
23
24            for j in range(n):
25                if j != i:
26                    factor = matrix[j][i]
27                    for k in range(i, m):
28                        matrix[j][k] -= factor * matrix[i][k]
29                    vector[j] -= factor * vector[i]
30
31    return vector
```

Fungsi ini digunakan untuk melakukan eliminasi Gauss untuk sistem persamaan linear. Fungsi ini mengambil matriks `matrix` dan vektor `vector` sebagai argumen. Fungsi ini melakukan operasi eliminasi Gauss pada matriks dan vektor tersebut, menghasilkan solusi dalam bentuk vektor. Jika solusi tunggal ditemukan, solusinya dikembalikan. Jika ada solusi tak terhingga atau tidak ada solusi, pesan yang sesuai dikembalikan.

5. Limit

```
1 def limit(matrix):  
2     return np.round(matrix, 5)
```

Fungsi ini digunakan untuk membatasi presisi desimal pada nilai-nilai dalam matriks. Menggunakan fungsi `np.round()`, fungsi ini membatasi angka desimal pada 5 digit.

6. Matrix to TXT

```
1 def matrix_to_txt(title, matrix, file_name):  
2     with open(file_name, 'a') as file:  
3         file.write(f"\n{title} :\n")  
4         if file_name[2] == '1' or file_name[2] == '2' or file_name[2] == '5':  
5             for i, sol in enumerate(matrix):  
6                 file.write(f"x{i+1} = {sol}\n")  
7         else:  
8             for line in matrix:  
9                 file.write(f"{line}\n")
```

Fungsi ini digunakan untuk menyimpan matriks atau solusi ke dalam file teks. Fungsi ini menerima argumen `title` sebagai judul matriks atau solusi, `matrix` sebagai matriks atau solusi yang akan disimpan, dan `file_name` sebagai nama file teks yang akan digunakan. Fungsi ini membuka file dengan mode 'a' (append) dan menulis judul serta isi matriks ke dalam file tersebut.

BAB IV EKSPERIMEN

Berikut analisis hasil eksekusi dari program perhitungan persamaan linear bilangan kompleks menggunakan bahasa pemrograman Python:

1. Studi Kasus 1A

Carilah Penyelesaian dari sistem Persamaan Linear berikut ini :

$$A = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 5 & -7 & -5 \\ 2 & -1 & 1 & 3 \\ 5 & 2 & -4 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -2 \\ 4 \\ 6 \end{bmatrix}$$

Untuk menyelesaikan kasus tersebut, kami menggunakan program yang telah kami buat dan didapatkan hasil sebagai berikut :

```
Menu :  
a. Normal Method  
b. Least-squares Method  
c. Gaussian Elimination Method  
Pilih : c  
  
Gaussian Elimination :  
No solution!
```

Berdasarkan hasil yang didapat tersebut dapat diketahui bahwa sistem persamaan linear tersebut tidak memiliki penyelesaian. Hal ini dikarenakan pada pertengahan eliminasi, salah satu baris menghasilkan persamaan $0x + 0y - 14z - 16z = -1$, yang dapat disederhanakan menjadi $0 = -1$. Sehingga, persamaan tersebut tidak konsisten yang artinya sistem persamaan linear tersebut tidak memiliki solusi.

2. Studi Kasus 1B

Carilah Penyelesaian dari sistem Persamaan Linear berikut ini :

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & -3 & 0 \\ 2 & -1 & 0 & 1 & -1 \\ -1 & 2 & 0 & -2 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 6 \\ 5 \\ -1 \end{bmatrix}$$

Untuk menyelesaikan kasus tersebut, kami menggunakan program yang telah kami buat dan didapatkan hasil sebagai berikut :

```
Menu :  
a. Normal Method  
b. Least-squares Method  
c. Gaussian Elimination Method  
Pilih : c  
  
