Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інституту атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

**Проектування та аналіз обчислювальних алгоритмів**

ЗВІТ

до лабораторної роботи № 3

«”Пряме перетворення Фур'є”»

Дата «27» листопад 2023 Виконав: студент *2* курсу

гр. *ТР-24*

Хоменко Олег Святославович

*(П.І.Б.)*

Оцінка «\_\_\_\_\_\_\_\_\_» Перевірив: доцент

Дата «\_\_\_» \_\_\_\_\_\_\_ 2023 Донець Андрій Георгійович

*(П.І.Б., підпис)*

Київ – 2023

**Завдання:**

1. Задача: x(i) → X(k) – виконати перехід (спектральний аналіз) з часової площини A= f(t) в частотну A= f(k).

2. На базі даних, які було отримано в попередніх роботах (ЛР No 1 + 2 – це файли з вхідними даними name.dat (вхідні дані:Т = 1 сек, N = 1000, Δt = 0.001) та файл 001.dat. x(i) − прочитати із файлу даних) необхідно розробити застосунок для виконання Прямого Перетворення Фур’є вхідних даних x(i).

Математична модель:

1.

2.

Розрахункова модель (без -j):

1.

2.

3.

3. У якості тестового сигналу (filename.tst) використати дані, що сформовані в попередніх роботах на базі наступної математичної моделі:

Де ;

fk = 50, 60, 400 Гц;

A = 220, 110, 36 B.

4. У якості «робочого» сигналу використати файл 001.dat.

5. Результати розрахунків (передбачити можливість модифікації вхідних даних) відобразити у вигляді графіків A = f(t) та A = f(k).

**Теоретичні відомості**

**Пряме перетворення Фур'є (FFT):**

Пряме перетворення Фур'є - це математичний метод, що використовується для перетворення сигналу з часового представлення в частотне. Це основний інструмент в аналізі сигналів і обробці сигналів. Пряме перетворення Фур'є може бути застосоване до будь-якого сигналу, який визначений на проміжку часу, і дозволяє отримати його частотне представлення.

**Математична модель для прямого перетворення Фур'є:**

Для дискретного сигналу x(i) довжиною N з інтервалом дискретизації , пряме перетворення Фур’є обчислюється за формулою:

де k - індекс частоти, j - уявна одиниця, а exp - функція експоненти.

Розрахункова модель для прямого перетворення Фур'є без уявної одиниці: Щоб обчислити дійсні та уявні частини прямого перетворення Фур'є, використовуються косинуси та синуси:

