Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Лабораторна робота №2

з предмету «Моделювання систем»

Виконав студент 3-го курсу

Групи ІПС-31

Навка Гліб Олександрович

2021

Завдання

Матрицю X будемо інтерпретувати як двовимірне вхідне зображення, а матрицю Y – як вихідне зображення. Потрібно побудувати лінійний оператор перетворення вхідного сигналу X у вихідний сигнал Y на основі формули (2.9).

1. Вивчити означення псевдооберненої матриці і її основні властивості.

2. Створити програму, яка за заданими двома зображеннями знаходить лінійний оператор переходу між цими зображеннями. Основою для програми є формула (2.9), де V – довільна матриця (наприклад, нульова). Псевдообернену матрицю в (2.9) шукати двома методами: на основі формули Мура-Пенроуза (див. (2.3) або (2.4)) і на основі формули Гревіля. Правильність знаходження псавдооберненої матриці перевірити за допомогою теореми 2.1 про характеристичну властивість псевдооберненої матриці.

3. Вивести вихідне зображення і образ вхідного зображення при одержаному перетворенні. Зробити порівняння. Проаналізувати одержаний результат.

Теорія

Означення псевдооберненої матриці

Нехай задана матриця A розмірності $m \times n$. За означенням Мура - Пенроуза, псевдооберненою матрицею A^+ називається матриця розмірності $n \times m$ вигляду

$$A^+ = \lim_{\delta^2 \to 0} \left\{ \left(A^T A + \delta^2 E_n \right)^{-1} A^T \right\} = \lim_{\delta^2 \to 0} \left\{ A^T \left(A A^T + \delta^2 E_m \right)^{-1} \right\} \quad (2.1)$$

Тут E_n – одинична матриця розмірності $n \times n$.

Властивості псевдооберненої матриці

1. Якщо матриця A – невироджена, то $A^+ = A^{-1}$.
2. $A^+ = \left(A^T A\right)^+ A^T$, $A^+ = A^T \left(A A^T\right)^+$.
3. Якщо матриця $A^T A$ – невироджена, то

$$A^+ = \left(A^T A\right)^{-1} A^T.$$

Якщо матриця AA^T – невироджена, то

$$A^+ = A^T \left(A A^T\right)^{-1}.$$

4. Якщо $a \in \mathbb{R}^n$ – вектор розмірності n, $a \neq 0$, то з означення Мура-Пенроуза (2.1) випливає, що

$$\left(a^T\right)^+ = \frac{a}{a^T a}, \quad a^+ = \frac{a^T}{a^T a}.$$

Якщо $a = 0$, то (2.1) випливає $a^+ = 0$.

5. $(A^+)^+ = A$.
6. $(A^T)^+ = (A^+)^T$

Теорема 2.1 (характеристична властивість псевдооберненої матриці). *Матриця A^+ розмірності $n \times m$ є псевдооберненою матрицею до матриці A розмірності $m \times n$ тоді і тільки тоді, якщо виконуються такі умови:*

- $AA^+A = A$;
- $A^+AA^+ = A^+$;
- AA^+ – *симетрична матриця розмірності* $m \times m$;
- A^+A – *симетрична матриця розмірності* $n \times n$.

Формула Гревіля

Якщо для матриці A відома псевдообернена (обернена) матриця A^+, то для розширеної матриці $\begin{pmatrix} A \\ a^T \end{pmatrix}$ справедлива формула

$$\begin{pmatrix} A \\ a^T \end{pmatrix}^+ = \begin{cases} \left(A^+ - \frac{Z(A)aa^TA^+}{a^TZ(A)a} \vdots \frac{Z(A)a}{a^TZ(A)a}\right), \, if\, a^TZ(A)a > 0 \\ \left(A^+ - \frac{R(A)aa^TA^+}{1+a^TR(A)a} \vdots \frac{R(A)a}{1+a^TR(A)a}\right), \, if\, a^TZ(A)a = 0 \end{cases} , \tag{2.2}$$

де $Z(A) = E - A^+A$ – проектор на ядро матриці A, $R(A) = A^+\left(A^+\right)^T$.

Алгоритми знаходження псевдооберненої матриці

Для знаходження псевдооберненої матриці реалізуються такі алгоритми:

I. алгоритм, заснований на означенні Мура-Пенроуза. З означення Мура-Пенроуза (2.1) випливає, що для наближеного визначення псевдо-оберненої матриці можна застосовувати одну з формул

$$A^+ \approx \left(A^TA + \delta_0^2 E_n\right)^{-1} A^T, \tag{2.3}$$

$$A^+ \approx A^T \left(AA^T + \delta_0^2 E_m\right)^{-1}. \tag{2.4}$$

Тут $\delta_0^2 > 0$ –число, яке підбирається експерементально. Одна з можливих схем є такою:

1. Задається початкове значення $\delta = \delta_0$;
2. Розраховується початкове наближення $A_0^+ = A^T(AA^T + \delta_0^2 E_m)^{-1}$;
3. На кроці k нове значення $\delta_k = \dfrac{\delta_{k-1}}{2}$;
4. Наближення $A_k^+ = A^T(AA^T + \delta_k^2 E_m)^{-1}$;
5. Якщо $\| A_k^+ - A_{k-1}^+ \| < \varepsilon$, то зупинитись з $A^+ = A_k^+$, інакше $k := k+1$ і продовжити з пункту 3.