Gaussian Elimination :  
No solution!
```

Berdasarkan hasil yang didapat tersebut dapat diketahui bahwa sistem persamaan linear tersebut tidak memiliki penyelesaian. Hal ini dikarenakan pada pertengahan eliminasi, baris ke-3 menghasilkan persamaan $0x + 0y + 0z - 2 = 1$, yang dapat disederhanakan menjadi $w = -1$. Namun, pada baris ke-4 terdapat persamaan $0x + 0y + 0z - w = 0$, yang dapat disederhanakan menjadi $w = 0$. Sehingga, persamaan tersebut tidak konsisten yang artinya sistem persamaan linear tersebut tidak memiliki solusi.

3. Studi Kasus 2A

Selesaikan sistem persamaan berbentuk berikut :

$$\begin{aligned} 3x_1 + 8x_2 - 3x_3 - 14x_4 &= 2 \\ 2x_1 + 3x_2 - x_3 - 2x_4 &= 1 \\ x_1 - 2x_2 + x_3 + 10x_4 &= 0 \\ x_1 + 5x_2 - 2x_3 - 12x_4 &= 1 \end{aligned}$$

Untuk menyelesaikan kasus tersebut, kami menggunakan program yang telah kami buat dan didapatkan hasil sebagai berikut :

```
Menu :  
a. Normal Method  
b. Least-squares Method  
c. Gaussian Elimination Method  
Pilih : c  
  
Gaussian Elimination :  
No solution!
```

Berdasarkan hasil yang didapat tersebut dapat diketahui bahwa sistem persamaan linear tersebut tidak memiliki penyelesaian. Hal ini dikarenakan pada pertengahan eliminasi, baris ke-3 terdapat persamaan $0x + 0y - (4/3)z + (28/3)w = (-4/3)$, yang dapat disederhanakan menjadi $-4/3z + 28/3w = -4/3$.

Pada baris keempat, terdapat persamaan $0x + 0y - (2/3)z + (10/3)w = 0$, yang dapat disederhanakan menjadi $-2/3z + 10/3w = 0$.

Karena sistem persamaan linear menghasilkan dua persamaan dengan dua variabel (z dan w), kita dapat mencari nilai z dan w yang memenuhi kedua persamaan tersebut. Namun, sistem ini tidak memiliki persamaan yang menentukan nilai x dan y , sehingga kita tidak dapat menentukan solusi unik untuk sistem persamaan linear tersebut. Dengan demikian, sistem persamaan linear ini tidak memiliki penyelesaian.

4. Studi Kasus 2B

Selesaikan sistem persamaan berbentuk berikut :

$$\begin{aligned}x_1 - x_2 + x_3 - x_4 &= 0 \\ -x_1 + x_2 + x_3 + x_4 &= 0 \\ x_1 + x_2 - x_3 + x_4 &= 0 \\ x_1 + x_2 + x_3 + x_4 &= 0\end{aligned}$$

Untuk menyelesaikan kasus tersebut, kami menggunakan program yang telah kami buat dan didapatkan hasil sebagai berikut :

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

Berdasarkan hasil yang didapat tersebut dapat diketahui bahwa sistem persamaan linear tersebut memiliki penyelesaian yang tak terbatas. Hal ini dikarenakan pada pertengahan eliminasi, baris ketiga dan baris keempat, terdapat persamaan $0x + 0y + z + 0w = 0$ dan $0x + y + 0z + w = 0$, yang menyiratkan bahwa variabel z dan w adalah variabel tak-terikat (variabel bebas). Variabel x dan y dapat diungkapkan sebagai fungsi dari variabel z dan w .

Karena terdapat variabel bebas, sistem persamaan linear ini memiliki solusi yang tidak terbatas. Artinya, kita dapat menemukan banyak pasangan nilai (x, y, z, w) yang memenuhi sistem persamaan tersebut, yang mengakibatkan solusi yang tidak terbatas.

5. Studi Kasus 3A

Carilah polynomial characteristic, eigenvalues, eigenvector dan (jika mungkin) carilah matrik P yang mempunyai invers sehingga $P^{-1}AP$ adalah diagonal

$$g. A = \begin{bmatrix} 3 & 1 & 1 \\ -4 & -2 & -5 \\ 2 & 2 & 5 \end{bmatrix}$$

Untuk menyelesaikan kasus tersebut, kami menggunakan program yang telah kami buat dan didapatkan hasil sebagai berikut :

```

Polynomial Characteristic:
[ 1. -6. 11. -6.]

