Федеральное государственное бюджетное образовательное учреждение
высшего образования «Сибирский государственный университет
телекоммуникаций и информатики»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине «Моделирование»

Выполнил:

Студент гр. ИВ-622

Свиридов В.О.

Проверила:

Ассистент Кафедры ВС

Петухова Я.В.

Новосибирск 2020

СОДЕРЖАНИЕ

ПОСТАНОВКА ЗАДАЧИ

Реализовать:

1. Непрерывное распределение методом отбраковки;
2. Дискретное распределение с возвратом;
3. Дискретное распределение без возврата.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Непрерывные распределения

Существует несколько методов генерации независимых случайных величин с заданным законом распределения. Наиболее точные из них основаны на преобразовании случайных величин. Так, большое количество датчиков получается исходя из известного результата о равномерном на [0, 1) распределении функции $F_\eta(\eta)$, где η — произвольная непрерывная случайная величина с функцией распределения $F_\eta(x)$.

Метод отбраковки

В некоторых случаях требуется точное соответствие заданному закону распределения при отсутствии эффективных методов генерации. В такой ситуации для ограниченных случайных величин можно использовать следующий метод. Функция плотности распределения вероятностей случайных величин $f_\eta(x)$ вписывается в прямоугольник *(a, b)* × *(0, c)*, такой, что a и b соответствуют границам диапазона изменения случайных величин η, а c – максимальному значению функции плотности её распределения. Тогда очередная реализация случайных величин определяется по следующему алгоритму:

Шаг 1. Получить два независимых случайных числа ξ_1 и ξ_2.

Шаг 2. Если $f_\eta(a + (b - a)\xi_1) > c\xi_2$ то выдать $a + (b - a)\xi_1$ в качестве результата. Иначе повторить Шаг 1.

Дискретное распределение с возвратом

Дискретной (прерывной) называют случайную величину ξ, которая принимает отдельные, изолированные возможные значения с определенными вероятностями. Число возможных значений дискретной случайной величины может быть конечным или бесконечным.

Законом распределения дискретной случайной величины называют соответствие между возможными значениями и их вероятностями.

Закон распределения дискретной случайной величины удобно задавать с помощью таблицы 1.

Таблица 1 – Ряд распределения дискретной случайной величины

x_i	x_1	x_2	$\cdots$	x_n	$\cdots$
p_i	p_n	p_n	$\cdots$	p_n	$\cdots$

При этом возможные значения x_1, x_2 … случайной величины X в верхней строке этой таблицы располагаются в определенном порядке, а в нижней — соответствующие вероятности $p_i = P\{X = x_i\}$ ($\sum_i p_i = 1$).

Функция распределения выглядит так:

$$F(x) = \sum_{i=1}^{n} p_i = 1$$

Дискретное распределение без возврата

Есть n случайных величин с одинаковой вероятностью (при следующих выборках вероятность распределяется поровну между величинами), мы выбираем $\frac{3n}{4}$ следующих величин без повторений, проделываем это большое количество раз и считаем частоты этих значений.

РЕЗУЛЬТАТЫ РАБОТЫ

Пусть случайная величина X задана плотностью вероятности:

$$f(x) = \begin{cases} 0, & x \le 1 \\ a(x^3 - 1), & 1 < x \le 3 \\ 0, & x > 3 \end{cases}$$

Известно, что несобственный интеграл от плотности вероятности есть вероятность достоверного события (условие нормировки):

$$\int_{-\infty}^{+\infty} f(x)dx = \int_{-\infty}^{1} 0dx + \int_{1}^{3} a(x^3 - 1)dx + \int_{3}^{+\infty} 0dx = a(\frac{x^4}{4} - x)|_1^3 = 1$$

$$a = \frac{1}{18}$$

Тогда функция плотности примет вид:

$$f(x) = \begin{cases} 0, & x \le 1 \\ \frac{(x^3 - 1)}{18}, & 1 < x \le 3 \\ 0, & x > 3 \end{cases}$$

Далее продемонстрирован график функции плотности распределения непрерывных случайных величин:

