Моделювання систем. Лабораторна робота №2

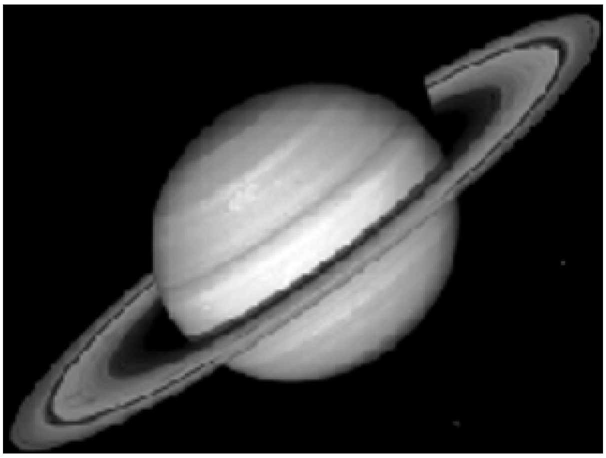
Краснощок Іван, ІПС-31

```
import numpy as np
from PIL import Image
```

Вхідний сигнал - x1.bmp, вихідний сигнал - y6.bmp (згідно з варіантом №6)

```
def scale_image(image, scale_factor=1.6):
    return image.resize((int(image.width * scale_factor), int(image.height
* scale_factor)))
```

```
x1_bmp = Image.open('x1.bmp')
scale_image(x1_bmp)
```



```
y6_bmp = Image.open('y6.bmp')
scale_image(y6_bmp)
```

Модель

$Dx_j + b = y_j$, де x_j – вхідний сигнал, y_j – вихідний сигнал

$$\begin{pmatrix} D \\ b^T \end{pmatrix} \begin{pmatrix} x_j \\ 1 \end{pmatrix} = y_j; \quad A \begin{pmatrix} x_1 & \dots & x_n \\ 1 & \dots & 1 \end{pmatrix} = (y_1 \dots y_n)$$

$$AX = Y$$

```
X = np.vstack((np.array(x1_bmp), np.ones(x1_bmp.width)))
Y = np.array(y6_bmp)
```

Формула Гревіля

A, A^+ — відомі матриці

$$\begin{pmatrix} A \\ a^T \end{pmatrix}^+ = \begin{cases} A^+ - \dfrac{Z(A)\,a a^T A^+}{a^T Z(A)\,a} \ \vdots\ \dfrac{Z(A)\,a}{a^T Z(A)\,a}, & \text{якщо } a^T Z(A)\,a > 0 \\[2ex] A^+ - \dfrac{R(A)\,a a^T A^+}{1 + a^T R(A)\,a} \ \vdots\ \dfrac{R(A)\,a}{1 + a^T R(A)\,a}, & \text{якщо } a^T Z(A)\,a = 0 \end{cases}$$

$$Z(A) = E - A^+A, \quad R(A) = A^+(A^+)^T$$

```python
def Z(A, A_pinv):
    return np.eye(A_pinv.shape[0]) - A_pinv @ A


def R(A_pinv):
    return A_pinv @ A_pinv.T


def pseudoinverse_Greville(A, eps=1e-9):
    a = A[0].reshape(-1, 1)
    A_pinv = a if np.abs((a.T @ a).item()) <= eps else a / (a.T @ a).item()
    A_pinv = A_pinv.reshape(-1, 1)

    m, _ = A.shape
    for i in range(1, m):
        Z_A = Z(A[:i], A_pinv)
        a = A[i].reshape(-1, 1)

        formula_denominator = (a.T @ Z_A @ a).item()

        if np.abs(formula_denominator) > eps:
            F_A = Z_A
        else:
            F_A = R(A_pinv)
            formula_denominator = 1. + a.T @ F_A @ a

        A_pinv = np.hstack((
            (A_pinv - F_A @ a @ a.T @ A_pinv / formula_denominator),
            (F_A @ a / formula_denominator).reshape(-1, 1)
        ))

    return A_pinv
```

Формула Мура-Пенроуза

A - матриця $m \times n$

$$A^+ = \lim_{\delta^2 \to 0} (A^T A + \delta^2 E_n)^{-1} A^T \quad \text{або}$$

$$A^+ = \lim_{\delta^2 \to 0} A^T (A A^T + \delta^2 E_m)^{-1}$$

```
def pseudoinverse_Moore_Penrose(A, eps=1e-9, delta=10):
    m, n = A.shape
    moore_penrose_formula = (
        (lambda A, delta: np.linalg.inv(A.T @ A + delta * delta *
np.eye(n)) @ A.T)
        if m > n else
        (lambda A, delta: A.T @ np.linalg.inv(A @ A.T + delta * delta *
np.eye(m)))
    )

    A_pinv = moore_penrose_formula(A, delta)

    while True:
        delta /= 2
        A_pinv_prev, A_pinv = A_pinv, moore_penrose_formula(A, delta)

        if np.linalg.norm(A_pinv - A_pinv_prev) < eps:
            break

    return A_pinv
```

```
def get_output_image(pseudoinverse_func):
    A = Y @ pseudoinverse_func(X)
    return Image.fromarray((A @ X).astype(np.uint8), mode='L')
```

Вихідне зображення при використанні алгоритму, що базується на формулі Гревіля

```
scale_image(get_output_image(pseudoinverse_Greville))
```

Вихідне зображення при використанні алгоритму, що базується на формулі Мура-Пенроуза

```
scale_image(get_output_image(pseudoinverse_Moore_Penrose))
```