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

Звіт

лабораторної роботи №1-2

з моделювання складних систем

Варіант 9

Виконала:
студентка групи ІПС-31
Мельник Поліна Володимирівна

Київ 2024

<u>Постановка задачі:</u>

Нехай дана дискретна функція $\hat{y}(t_i)$, i = 1, 2, … N, що подана у вигляді значень у i-й момент часу.

<u>Завдання:</u>

Визначити медель в класі функцій

$$y(t) = a_1 t^3 + a_2 t^2 + a_3 t + \sum_{i=4}^{k} a_i \sin\left(2\pi f_{i-3} t\right) + a_{k+1}$$

для дискретної функції задані інтервал спостереження [0,T], T = 5, спостереження $\hat{y}(t_i)$ в дискретні моменти часу $t_i \in [0,T]$, i = 0,1,...N−1, $t_{i+1} - t_i = \Delta t$ = 0.01.

Для виконання використовуємо дискретне перетворення Фур'є для дискретної послідовності x(j), j=0,1,2, ..., N−1, яке визначається наступним способом:

$$c_x(k) = \frac{1}{N} \sum_{m=0}^{N-1} x(m) e^{-i2\pi km/N}.$$

Наступним кроком потрібно визначити суттєві внески частот за спостереження-ми. Отже, задані інтервал спостереження [0,T], T = 5, спостереження $\hat{y}(t_i)$ в дискретні моменти часу $t_i \in [0,T]$, i = 0,1,...N−1, $t_{i+1} - t_i = \Delta t$ =0.01. А спостереження записані послідовно у наданий файл f9.txt.

1. Знаходимо $\triangle f = \dfrac{1}{T}$
2. Для всіх k= 0,1,...,N−1 визначаємо модуль перетворення Фур'є $\left| c_{\hat{y}}(k) \right|$, за спостереженнями $\hat{y}(t_j)$, j = 0,1,...N−1.
3. Визначаємо локальні максимуми k_* модуля перетворення Фур'є $\left| c_{\hat{y}}(k) \right|$, k= 0,1,...,[N/2]−1.
4. Знаходимо частоти $f_* = k_* \triangle f$

Після знаходження частот із найбільшим вкладом, переходимо до визначення невідомих параметрів a_j, j = 1,2,...,k+ 1. Для їх визначення застосовуємо метод найменших квадратів. Отже, записуємо функціонал похибки

$$F(a_1, a_2, \ldots, a_{k+1}) = \frac{1}{2} \sum_{j=0}^{N-1} (a_1 t_j^3 + a_2 t_j^2 + a_3 t_j + \sum_{i=4}^{k} a_i \sin\left(2\pi f_{i-3} t_i\right) + a_{k+1} - \hat{y}(t_j))^2$$

Параметри a_j, j = 1,2,...,k+ 1 шукаємо з умови

$$F(a_1, a_2, \ldots, a_{k+1}) \to \min_{a_1, a_2, \ldots, a_{k+1}}$$

Для цього записуємо систему рівнянь

$$\frac{\delta F(a_1, a_2, \ldots, a_{k+1})}{\delta a_j} = 0,$$

j = 1,2,...,k+ 1. Ця система є системою лінійних алгебраїчних рівнянь. Розв'язавши одним з відомих методів, знаходмо a_j , j = 1,2,...,k+1.

<u>Реалізація з фрагментами коду:</u>

Завантажуємо дані з файлу f9.txt:

