ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____
ПРЕПОДАВАТЕЛЬ

д-р техн. наук, профессор		Т.М. Татарникова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

МОДЕЛИРОВАНИЕ ДИСКРЕТНОЙ СЛУЧАЙНОЙ ВЕЛИЧИНЫ

Вариант 6

по курсу: МОДЕЛИРОВАНИЕ СИСТЕМ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4128		В.А.Тарапанов
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

1 Цель и постановка задачи

1.1 Цель работы

Выполнить программную реализацию генератора дискретной случайной величины.

1.2 Задание

1. Выполнить программную реализацию датчика заданной дискретной СВ и сгенерировать выборку из 500 значений дискретной СВ xi.

2. Найти эмпирические оценки M и D и сравнить их с теоретическими значениями.

3. Построить в одном графическом окне две гистограммы: первая − распределение эмпирических вероятностей значений случайной величины x и вторая − распределение теоретических вероятностей СВ.

4. Дать сравнительную оценку гистограммам распределения эмпирических и теоретических вероятностей случайной величины x.

1.3 Условия варианта

Таблица 1

Параметры	j						
	1	2	3	4	5	6	7
x_j	-21.4	4.6	17.1	20.1	37.1	39.1	93.4
p_j	0,137	0,0098	0,065	0,240	0,258	0,108	0,094

1.4 Датчик БСВ

Мультипликативно-конгруэнтный датчик

$$A_i = (A_{i-55} + A_{i-24}) \bmod (2^{32})$$
$$B_i = (B_{i-57} + B_{i-7}) \bmod (2^{32})$$
$$C_i = (C_{i-58} + C_{i-19}) \bmod (2^{32})$$
$$z_i = \frac{A_i}{2^{32}},\ z_{i+1} = \frac{B_i}{2^{32}},\ z_{i+2} = \frac{C_i}{2^{32}} \quad (1)$$

2 Ход работы

Работа выполнялась при помощи пакета прикладных программ для решения задач технических вычислений.

При помощи формулы 1, был запрограммирован мультипликативно-конгруэнтный датчик (исследование его работы было проведено в одной из предыдущих лабораторных работ).

На основе разработанного датчика был запрограммирован генератор дискретных СВ, реализованы эмпирическая и теоретическая гистограммы распределения относительных частот и вывод теоретических и экспериментальных значений математического ожидания и дисперсии.

Теоретические значения:

$$M(x) = \sum_{j=1}^{K} p_j x_j ;$$

$$D(x) = \sum_{j=1}^{K} p_j x_j^2 - M^2(x), \qquad (2)$$

Таблица 2 – Первые 30 значений x

20.1	4.6	37.1	4.6	93.4	93.4	-21.4	-21.4	4.6	21.4
39.1	37.1	37.1	39.1	39.1	-21.4	4.6	20.1	-21.4	93.4
37.1	39.1	17.1	37.1	-21.4	20.1	20.1	93.4	20.1	-21.4

