Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Iнститут прикладної математики та фундаментальних наук

Кафедра прикладної математики

ЗВІТ

про виконання лабораторної роботи №2

з курсу «Математичні основи цифрової обробки сигналів»

Варіант №12

**Виконав:**

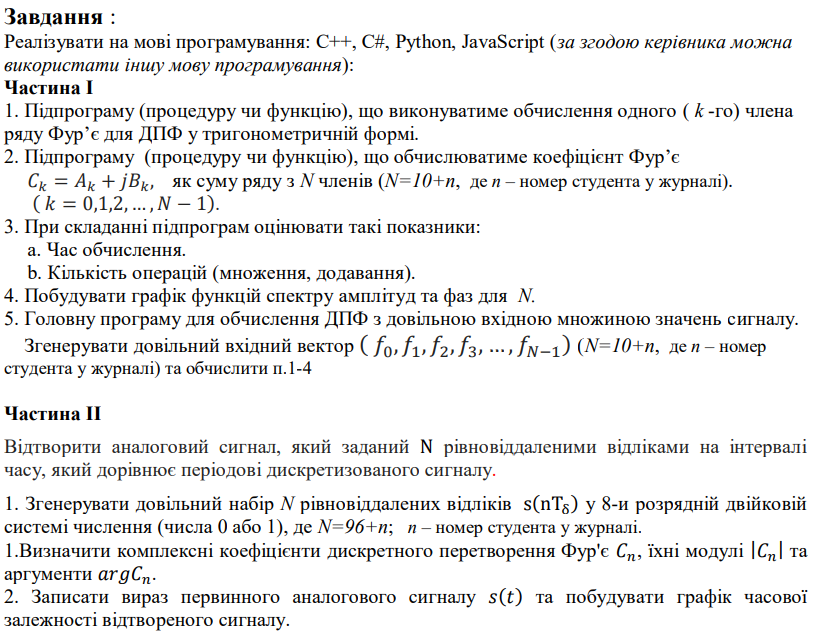
Студент групи ПМ-43

Музика Д.

**Прийняв:**

Пабирівський В.В.

**Мета роботи**: розробити комп’ютерну програму для виконання дискретного перетворення Фур’є (ДПФ).

****

**Виконання**:

Реалізував завдання мовою Python:

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** time

*# 1. Функція для обчислення одного члена ряду Фур'є (k-го)*

**def** compute\_dft\_term(f, k, N):

*"""Обчислення k-го члена ряду Фур'є для ДПФ."""*

real\_part = 0

imag\_part = 0

**for** n **in** range(N):

angle = 2 \* np.pi \* k \* n / N

real\_part += f[n] \* np.cos(angle)

imag\_part -= f[n] \* np.sin(angle) *# Тут використовується знак мінус для уявної частини*

**return** real\_part + 1j \* imag\_part

*# 2. Функція для обчислення всіх коефіцієнтів ряду Фур'є*

**def** compute\_dft(f, N):

*"""Обчислення всіх коефіцієнтів DFT для сигналу f."""*

dft\_result = []

**for** k **in** range(N):

real\_part = 0

imag\_part = 0

**for** n **in** range(N):

angle = 2 \* np.pi \* k \* n / N

real\_part += f[n] \* np.cos(angle)

imag\_part -= f[n] \* np.sin(angle)

dft\_result.append(real\_part + 1j \* imag\_part)

**return** np.array(dft\_result)

*# 3. Функція для обчислення часу та кількості операцій*

**def** compute\_dft\_with\_metrics(f, N):

*"""Обчислення DFT разом із метриками часу та кількості операцій."""*

start\_time = time.time()

mul\_count = 0

add\_count = 0

dft\_result = []

