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

Лабораторна робота №1
з Моделювання складних систем
“Дискретне перетворення Фур’є”
Варіант №12

Виконав студент групи ІПС-31
Тесленко Назар Олександрович

Київ - 2025

Постановка задачі:

На інтервалі спостереження [0, T], де T=5, спостережуваний сигнал $\widehat{y}(t_i)$ заданий у дискретні моменти часу $t_i \in [0, T],\ i = 1, 2, ..., N$-1 з кроком дискретизації $t_{i+1} - t_i = \Delta t = 0.01$

$$N = T/\Delta t = 5/0.01 = 500$$

Таким чином для подальшого аналізу маємо 500 дискретних значень сигналу у відповідних моментах часу.

Для визначення суттєвих частотних складових сигналу застосовується дискретне перетворення Фур'є (ДПФ). Воно дає змогу перейти від аналізу сигналу у часовій області до його представлення у частотній області. Це дає можливість встановити гармонічні компоненти, що роблять найбільший внесок у формування сигналу - тобто визначити частоти, на яких спостерігаються пікові значення амплітуди.

Для обчислення ДПФ використовується наступна формула:

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

- $i^2 = -1$ - комплексна одиниця,
- $e^{i\phi} = cos(\phi) + isin(\phi)$

Результатом є масив комплексних чисел. Для подальшого аналізу використовуються їх модулі $|c_x(k)|$, які відповідають амплітудам спектра.

Для дійсних сигналів спектр $|c_x(k)|$ - є симетричним, тож для подальших досліджень будемо використовувати лише половину його значень: $k = 0, 1, 2, ..., N/2$

Крок по частоті визначається як:

$$\Delta f = 1/T \Rightarrow 1/5 = 0.2 \text{ Гц}$$

Таким чином, кожному індексу k відповідає частота:

$$f_k = k * \Delta f$$

Для визначення найбільш значущих складових сигналу проаналізуємо локальні максимуми спектра $|c_x(k)|$

Частота, що відповідає піковому значенню, обчислюється як:

$$f_* = k_* * \Delta f$$

Знайшовши частоти із найбільшим вкладом, можна приступати до визначення невідомих параметрів $a_i,\ i = k + 1$. Застосовуємо метод найменших квадратів для їх визначення

Функціонал похибки:

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

Далі шукаємо параметри a_i з умови:

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

Для цього записуємо систему рівнянь:

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

Ця система є системою лінійних алгебраїчних рівнянь, розв’язавши яку, одним з відоих методів, знаходимо $a_i,\ i = 1, 2,\ ...\ , k + 1$

Кінцевим кроком є побудова математичної моделі у вигляді суми поліном. частини та гармонічних складових:

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

де :

- $a_1,\ a_2,\ a_3$ - коефіцієнти поліноміальної частини
- a_i - амплітуди гармонічних складових
- f_i - значущі частоти
- a_{k+1} - стала складова сигналу

Хід роботи

Мова реалізації програми: Python
Використані бібліотеки: NumPy, Matplotlib, SciPy

1. Реалізація власної функції дискретного перетворення Фур'є.

Була реалізована функція **`dft()`**, для обчислення ДПФ для конкретної дискретної послідовності.

```
#DFT - Discrete Fourie Transformation
def dft(samples:np.ndarray)->np.ndarray:
    """
    Own implementation of Discrete Fourier Transform (DFT).
    """
    N=len(samples)
    dft_computation=np.zeros(N,dtype=complex)
    for k in range(N):
        for m in range(N):
            dft_computation[k] += (samples[m]*np.exp(-2j*np.pi*k*m/N))
    return dft_computation
```

Перевіримо корректність обчислень функції за допомогою функції ДПФ з бібліотеки NumPy:

```
def dft_comparer(samples:np.ndarray)->None:
    """
    Comparing DFT result of lib numpy's func and own DFT func
    """
    my_dft=dft(samples)
    numpy_fft_res=np.fft.fft(samples)
    max_absolute_error = np.max(np.abs(my_dft - numpy_fft_res))

    if max_absolute_error < 1e-6:
        print("Own Discrete Fourier Transformation matches NumPy's FFT.")
    else:
        print("Own Discrete Fourier Transformation does not match NumPy's FFT.")
        print(f"Maximum absolute error: {max_absolute_error}")
```

```
Own Discrete Fourier Transformation matches NumPy's FFT.
```

Отже результати обчислень власної функції ДПФ - правильні.