II. алгоритм на основі формули Гревіля (2.2). Цей алгоритм є рекурентним. Представляємо матрицю A у вигляді

$$A = \begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_n^T \end{pmatrix}.$$

Для першого кроку алгоритму $\left(a_1^T\right)^+ = \frac{a_1}{a_1^Ta_1}$, при $a_1 \neq 0$; $\left(a_1^T\right)^+ = 0$, якщо $a_1 = 0$. На наступному кроці додаємо до матриці другий рядок і шукаємо псевдообернену матрицю згідно формули Гревіля. Потім знову додаємо рядок і т.д. поки не вичерпаються всі рядки матриці A.

III. алгоритм, що базується на сингулярному розкладі матриці (теорема 2.2).

<u>Розв’язок</u>

Варіант **№5**

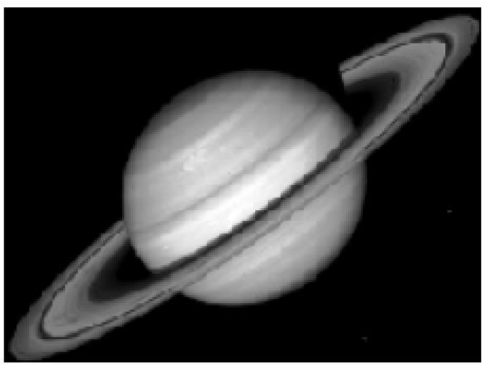
Файли **x1.bmp** та **y5.bmp**

Зчитав зображення з файлів

```
x_image, y_image = image.imread('x1.bmp'), image.imread('y5.bmp')
```

Вивід даних зображень на екран

X



Y

Реалізація методу Мура-Пенроуза

```
def moore_penrose_method(matrix, sigma0, eps=1e-5):
    # Step 1 (sigma0 init by user)

    matrix = np.array(matrix, dtype=float)
    e = np.eye(matrix.shape[0])

    # Step 2
    plus_matrix = matrix.T @ np.linalg.inv(matrix @ matrix.T + sigma0 * e)

    while True:
        # Step 3
        sigma_k = sigma0 / 2

        previous = plus_matrix

        # Step 4
        plus_matrix = matrix.T @ np.linalg.inv(matrix @ matrix.T + sigma_k * e)

        # Step 5
        if np.linalg.norm(plus_matrix - previous) < eps:
            return plus_matrix
```

Реалізація методу на основі формули Гревіля

```python
def greville_method(matrix):
    matrix = np.array(matrix, dtype=float)

    # Get first row
    a = matrix[0:1]

    if np.count_nonzero(a[0]) == 0:
        result = np.zeros_like(a.T)
    else:
        result = a.T / a @ a.T

    # Greville formula
    for i in range(1, matrix.shape[0]):
        z_a = np.eye(result.shape[0]) - result @ matrix[:i]
        r_a = result @ result.T
        a = matrix[i:i + 1]

        dot_product = (a @ z_a) @ a.T

        if np.count_nonzero(dot_product) == 0:
            part_a = (r_a @ a.T) / (1 + (a @ r_a) @ a.T)
        else:
            part_a = (z_a @ a.T) / dot_product

        result = np.concatenate((result - part_a @ (a @ result), part_a), axis=1)

    return result
```

Реалізація Теореми 2.1 (характеристична властивість матриці)

```python
def pseudoinverse_matrix_check(x_plus, x):
    result = True

    result = result and ((x @ x_plus) @ x).all() == x.all()
    result = result and ((x_plus @ x) @ x_plus).all() == x_plus.all()
    result = result and np.allclose(x @ x_plus, (x @ x_plus).T)
    result = result and np.allclose(x_plus @ x, (x_plus @ x).T)

    return result
```

Отримав матрицю за допомогою метода Мура-Пенроуза

```python
moore_penrose_matrix = moore_penrose_method(x_image, 1)
```

Отримав матрицю за допомогою метода на основі формули Гревіля

```python
greville_matrix = greville_method(x_image)
```

Перевірив отриману матрицю moore_penrose_matrix ***за допомогою метода*** pseudoinverse_matrix_check

```
moore_penrose_status = 'Ok' if pseudoinverse_matrix_check(moore_penrose_matrix, x_image) else 'Bad'
print(f'Status Moore-Penrose method: {moore_penrose_status}')
```

Перевірив отриману матрицю greville_matrix ***за допомогою метода*** pseudoinverse_matrix_check

```
greville_status = 'Ok' if pseudoinverse_matrix_check(greville_matrix, x_image) else 'Bad'
print(f'Status Greville method: {greville_status}')
```

