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

Звіт
з лабораторної роботи №1
з моделювання складних систем
Варіант 4

Виконав:
Студент групи ІПС-31
Вербицький Артем Віталійович

Київ
2024

1. Постановка задачі

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

Потрібно визначити модель в класі функцій

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

для спостережуваної дискретної функції $\hat{y}(t_i),\ i = 1,2,\dots,N,\ t_{i+1} - t_i = \Delta t = 0.01$, інтервал спостереження $[0, T],\ T = 5$.

Для цього виконуємо дискретне перетворення Фур'є для дискретної послідовності $x(i),\ i = 0,1,2,\dots,N-1$

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

Далі потрібно визначити за модулем дискретного перетворення Фур'є (далі – модуль ДПФ) частоти із найбільшим вкладом. Для цього визначаємо момент, у який модуль ДПФ приймає найбільше значення, тобто є локальним екстремумом. Позначимо такі моменти через k^*. Щоб знайти саме частоти із найбільшим вкладом, які позначимо через f^*, потрібно виконати множення $f^* = k^*$ f $= k^*/T.$

Знайшовши частоти із найбільшим вкладом, можна приступати до безпосереднього визначення невідомих параметрів a_i, $i = k+1$. Для їх визначення застосовуємо метод найменших квадратів. Для цього записуємо функціонал похибки:

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

Параметри a_i , i = 1, 2, . . . , k + 1 шукаємо з умови:

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

Для цього записуємо систему рівнянь:

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

Ця система є системою лінійних алгебраїчних рівнянь. Розв'язавши цю систему одним з відомих методів, знаходимо a_i , i = 1, 2, . . . , k+1.

2. Хід роботи

Програмна реалізація виконана мовою Python.

Почнемо з ініціалізації всіх необхідних параметрів:

```
def main():
    observations = load_data('f4.txt')
    dt = 0.01
    t = np.arange(0, len(observations) * dt, dt)
```

Виконуємо дискретне перетворення Фур’є й знаходимо і виводимо локальні максимуми (суттєві частоти). При цьому ми виводимо лише першу половину спектру, адже у дискретному перетворенні Фур’є він є симетричний:

```
def custom_dft(signal):
    N = len(signal)
    dft = np.zeros(N, dtype=complex)
    for k in range(N):
        for n in range(N):
            dft[k] += signal[n] * np.exp(-2j * np.pi * n * k / N)
    return dft

def find_significant_frequencies(dft, dt):
    f_magnitude = np.abs(dft)
    frequencies = np.fft.fftfreq(len(dft), dt)
    peaks, _ = find_peaks(f_magnitude[:len(dft) // 2])
    peak_frequencies = frequencies[peaks]
    return peak_frequencies
```