Eigenvalues:
[1. 2. 3.]

Eigenvectors:
[[ 0.30151 -0.70711  0.      ]
 [-0.90453  0.70711 -0.70711]
 [ 0.30151 -0.      0.70711]]

P:
[[ 0.30151 -0.70711  0.      ]
 [-0.90453  0.70711 -0.70711]
 [ 0.30151 -0.      0.70711]]

D:
[[1. 0. 0.]
 [0. 2. 0.]
 [0. 0. 3.]]

Diagonalized A:
[[ 1.  0.  0.]
 [-0.  2.  0.]
 [ 0. -0.  3.]]

```

Proses untuk menghitung eigenvalues dan eigenvectors melibatkan mencari nilai-nilai λ (lambda) yang memenuhi persamaan determinan ($|A - \lambda I| = 0$), di mana A adalah matriks asli dan I adalah matriks identitas. Setiap λ yang memenuhi persamaan tersebut adalah eigenvalue, sedangkan vektor yang terkait dengan setiap eigenvalue adalah eigenvector.

Eigenvalues dalam hal ini adalah 1.0, 2.0, dan 3.0, yang dihitung dari polynomial characteristic. Eigenvectors terkait dengan eigenvalues tersebut adalah [0.30151, -0.70711, 0.0], [-0.90453, 0.70711, -0.70711], dan [0.30151, 0.0, 0.70711].

Matriks P adalah matriks eigenvectors yang kolom-kolomnya merupakan eigenvectors terkait dengan eigenvalues yang sesuai. Matriks D adalah matriks diagonal yang memiliki eigenvalues pada diagonal utama. Diagonalized A adalah matriks A dalam bentuk diagonal, yang diperoleh dengan melakukan perhitungan $D = P^{-1} * A * P$, di mana P^{-1} adalah invers dari matriks P.

Dalam kasus ini, matriks A tidak sepenuhnya diagonal karena memiliki nilai-nilai di luar diagonal utama yang bukan nol. Namun, setelah dilakukan perhitungan eigenvectors dan eigenvalues, kita dapat mendapatkan matriks diagonal D yang memiliki eigenvalues pada diagonal utama, dan matriks P yang berisi eigenvectors sebagai kolom-kolomnya.

6. Studi Kasus 3B

Carilah polynomial characteristic, eigenvalues, eigenvector dan (jika mungkin) carilah matrik P yang mempunyai invers sehingga $P^{-1}AP$ adalah diagonal

$$\text{h. } A = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & -1 & 2 \end{bmatrix}$$

Untuk menyelesaikan kasus tersebut, kami menggunakan program yang telah kami buat dan didapatkan hasil sebagai berikut :

```
Polynomial Characteristic:
[ 1. -5.  7. -3.]

Eigenvalues:
[3. 1. 1.]

Eigenvectors:
[[ 0.70711 -0.70711 -0.70711]
 [ 0.      0.      0.      ]
 [ 0.70711  0.70711  0.70711]]

P:
[[ 0.70711 -0.70711 -0.70711]
 [ 0.      0.      0.      ]
 [ 0.70711  0.70711  0.70711]]

