1830

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»

Тема: «Генераторыпсевдослучайных чисел»

Студент: Ле Ни Куанг

Группа: ИУ7И-76Б

Преподаватель: Рудаков И. В.

Москва

1 Задание

Изучить и реализовать генератор псевдослучайных чисел программным и табличным методом. Разрядность чисел должна быть равна 1, 2, 3. Сравнить методы по определенному критерию и сделать выводы.

2 Теоритическая часть

2.1 Программный генератор

Программный генератор формирует псевдослучайные числа. Каждое последующее число в такой последовательности зависит от предыдущего. Сгенерированные числа статистически случайны, несмотря на то, что это полностью детерминированный и повторяемый процесс.

2.1.1 Линейный конгруэнтный метод

Генератор определяется рекуррентным соотношением:

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

где: m > 0, модуль

0 <= a <= m, множитель

0 <= c <= m, приращение

 $0 <= X_0 <= m$, начальное число

2.2 Табличный генератор

Табличный генератор использует таблицу проверенных некоррелированных цифр в качестве источника случайных чисел.

2.3 Критерий равномерности

Пусть имеется последовательность целых чисел X_n , $0 <= X_i < d$.

При оценке равномерности для каждого r (0 <= r < d) подсчитывается количество случаев, когда элемент последовательности $X_i = r$.

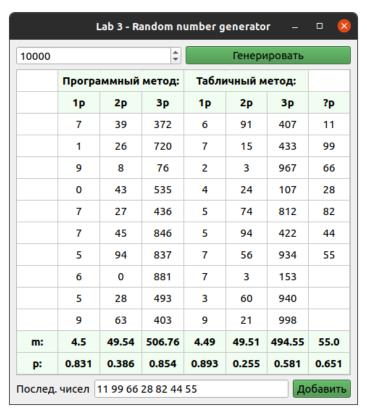
После этого применяется критерий χ^2 , в котором вычисляется статистика:

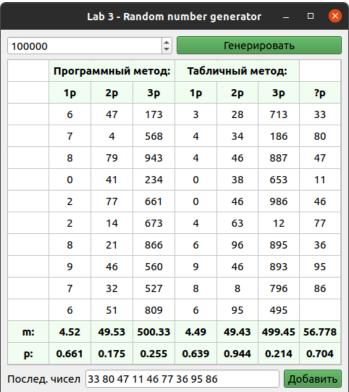
$$\chi^2 = \sum_{i=0}^{k-1} \frac{(x_i - m_i)^2}{m_i} \sim \chi^2(k-1)$$

Где $m_i = np_i = n/d$

Рассчитать значение p для уровня погрешность $\alpha = 0.05$ допускать 0.05 .

3 Результаты





При уровня погрешность $\alpha=0.05,\,0.05 в большинстве случаев. Чем больше количество генерируемых случайных чисел, тем равномернее они распределены. Но для большого количества чисел какая-то закономерность повторяется.$

4 Листинг кода

Листинг 1 — рограммная реализация генерации псевдослучайных чисел программным и табличным методом

```
import enum
from datetime import datetime
from random import randrange
from scipy.stats import chisquare
class RandomGenerator:
    class Method(enum.Enum):
        Standard = 1
        Table = 2
        LinearCongruential = 3
    def __init__(self, table_path='digits.txt'):
        self.digits = ''
        with open(table_path) as table:
            for line in table:
                self.digits += line[:-1]
        self.table_len = len(self.digits)
        self.seed = int(datetime.now().microsecond)
        self.table_idx = self.seed % self.table_len
        self.table_idx = 0
    def __standard(self, n_digits):
        return randrange(10 ** n_digits)
    # https://en.wikipedia.org/wiki/Linear_congruential_generator
    def __linear_congruential(self, n_digits, modulus=2e31, a=1664525,
       c=1013904223):
        self.seed = (a * self.seed + c) % modulus
        return int(self.seed / modulus * (10 ** n_digits))
    def __table(self, n_digits):
        n_{11}m = 
        for _ in range(n_digits):
            num += self.digits[self.table_idx]
            self.table_idx = (self.table_idx + 1) % self.table_len
        return int(num)
    def rand(self, method: Method, n_digits, n_numbers):
        res = []
        if method == RandomGenerator.Method.Standard:
```

```
res = [self.__standard(n_digits) for _ in range(n_numbers)]
        elif method == RandomGenerator.Method.Table:
            res = [self.__table(n_digits) for _ in range(n_numbers)]
        elif method == RandomGenerator.Method.LinearCongruential:
            res = [self.__linear_congruential(n_digits) for _ in
               range(n_numbers)]
        return res
class RandomCriteria:
    @staticmethod
    def p_of_chi_square_test(sequence, n_digits):
        d = 10 ** n_digits
        f_{obs} = [0] * d
        for i in sequence:
            f_{obs[i]} += 1
        # Got same result: y == chisq
        # y = 0
        # mi = len(sequence) / d
        # for i in range(d):
        # y += (f_obs[i] - mi) ** 2
        # y /= mi
        chisq, p = chisquare(f_obs)
        return p
```