Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра автоматизації проектування

 енергетичних процесів і систем

**Звіт**

**з циклу лабораторних робіт**

**з дисципліни**

**„Високонавантажені застосунки”**

Лабораторна  робота № 3

Тема : Пряме перетворення Фур'є

Виконала: **Юрченко Богдана**

Студентка групи:   **ТВ-01 мп**

Перевірив викладач: Крячок О. С.

Київ - 2021

# Завдання до роботи

**1.** На базі даних, які було отримано в попередніх роботах (ЛР № 1 + 2 – це файли з вхідними даними name.dat (Т = 1 сек, N = 1000, Δt = 0.001) та файл 001.bk0) необхідно розробити застосунок для виконання Прямого Перетворення Фур’є вхідних даних x(i). Скористатися теоретичними відомостями із доданого файлу.

**2.** У якості тестового сигналу (файл name.tst) використати дані, що сформовані в попередніх роботах на базі наступної математичної моделі  
x(i) = ∑(A1 \* sin(w1 \* Δt + φ0) + A2 \* sin(w2 \* Δt + φ0) + A3 \* sin(w3 \* Δt + φ0)), де w = 2 \* π \* fi; fk = 50, 60, 400 Гц; А = 220, 110, 36 В.

**3.** У якості «робочого» сигналу використати файл 001.bk0.

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

# Результати роботи

Було розроблено додаток для виконання Прямого Перетворення Фур’є вхідних даних на мові програмування Python.

В якості даних були використані дані зегеровані в попередній роботі та дані із файлу 001.bk0.

Було використано модуль бібліотеки numpy fft.

Додаток має функції зчтування даних із двійкового файлу, генерації даних, виконання прямого теретворення Фур’є та генерації графіків.

Результатами запуску програми є графіки зображені далі.

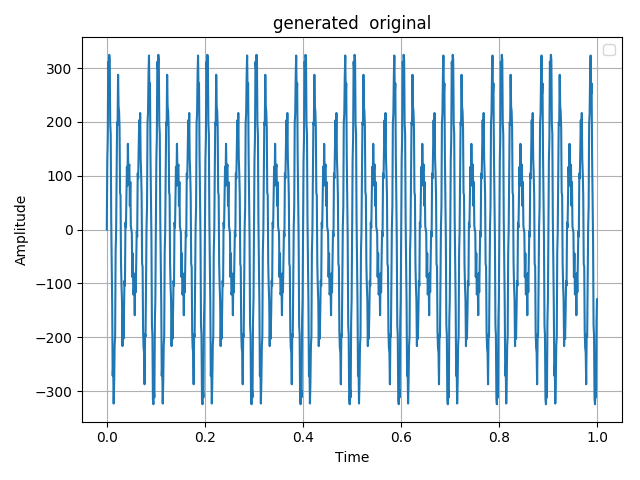


Рис. 1 Оригінальний згенерований сигнал

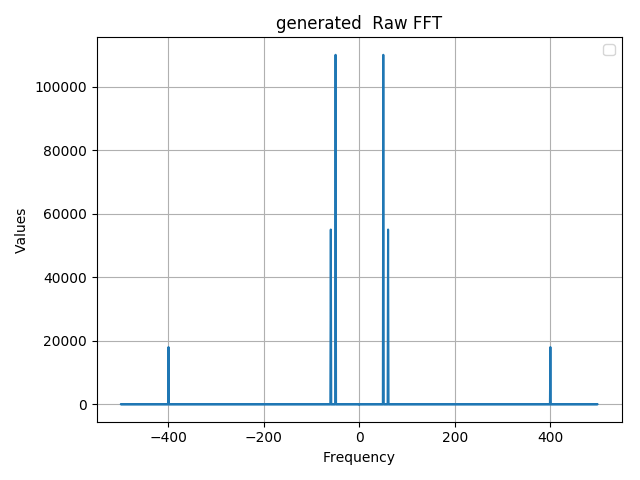


Рис. 2. Перетворений сигнал (Пряме Перетворення Фур’є)

