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

Звіт
з лабораторної роботи №3
з моделювання складних систем
Варіант 11

Виконав:
Студент групи ІПС-32
Гончаренко Ілля Сергійович

Київ
2024

1. Мета роботи

Для математичної моделі коливання трьох мас m_1, m_2, m_3, які поєднані між собою пружинами з відповідними жорсткостями c_1, c_2, c_3, c_4, і відомої функції спостереження координат моделі $\overline{y}(t),\ t \in [t_0, t_k]$ потрібно оцінити частину невідомих параметрів моделі з використанням функції чутливості.

2. Постановка задачі

Математична модель коливання трьох мас описується наступною системою

$$\frac{dy}{dt} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{(c_2 + c_1)}{m_1} & 0 & \frac{c_2}{m_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{c_2}{m_2} & 0 & -\frac{(c_2 + c_3)}{m_2} & 0 & \frac{c_3}{m_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{c_3}{m_3} & 0 & -\frac{(c_4 + c_3)}{m_3} & 0 \end{pmatrix} y = Ay.$$

Показник якості ідентифікації параметрів невідомих параметрів β має вигляд

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

Якщо представити вектор невідомих параметрів $\beta = \beta_0 + \Delta\beta$, де β_0 – початкове наближення вектора параметрів,

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

Матриці чутливості $U(t)$ визначається з наступної матричної системи диференціальних рівнянь

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

$$U(t_0) = 0,\ \beta = \beta_0.$$

В даному випадку $\frac{\partial (Ay)}{\partial y^T} = A$.

Спостереження стану моделі проведені на інтервалі часу $t_0 = 0,\ t_k = 50,\ \Delta t = 0.2$.

Для чисельного інтегрування застосувати метод Рунге-Кутта 4-го порядку:

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

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

де

$$k_1 = hf(y_n, t_n).$$

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

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

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

$$t_{n+1} = t_n + h.$$

При чому показником якості для нашого шуканого вектора невідомих параметрів буде:

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

Наступне наближення вектору невідомих параметрів вираховується за формулою $\beta_{m+1} = \beta_m + \Delta\beta$, де $\Delta\beta$:

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

$U(t)$ – матриця чутливості, що виражається з системи диференційних рівнянь:

$$\frac{dU(t)}{dt} = A * U(t) + B,\ U(t_0) = 0$$

Умовами завершення роботи даного алгоритму є:

$I(\beta) < \varepsilon$ або $\Delta\beta < \varepsilon$, де я вирішив взяти ε як 10^{-6}

3. Хід роботи

Релізовано мовою Python

Для початку готуємо всі необхідні дані для програми:

```
input_data = read_file('y1.txt')
c1, c2, c3, c4, m1, m2, m3 = sp.symbols('c1 c2 c3 c4 m1 m2 m3')
to_approx = {c1: 0.1, m1: 11, m2: 23}
```

Далі виконуємо обчислення, виводячи на кожному кроці отримані проміжні результати:

```python
def approximate(y_matr, params, beta_symbols, beta_values, eps, h=0.2):
    a_matrix = init_matr().subs(params)
    beta_vector = np.array([beta_values[beta_symbols[0]], beta_values[beta_symbols[1]], beta_values[beta_symbols[2]]])
    iteration = 0  # To count the iterations
    while True:
        a_complete = np.array((a_matrix.subs(beta_values)).tolist())
        u_matr = np.zeros((6, 3))
        quality_degree = 0
        integral_part_inverse = np.zeros((3, 3))
        integral_part_mult = np.zeros((1, 3))
        y_approximation = y_matr[0]
        for i in range(len(y_matr)):
            b_derivative_matr = get_derivative(a_matrix * sp.Matrix(y_approximation), beta_symbols, beta_values)

            integral_part_inverse = (integral_part_inverse + np.dot(u_matr.T, u_matr)).astype('float64')

            integral_part_mult = (integral_part_mult + np.dot(u_matr.T, y_matr[i] - y_approximation)).astype('float64')

            quality_degree = quality_degree + np.dot((y_matr[i] - y_approximation).T, y_matr[i] - y_approximation)

            u_matr = get_u_matr(a_complete, b_derivative_matr, u_matr, h)
            y_approximation = get_y(a_complete, y_approximation, h)

        integral_part_inverse = integral_part_inverse * h
        integral_part_mult = integral_part_mult * h
        quality_degree = quality_degree * h

        delta_beta = np.dot(np.linalg.inv(integral_part_inverse), integral_part_mult.flatten())
        beta_vector = beta_vector + delta_beta

        beta_values = {
            beta_symbols[0]: beta_vector[0],
            beta_symbols[1]: beta_vector[1],
            beta_symbols[2]: beta_vector[2]
        }

        # Clean and formatted output
        iteration += 1
        print(f"--- Iteration {iteration} ---")
        print(f"Current approximated values:")
        for idx, beta_val in enumerate(beta_vector):
            print(f"  β{beta_symbols[idx]} = {beta_val:.6f}")

        print(f"Quality degree (delta) = {quality_degree:.6f}")

        if quality_degree < eps:
            print("\nConvergence achieved!")
            return beta_values
        else:
            print(f"Delta is greater than {eps:.6e} -> proceeding to next iteration\n")
```

