BAB II
MATRIKS

I. TUJUAN

- Mahasiswa dapat membuat matriks dan melakukan fungsi-fungsi dalam matriks menggunakan Python dengan library NumPy.
- Mahasiswa dapat melakukan operasi dalam matriks menggunakan NumPy.

II. PENDAHULUAN

Matriks adalah kelompok bilangan yang disusun dalam bentuk persegi atau persegi Panjang yang terdiri atas baris-baris atau kolom-kolom. Dengan library NumPy di Python, berbagai operasi matriks seperti penjumlahan, pengurangan, perkalian, invers, transpose, dan lainnya dapat dilakukan secara efisien. NumPy menyediakan berbagai fungsi untuk memanipulasi matriks, seperti operasi aljabar matriks dan operasi elemen per elemen. NumPy memungkinkan perhitungan numerik yang mudah untuk berbagai aplikasi.

Operasi Penjumlahan dan Pengurangan

Operasi penjumlahan dua matriks, A+B, dan selisih dua matriks, A–B, terdefinisi apabila matrik A dan B berukuran sama. Penjumlahan menambahkan elemen-elemen yang bersesuaian dalam matriks, sedangkan pengurangan mengurangkannya. Walau demikian, penjumlahan atau pengurangan juga bisa dilakukan antara matriks dengan skalar.

Operasi Perkalian

Perkalian matriks, seperti C = AB, dilakukan menggunakan aturan aljabar matriks, yaitu menjumlahkan hasil kali elemen-elemen baris pada matriks pertama dengan elemen-elemen kolom pada matriks kedua.

Transposisi

Salah satu operasi yang utama dalam matriks adalah transposisi, yang berfungsi untuk mempertukarkan baris dan kolom dalam suatu matriks atau vector. Dalam NumPy, transposisi dapat dilakukan menggunakan atribut **.T** atau fungsi **np.transpose()**.

- **.T**, Atribut bawaan untuk melakukan transposisi pada matriks atau array.
- **np.transpose()**, fungsi untuk melakukan transposisi dengan opsi tambahan seperti pengaturan urutan dimensi (khusus array multidimensi).

Operasi Elemen

Di dalam Python dengan NumPy, operasi matematika dapat dilakukan pada setiap elemen dalam array atau matriks. Matriks atau vektor yang terlibat harus memiliki ukuran yang sama. Operasi yang dapat dilakukan meliputi:

- Penjumlahan dan Pengurangan Elemen.
- Perkalian Elemen-per-Elemen.
- Pembagian Elemen-per-Elemen.
- Pangkat Elemen-per-Elemen.

NumPy juga menyediakan banyak fungsi matriks yang berguna untuk menyelesaikan berbagai permasalahan dalam berbagai bidang seperti: komputasi numerik, pengolahan citra, dan kecerdasan buatan. Beberapa fungsi matriks yang umum digunakan dalam NumPy adalah:

- *np.linalg.det*(*A*) : determinan matriks,
- *np.linalg.eig*(*A*) : nilai *eigenvalues* dan *eigenvectors*.
- *np.linalg.inv(A)* : invers matriks.
- *np.linalg.cholesky*(*A*) : faktorisasi *Cholesky.*
- *Scipy.linalg.lu*(*A*) : eliminasi *gauss* (*LU Decomposition*).
- *np.linalg.matrix_rank*(*A*) : rank matriks.

Fungsi matriks khusus dalam NumPy adalah sebagai berikut:

- *np.empty()* : matriks kosong.

- *np.eye()* : matriks identitas.
- *np.ones()* : matriks dengan semua elemen bernilai satu.
- *np.zeros()* : matriks dengan semua elemen bernilai nol.
- *np.random.rand()* : matriks dengan nilai acak.

Format data dalam NumPy terdapat empat jenis, yaitu skalar, vektor, matriks, dan array multidimensi.

- Skalar, ialah sebuah bilangan tunggal.
- Vektor, ialah sekelompok bilangan yang tersusun 1-dimensi. Dalam Numpy umumnya disajikan array 1-dimensi.
- Matriks, ialah sebuah array 2-dimensi yang memiliki baris dan kolom. Di dalam NumPy, matriks didefinisikan dengan jumlah baris dan kolomnya.
- Array Multidimensi, ialah masebuah array dengan lebih dari dua dimensi untuk merepresentasikan data yang lebih kompleks.

.