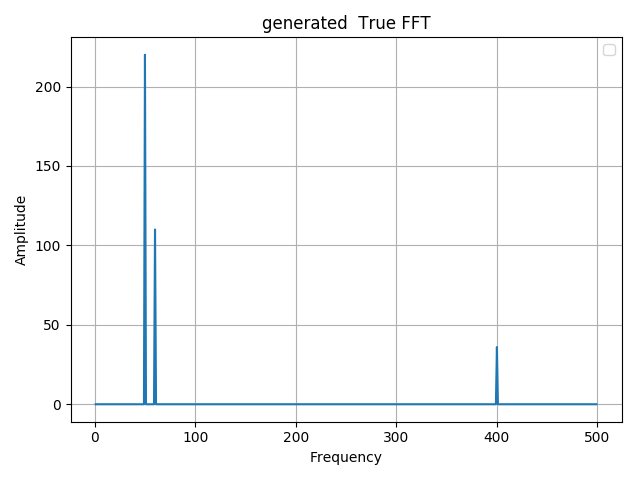


Рис. 3. Перетворений сигнал (Прямого Перетворення Фур’є)

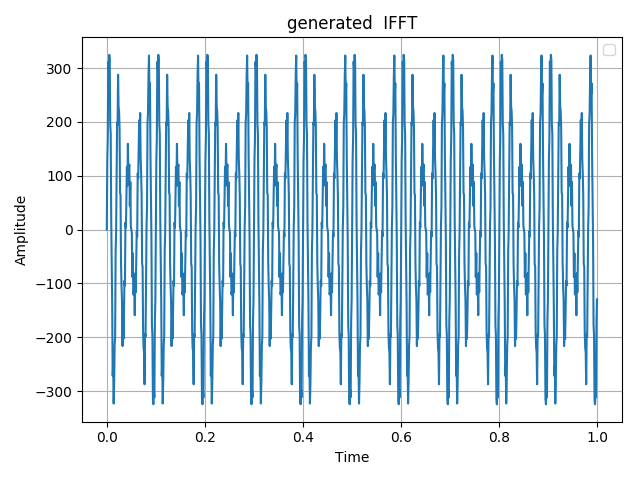


Рис. 4. Відновлений початковий сигнал (Обернене перетворення Фур'є)

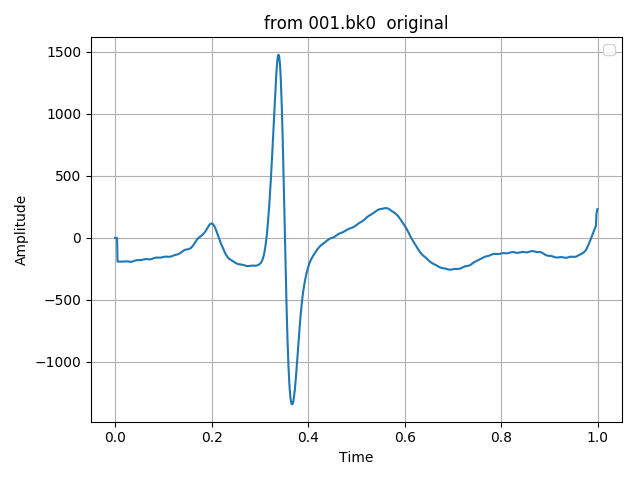


Рис. 5 Оригінальний зчитаний сигнал

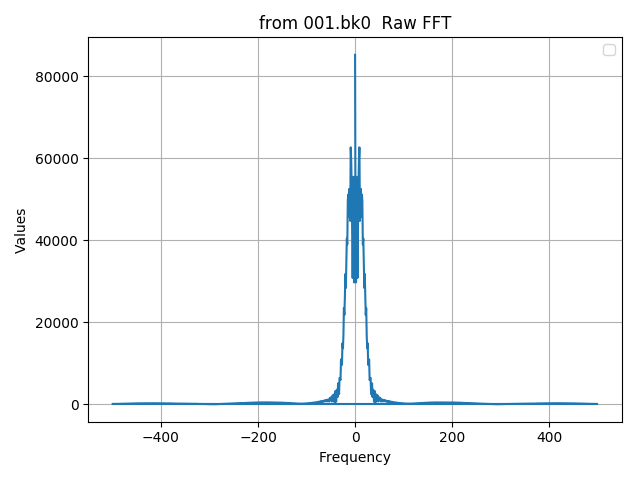


Рис. 6. Перетворений сигнал (Пряме Перетворення Фур’є)