2. Пошук локальних максимумів

Була реалізована функція **`find_frequencies_contribution()`**, яка використовує дані ДПФ по модулю, що відповідає амплітудам спектра, і відповідно занходить локальні максимуми

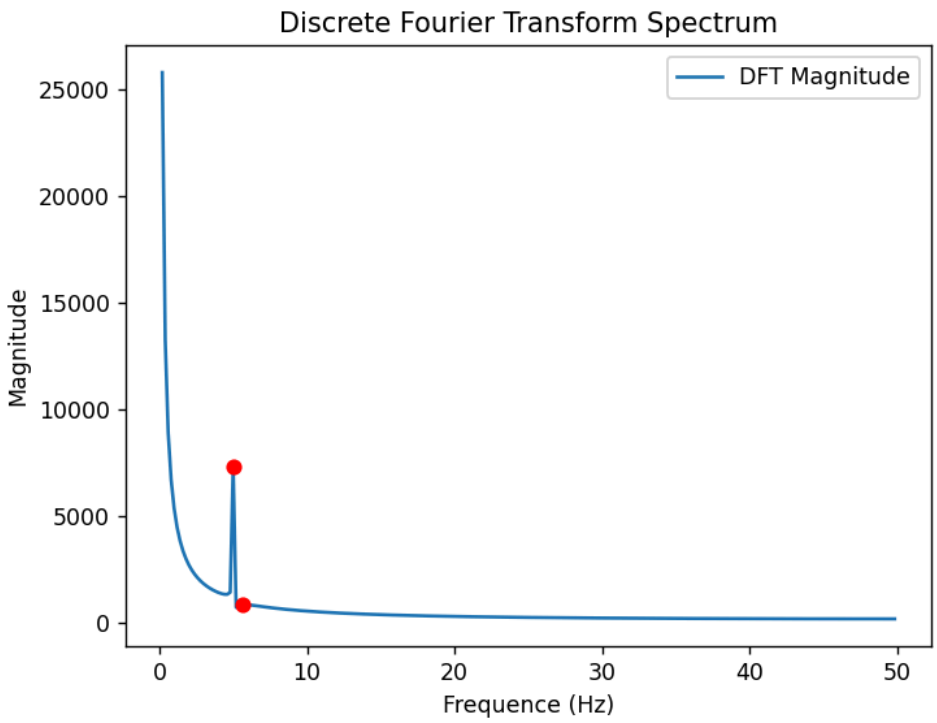
```python
def find_frequencies_contribution(samples:np.ndarray, interval:int=5):
    """
    Find main frequency contributions in the signal using DFT peaks.
    """

    step=1/interval # deltaf=1/T
    samples_number=len(samples)
    dft_abs=np.abs(dft(samples)) #Find the magnitude
    print(dft_abs)
    peak_indices=[]

    #As the right part of signal is a mirrored left one using only the half of range
    for k in range(1,(samples_number//2)-1):
        if(dft_abs[k]>dft_abs[k-1] and dft_abs[k]>dft_abs[k+1]):
            peak_indices.append(k)

    plot_dft(dft_abs, step, 'dicrete_fourier_transform.png',peak_indices)

    frequencies = [k*step for k in peak_indices]
    return frequencies
```

Результат:

```
Peak frequencies:  [5.  5.6] Hz
```

3. Побудова графіка спектра амплітуд ДПФ

Для побудови графіка була реалізована функція **`plot_dft()`**



4. Побудова математичної моделі сигналу

```python
def fit_model(t,data, peak_frequencies):
    """
    Fit polynomial + sinusoidal model to the signal.
    """
    def model (t, a1,a2,a3,a_const, *params):
        k=len(params)//2
        y=a1*t**3+a2*t**2+a3*t + a_const
        for i in range(k):
            fi=params[i]
            ai=params[i+k]
            y+= ai*np.sin(2*np.pi*fi*t)
        return y
    initial_guess=[0,0,0,0]+peak_frequencies+[1]*len(peak_frequencies)
    params,covariance = curve_fit(model,t,data,p0=initial_guess)
    fitted_values=model(t, *params)
    return params, fitted_values
```