III. LANGKAH PRAKTIKUM

Instalasi Library Numpy

Sebelum menggunakan library NumPy, kita perlu menginstal *library*/*package* tersebut dalam Python lokal kita dengan perintah sebagai berikut:

```
pip install numpy
```

Pendefinisian Matriks

```
import numpy as np

# Skalar
scalar = np.array(5)
print("Skalar:", scalar)

# Vektor baris (1×n)
row_vector = np.array([1, 2, 3])
print("Vektor Baris:", row_vector)

# Vektor kolom (n×1)
column_vector = np.array([[1], [2], [3]])
print("Vektor Kolom:\n", column_vector)
```

```
# Matriks 2D
matrix = np.array([[1, 2, 3], [4, 5, 6]])
print("Matriks 2D:\n", matrix)

# Matriks 3x3
matrix_3x3 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Matriks 3x3:\n", matrix_3x3)
```

Pendefinisian Matriks dengan Fungsi Khusus

```
zeros_matrix = np.zeros((3, 3))
print("Matriks Nol (3x3):\n", zeros_matrix)

ones_matrix = np.ones((2, 4))
print("Matriks Elemen Satu (2x4):\n", ones_matrix)

identity_matrix = np.eye(4)
print("Matriks Identitas (4x4):\n", identity_matrix)

random_matrix = np.random.rand(3, 3)
print("Matriks Acak (3x3):\n", random_matrix)
```

Mengambil Elemen Matriks

Untuk mengakses elemen dalam matriks NumPy, indeks digunakan dengan format [baris, kolom]. Indeks dimulai dari 0.

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Mengakses elemen tertentu
print("Elemen baris ke-1, kolom ke-2:", A[0, 1])

# Mengakses seluruh elemen di baris tertentu
print("Baris ke-2:", A[1, :])

# Mengakses seluruh elemen di kolom tertentu
print("Kolom ke-3:", A[:, 2])

# Mengakses sub-matriks (bagian matriks)
print("Sub-matriks:", A[0:2, 1:3])
```

- A[0, 1]: Mengakses elemen di baris ke-0 dan kolom ke-1 (nilai = 2).
- A[1, :]: Mengambil seluruh elemen di baris ke-1 (hasil = [4, 5, 6]).
- A[:, 2]: Mengambil seluruh elemen di kolom ke-2 (hasil = [3, 6, 9]).
- A[0:2, 1:3]: Mengambil elemen dari baris ke-0 hingga baris ke-1, kolom ke-1 hingga kolom ke-2 (hasil = [[2, 3], [5, 6]]).

Operasi Elemen

Operasi elemen dalam NumPy memungkinkan perhitungan dilakukan pada setiap elemen matriks secara langsung. Matriks yang terlibat harus memiliki ukuran yang sama. Operasi yang dapat dilakukan meliputi:

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

# Perkalian elemen-per-elemen
C = A * B
print("Perkalian Elemen-per-Elemen:\n", C)

# Pembagian elemen-per-elemen
D = A / B
print("Pembagian Elemen-per-Elemen:\n", D)

# Penjumlahan dan Pengurangan elemen
E = A + B
F = A - B
print("Penjumlahan Elemen:\n", E)
print("Pengurangan Elemen:\n", F)

# Pangkat elemen-per-elemen
G = A ** 2
print("Pangkat Elemen:\n", G)
```

Operasi Aljabar Matriks

Operasi aljabar matriks dalam NumPy mengikuti aturan matematika untuk matriks. Berikut adalah beberapa contoh:

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

# Perkalian Matriks
C = np.dot(A, B)
print("Perkalian Matriks:\n", C)

# Invers Matriks
inv_A = np.linalg.inv(A)
print("Invers Matriks:\n", inv_A)

# Transpose Matriks
A_T = A.T
print("Transpose Matriks:\n", A_T)


# Determinan Matriks
```

```
det_A = np.linalg.det(A)
print("Determinan Matriks:\n", det_A)

# Eigenvalues & Eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(A)
print("Nilai Eigen:\n", eigenvalues)
print("Vektor Eigen:\n", eigenvectors)

# Rank Matriks
rank_A = np.linalg.matrix_rank(A)
print("Rank Matriks:\n", rank_A)
```

Persamaan Linier dalam Matriks

Jika terdapat persamaan linier dengan variabel x_1 dan x_2.

$$x_1 - 2x_2 = 32$$

$$12x_1 + 5x_2 = 7$$

Dalam matrik dapat ditulis sebagai

$$\begin{pmatrix} 1 & -2 \\ 12 & 5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 32 \\ 7 \end{pmatrix} \Leftrightarrow \mathbf{AX = B}$$

$X = A^{-1} B$; di mana A^{-1} ialah invers matriks A

```
import numpy as np

# Matriks koefisien
A = np.array([[1, -2], [12, 5]])

# Vektor konstanta
B = np.array([32, 7])

# Menghitung invers matriks
A_inv = np.linalg.inv(A)

# Menghitung solusi
X = np.dot(A_inv, B)
print("Solusi:\n", X)
```

Sehingga kita dapatkan solusi $x_1 = 6$ dan $x_2 = -13$. Atau dapat diselesaikan dengan menggunakan operator pembagian terbalik:

```
# Menggunakan pembagian balik
X = np.linalg.solve(A, B)
print("Solusi dengan np.linalg.solve:\n", X)
```

Pengabungan Matriks

Penggabungan matriks pada NumPy dilakukan untuk menyatukan dua atau lebih matriks menjadi satu matriks yang lebih besar. Hal ini dapat dilakukan secara horizontal atau vertikal, menggunakan fungsi **np.hstack** dan **np.vstack**.

1. Penggabungan Horizontal dengan **np.hstack**

   np.hstack digunakan untuk menggabungkan matriks sejajar kolom (dari kiri ke kanan). Fungsi ini memiliki syarat, yaitu semua matriks yang digabungkan harus memiliki jumlah baris yang sama.