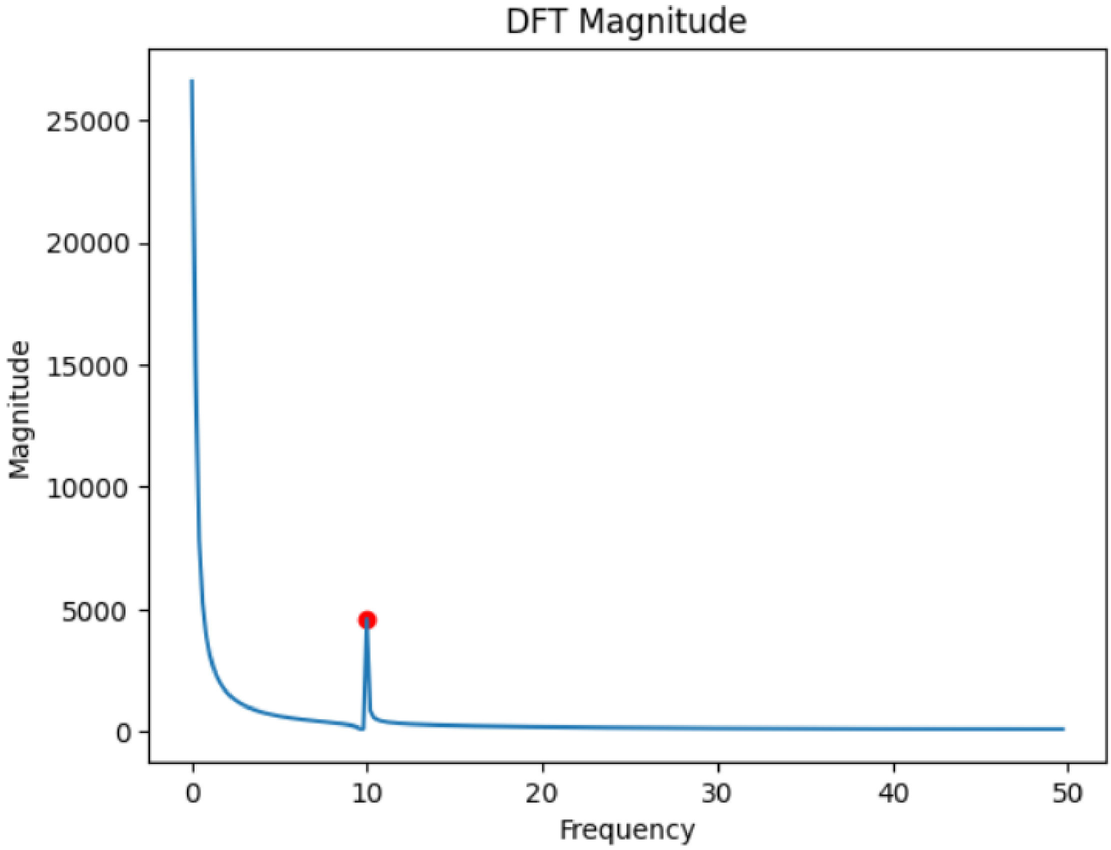
```
# 1. Discrete Fourier Transform
dft = custom_dft(observations)
peak_frequencies = find_significant_frequencies(dft, dt)
print("Significant frequencies:", peak_frequencies)
```

Будуємо графік:

```
# 2. Plot the DFT magnitude
f_magnitude = np.abs(dft)
frequencies = np.fft.fftfreq(len(dft), dt)
plt.figure()
plt.plot(frequencies[:len(dft) // 2], f_magnitude[:len(dft) // 2])
plt.scatter(peak_frequencies, f_magnitude[find_peaks(f_magnitude[:len(dft) // 2])[0]], color='red')
plt.xlabel('Frequency')
plt.ylabel('Magnitude')
plt.title('DFT Magnitude')
plt.show()
```

У висновку отримали:

```
Significant frequencies: [9.98003992]
```



Маємо єдиний значущий вклад частоти 9.98 Гц (близьке до 0 значення ігноруємо, так як це поліноміальний вклад).

Приступаємо до визначення невідомих параметрів методом найменших квадратів:

```
def fit_model(t, observations, peak_frequencies):
    def model(t, a1, a2, a3, *params):
        k = len(params) // 2
        y = a1 * t**3 + a2 * t**2 + a3 * t
        for i in range(k):
            fi = params[i]
            ai = params[k + i]
            y += ai * np.sin(2 * np.pi * fi * t - 3 * t)
        return y
```

```
# 3. Least-squares fitting
params, fitted_values = fit_model(t, observations, peak_frequencies)
print("Found parameters:", params)
```

Отримуємо:

```
Found parameters: [ 2.16038579 -4.52119607  9.15526181  1.01278389  0.49308587]
```

І нарешті, отримуємо апроксимовану функцію, будуємо її графік та моделі:

```
# 4. Plot the original and fitted values
plt.figure()
plt.plot(t, observations, label='Observations')
plt.plot(t, fitted_values, label='Model', linestyle='--')
plt.xlabel('Time')
plt.ylabel('y(t)')
plt.legend()
plt.title('Comparison of Observations and Model')
plt.show()
```

Отримуємо: