Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по Лабораторной работе №3

по курсу «Моделирование»

на тему: «Исследование псевдослучайных чисел»

Студент ИУ7-73Б _____ Миронов Г. А.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Рудаков И. В.
(Подпись, дата) (И. О. Фамилия)

2022 г.

1 Задание

Изучить методы генерации псевдослучайных последовательностей чисел, а так же критерии оценки случайности последовательности.

Реализовать критерий оценки случайности последовательности. Сравнить результаты работы данного критерия для последовательностей одно-, двух- и трехразрядных целых чисел. Последовательности получать алгоритмическим и табличным способами. Предусмотреть ввод собственных значений.

2 Теоретическая часть

2.1 Методы генерации псевдослучайных последовательностей

2.1.1 Линейный конгруэнтный метод

Данный метод использует 4 числа:

- $m > 0$ - модуль;
- $0 \leq a \leq m$ - множитель;
- $0 \leq c \leq m$ - приращение;
- $0 \leq X_0 \leq m$ - начальное число;

Последовательность случайных чисел генерируется следующим образом:

$$X_{n+1} = (aX_n + c) \mod m \tag{2.1}$$

2.1.2 Табличный метод

Табличные методы опираются на заранее составленные таблицы, содержащие некореллированные числа, т.е. числа не зависящие друг от друга.

От содержимого таблицы зависит качество генерируемой последовательности.

2.2 Критерий оценки

В качестве критерия оценки были выбран критерий монотонности, опирающийся на критерий χ^2.

2.3 Критерий χ^2

Это один из самых известных статистических критериев, также это основной метод, используемый в сочетании с дргуими критериями. С помощью этого критерия можно узнать, удовлетворяет ли генератор случайных чисел требованию равномерного распределения или нет. Для оценки по этому критерию необходимо вычислить статистику V по формуле:

$$V = \frac{1}{n} \sum_{s=1}^{k} (\frac{Y_s^2}{p_s}) - n \tag{2.2}$$

где n — количество независимых испытаний, k — количество категорий, Y_s — число наблюдений, которые действительно относятся к категории S, p_s — вероятность того, что каждое наблюдение относится к категории S.

2.4 Критерий монотонности

Критерий применяется для проверки рас- пределения длин монотонных подпоследовательностей в последовательностях вещественных чисел.

Рассмотрим пример. Допустим, у нас есть следующая выборка:

$$0.8, 0.13, 0.5, 0.27, 0.43, 0.85, 0.63, 0.74$$

В данной последовательности найдено 4 отрезка возрастания:

$$[0.8], [0.5], [0.43, 0.85], [0.74]$$

Таким образом, мы имеем три отрезка длиной 1 и один отрезок длиной 2.

Смежные отрезки не являются независимыми, поэтому необходимо «выбросить» элемент, который следует непосредственно за серией. Таким образом, когда U_j больше U_{j+1}, начнем следующую серию с U_{j+2}.

В таком случае, после подсчета количества отрезков возрастания с различной длиной, можем использовать критерий χ^2 со следующими вероятностями:

$$\begin{cases} p_r = \frac{1}{r!} - \frac{1}{(r+1)!}, r < t \\ p_t = \frac{1}{r!}, r \geq t \end{cases} \tag{2.3}$$

3 Результат работы

3.1 Общий случай

В таблице 3.1 приведены исходные данные.

Таблица 3.1 – Исходные данные

	ввод	алг. 0-9	алг. 10-99	алг. 100-999	табл. 0-9	табл. 10-99	табл. 100-999
0	1	5	37	791	5	75	451
500	9	5	53	675	2	88	355
1000	1	3	45	319	4	80	940
1500	9	1	13	335	4	24	367
2000	1	3	29	219	5	77	664
2500	9	9	93	619	7	97	279
3000	1	9	25	599	2	15	268
3500	9	7	95	807	2	77	515
4000	1	7	51	415	3	16	851
4500	9	1	73	611	3	18	662

В таблице 3.2 приведены значения критерия монотонности для соответствующих входных данных.

Таблица 3.2 – Критерий монотонности

	0-9	10-99	100-999
ввод	34.000000	-	-
алг.	1585.342147	1459.579608	1055.301611
табл.	498.246442	1112.232249	1108.830996

На рисурке 3.1 представлены гистограммы входных данных.