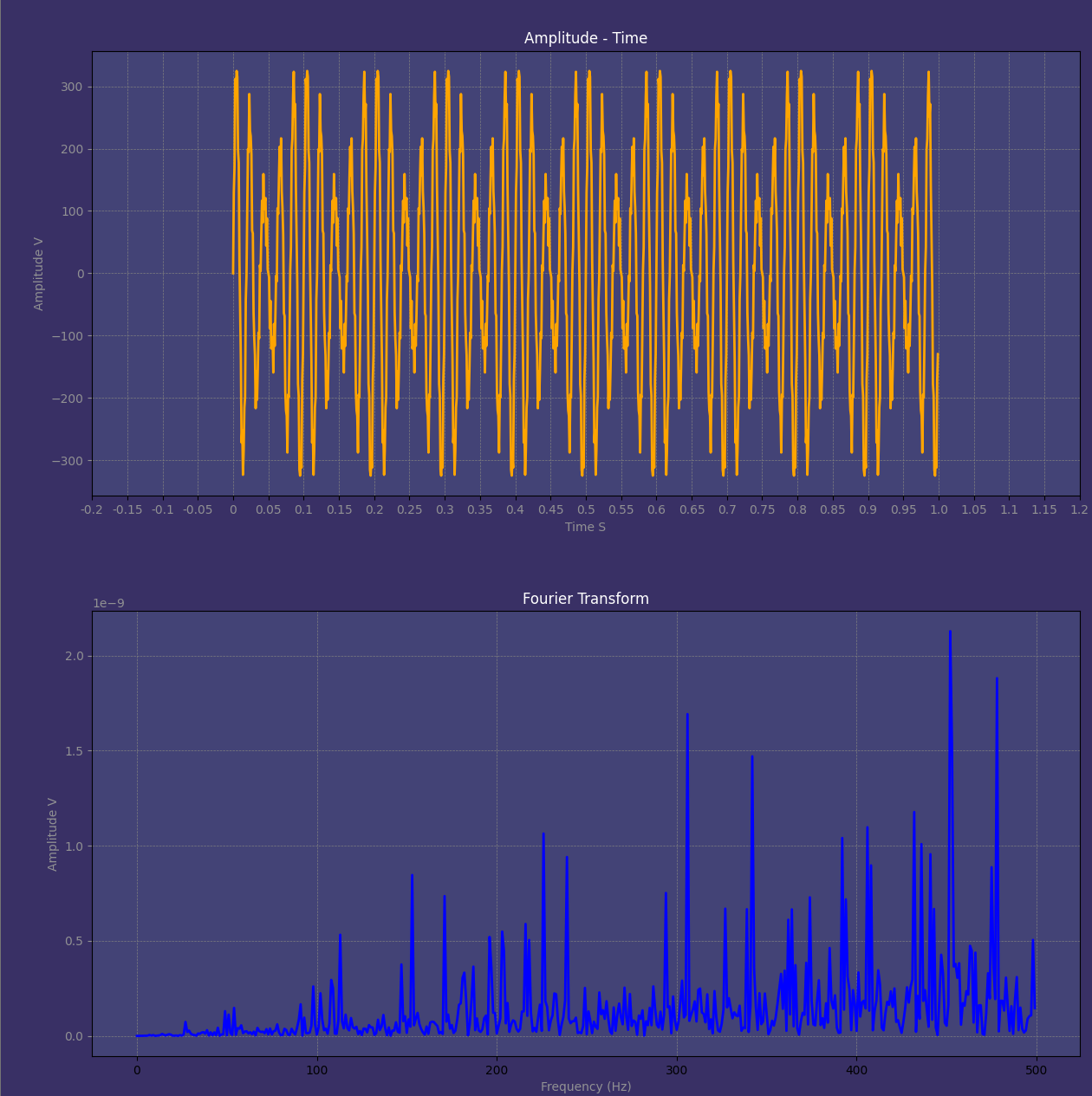
1.

2.

Амплітуда A(k) обчислюється як корінь квадратний від суми квадратів дійсної та уявної частин:



**Результати роботи**

Використання FFT для тестового сигналу:

Використання FFT для робочого сигналу:

Посилання на завдання [Replit](https://replit.com/join/kuxgkqyjqv-mrfounder) і на [GitHub](https://github.com/FounderC/PAOA-Laboratory/tree/main/%D0%9B%D0%B0%D0%B1%203).

**Висновок:** Під час виконання лабораторної роботи я розробив та впровадив алгоритм для обробки та аналізу сигналів за допомогою прямого перетворення Фур'є. Для тестового сигналу згенерував суму гармонік з визначеними частотами та амплітудами, зберігши результат у файлі. Після цього застосував фільтрацію до тестового сигналу, використовуючи різні діапазони частот, і відобразив результати на графіках. Також, успішно обробив робочий сигнал, який був зчитаний з файлу 001.dat. Обчислив та відобразили пряме перетворення Фур'є цього сигналу, представивши його у вигляді графіків амплітуди від часу та амплітудного спектру від частоти.

