КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ
КАФЕДРА ТЕОРЕТИЧНОЇ КІБЕРНЕТИКИ

Самостійна робота
з дисципліни "Інтелектуальна обробка даних"

виконала:
студентка 4-го курсу
групи ТК-41
Квішко Олександра

Київ – 2024

Зміст

ВИЗНАЧЕННЯ МЕТИ ТА ОГОЛОШЕННЯ ЗАВДАННЯ

Мета: вивчення основних та найбільш перспективних напрямків аналізу даних: зберігання інформації, оперативний і інтелектуальний аналіз, а також методів та алгоритмів інтелектуального аналізу; знайомство з актуальними питаннями, що постають при розробці програмних продуктів, що обробляють великі обсяги даних. Протягом вивчення студенти мають опанувати основні методи та моделі інтелектуальної обробки та аналізу даних, засоби їх реалізації, навчитися аналізувати та уникати сучасних проблем, пов’язаних із збиранням та обробленням інформації.

Завдання. У файлі A3.txt міститься запис кардіограми людини по 12 каналах. Час запису – 10 сек. Дискретність: 500 точок за 1 сек. Структура файлу – 1-й канал, 2-й канал, … 12-й канал (амплітуда у відносних одиницях). Довжина запису N=5000, t =1/500 = 0.002

Алгоритм обробки

1. Попередній аналіз

Завдання: Побудувати графік кардіограми по кожному каналу. Для заданих змінних оцінити основні статистичні параметри (середнє, середнє гармонічне, середнє геометричне, дисперсію, середню різницю Джині, моду, медіану, коефіцієнт асиметрії, коефіцієнт ексцесу, побудувати гістограму, перевірити гіпотезу про закон нормальний закон розподілу). Нормалізувати дані по кожному стовпчику (математичне сподівання рівне нулю, дисперсія рівна 1).

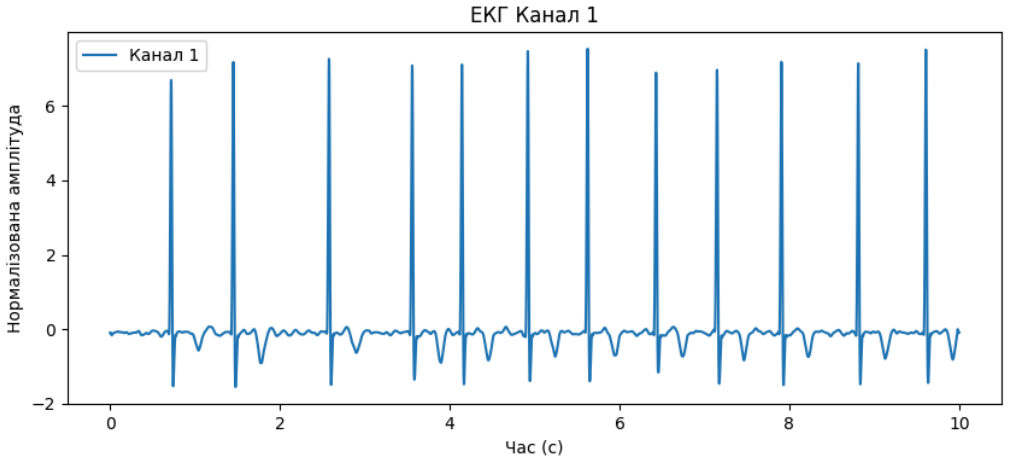
Код:

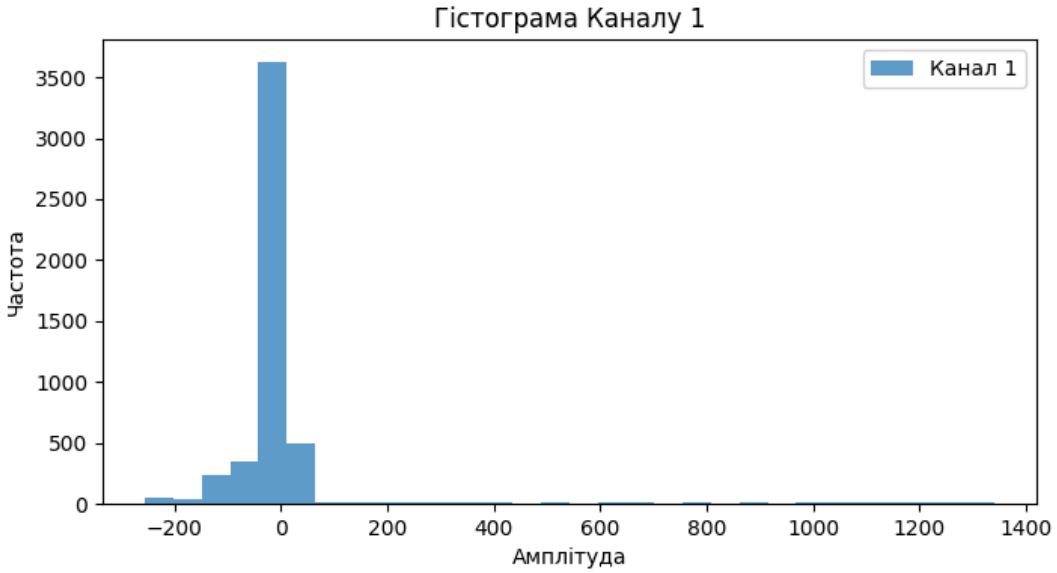
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import hmean, gmean, skew, kurtosis, shapiro

# Параметри завдання
N = 5000  # кількість точок у файлі
channels = 12  # кількість каналів
sampling_rate = 500  # частота дискретизації
time = np.linspace(0, N / sampling_rate, N)

# Завантаження даних з одного файлу
file = "A3.txt"
data = pd.read_csv(file, sep=",", header=None)

# Перевірка структури даних
print("Структура даних:")
print(data.head())  # Виводимо перші кілька рядків для перевірки

# Заміна ком на крапки для правильної інтерпретації чисел
data = data.apply(lambda x: x.map(lambda y: str(y).replace(",", ".")))

# Перетворення даних у числовий формат
data = data.apply(pd.to_numeric, errors='coerce')

# Видалення рядків з NaN (якщо такі є)
data = data.dropna()

# Перевірка розміру даних
print(f"Розмір даних: {data.shape}")

# Ініціалізація списку для збереження статистичних параметрів кожного каналу
stats_summary = []

# Обробка кожного каналу
```

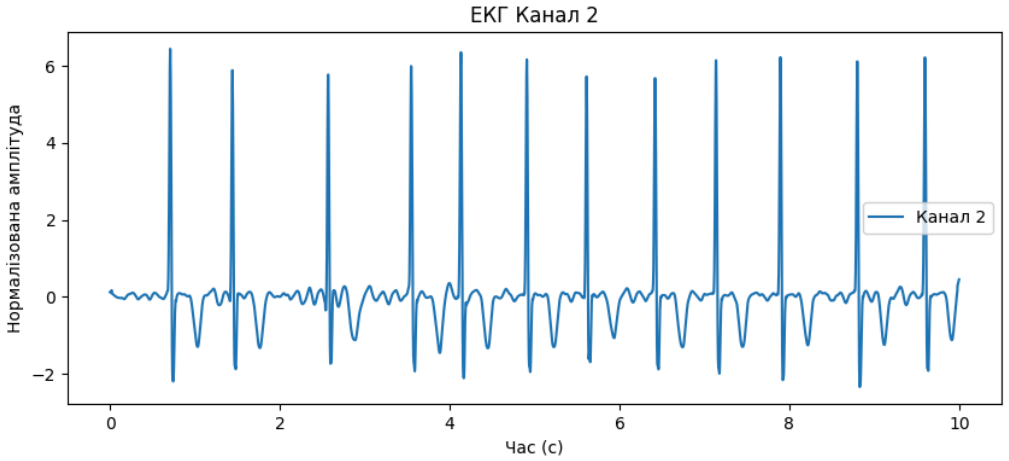
```
for channel in range(channels):
    if channel >= data.shape[1]:
        print(f"Канал {channel + 1} відсутній у даних.")
        continue

    channel_data = data[channel]

    # Перевірка, чи не порожні дані
    if channel_data.empty:
        print(f"Канал {channel + 1} порожній, пропускаємо.")
        continue

    # Розрахунок статистичних параметрів
    mean_value = channel_data.mean()
    hmean_value = hmean(channel_data[channel_data > 0]) if (
            channel_data[channel_data > 0].size > 0) else np.nan  # Гармонійне середнє (без нулів)
    gmean_value = gmean(channel_data[channel_data > 0]) if (
            channel_data[channel_data > 0].size > 0) else np.nan  # Геометричне середнє
    variance_value = channel_data.var()
    std_dev_value = channel_data.std()  # Середньоквадратичне відхилення
    gini_diff = channel_data.diff().abs().mean()

    # Обробка моди з використанням pandas .mode() з перевіркою на наявність значень
    most_frequent_value = channel_data.mode().iloc[0] if not channel_data.mode().empty else np.nan

    # Знаходимо медіану
    median_value = channel_data.median()
    min_median_value = channel_data[channel_data == channel_data.min()].median()  # Найменша медіана
    skewness_value = skew(channel_data) if len(channel_data) > 1 else np.nan  # Перевірка на наявність даних
для розрахунку асиметрії
    kurtosis_value = kurtosis(channel_data) if len(channel_data) > 1 else np.nan  # Перевірка на наявність
даних для розрахунку ексцесу

    # Додавання статистики каналу у список
    stats_summary.append({
        'Канал': channel + 1,
        'Середнє': mean_value,
        'Гармонійне середнє': hmean_value,
        'Геометричне середнє': gmean_value,
        'Дисперсія': variance_value,
        'Середнє квадратичне відхилення': std_dev_value,
        'Найменша медіана': min_median_value,
        'Мода': most_frequent_value,
        'Медіана': median_value,
        'Асиметрія': skewness_value,
        'Ексцес': kurtosis_value
    })

    # Перевірка на нормальність за допомогою тесту Шапіро-Вілка
    stat, p_value = shapiro(channel_data)
    stats_summary[-1]['p-значення нормальності'] = p_value
```

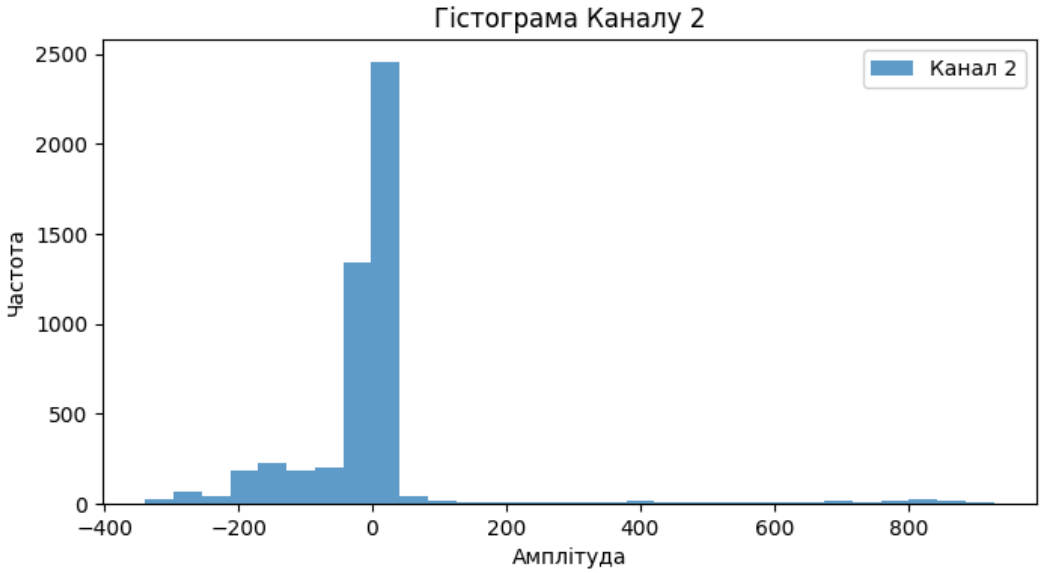
```
    # Нормалізація даних для кожного каналу
    normalized_data = (channel_data - mean_value) / np.std(channel_data)
    data[channel] = normalized_data

    # Побудова графіка для кожного каналу
    plt.figure(figsize=(10, 4))
    plt.plot(time, normalized_data[:N], label=f"Канал {channel + 1}")
    plt.xlabel("Час (с)")
    plt.ylabel("Нормалізована амплітуда")
    plt.title(f"ЕКГ Канал {channel + 1}")
    plt.legend()
    plt.show()

    # Побудова гістограми даних
    plt.figure(figsize=(8, 4))
    plt.hist(channel_data, bins=30, alpha=0.7, label=f"Канал {channel + 1}")
    plt.xlabel("Амплітуда")
    plt.ylabel("Частота")
    plt.title(f"Гістограма Каналу {channel + 1}")
    plt.legend()
    plt.show()

