ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГОБУ ВПО “СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ”

Кафедра вычислительных систем

Лабораторная работа №1
по дисциплине «Моделирование»

Выполнил:
студент гр. ИВ-622
Гайнулин Р.М.

Проверил:
Ассистент кафедры ВС
Петухова Я.В.

Новосибирск, 2020

Оглавление

Постановка задачи

Реализовать:

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

Теоретические сведения

1. Непрерывное распределение методом отбраковки

Распределение вероятностей — это закон, описывающий область значений случайной величины и соответствующие вероятности появления этих значений. Случайной величиной называется переменная, которая может принимать те или иные значения в зависимости от различных обстоятельств, и случайная величина называется непрерывной, если она может принимать любое значение из какого-либо ограниченного или неограниченного интервала.

Для непрерывной случайной величины невозможно указать все возможные значения, поэтому обозначают интервалы этих значений, которые связаны с определёнными вероятностями. Для ограниченных случайных величин используется метод отбраковки. Данный метод используется в таком случае, когда функция задана аналитически. Этот метод используется для моделирования непрерывной случайной величины. Функция плотности распределения вероятностей случайных величин $f_\eta(x)$ вписывается в прямоугольник $(a, b) \times (0, c)$, где a и b соответствует границам диапазона случайной величины η, c соответствует максимальному значению функции плотности её распределения.

Реализация выглядит следующим образом:

- Получить 2 независимых случайных числа ξ_1и ξ_2.
- Если $f_\eta(a + (b - a)\xi_1) > \xi_2 c$ то выдать в качестве результата $a + (b - a)\xi_1$, иначе повторить предыдущий шаг.

2. Дискретное распределение с возвратом

Дискретная случайная величина — случайная величина ξ, которая может принимать дискретное множество значений $(x_1 x_2 \ldots x_n)$. Дискретная случайная величина ξ определяется таблицей (распределением случайной величины ξ) или аналитически в виде формулы:

$$\xi = (x_1 x_2 \ldots x_n p_1 p_2 \ldots p_3), \text{ где}$$

- $(x_1 x_2 \ldots x_n)$ – возможные значения величины ξ (возможно любое значение);
- $(p_1 p_2 \ldots p_n)$ – соответствующие им вероятности (должны удовлетворять следующим условиям: $\sum_{i=1}^{n} p_i = 1$ и $p_i > 0$).

Функция распределения случайной величины данного распределения имеет следующий вид:

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

Алгоритм построения генератора дискретных случайных величин: 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 и определяет индекс дискретного значения. Номер полуинтервала можно вычислить следующим образом:

$$min\{i \mid \xi < v_i\} = min\{i \mid \xi < \sum_{j=1}^{i} p_j\}.$$

3. Дискретное распределение без возврата

В дискретном распределении без возврата n случайных величин с одинаковой вероятностью. При следующих выборках вероятность распределяется поровну между величинами. Выберем $\frac{3*n}{4}$ следующих величин без повторений, проделываем это большое количество раз и считаем частоты этих значений.

Результаты экспериментов

Допустим, функция плотности распределения:

$$f(x) = \begin{cases} 0, & x \le 1 \\ a(x^2 - 1), & 1 < x \le 3 \\ 0, & x > 3 \end{cases}$$

По условию нормировки несобственный интеграл от функции плотности вероятности равен вероятности достоверного события:

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

Таким образом, можем найти параметр а:

$$\int_{-\infty}^{+\infty} f(x)dx = \int_{-\infty}^{1} 0dx + \int_{1}^{3} a(x^2 - 1)dx + \int_{3}^{+\infty} 0dx = a(\frac{x^3}{3} - x) \mid \frac{3}{1} = 1$$

$$a = \frac{3}{20}$$

Подставляя найденный параметр в функцию плотности распределения, получим следующую плотность распределения:

$$f(x) = \begin{cases} 0, & x \le 1 \\ \frac{3(x^2 - 1)}{20}, & 1 < x \le 3 \\ 0, & x > 3 \end{cases}$$

Получим такой график плотности распределения:

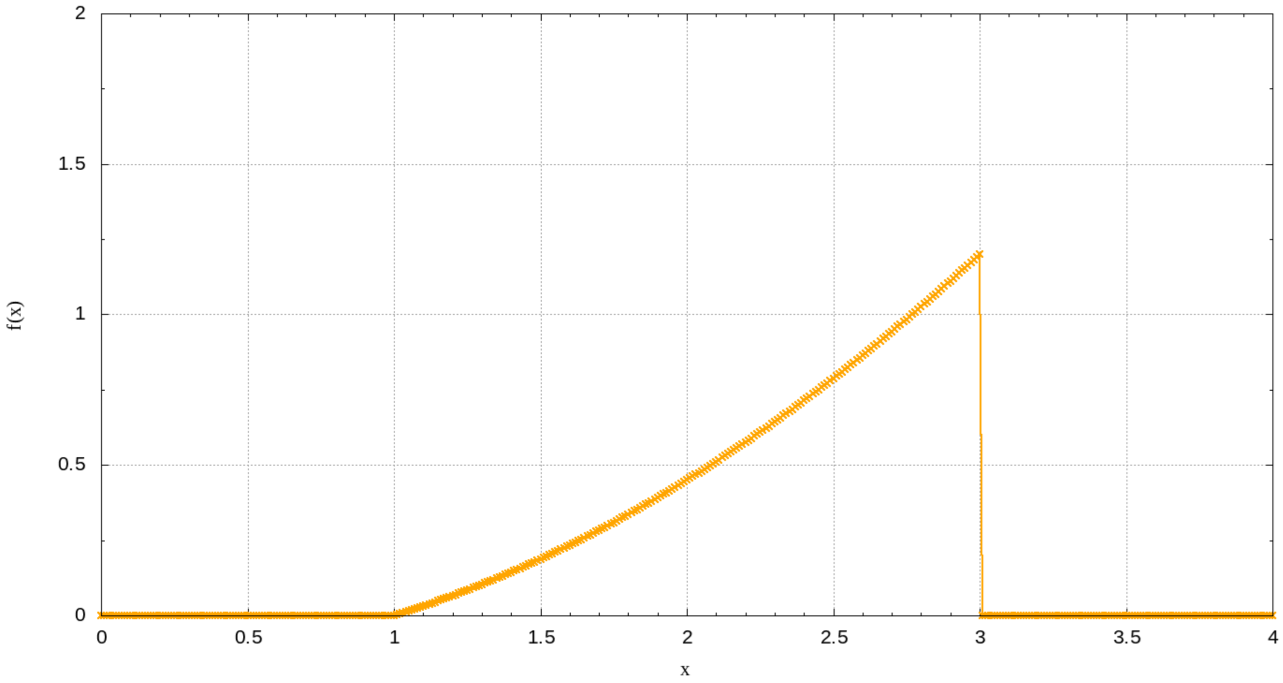

Рис. 1 — График функции плотности распределения

Результаты работы метода отбраковки:

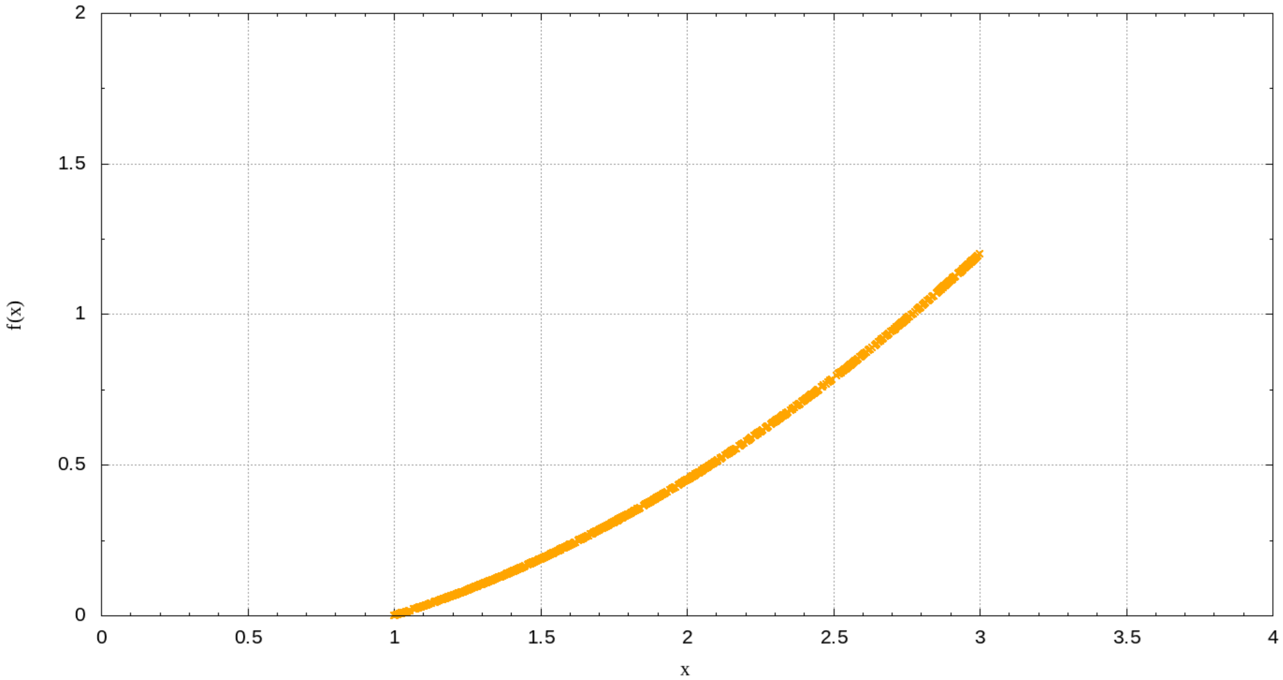

Рис. 2 — График непрерывного распределения метода отбраковки

Покажем график моделирование дискретной случайной величины с возвратом:

