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

Лабораторна робота №1.1
з дисципліни
«Інтелектуальні вбудовані системи»
на тему
«Дослідження і розробка моделей випадкових сигналів. Аналіз їх характеристик»

Виконала:

студентка
групи ІП-83
Гомілко Д. В.

Перевірив:

Регіда П. Г.

Київ 2021

Основні теоретичні відомості, необхідні для виконання лабораторної роботи

СРЧ обов'язково пов'язані з деяким зовнішнім середовищем. СРЧ забезпечує контроль за зміною параметрів зовнішнього середовища і в ряді випадків забезпечує управління параметрами середовища через деякі впливу на неї. Параметри середовища представляються деякою зміною фізичного середовища. При вимірах фізичного параметра ми отримуємо певний електричний сигнал на вході вимірювального датчика. Для подання такого електричного сигналу можна використовувати різні моделі. Найкращою моделлю досліджуваного сигналу є відповідна математична інтерпретація випадкового процесу. Випадковий сигнал або процес завжди представляється деякою функцією часу x(t) , значення якої не можна передбачити з точністю засобів вимірювання або обчислень, які б кошти моделі ми не використовували.

Для випадкового процесу його значення можна передбачити лише основні його характеристики: математичне сподівання Mx(t) , дисперсію Dx(t) , автокореляційну функцію.

Ці характеристики для випадкового нестаціонарного процесу теж є функціями часу, але вони детерміновані. Для оцінки цих характеристик використовуються СРВ, які повинні обробити значну кількість інформації; для отримання їх при нестаціонарному процесі необхідно мати безліч реалізацій цього процесу.

Умови завдання для варіанту

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

Варіант: 04 (номер заліковки — 8304):

Варіант	Число гармонік в сигналі, n	Гранична частота, w	Кількість дискретних відліків, N
4	12	2400	1024

Лістинг програми із заданими умовами завдання

signalGenerator.py

```
import random

import math

def createSignal(harmonics, maxFrequency, calls):
sumsArr = [0] * calls
frequencyStep = maxFrequency / harmonics
for i in range(harmonics):
frequency = frequencyStep * (i + 1)
amplitude = random.random()
phase = random.random()
for t in range(calls):
sumsArr[t] += amplitude * math.sin(frequency * t + phase)
return sumsArr
```

calc.py

```
mean = lambda vals: sum(vals) / len(vals)

def variance(vals):
M = mean(vals)
N = len(vals)
return sum((x - M) ** 2 for x in vals) / (N - 1)

def correlation(signalX, signalY = []):
N = len(signalX)
calcRange = N / 2
Mx = mean(signalX)
My = mean(signalY) if signalY else Mx
comparedSignal = signalY or signalX
correlation = []
for tau in range(int(calcRange)):
res = 0
for t in range(int(calcRange)):
res += (signalX[t] - Mx) * (comparedSignal[t + tau] - My)
correlation.append(res / (calcRange - 1))
return correlation
```

complexity.py

```
import time

import signalGenerator
import calc

def getGenerationComplexity(stepsCount, maxFrequency, calls):
elapsed = []
harmonics = []
for i in range(stepsCount):
count = int(10 * (i + 1))
harmonics.append(count)
start = time.perf_counter()
signalGenerator.createSignal(count, maxFrequency, calls)
stop = time.perf_counter()
elapsed.append(stop - start)
return harmonics, elapsed
```

