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 № 1-2

Тема : Цифрові фільтри

Виконала: **Юрченко Богдана**

Студентка групи:   **ТВ-01 мп**

Перевірив викладач: Крячок О. С.

Київ - 2021

# Завдання до роботи

1. Розробити застосунок для цифрової фільтрації вхідних даних x(i), що попередньо записані в файлі name.dat (Т = 1 сек, N = 1000, Δt = 0.001).

2. Для обробки вхідних даних застосувати низькочастотний або високочастотний фільтр 1-го та записати вихідні дані y(i) у файл name.da1.

3. У якості тестового сигналу (файл name.tst) використати дані, що сформовані 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.

4. У якості вхідного сигналу використати файл 001.bk0 з додатку.

5. Результати розрахунків (передбачити можливість модифікації вхідних даних) відобразити у вигляді графіків.

# Теоретичні відомості

Фільтр низьких частот (ФНЧ) - це фільтр, який передає сигнали з частотою, нижчою від вибраної частоти відсічення, і послаблює сигнали з частотами, що перевищують частоту відсічення.

В практиці фільтри низьких частот використовуються в оптиці, акустиці, електроніці. Телефонні лінії, обладнані роздільниками DSL, використовують фільтри низьких і високих частот, щоб розділити сигнали DSL і POTS, що мають однакову пару проводів.

Фільтри низьких частот також відіграють значну роль у формуванні звуку, створеного аналоговими та віртуальними аналоговими синтезаторами.

Фільтр низьких частот використовується як фільтр згладжування перед вибіркою та для реконструкції в цифро-аналогове перетворення.

# Результати роботи

Було розроблено додаток для цифрової фільтрації вхідних даних на мові програмування Python. Було реалізовано фільтр низьких частот першого порядку.

Додаток має функції зчтування даних із двійкового файлу, фільтрації та запису даних у двійковий файл. В якості вхідного сигналу був використаний файл 001.bk0.

Результатами запуску програми є файл name.da1. Та графік 1.

