Київський національний університет імені Тараса Шевченка
Факультет комп’ютерних наук та кібернетики

Лабораторна робота №2
з Моделювання складних систем
“Метод побудови лінійної моделі з допомогою псевдообернених операторів”
Варіант №12

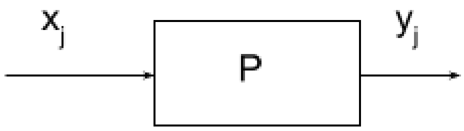
Виконав студент групи ІПС-31
Тесленко Назар Олександрович

Київ - 2025

Мета роботи:

Побудова лінійної моделі з допомогою псевдообернених операторів.

Будемо вважати, що на вхід системи перетворення, математична модель якої невідома, поступають послідовно дані у вигляді $m-1$ вимірних векторів $\mathbf{x}_j$. На виході системи спостерігається сигнал у вигляді вектора $\mathbf{y}_j$ розмірності p.



Постановка задачі:

Для послідовності вхідних сигналів $\mathbf{x}_j$, $j = 1, 2, \ldots, n$ та вихідних сигналів $\mathbf{y}_j$, $j = 1, 2, \ldots, n$ знайти оператор P перетворення вхідного сигналу у вихідний.

Будемо шукати математичну модель оператора об'єкту в класі лінійних операторів

$$\mathbf{A}\begin{pmatrix} \mathbf{x}_j \\ 1 \end{pmatrix} = \mathbf{y}_j, \quad j = 1, 2, \ldots, n. \tag{1}$$

Невідома матриця $\mathbf{A}$ математичної моделі об'єкту розмірності $p \times n$.
Систему (1) запишемо у матричній формі

$$\mathbf{A}\begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ 1 & 1 & \cdots & 1 \end{pmatrix} = (\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n),$$

або

$$\mathbf{AX} = \mathbf{Y}, \tag{2}$$

де $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ 1 & 1 & \cdots & 1 \end{pmatrix}$ – матриця вхідних сигналів розмірності $m \times n$, $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n)$ – матриця вихідних сигналів розмірності $p \times n$.

Матрицю $\mathbf{X}$ будемо інтерпретувати як двовимірне вхідне зображення, а матрицю $\mathbf{Y}$ вихідне зображення.
Тоді

$$\mathbf{A} = \mathbf{YX}^{+} + \mathbf{VZ}^{T}(\mathbf{X}^{T}),$$

де матриця

$$\mathbf{V} = \begin{pmatrix} \mathbf{v}_{(1)}^T \\ \mathbf{v}_{(2)}^T \\ \vdots \\ \mathbf{v}_{(p)}^T \end{pmatrix},$$

розмірності $p \times m$, $\mathbf{Z}(\mathbf{X}^T) = \mathbf{I}_m - \mathbf{X}\mathbf{X}^+$.

Формула Мура - Пенроуза для знаходження оберненої (псевдооберненої) матриці:

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

матриця A розмірності $m \times n$.

Формула Гревіля для псевдообернення матриці:

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

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

де $Z(A) = E - A^+ A,\ R(A) = A^+ \left(A^+ \right)^T$.

Для першого кроку алгоритму $\left(a_1^T \right)^+ = \dfrac{a_1}{a_1^T a_1}$, де $A = \begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_n^T \end{pmatrix}$.

12) Вхідний сигнал – x3.bmp, вихідний сигнал – y3.bmp

Хід роботи

Мова реалізації: Python

Файл: **main.py:** головний файл, для викликів методів

Бібліотеки: NumPy, cv2, os

Для подальшої роботи з алгоритмами зчитуємо надані вхідні та вихідні зображення та приводимо їх до матричного формату типу float. Для цього реалізована функція read_img(), яка повертає матриці зчитаних зображень.

```python
def main():
    #Firstly init and fill X, Y matrix from imgs
    X,Y= read_img()

    save_image(Y,"Y_orig")
```

Також реалізована функція save_img(), яка перетворює матрицю у зображення та зберігає його у відповідну директорію. Після ініціалізації вхідних та вихідних даних (матриці X та Y) виконується збереження оригінального зображення вихідних даних.

```python
def save_image(matrix, filename="output.bmp"):
    #make folder if not exists
    dir_res= "results"
    if not os.path.exists(dir_res):
        os.makedirs(dir_res)

    if not filename.endswith(".bmp"):
        filename=filename + ".bmp"
    save_path = os.path.join(dir_res,filename)

    #Normalizing matrix to [0,1] values
    normalized = (matrix - matrix.min()) / (matrix.max() - matrix.min())
    # Then scale to 0-255 range(as unit8 values)
    img = (normalized * 255).astype('uint8')
    # Save the img
    cv2.imwrite(save_path, img)

    # Display the image
    cv2.imshow("Result image", img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    return img
```

Після отримання всіх необхідних даних починаємо виконання методів для знаходження оператора за допомогою псевдооберненої матриці:

- Метод Мура-Пенроуза

```
#Moore-Penrose method
print("#"*20)
print("MOORE-PENROSE")
print("#"*20)
A_MP=find_operator(X,Y,pim.pim_MoorePenrose, eps=1e-6)
if A_MP is not None:
    Y_MP=applyOperator(X,A_MP)
    save_image(Y_MP,"MP_result")
else:
        print("Failed to compute Moore-Penrose operator")
```

