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

Кафедра Вычислительных систем (ВС)

Лабораторная работа №1

по дисциплине «Моделирование»

Выполнил:
студент гр. ИВ-622
Гайдук П.А

Работу проверила:
Ассистент кафедры ВС
Петухова Я.В

Новосибирск 2020

Оглавление

Постановка задачи

Генерация независимых, одинаково распределенных случайных величин.

1. Непрерывное распределение с помощью метода отбраковки
2. Дискретное распределение с возвратом
3. Дискретное распределение без возврата

Теоретические сведения

1. Непрерывное распределение с помощью метода отбраковки.

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

a. Построить функцию плотности распределения вероятностей
b. Получить два независимых случайных числа ξ_1 и ξ_2
c. Если $f\eta(a + (b - a)\,\xi_1) > c\,\xi_2$ то выдать $a + (b - a)\xi_1$ в качестве результата. Иначе повторить Шаг b.

Неэффективность алгоритма для распределений с длинными “хвостами” очевидна, поскольку в этом случае часты повторные испытания.

2. Дискретное распределение с возвратом

Случайная величина X называется дискретной, если она может принимать дискретное множество значений $(x_1, x_2, x_3, ..., x_n)$. Дискретная случайная величина описывается с помощью таблицы (распределение случайной величины X) или в виде аналитической формулы.

Функция распределения случайной величины, с дискретным распределением, имеет вид:

$$\begin{cases} 0, & \text{при } x < x_1, \\ p_1, & \text{при } x_1 \le x < x_2, \\ p_1 + p_2, & \text{при } x_2 \le x < x_3, \\ \ldots, & \\ p_1 + p_2 + \ldots + p_{n-1}, & \text{при } x_{n-1} \le x < x_n, \\ 1, & \text{при } x \ge x_n. \end{cases}$$

3. Дискретное распределение без возврата

Есть n случайных величин с одинаковой вероятностью (при следующих выборках вероятность распределяется поровну между величинами), мы выбираем 3 ∗ n 4 следующих величин без повторений, проделываем это большое количество раз и считаем частоты этих значений. Универсальный алгоритм для выборки без повторения. Входные данные: n – число индексов; m – длина выборки. Выходные данные: L[1..m] – результирующий вектор индексов.

Выполнение и оценка результатов экспериментов

Выбрана плотность распределения:

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

Найдем параметр A из условия нормировки $\int_{-\infty}^{\infty} f(x)dx = 1$. Получаем:

$$\int_{-\infty}^{\infty} f(x)dx = \int_{-\infty}^{0} 0dx + \int_{0}^{2} \left(\frac{ax^2}{2}\right) dx + \int_{2}^{\infty} 0dx = \frac{ax^3}{6}\Big|_0^2 = a\frac{4}{3} - 0 = 1$$

, отсюда a = $\frac{3}{4}$

Тогда функция плотности примет вид:

$$f(x) = \begin{cases} 0, \text{при } x \le 0; \\ \frac{3x^2}{8}, \text{при } 0 < x \le 2; \\ 0, \text{при } x > 2. \end{cases}$$