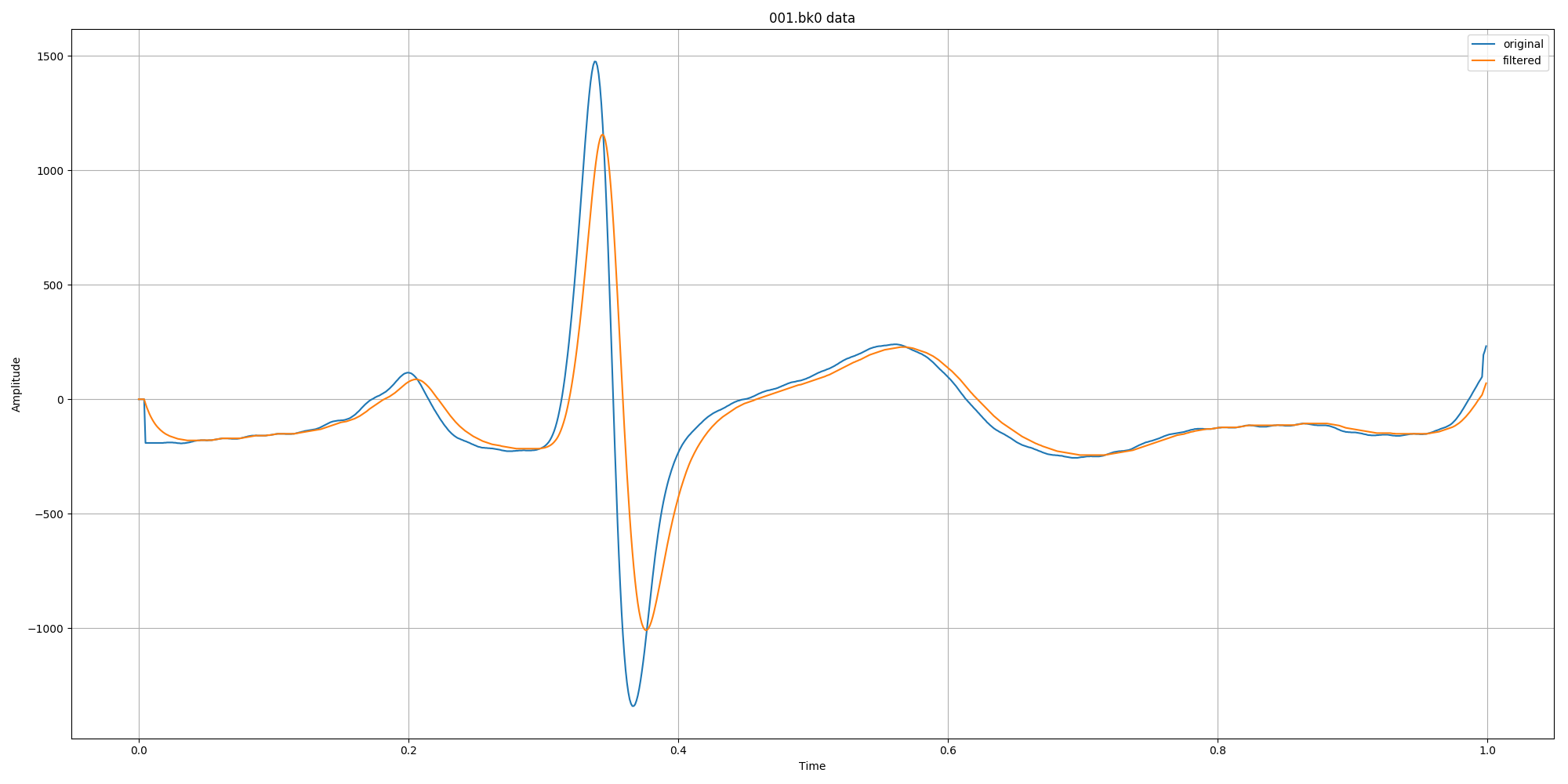


Рис. 1 Оригінальний та відфільтровані сигнали із файлу 001.bk0

На рисунку 2 зображено оригінальний згенерований та відфільтрований сигнали.

