Моделювання систем. Лабораторна робота №3

Краснощок Іван, ІПС-31

```
import numpy as np
import pandas as pd
```

Завантажуємо дані з файлу y6.txt (за номером у групі):

```
y_observed = np.loadtxt('y6.txt').T
n, p = y_observed.shape
y_observed = y_observed.reshape((n, p, 1))

T = 50.
delta_t = T / (n - 1)
```

Початкове наближення оцінюваних параметрів c_1, c_2, m_2:

```
beta = np.array([0.1, 0.15, 19])
```

Модель:

$$\frac{dy}{dt} = Ay$$

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{(c_2+c_1)}{m_1} & 0 & \frac{c_2}{m_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{c_2}{m_2} & 0 & -\frac{(c_2+c_3)}{m_2} & 0 & \frac{c_3}{m_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{c_3}{m_3} & 0 & -\frac{(c_4+c_3)}{m_3} & 0 \end{pmatrix}$$

```
def calculate_A(m, c):
    A = np.zeros((6, 6), dtype=np.float64)
    A[0, 1] = 1.
    A[1, 0] = -(c[0] + c[1]) / m[0]
    A[1, 2] = c[1] / m[0]
    A[2, 3] = 1.
    A[3, 0] = c[1] / m[1]
    A[3, 2] = -(c[1] + c[2]) / m[1]
    A[3, 4] = c[2] / m[1]
    A[4, 5] = 1.
    A[5, 2] = c[2] / m[2]
    A[5, 4] = -(c[3] + c[2]) / m[2]

    return A


def create_dy_dt(A):
    def dy_dt(y, t):
        return A @ y

    return dy_dt
```

Матрицю чутливості визначаємо з наступної системи:

$$\frac{dU(t)}{dt} = \frac{\partial (Ay)}{\partial y^T} U(t) + \frac{\partial (Ay)}{\partial \beta^T}, \quad U(t_0) = 0, \quad \beta = \beta_0, \quad \frac{\partial (Ay)}{\partial y^T} = A$$

$$Ay = F = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix}; \quad \frac{\partial F}{\partial \beta} = \begin{pmatrix} \frac{\partial f_1}{\partial c_1} & \frac{\partial f_1}{\partial c_2} & \frac{\partial f_1}{\partial m_2} \\ \frac{\partial f_2}{\partial c_1} & \frac{\partial f_2}{\partial c_2} & \frac{\partial f_2}{\partial m_2} \\ \frac{\partial f_3}{\partial c_1} & \frac{\partial f_3}{\partial c_2} & \frac{\partial f_3}{\partial m_2} \\ \frac{\partial f_4}{\partial c_1} & \frac{\partial f_4}{\partial c_2} & \frac{\partial f_4}{\partial m_2} \\ \frac{\partial f_5}{\partial c_1} & \frac{\partial f_5}{\partial c_2} & \frac{\partial f_5}{\partial m_2} \\ \frac{\partial f_6}{\partial c_1} & \frac{\partial f_6}{\partial c_2} & \frac{\partial f_6}{\partial m_2} \end{pmatrix}$$

```
def calculate_dAy_dbeta(m, c, y):
    dA_dbeta_1 = np.zeros((6, 6))
    dA_dbeta_1[1, 0] = -1. / m[0]
```

```
    dA_dbeta_2 = np.zeros((6, 6))
    dA_dbeta_2[1, 0] = -1. / m[0]
    dA_dbeta_2[1, 2] = 1. / m[0]
    dA_dbeta_2[3, 0] = 1. / m[1]
    dA_dbeta_2[3, 2] = -1. / m[1]

    dA_dbeta_3 = np.zeros((6, 6))
    dA_dbeta_3[3, 0] = -c[1] / (m[1] * m[1])
    dA_dbeta_3[3, 2] = (c[1] + c[2]) / (m[1] * m[1])
    dA_dbeta_3[3, 4] = -c[2] / (m[1] * m[1])

    return np.hstack((dA_dbeta_1 @ y, dA_dbeta_2 @ y, dA_dbeta_3 @ y))


def create_dU_dt(A, m, c, y):
    dAy_dbeta = calculate_dAy_dbeta(m, c, y)

    def dU_dt(U, t):
        return A @ U + dAy_dbeta

    return dU_dt
```