```
import numpy as np

# Definisi matriks
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

# Penggabungan horizontal
H = np.hstack((A, B))
print("Hasil Penggabungan Horizontal:\n", H)
```

2. Penggabungan Vertikal dengan **np.vstack**

   np.vstack digunakan untuk menggabungkan matriks sejajar baris (dari atas ke bawah). Fungsi ini memiliki syarat, yaitu semua matriks yang digabungkan harus memiliki jumlah kolom yang sama.

```
import numpy as np

# Definisi matriks
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

# Penggabungan vertikal
V = np.vstack((A, B))
print("Hasil Penggabungan Vertikal:\n", V)
```

3. Menggabungkan Lebih dari Dua Matriks

   Baik **np.hstack** maupun **np.vstack** dapat digunakan untuk menggabungkan lebih dari dua matriks sekaligus.

```
import numpy as np
```

```
# Definisi matriks
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
C = np.array([[9, 10], [11, 12]])

# Penggabungan horizontal tiga matriks
H_multi = np.hstack((A, B, C))
print("Gabungan Horizontal Tiga Matriks:\n", H_multi)

# Penggabungan vertikal tiga matriks
V_multi = np.vstack((A, B, C))
print("Gabungan Vertikal Tiga Matriks:\n", V_multi)
```

Pembentukan Ulang Matriks

Terdapat beberapa perintah dalam NumPy yang dapat dipakai untuk mengubah bentuk atau dimensi sebuah matriks.

1. ***np.reshape***

   Fungsi np.reshape digunakan untuk mengubah dimensi matriks atau array. Matriks hasil reshaping harus memiliki jumlah elemen yang sama dengan matriks awal.

```
import numpy as np

# Matriks awal
A = np.array([1, 2, 3, 4, 5, 6])

# Mengubah bentuk menjadi matriks 2x3
reshaped_A = A.reshape(2, 3)
print("Matriks Reshape 2x3:\n", reshaped_A)

# Mengubah bentuk menjadi matriks 3x2
reshaped_A2 = A.reshape(3, 2)
print("Matriks Reshape 3x2:\n", reshaped_A2)
```

2. ***np.flatten***

   Fungsi np.flatten digunakan meratakan dimensi matriks menjadi satu dimensi.

```
# Matriks awal
B = np.array([[1, 2, 3], [4, 5, 6]])

flattened_B = B.flatten()
print("Matriks Diratakan (flatten):", flattened_B)
```

3. ***np.resize***

Fungsi resize() mengubah ukuran matriks, tetapi berbeda dari reshape() karena matriks dapat diubah menjadi ukuran baru dengan menambahkan elemen 0 jika diperlukan.

```
# Matriks awal
C = np.array([1, 2, 3, 4])

# Mengubah ukuran menjadi 3x3
resized_C = np.resize(C, (3, 3))
print("Matriks Resize 3x3:\n", resized_C)
```

4. Kesimpulan
    - Gunakan **reshape()** untuk mengubah dimensi matriks sesuai kebutuhan.
    - Gunakan **flatten()** untuk meratakan matriks menjadi array 1D.
    - Gunakan **resize()** untuk mengubah ukuran matriks tanpa memperhatikan jumlah elemen.

IV. TUGAS PRAKTIKUM

1. Buatlah matriks 3×3 dengan elemen berupa bilangan berurutan dari 1 hingga 9. Cetak hasilnya.
2. Definisikan dua matriks 2×2:

   $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$

   Hitung hasil penjumlahan dari kedua matriks tersebut.
3. Hitunglah hasil perkalian matriks berikut:

   $C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad D = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}$

   Cetak hasilnya dalam bentuk matriks.
4. Diberikan matriks 2×3 berikut:

   $E = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

   Lakukan transpose dan cetak hasilnya.
5. Selesaikan sistem persamaan linier berikut menggunakan matriks:

   $x + 2y = 10$

   $3x + 4y = 24$
6. Diberikan dua matriks:

   $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$

   Gabungkan kedua matriks:
   - Secara horizontal.
   - Secara vertikal.
7. Definisikan array 1D berikut:

   $X = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

   Ubah array tersebut menjadi matriks:
   - Matriks 2×3
   - Matriks 3×2