Результат:

```
Found parameters: [ 3.00000019e+00 -2.00000143e+00  2.00000294e+00 -1.00000013e+01
  5.00000000e+00  5.35446844e+00 -2.49999935e+01  4.93737603e-07]
```

У практичній реалізації розв’язання системи рівнянь для пошуку невідомих параметрів a_i виконується автоматично за допомогою функції curve_fit .
Функція знаходить невдомі параметри за методом наймменших квадратів, тобто мінімізую функціонал похибки F відносно параметрів a_i.

Отримали парамтери, підставимо їх та випишемо моделюючу функцію:

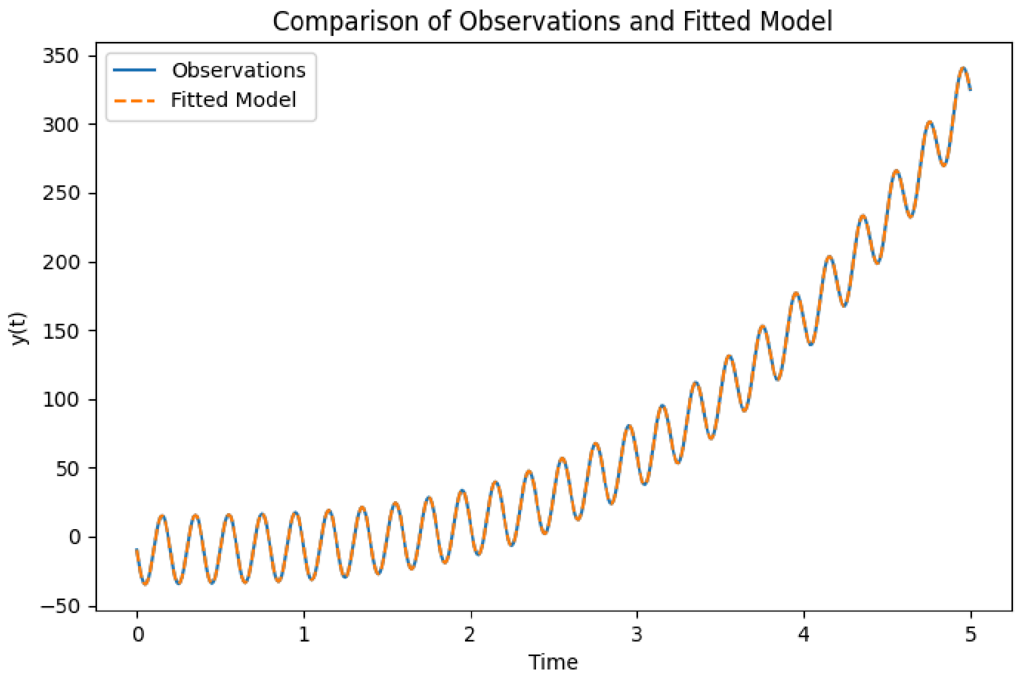
$$\hat{y}(t) = a_1 t^3 + a_2 t^2 + a_3 t + \sum_{i=4}^{k} a_i sin(2\pi f_{i-3} t) + a_{k+1}$$

$$\hat{y}(t) \approx 3t^3 - 2t^2 + 2t - 25\sin(2\pi * 5t) - 10$$

У результаті аналізу спектра було виявлено домінуючу гармоніку на частоті 5 Гц, використовуємо лише її у даній функції.

5. Побудова графіка для порівняння

```
params, fitted_values=fit_model(t,observations,peak_frequencies)
print("Found parameters:", params)
plt.figure(figsize=(8, 5))
plt.plot(t, observations, label='Observations')
plt.plot(t, fitted_values, label='Fitted Model', linestyle='--')
plt.legend()
plt.xlabel('Time')
plt.ylabel('y(t)')
plt.title('Comparison of Observations and Fitted Model')
plt.savefig("fitmodel_comparison.png")
plt.show()
```



Можемо бачити, що створена модель відповідає спостереженням, поліном та знайдені сінусоїди повністю відтворюють структуру сигналу.