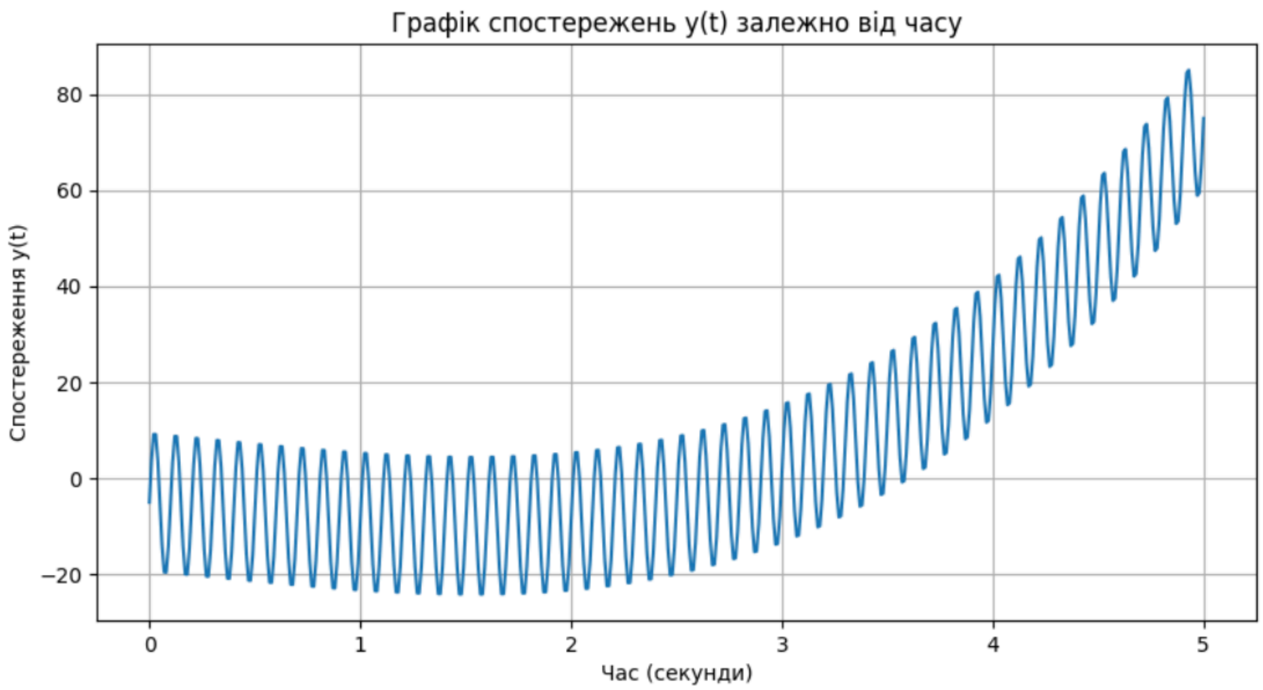
```
# Завантаження даних з файлу f9.txt
data = np.loadtxt('/Users/polyamelnik/Desktop/f9.txt')
```

Ініціалізуємо необхідні параметри:

```
# Параметри
dt = 0.01
T = 5
N = len(data)
dlt = T / N
t = np.arange(0, T+dt, dt)
```

Будуємо графік початкових спостережень в залежності від моменту часу:

```
# Графік спостережень залежно від часу
plt.figure(figsize=(10, 5))
plt.plot( *args: t, data)
plt.title('Графік спостережень y(t) залежно від часу')
plt.xlabel('Час (секунди)')
plt.ylabel('Спостереження y(t)')
plt.grid(True)
plt.show()
```



В результаті маємо поліноміальну функцію із частотними вкладеннями у вигляді коливань.