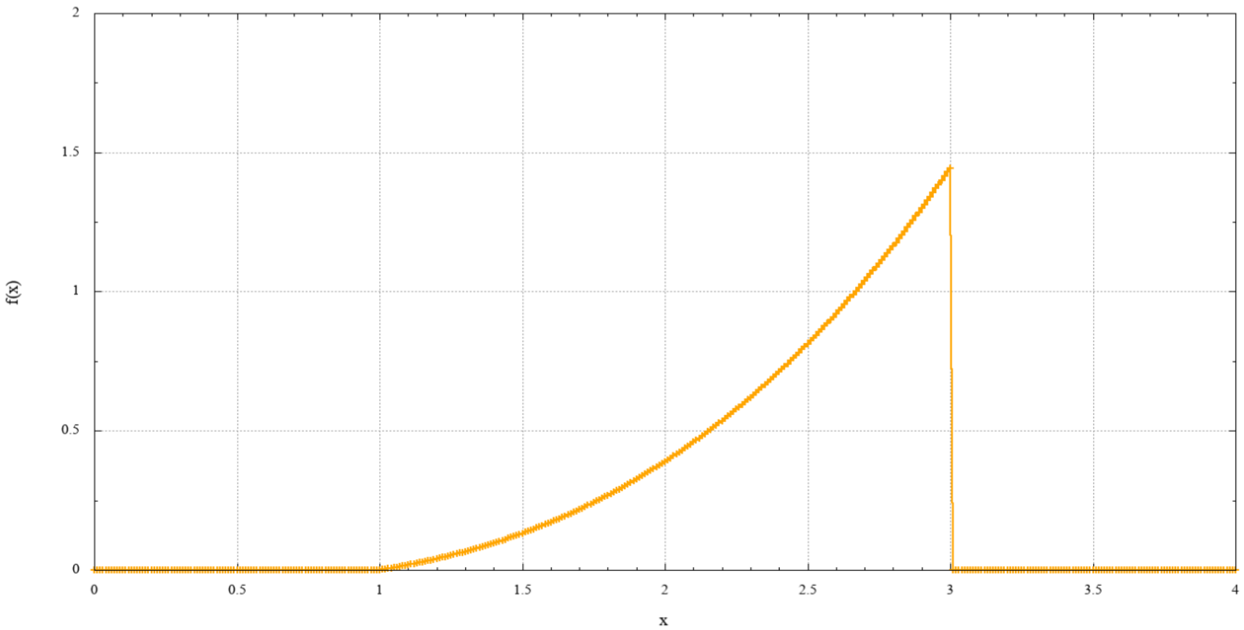

Рис.1 — График значений *f(x) случайных величин*

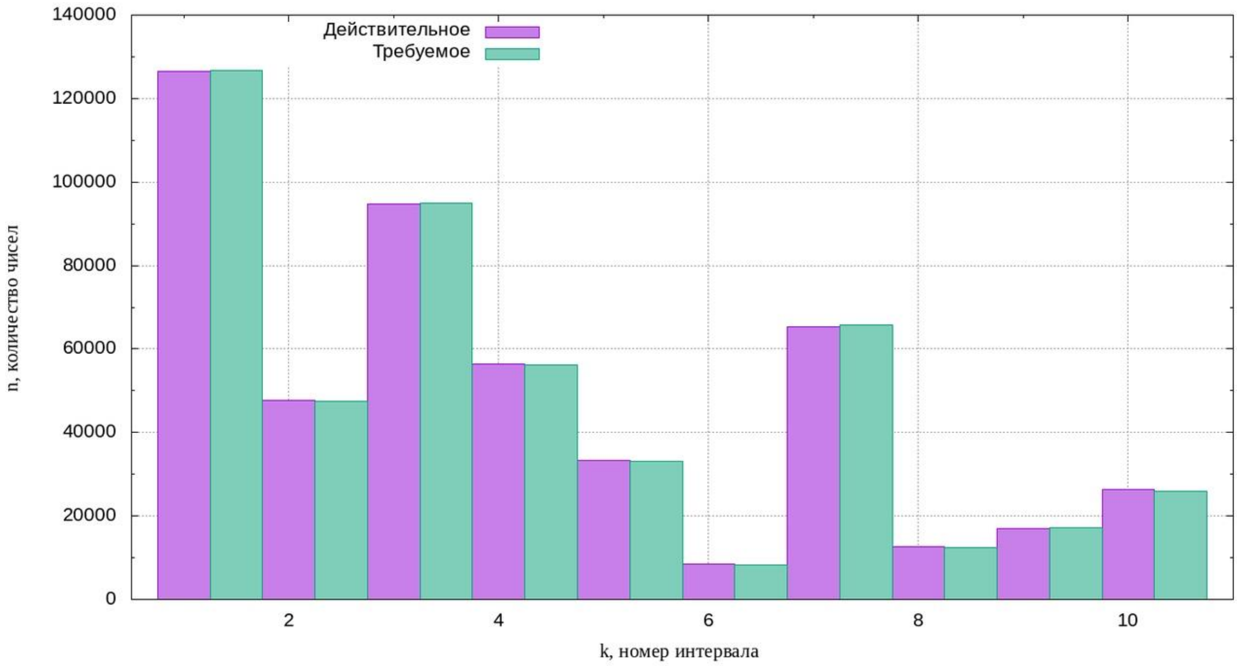

Рис.2 — Моделирование дискретной случайной величины с возвратом

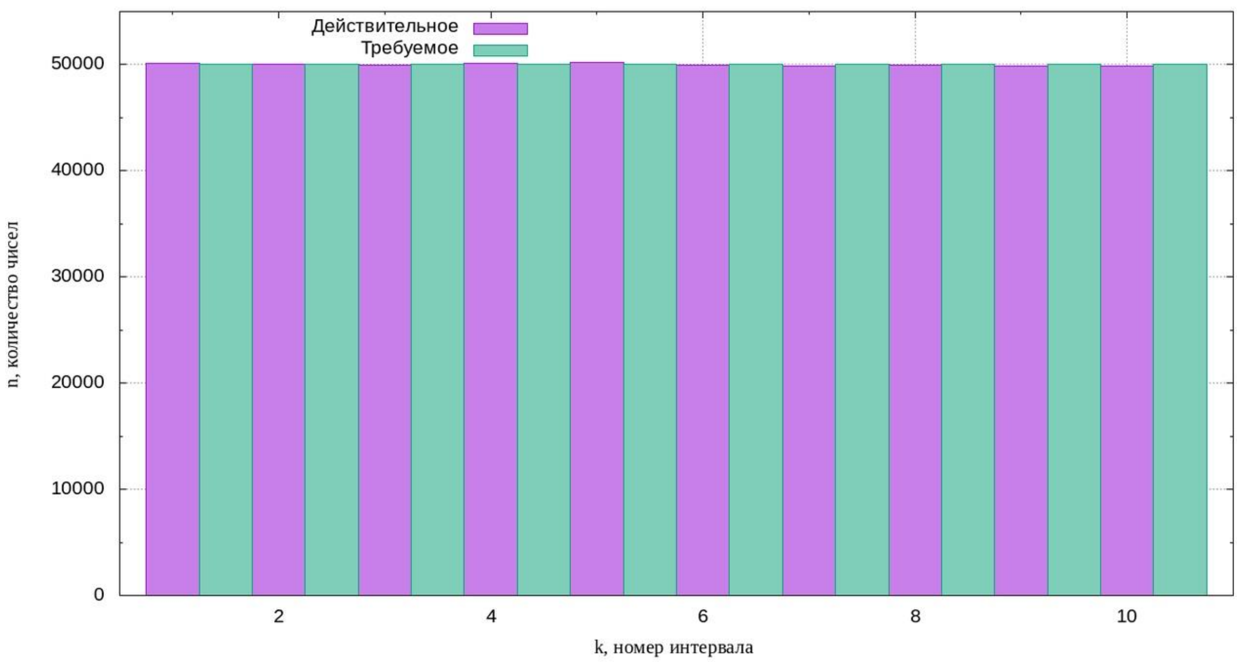

Рис.3 — Моделирование дискретной случайной величины без возврата

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе мы изучили и реализовали непрерывное распределение методом отбраковки, дискретное распределение с возвратом, дискретное распределение без возврата. В качестве генератора случайных чисел был выбран генератор SplittableRandom языка Java из библиотеки java.util.SplittableRandom.

По результатам экспериментов мы выяснили, что при генерации непрерывным распределением по методу отбраковки, значения соответствуют заданному закону распределения.

