Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №4**

По дисциплине: «Компьютерные системы моделирования»

Тема: " Имитационное моделирование. Моделирование частотной характеристики. Моделирование случайных событий"

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС-22.06  Соболев Р.В. |
| Проверила:  старший преподаватель  кафедры ПО  Корнеева Е.И. |

Тверь, 2025

**Содержание**

[**Цель и постановка задачи для варианта.** 3](#_Toc191257819)

[**Описание алгоритмов.** 4](#_Toc191257820)

[**Отчет по реализации.** 5](#_Toc191257821)

**Заключение**………………………………………………………………………..14

# **Цель и постановка задачи для варианта.**

**Цель работы**: «Изучение методов и алгоритмов моделирования случайных чисел».

**Постановка задачи:**

# Задание на лабораторную работы

Согласно варианту задания необходимо составить и отладить программу (подпрограмму) генерирования случайных чисел:

1. с равновероятным распределением на интервале [0; 1), используя формулу
2. используя готовый пакет Google | Excel надстройку Анализ данных и библиотеку numpy алгоритм из объекта random.

Провести статистическое исследование генератора при различных значениях выборки: малых n<25, средних , больших n>500.

**Вариант:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вариант** | **Тип генератора** | **Начальные данные** | **Метод в numpy**  **(тип распределения в numpy.random)** | **Метод в Google|Excel**  **(надстройка ГСЧ)** |
| 2 | Линейный конгруэнтный метод. Формула в теории (3). | *а*=16070093; *b*=453816693; *k*=31 | Бета- распределение | Нормальное |

# **Описание алгоритмов.**

***Линейный конгруэнтный метод***

Последовательность случайных чисел  получается с помощью следующего итерационного соотношения:

, (1)

где  и  - очередное и последующее случайное число;

*m* - модуль, *m* > 0;

*a*  -  множитель, 0 ≤ *a*< *m*;

 - начальное значение,  .

Если *m* и *а* являются целыми, то создается последовательность целых чисел в диапазоне . Из рекуррентной природы формулы следует, что при одном и том же значении  мы получим при повторной генерации ту же самую последовательность.

Последовательность случайных чисел периодически повторяются. Это связано с тем, что числа  могут принимать только значения . То есть, самое большее через  шагов уже один раз полученное число должно появиться опять, а с ним повторяется и вся последовательность. Таким образом, длина периода при модуле  не может превышать . Поэтому выбор значений *а* и *m* является критичным для разработки хорошего генератора случайных чисел. Для практических расчетов принимают .

Последовательность чисел , равномерно распределенных на интервале от нуля до единицы, рассчитывается по формуле

. (2)

# **Отчет по реализации.**

Схема метода предложена Д. Лехмером в 1949 году. Работа этих генераторов основана на использовании формулы:

. (3)

Число *m* выбирается аналогично предыдущему пункту. Множитель, *а* предпочтительно выбирать в интервале [0,01*m*; 0,99*m*]. Значение *с* может быть произвольным, но не должно иметь общего множителя с *m*. При *b*=0 получаем, рассмотренный выше мультипликативный метод.

С вычислительной точки зрения линейный метод генерации последовательности неотрицательных целых чисел сложнее мультипликативного на одну операцию сложения, но при этом возможность выбора дополнительного параметра позволяет уменьшить возможную корреляцию получаемых чисел. Однако экспериментальная проверка качества генерируемой последовательности чисел на основе этой формулы является сложнее.

**Программная реализация**

class LinearCongruentialGenerator:

    def \_\_init\_\_(*self*, *k*, *a*, *b*, *x0*):

        self.k = k

        self.a = a        *# Множитель*

        self.b = b        *# Приращение*

        self.m = 2\*\*k    *# Модуль (используем 2^k для эффективности)*

        self.x = x0       *# Начальное значение*

    def next(*self*):

*# Генерируем следующее случайное число по линейному конгруэнтному методу*

        self.x = (self.a \* self.x + self.b) % self.m

*return* self.x / self.m  *# Нормализуем к [0, 1)*

    def next(self):

        # Генерируем следующее случайное число по формуле

        self.x = (self.a \* self.x) % self.m

        return self.x / self.m

**Бета-распределение**

import numpy as np

class BetaGenerator:

def \_\_init\_\_(self, size=1, alpha=1.0, beta\_param=1.0):

self.size = size

self.alpha = alpha

self.beta\_param = beta\_param

def generate(self):

# Генерация случайных чисел из бета-распределения

return np.random.beta(self.alpha, self.beta\_param, self.size)

**Реализация Excel:**

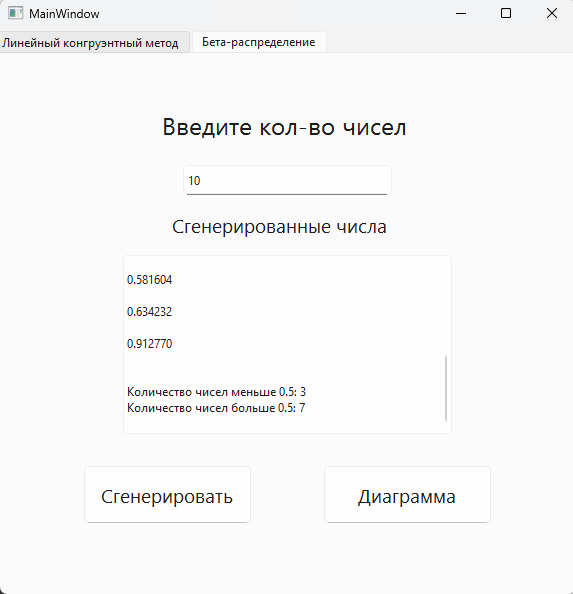
**Изображение выглядит как текст, число, меню, документ