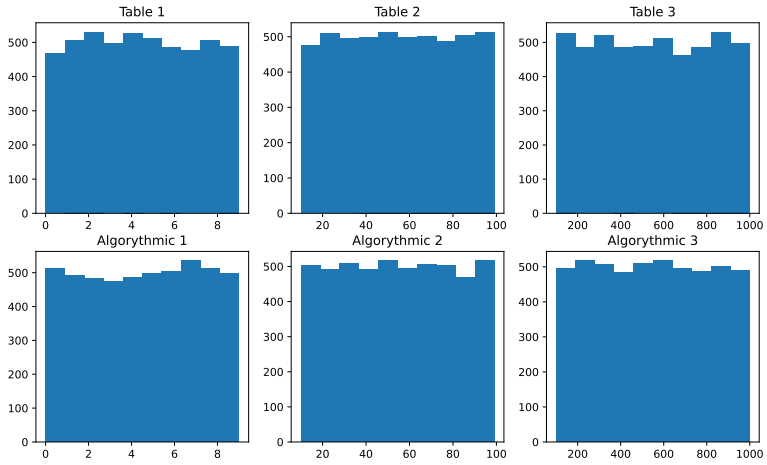

Рисунок 3.1 – Гистограммы

3.2 Отсортированный ввод

Таблица 3.3 – Исходные данные

	алг. 0-9	алг. 10-99	алг. 100-999	табл. 0-9	табл. 10-99	табл. 100-999	
0	1	0	10	100	0	10	100
500	1	0	18	191	1	18	184
1000	1	1	28	277	2	27	275
1500	1	3	36	367	2	36	366
2000	1	4	46	460	4	45	453
2500	9	5	54	547	4	54	547
3000	9	6	63	633	5	63	636
3500	9	7	72	726	6	72	726
4000	9	8	81	818	7	81	819
4500	9	8	91	908	8	90	906

Таблица 3.4 – Исходные данные

	0-9	10-99	100-999
ввод	139.0	-	-
алг.	139.0	139.0	139.0
табл.	139.0	139.0	139.0

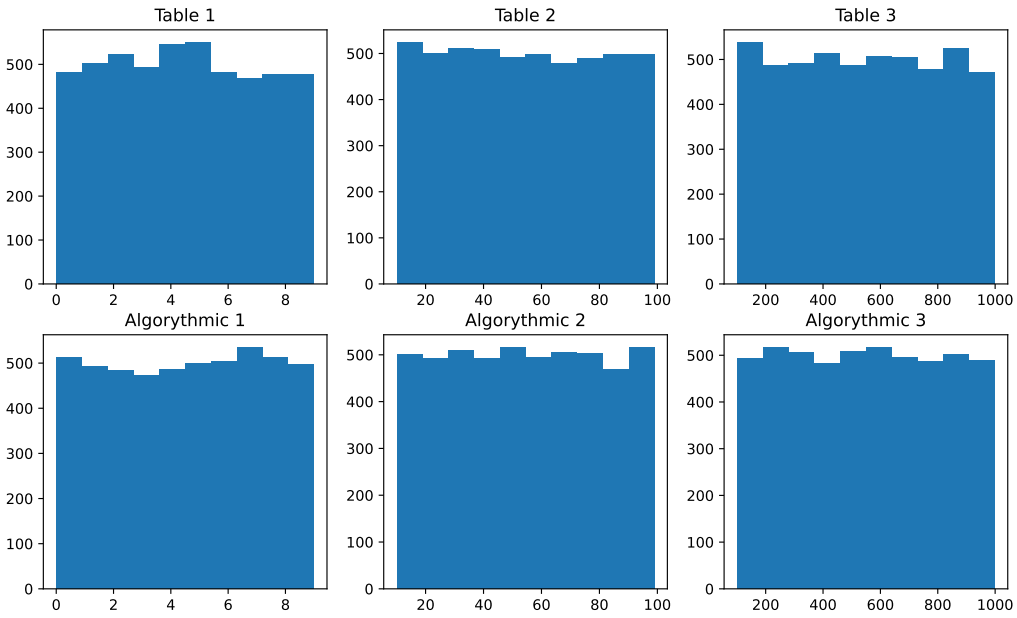

Рисунок 3.2 – Граф, описывающий систему

4 Исходный код программы

Листинг 4.1 – Исходный код программы. Часть 1

```
import os
from itertools import islice

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

NUMBERS_COUNT = 5000

notebook_path = os.path.abspath("main.py")

table = os.path.join(os.path.dirname(notebook_path), "data/numbers.txt")

def read_numbers(filename: str, count: int) -> set[int]:
    numbers = set()
    with open(filename) as file:
        line_num = 0

        for l in islice(file, line_num, None):
            numbers.update(set(l.split(" ")[1:-1]))
            if len(numbers) >= count + 1:
                break
            line_num += 1

        numbers.remove("")
        numbers = list(numbers)[:count]

    return numbers

def table_rand(filename: str, count: int=5000) -> tuple[np.ndarray]:
    numbers = read_numbers(filename, 3*count)

    one_digit = np.array([int(i)%10 for i in numbers[:count]])
    two_digits = np.array([int(i)%90 + 10 for i in numbers[count:count*2]])
    three_digits = np.array([int(i)%900 + 100 for i in numbers[count*2:3*count]])

    return one_digit, two_digits, three_digits
```