Из минусов данного метода можно отметить:

- метод не эффективен для распределений с длинными «хвостами», поскольку в этих распределениях повышается частота повторных испытаний;
- метод применим только для аналитических функций;
- метод отбрасывает точки, которые попадают выше кривой распределения, а значит время, затраченное на их вычисление, было излишним.

По результатам моделирования дискретных случайных величин с реализацией выборки с возвратом и без возврата, можно заметить, что исследуемый генератор случайных чисел выдает распределение близкое к требуемому, из чего можно сделать вывод, что генератор имеет равномерное распределение с малой долей погрешности.

ЛИСТИНГ ПРОГРАММЫ

```
import java.io.*;
import java.util.SplittableRandom;
import java.util.Vector;
import static java.lang.Math.*;


public class Main {

    public static double f(double x) {
        if (x >= 1 && x <= 3) return (x*x*x- 1) / 18;
        else
            return 0;
    }

    public static void rejection(int max_n) throws IOException {
        double a = 1, b = 3, c = f(b), xsi1, xsi2;
        String str = null;
        FileWriter file = new FileWriter("file1.txt");

        for (int i = 0; i < max_n; i++) {
            xsi1 = new SplittableRandom().nextDouble(0, max_n);
            xsi2 = new SplittableRandom().nextDouble(0, max_n);
            double def = a + ((b-a) * xsi1);
            if (def > (c * xsi2)) {
                double fabs_fun = abs(f(def));
                str += def + " " + fabs_fun;
                file.write(str);
            }
        }
        file.close();
    }

    public static void with_return(int max_n, int n) throws IOException {
        double[] probability = new double[n];
        double[] hit_to_int= new double[n];
        double chance_to_minus = 1;
        for (int i = 0; i < n; i++) probability[i] = 1 / n;
        for (int i = 0; i < n; i++) {
            double rand_num1 = new SplittableRandom().nextDouble(0, max_n);
            probability[i] = abs((rand_num1 % 1));
            chance_to_minus-= probability[i];
        }
        probability[n-1] = chance_to_minus;
        for (int i = 0; i < n; i++) hit_to_int[i] = 0;
        for (int i = 0; i < max_n*100; i++) {
            double rand_num2 = new SplittableRandom().nextDouble(0, max_n);
            double summa = 0.0;
            for (int j = 0; j < n; j++) {
                summa += probability[j];
                if (rand_num2 < summa) {
                    hit_to_int[j] += 1;
                    break;
                }
            }
        }
        String str = null;
        FileWriter file = new FileWriter("file2.txt");
        for (int i = 0; i < n; i++) {
            str += i+1 + " " + i+1.5 +
```

```
                max_n*100*probability[i] + " " + hit_to_int[i];
            file.write(str);
        }
        file.close();
    }

    public static void without_return(int max_n, int n) throws IOException {
        Vector<Integer> array_num1, array_num2 = null;
        int k = 3 * n / 4, max_n_to_def = (max_n * 100 / k) + 1;
        double[] hit_to_int = new double[n];
        for (int i = 0; i < n; i++) hit_to_int[i] = 0;
        for (int i = 0; i < n; i++) array_num2.addElement(i);
        for (int i = 0; i < max_n_to_def; i++) {
            array_num1 = array_num2;
            if (i == max_n_to_def - 1) k = (max_n * 100) % k;
            for (int j = 0; j < k; j++) {
                float p = (float) (1.0 / (n-j));
                float num_rand = (float) new SplittableRandom().nextDouble(0, max_n*100);
                int num_rand_to = (int) (num_rand / p);
                hit_to_int[array_num1[num_rand_to]] += 1;
                array_num1.remove(array_num1.begin() + num_rand_to);
            }
        }
        String str = null;
        FileWriter file = new FileWriter("file3.txt");
        for (int i = 0; i < n; i++) {
            str += i+1 + " " + i+1.5 + " " + hit_to_int[i] + " " + max_n*100/20;
            file.write(str);
        }
        file.close();
    }

    public static void main(String[] args) throws IOException {
        int max_n = 5000, n = 10;
        rejection(max_n);
        with_return(max_n, n);
        without_return(max_n, n);
        return 0;
    }
}
```