**for** k **in** range(N):

real\_part = 0

imag\_part = 0

**for** n **in** range(N):

angle = 2 \* np.pi \* k \* n / N

real\_part += f[n] \* np.cos(angle)

imag\_part -= f[n] \* np.sin(angle)

mul\_count += 2 *# Два множення (f[n] \* cos та f[n] \* sin)*

add\_count += 1 *# Одне додавання для кожної складової*

dft\_result.append(real\_part + 1j \* imag\_part)

elapsed\_time = time.time() - start\_time

**return** np.array(dft\_result), elapsed\_time, mul\_count, add\_count

*# 4. Функція для побудови графіку спектру амплітуд та фаз*

**def** plot\_spectrum(dft\_result, N):

*"""Побудова графіків амплітудного та фазового спектрів."""*

amplitudes = np.abs(dft\_result) *# Амплітуди - модулі комплексних чисел*

phases = np.angle(dft\_result) *# Фази - аргументи комплексних чисел*

*# Графік амплітуд*

plt.figure(figsize=(10, 5))

plt.stem(range(N), amplitudes, basefmt=**" "**)

plt.title(**"Спектр амплітуд"**)

plt.xlabel(**"k"**)

plt.ylabel(**"Амплітуда"**)

plt.grid(**True**)

plt.show()

*# Графік фаз*

plt.figure(figsize=(10, 5))

plt.stem(range(N), phases, basefmt=**" "**)

plt.title(**"Фазовий спектр"**)

plt.xlabel(**"k"**)

plt.ylabel(**"Фаза (радіани)"**)

plt.grid(**True**)

plt.show()

*# 5. Функція для відтворення аналогового сигналу*

**def** reconstruct\_signal(dft\_result, N, t\_values):

*"""Відтворення аналогового сигналу за допомогою зворотного ДПФ."""*

reconstructed\_signal = np.zeros\_like(t\_values, dtype=np.float64)

**for** k **in** range(N):

C\_k = dft\_result[k]

**for** n, t **in** enumerate(t\_values):

reconstructed\_signal[n] += np.real(C\_k \* np.exp(1j \* 2 \* np.pi \* k \* t / N))

**return** reconstructed\_signal

**def** get\_signal\_expression(dft\_result, N):

expression = **""**

**for** k, C\_k **in** enumerate(dft\_result):

amplitude = np.abs(C\_k)

phase = np.angle(C\_k)

expression += **f" + {**amplitude**:.2f} \* cos(2π \* {**k**} \* t / N + {**phase**:.2f})"**

**return** expression.strip(**" +"**)

*# 6. Функція для побудови графіку часової залежності*

**def** plot\_time\_domain\_signal(t\_values, reconstructed\_signal):

*"""Побудова графіку відтвореного сигналу у часовій області."""*

plt.figure(figsize=(10, 5))

plt.plot(t\_values, reconstructed\_signal, label=**"Відтворений сигнал"**)

plt.title(**"Часова залежність відтвореного сигналу"**)

plt.xlabel(**"Час (t)"**)

plt.ylabel(**"Амплітуда"**)

plt.grid(**True**)

plt.legend()

plt.show()

*# Частина 1: DFT для довільного сигналу*

**def** part\_one():

N = 10 + 12

*# Генеруємо довільний сигнал*

f = np.random.rand(N)

print(**f"Довільний сигнал: {**f**}"**)

*# Обчислення ДПФ із метриками*

dft\_result, elapsed\_time, mul\_count, add\_count = compute\_dft\_with\_metrics(f, N)

*# Виведення результатів*

print(**f"Час обчислення ДПФ: {**elapsed\_time**:.6f} секунд"**)

print(**f"Кількість операцій множення: {**mul\_count**}"**)

print(**f"Кількість операцій додавання: {**add\_count**}"**)

*# Побудова графіків*

plot\_spectrum(dft\_result, N)

*# Частина 2: Відтворення аналогового сигналу*

**def** part\_two():

N = 96 + 12

*# Генеруємо довільний сигнал з 8-розрядними двійковими числами (0 або 1)*

f = np.random.randint(0, 2, size=N)

print(**"\n\nЧастина 2\n"**)

print(**f"Довільний сигнал з 8-розрядними двійковими числами (0 або 1): {**f**}"**)

*# Обчислення ДПФ*

dft\_result = compute\_dft(f, N)

*# Визначаємо комплексні коефіцієнти DFT*

amplitudes = np.abs(dft\_result) *# |C\_n|*

phases = np.angle(dft\_result) *# arg(C\_n)*

*# Виведення результатів*

print(**f"Коефіцієнти DFT (C\_n): {**dft\_result**}"**)

print(**f"Амплітуди |C\_n|: {**amplitudes**}"**)

print(**f"Фази arg(C\_n): {**phases**}"**)