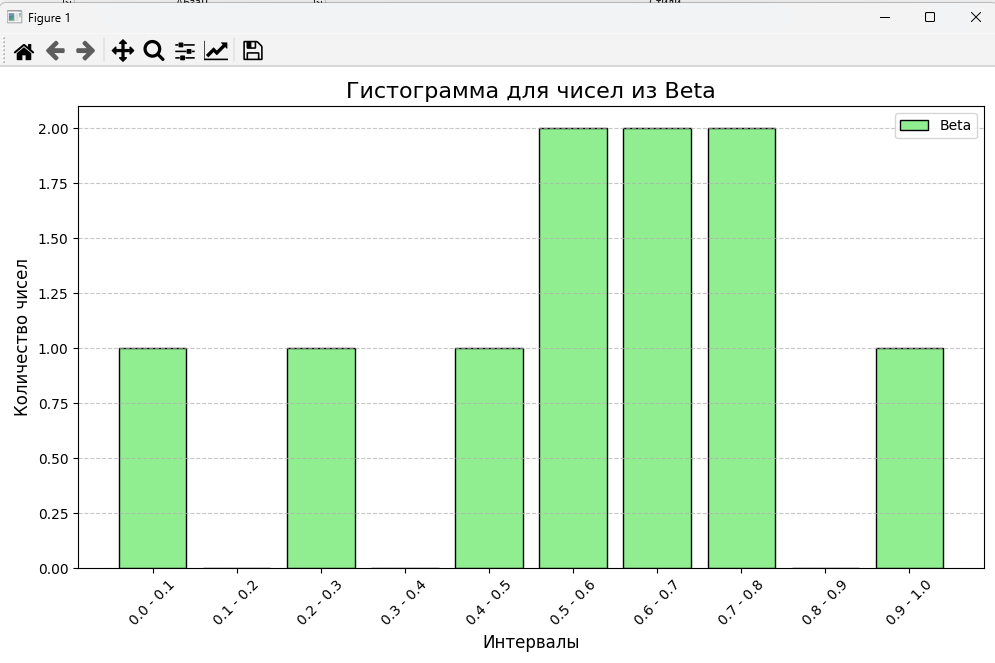
Контент, сгенерированный ИИ, может содержать ошибки.**

**Пример вывода программы**

**При n=10**

Бета-распределение:





Линейный конгруэнтный метод:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.

Анализ данных excel:

Вернее число – числа меньше 0.5

Нижнее число – числа больше 0.5

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Анализ:**  
При небольшом числе сгенерированных значений (n=10) оба метода показывают схожие результаты, с небольшим перевесом чисел больше 0.5 для Бета-распределения и для Линейного Конгруэнтного метода. Распределение чисел в таких малых выборках еще не является равномерным.

**При n=150**

Бета-распределение:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Количество чисел меньше 0.5: 72

Количество чисел больше 0.5: 78

Линейный конгруэнтный:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Количество чисел меньше 0.5: 79

Количество чисел больше 0.5: 71

Анализ данных excel:

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.**

**Анализ:**  
С увеличением выборки (n=150) распределение чисел становится более равномерным. В методе с Бета-распределением наблюдается схожая ситуация с нормальным распределением и линейным конгруэнтным методотом, но в целом результаты близки к теоретическому распределению. Нормальное распределение выдало идеальное распределение!

**При n=1000**

Бета-распределение:

Больше 0.5 = 524

Меньше 0.5 = 476

Линейный конгруэнтный метод:

Больше 0.5 = 503

Меньше 0.5 = 497

Анализ данных excel:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Анализ:**  
С увеличением выборки до n=1000 наблюдается практически идеальное распределение для обоих методов. В линейном конгруэнтном методе все равно остаются колебания, однако в алгоритме с бета-распределением и данными из Excel результаты совпали, а также различия между числами больше и меньше 0.5 минимальны, что свидетельствует о хорошей случайности чисел.

**Общие выводы:**

1. **Бета-распределение**:
   * Показало довольно равномерное распределение при увеличении выборки.
   * В небольших выборках (n=20) возможны небольшие отклонения, но при n=150 и выше результаты близки к ожидаемым (50% больше 0.5 и 50% меньше 0.5).
2. **Линейный конгруэнтный метод**:
   * Показал некоторые отклонения от равенства, особенно при малых значениях выборки, но в целом результаты также близки к равномерному распределению.
   * В больших выборках результаты были близки к теоретическому распределению, но с небольшими отклонениями.
3. **Анализ данных Excel**:
   * Данные из Excel подтверждают теоретическую модель случайных чисел, но также могут показывать небольшие отклонения из-за случайных факторов или ограничений методов анализа.

**Заключение**:  
Методы генерации случайных чисел, как метод с бета-распределением, так и алгоритм линейного конгруэнтного метода, показали хорошие результаты, особенно при увеличении размера выборки. В небольших выборках возможны отклонения от равномерного распределения, но при n ≥ 150 результаты становятся близкими к теоретическим, что подтверждает эффективность этих алгоритмов для генерации случайных чисел.