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 ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение эвм и информационные технологии»

ОТЧЕТ

По лабораторной работе №3

«Случайные числа»

По курсу «Моделирование»

Студент Т. А. Казаева

Группа ИУ7-76Б

Преподаватель И. В. Рудаков

2022 г.

1. Задание

Написать программу, которая генерирует псевдослучайную последовательность одноразрядных, двухразрядных и трёхразрядных целых чисел с использование табличного и алгоритмического способа.

Для каждой сгенерированной последовательности чисел вычислить критерий оценки случайности.

2. Математическая формализация

Существует три способа получения последовательности случайных чисел:

- аппаратный способ;
- табличный (файловый) способ;
- алгоритмический способ.

При использовании аппаратного способа случайная величина вырабатывается специальной электрической приставкой (генератор случайных чисел) как правило внешнее устройство компьютера не требует других устройств и операций кроме обращения к устройству.

Если случайные числа, оформленные в виде таблицы, помещать во внешнюю или оперативную память ЭВМ, предварительно сформировав из них соответствующий файл, то такой способ будет называться табличным. Однако, хранение файла во внешней памяти при частном обращении в процессе статистического моделирования не рационально, так как вызывает увеличение затрат машинного времени при моделировании системы из-за необходимости обращения к внешнему накопителю.

Алгоритмический способ – это способ получения последовательности случайных чисел, основанный на формировании случайных чисел в ЭВМ с использованием специальных алгоритмов и реализующих их программ. В качестве используемого метода генерации последовательности случайных чисел был выбран линейный конгруэнтный метод. Вычисление последовательности случайных чисел происходит следующим образом (2.1):

$$X_{n+1} = (a \cdot X_n + c) \; mod \; m, \tag{2.1}$$

где X_{n+1} – это следующее число в последовательности, a – множитель, причём $0 \leq a \leq m$, c – приращение, причём $0 \leq c \leq m$, m – натуральное число, относительно которого вычисляется остаток от деления, причём $m \geq 2$.

При выборе значения m необходимо учитывать следующие условия:

- данное число должно быть довольно большим, так как период не может иметь более m элементов;
- данное значение должно быть таким, чтобы случайные значения вычислялись быстро.

В качестве констант в данной работе используются следующие значения: $m = 36261,\ a = 66037,\ c = 312500$.

Для реализации табличного способа заранее подготовлено три таблицы с одноразрядными, двухразрядными и трехразрядными числами.

Для проверки случайности был использован собственный критерий, который формируется следующим способом:
Пусть на вход алгоритму представлена последовательность (2):

$$a = a_1, a_2, a_3, \cdots, a_n. \tag{2.2}$$

С помощью этой последовательности вычисляется следующая последовательность 2.3:

$$\Delta = \Delta_1, \Delta_2, \Delta_3, \cdots, \Delta_m, \text{ где } \Delta_i = a_{i+1} - a_i,\ m = n - 1. \tag{2.3}$$

Для последовательности Δ вычисляется коэффициент вариации 2.4:

$$V = \frac{\sigma}{\bar{\Delta}}, \tag{2.4}$$

где дисперсия вычисляется согласно 2.5:

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\Delta_i - \bar{\Delta}\right)^2}, \tag{2.5}$$

а выборочное среднее согласно 2.6:

$$\overline{\Delta} = \frac{1}{m}\sum_{i=1}^{m}\Delta_i. \tag{2.6}$$

3. Результат

На рисунке 3.1 представлен графический интерфейс приложения.