# Створення DataFrame зі статистиками для всіх каналів
df_stats = pd.DataFrame(stats_summary)

# Збереження таблиці в Excel
output_file = "EKG_Statistics.xlsx"
df_stats.to_excel(output_file, index=False)

# Виведення повідомлення про успішне збереження
print(f"Таблиця статистики збережена в {output_file}")

# Побудова графіка перших 6 каналів
plt.figure(figsize=(12, 6))
for i in range(6):
    if i < data.shape[1]:
        plt.plot(time, data[i][:N], label=f"Канал {i + 1}")
    else:
        print(f"Канал {i + 1} відсутній у даних.")
plt.xlabel("Час (с)")
plt.ylabel("Нормалізована амплітуда")
plt.title("ЕКГ - Перші 6 Каналів")
plt.legend()
plt.show()
```

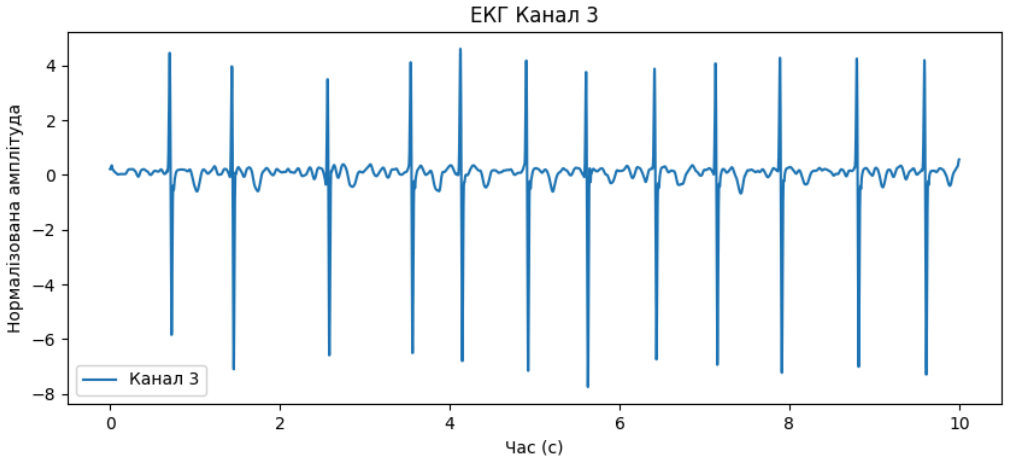
Результат:

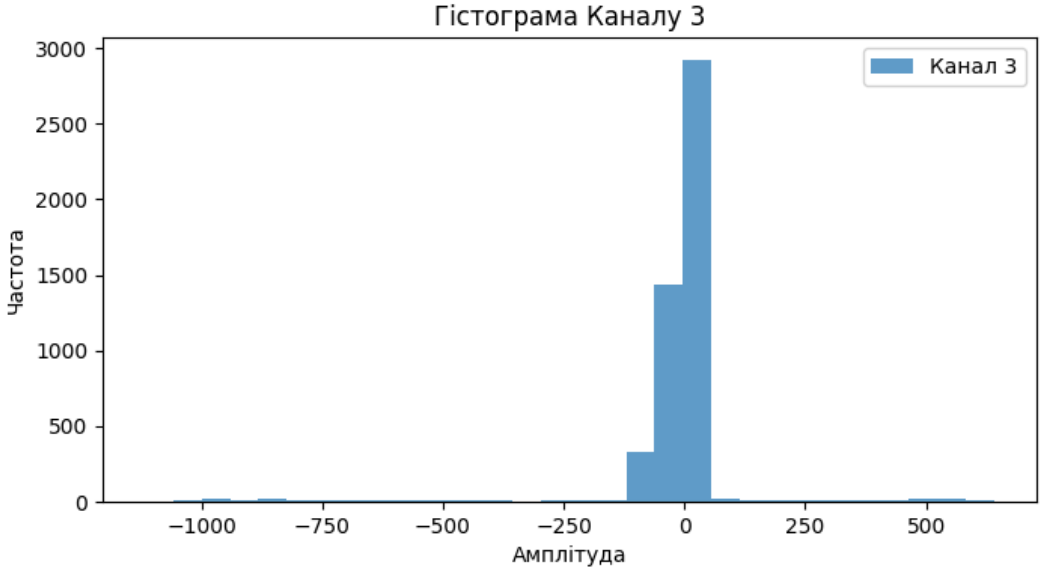

ЕКГ Канал 1
Канал 1
Нормалізована амплітуда
Час (с)

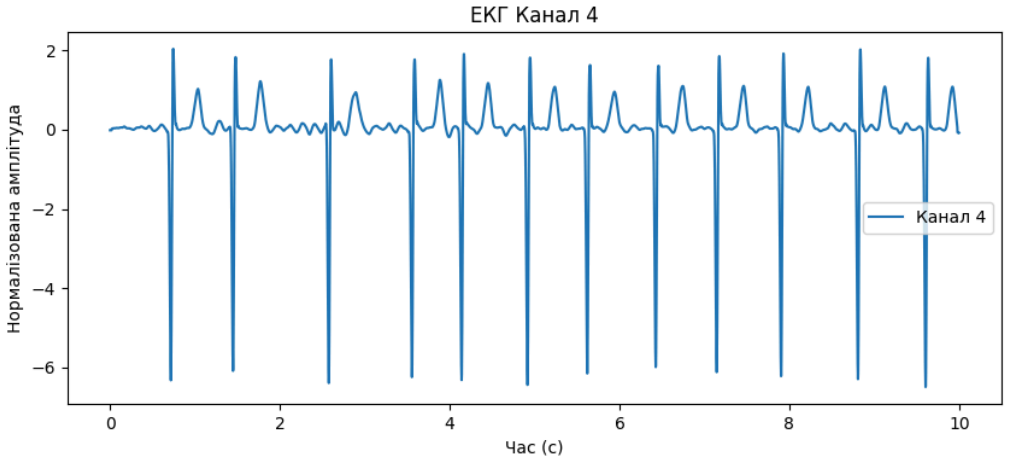

Гістограма Каналу 1
Канал 1
Частота
Амплітуда

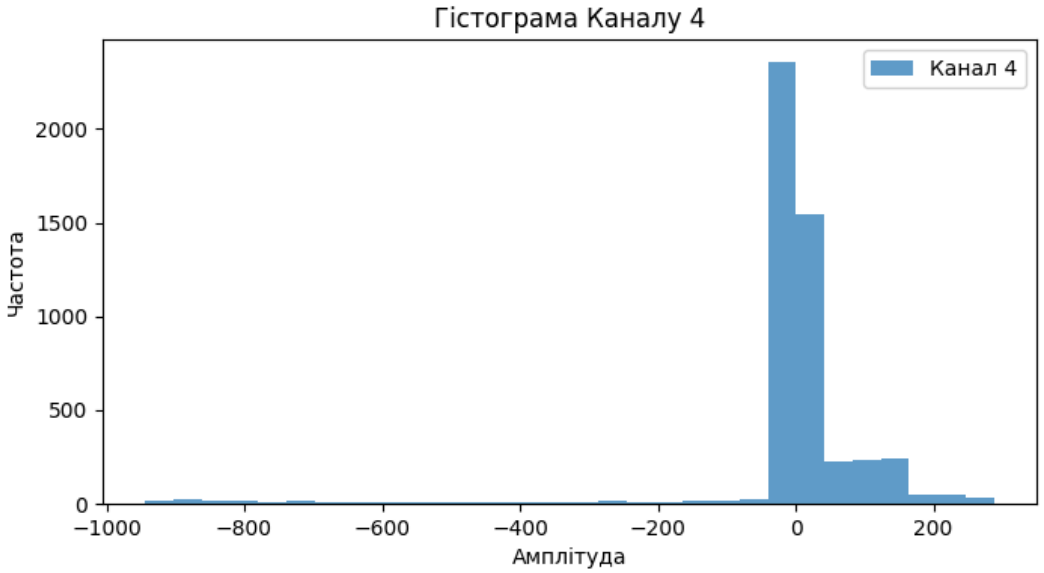

ЕКГ Канал 2
Канал 2
Нормалізована амплітуда
Час (с)

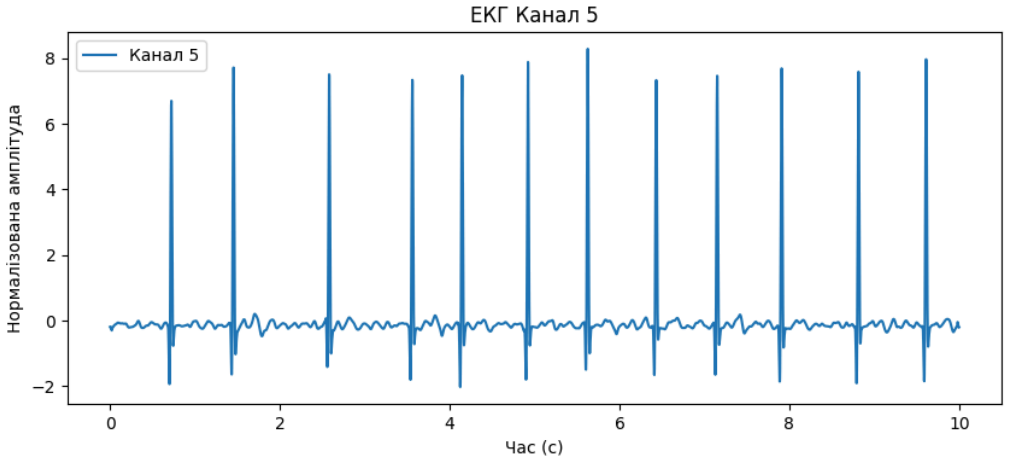

Гістограма Каналу 2
Канал 2
Частота
Амплітуда
2500
2000
1500
1000
500
0
−400
−200
0
200
400
600
800

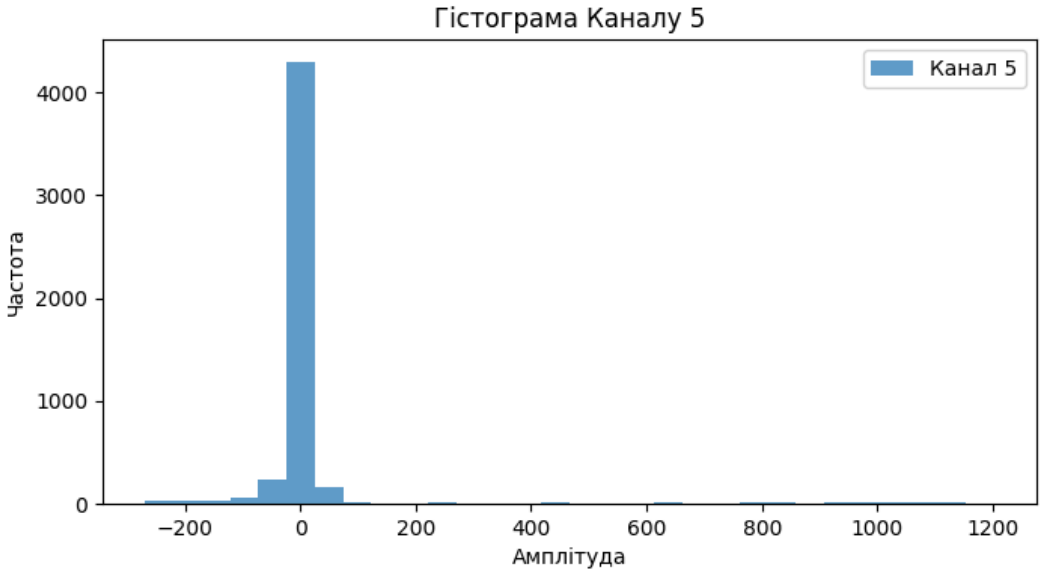

ЕКГ Канал 3
Нормалізована амплітуда
Час (с)
Канал 3
4
2
0
−2
−4
−6
−8
0
2
4
6
8
10

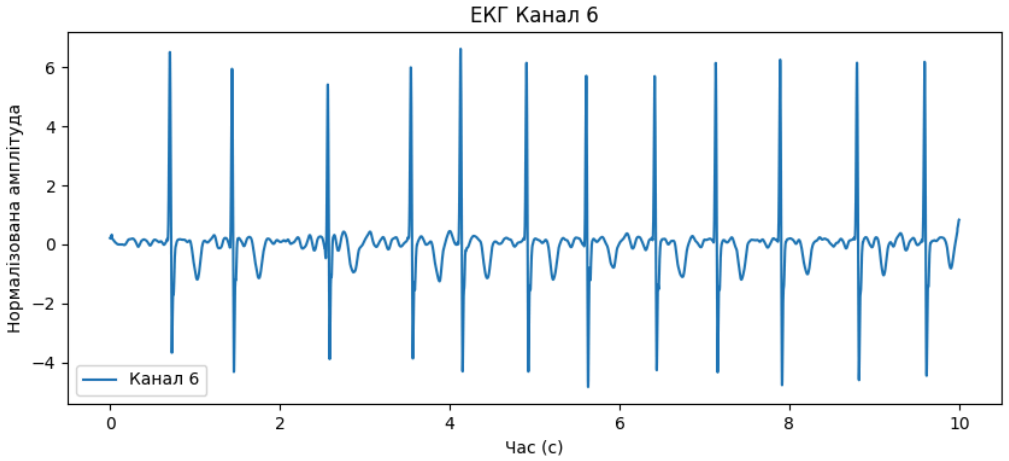

Гістограма Каналу 3
Канал 3
Частота
Амплітуда
3000
2500
2000
1500
1000
500
0
−1000
−750
−500
−250
0
250
500

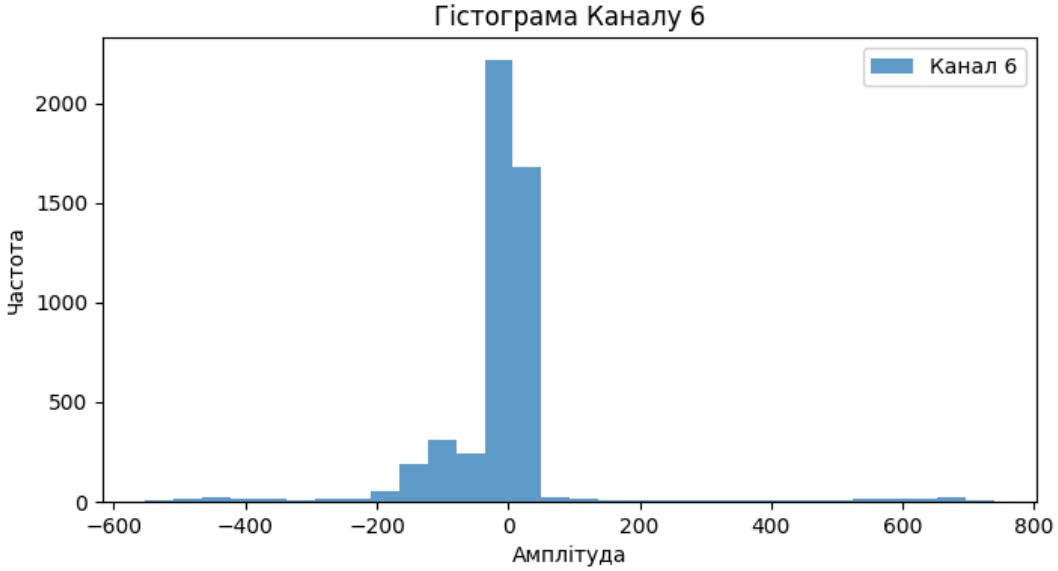

ЕКГ Канал 4
Нормалізована амплітуда
Час (с)
Канал 4

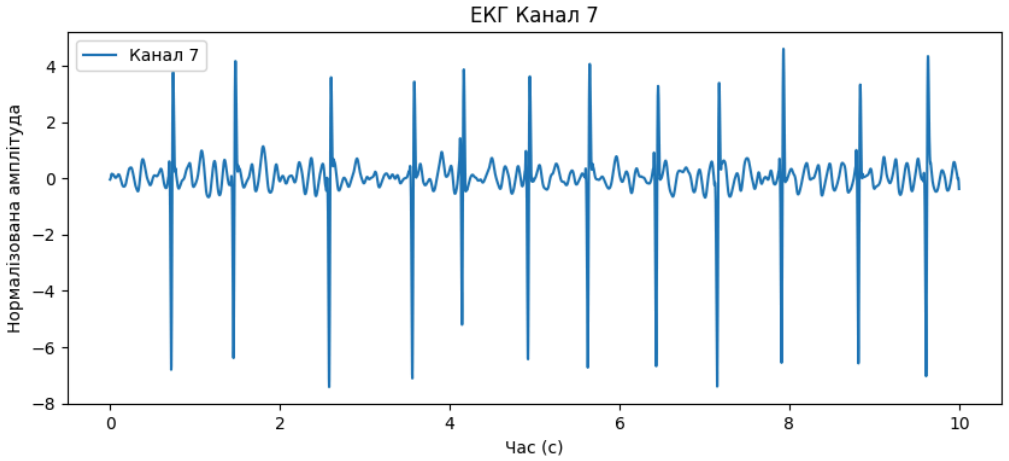

Гістограма Каналу 4
Частота
Амплітуда
Канал 4

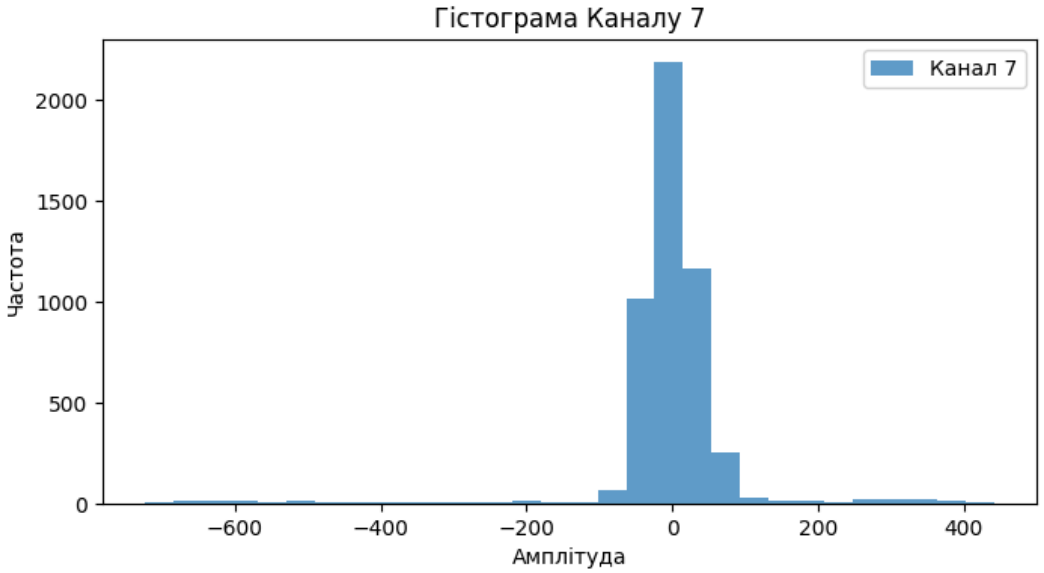

ЕКГ Канал 5
Нормалізована амплітуда
Час (с)
Канал 5

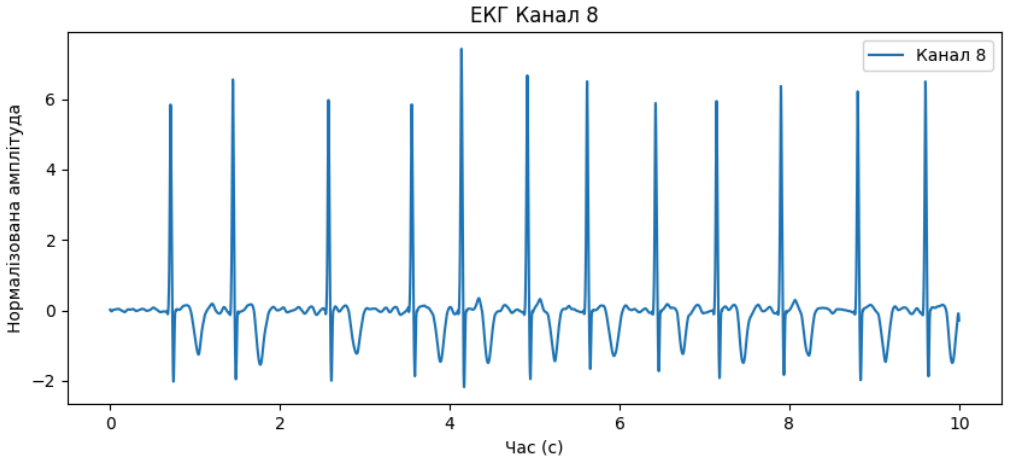

