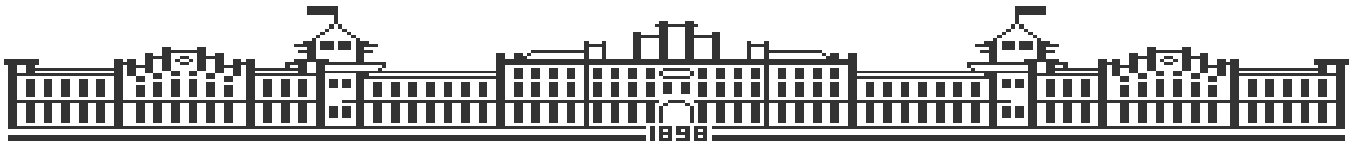


**Міністерство освіти і науки України**
**Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"**
**Факультет інформатики та обчислювальної техніки**
**Кафедра обчислювальної техніки**

# ЗВІТ

про виконання лабораторної роботи №1.1

з дисципліни

**«Інтелектуальні вбудовані системи»**

на тему:

## «ДОСЛІДЖЕННЯ І РОЗРОБКА МОДЕЛЕЙ ВИПАДКОВИХ СИГНАЛІВ. АНАЛІЗ ЇХ ХАРАКТЕРИСТИК»

Перевірив:
асистент кафедри ОТ
Регіда П. Г

ВИКОНАВ:
студент 3 курсу
групи ІП-83, ФІОТ
Мінченко В.Ю.
Залікова книжка №8315
Варіант – 18

Київ 2021

**Завдання на лабораторну роботу**

Згенерувати випадковий сигнал по співвідношенню (див. нижче) відповідно варіантом по таблицею (Додаток 1) і розрахувати його математичне сподівання і дисперсію. Розробити відповідну програму і вивести отримані значення і графіки відповідних параметрів

Варіант 18:

| Варіант | Число гармонік в сигналі n | Гранична частота, $\omega_{гр}$ | Кількість дискретних відліків, N |
|---|---|---|---|
| 18 | 10 | 1500 | 256 |

Програмний код:

Complexity_generator.py

```
import time as t
import signal_generator


def calculate_time(signal_harmonics, start_frequency, number_discrete_calls):
    time = []
    for n in range(number_discrete_calls):
        start_time = t.time()
        signal_generator.generate_signal(signal_harmonics, start_frequency, n)
        end_time = t.time()
        time.append(end_time - start_time)
    return time
```

signal_generator.py

```
import random
import numpy as np


def generate_signal(signal_harmonics, frequency, discrete_calls):
    signals = np.zeros(discrete_calls)
    for i in range(signal_harmonics):
        frequency_step = frequency / signal_harmonics * (i + 1)
        amplitude = random.random()
        phase = random.random()
        for t in range(discrete_calls):
            signals[t] += amplitude * np.sin(frequency_step * t + phase)
    return signals
```

statistics_utils.py

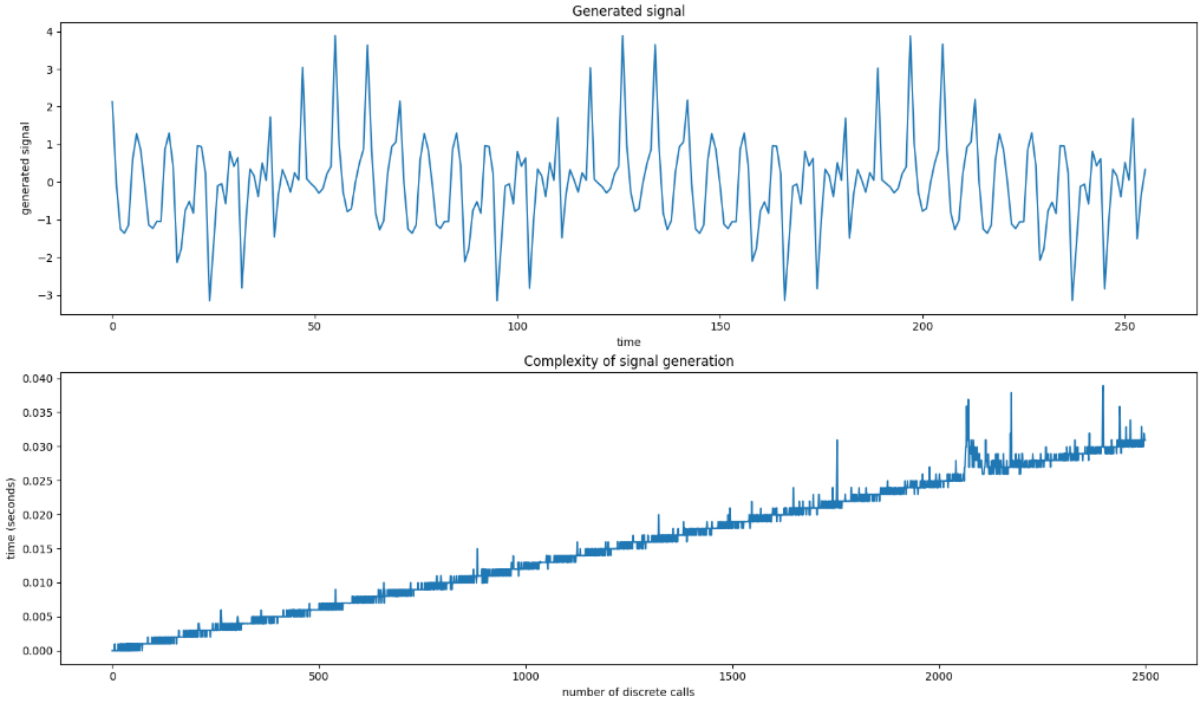
```
import numpy as np


def math_expectation(data):
    return np.mean(data)


def math_variance(data):
    return np.var(data)
```

Lab1_1.py

```
import matplotlib.pyplot as plt
import signal_generator
import complexity_generator
import statistics_utils

HARMONICS = 10
FREQUENCY = 1500
DISCRETE_CALLS = 256

signal = signal_generator.generate_signal(
  HARMONICS,
  FREQUENCY,
  DISCRETE_CALLS
)

time = complexity_generator.calculate_time(
  HARMONICS,
  FREQUENCY,
  2500
)

print("Math expectation = " + str(statistics_utils.math_expectation(signal)))
print("Math variance = " + str(statistics_utils.math_variance(signal)))

fig, (ax1, ax2) = plt.subplots(2)
fig.suptitle('Laboratorka 1.1')
fig.set_size_inches(18.5, 10.5)

ax1.plot(signal)
ax1.set_title('Generated signal')
ax1.set(xlabel='time', ylabel='generated signal')

ax2.plot(time)
ax2.set_title('Complexity of signal generation')
ax2.set(xlabel='number of discrete calls', ylabel='time (seconds)')
fig.savefig('lab1_1.png')

plt.show()
```

Результати роботи програми:

Графіки згенерованого сигналу та складності алгоритму обчислень:




Обчислення математичного очікування та дисперсії:

```
Math expectation = -0.07313164833245711
Math variance = 1.8019850346334918
```

Висновки:
Отже, в ході лабораторної роботи, ми отримали практичні вміння генерувати випадковий сигнал згідно формули та вхідних значень за допомогою власноруч написаної програми. Крім того, було отримано графік, на якому зображено складність алгоритму обчислень O(n). Результати наведено в звіті та в репозиторії. Кінцеву мету було досягнуто.