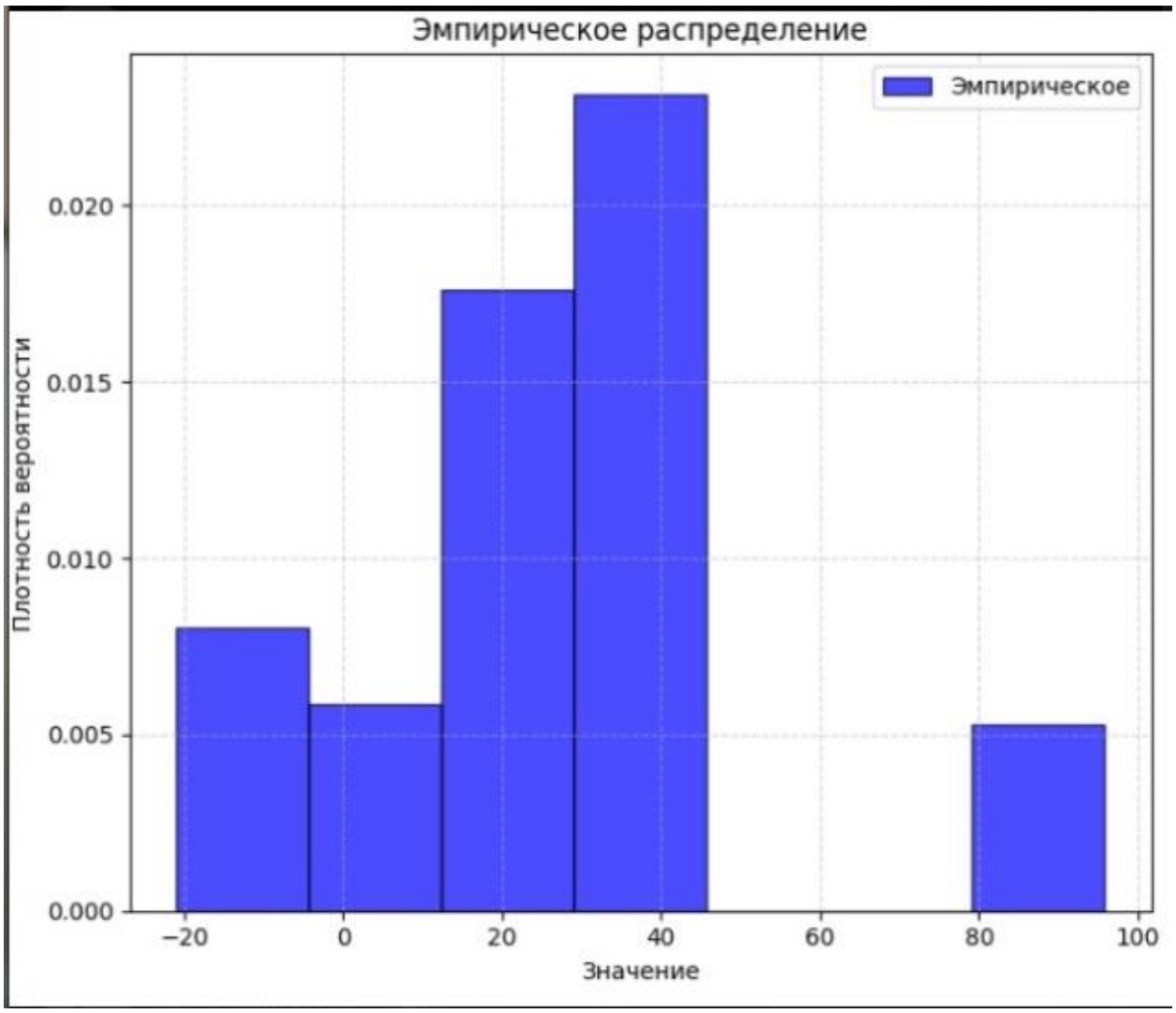

Рисунок 1 – Эмпирические значения

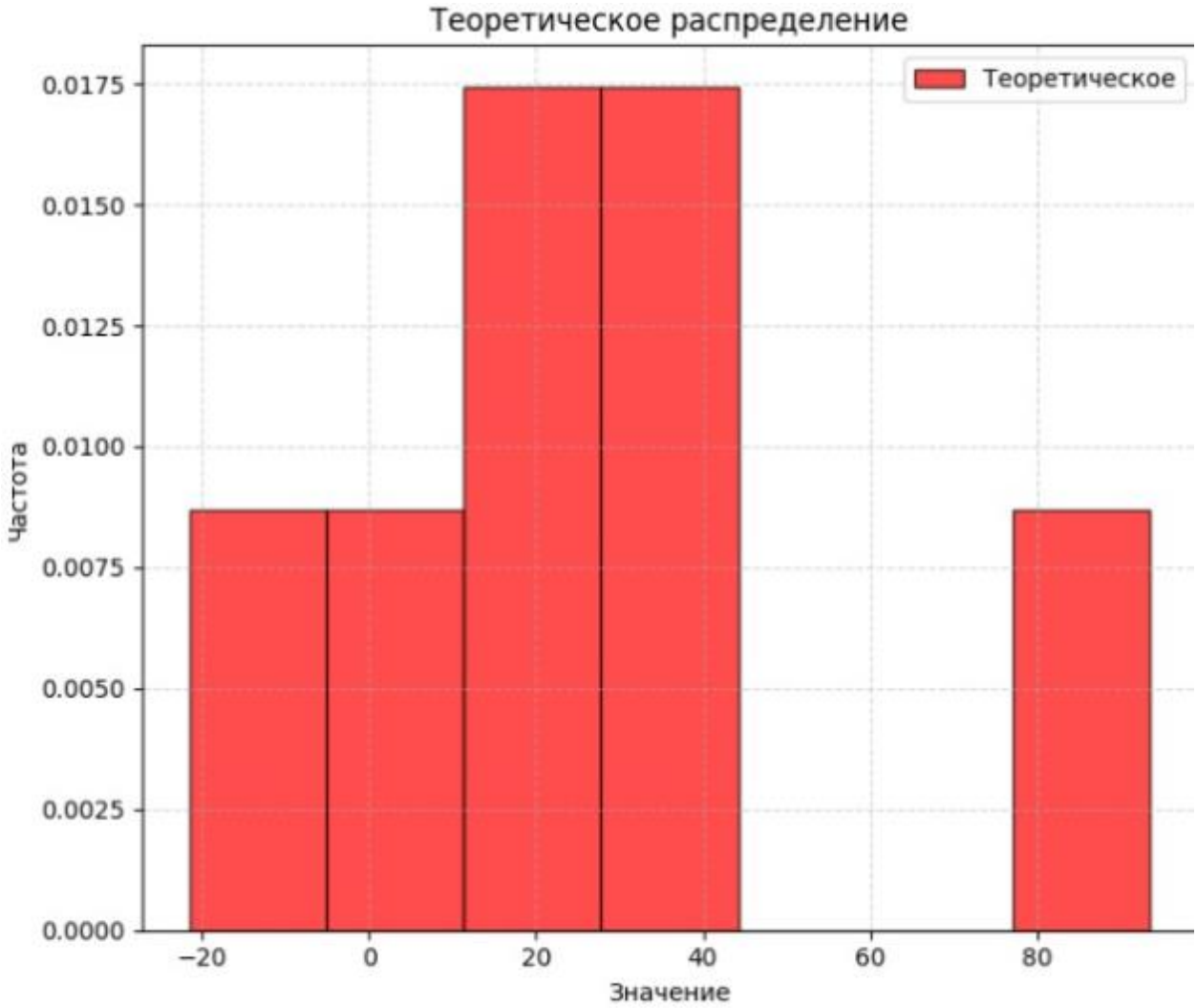

Рисунок 2 – Теоретические значения

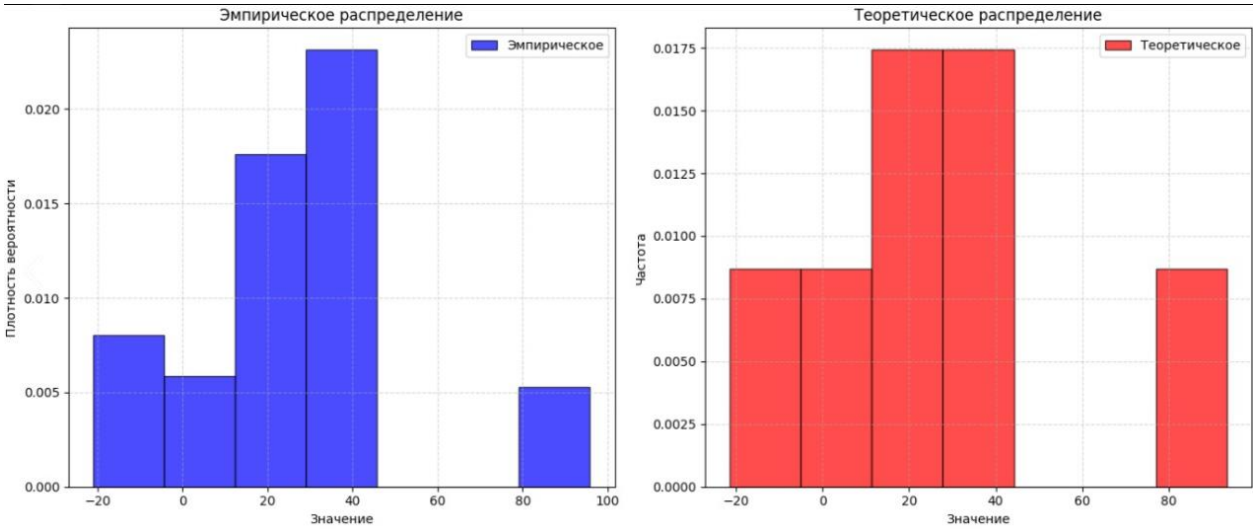

Рисунок 3-Эмпирические и теоретические значения

```
Эмпирическое M: 27.548482579673273
Эмпирическая дисперсия: 811.9053089721314
Теоретическое M: 26.0287
Теоретическая дисперсия: 843.5299263100002
```

Рисунок 4-Результаты эмпирических и теоретических значений M и D

ВЫВОД

В результате анализа выявлено, что генератор дискретных величин, основанный на мультипликативно-конгруэнтном датчике, демонстрирует отклонения от теоретических значений, которые, вероятно, были бы менее заметны при использовании большего размера выборки.

На гистограммах распределения (рис.3) отклонение особенно заметно при сравнении значений x в интервалах от 20 до 40.