**Додаток**

MAX\_NUMBER = 65535  
BYTE\_ORDER = 'little'  
  
def generate\_test\_signal(T, N, dt, frequencies, amplitudes):  
 t = np.linspace(0.0, T, N, endpoint=False)  
 x = np.zeros(N)  
 for f, A in zip(frequencies, amplitudes):  
 x += A \* np.sin(2 \* np.pi \* f \* t)  
 return t, x  
  
def filter\_signal(signal, lowcut, highcut, fs, order=2):  
 nyquist = 0.5 \* fs  
 low = lowcut / nyquist  
 high = highcut / nyquist  
 b, a = butter(order, [low, high], btype='band')  
 y = lfilter(b, a, signal)  
 return y  
  
def parse\_sequence(filepath: Union[Path, str]) -> List[int]:  
 chunk\_size = 2  
 y = []  
 with open(filepath, 'rb') as f:  
 while True:  
 chunk = f.read(chunk\_size)  
 if not chunk:  
 break  
 y\_i = int.from\_bytes(chunk, BYTE\_ORDER)  
 if y\_i > 5000:  
 y\_i = y\_i - MAX\_NUMBER  
 y.append(y\_i)  
 return y  
  
  
signal\_type = input("Виберіть тип сигналу (тестовий/робочий/перетворення): ")  
  
if signal\_type.lower() == "тестовий":  
 frequencies = [50, 60, 400]  
 amplitudes = [220, 110, 36]  
 T = 1.0  
 N = 1000  
 dt = 1.0 / N  
  
 t, x = generate\_test\_signal(T, N, dt, frequencies, amplitudes)  
  
 lowcut = 10  
 highcut = 100  
  
 np.savetxt('C:/Users/Khome/Desktop/ЛабаФайл/filename.tst', np.column\_stack((t, x)), fmt='%0.6f', delimiter='\t')  
  
 filtered\_signal = filter\_signal(x, lowcut, highcut, 1.0 / dt)  
 np.savetxt('C:/Users/Khome/Desktop/ЛабаФайл/filename.dL1', np.column\_stack((t, filtered\_signal)), fmt='%0.6f', delimiter='\t')  
  
 filtered\_signal\_l2 = filter\_signal(x, lowcut, 20, 1.0 / dt)  
 np.savetxt('C:/Users/Khome/Desktop/ЛабаФайл/filename.dL2', np.column\_stack((t, filtered\_signal\_l2)), fmt='%0.6f', delimiter='\t')  
  
 filtered\_signal\_h3 = filter\_signal(x, 20, highcut, 1.0 / dt)  
 np.savetxt('C:/Users/Khome/Desktop/ЛабаФайл/filename.dh3', np.column\_stack((t, filtered\_signal\_h3)), fmt='%0.6f', delimiter='\t')  
  
 filtered\_signal\_h4 = filter\_signal(x, 100, 400, 1.0 / dt)  
 np.savetxt('C:/Users/Khome/Desktop/ЛабаФайл/filename.dh4', np.column\_stack((t, filtered\_signal\_h4)), fmt='%0.6f', delimiter='\t')  
  
 t\_min, t\_max = -0.2, 1.2  
 custom\_xticks = [-0.2, -0.15, -0.1, -0.05, 0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6,  
 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0, 1.05, 1.1, 1.15, 1.2]  
 custom\_xlabels = ['-0.2', '-0.15', '-0.1', '-0.05', '0', '0.05', '0.1', '0.15', '0.2', '0.25', '0.3', '0.35', '0.4',  
 '0.45', '0.5', '0.55', '0.6', '0.65', '0.7', '0.75', '0.8', '0.85', '0.9', '0.95', '1.0', '1.05', '1.1', '1.15', '1.2']  
  
 fig = plt.figure()  
 fig.patch.set\_facecolor('#393065')  
 ax = fig.add\_subplot(111)  
 ax.plot(t, x, color='orange', linewidth=2)  
 ax.set\_xlabel('Time S', color='#919193')  
 ax.set\_ylabel('Amplitude V', color='#919193')  
 ax.grid(color='gray', linestyle='--', linewidth=0.5)  
 ax.set\_title('Amplitude - Time', color='white')  
 ax.set\_facecolor('#434376')  
 ax.set\_xticks(custom\_xticks)  
 ax.set\_xticklabels(custom\_xlabels, color='#919193')  
 ax.tick\_params(axis='y', colors='#919193')  
 plt.xlim(t\_min, t\_max)  
 plt.show()  
  