Отриманий результат у консолі

```
Status Moore-Penrose method: Ok
Status Greville method: Ok
```

Знайшов операторі відповідних матриць

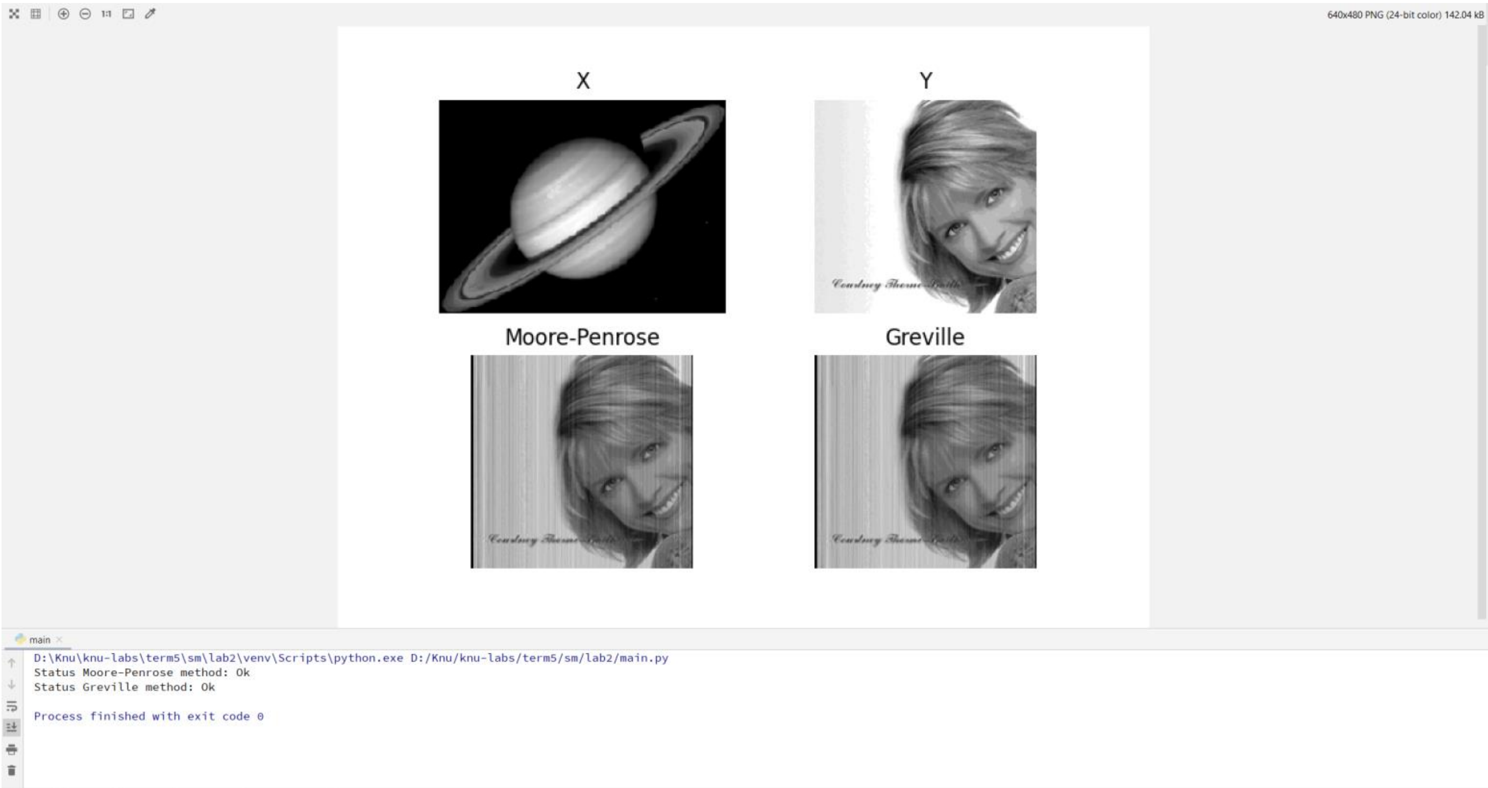
```
moore_penrose_operator = y_image @ moore_penrose_matrix
greville_operator = y_image @ greville_matrix
```

Вивід отриманого вихідного зображення за допомогою операторів

```
ax = fig.add_subplot(2, 2, 3)
ax.set_title('Moore-Penrose')
plt.imshow(moore_penrose_operator @ x_image, cmap='gray')
plt.axis('off')

ax = fig.add_subplot(2, 2, 4)
ax.set_title('Greville')
plt.imshow(greville_operator @ x_image, cmap='gray')
plt.axis('off')
```

Загальний результат роботи програми



Висновок

Як бачимо, дане зображення та зображень, які отримані за допомогою методу Мура-Пенроуза та методу Гревіля майже збігаються. Це означає, що отримані операторі є правильні