Сравнивая эмпирические и теоретические параметры распределений дискретной случайной величины (СВ) x, можно сделать следующие выводы:

1. **Средние значения (M)**:
    - Эмпирическое среднее (27.55) немного выше теоретического значения (26.0287).
    - Это может указывать на некоторые отклонения в распределении, возможно, вызванные случайными факторами или ограничениями в данных.
2. **Дисперсии**:
    - Эмпирическая дисперсия (811.189) ниже теоретической (843.530).
    - Это может указывать на то, что разброс значений в эмпирическом распределении менее выражен, чем в теоретическом распределении.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

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


class PeakGenerator:
    def __init__(self, seed=None):
        self.A = [random.randint(0, 2**32 - 1) for _ in range(55)]
        self.B = [random.randint(0, 2**32 - 1) for _ in range(57)]
        self.C = [random.randint(0, 2**32 - 1) for _ in range(58)]
        self.index_A = 0
        self.index_B = 0
        self.index_C = 0

    def peak(self):
        while True:
            self.A[self.index_A] = (
                self.A[(self.index_A - 55) % 55] + self.A[(self.index_A - 24) %
55]) % (2**32)
            self.B[self.index_B] = (
                self.B[(self.index_B - 57) % 57] + self.B[(self.index_B - 7) %
57]) % (2**32)
            self.C[self.index_C] = (
                self.C[(self.index_C - 58) % 58] + self.C[(self.index_C - 19) %
58]) % (2**32)


            carry_A = self.A[self.index_A] >> 31
            carry_B = self.B[self.index_B] >> 31
            carry_C = self.C[self.index_C] >> 31


            if carry_A == carry_B == carry_C:
                yield self.A[self.index_A] / (2 ** 32), self.B[self.index_B] / (2
** 32), self.C[self.index_C] / (2 ** 32)
            elif carry_A == carry_B:
                yield self.A[self.index_A] / (2 ** 32), self.B[self.index_B] / (2
** 32), self.C[self.index_C] / (2 ** 32)
            elif carry_A == carry_C:
                yield self.A[self.index_A] / (2 ** 32), self.B[self.index_B] / (2
** 32), self.C[self.index_C] / (2 ** 32)
            elif carry_B == carry_C:
                yield self.A[self.index_A] / (2 ** 32), self.B[self.index_B] / (2
** 32), self.C[self.index_C] / (2 ** 32)

            self.index_A = (self.index_A + 1) % 55
```

```
            self.index_B = (self.index_B + 1) % 57
            self.index_C = (self.index_C + 1) % 58


class DiscreteRandomVariable:
    def __init__(self, x_values, probabilities):
        self.x_values = x_values
        self.probabilities = probabilities
        self.cumulative_probabilities = np.cumsum(probabilities)

    def generate(self):
        rand = np.random.random()
        for i, p in enumerate(self.cumulative_probabilities):
            if rand < p:
                return self.x_values[i]

    def mean(self):
        return np.sum(np.array(self.x_values) * np.array(self.probabilities))

    def variance(self):
        mean = self.mean()
        return np.sum([p * x ** 2 for x, p in zip(self.x_values,
self.probabilities)]) - mean ** 2


# Данные
x_values = [-21.4, 4.6, 17.1, 20.1, 37.1, 39.1, 93.4]
probabilities = [0.137, 0.098, 0.065, 0.240, 0.258, 0.108, 0.094]

# Создание дискретной случайной величины
drv = DiscreteRandomVariable(x_values, probabilities)

# Создание генератора пиков
peak_gen = PeakGenerator()

# Шаг 1: Генерация 500 образцов с учетом пиков
sample_size = 500
sample = []
for _ in range(sample_size):
    peak_values = next(peak_gen.peak())
    sample.append(drv.generate() + sum(peak_values))

first_30_values = [drv.generate() for _ in range(30)]
print("Первые 30 значений xi:", first_30_values)
# Шаг 2: Расчет эмпирического среднего и дисперсии
empirical_mean = np.mean(sample)
empirical_variance = np.var(sample)


plt.figure(figsize=(14, 6))
```

```
plt.subplot(1, 2, 1)
plt.hist(sample, bins=7, density=True, alpha=0.7, color='blue',
edgecolor='black', label='Эмпирическое')
plt.title('Эмпирическое распределение')
plt.xlabel('Значение')
plt.ylabel('Плотность вероятности')
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()

plt.subplot(1, 2, 2)
plt.hist(x_values, bins=7,density=True, alpha=0.7, color='red',
edgecolor='black', label='Теоретическое')
plt.title('Теоретическое распределение')
plt.xlabel('Значение')
plt.ylabel('Частота')
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()

plt.tight_layout()
plt.show()


# Шаг 3: Сравнение эмпирического и теоретического распределений

print("Эмпирическое М:", empirical_mean)
print("Эмпирическая дисперсия:", empirical_variance)
print("Теоретическое М:", drv.mean())
print("Теоретическая дисперсия:", drv.variance())
```