Гістограма Каналу 5
Канал 5
Частота
Амплітуда
4000
3000
2000
1000
0
−200
0
200
400
600
800
1000
1200


ЕКГ Канал 6
Нормалізована амплітуда
Канал 6
Час (с)
6
4
2
0
−2
−4
0
2
4
6
8
10

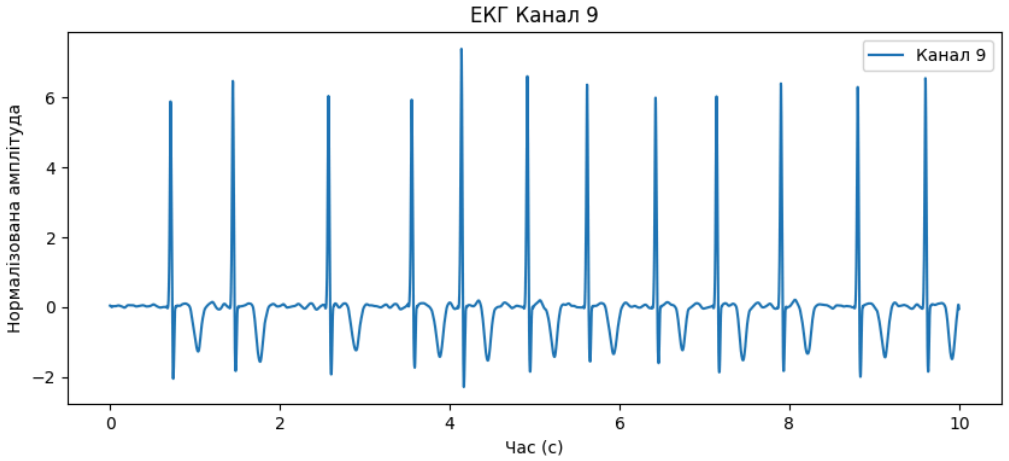

Гістограма Каналу 6
Канал 6
Частота
Амплітуда
2000
1500
1000
500
0
−600
−400
−200
0
200
400
600
800

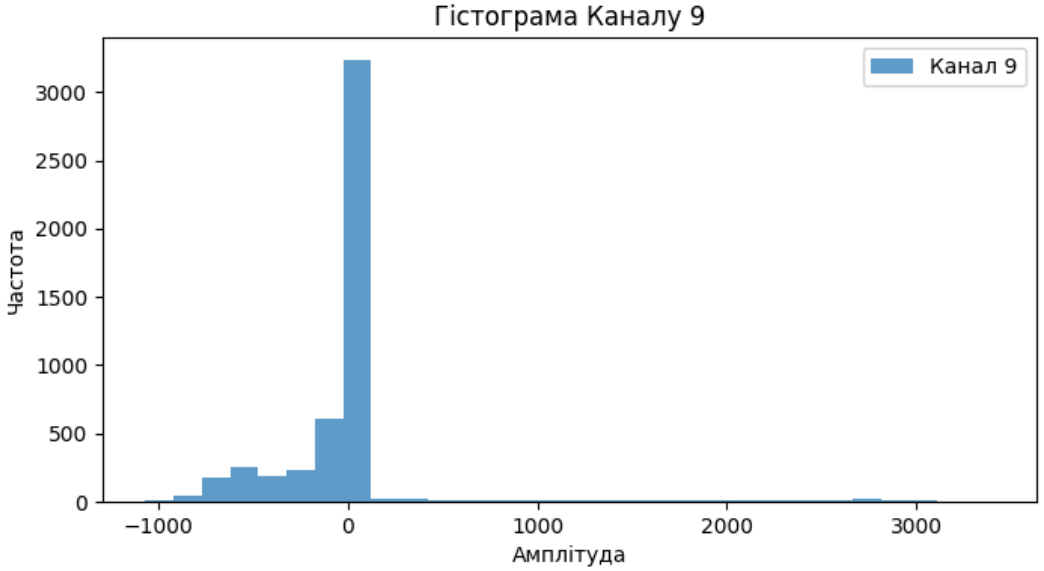

ЕКГ Канал 7
Канал 7
Нормалізована амплітуда
Час (с)

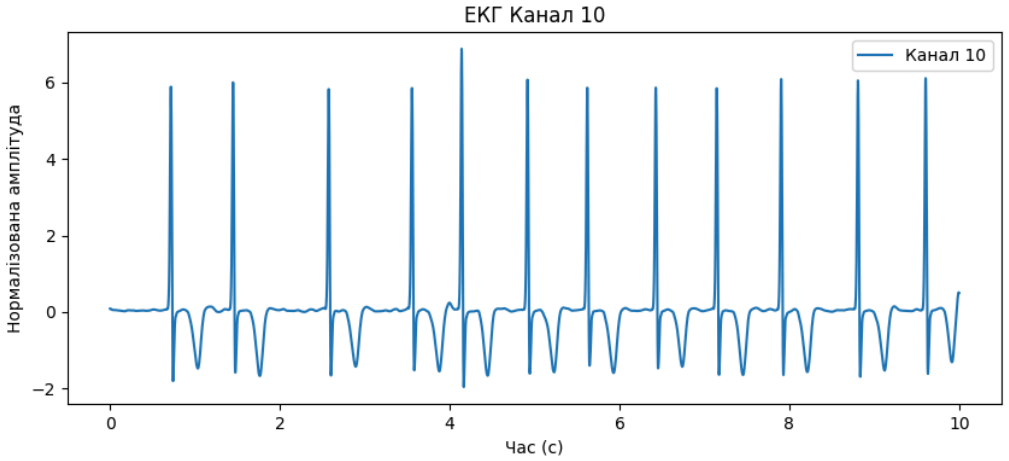

Гістограма Каналу 7
Канал 7
Частота
Амплітуда

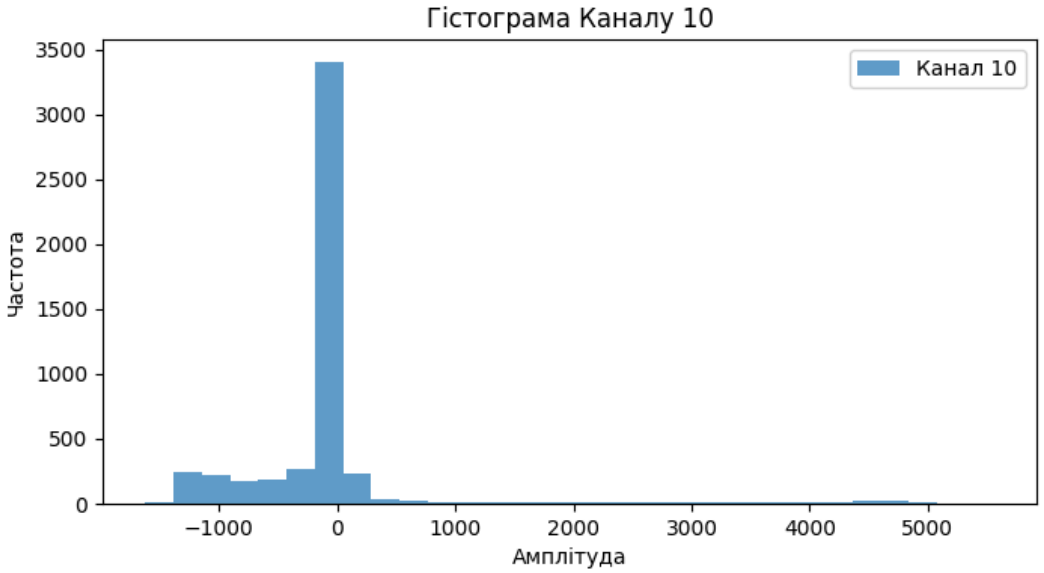

ЕКГ Канал 8
Канал 8
Нормалізована амплітуда
Час (с)

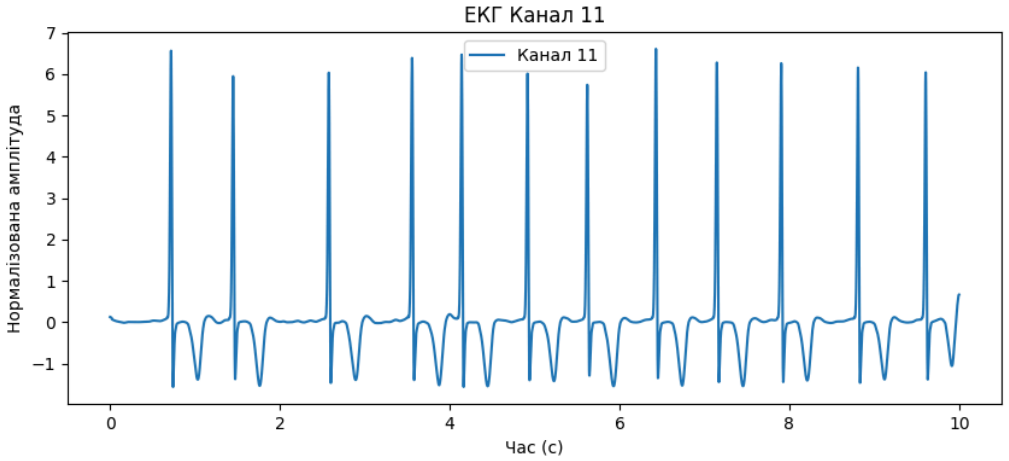

Гістограма Каналу 8
Канал 8
Частота
Амплітуда
2500
2000
1500
1000
500
0
−1000
−500
0
500
1000
1500
2000
2500
3000

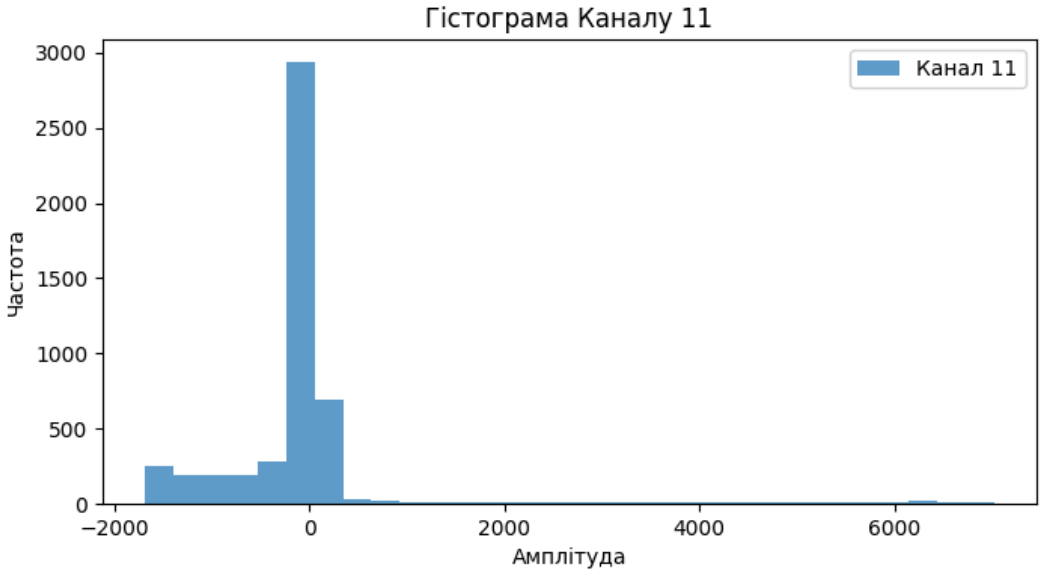

ЕКГ Канал 9
Канал 9
Нормалізована амплітуда
Час (с)
6
4
2
0
−2
0
2
4
6
8
10

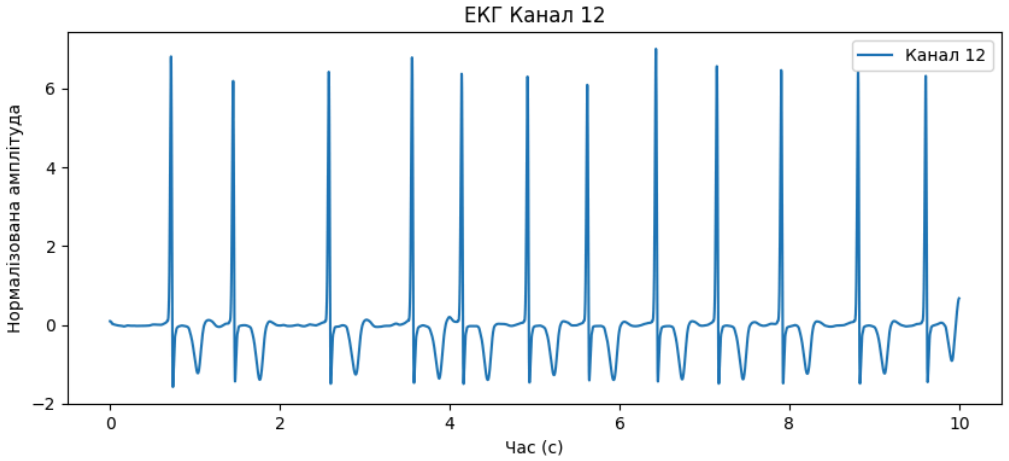

Гістограма Каналу 9
Канал 9
Частота
Амплітуда
3000
2500
2000
1500
1000
500
0
−1000
0
1000
2000
3000

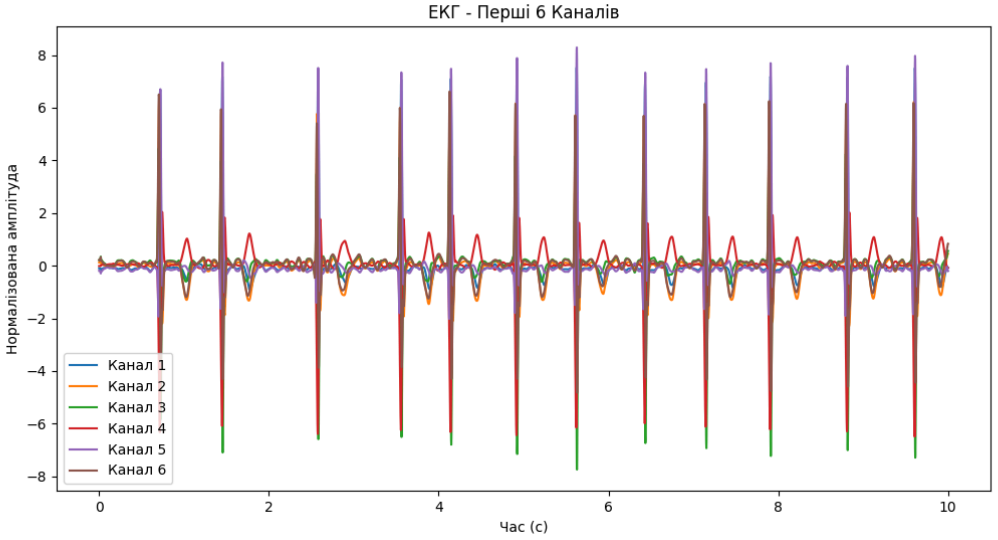

ЕКГ Канал 10
Канал 10
Нормалізована амплітуда
Час (с)


Гістограма Каналу 10
Канал 10
Частота
Амплітуда


ЕКГ Канал 11
Канал 11
Нормалізована амплітуда
Час (с)


Гістограма Каналу 11
Канал 11
Частота
Амплітуда
3000
2500
2000
1500
1000
500
0
−2000
0
2000
4000
6000


ЕКГ Канал 12
Канал 12
Нормалізована амплітуда
Час (с)
6
4
2
0
−2
0
2
4
6
8
10



Вивід:

Структура даних:

	0	1	2	3	...	8	9	10	11
0	0.70979	16.157	16.278	-8.4604	...	8.2307	38.686	114.69	91.211
1	1.02570	15.684	15.475	-8.3826	...	7.4234	39.779	113.46	90.818
2	1.29450	15.330	14.837	-8.3406	...	6.2299	40.222	112.02	89.733
3	1.42430	15.247	14.610	-8.3645	...	4.4093	39.536	110.23	87.490