 X = np.fft.fft(x)  
 freq = np.fft.fftfreq(N, dt)  
  
 fig, ax = plt.subplots()  
 ax.plot(freq[:N // 2], np.abs(X)[:N // 2])  
 fig.patch.set\_facecolor('#393065')  
 ax.set\_xlabel('Frequency (Hz)', color='#919193')  
 ax.set\_title('Amplitude - Time', color='#919193')  
 ax.set\_title('Frequency Spectrum', color='white')  
 ax.grid(color='gray', linestyle='--', linewidth=0.5)  
 ax.set\_facecolor('#434376')  
 ax.tick\_params(axis='y', colors='#919193')  
 plt.show()  
  
elif signal\_type.lower() == "робочий":  
  
 data\_file\_path = 'C:/Users/Khome/Desktop/ЛабаФайл/001.dat'  
  
 y = parse\_sequence(data\_file\_path)  
  
 t = np.linspace(-0.2, 1.2, len(y))  
  
 t\_min, t\_max = -2.5, 3  
 custom\_xticks = [-2.5, -2.25, -2, -1.75, -1.5, -1.25, -1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3]  
 custom\_xlabels = custom\_xticks  
  
 fig = plt.figure()  
 fig.patch.set\_facecolor('#393065')  
  
 ax = fig.add\_subplot(111)  
  
 ax.plot(t, y, color='#2B82D5', linewidth=2)  
 ax.set\_xlabel('Time S', color='#919193')  
 ax.set\_ylabel('Amplitude V', color='#919193')  
 ax.grid(color='gray', linestyle='--', linewidth=0.5)  
 ax.set\_title('Amplitude - Time', color='white')  
 ax.set\_facecolor('#434376')  
 ax.set\_xticks(custom\_xticks)  
 ax.set\_xticklabels(custom\_xlabels, color='#919193')  
 ax.tick\_params(axis='y', colors='#919193')  
 plt.xlim(t\_min, t\_max)  
 ax.axhline(y=0, color='#2B82D5', linestyle='-', linewidth=2)  
 plt.show() # Відображення графіку  
  
 Y = np.fft.fft(y)  
 freq = np.fft.fftfreq(len(y), abs(t[1] - t[0]))  
  
 fig, ax = plt.subplots()  
 ax.plot(freq[:len(y) // 2], np.abs(Y)[:len(y) // 2])  
 fig.patch.set\_facecolor('#393065')  
 ax.set\_xlabel('Frequency (Hz)', color='#919193')  
 ax.set\_title('Amplitude - Time', color='#919193')  
 ax.set\_title('Frequency Spectrum', color='white')  
 ax.grid(color='gray', linestyle='--', linewidth=0.5)  
 ax.set\_facecolor('#434376')  
 ax.tick\_params(axis='y', colors='#919193')  
 plt.show()  
  
elif signal\_type.lower() == "перетворення":  
  
 def compute\_fourier\_transform(x, dt):  
 N = len(x)  
 freq = np.fft.fftfreq(N, dt)  
 X = np.fft.fft(x)  
  
 Xr = dt \* np.sum(x \* np.cos(2 \* np.pi \* np.outer(freq, np.arange(N))), axis=1)  
 Xm = dt \* np.sum(x \* np.sin(2 \* np.pi \* np.outer(freq, np.arange(N))), axis=1)  
 A = np.sqrt(Xr\*\*2 + Xm\*\*2)  
  
 return freq, A  
  
 frequencies\_test = [50, 60, 400]  
 amplitudes\_test = [220, 110, 36]  
 T\_test = 1.0  
 N\_test = 1000  
 dt\_test = 1.0 / N\_test  
  
 t\_test, x\_test = generate\_test\_signal(T\_test, N\_test, dt\_test, frequencies\_test, amplitudes\_test)  
  
 freq\_test, A\_test = compute\_fourier\_transform(x\_test, dt\_test)  
  