D:
[[3. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

Diagonalized A:
[[ 3.  0.  0.]
 [ 0.  1. -0.]
 [ 0.  0.  1.]]
```

Proses untuk menghitung eigenvalues dan eigenvectors melibatkan mencari nilai-nilai λ (lambda) yang memenuhi persamaan determinan ($|A - \lambda I| = 0$), di mana A adalah matriks asli dan I adalah matriks identitas. Setiap λ yang memenuhi persamaan tersebut adalah eigenvalue, sedangkan vektor yang terkait dengan setiap eigenvalue adalah eigenvector.

Eigenvalues dalam hal ini adalah 3.0, 1.0, dan 1.0, yang dihitung dari polynomial characteristic. Eigenvectors terkait dengan eigenvalues tersebut adalah $[0.70711, -0.70711, -0.70711]$, $[0.0, 0.0, 0.0]$, dan $[0.70711, 0.70711, 0.70711]$.

Matriks P adalah matriks eigenvectors yang kolom-kolomnya merupakan eigenvectors terkait dengan eigenvalues yang sesuai. Matriks D adalah matriks diagonal yang memiliki eigenvalues pada diagonal utama. Diagonalized A adalah matriks A dalam bentuk diagonal, yang diperoleh dengan melakukan perhitungan $D = P^{-1} * A * P$, di mana P^{-1} adalah invers dari matriks P.

Namun, perhatikan bahwa dalam kasus ini, terdapat eigenvalue yang memiliki eigenvector yang bernilai nol $[0.0, 0.0, 0.0]$. Hal ini menunjukkan bahwa eigenvector yang terkait dengan eigenvalue tersebut adalah vektor nol atau vektor trivial. Eigenvector ini tidak dapat digunakan untuk membentuk matriks P, sehingga kolom-kolom matriks P yang terkait dengan eigenvalue 1.0 menjadi $[0.0, 0.0, 0.0]$.

Dalam diagonalized A, kolom-kolom yang terkait dengan eigenvalue 1.0 pada matriks D menjadi $[0.0, 1.0, 0.0]$ karena tidak ada eigenvector yang terkait dengan eigenvalue tersebut.

Dengan demikian, hasil yang diperoleh untuk eigenvectors, matriks P, matriks D, dan diagonalized A menunjukkan bahwa matriks tersebut memiliki eigenvalues yang berbeda-beda dan matriks A dapat dibentuk menjadi matriks diagonal menggunakan eigenvectors dan eigenvalues yang sesuai, namun perlu diperhatikan bahwa terdapat eigenvector yang bernilai nol.

7. Studi Kasus 4A

Carilah sebuah SVD dari matrik berikut ini :

$$\text{a. } A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Untuk menyelesaikan kasus tersebut, kami menggunakan program yang telah kami buat dan didapatkan hasil sebagai berikut :

```

U :
[[-0.8165  0.      -0.57735]
 [ 0.40825 -0.70711 -0.57735]
 [-0.40825 -0.70711  0.57735]]

SVD :
[1.73205 1.      ]

V :
[[-0.70711  0.70711]
 [-0.70711 -0.70711]]

```

SVD memberikan dekomposisi matriks awal menjadi tiga komponen: matriks U yang menggambarkan transformasi dari ruang asal ke ruang singular, matriks S yang menggambarkan "stretched" atau "compressed" dari ruang singular, dan matriks V yang menggambarkan transformasi dari ruang singular ke ruang tujuan. Secara matematis, matriks asli dapat dinyatakan sebagai $U * S * V^T$.

8. Studi Kasus 4B

Carilah sebuah SVD dari matrik berikut ini :

$$b. \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & -2 \\ 1 & 2 & 0 \end{bmatrix}$$

Untuk menyelesaikan kasus tersebut, kami menggunakan program yang telah kami buat dan didapatkan hasil sebagai berikut :

```

U :
[[-0.57735  0.      -0.8165 ]
 [ 0.57735 -0.70711 -0.40825]
 [-0.57735 -0.70711  0.40825]]

SVD :
[3.  2.  0.]