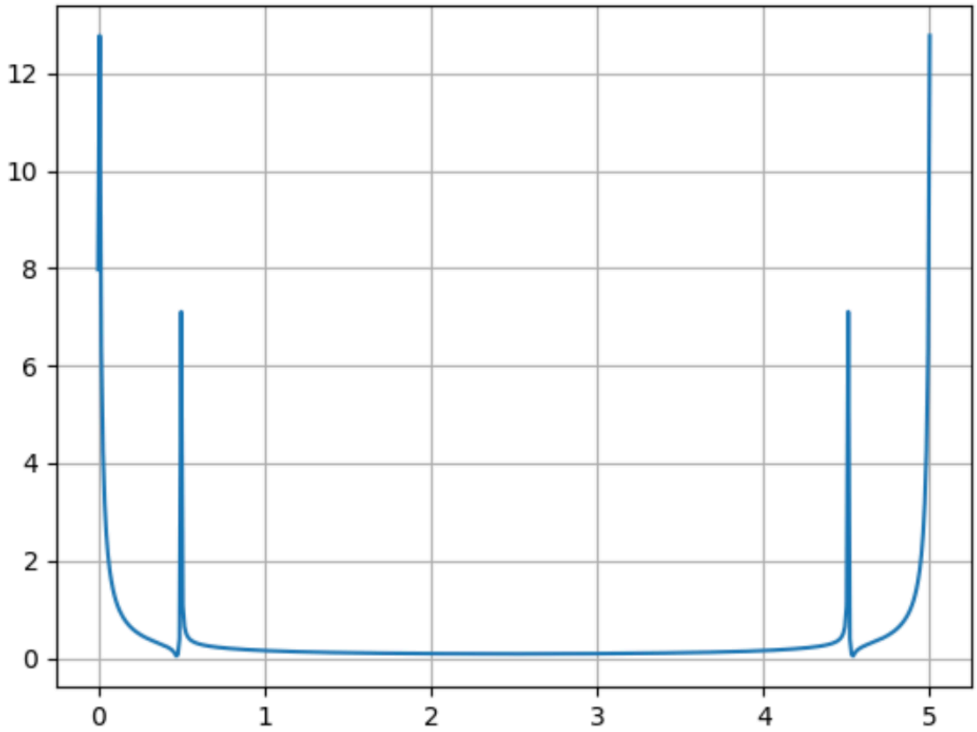
Далі виконаємо дискретне перетворення Фур'є і побудуємо графік його модуля. Реалізовуємо необхідну функцію самостійно для кращого розуміння виконання:

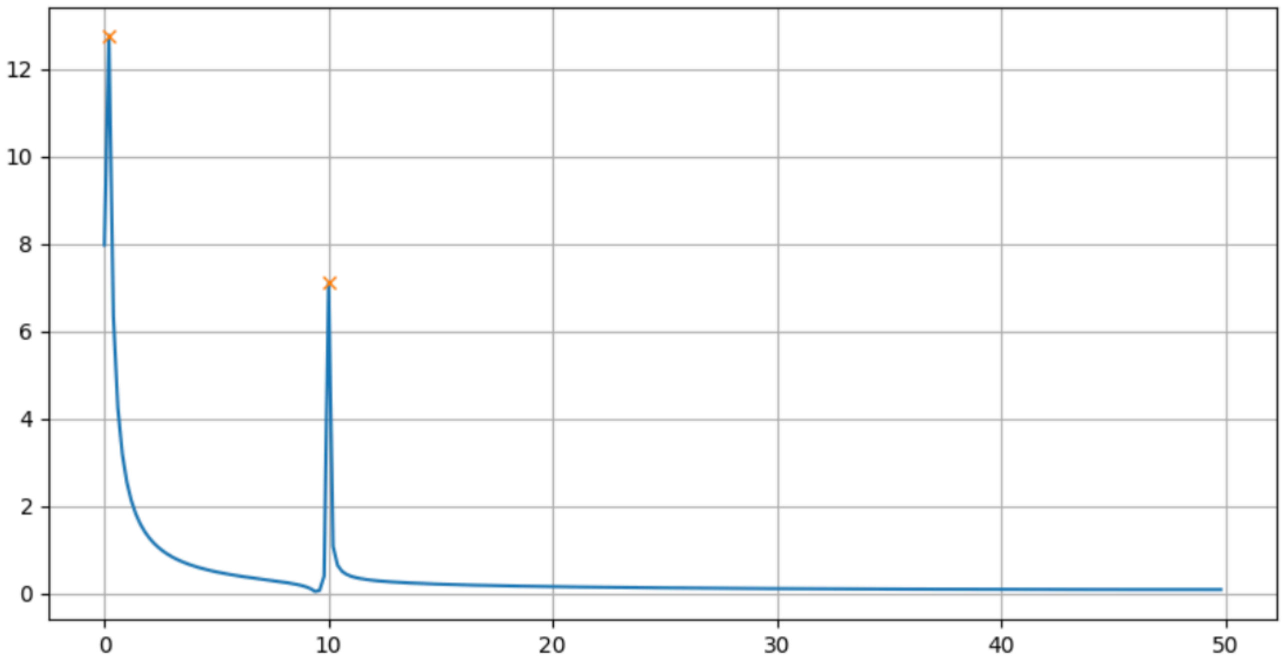
```
# Дискретне перетворення Фур'є
def df(x):  1 usage
    N = len(x)
    df_result = np.zeros(N, dtype=complex)
    for k in range(N):
        for m in range(N):
            df_result[k] += x[m] * np.exp(-2j * np.pi * k * m / N)
        df_result[k] /= N
    return df_result
```