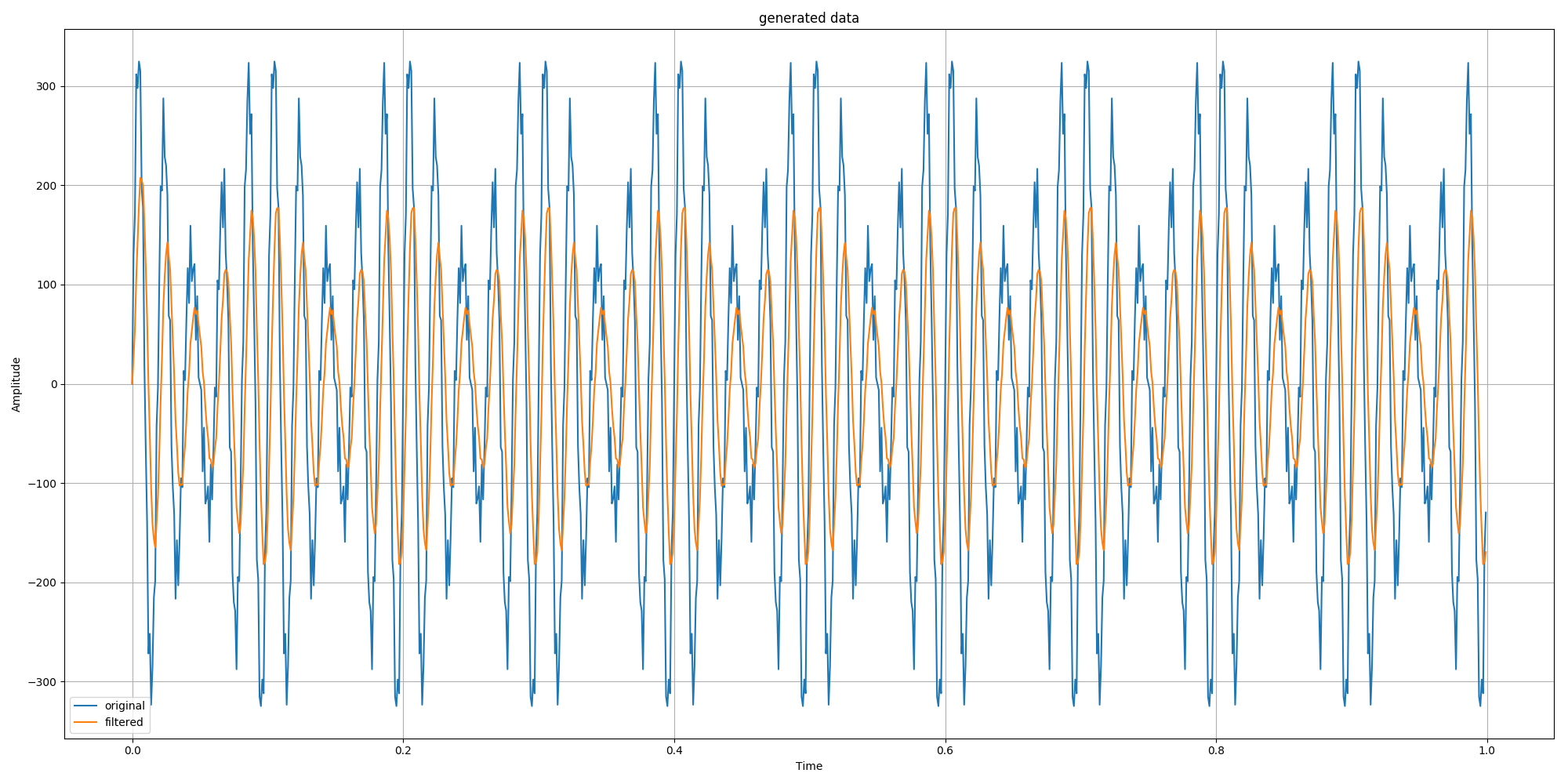


Рис. 2 Згенерований та відфільтрований сигнали

На рисунку 3 зрображено згенерований файл name.da1.

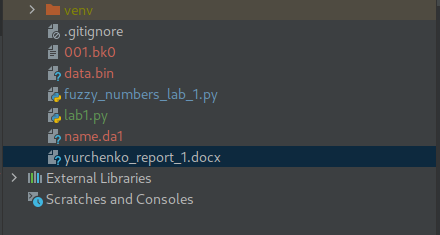


Рис. 3 Згенерований файл name.da1

# Висновки по роботі

В даній роботі була досліджена можливість використання цифрових фільтрів для фільрації деякого сигналу. Також було розроблено програмне забеспечення для генерації тестового сигналу.

Фільтри застововуються для отримання більш чистого сигналу для його подальшого аналізу.

Також було реалізовано програмне забеспечення для зчитування та запису даних у двійкові файли.

# Програмний код

import numpy as np

import matplotlib.pyplot as plt

def write\_to\_file(data, file\_name):

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

f.write(data.tobytes())

f.close()

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 filter\_frequency(data, a):

filtered\_data = np.zeros\_like(data)

filtered\_data[0] = data[0] / 2

for i in range(1, len(data)):

filtered\_data[i] = (1 - a) \* data[i] + a \* filtered\_data[i - 1]

return filtered\_data

def show(name, x, y\_original, y\_filtered):

global fig, ax

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

# ax.plot(x, generated\_original\_data, label='original')

ax.plot(x, y\_original, label='original')

# ax.plot(x, generated\_filtered\_data, label='filtered')

ax.plot(x, y\_filtered, label='filtered')

ax.set\_title(name)

# ax.set\_title()

ax.set\_xlabel('Time')

ax.set\_ylabel('Amplitude')

ax.legend()

plt.grid(True)

plt.show()

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

original\_data = read\_from\_file("001.bk0")

filtered\_data = filter\_frequency(original\_data, 0.9)

write\_to\_file(filtered\_data, "name.da1")

show("001.bk0 data", x, original\_data, filtered\_data)

def make\_wave(x, frequency, amplitude):

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

wave1 = make\_wave(x, 50, 220)

wave2 = make\_wave(x, 60, 110)

wave3 = make\_wave(x, 400, 36)

generated\_original\_data = [wave1[i] + wave2[i] + wave3[i] for i in range(len(x))]

generated\_filtered\_data = filter\_frequency(generated\_original\_data, 0.8)

show("generated data", x, generated\_original\_data, generated\_filtered\_data)