V :
[[-0.57735 -0.57735 -0.57735]
 [ 0.      -0.70711  0.70711]
 [ 0.8165  -0.40825 -0.40825]]

```

SVD memberikan dekomposisi matriks awal menjadi tiga komponen: matriks U yang menggambarkan transformasi dari ruang asal ke ruang singular, matriks S yang menggambarkan "stretched" atau "compressed" dari ruang singular, dan matriks V yang

menggambarkan transformasi dari ruang singular ke ruang tujuan. Secara matematis, matriks asli dapat dinyatakan sebagai $U * S * V^T$.

9. Studi Kasus 5

Selesaikan sistem persamaan linear kompleks dengan menggunakan SVD. Diberikan sistem persamaan linear kompleks dengan 6 persamaan dan 5 variabel sebagai berikut:

$$\begin{aligned} ix_1 - x_3 + x_5 &= 3 \\ x_1 + x_2 - ix_3 &= 4i \\ 2x_1 + 5ix_3 &= -3i \\ 2ix_2 + 2x_3 &= -5 \\ x_4 + x_5 &= 5 \\ ix_1 + ix_2 &= -3 \end{aligned}$$

Untuk menyelesaikan kasus tersebut, kami menggunakan program yang telah kami buat dan didapatkan hasil sebagai berikut :