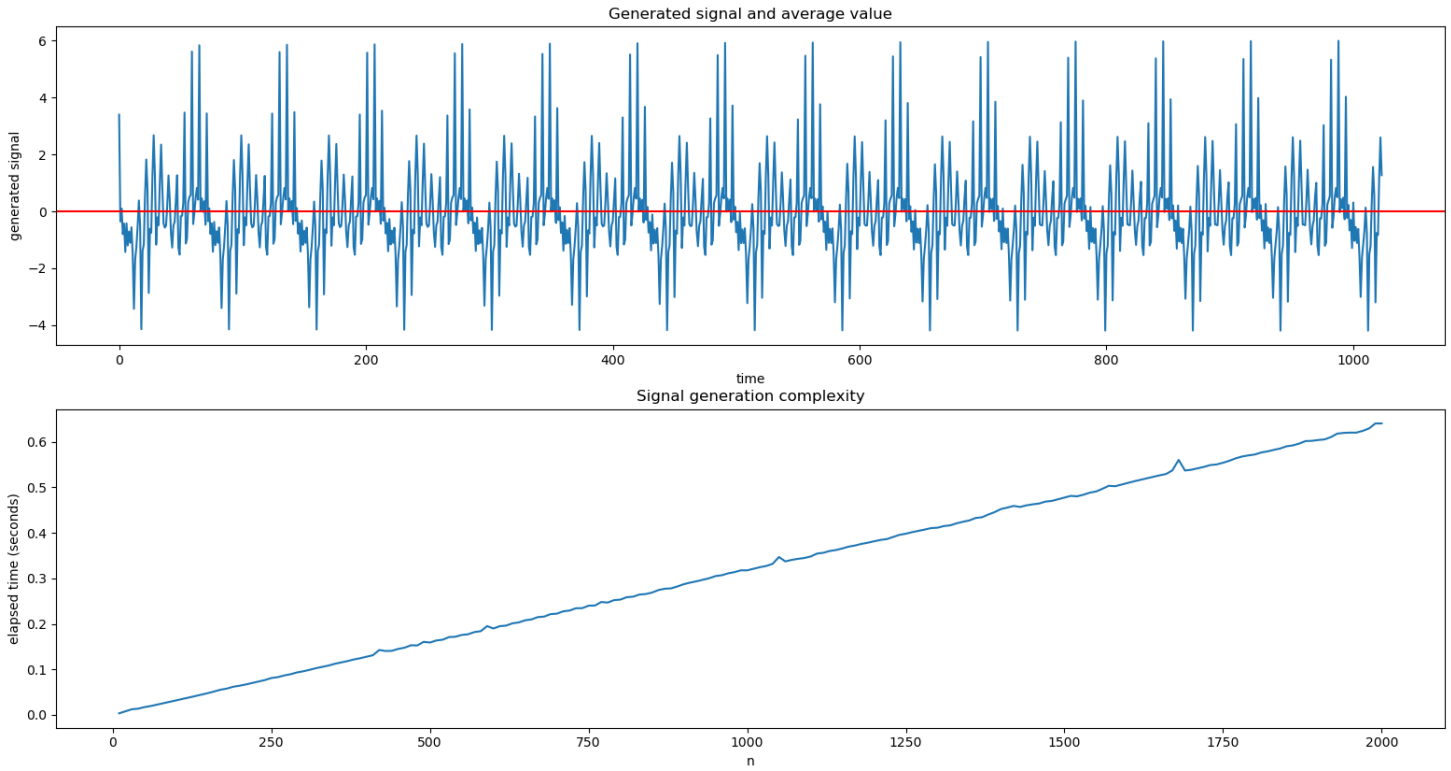
```
def getCorrealtionComplexity(stepsCount, harmonics, maxFrequency):
elapsed = []
harmonics = []
for i in range(stepsCount):
count = int(10 * (i + 1))
harmonics.append(count)
signal = signalGenerator.createSignal(count, maxFrequency, count)
start = time.perf_counter()
calc.correlation(signal)
stop = time.perf_counter()
elapsed.append(stop - start)
return harmonics, elapsed
```

lab1-1.py

```
import matplotlib.pyplot as plt

import signalGenerator
import calc
import complexity

HARMONICS = 12
MAX_FREQUENCY = 2400
DISCRETE_CALLS = 1024
COMPLEXITY_COUNT_LOOPS = 200

signal = signalGenerator.createSignal(
HARMONICS,
MAX_FREQUENCY,
DISCRETE_CALLS
)

mean = calc.mean(signal)
variance = calc.variance(signal)
print('Mean: ', mean)
print('Variance: ', variance)

harmonics, time = complexity.getGenerationComplexity(
COMPLEXITY_COUNT_LOOPS,
MAX_FREQUENCY,
DISCRETE_CALLS
)

fig, axs = plt.subplots(2)
fig.suptitle('Lab 1.1')

axs[0].plot(signal)
axs[0].axhline(y=mean, color='r')
axs[0].set_title('Generated signal and average value')
axs[0].set(xlabel='time', ylabel='generated signal')

axs[1].plot(harmonics, time)
axs[1].set_title('Signal generation complexity')
axs[1].set(xlabel='n', ylabel='elapsed time (seconds)')

plt.show()
fig.savefig('graphs/lab1-1.png')
```

Результати виконання кожної програми

Графіки згенерованого сигналу, його математичного очікування та складності алгоритму:



Результат обчислення математичного очікування та дисперсії:

```
Mean:  -0.01463070922138415
Variance:  2.598433593029807
```

Висновки щодо виконання лабораторної роботи

Під час виконання лабораторної роботи ми отримали практичні навички з генерації випадкових сигналів та обрахування відповідних їм значень математичного очікування та дисперсії. У ході роботи було створено генератор сигналів, що працює відповідно до заданих за варіантом параметрів та побудовано графік, що демонструє його роботу. При цьому розраховані значення математичного очікування та дисперсії виводяться у консоль. Крім того, було побудовано графік залежності витраченого на генерацію сигналу часу від кількості гармонік для візуалізації оцінки складності алгоритму.