



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»

Лабораторная работа №3 по курсу «Моделирование»

Тема Случайные числа

Студент Прянишников А. Н.

Группа ИУ7-75Б

Оценка (баллы) _____

Преподаватель Рудаков И. В.

Москва — 2022 г.

Условие лабораторной работы

Требуется реализовать ПО, позволяющее генерировать алгоритмическим методом последовательность случайных чисел и проверять итоговую последовательность на случайность по любому критерию. Также нужно добавить пользователю возможность генерировать последовательность из однозначных чисел и реализовать генерацию табличным методом.

Теоретическая часть

В этом разделе будет приведено описание методов генерации последовательности случайных чисел и описан критерий проверки последовательности на случайность.

Виды генераторов случайных чисел

Всего можно выделить четыре типа генераторов случайных чисел:

1. **Аппаратные генераторы** используют результаты определённых физических процессов для создания требуемой последовательности. Аппаратный генератор случайных чисел состоит из источника энтропии и устройства, преобразующего значения, полученные с источника энтропии, в нужный формат.

К такому типу относятся генераторы, основанные на фотоэффекте или тепловом шуме при работе полупроводникового диода. На выходе получается последовательность, обладающая значительной степенью случайности, но у таких генераторов есть два недостатка: системы трудно реализовать в жизни, а процессов, позволяющие преобразовать энтропию в последовательность.

2. **Алгоритмические генераторы** основаны на фиксированных алгоритмах, которые, в зависимости от некоторых физических параметров (например,

содержимого ввода/вывода), выдают нужный результат. Подобные алгоритмы имеют программную реализацию и используются в коммерческом ПО.

3. **Табличные генераторы** принимают на вход уже готовую последовательность, обладающую свойством случайности, после чего проводят различные манипуляции с ней (комбинирование, перемешивание), и выдают результат. К недостаткам этого подхода можно отнести лишнее использование памяти, предопределённость значений и ограниченность последовательности.

Выбранный алгоритмический метод

В качестве алгоритмического метода генерации случайной последовательности выбран метод **линейной конгруэнтной последовательности**.

Для осуществления генерации чисел данным методом, необходимо задать 4 числа: модуль $m > 0$, множитель $a \in [0, m]$, приращение $c \in [0, m]$ и начальное число $x_0 \in [0, m]$

Последовательность случайных чисел генерируется рекуррентно при помощи формулы 1:

$$X_{n+1} = (a * X_n + c) \bmod m \quad (1)$$

Одним из требований к линейным конгруэнтным последовательностям является как можно большая длина периода. Длина периода зависит от значений m, a, c . Линейная конгруэнтная последовательность, определенная числами m, a, c, x_0 , имеет период длиной M тогда и только тогда, когда:

- числа m и c взаимно простые;
- $a - 1$ кратно p для каждого простого p , являющегося делителем m ;
- $a - 1$ кратно 4, если m кратно 4.

Выбранный критерий определения случайности

В качестве критерия был выбран критерий Бартерса.

Пусть H_0 – гипотеза о том, что последовательность обладает свойством случайности. На этом этапе требуется выбрать также уровень значимости α . На основе выборки X создаётся вариационный ряд X_1 , а затем рассчитывается ряд R , где R_i – количество вхождений элемента в последовательность.

После предварительных расчётов вычисляется эмпирический коэффициент B по формуле 2:

$$B = \frac{\sum_i^{n-1} (R_i - R_{i+1})^2}{\sum_i^n (R_i - R_{mean})^2} \quad (2)$$

Затем, исходя из α и таблицы 1, требуется выбрать коэффициенты a, b, c, d .

Таблица 1: Таблица выбора коэффициентов

α	0.01	0.025	0.05	0.1
a	-0.023	-0.004	0.119	-0.465
b	0.261	0.381	0.440	1.184
c	-0.345	-0.266	-0.230	-0.088
d	2.212	1.748	1.520	0.674

Для того, чтобы определить, случайна ли последовательность, требуется рассчитать ещё один коэффициент B_α по формуле 3

$$B_\alpha = a + bn^c(\ln n)^d \quad (3)$$

Если коэффициент B вошёл в интервал $[(2 - B_\alpha), 2 + B_\alpha]$, то принимается решение об истинности гипотезы H_0 .

В качестве величины случайности было выбрано значение $(B * (2 - B_\alpha)) * ((2 + B_\alpha) * B)$. Чем больше это число, тем выше величина случайности.

Реализация

В этом разделе будут приведены листинги кода реализации алгоритмов, продемонстрирована работа программы и построены таблицы с результатами.

Листинги кода

Для реализации ПО был использован язык Python, так как был использована библиотека `numpy`, обладающая значительными возможностями в генерации случайных чисел и манипулирования большим объемом данных.

На листинге 1 представлена реализация алгоритма генерации табличной последовательности.

На листинге 2 представлена реализация алгоритма генерации алгоритмической последовательности.

На листинге 3 представлена реализация критерия проверки последовательности на случайности.