```
SVD Complex :
x1 = (9.93980007734975e-16+5.18055555555553j)
x2 = (-3.7389718323779235e-16-5.54166666666665j)
x3 = (2.77777777777777-5.385455230331274e-16j)
x4 = (4.40277777777777+1.4439216805301409e-15j)
x5 = (0.5972222222222272-2.86150720776992e-16j)
```

Berdasarkan hasil yang didapat tersebut dapat diketahui nilai dari masing-masing X dari sistem persamaan linear kompleks. hasil yang diperoleh berupa desimal dan kekurangan yang ada dan hanya menampilkan hasil akhir dari program saja tanpa menunjukan proses pengerjaannya.

BAB V

KESIMPULAN

A. KESIMPULAN

Dalam proyek ini, kami berhasil mengembangkan aplikasi Aljabar Linear menggunakan bahasa pemrograman Python. Aplikasi ini mencakup beberapa fungsi penting dalam Aljabar Linear, termasuk penyelesaian Sistem Persamaan Linier (SPL) menggunakan metode Normal, Metode Least-squares, dan Metode Eliminasi Gauss. Selain itu, kami juga mengimplementasikan fungsi untuk mencari Polynomial Characteristic, eigenvalue, eigenvector, dan diagonalisasi matriks menggunakan eigen decomposition. Terakhir, kami mengimplementasikan Singular Value Decomposition (SVD) untuk menyelesaikan SPL kompleks.

Melalui proyek ini, kami mencapai beberapa hasil penting. Pertama, kami berhasil mengembangkan program yang dapat memecahkan SPL dengan akurat menggunakan berbagai metode, seperti metode Normal, metode Least-squares, dan metode Eliminasi Gauss. Ini memungkinkan kami untuk menyelesaikan SPL dalam berbagai kasus, termasuk SPL dengan solusi tunggal, solusi tak terhingga, dan tanpa solusi.

Kedua, kami berhasil mengimplementasikan fungsi-fungsi yang memungkinkan kami melakukan analisis matriks yang lebih dalam. Kami dapat mencari Polynomial Characteristic, Eigenvalue, Eigenvector, dan melakukan diagonalisasi matriks. Ini memberikan pemahaman yang lebih baik tentang struktur dan sifat matriks yang diberikan.

Ketiga, kami juga berhasil mengimplementasikan Singular Value Decomposition (SVD) untuk menyelesaikan SPL kompleks. SVD adalah alat yang penting dalam Aljabar Linear dan memiliki aplikasi yang luas dalam berbagai bidang. Kami dapat mengaplikasikan SVD untuk mencari solusi SPL dalam kasus SPL kompleks.

B. SARAN PENGEMBANGAN

Meskipun kami mencapai hasil yang signifikan dalam proyek ini, masih ada beberapa saran pengembangan yang dapat meningkatkan kualitas aplikasi Aljabar Linear kami. Beberapa saran pengembangan yang dapat dipertimbangkan adalah:

1. Pengembangan antarmuka pengguna: Menambahkan antarmuka pengguna yang lebih interaktif dan intuitif akan meningkatkan pengalaman pengguna dalam menggunakan aplikasi. Antarmuka pengguna grafis (GUI) atau antarmuka konsol yang lebih ramah

pengguna dapat mempermudah pengguna dalam memasukkan matriks, memilih metode penyelesaian SPL, dan melihat hasil dengan jelas.

2. Pengoptimalan kinerja: Melakukan pengoptimalan kinerja pada algoritma penyelesaian SPL dan implementasi SVD akan meningkatkan efisiensi aplikasi. Penggunaan teknik yang lebih efisien atau pemrosesan paralel dapat mengurangi waktu eksekusi dan meningkatkan kecepatan aplikasi, terutama dalam kasus matriks yang lebih besar.
3. Penanganan eror yang lebih baik: Meningkatkan penanganan error dan pesan kesalahan akan membuat aplikasi lebih tanggap dan membantu pengguna dalam menyelesaikan masalah atau kesalahan yang mungkin terjadi selama penggunaan aplikasi.

C. REFLEKSI

Proyek ini memberi kami pemahaman yang lebih mendalam tentang Aljabar Linear dan penerapannya dalam pemrograman Python. Kami mempelajari berbagai metode penyelesaian SPL, analisis matriks, dan konsep seperti eigenvalue, eigenvector, dan diagonalisasi matriks. Kami juga memperluas pengetahuan kami tentang Singular Value Decomposition (SVD) dan penggunaannya dalam menyelesaikan SPL kompleks.

Selama proyek ini, kami menghadapi tantangan dalam mengimplementasikan algoritma-algoritma yang kompleks dan memastikan keakuratan hasil. Namun, dengan dedikasi dan penelitian yang teliti, kami berhasil mengatasi hambatan-hambatan tersebut dan menghasilkan aplikasi Aljabar Linear yang fungsional.

Proyek ini juga memberi kami kesempatan untuk mengembangkan keterampilan pemrograman Python kami, terutama dalam penggunaan pustaka NumPy untuk operasi matriks. Kami meningkatkan pemahaman kami tentang pemrograman berorientasi objek dan penggunaan fungsi-fungsi NumPy yang kuat.

Secara keseluruhan, proyek ini memberi kami pengalaman berharga dalam menerapkan konsep-konsep Aljabar Linear dalam pemrograman Python. Kami merasa lebih percaya diri dalam kemampuan kami untuk menyelesaikan masalah SPL, menganalisis matriks, dan memahami konsep-konsep yang terkait. Proyek ini juga membantu kami memperluas pengetahuan kami tentang aplikasi Aljabar Linear dalam berbagai bidang ilmu dan teknik.

DAFTAR PUSTAKA

- Ahmad, I. H., & Ratnasari, L. (2011). *MENYELESAIKAN SISTEM PERSAMAAN LINIER MENGGUNAKAN ANALISIS SVD*.
- Eliminasi Gauss Jordan beserta Contoh Penerapannya - Profematika. (2019). Diambil 25 Juni 2023, dari <https://www.profematika.com/eliminasi-gauss-jordan-beserta-contoh-penerapannya/>
- Maryatun, Siswanto, & Wiyono, S. B. (t.t.). *POLINOMIAL KARAKTERISTIK MATRIKS DALAM ALJABAR MAKS-PLUS*.
- Matriks Eselon Baris dan Eselon Baris Tereduksi. (t.t.). Diambil 25 Juni 2023, dari <https://jagostat.com/aljabar-linear/bentuk-eselon-baris-dan-eselon-baris-tereduksi>
- Munir, R. (2018). *Matriks Eselon Bahan kuliah IF2123 Aljabar Geometri Program Studi Informatika ITB*.
- Munir, R. (2020). *Singular Value Decomposition (SVD) Bahan kuliah IF2123 Aljabar Linier dan Geometri Seri bahan kuliah Algeo #19b*.
- Sistem Persamaan Linear – Menara Ilmu Aljabar Linear. (t.t.). Diambil 25 Juni 2023, dari <https://aljabarlinear.mipa.ugm.ac.id/category/matriks/sistem-persamaan-linear/>