Метод Рунге-Кутта (для чисельного ітегрування):

$$\frac{dy}{dt} = f(y, t), \quad y(t_0) = y_0$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = h f(y_n, t_n)$$

$$k_2 = h f\left(y_n + \frac{1}{2}k_1, t_n + \frac{1}{2}h\right)$$

$$k_3 = h f\left(y_n + \frac{1}{2}k_2, t_n + \frac{1}{2}h\right)$$

$$k_4 = h f(y_n + k_3, t_n + h)$$

```
def Runge_Kutta(h, f, y_j, t_j):
    k_1 = h * f(y_j, t_j)
    k_2 = h * f(y_j + 1 / 2 * k_1, t_j + 1 / 2 * h)
    k_3 = h * f(y_j + 1 / 2 * k_2, t_j + 1 / 2 * h)
    k_4 = h * f(y_j + k_3, t_j + h)

    return y_j + 1 / 6 * (k_1 + 2 * k_2 + 2 * k_3 + k_4)
```

Показник якості ідентифікації вектора невідомих параметрів β:

$$I(\beta) = \int_{t_0}^{t_k} (\bar{y}(t) - y(t))^T (\bar{y}(t) - y(t)) dt$$

$\beta - \beta_0$:

$$\Delta\beta = \left(\int_{t_0}^{t_k} U^T(t) U(t) dt\right)^{-1} \int_{t_0}^{t_k} U^T(t) (\bar{y}(t) - y(t)) dt$$

```
EPS = 1e-9
ABS_EPS = 1e-12

I_beta_list = []
I_beta = None

while True:
    m = np.array([12, beta[2], 18])
    c = np.array([beta[0], beta[1], 0.2, 0.12])

    A = calculate_A(m, c)

    integral_UT_U, integral_UT_y = np.zeros((3, 3)), np.zeros((3, 1))

    U_i = np.zeros((6, 3))
    y_i = y_observed[0]
    I_beta_prev, I_beta = I_beta, 0

    for i in range(1, n):
        t = i * delta_t

        dU_dt = create_dU_dt(A, m, c, y_i)
        U_i = Runge_Kutta(delta_t, dU_dt, U_i, t)

        dy_dt = create_dy_dt(A)
        y_i = Runge_Kutta(delta_t, dy_dt, y_i, t)

        integral_UT_U += U_i.T @ U_i * delta_t
        integral_UT_y += U_i.T @ (y_observed[i] - y_i) * delta_t

        I_beta += delta_t * ((y_observed[i] - y_i).T @ (y_observed[i] -
y_i)).item()

    delta_beta = (np.linalg.inv(integral_UT_U) @ integral_UT_y).reshape(-1)
    beta += delta_beta

    I_beta_list.append(I_beta)
```

```
    if I_beta <= EPS or (I_beta_prev is not None and np.abs(I_beta_prev - I_beta) <= ABS_EPS):
        break
```

Значення показника якості ідентифікації на кожній ітерації алгоритму:

```
I_beta_table = pd.DataFrame(I_beta_list, columns=['I(β)'])
I_beta_table.index += 1
I_beta_table.index.name = '№ ітерації'
I_beta_table
```

	I(β)
№ ітерації	
1	4.470590e+01
2	6.062049e+00
3	1.225347e-01
4	6.741278e-05
5	1.739590e-08
6	1.068910e-08
7	1.068860e-08

Отримані значення оцінюваних параметрів:

```
c_1, c_2, m_2 = tuple(beta.round(3))
print(f'c₁={c_1}, c₂={c_2}, m₂={m_2}')
```

```
c₁=0.14, c₂=0.3, m₂=28.0
```