4	1.26990	15.613	15.116	-8.4709	...	1.9392	37.540	108.12	83.951

[5 rows x 12 columns]

Розмір даних: (5000, 12)

Таблиця статистики збережена в EKG_Statistics.xlsx

Таблиця:

Канал	Середнє	Гарм. середнє	Геом. середнє	Дисперсія	Сер. кв. відхилення	Найменша медіана	Мода	Медіана	Асиметрія	Ексцес	p-знач. норм.
1	17,31012639	1,430243765	7,290921573	30870,07815	175,698828	-255,05	-242,21	0,230665	5,382026587	31,32267734	0
2	-1,596182662	1,717715038	11,85818305	20802,36143	144,2302376	-339,04	10,058	1,06945	3,536346354	18,10694638	0
3	-14,20829142	1,765196293	11,37538897	20279,28947	142,4053702	-1116,6	12,485	3,2317	-3,434591052	25,90081412	0
4	-6,672493454	1,808913279	14,10168458	20857,17876	144,4201467	-943,9	-4,2486	-0,233955	-4,167103443	21,06159679	0
5	19,20288493	2,142150296	10,25528015	20378,37462	142,7528445	-269,22	-19,435	-0,330345	5,862274635	36,79959166	0
6	-8,084656272	3,780127532	11,02081674	12754,04354	112,9338016	-552,5	10,977	1,40195	1,891511665	16,85747999	0
7	-4,642444054	3,674485512	19,31908465	9399,591203	96,95148891	-723,2	-11,658	-1,29775	-2,858090781	21,93937991	0
8	-1,815386454	3,405346192	26,28551946	144751,8397	380,4626654	-828,64	-558,49	-0,36307	3,740642905	19,59110232	0
9	-10,13567158	3,597466758	21,44965806	213604,2597	462,1734087	-1068,6	11,215	-0,253335	3,746963748	19,83107468	0
10	-32,9769236	3,512561247	27,36832276	662601,5065	814,0033823	-1624,7	-16,952	-3,2939	3,533425018	17,73430062	0
11	-19,51645065	6,258396521	58,68690698	1134887,804	1065,31113	-1689,1	-1659,2	-13,6245	3,844260367	19,59727607	0
12	5,304993271	6,222020566	56,0629169	807049,0361	898,3590797	-1406,5	-17,696	-11,8625	4,215815086	22,1674676	0

2. Однофакторний дисперсійний аналіз.

Код:

```
import numpy as np
import pandas as pd
from scipy import stats

# Завантажуємо дані з файлу
data = np.genfromtxt('A3.txt', delimiter=',')
data = data.reshape(5000, 12)  # Перетворюємо дані в форму (5000 точок, 12 каналів)

# Параметри
n = 5000  # Кількість точок
k = 12    # Кількість каналів
alpha = 0.05  # Рівень значущості

# 1. Обчислюємо середнє значення для кожного каналу
means = np.mean(data, axis=0)

# 2. Обчислюємо дисперсії для кожного каналу
S_squared_i = np.var(data, axis=0, ddof=1)

# 3. Перевірка рівності дисперсій:
# Тест Левене на рівність дисперсій
_, p_value = stats.levene(*[data[:, i] for i in range(k)])

print(f"\nP-значення для тесту Левене (перевірка рівності дисперсій): {p_value}")

if p_value > alpha:
  print("Гіпотеза про рівність дисперсій не відкидається.")
else:
  print("Гіпотеза про рівність дисперсій відкидається.")

# 4. Перевірка, чи є статистично значущим вплив каналу (фактора) за допомогою F-статистики

# Загальна дисперсія S_0^2
S_0_squared = np.var(data.flatten(), ddof=k-1)

# Дисперсія, пов'язана з фактором (S_A^2)
mean_all = np.mean(data)  # Загальне середнє по всіх даних
```

```
S_A_squared = (n / (k - 1)) * np.sum((means - mean_all) ** 2)

# F-статистика
F_statistic = S_A_squared / S_0_squared

# Критичне значення F для заданого рівня значущості та ступенів свободи
F_critical = stats.f.ppf(1 - alpha, k - 1, k * (n - 1))

print(f"\nF-статистика: {F_statistic}")
print(f"Критичне значення F: {F_critical}")

if F_statistic > F_critical:
  print("Вплив канала (фактора) статистично значущий")
else:
  print("Вплив канала (фактора) не статистично значущий")

# 5. Виводимо результати (середні значення та дисперсії для кожного каналу)
results = pd.DataFrame({
  'Канал': [f'Канал {i+1}' for i in range(k)],
  'Середнє значення': means,
  'Дисперсія': S_squared_i
})

print("\nРезультати для кожного каналу:")
print(results)

# 6. Зберігаємо результати в Excel
results.to_excel('dispersions_analysis_results.xlsx', index=False)
```

Вивід:
P-значення для тесту Левене (перевірка рівності дисперсій): 0.0
Гіпотеза про рівність дисперсій відкидається.

F-статистика: 4.095675094588742
Критичне значення F: 1.7888080235820565
Вплив канала (фактора) статистично значущий

Результати для кожного каналу:

	Канал	Середнє значення	Дисперсія
0	Канал 1	17.310126	3.087008e+04
1	Канал 2	-1.596183	2.080236e+04
2	Канал 3	-14.208291	2.027929e+04
3	Канал 4	-6.672493	2.085718e+04
4	Канал 5	19.202885	2.037837e+04
5	Канал 6	-8.084656	1.275404e+04

6	Канал 7	-4.642444	9.399591e+03
7	Канал 8	-1.815386	1.447518e+05
8	Канал 9	-10.135672	2.136043e+05
9	Канал 10	-32.976924	6.626015e+05
10	Канал 11	-19.516451	1.134888e+06
11	Канал 12	5.304993	8.070490e+05

Process finished with exit code 0

Таблиця:

Канал	Середнє значення	Дисперсія
Канал 1	17,31012639	30870,07815
Канал 2	-1,596182662	20802,36143
Канал 3	-14,20829142	20279,28947
Канал 4	-6,672493454	20857,17876
Канал 5	19,20288493	20378,37462
Канал 6	-8,084656272	12754,04354
Канал 7	-4,642444054	9399,591203
Канал 8	-1,815386454	144751,8397
Канал 9	-10,13567158	213604,2597
Канал 10	-32,9769236	662601,5065
Канал 11	-19,51645065	1134887,804
Канал 12	5,304993271	807049,0361

3. Двофакторний аналіз.

Код:

```
import numpy as np
import pandas as pd
from scipy.stats import f

# Завантаження даних із файлу
data = np.genfromtxt('A3.txt', delimiter=',')
data = data.reshape(5000, 12)

# Параметри задачі
n_channels = 12  # кількість каналів (A)
n_parts = 5      # кількість частин у кожному каналі (B)
n_points = 1000  # кількість значень у кожній частині

# Розбиваємо дані на канали і частини та обчислюємо середні значення для кожної клітинки
channels = data.T  # Транспонуємо, щоб кожен канал був окремим рядком (12 x 5000)
table = [[np.mean(channels[i][j * n_points:(j + 1) * n_points]) for j in range(n_parts)] for i in range(n_channels)]

# Створення DataFrame для експорту в Excel із середніми значеннями
```

```
rows = []
for j in range(n_parts):
    row = {f'A{i + 1}': table[i][j] for i in range(n_channels)}
    rows.append(row)

# Перетворюємо список рядків у DataFrame
df = pd.DataFrame(rows)

# Перейменовуємо індекси рядків, щоб відобразити рівні фактора B
df.index = [f'B{j + 1}' for j in range(n_parts)]

# Зберігаємо таблицю з середніми значеннями в Excel
df.to_excel("кардіограма_середні_значення.xlsx", index=True)

# === Розрахунок статистичних показників ===

# Обчислюємо середні значення x_ij для кожної клітинки (вже обчислені раніше)
xij = np.array(table)

# Основні показники
Q1 = np.sum(xij**2)
Q2 = (1 / n_parts) * np.sum(np.sum(xij, axis=1)**2)
Q3 = (1 / n_channels) * np.sum(np.sum(xij, axis=0)**2)
Q4 = (1 / (n_channels * n_parts)) * (np.sum(xij)**2)

# Оцінка дисперсій
S2_0 = (Q1 + Q4 - Q2 - Q3) / ((n_channels - 1) * (n_parts - 1))
S2_A = (Q2 - Q4) / (n_channels - 1)
S2_B = (Q3 - Q4) / (n_parts - 1)

# Виведення оцінок дисперсій
print(f"Оцінка дисперсії S^2_0: {S2_0}")
print(f"Оцінка дисперсії S^2_A: {S2_A}")
print(f"Оцінка дисперсії S^2_B: {S2_B}")

# Перевірка значущості факторів за допомогою критерію Фішера
alpha = 0.05  # рівень значущості
F_a_A = f.ppf(1 - alpha, dfn=n_channels - 1, dfd=(n_channels - 1) * (n_parts - 1))
F_a_B = f.ppf(1 - alpha, dfn=n_parts - 1, dfd=(n_channels - 1) * (n_parts - 1))

is_A_significant = (S2_A / S2_0) > F_a_A
is_B_significant = (S2_B / S2_0) > F_a_B

print("Значущість фактора A:", "Значущий" if is_A_significant else "Не значущий")
print("Значущість фактора B:", "Значущий" if is_B_significant else "Не значущий")

# Якщо фактори A і B залежні, додаємо розрахунок Q5 і S^2_AB
if is_A_significant and is_B_significant:
    Q5 = np.sum([np.sum([np.sum(channels[i][j * n_points:(j + 1) * n_points]**2) for j in range(n_parts)]) for i in
range(n_channels)])
    S2_AB = (Q5 - n_points * Q1) / (n_channels * n_parts * (n_points - 1))
```

```
# Перевірка взаємодії факторів
F_ab = (n_points * S2_0) / S2_AB
f1 = (n_channels - 1) * (n_parts - 1)
f2 = n_channels * n_parts * (n_points - 1)
F_crit_AB = f.ppf(1 - alpha, dfn=f1, dfd=f2)

is_AB_interaction_significant = F_ab > F_crit_AB
print("Взаємодія факторів A і B:", "Значуща" if is_AB_interaction_significant else "Не значуща")
```

Вивід:

Оцінка дисперсії S^2_0: 255.72966625213576
Оцінка дисперсії S^2_A: 1058.2242289224246
Оцінка дисперсії S^2_B: 1538.3496428512674
Значущість фактора A: Значущий
Значущість фактора B: Значущий
Взаємодія факторів A і B: Не значуща

Process finished with exit code 0

Таблиця:

	A1	A2	A3	A4	A5	A6
B1	13,2992148691	-1,713907949	-11,529367142	-4,507044319	14,4459652909	-6,683022076
B2	14,086441551	-2,189478962	-12,6455013483	-5,198143405	15,819136267	-7,303274253
B3	22,3712662047	-2,050125337	-17,831653278	-8,9988483313	23,972638738	-10,313645498
B4	25,105559766	2,9505048696	-16,8612666085	-13,2650007512	24,8356616004	-8,264848404
B5	11,688149558	-4,977905932	-12,17366874	-1,393430463	16,94102275	-7,858491131

	A7	A8	A9	A10	A11	A12
B1	-1,494176212	-4,067592556	-12,161857606	-34,004060363	-19,12449924	0,180792879000008
B2	-5,236227563	-8,982548903	-14,187596201	-30,510720967	-26,77853327	-1,45989934129998
B3	-3,217212492	4,7144099745	-8,100563328	-52,766177051	-50,99244312	-15,51350697
B4	-9,10688114	20,467196045	19,002257118	19,839550046	48,439455686	57,34482481

B5	-4,157722862	-21,208396828	-35,230597895	-67,44320966	-49,126233323	-14,027245023

4. Кореляційний аналіз

Код:

```
import numpy as np
import pandas as pd

# Завантаження даних
data = np.genfromtxt('A3.txt', delimiter=',')
data = data.reshape(5000, 12)  # Перетворення даних в форму (5000 рядків, 12 каналів)

# Нормалізація даних
mean = np.mean(data, axis=0)
std = np.std(data, axis=0)
normalized_data = (data - mean) / std

# Обчислення кореляційної матриці
correlation_matrix = np.corrcoef(normalized_data, rowvar=False)

# Перетворення в DataFrame для зручного відображення
correlation_df = pd.DataFrame(correlation_matrix,
                              columns=[f'Канал {i+1}' for i in range(12)],
                              index=[f'Канал {i+1}' for i in range(12)])

# Виведення таблиці
print("Кореляційна матриця:")
print(correlation_df)

# Збереження таблиці в файл Excel для звіту
correlation_df.to_excel('correlation_matrix.xlsx', index=True)

# Крок 3: Пошук пар з високою кореляцією (коефіцієнт > 0.9 або < -0.9)
high_corr_pairs = []
threshold = 0.9
for i in range(len(correlation_matrix)):
    for j in range(i):
        if abs(correlation_matrix[i, j]) > threshold:
            high_corr_pairs.append((f'Канал {i+1}', f'Канал {j+1}'))

print("\nПари з високою кореляцією:")
for pair in high_corr_pairs:
    print(pair)

# Функція для обчислення часткового коефіцієнта кореляції
def partial_corr(r_ab, r_ac, r_bc):
    return (r_ab - r_ac * r_bc) / np.sqrt((1 - r_ac**2) * (1 - r_bc**2))

# Крок 5: Часткові коефіцієнти кореляції
# 1. Між ознаками a та c без урахування впливу ознаки b
a, b, c = 0, 1, 2  # Канали для аналізу (a, b, c)
```

```
r_ab = correlation_matrix[a, b]
r_ac = correlation_matrix[a, c]
r_bc = correlation_matrix[b, c]

r_ac_b = partial_corr(r_ac, r_ab, r_bc)
print(f"\nЧастковий коефіцієнт кореляції між Каналом {a+1} та Каналом {c+1}, без урахування Канала {b+1}: {r_ac_b}")

# 2. Між ознаками a та b без урахування впливу ознак c та d
d = 3  # Канал для d
r_ad = correlation_matrix[a, d]
r_bd = correlation_matrix[b, d]
r_cd = correlation_matrix[c, d]

r_ab_cd = partial_corr(r_ab, r_ac, r_bd)
print(f"\nЧастковий коефіцієнт кореляції між Каналом {a+1} та Каналом {b+1}, без урахування Каналів {c+1} та {d+1}: {r_ab_cd}")

# 3. Між ознаками a та c без урахування впливу ознак b та d
r_ac_bd = partial_corr(r_ac, r_ab, r_bd)
print(f"\nЧастковий коефіцієнт кореляції між Каналом {a+1} та Каналом {c+1}, без урахування Каналів {b+1} та {d+1}: {r_ac_bd}")

# 4. Між ознаками a та d без урахування впливу ознак b та c
r_ad_bc = partial_corr(r_ad, r_ac, r_bc)
print(f"\nЧастковий коефіцієнт кореляції між Каналом {a+1} та Каналом {d+1}, без урахування Каналів {b+1} та {c+1}: {r_ad_bc}")

# Крок 6: Множинний коефіцієнт кореляції для канала a, при лінійному двофакторному зв'язку з b та c
def multiple_corr(r_ab, r_ac, r_bc):
    return np.sqrt((r_ab**2 + r_ac**2 - 2 * r_ab * r_ac * r_bc) / (1 - r_bc**2))

r_a_bc = multiple_corr(r_ab, r_ac, r_bc)
print(f"\nМножинний коефіцієнт кореляції для Канала {a+1} з Каналом {b+1} та Каналом {c+1}: {r_a_bc}")

# Крок 7: Множинний коефіцієнт кореляції для канала a, при лінійному трифакторному зв'язку з b, c та d
def multiple_corr_3factors(r_ab, r_ac, r_ad, r_bc, r_bd, r_cd):
    return np.sqrt(1 - (1 - r_ab**2) * (1 - r_ac**2) * (1 - r_ad**2))

r_a_bcd = multiple_corr_3factors(r_ab, r_ac, correlation_matrix[0, 3], r_bc, correlation_matrix[1, 3], correlation_matrix[2, 3])
print(f"\nТрифакторний множинний коефіцієнт кореляції для Канала {a+1}, Канала {b+1}, Канала {c+1} та Канала {d+1}: {r_a_bcd}")