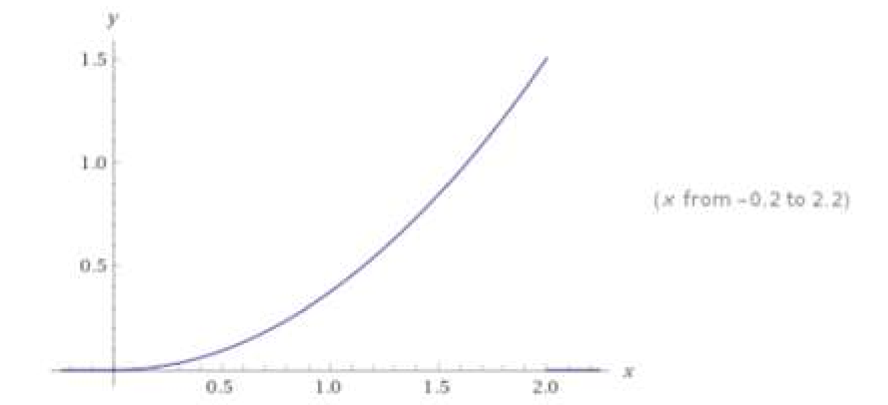

Рисунок 1 - График плотности распределения

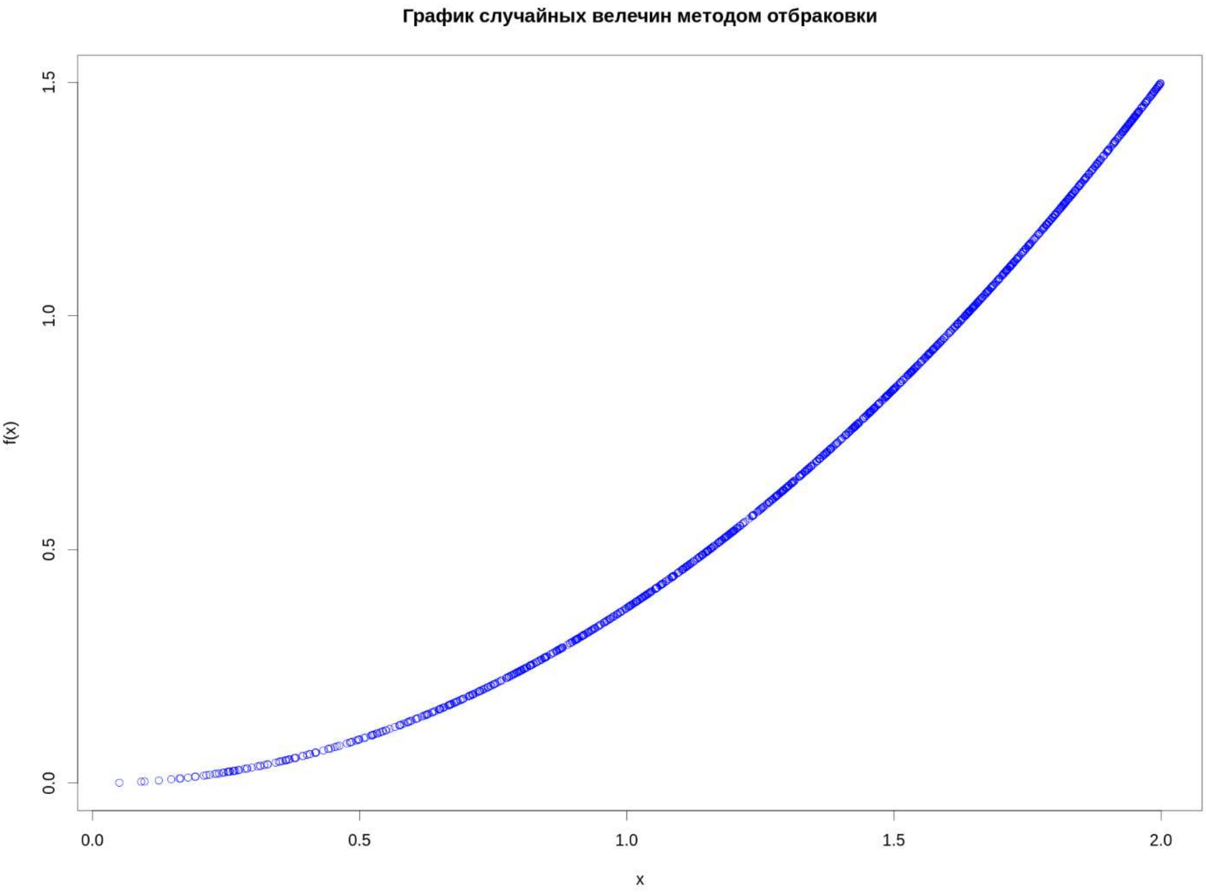

Рисунок 2 – Результат работы метода отбраковки

Рисунок 3 – Моделирование дискретного распределения случайной величины с возвратом

Рисунок 4 – Моделирование дискретной случайной величины без возврата

Заключение

В данной лабораторной работе изучили и реализовали непрерывное распределение методом отбраковки, дискретное распределение с возвратом и дискретное распределение без возврата. В качестве генератора случайных чисел был выбран стандартный генератор *Scala* из библиотеки *scala.util.Random*

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 реализации моделирования дискретных случайных величин с возвратом и без него можно сделать вывод, что генератор случайных чисел стандартной библиотеки *Scala* выдает распределение, близкое к равномерному.

Приложение

Lab1.scala

```
package com.pgaiduk.distribs

import scala.math.pow
import java.io._
import scala.util.control.Breaks._

object Program {

  def reject(a:Float, b:Float, ksi:Float): Float = a + (b - a) * ksi

  def dens(x:Float): Float = (pow(x,2) * 3 / 8).toFloat

  def main(args: Array[String]){

    /* Метод отбраковки*/
    var N:Int = 0
    var a,b,ksi1,ksi2,c:Float = 0
    a = 0
    b = 2
    c = (pow(b,2) * 3 / 8).toFloat
    N=100000
    val r = scala.util.Random
    var k:Int = 0
    val pw = new PrintWriter(new File("task1.txt" ))
    for (i <-0 until N){
      ksi1 = r.nextFloat()
      ksi2 = r.nextFloat()
      var tmp:Float = reject(a,b,ksi1)
      if ((tmp > (c * ksi2))){
        k += 1
        //println(tmp + "," + dens(tmp).abs)
        pw.write(tmp + "," + dens(tmp).abs + "\n")
      }
    }
    pw.close
    println("------------")

    /* Дискретная случайная величина с возвратом*/
    var g_num:Int = 30
    var rem_chance:Float = 1
    var hit_c = new Array[Float](g_num)
    var chance = 0
    for(i <- 0 until g_num){
      var f:Float = 0
      hit_c(i) = (r.nextFloat() % rem_chance).abs
      rem_chance = rem_chance - hit_c(i)
    }
    hit_c(g_num - 1) = rem_chance
    var hit_a = new Array[Int](g_num)
     for (i <- 0 until N){
      var sum:Float = 0
       breakable {for (j <- 0 until g_num){
        sum += hit_c(j)
        if (chance < sum){
```

```
          hit_a(j) = hit_a(j) + 1
          break
        }
      }}
    }
    for (i <- 0 until g_num){
      println(i + "      " + hit_c(i) * N + "     " + hit_a(i))
    }

    println("-----------------")

    /* Дискретная случайная величина с без возврата*/
    var nums = new Vector[Int]()
    var temp = new Vector[Int]()

    var g_nums:Int = 30
    var it:Int = 0
    var k1:Int = 3 * g_nums / 4
    var hit = new Array[Int](g_nums)
    var part:Int = N / k1 + 1
    var p:Float = 0

    for (number <- 0 until g_nums){
      temp :+ number
    }

    for (j <- 0 until part ){
      nums = temp
      if (j == part - 1){
        k = 100000 % k
      }
      for (i <- 0 until k){
        p = (1.0 / (g_nums - i)).toFloat
        it = (r.nextFloat() / p).toInt
        hit(nums(it)) = hit(nums(it)) + 1
      }
    }
    for (i <- 0 until g_nums){
      println(i + ": " + hit(i))
    }
  }
}
```