 fig, ax = plt.subplots(2, 1, figsize=(10, 8))  
  
 custom\_xticks = [-0.2, -0.15, -0.1, -0.05, 0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6,  
 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0, 1.05, 1.1, 1.15, 1.2]  
 custom\_xlabels = ['-0.2', '-0.15', '-0.1', '-0.05', '0', '0.05', '0.1', '0.15', '0.2', '0.25', '0.3', '0.35', '0.4',  
 '0.45', '0.5', '0.55', '0.6', '0.65', '0.7', '0.75', '0.8', '0.85', '0.9', '0.95', '1.0', '1.05', '1.1', '1.15', '1.2']  
  
 fig.patch.set\_facecolor('#393065')  
  
 ax[0].plot(t\_test, x\_test, color='orange', linewidth=2)  
 ax[0].set\_xlabel('Time S', color='#919193')  
 ax[0].set\_ylabel('Amplitude V', color='#919193')  
 ax[0].grid(color='gray', linestyle='--', linewidth=0.5)  
 ax[0].set\_title('Amplitude - Time', color='white')  
 ax[0].set\_facecolor('#434376')  
 ax[0].set\_xticks(custom\_xticks)  
 ax[0].set\_xticklabels(custom\_xlabels, color='#919193')  
 ax[0].tick\_params(axis='y', colors='#919193')  
  
 ax[1].plot(freq\_test[:N\_test // 2], A\_test[:N\_test // 2], color='blue', linewidth=2)  
 ax[1].set\_xlabel('Frequency (Hz)', color='#919193')  
 ax[1].set\_ylabel('Amplitude V', color='#919193')  
 ax[1].grid(color='gray', linestyle='--', linewidth=0.5)  
 ax[1].set\_title('Fourier Transform', color='white')  
 ax[1].tick\_params(axis='y', colors='#919193')  
 ax[1].set\_facecolor('#434376')  
  
 plt.tight\_layout()  
 plt.show()  
  
 data\_file\_path = 'C:/Users/Khome/Desktop/ЛабаФайл/001.dat'  
  
 y = parse\_sequence(data\_file\_path)  
  
 t = np.linspace(-0.2, 1.2, len(y))  
  
 dt\_work = t[1] - t[0]  
  
 freq\_work, A\_work = compute\_fourier\_transform(y, dt\_work)  
  
 fig, ax = plt.subplots(2, 1, figsize=(10, 8))  
  
 custom\_xticks = [-2.5, -2.25, -2, -1.75, -1.5, -1.25, -1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3]  
 custom\_xlabels = custom\_xticks  
  
 fig.patch.set\_facecolor('#393065')  
  
 ax[0].plot(t, y, color='#2B82D5', linewidth=2)  
 ax[0].set\_xlabel('Time (s)', color='#919193')  
 ax[0].set\_ylabel('Amplitude', color='#919193')  
 ax[0].set\_facecolor('#434376')  
 ax[0].set\_xticks(custom\_xticks)  
 ax[0].set\_xticklabels(custom\_xlabels, color='#919193')  
 ax[0].set\_title('Original Signal', color='white')  
 ax[0].tick\_params(axis='y', colors='#919193')  
 ax[0].axhline(y=0, color='#2B82D5', linestyle='-', linewidth=2)  
  
 ax[1].plot(freq\_work[:len(y) // 2], A\_work[:len(y) // 2], color='#2B82D5', linewidth=2,)  
 ax[1].set\_xlabel('Frequency (Hz)', color='#919193')  
 ax[1].grid(color='gray', linestyle='--', linewidth=0.5)  
 ax[1].set\_facecolor('#434376')  
 ax[1].tick\_params(axis='y', colors='#919193')  
 ax[1].set\_ylabel('Amplitude', color='#919193')  
 ax[1].set\_title('Fourier Transform', color='white')  
  
 plt.tight\_layout()  
 plt.show()  
  
else:  
 print("Ви ввели невірний варіант. Спробуйте ще раз.")