# Крок 8: Визначення незалежних параметрів
independent_params = []
for col in range(data.shape[1]):
    if all(abs(correlation_matrix[col, other_col]) < 0.3 for other_col in range(data.shape[1]) if other_col != col):
        independent_params.append(f'Канал {col+1}')

print("\nНезалежні канали (кореляція з іншими факторами < 0.3):")
```

```
print(independent_params)
```

Вивід:
Пари з високою кореляцією:
('Канал 4', 'Канал 1')
('Канал 5', 'Канал 1')
('Канал 8', 'Канал 4')
('Канал 9', 'Канал 4')
('Канал 9', 'Канал 8')
('Канал 10', 'Канал 1')
('Канал 10', 'Канал 4')
('Канал 10', 'Канал 8')
('Канал 10', 'Канал 9')
('Канал 11', 'Канал 1')
('Канал 11', 'Канал 4')
('Канал 11', 'Канал 8')
('Канал 11', 'Канал 9')
('Канал 11', 'Канал 10')
('Канал 12', 'Канал 1')
('Канал 12', 'Канал 4')
('Канал 12', 'Канал 8')
('Канал 12', 'Канал 9')
('Канал 12', 'Канал 10')
('Канал 12', 'Канал 11')

Частковий коефіцієнт кореляції між Каналом 1 та Каналом 3, без урахування Канала 2: -0.998769045586596

Частковий коефіцієнт кореляції між Каналом 1 та Каналом 2, без урахування Каналів 3 та 4: 0.20650398846940585

Частковий коефіцієнт кореляції між Каналом 1 та Каналом 3, без урахування Каналів 2 та 4: -0.19398244958329272

Частковий коефіцієнт кореляції між Каналом 1 та Каналом 4, без урахування Каналів 2 та 3: -1.009401483109192

Множинний коефіцієнт кореляції для Канала 1 з Каналом 2 та Каналом 3: 0.9992393850595984

Трифакторний множинний коефіцієнт кореляції для Канала 1, Канала 2, Канала 3 та Канала 4: 0.9698097961090033

Незалежні канали (кореляція з іншими факторами < 0.3):
[]

Process finished with exit code 0

Таблиця (Кореляційна матриця):

	Канал 1	Канал 2	Канал 3	Канал 4	Канал 5
Канал 1	1	0,6180193599	-0,6153946083	-0,919308752	0,9175084524
Канал 2	0,6180193599	1	0,2381844162	-0,877341692	0,2554136739
Канал 3	-0,6153946083	0,2381844162	1	0,2565489922	-0,8772387783
Канал 4	-0,919308752	-0,877341692	0,2565489922	1	-0,6875299786
Канал 5	0,9175084524	0,2554136739	-0,8772387783	-0,6875299786	1
Канал 6	0,005982289742	0,7892947934	0,7837829593	-0,3981918306	-0,3907416757
Канал 7	-0,8649462989	-0,4828347895	0,581751341	0,7684413999	-0,8191984909
Канал 8	0,8961545174	0,8596779582	-0,2471002469	-0,9776601764	0,6662766962
Канал 9	0,8835883312	0,8882761209	-0,2031554643	-0,9841913844	0,6372010494
Канал 10	0,9200583175	0,8266843983	-0,3125685716	-0,9756894987	0,7144115736
Канал 11	0,922369278	0,8238347597	-0,3181345922	-0,9754551468	0,7193245952
Канал 12	0,9271426001	0,8327197524	-0,3139319312	-0,982575197	0,720922927

Канал 6	Канал 7	Канал 8	Канал 9	Канал 10	Канал 11	Канал 12

0,005982289742	-0,8649462989	0,8961545174	0,8835883312	0,9200583175	0,922369278	0,9271426001
0,7892947934	-0,4828347895	0,8596779582	0,8882761209	0,8266843983	0,8238347597	0,8327197524
0,7837829593	0,581751341	-0,2471002469	-0,2031554643	-0,3125685716	-0,3181345922	-0,3139319312
-0,3981918306	0,7684413999	-0,9776601764	-0,9841913844	-0,9756894987	-0,9754551468	-0,982575197
-0,3907416757	-0,8191984909	0,6662766962	0,6372010494	0,7144115736	0,7193245952	0,720922927
1	0,05784644763	0,3918510268	0,438257031	0,3304237754	0,3253826761	0,3338914453
0,05784644763	1	-0,7140209372	-0,716470035	-0,7548021088	-0,7553613077	-0,766034125
0,3918510268	-0,7140209372	1	0,9942845375	0,977792658	0,9660387918	0,9628160151
0,438257031	-0,716470035	0,9942845375	1	0,983583904	0,9743838187	0,9736940152
0,3304237754	-0,7548021088	0,977792658	0,983583904	1	0,9957690048	0,9901704032
0,3253826761	-0,7553613077	0,9660387918	0,9743838187	0,9957690048	1	0,9972089747
0,3338914453	-0,766034125	0,9628160151	0,9736940152	0,9901704032	0,9972089747	1

5. Факторний аналіз

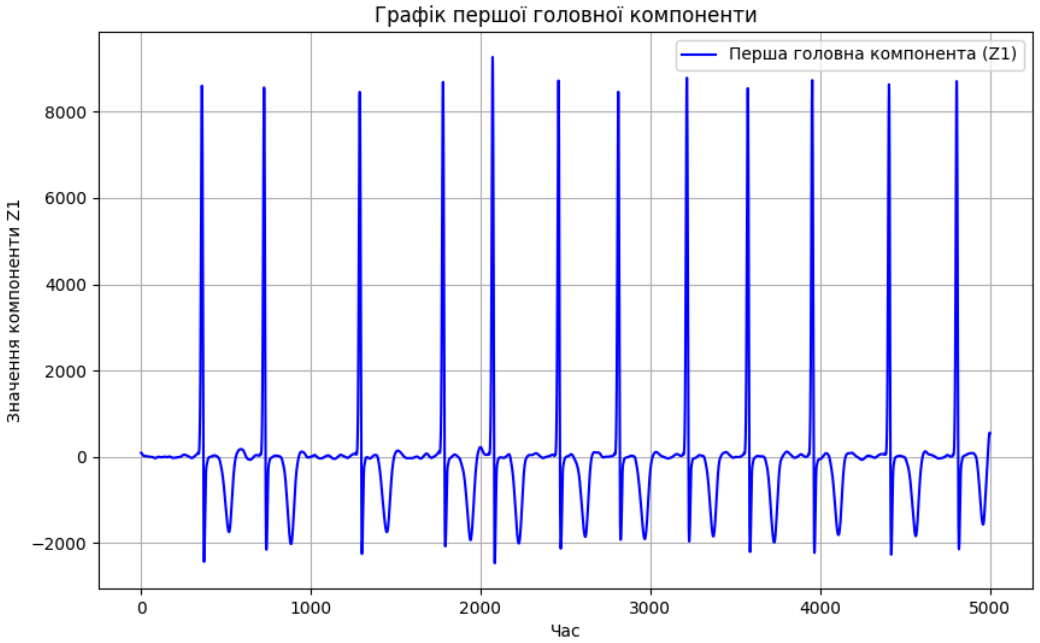
Код:

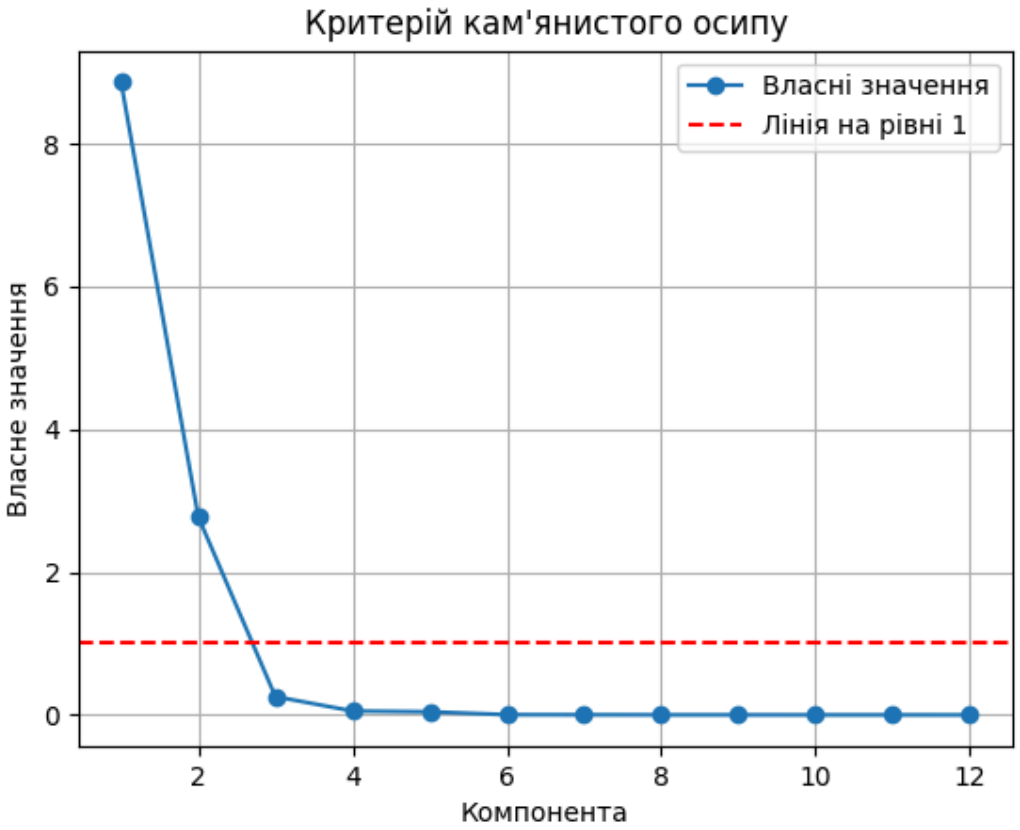
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Завантаження даних
data = np.genfromtxt('A3.txt', delimiter=',')
data = data.reshape(5000, 12)  # Перетворення даних у форму (5000 рядків, 12 каналів)

# Побудова кореляційної матриці для всіх 12 каналів.
R = np.corrcoef(data.T)

# Крок 1: Знаходження власних чисел і частка дисперсії
# Обчислюємо власні числа та власні вектори для кореляційної матриці.
eigenvalues, eigenvectors = np.linalg.eig(R)

# Сортуємо власні числа за спаданням, щоб знайти найбільші компоненти
sorted_indices = np.argsort(eigenvalues)[::-1]
sorted_eigenvalues = eigenvalues[sorted_indices]
sorted_eigenvectors = eigenvectors[:, sorted_indices]

# Розрахунок частки дисперсії для кожної компоненти
explained_variance = sorted_eigenvalues / np.sum(sorted_eigenvalues)
cumulative_variance = np.cumsum(explained_variance)

# Крок 2: Побудова графіка критерію кам’янистого осипу
# Це графік власних чисел для визначення значущих компонент (кам’янистий осип).
plt.figure()
plt.plot(range(1, len(sorted_eigenvalues) + 1), sorted_eigenvalues, 'o-', label="Власні значення")
plt.axhline(y=1, color='r', linestyle='--', label="Лінія на рівні 1")  # Добавляем горизонтальную линию на
уровне 1
plt.xlabel("Компонента")
plt.ylabel("Власне значення")
plt.title("Критерій кам'янистого осипу")
plt.legend()
```

```
plt.grid()
plt.show()

# Обчислення інформативності кожної компоненти (I_k)
# Це значення, що відображає вклад кожної компоненти в загальну дисперсію.
informative_index = np.cumsum(sorted_eigenvalues) / np.sum(sorted_eigenvalues)
print("\nІнформативність компонент:")
print(informative_index)

# Таблиця 1 - Власні числа та частка дисперсії
table1 = pd.DataFrame({
  "№п/п": np.arange(1, len(sorted_eigenvalues) + 1),
  "Власні числа": sorted_eigenvalues,
  "Частка дисперсії": explained_variance,
  "Сумарна дисперсія": cumulative_variance
})
print("Таблиця 1: Власні числа та частка дисперсії")
print(table1)

# Крок 3: Власні вектори (матриця L)
L = sorted_eigenvectors  # Матриця власних векторів
table2 = pd.DataFrame(L, columns=[f"Компонента {i + 1}" for i in range(L.shape[1])])
print("\nТаблиця 2: Власні вектори (матриця L)")
print(table2)

# Таблиця 3 - Власний вектор для максимального власного числа
# Знайдемо власний вектор для максимального власного числа (першого вектора).
principal_vector = L[:, 0]  # Вектор максимального власного числа
table3 = pd.DataFrame({"Власний вектор максимального власного числа": principal_vector})
print("\nТаблиця 3: Власний вектор максимального власного числа")
print(table3)

# Крок 5: Знаходження головних компонент
principal_components = data.dot(L)  # Матриця головних компонент

# Крок 4: Перевірка ортогональності власних векторів
# Перевіряємо, чи є власні вектори ортогональними, тобто чи їх скалярний добуток дорівнює нулю для i
!= j.
print("\nПеревірка ортогональності власних векторів (a'_j * a_k):")
for i in range(len(sorted_eigenvectors)):
  for j in range(i + 1, len(sorted_eigenvectors)):
    dot_product = np.dot(sorted_eigenvectors[:, i], sorted_eigenvectors[:, j])
    print(f"Скалярний добуток a_{i + 1}' та a_{j + 1}: {dot_product}")

# Перевірка норм власних векторів (a'_k * a_k = 1)
# Перевіряємо, чи норма кожного власного вектора дорівнює 1.
print("\nПеревірка норм власних векторів (a'_k * a_k = 1):")
for i in range(len(sorted_eigenvectors)):
  norm = np.linalg.norm(sorted_eigenvectors[:, i])
  print(f"Норма власного вектора a_{i + 1}: {norm}")

# Таблиця 8: Перші три головні фактори
```

```
table4 = pd.DataFrame({
    "№п/п": np.arange(1, principal_components.shape[0] + 1),
    "Z1": principal_components[:, 0],
    "Z2": principal_components[:, 1],
    "Z3": principal_components[:, 2]
})
print("\nТаблиця 4: Перші три головні фактори")
print(table4)

# Графік першої головної компоненти
plt.figure(figsize=(10, 6))
plt.plot(table4["№п/п"], table4["Z1"], label="Перша головна компонента (Z1)", color="b")
plt.xlabel("Час")
plt.ylabel("Значення компоненти Z1")
plt.title("Графік першої головної компоненти")
plt.legend()
plt.grid()
plt.show()

# Крок 6: Перевірка умов для головних компонент: сума всіх значень головної компоненти повинна
дорівнювати нулю.
for i in range(principal_components.shape[1]):
    sum_z = np.sum(principal_components[:, i])
    print(f"Сума компонент Z{i + 1}: {sum_z}")

# Перевірка дисперсії: 1/n * z'_k * z_k = лямда_k
for i in range(principal_components.shape[1]):
    variance_check = np.mean(principal_components[:, i] ** 2)
    print(f"Перевірка для компоненти Z{i + 1}: {variance_check} ≈ {sorted_eigenvalues[i]}")

# Перевірка ортогональності головних компонент: z'_j * z_k = 0 для j != k
for i in range(principal_components.shape[1]):
    for j in range(i + 1, principal_components.shape[1]):
        dot_product = np.dot(principal_components[:, i], principal_components[:, j])
        print(f"Скалярний добуток Z_{i + 1} та Z_{j + 1}: {dot_product}")

# Збереження всіх таблиць в Excel
with pd.ExcelWriter("Результати_факторного_аналізу.xlsx") as writer:
    table1.to_excel(writer, sheet_name="Таблиця 1 - Власні числа", index=False)
    table2.to_excel(writer, sheet_name="Таблиця 2 - Власні вектори", index=False)
    table3.to_excel(writer, sheet_name="Таблиця 3 - Макс. власн. вектор", index=False)
    table4.to_excel(writer, sheet_name="Таблиця 4 - Головні фактори", index=False)

print("Усі таблиці збережено у файл 'Результати_факторного_аналізу.xlsx'")
```