*# Запис виразу для s(t)*

t\_values = np.linspace(0, N, 1000) *# Часові значення для реконструкції*

reconstructed\_signal = reconstruct\_signal(dft\_result, N, t\_values)

signal\_expression = get\_signal\_expression(dft\_result, N)

print(**f"s(t) = {**signal\_expression**}"**)

*# Побудова графіку часової залежності відтвореного сигналу*

plot\_time\_domain\_signal(t\_values, reconstructed\_signal)

**def** main():

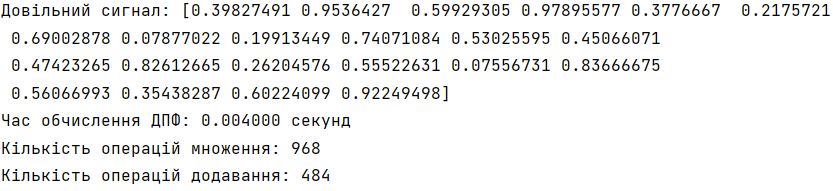
part\_one()

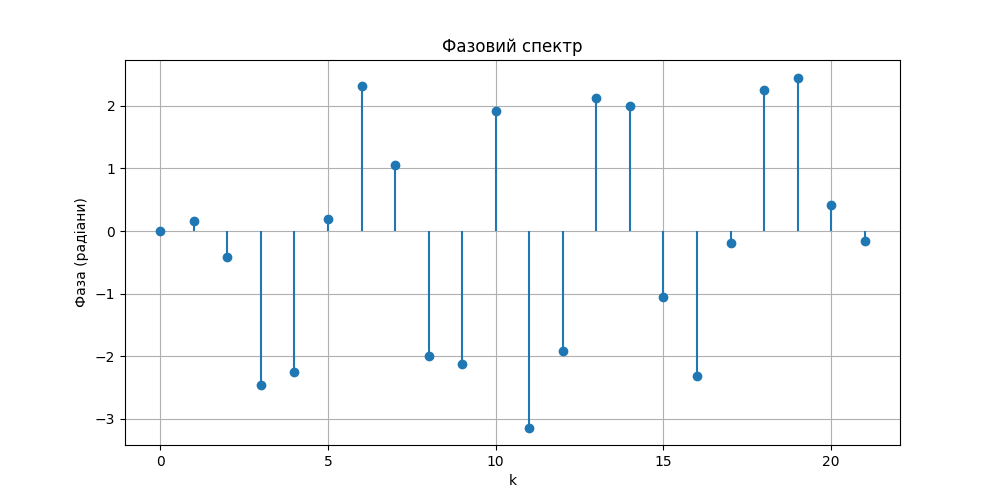
part\_two()

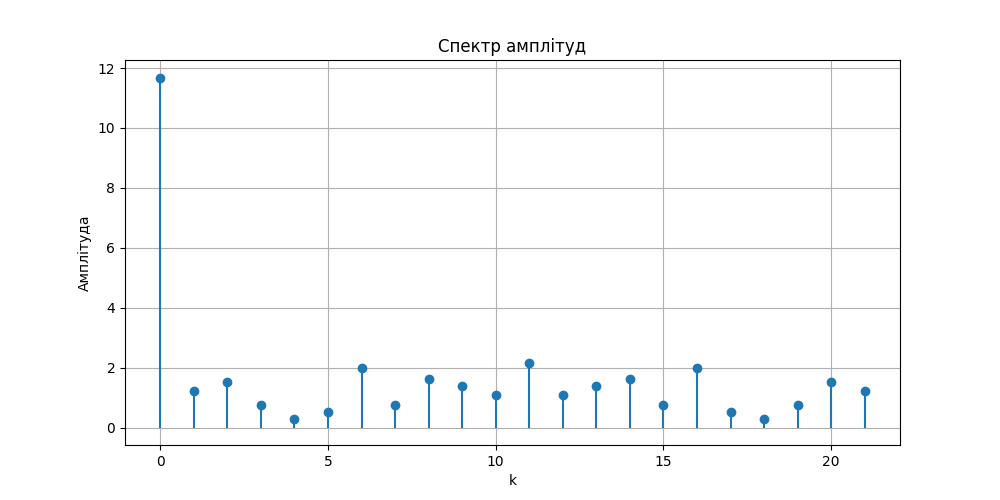
**if** \_\_name\_\_ == **"\_\_main\_\_"**:

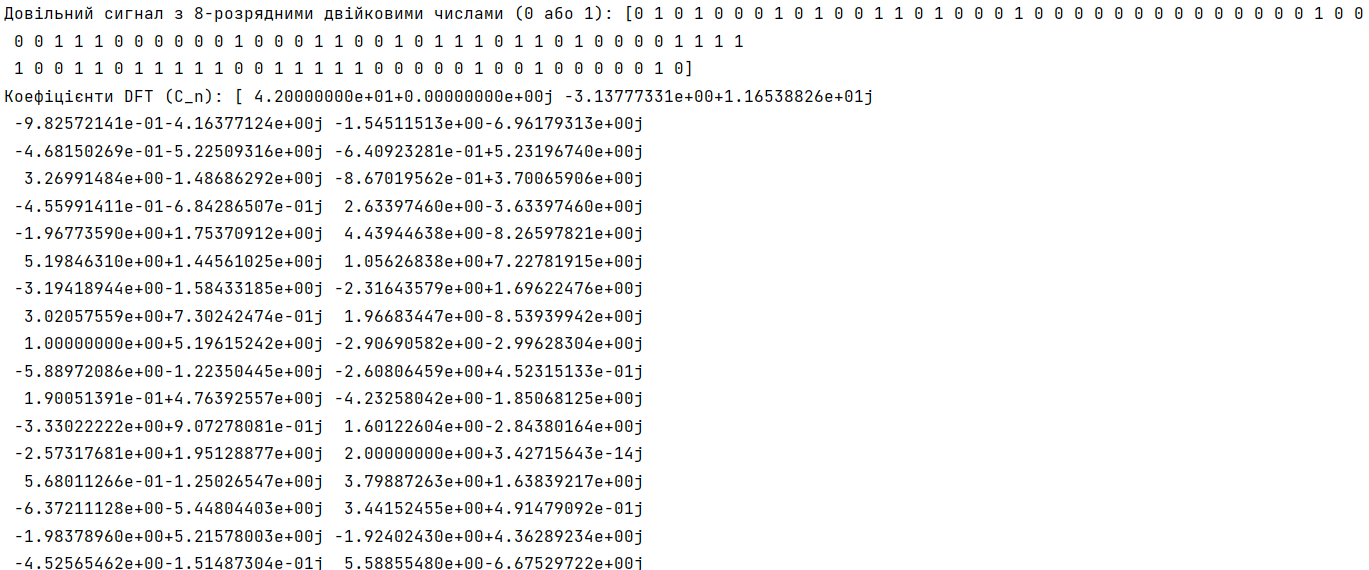
main()