Будуємо графіки(до та після відкидання симетричної частини):

```
# Графік модуля перетворення Фур'є(до/після виключення симетрії)
plt.figure()
plt.plot(t, np.abs(c_y))
plt.grid()
plt.title('Модуль перетворення Фур\'є')
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(fr[:half_N], mod_c_y[:half_N])
plt.plot(fr[peaks], mod_c_y[peaks], 'x')
plt.title('Модуль перетворення Фур\'є')
plt.grid(True)
plt.show()
```

Отримані графіки:





На останньому графіку можемо побачити, що найбільший вклад мають частоти 0.2 та 10(близьке до 0 значення ігноруємо, так як це поліноміальний вклад). Отже, маємо єдиний значущий вклад частоти 10Гц.

Звідси, періодична модель нашої моделі матиме вигляд: $\sin(2 \cdot 10\pi \cdot t)$.
А вигляд моделі матиме вигляд: $y(t) = a_1t^3 + a_2t^2 + a_3t + a_4\sin(2 \cdot 10\pi \cdot t) + a_5$.

Залишається знайти невідомі параметри a. Для цього використаємо метод найменших квадратів.

```
# Метод найменших квадратів
# Функція для обчислення синусів
def sinf(t, f):  10 usages
    return np.sin(2 * np.pi * f * t)
# Побудова матриці A
def matrix_A(t, filtered_fr):  1 usage
    N = len(t)
    A = np.zeros((5, 5))

    A[0, 0] = np.sum(t ** 6)
    A[0, 1] = np.sum(t ** 5)
    A[0, 2] = np.sum(t ** 4)
    A[0, 3] = np.sum(sinf(t, filtered_fr[0]) * t ** 3)
    A[0, 4] = np.sum(t ** 3)

    A[1, 0] = np.sum(t ** 5)
    A[1, 1] = np.sum(t ** 4)
    A[1, 2] = np.sum(t ** 3)
    A[1, 3] = np.sum(sinf(t, filtered_fr[0]) * t ** 2)
    A[1, 4] = np.sum(t ** 2)

    A[2, 0] = np.sum(t ** 4)
    A[2, 1] = np.sum(t ** 3)
    A[2, 2] = np.sum(t ** 2)
    A[2, 3] = np.sum(sinf(t, filtered_fr[0]) * t)
    A[2, 4] = np.sum(t)

    A[3, 0] = np.sum(sinf(t, filtered_fr[0]) * t ** 3)
    A[3, 1] = np.sum(sinf(t, filtered_fr[0]) * t ** 2)
    A[3, 2] = np.sum(sinf(t, filtered_fr[0]) * t)
    A[3, 3] = np.sum(sinf(t, filtered_fr[0]) ** 2)
    A[3, 4] = np.sum(N * sinf(t, filtered_fr[0]))
    A[4, 0] = np.sum(t ** 3)
    A[4, 1] = np.sum(t ** 2)
    A[4, 2] = np.sum(t)
    A[4, 3] = np.sum(N * sinf(t, filtered_fr[0]))
    A[4, 4] = N

    return A
# Побудова вектора c
def vector_c(t, y, filtered_fr):  1 usage
    c = np.array([
        np.sum(y * t ** 3),
        np.sum(y * t ** 2),
        np.sum(y * t),
        np.sum(y * sinf(t, filtered_fr[0])),
        np.sum(y)
    ])
    return c
# Визначення невідомих параметрів
def least_squares_solution(t, y, filtered_fr):  1 usage
    A = matrix_A(t, filtered_fr)
    c = vector_c(t, y, filtered_fr)
    x = np.linalg.inv(A).dot(c)
    return x

result = least_squares_solution(t, data, filtered_fr)
optparams = np.round(result).astype(int)
```