Результат:





Вивід:

Інформативність компонент:

[0.73850745 0.96976888 0.99109544 0.99569298 0.99934447 0.99964114
 0.99983837 0.99991094 0.9999573 0.999979   0.99999516 1.   ]

Таблиця 1: Власні числа та частка дисперсії

	№п/п	Власні числа	Частка дисперсії	Сумарна дисперсія
0	1	8.862089	0.738507	0.738507
1	2	2.775137	0.231261	0.969769
2	3	0.255919	0.021327	0.991095
3	4	0.055170	0.004598	0.995693
4	5	0.043818	0.003651	0.999344
5	6	0.003560	0.000297	0.999641
6	7	0.002367	0.000197	0.999838
7	8	0.000871	0.000073	0.999911
8	9	0.000556	0.000046	0.999957
9	10	0.000260	0.000022	0.999979
10	11	0.000194	0.000016	0.999995
11	12	0.000058	0.000005	1.000000

Таблиця 2: Власні вектори (матриця L)

	Компонента 1	Компонента 2	...	Компонента 11	Компонента 12
0	0.321718	-0.164049	...	-0.576705	-0.351941
1	0.270876	0.351250	...	0.359183	-0.576656
2	-0.126457	0.554288	...	0.338589	-0.033494
3	-0.331899	-0.075663	...	-0.122710	-0.700598
4	0.258873	-0.379079	...	0.470211	0.044104
5	0.093126	0.574464	...	-0.412357	0.222324
6	-0.275824	0.199048	...	-0.006267	-0.001068
7	0.327350	0.082249	...	0.053985	0.008517
8	0.328037	0.109380	...	-0.064950	-0.011232
9	0.331853	0.041358	...	0.044539	-0.004031
10	0.331517	0.037372	...	-0.073125	0.007492
11	0.332283	0.039822	...	0.055300	-0.000002

[12 rows x 12 columns]

Таблиця 3: Власний вектор максимального власного числа

	Власний вектор максимального власного числа
0	0.321718
1	0.270876
2	-0.126457

3	-0.331899
4	0.258873
5	0.093126
6	-0.275824
7	0.327350
8	0.328037
9	0.331853
10	0.331517
11	0.332283

Перевірка ортогональності власних векторів (a'_j * a_k):
Скалярний добуток a_1' та a_2: -9.8879238130678e-17
Скалярний добуток a_1' та a_3: -1.6653345369377348e-16
Скалярний добуток a_1' та a_4: 1.6653345369377348e-16
Скалярний добуток a_1' та a_5: 1.5265566588595902e-16
Скалярний добуток a_1' та a_6: 8.326672684688674e-17
Скалярний добуток a_1' та a_7: -7.632783294297951e-17
Скалярний добуток a_1' та a_8: 4.215378046623641e-16
Скалярний добуток a_1' та a_9: 2.7755575615628914e-17
Скалярний добуток a_1' та a_10: -5.551115123125783e-17
Скалярний добуток a_1' та a_11: -1.734723475976807e-16
Скалярний добуток a_1' та a_12: 1.3366688127441212e-16
Скалярний добуток a_2' та a_3: -1.1449174941446927e-16
Скалярний добуток a_2' та a_4: 2.42861286636753e-17
Скалярний добуток a_2' та a_5: 1.214306433183765e-16
Скалярний добуток a_2' та a_6: -4.5102810375396984e-17
Скалярний добуток a_2' та a_7: 4.683753385137379e-17
Скалярний добуток a_2' та a_8: 4.2500725161431774e-17
Скалярний добуток a_2' та a_9: -2.5847379792054426e-16
Скалярний добуток a_2' та a_10: 1.491862189340054e-16
Скалярний добуток a_2' та a_11: -2.7755575615628914e-16
Скалярний добуток a_2' та a_12: -1.263990606542844e-16
Скалярний добуток a_3' та a_4: 3.8163916471489756e-16
Скалярний добуток a_3' та a_5: 3.677613769070831e-16
Скалярний добуток a_3' та a_6: 9.020562075079397e-17
Скалярний добуток a_3' та a_7: 8.396061623727746e-16
Скалярний добуток a_3' та a_8: -6.765421556309548e-17

Скалярний добуток a_3' та a_9: 2.7929047963226594e-16
Скалярний добуток a_3' та a_10: -9.43689570931383e-16
Скалярний добуток a_3' та a_11: 4.432218481120742e-16
Скалярний добуток a_3' та a_12: -5.491360654466136e-16
Скалярний добуток a_4' та a_5: -1.326716514427062e-14
Скалярний добуток a_4' та a_6: -2.942091015256665e-15
Скалярний добуток a_4' та a_7: 1.3739009929736312e-15
Скалярний добуток a_4' та a_8: 1.6115581091824538e-15
Скалярний добуток a_4' та a_9: -2.8033131371785203e-15
Скалярний добуток a_4' та a_10: 2.886579864025407e-15
Скалярний добуток a_4' та a_11: -3.400058012914542e-15
Скалярний добуток a_4' та a_12: 3.3286018899676955e-15
Скалярний добуток a_5' та a_6: 2.4980018054066022e-15
Скалярний добуток a_5' та a_7: -4.2327252813834093e-16
Скалярний добуток a_5' та a_8: -8.760353553682876e-16
Скалярний добуток a_5' та a_9: 1.8110513089197866e-15
Скалярний добуток a_5' та a_10: 1.1102230246251565e-15
Скалярний добуток a_5' та a_11: -1.8943180357666733e-15
Скалярний добуток a_5' та a_12: -1.796019438721338e-15
Скалярний добуток a_6' та a_7: -4.5755066402364264e-14
Скалярний добуток a_6' та a_8: 5.6239735091168086e-15
Скалярний добуток a_6' та a_9: 5.703770789011742e-15
Скалярний добуток a_6' та a_10: -2.7977620220553945e-14
Скалярний добуток a_6' та a_11: -7.077671781985373e-16
Скалярний добуток a_6' та a_12: -4.752836333524256e-14
Скалярний добуток a_7' та a_8: -7.601558271730369e-14
Скалярний добуток a_7' та a_9: 4.714978407704962e-15
Скалярний добуток a_7' та a_10: 1.7687240561059525e-14
Скалярний добуток a_7' та a_11: 3.804161846643339e-14
Скалярний добуток a_7' та a_12: 4.767681147084215e-14
Скалярний добуток a_8' та a_9: -1.8610113450279187e-14
Скалярний добуток a_8' та a_10: 4.429095978863984e-14
Скалярний добуток a_8' та a_11: 1.5097255043339253e-13
Скалярний добуток a_8' та a_12: 9.891869543999017e-15
Скалярний добуток a_9' та a_10: -6.358802373540584e-13
Скалярний добуток a_9' та a_11: -1.967488671983375e-13
Скалярний добуток a_9' та a_12: -4.0805834987398464e-13

Скалярний добуток a_10' та a_11: 3.720534297313449e-12
Скалярний добуток a_10' та a_12: 2.73360371051836e-13
Скалярний добуток a_11' та a_12: -2.877646043206324e-12

Перевірка норм власних векторів (a'_k * a_k = 1):
Норма власного вектора a_1: 1.0
Норма власного вектора a_2: 0.9999999999999999
Норма власного вектора a_3: 0.9999999999999999
Норма власного вектора a_4: 0.9999999999999999
Норма власного вектора a_5: 1.0
Норма власного вектора a_6: 1.0
Норма власного вектора a_7: 0.9999999999999999
Норма власного вектора a_8: 1.0
Норма власного вектора a_9: 1.0
Норма власного вектора a_10: 1.0
Норма власного вектора a_11: 0.9999999999999999
Норма власного вектора a_12: 0.9999999999999999

Таблиця 4: Перші три головні фактори

	№п/п	Z1	Z2	Z3
0	1	92.567707	36.607716	-21.470373
1	2	91.513663	35.263624	-21.965284
2	3	89.629499	34.124255	-22.391706
3	4	86.358953	33.527619	-22.609488
4	5	81.581798	33.960635	-22.443550
...	...	...	...	...
4995	4996	541.299579	153.948255	-188.198545
4996	4997	555.994259	163.498137	-187.997235
4997	4998	559.342388	166.796005	-179.589872
4998	4999	552.899770	164.163846	-164.103723
4999	5000	541.301590	158.113050	-144.748187

[5000 rows x 4 columns]
Сума компонент Z1: -24616.20918695443
Сума компонент Z2: -133793.6270887158
Сума компонент Z3: 41015.296824623416
Сума компонент Z4: -131689.97144610126

Сума компонент Z5: 130565.28455378974
Сума компонент Z6: -97969.64728288454
Сума компонент Z7: -5139.728983990724
Сума компонент Z8: 12906.407327459681
Сума компонент Z9: 3938.7049084031573
Сума компонент Z10: 26835.478113949983
Сума компонент Z11: -6719.046053006619
Сума компонент Z12: -4407.641903346112
Перевірка для компоненти Z1: 1902647.2170501107 ≈ 8.862089399753202
Перевірка для компоненти Z2: 76452.57660440236 ≈ 2.775137120625244
Перевірка для компоненти Z3: 207104.99892024504 ≈ 0.25591879361992553
Перевірка для компоненти Z4: 810159.8392287598 ≈ 0.05517044847359576
Перевірка для компоненти Z5: 67081.99181396748 ≈ 0.04381784122626612
Перевірка для компоненти Z6: 24301.971228075792 ≈ 0.0035600531481040664
Перевірка для компоненти Z7: 6139.541029040308 ≈ 0.002366809939815533
Перевірка для компоненти Z8: 1155.2568809714112 ≈ 0.0008707984991210818
Перевірка для компоненти Z9: 3216.9585401114978 ≈ 0.000556368834914687
Перевірка для компоненти Z10: 1262.026148098647 ≈ 0.00026035620707545097
Перевірка для компоненти Z11: 583.9775900374316 ≈ 0.0001939835410864226
Перевірка для компоненти Z12: 116.15130223398933 ≈ 5.802613163966975e-05
Скалярний добуток Z_1 та Z_2: 1078960032.6359026
Скалярний добуток Z_1 та Z_3: -3111036702.0245194
Скалярний добуток Z_1 та Z_4: 6151178824.227955
Скалярний добуток Z_1 та Z_5: -1665473219.7419455
Скалярний добуток Z_1 та Z_6: -970199749.025198
Скалярний добуток Z_1 та Z_7: -466621290.24959683
Скалярний добуток Z_1 та Z_8: 206931814.10407364
Скалярний добуток Z_1 та Z_9: -385873200.40375495
Скалярний добуток Z_1 та Z_10: -170239490.9007429
Скалярний добуток Z_1 та Z_11: -117008956.97374673
Скалярний добуток Z_1 та Z_12: -68649700.1548002