Листинг 4.2 – Исходный код программы. Часть 2

```
# Linear congruent method

# one of
# a           c           m
# 4096        150889      714025
# 36261       66037       312500
# 84589       45989       217728

# 1664525     1013904223  2^32
# 22695477    1           2^32
# 1103515245  12345       2^31

class RNG:
    def __init__(self, seed: int = 10):
        self.a, self.c, self.m = 84589, 45989, 217728
        self.current = seed

    def get_number(self, low=0, high=100) -> int:
        self.current = (self.a * self.current + self.c) % self.m
        return int(low + self.current % (high - low))

def alg_rand(seed: int=10, count: int=5000) -> tuple[np.ndarray]:
    random = RNG(seed)

    one_digit = np.array([random.get_number(0, 10) for i in range
       (count)])
    two_digits = np.array([random.get_number(10, 100) for i in
       range(count)])
    three_digits = np.array([random.get_number(100, 1000) for i
       in range(count)])

    return  one_digit, two_digits, three_digits

def calc_hi(cnt: np.ndarray, n: int) -> int:
    r = 0
    for i in range(len(cnt)):
        if i == len(cnt) - 1:
            p = (1/np.math.factorial(i+1))
        else:
            p = (1/np.math.factorial(i+1) - 1/np.math.factorial(i
               +1+1))

        r += cnt[i]**2 / p

    r = r / n - n

    return r
```

Листинг 4.3 – Исходный код программы. Часть 3

```
84  def monotonicity_criterion(arr: np.ndarray):
85      tabs = np.zeros(6, dtype='int64')
86
87      i, length = 0, 1
88      while i < len(arr):
89          if (i == len(arr) - 1) or (arr[i] > arr[i+1]):
90              tabs[5 if length >= 6 else length-1] += 1
91
92              i, length = i+1, 0
93
94          i += 1
95          length += 1
96
97      n = sum(i * tabs[i] for i in range(len(tabs)))
98      return calc_hi(tabs, n)
99
100 if __name__ == "__main__":
101     tbl = table_rand(table, NUMBERS_COUNT)
102
103     alg = alg_rand(seed=150, count=NUMBERS_COUNT)
104
105     io = [1, 9, 1, 9, 1, 9, 1, 9, 1, 9]
106
107     data = pd.DataFrame()
108
109     data["алг. 0-9"] = alg[0]
110     data["алг. 10-99"] = alg[1]
111     data["алг. 100-999"] = alg[2]
112     data["табл. 0-9"] = tbl[0]
113     data["табл. 10-99"] = tbl[1]
114     data["табл. 100-999"] = tbl[2]
115
116     print(data[::500])
117
118     data = pd.DataFrame()
119     data.loc["ввод", "0-9"] = monotonicity_criterion(io)
120     data.loc["ввод", "10-99"] = data.loc["ввод", "100-999"] = "-"
121
122     data.loc["алг.", "0-9"] = monotonicity_criterion(alg[0])
123     data.loc["алг.", "10-99"] = monotonicity_criterion(alg[1])
124     data.loc["алг.", "100-999"] = monotonicity_criterion(alg[2])
125     data.loc["табл.", "0-9"] = monotonicity_criterion(tbl[0])
126     data.loc["табл.", "10-99"] = monotonicity_criterion(tbl[1])
127     data.loc["табл.", "100-999"] = monotonicity_criterion(tbl[2])
128
129     print(data)
```

Листинг 4.4 – Исходный код программы. Часть 4

```
131     fig, axis = plt.subplots(2, 3, figsize=(12,7))
132 
133     for i in range(len(tbl)):
134         axis[0][i].set_title(f"Table␣{i+1}")
135         axis[0][i].hist(tbl[i])
136 
137     for i in range(len(alg)):
138         axis[1][i].set_title(f"Algorythmic␣{i+1}")
139         axis[1][i].hist(alg[i])
140 
141     plt.show()
```