Додаткові функції:

```python
def read_file(file_name):
    file = open(file_name, 'r')
    lines = file.readlines()
    input_data = []
    for line in lines:
        values = line.strip().split()
        row = []
        for value in values:
            row.append(float(value))
        input_data.append(row)
    return np.array(input_data).T

def get_derivative(y_vec, b_vec, b_values):
    derivs = []
    for y_i in y_vec:
        for b_i in b_vec:
            d = sp.diff(y_i, b_i)
            d = d.subs(b_values)
            derivs.append(d)

    cols_n = len(b_vec)
    der_matr = []
    for i in range(0, len(derivs), cols_n):
        der_matr.append(derivs[i:i + cols_n])
    return sp.Matrix(der_matr)


def get_u_matr(a_matr, b_matr, u_matr, h):
    b_arrayed = np.array(b_matr.tolist())
    k1 = h * (np.dot(a_matr, u_matr) + b_arrayed)
    k2 = h * (np.dot(a_matr, u_matr + k1 / 2) + b_arrayed)
    k3 = h * (np.dot(a_matr, u_matr + k2 / 2) + b_arrayed)
    k4 = h * (np.dot(a_matr, u_matr + k3) + b_arrayed)
    return u_matr + (k1 + 2 * k2 + 2 * k3 + k4) / 6


def get_y(a_matr, y_cur, h):
    k1 = h * np.dot(a_matr, y_cur)
    k2 = h * np.dot(a_matr, y_cur + k1 / 2)
    k3 = h * np.dot(a_matr, y_cur + k2 / 2)
    k4 = h * np.dot(a_matr, y_cur + k3)
    return y_cur + (k1 + 2 * k2 + 2 * k3 + k4) / 6

def init_matr():
    c1, c2, c3, c4, m1, m2, m3 = sp.symbols('c1 c2 c3 c4 m1 m2 m3')
    matr = [
        [0, 1, 0, 0, 0, 0],
        [-(c2 + c1) / m1, 0, c2 / m1, 0, 0, 0],
        [0, 0, 0, 1, 0, 0],
        [c2 / m2, 0, -(c2 + c3) / m2, 0, c3 / m2, 0],
        [0, 0, 0, 0, 0, 1],
        [0, 0, c3 / m3, 0, -(c4 + c3) / m3, 0]
    ]
    return sp.Matrix(matr)
```

Результати:

```
--- Iteration 1 ---
Current approximated values:
  βc1 = 0.140320
  βm1 = 11.926534
  βm2 = 27.296301
Quality degree (delta) = 3.320568
Delta is greater than 1.000000e-06 -> proceeding to next iteration

--- Iteration 2 ---
Current approximated values:
  βc1 = 0.139914
  βm1 = 11.997489
  βm2 = 27.989467
Quality degree (delta) = 0.071458
Delta is greater than 1.000000e-06 -> proceeding to next iteration

--- Iteration 3 ---
Current approximated values:
  βc1 = 0.139999
  βm1 = 11.999978
  βm2 = 27.999920
Quality degree (delta) = 0.000012
Delta is greater than 1.000000e-06 -> proceeding to next iteration

--- Iteration 4 ---
Current approximated values:
  βc1 = 0.140000
  βm1 = 11.999991
  βm2 = 28.000002
Quality degree (delta) = 0.000000

Convergence achieved!

--- Final Approximation ---
c1: 0.140000
m1: 11.999991
m2: 28.000002
```

Висновок:

У роботі було проведено ідентифікацію невідомих параметрів математичної моделі коливання трьох мас через пружини з використанням методу функції чутливості. Чисельне інтегрування здійснене методом Рунге-Кутта 4-го порядку. Ітераційний процес призвів до отримання значень параметрів: $\beta c1 = 0.140000$, $\beta m1 = 11.999991$, $\beta m2 = 28.000002$, що свідчить про досягнення збіжності алгоритму та точність результатів.