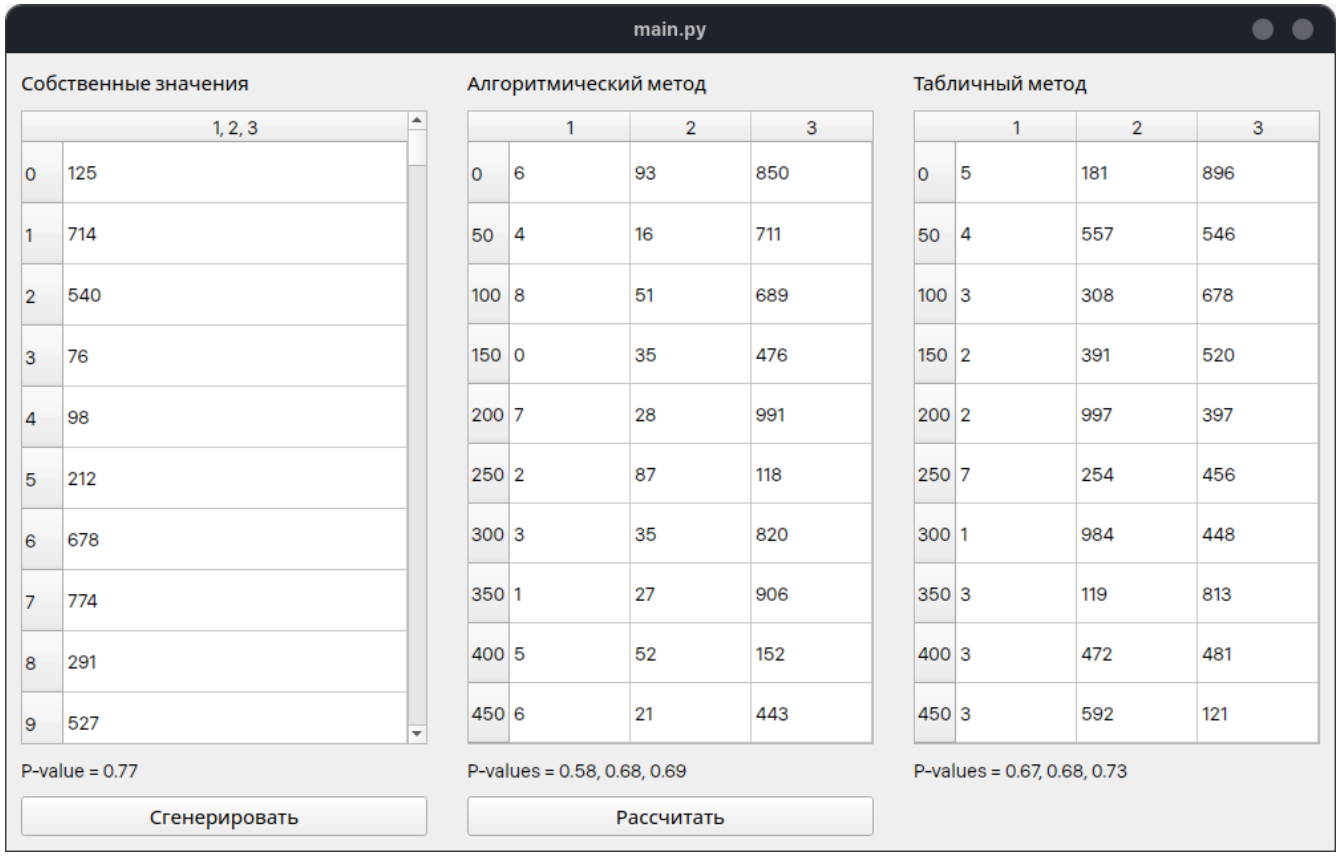

Рис. 3.1: Графический интерфейс приложения

Программа генерирует последовательность размера 500 чисел. Заголовки столбцов в таблице обозначают разрядность чисел, представленных в этом столбце. Заголовки строк – номер числа в последовательности. Для алгоритмического и табличного метода значения выводятся с шагом 50. Для ручного ввода выводятся все значения. Таблица для ручного ввода доступна к редактированию.

Кнопка «Сгенерировать» заполняет таблицу случайно сгенерированными числами. Кнопка «Рассчитать» выводит на экран значения критериев оценки случайности для каждого столбца таблицы слева направо.

Получившиеся значения представлены под таблицами. Как можно заметить, процент случайности растет с увеличением количества разрядов.

4. Программный код

Для реализации программы был выбран язык Python.

На листинге 4.1 представлена реализация линейного конгруэнтного метода.

Листинг 4.1: Релаизация линейного конгруэнтного метода

```
class Congruential():
    def __init__(self):
        self.cur = 10
        self.m = 36261
        self.a = 66037
        self.c = 312500

    def Generate(self, lo=0, hi=9, n=500):
        res = []
        for i in range(n):
            self.cur = (self.a * self.cur + self.c) % self.m
            num = int(lo + self.cur % (hi - lo))
            res.append(num)
        return res
```

На листинге 4.2 представлена реализация табличного метода.

Листинг 4.2: Релаизация табличного метода

```
from pathlib import Path

class BuiltinTable():
    def __init__(self):
        self.dataFolder = Path("rand/meta/")
        self.oneDigitFilename = 'onedigit.txt'
        self.twoDigitsFilename = 'twodigits.txt'
        self.threeDigitsFilename =  "threedigits.txt"
    def Generate(self, digits = 1):
        if digits == 1:
            fileToOpen = self.dataFolder / self.oneDigitFilename
        elif digits == 2:
            fileToOpen = self.dataFolder / self.twoDigitsFilename
        else:
```

```
15              fileToOpen = self.dataFolder / self.threeDigitsFilename
16
17          f = open(fileToOpen, 'r')
18
19          nums = []
20          for line in f:
21              nums.append(int(line.strip('\n')))
22
23          return nums
```

На листинге 4.3 представлена реализация собственного критерия проверки случайности.

Листинг 4.3: Реализация собственного критерия проверки случайности

```
1  def getDeltas(a):
2      deltas = []
3      for i in range(len(a) - 1):
4          deltas.append(abs(a[i + 1] - a[i]))
5      return deltas
6
7  def variance(data):
8      n = len(data)
9      mean = sum(data) / n
10     deviations = [(x - mean) ** 2 for x in data]
11     variance = sum(deviations) / n
12
13 return variance
14
15 def randomnessTest(a):
16     deltas = getDeltas(a)
17     n = len(deltas)
18     sample = 1 / n * sum(deltas)
19
20     if sample == 0:
21     return 0.0
22
23     Vkoef = math.sqrt(variance(deltas)) / sample
24     return Vkoef
```