Скалярний добуток Z_2 та Z_3: -351989831.96096075
Скалярний добуток Z_2 та Z_4: 699095320.8051293
Скалярний добуток Z_2 та Z_5: -236683969.76837546
Скалярний добуток Z_2 та Z_6: -123378585.65552498
Скалярний добуток Z_2 та Z_7: -31269691.947784007
Скалярний добуток Z_2 та Z_8: 7048886.305556643
Скалярний добуток Z_2 та Z_9: -47823807.11840004
Скалярний добуток Z_2 та Z_10: -28452132.913424917
Скалярний добуток Z_2 та Z_11: 6061981.23550803
Скалярний добуток Z_2 та Z_12: -11115050.665863093
Скалярний добуток Z_3 та Z_4: -2020670798.3866813
Скалярний добуток Z_3 та Z_5: 551018887.3613732
Скалярний добуток Z_3 та Z_6: 312179790.2679731
Скалярний добуток Z_3 та Z_7: 156842877.45927626
Скалярний добуток Z_3 та Z_8: -68901783.06817697
Скалярний добуток Z_3 та Z_9: 127055659.80487284
Скалярний добуток Z_3 та Z_10: 58893017.53425325
Скалярний добуток Z_3 та Z_11: 38615387.08772141
Скалярний добуток Z_3 та Z_12: 21951536.530954495
Скалярний добуток Z_4 та Z_5: -1052508679.8436272
Скалярний добуток Z_4 та Z_6: -628455369.2849874
Скалярний добуток Z_4 та Z_7: -315076240.0366822
Скалярний добуток Z_4 та Z_8: 134025510.18038999
Скалярний добуток Z_4 та Z_9: -252245510.28000164
Скалярний добуток Z_4 та Z_10: -118595323.41525939
Скалярний добуток Z_4 та Z_11: -75841995.78621851
Скалярний добуток Z_4 та Z_12: -43238247.54023595
Скалярний добуток Z_5 та Z_6: 151956231.48514038
Скалярний добуток Z_5 та Z_7: 73596595.22375078
Скалярний добуток Z_5 та Z_8: -34511980.28367011
Скалярний добуток Z_5 та Z_9: 68583803.40407899
Скалярний добуток Z_5 та Z_10: 32376355.04831916
Скалярний добуток Z_5 та Z_11: 16973616.320624962
Скалярний добуток Z_5 та Z_12: 12393286.334248343
Скалярний добуток Z_6 та Z_7: 46605036.45360081
Скалярний добуток Z_6 та Z_8: -19244202.015677586
Скалярний добуток Z_6 та Z_9: 38175808.3923284

Скалярний добуток Z_6 та Z_10: 15604942.769252583
Скалярний добуток Z_6 та Z_11: 10873513.673545172
Скалярний добуток Z_6 та Z_12: 7519116.265265649
Скалярний добуток Z_7 та Z_8: -11972307.924962256
Скалярний добуток Z_7 та Z_9: 19552765.438637137
Скалярний добуток Z_7 та Z_10: 10664387.688521726
Скалярний добуток Z_7 та Z_11: 7512062.359464159
Скалярний добуток Z_7 та Z_12: 2821263.532564713
Скалярний добуток Z_8 та Z_9: -8289164.307115899
Скалярний добуток Z_8 та Z_10: -3538681.7879201416
Скалярний добуток Z_8 та Z_11: -3782970.680343068
Скалярний добуток Z_8 та Z_12: -1244480.6704059183
Скалярний добуток Z_9 та Z_10: 7867484.164265677
Скалярний добуток Z_9 та Z_11: 4497412.301264692
Скалярний добуток Z_9 та Z_12: 2772332.649210372
Скалярний добуток Z_10 та Z_11: 1608056.3758663884
Скалярний добуток Z_10 та Z_12: 1102339.4669332616
Скалярний добуток Z_11 та Z_12: 587198.7504404434

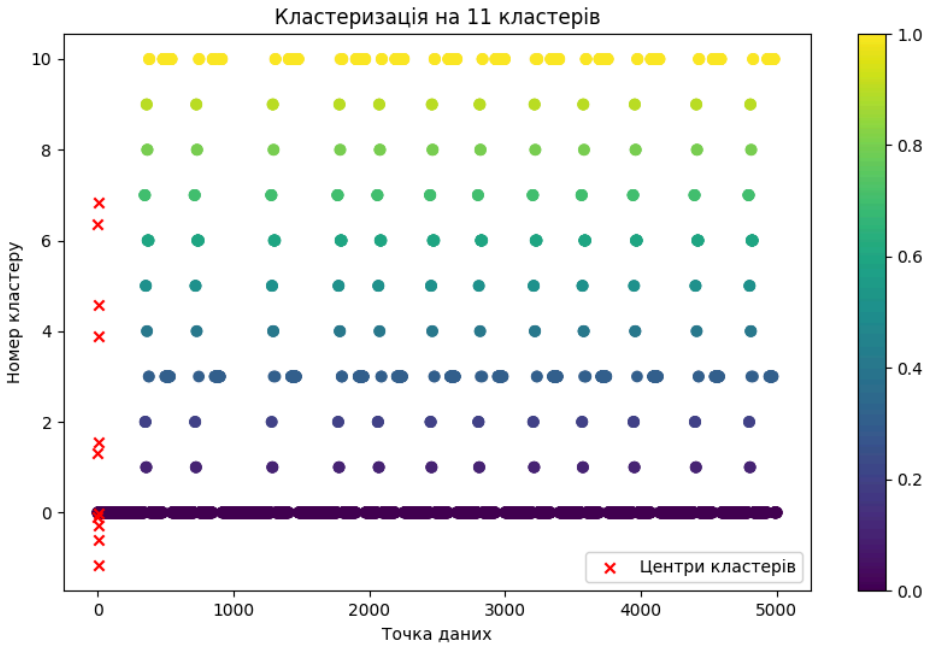
6. Кластерний аналіз

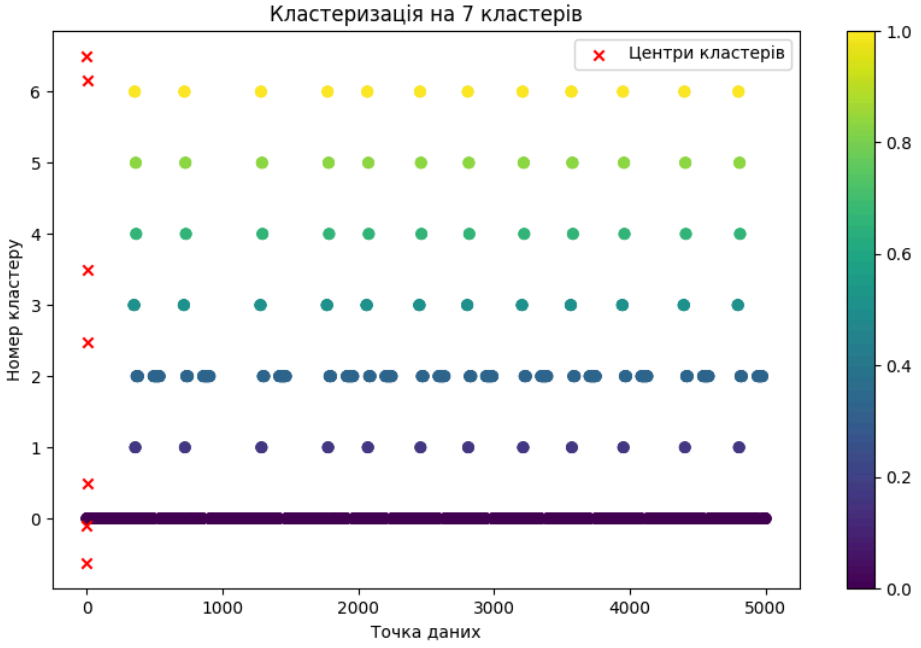
Код:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from scipy.signal import find_peaks

# Завантаження даних
data = np.genfromtxt('A3.txt', delimiter=',')
data = data.reshape(5000, 12)  # Розділяємо на 12 каналів
N = 5000  # кількість точок
t = np.linspace(0, 10, N)  # Час запису 10 секунд
# Нормалізація даних для кращої кластеризації
scaler = StandardScaler()
data_normalized = scaler.fit_transform(data)

# 1. Кластеризація на 11 кластерів
kmeans_11 = KMeans(n_clusters=11, random_state=42)
kmeans_11.fit(data_normalized)
labels_11 = kmeans_11.labels_
centroids_11 = kmeans_11.cluster_centers_

# Візуалізація кластерів для 11 кластерів
```

```
plt.figure(figsize=(10, 6))
plt.scatter(range(N), labels_11, c=labels_11, cmap='viridis')
plt.title('Кластеризація на 11 кластерів')
plt.xlabel('Точка даних')
plt.ylabel('Номер кластеру')

# Додаємо центри кластерів як червоні хрестики
plt.scatter(np.arange(11), centroids_11[:, 0], color='red', marker='x', label='Центри кластерів')
plt.colorbar()
plt.legend()
plt.show()

# 2. Кластеризація на 7 кластерів
kmeans_7 = KMeans(n_clusters=7, random_state=42)
kmeans_7.fit(data_normalized)
labels_7 = kmeans_7.labels_
centroids_7 = kmeans_7.cluster_centers_

# Візуалізація кластерів для 7 кластерів
plt.figure(figsize=(10, 6))
plt.scatter(range(N), labels_7, c=labels_7, cmap='viridis')
plt.title('Кластеризація на 7 кластерів')
plt.xlabel('Точка даних')
plt.ylabel('Номер кластеру')

# Додаємо центри кластерів як червоні хрестики
plt.scatter(np.arange(7), centroids_7[:, 0], color='red', marker='x', label='Центри кластерів')
plt.colorbar()
plt.legend()
plt.show()

# 3. Зниження розмірності до 3-х головних компонентів
pca = PCA(n_components=3)
data_pca = pca.fit_transform(data_normalized)

# 4. Кластеризація на 3-х компонентах (PCA)
kmeans_pca = KMeans(n_clusters=11, random_state=42)
kmeans_pca.fit(data_pca)
labels_pca = kmeans_pca.labels_
centroids_pca = kmeans_pca.cluster_centers_

# Візуалізація результатів кластеризації після PCA
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(data_pca[:, 0], data_pca[:, 1], data_pca[:, 2], c=labels_pca, cmap='viridis')
ax.set_title('Кластеризація після PCA на 11 кластерів')
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
```

Результат:





7. Перетворення Фур’є

Код:

```
import numpy as np
import matplotlib.pyplot as plt

# Завантаження даних
data = np.genfromtxt('A3.txt', delimiter=',')
data = data.reshape(5000, 12)  # Розділяємо на 12 каналів
```

```
N = 5000  # кількість точок
t = np.linspace(0, 10, N)  # Час запису 10 секунд
channels = data.T  # Кожен канал як окремий рядок (12 x 5000)

# Крок по частоті
freq_step = 1 / 10  # Δf = 1/T = 1/10 = 0.1 Гц
print(f"Частота першої синусоїди (крок по частоті) = {freq_step} Гц")

# Обчислення коефіцієнтів Фур'є для кожного каналу
A = []
B = []
C = []

for channel in channels:
  # Розрахунок A_j і B_j
  A_j = np.zeros(N // 2 + 1)
  B_j = np.zeros(N // 2 + 1)

  for j in range(N // 2 + 1):
    if j == 0:
      A_j[j] = (1 / N) * np.sum(channel * np.cos((2 * np.pi * np.arange(N) * 0) / N))
    elif j == N // 2:
      A_j[j] = (1 / N) * np.sum(channel * np.cos((np.pi * np.arange(N)) / 1))
    else:
      A_j[j] = (2 / N) * np.sum(channel * np.cos((2 * np.pi * np.arange(N) * j) / N))
      B_j[j] = (2 / N) * np.sum(channel * np.sin((2 * np.pi * np.arange(N) * j) / N))

  # Розрахунок спектру C_j
  C_j = np.sqrt(A_j ** 2 + B_j ** 2)
  A.append(A_j)
  B.append(B_j)
  C.append(C_j)

# Перетворюємо A, B, C у масиви для зручності
A = np.array(A)
B = np.array(B)
C = np.array(C)

#Обернене перетворення Фур'є для відновлення сигналу
restored_data = []

for k in range(12):
  restored_signal = np.zeros(N)
  for i in range(N):
    restored_signal[i] = np.sum(A[k] * np.cos((2 * np.pi * np.arange(N // 2 + 1) * i) / N) +
                B[k] * np.sin((2 * np.pi * np.arange(N // 2 + 1) * i) / N))
  restored_data.append(restored_signal)

restored_data = np.array(restored_data)

#Спектр модуля сигналу для кожного каналу
for i in range(12):
```

```
  plt.figure(figsize=(10, 4))
  plt.plot(C[i])  # Загальний спектр для каналу
  plt.title(f"Рис. 14: Спектр C_j для каналу {i + 1}")
  plt.xlabel("Частота (Гц)")
  plt.ylabel("Амплітуда")
  plt.grid(True)
  plt.show()

#Перші 200 точок початкового і відновленого сигналу для кожного каналу
for i in range(12):
  plt.figure(figsize=(12, 6))
  plt.plot(t[:200], channels[i, :200], label='Початковий сигнал')
  plt.plot(t[:200], restored_data[i, :200], label='Відновлений сигнал', linestyle='--')
  plt.title(f"Рис. 15: Порівняння початкового і відновленого сигналу для каналу {i + 1} (перші 200
точок)")
  plt.xlabel("Час (с)")
  plt.ylabel("Амплітуда")
  plt.legend()
  plt.grid(True)
  plt.show()

#Обчислення середньої похибки для кожного каналу
for i in range(12):
  error = np.mean(np.abs(channels[i] - restored_data[i]))
  print(f"Середня похибка для каналу {i + 1}: {error:.10f}")
```

Результат:

Частота першої синусоїди (крок по частоті) = 0.1 Гц

Середня похибка для каналу 1: 0.0000000001

Середня похибка для каналу 2: 0.0000000000

Середня похибка для каналу 3: 0.0000000000

Середня похибка для каналу 4: 0.0000000000

Середня похибка для каналу 5: 0.0000000000

Середня похибка для каналу 6: 0.0000000000

Середня похибка для каналу 7: 0.0000000000

Середня похибка для каналу 8: 0.0000000001

Середня похибка для каналу 9: 0.0000000001

Середня похибка для каналу 10: 0.0000000003

Середня похибка для каналу 11: 0.0000000003

Середня похибка для каналу 12: 0.0000000003

Process finished with exit code 0