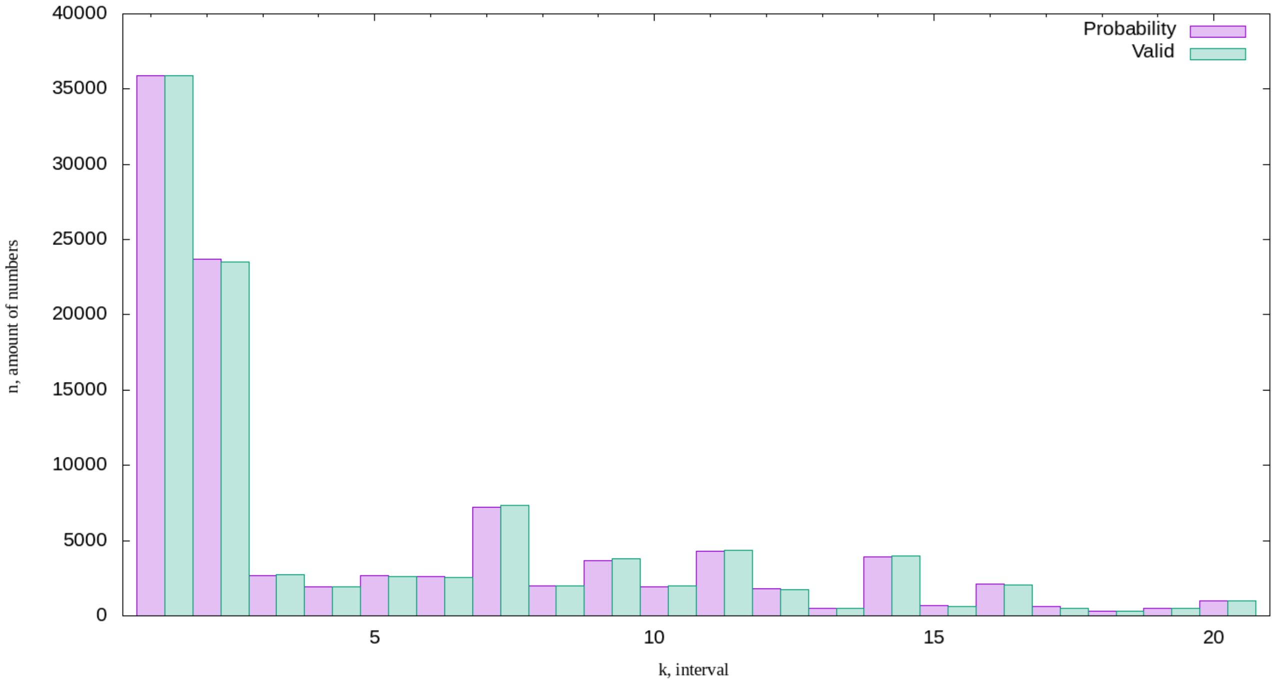

Рис. 3 — Моделирование дискретной случайной величины с возвратом

Покажем график моделирование дискретной случайной величины без возврата:

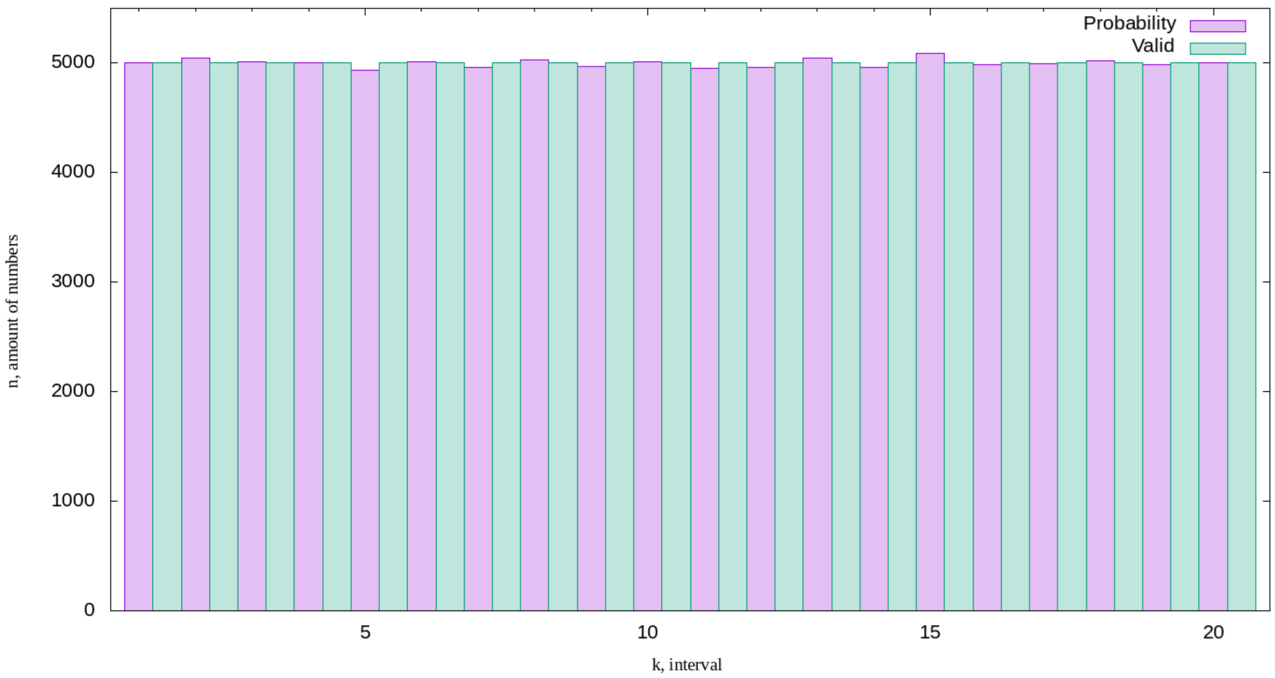

Рис. 4 — Моделирование дискретной случайной величины без возврата

Заключение

В данной лабораторной работе мы изучили и реализовали непрерывное распределение методом отбраковки, дискретное распределение с возвратом, дискретное распределение без возврата. В качестве реализации генератора случайных чисел был использован Mersenne twister (вихрь Мерсенна) на языке программирования C++.

