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

Тимофеев Д.А.

Проверила:

Ассистент Кафедры ВС

Петухова Я.В.

Новосибирск 2020

ИЗУЧЕНИЕ И РЕАЛИЗАЦИЯ НЕПРЕРЫВНОГО РАСПРЕДЕЛЕНИЯ РАЗЛИЧНЫМИ МЕТОДАМИ

ИВ-622 Тимофеев Д.А.

СОДЕРЖАНИЕ:

1. Постановка задачи

Реализовать:

1) Непрерывное распределение методом отбраковки
2) Дискретное распределение с возвратом
3) Дискретное распределение без воврата

2. Теория

2.1. 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 $F_\eta(x)$.

2.2. 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*, *a c* – максимальному значению функции плотности еѐ распределения. Тогда очередная реализация случайных величин определяется по следующему алгоритму:

Шаги выполнения

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

2.3. Дискретное распределение с возвратом

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.

3. Выполнение работы

Пусть случайная величина X задана плотностью вероятности

$$f(x) = \begin{cases} -x, & x < 0 \\ ax^4/4, & 0 \le x \le 2 \\ x, & x > 2 \end{cases}$$

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

$$\int_{-\infty}^{+\infty} f(x)dx = \int_{-\infty}^{0} 0dx + \int_{0}^{2} f(x)dx + \int_{2}^{+\infty} 0dx = \left(\frac{ax^5}{16}\right)\Big|_0^2 = \frac{a2^5}{16} - 0 = 1$$

$$2a = 1$$

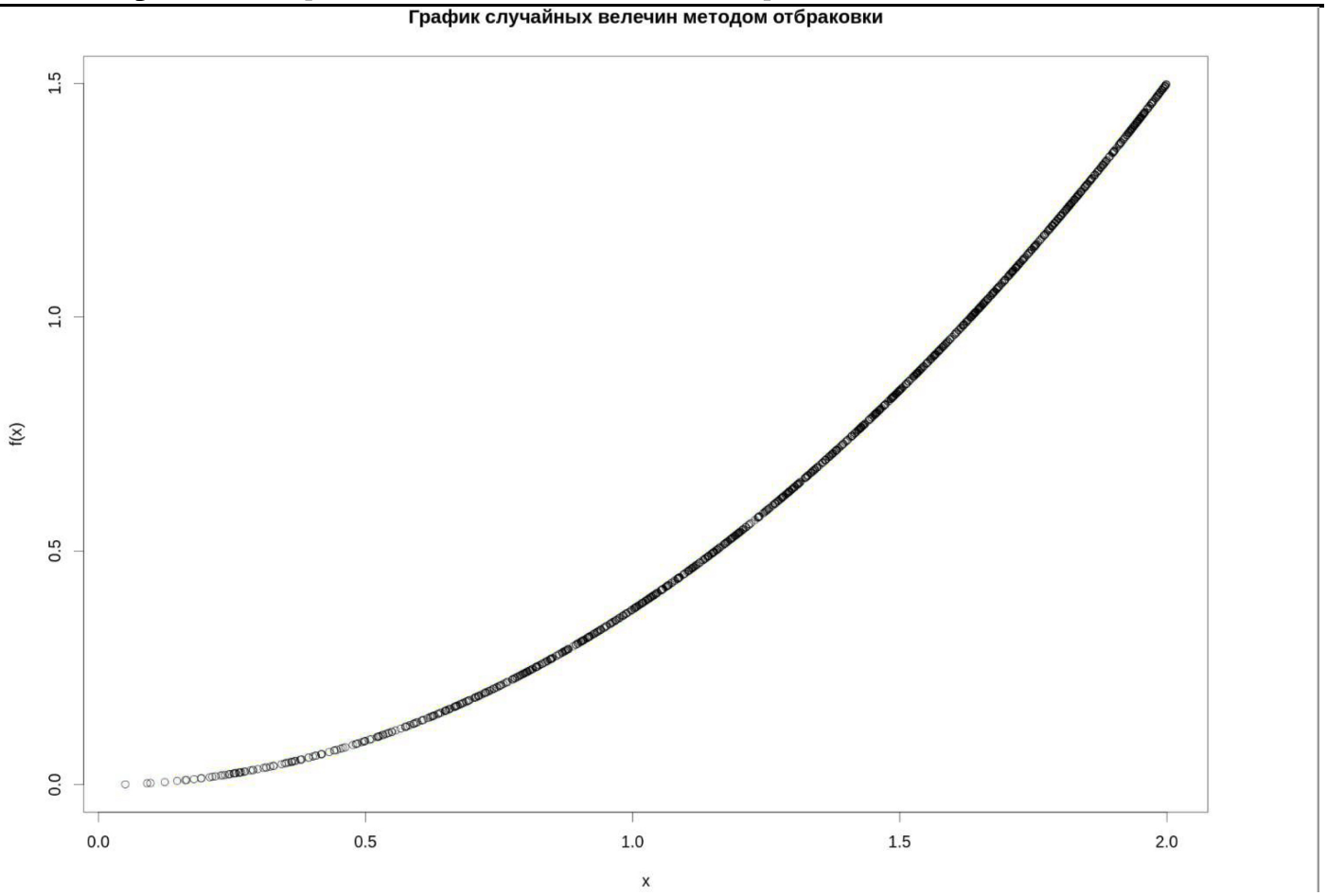
$$a = 0.5$$

Тогда функция плотности распределения имеет вид:

$$f(x) = \begin{cases} -x, & x < 0 \\ x^4/8, & 0 \le x \le 2 \\ x, & x > 2 \end{cases}$$

4. Результат работы

4.1. Результат работы метода отбраковки



4.2. Моделирование дискретного распределения случайной величины с возвратом

4.3. Моделирование дискретного распределения случайной величины без возврата

5. Вывод

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 работы метода отбраковки для заданной функции плотности можно сделать вывод, что с его помощью удалось смоделировать эту функцию, графики для обоих случаев получились идентичны (при построении по заданной функции и при построении с помощью метода отбраковки). К недостаткам данного метода можно отнести то, что точки, не попавшие в интервал, отбрасываются, несмотря на то что было потрачено время на их генерацию. Также данный метод не эффективен для распределений с длинными «хвостами», поскольку в этом случае имеются частые повторные испытания.

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.

6. Листинг

Main.java

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;
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 {
        List<Integer> array_num1 = new ArrayList<Integer>();
        List<Integer> array_num2 = new ArrayList<Integer>();
        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.add(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.get(num_rand_to)] += 1;
                array_num1.remove(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);
    }
}
```