Листинг 1: Реализация алгоритма генерации табличной последовательности

```
def start_gen_sequence(a = 1, b = 999, seqsize = 1000):  
    return np.random.choice(np.arange(a, b, 1), size=seqsize)  
  
def single_table(sequence):  
    np.random.shuffle(sequence)  
    yield sequence  
  
def generate_table_sequence(a = 1, b = 999, seqsize = 1000):  
    X = start_gen_sequence(a, b, seqsize)  
    np.random.shuffle(X)
```

Листинг 2: Реализация алгоритма генерации алгоритмической последовательности

```
X = []  
tmp = x0  
for i in range(seqsize):  
    X.append(tmp)  
    tmp = (tmp*a + c) % m
```

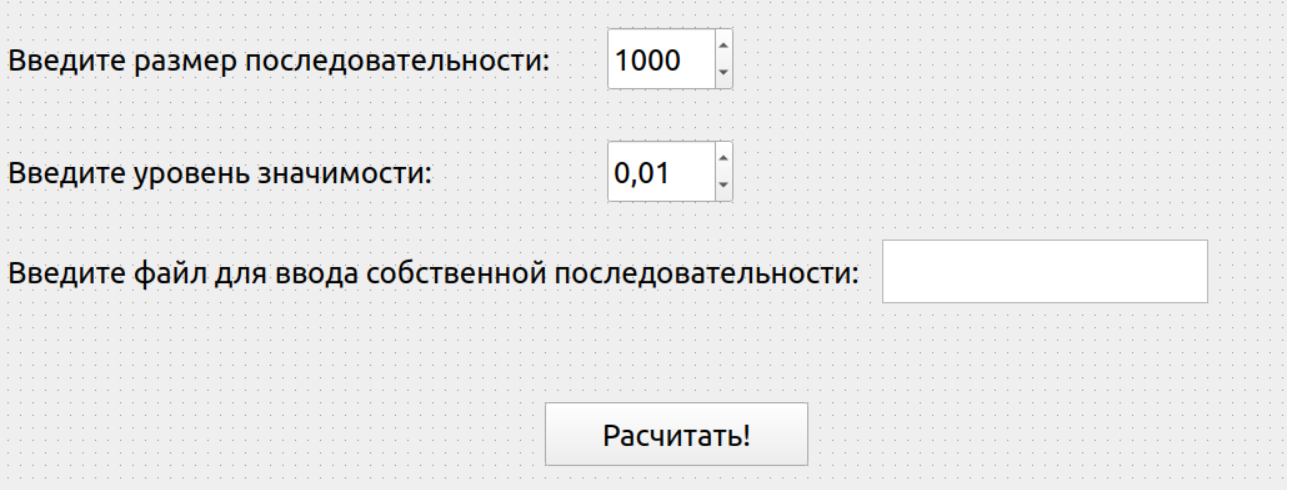
Листинг 3: Реализация критерия проверки последовательности на случайности

```
def burton_critery(x, a, b, c, d):
    unique, counts = np.unique(x, return_counts=True)
    mean = np.mean(counts)
    numerator = 0
    denominator = 0
    n = counts.shape[0]
    for i in range(n - 1):
        numerator += np.power((counts[i] - counts[i + 1]), 2)
        denominator += np.power(counts[i] - mean, 2)
    denominator += np.power(counts[-1] + 1 - mean, 2)
    result = numerator / denominator
    ba = a + b*np.power(n, c)*np.power(np.log(n), d)
    return (result - (2 - ba)) * ((2 + ba) - result)
```

Демонстрация работы программы

В ходе выполнения программы пользователю доступно меню, в котором он может выбрать размер последовательности, который нужно сгенерировать, уровень значимости, а также файл, если он хочет также проверить собственную последовательность.

На рисунке представлена демонстрация главного меню программы:



Введите размер последовательности:

Введите уровень значимости:

Введите файл для ввода собственной последовательности:

Рисунок 1: Демонстрация работы программы

Полученные результаты

Тестирование проводилось на последовательностях разной длины для каждого из трёх способов с уровнем значимости $\alpha = 0.1$. Полученные результаты представлены в таблицах 2-4:

Таблица 2: Таблица полученных значений для табличного способа

Размер	Последовательность	Величина случайности	Случайна?
5	103 436 861 271 107	-0.006	Нет
25	872 344 492 309 467 21 107 331 615 271 701 131 861 664 459 88 373 122 100 770 215 103 662 436 72	1.207	Да
50	-	1.831	Да
100	-	2.354	Да
1000	-	1.748	Да

Таблица 3: Таблица полученных значений для алгоритмического способа

Размер	Последовательность	Величина случайности	Случайна?
5	7, 80, 883, 707, 773	0	Да
25	7, 80, 883, 707, 773, 498, 476, 234, 575, 322, 542, 960, 553, 80, 883, 707, 773, 498, 476, 234, 575, 322, 542, 960, 553	-1.596	Нет
50	-	0.503	Да
100	-	0.241	Да
1000	-	0.09	Да

Таблица 4: Таблица полученных значений для ручного способа

Размер	Последовательность	Величина случайности	Случайна?
5	1, 7, 9, 5, 7	-0.006	Нет
25	1, 2, 9, 3, 3, 5, 4, 8, 9, 1, 3, 8, 1, 5, 4, 9, 5, 1, 2, 3, 6, 9, 2, 3, 5	0.382	Да
50	-	-0.212	Нет

Выводы

В случае малого размера последовательности величина случайности последовательности близка к нулю, поэтому выводы можно сделать только на большом объёме последовательности. Табличный метод показал самые высокие показатели, все последовательности которой прошли порог, начиная с $N = 25$. В случае алгоритмического метода на $N = 25$ получилось не случайная последовательности, и на других N величина случайности оказалась меньше, чем в табличном методе. Проблема связана с тем, что по условию задачи нельзя генерировать числа, большие 1000, соответственно, под такой маленький диапазон нет подходящих параметров под метод линейной конгруэнтной последовательности.