- Метод Гревілля

```
print("#"*20)
print("GREVILLES METHOD")
print("#"*20)
A_MG=find_operator(X,Y, pim.pim_Grevilles, eps=1e-6)
if A_MG is not None:
    Y_MG=applyOperator(X,A_MG)
    save_image(Y_MG, "Greville's method")
else:
    print("Failed to compute Greville operator")
```

- Для обчислення лінійного оператора за допомогою псевдообернених матриць реалізована функція find_operator(), яка приймає такі параметри:
- X – матриця вхідних даних
- Y – матриця вихідних даних
- pim_func – функція для обчислення псевдооберненої матриці
- V – додаткова матриця (для врахування компонентів, що лежать в ортогональному доповненні простору X)

- eps – точність обчислень (за замовчуванням 1e-6)
- delta – параметр регуляризації (за замовчуванням 1000)

Функція повертає матрицю оператора A, що описує лінійне перетворення між X та Y.

```python
def find_operator(X,Y,pim_func,V=None,eps=1e-6, delta=1000):
    if V is None:
        V = np.zeros((Y.shape[0], X.shape[0]))
    print("\nFinding operator...\n")
    print("Finding pseudo-inveresed X")
    X_pi=pim_func(X,eps,delta)

    if not (pim.isPseudoInversed(X,X_pi)): return None

    print("Finding A=YX+ + VZ^T(X^T)")
    #General formula: A=YX+ + VZ^T(X^T)
    #Find Z(X^T)=I_m-XX+
    Z=np.identity(X.shape[0])-X@X_pi
    YX_pi= Y@X_pi
    VZ=V@Z.T
    A=YX_pi+VZ
    return A
```

Функція applyOperator() застосовує знайдений лінійний оператор A до вхідної матриці X і отримує нову вихідну матрицю Y_:

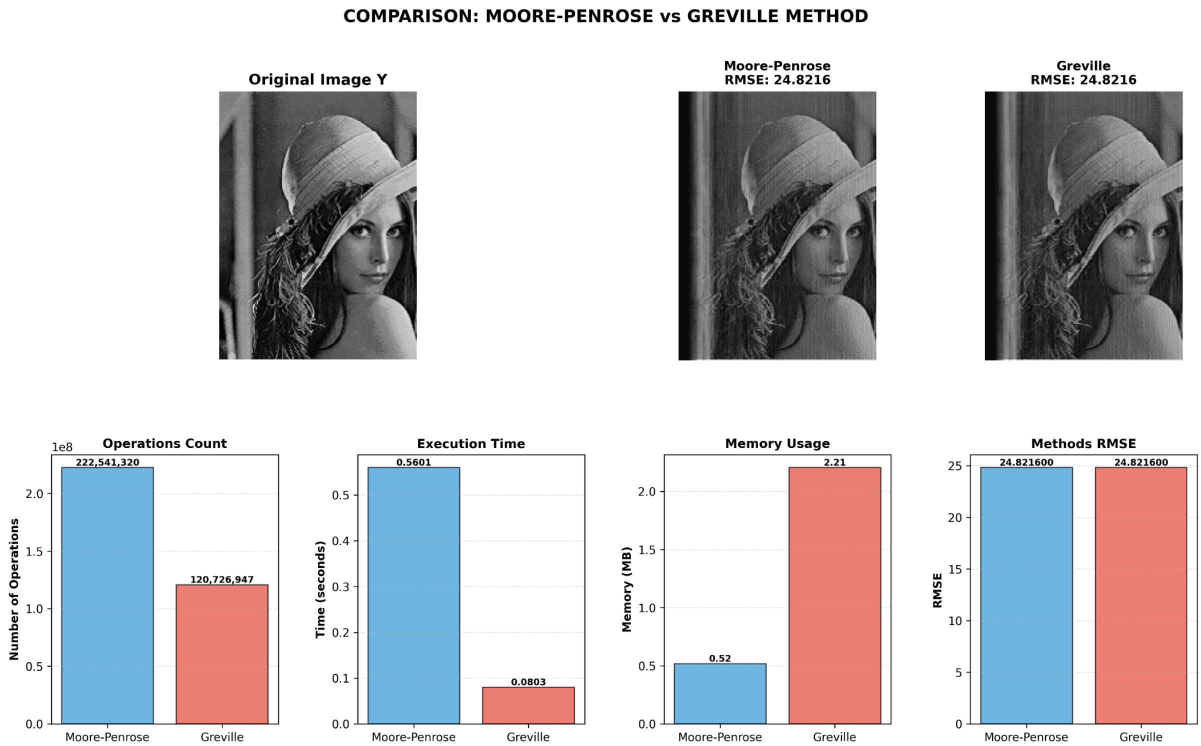
```python
def applyOperator(X,A):
    return A@X
```

Файл: **pim_methods.py -** реалізація методів знаходження псевдооберненої матриці, та відповідної перевірки за допомгою її властивостей

Функції:

```python
def isPseudoInversed(A,A_pi, rtol=1e-3, atol=1e-1) -> bool:
def pim_MoorePenrose(A,eps=1e-6, delta=100):
def pim_Grevilles(A, eps, delta=None):
```

Порівняння:



На діаграмах порівняно два методи — Мура–Пенроуза та Гревіля.
Аналіз результатів показує, що метод Гревіля виконується значно швидше (приблизно у 7 разів) і потребує майже вдвічі менше арифметичних операцій.
Водночас метод Мура–Пенроуза демонструє більш економне використання пам’яті (0.52 МБ проти 2.21 МБ у методу Гревіля).
Обидва методи забезпечують однакову точність відтворення, про що свідчить рівне значення RMSE = 24.8216 (Root Mean Square Error)