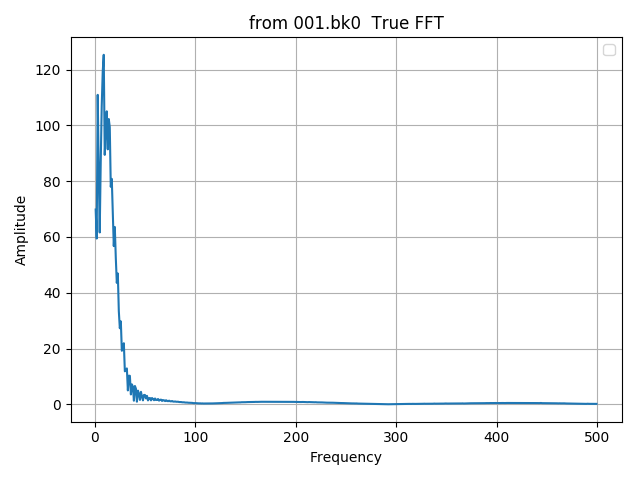


Рис. 7. Перетворений сигнал (Прямого Перетворення Фур’є)

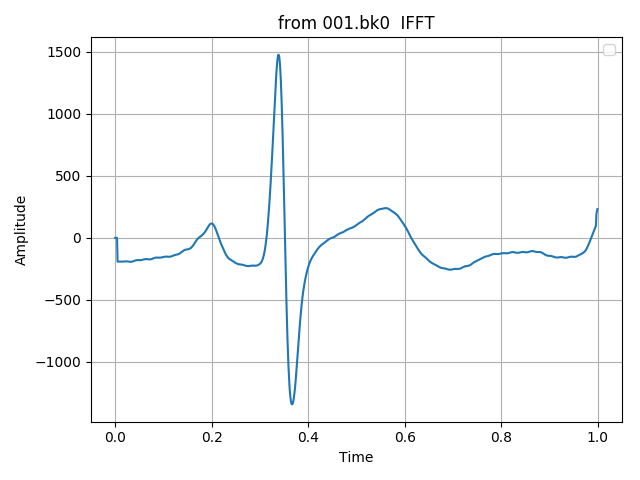


Рис. 8. Відновлений початковий сигнал (Обернене перетворення Фур'є)

# Висновки по роботі

В даній роботі була досліджена перетворення сигналів за допомогою Прямого Перетворення Фур’є та Оберненого перетворення Фур'є.

Було розроблено програмне забеспечення для фільтрації чатот сигналів для подальшого використання очищених сигналів. .

Розроблене програмне забеспечення також уміє зчитувати дані із двійкових файлів.

# Програмний код

import numpy as np

import matplotlib.pyplot as plt

def read\_from\_file(file\_name):

with open(file\_name, 'rb') as f:

data = f.read()

f.close()

arr = np.frombuffer(data, dtype=np.int16)

return arr

def show(title, x, y, x\_label, y\_label):

fig, ax = plt.subplots(figsize=(20, 10))

ax.plot(x, y)

ax.set\_title(title)

ax.set\_xlabel(x\_label)

ax.set\_ylabel(y\_label)

ax.legend()

plt.grid(True)

plt.show()

def generate(x, frequency, amplitude):

return [amplitude \* np.sin(2 \* np.pi \* frequency \* i) for i in x]

def generate\_data(generated\_x):

w1 = generate(generated\_x, 50, 220)

w2 = generate(generated\_x, 60, 110)

w3 = generate(generated\_x, 400, 36)

return [w1[i] + w2[i] + w3[i] for i in range(len(generated\_x))]

def process\_data(title, x, y, N):

show(title + "original", x, y, "Time", "Amplitude")

# Raw FFT

# Compute the Discrete Fourier Transform sample frequencies

frequencies = np.fft.fftfreq(N, d=0.001)

mask = frequencies > 0

# Compute the one-dimensional discrete Fourier Transform

fft\_values = np.fft.fft(y)

show(title + "Raw FFT", frequencies, np.abs(fft\_values), "Frequency", "Values")

# True FFT

fft\_theo = 2.0 \* np.abs(fft\_values / N)

show(title + "True FFT", frequencies[mask], fft\_theo[mask], "Frequency", "Amplitude")

# IFFT

# Compute the one-dimensional inverse discrete Fourier Transform

ifft\_values = np.fft.ifft(fft\_values)

show(title + "IFFT", x, ifft\_values.real, "Time", "Amplitude")

N = 1000

generated\_x = [i \* 1.0 / 1000 for i in np.arange(N)]

generated\_y = generate\_data(generated\_x)

process\_data("generated ", generated\_x, generated\_y, N)

file\_y = read\_from\_file("001.bk0")

process\_data("from 001.bk0 ", generated\_x, file\_y, N)