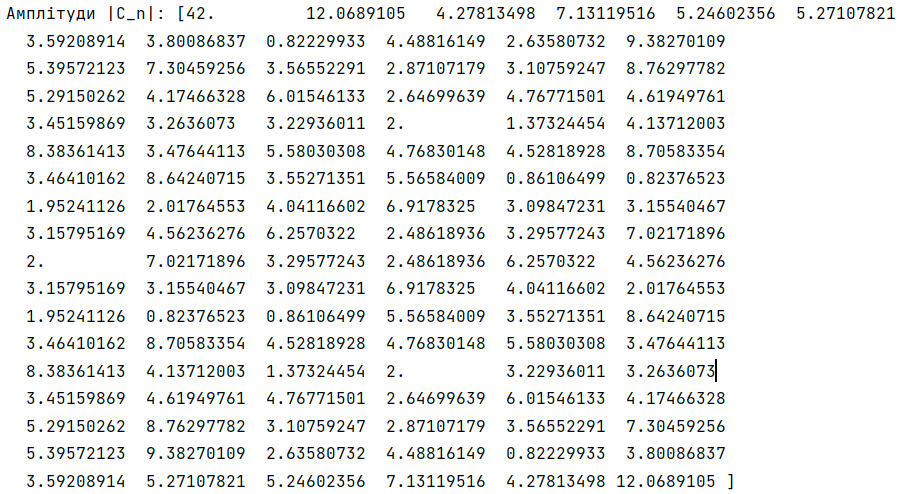
**Результат виконання:**

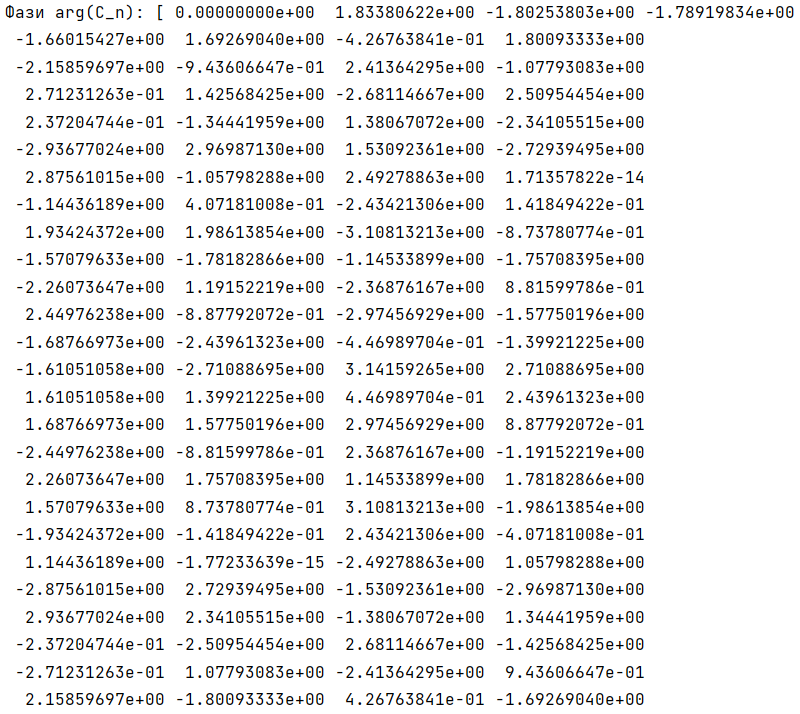
****

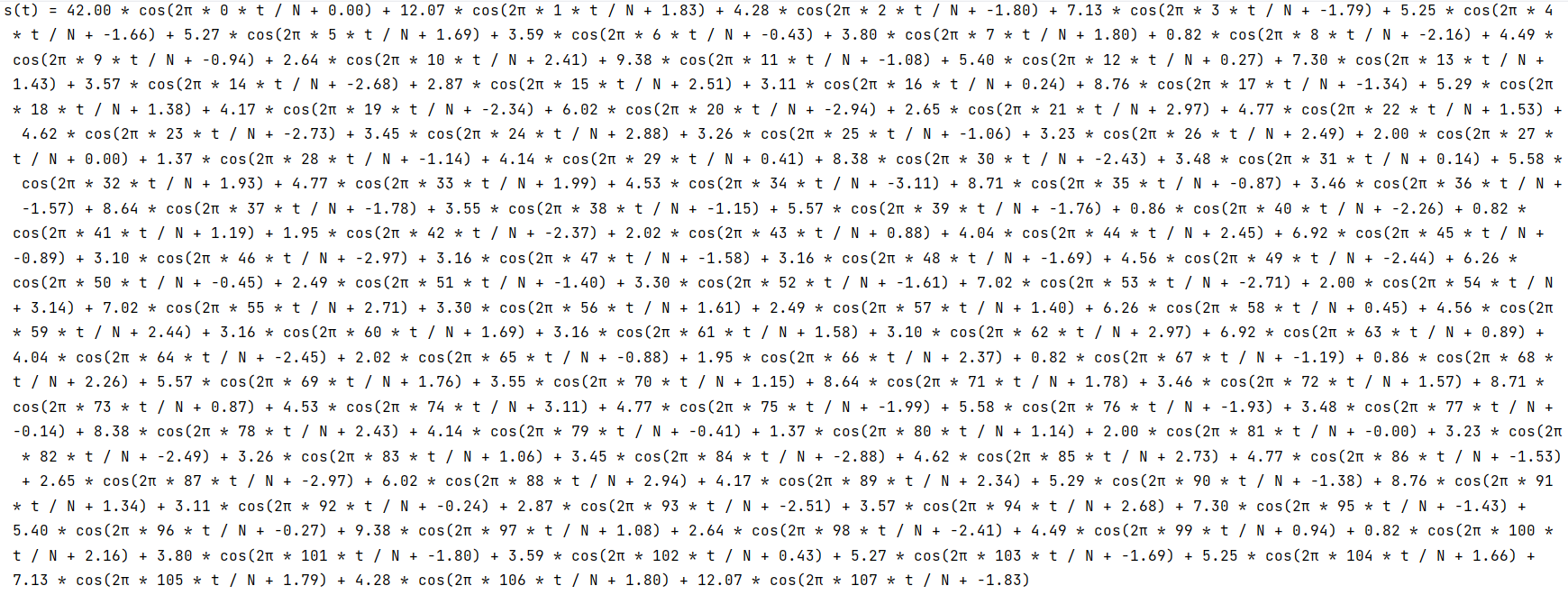
****

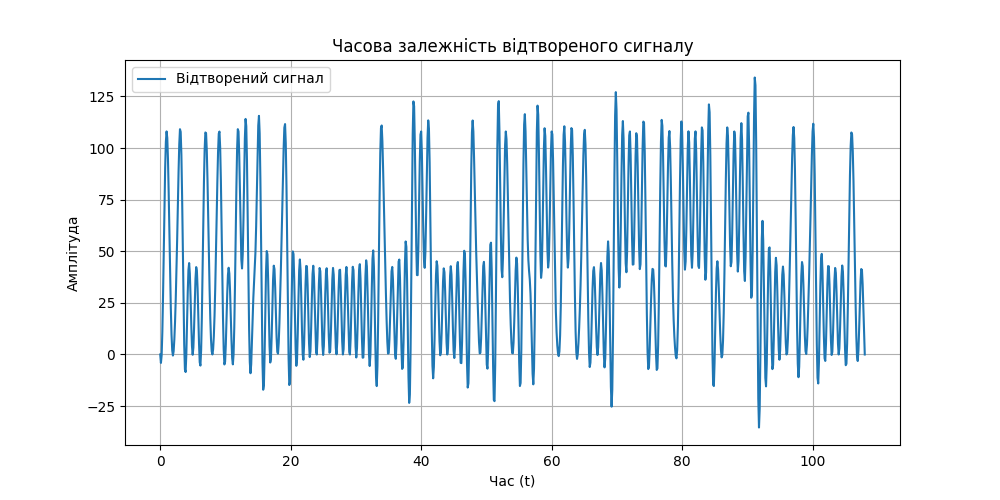
****

****

****

****

****

****

**Висновок:**

**https://github.com/DenysMuzykaPm/Signals**

У ході лабораторної роботи було реалізовано обчислення дискретного перетворення Фур'є (ДПФ) для довільного сигналу. Під час роботи ми виконали такі основні етапи:

1. **Генерація вхідного сигналу**: Було згенеровано довільні сигнали для різних значень NNN з використанням як випадкових чисел, так і двійкових відліків.
2. **Обчислення ДПФ**: Було реалізовано алгоритм обчислення ДПФ вручну, без використання бібліотечних функцій, що дозволило детально зрозуміти процеси перетворення сигналу з часової області в частотну.
3. **Побудова спектрів амплітуд і фаз**: Для вхідних сигналів були побудовані графіки амплітудного та фазового спектрів, що дозволило проаналізувати частотний склад сигналу.
4. **Відтворення аналогового сигналу**: За допомогою зворотного перетворення Фур'є було відтворено аналоговий сигнал, а також побудовано його часову залежність, що продемонструвало можливість відновлення сигналу на основі його частотних компонентів.
5. **Аналіз метрик**: Було проаналізовано час виконання алгоритму, а також кількість операцій множення та додавання, що показало ефективність розробленого алгоритму.

У результаті лабораторної роботи ми отримали практичний досвід обчислення та аналізу дискретного перетворення Фур'є. Це дало нам можливість зрозуміти, як можна використовувати частотний аналіз для представлення та обробки сигналів, а також відновлення їх із частотного спектра.