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

Кафедра ВС

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к первой лабораторной работе
«Генерация случайных чисел с заданным распределением»
по дисциплине «Моделирование»

Выполнил студент Гурулев Дмитрий Александрович
Ф.И.О.

Группы ИВ-621

Проверил: _____ ассистент кафедры ВС
подпись Я.В. Петухова

Новосибирск, 2020

ОГЛАВЛЕНИЕ

ПОСТАНОВКА ЗАДАЧИ

В рамках лабораторной работы необходимо смоделировать генерацию независимых случайных величин:

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

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Метод отбраковки

Данный метод является методом моделирования независимых случайных величин с заданным законом распределения и используется, когда функция задана аналитически.

На ось Y и X подают случайное равномерно распределенное число из ГПСЧ. Если точка в пересечении этих двух координат лежит ниже кривой плотности вероятности, то событие X произошло, иначе нет.

Недостаток метода: точки, которые оказались выше кривой распределения плотности вероятности, отбрасываются как ненужные, и время, затраченное на их вычисление, оказывается напрасным. Метод применим только для аналитических функций плотности вероятности.

Алгоритм:

1. В цикле генерируется два случайных числа из диапазона от 0 до 1.
2. Числа масштабируются в шкалу X и Y, и проверяется попадание точки со сгенерированными координатами под график заданной функции $Y = f(X)$.
3. Если точка находится под графиком функции, то событие X произошло с вероятностью Y, иначе точка отбрасывается.

Моделирование выборок дискретных случайных величин

Случайная величина ξ называется дискретной, если она может принимать дискретное множество значений $(x_1\ x_2\ x_3\ \dots\ x_n)$. Величина ξ

определяется таблицей (распределением случайной величины ξ) или аналитически в виде формулы:

$$\xi = \begin{pmatrix} x_1\ x_2\ x_3 \dots\ x_n \\ p_1\ p_2\ p_3 \dots\ p_n \end{pmatrix}$$

,где $(x_1\ x_2\ x_3 \dots\ x_n)$ – возможные значения величины ξ; $(p_1\ p_2\ p_3 \dots\ p_n)$ – соответствующие значениям величины ξ вероятности:

$$P(\xi = x_i) = p_i$$

Числа $(x_1\ x_2\ x_3 \dots\ x_n)$ могут быть любыми, а вероятности $(p_1\ p_2\ p_3 \dots\ p_n)$ удовлетворяют условиям: $p_i > 0$; $\sum_{i=1}^{n} p_i = 1$.

Дискретная функция распределения имеет вид:

$$F(x) = \sum_{x_k < x} P(x = x_k)$$

Наиболее общий случай построения генератора дискретных случайных величин основывается на следующем алгоритме. Пусть имеется таблица пар $(x_i p_i)$:

X	x_1	x_2	…	x_i	…
$P(X = x_i)$	p_1	p_2	…	p_i	…

Тогда кумулятивную сумму можно представить полуинтервалом [0,1), разбитым на полуинтервалы $[v_i - 1, v_i), v_0 = 0, v_n = 1$ длины p_i.

Случайная величина ξ принадлежит одному из этих полуинтервалов, определяя индекс дискретного значения. Поэтому номер полуинтервала, определяется как:

$$\min\{i | \xi < v_i\} = \min\{i | \xi < \sum_{j=1}^{i} p_i\}$$

Дискретное распределение без возврата

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

РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

Метод отбраковки

Для того, чтобы построить функцию распределения необходимо вычислить значение a функции плотности распределения. Сама наша плотность распределения $f(x)$ выглядит следующим образом:

$$f(x) = \begin{cases} 0 & x \leq 3 \\ a * \sqrt{x-2} & 3 < x \leq 6 \\ 0 & x > 6 \end{cases}$$

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

$$\int_{-\infty}^{\infty} f(x)dx = P\{-\infty < X < \infty\} = P\{\Omega\} = 1$$

Исходя из этого свойства, можем найти параметр a:

$$\int_{-\infty}^{\infty} f(x)dx = \int_{3}^{6} a \cdot \sqrt{x-2}dx = \frac{2}{3} \cdot a \cdot \sqrt{(x-2)^3}\Big|_3^6$$

$$= \frac{2}{3} \cdot a \cdot \sqrt{4^3} - (\frac{2}{3} \cdot a \cdot \sqrt{1^3}) = 1$$

$$a = \frac{3}{14}$$

Далее продемонстрирован график функции плотности распределения непрерывных случайных величин:

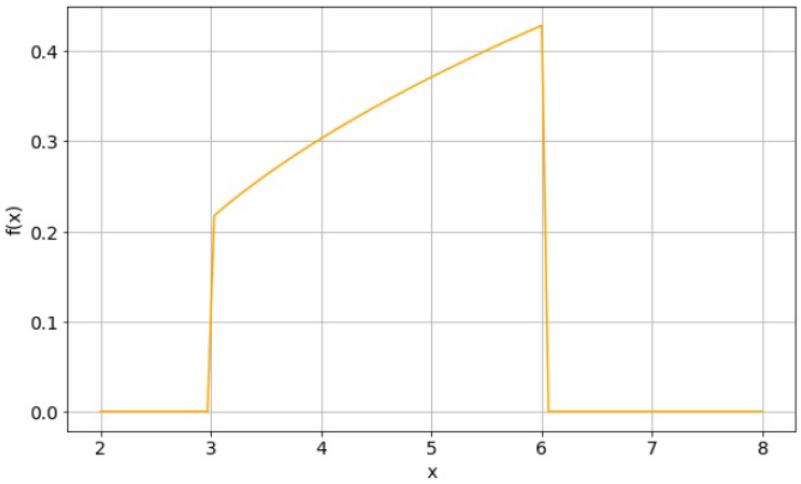

Рисунок 1. График функции плотности

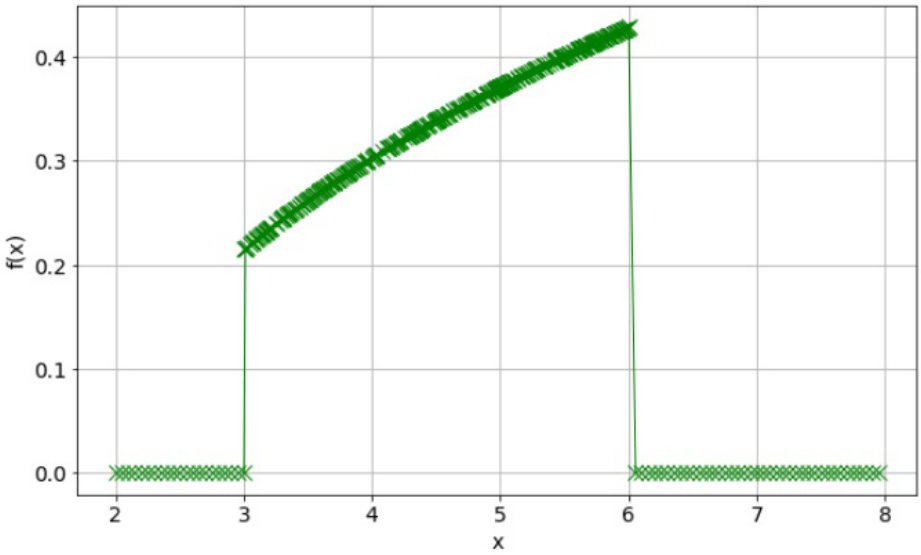

Рисунок 2. График с результатами работы метода отбраковки

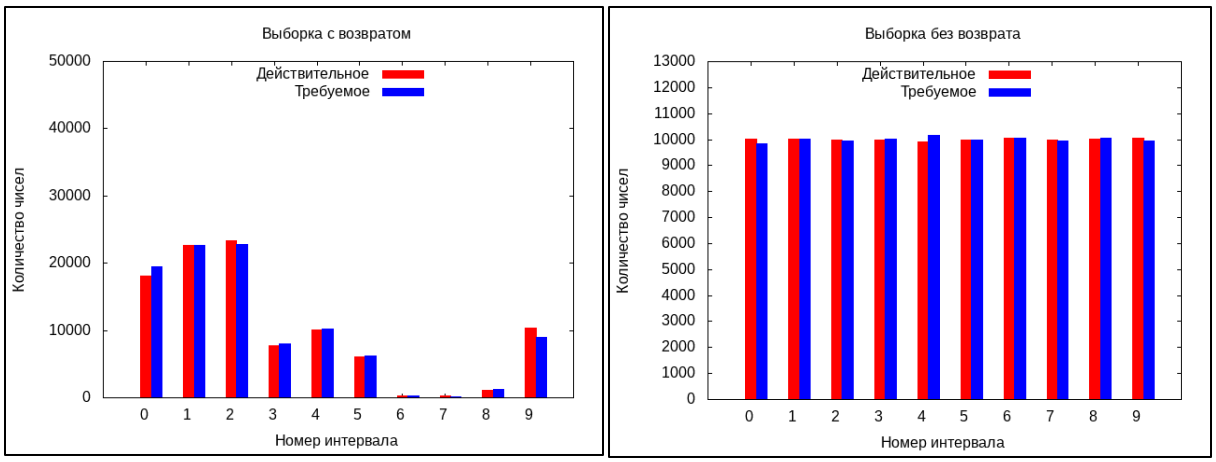

Рисунок 3. Графики с результатами работы моделирования дискретной с.в.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были изучены методы моделирования случайных величин, реализованы алгоритмы с непрерывным распределением методом отбраковки и с дискретным распределением с возвратом и без возврата. Для реализации алгоритмов был использован язык C#. Для генерации случайных чисел была использована измененная версия алгоритма генератора псевдослучайных чисел с вычитанием Д.Кнута, для визуализации - библиотека ScottPlot.

По результатам работы метода отбраковки можно утверждать, что при генерации непрерывным распределением, значения соответствуют заданному закону распределения. Однако, результаты указывают на неэффективность метода отбраковки, которая заключается в том, что избыточные вычисления тех точек, которые попали выше требуемой области, так как они отбрасываются как ненужные. Исходя из вышесказанного, можно сделать вывод, что данный метод применим только для аналитических функций плотности вероятности.

По результатам моделирования дискретных случайных величин с реализацией выборки “с возвратом” и “без возврата”, можно сделать вывод, что ГСЧ выдает распределение близкое к равномерному с малой долей погрешности, т.к. действительные значения в сравнении с ожидаемыми имеют достаточно малое отклонение.

ПРИЛОЖЕНИЕ

Листинг программы

```
using System;
using System.Collections.Generic;
using ScottPlot;

namespace ModelingLab2
{
    class Program
    {
        public static double A = 3.0 / 14.0;
        public static int REJECTS = 10000;
        public static double Fn(double x)
        {
            double returnValue;
            if (x.CompareTo(3.0) > 0 && x.CompareTo(6.0) < 0) returnValue = A * x * x;
            else returnValue = 0.0;
            return returnValue;