Результат:

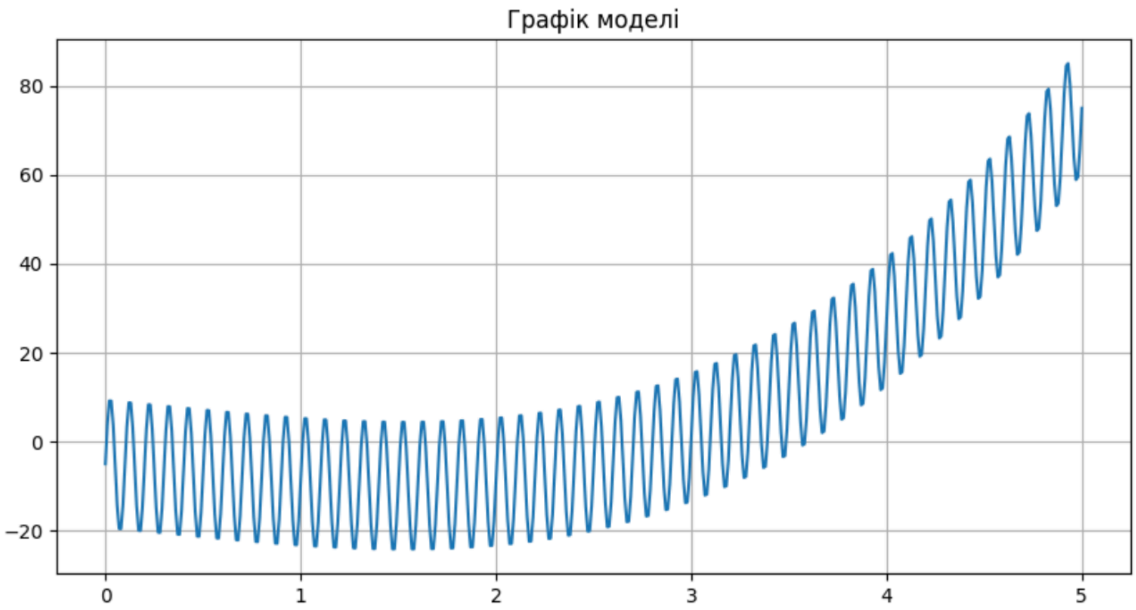
```
a: [ 1 -1 -4 15 -5]
```

Обчислюємо похибку та будуємо графік.

```python
# Графік моделі з знайденими параметрами
def model(params, t, f):  1 usage
    return (params[0] * t ** 3 + params[1] * t ** 2 + params[2] * t + params[3] * np.sin(2 * np.pi * f[0] * t)
            + params[4])

fitted_data = model(optparams, t, filtered_fr)

plt.figure(figsize=(10, 5))
plt.plot( *args: t, fitted_data)
plt.title('Графік моделі')
plt.xlabel('Час (секунди)')
plt.ylabel('y(t)')
plt.grid(True)
plt.show()

# Обчислення квадратичної похибки
mse = np.mean((data - fitted_data) ** 2)
print(f"Error_value: {mse}")
```

Результати:

```
Error_value: 6.702169721456103e-10
```



Апроксимуюча функція матиме наступний вигляд:

$$y(t) = t^3 - t^2 - 4t + 15\sin(2 \cdot 10\pi \cdot t) - 5$$

```
y(t) = 1 * t^3 + -1 * t^2 + -4 * t + 15 * sin(2π * 10.0 * t) + -5
```