        }
        public static double rejectMethod()
        {
            Random random = new Random();
            double a = 3.0, b = 6.0, c = 1.0;
            double x, y, leftHandSide, rightHandSide;
            do
            {
                x = random.NextDouble();
                y = random.NextDouble();
                leftHandSide = Fn(a + (b - a) * x);
                rightHandSide = c * y;
            } while (leftHandSide.CompareTo(rightHandSide) <= 0);
            return a + (b - a) * x;
        }
        public static void continous()
        {
            List<long> vector = new List<long>();
            for (int i = 0; i < REJECTS * 1000; i++) vector.Add(0);
            for (int i = 0; i < REJECTS * 1000; i++)
            {
                double pX = rejectMethod();
                //Console.WriteLine("Rand pX: " + pX);
                int index = (int)(Math.Floor(pX * 1000));
                //Console.WriteLine("INDEX: " + index);
                if (index < 5000)
                {
                    //Console.WriteLine("Found!");
                    long val = vector[index];
                    val++;
                    vector[index] = val;
                }
            }
            double[] xData = new double[REJECTS];
            double[] yData = new double[REJECTS];
            for (int i = 0; i < 5000; i += 25)
            {
                xData[i] = i / 1000.0;
                yData[i] = vector[i];
                //Console.WriteLine("xVal: " + xData[i] + "yVal: " + yData[i]);
            }

            var plt = new ScottPlot.Plot(600, 400);
            plt.PlotScatter(xData, yData, label: "Data(y/x)", lineWidth: 0);
            plt.Legend();
            plt.Title("Reject Method");
            plt.XLabel("x");
```

```
            plt.YLabel("f(x)");
            plt.SaveFig("reject.png");
        }
        static void Main(string[] args)
        {
            continous();
            Console.WriteLine("Great!");
            Console.ReadLine();
        }
    }
}
```