Сделаем выводы по результатам работы метода отбраковки для заданной функции плотности:

- метод оказался применим для аналитических функций;
- данный метод достаточно точно смоделировал заданную функцию плотности.

Так же данный метод имеет некоторые недостатки:

- метод не имеет эффективности для распределений с длинными хвостами, так как очень часто происходят повторные испытания;
- точки, попадающие выше кривой распределения, отбрасываются как ненужные, следовательно, время на их вычисление оказывается излишне затратным.

Таким образом, исследуя дискретные случайные величины с возвратом и дискретные случайные величины без возврата можно сказать, что данное действительное распределение достаточно приближено к требуемому, это значит, что заданный генератор случайных чисел Mersenne twister выдает близкое к равномерному распределение.

Приложение. Листинг

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <random>
#include <limits.h>
#include <malloc.h>
#include <iostream>
#include <vector>
#include <climits>
#include <fstream>

int rand_mt19937_64(int begin, int end) {
    std::random_device rd;
    std::mt19937_64 gen(rd());
    std::uniform_int_distribution<int> uid(begin, end);
    return uid(gen);
}

double rander(int max_n) {
    int xsi = rand_mt19937_64(0, max_n);
    return (double)xsi / (double)max_n;
}

double f(double x) {
    if (x >= 1 && x <= 3) return 3 * (x*x - 1) / 20;
    else return 0;
}

void rejection(int max_n) {
  double a = 1, b = 3, c = f(b), xsi1, xsi2;
  FILE *file1 = fopen("file1.txt", "w");
  for (double i = a-1; i < b+1; i += 0.01) {
      fprintf(file1, "%lf %lf\n", i, f(i));
  }
  fclose(file1);
  FILE *file2 = fopen("file2.txt", "w");
  for (int i = 0; i <  max_n; i++) {
      xsi1 = rander(max_n);
      xsi2 = rander(max_n);
      double def = a + ((b-a) * xsi1);
      if (def > (c * xsi2)) {
          double fabs_fun = fabs(f(def));
          fprintf(file2, "%lf %lf\n", def, fabs_fun);
      }
  }
  fclose(file2);
}

void drn_with_return(int max_n, int n) {
    double probability[n], hit_to_int[n], 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 = rander(max_n);
        probability[i] = abs(remainder(rand_num1, chance_to_minus));
        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 = rander(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;
            }
        }
    }
    FILE *a = fopen("11.txt", "w");
    for (int i = 0; i < n; i++) {
```

```
        printf("%d\t%.4lf\t%.4lf\n",i+1, max_n*100*probability[i], hit_to_int[i]);
        fprintf(a, "%d\t%.2lf\t%.4lf\t%.4lf\n", i+1, i+1.5, max_n*100*probability[i], hit_to_int[i]);
    }
    fclose(a);
}

void drn_without_return(int max_n, int n) {
    std::vector<int> array_num1, array_num2;
    int k = 3 * n / 4, max_n_to_def = (max_n * 100 / k) + 1;
    double hit_to_int[n];
    for (int i = 0; i < n; i++) hit_to_int[i] = 0;
    for (int i = 0; i < n; i++) array_num2.push_back(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 = 1.0 / (n-j);
          float num_rand = rander(max_n*100);
          int num_rand_to = num_rand / p;
          hit_to_int[array_num1[num_rand_to]] += 1;
          array_num1.erase(array_num1.begin() + num_rand_to);
        }
    }

    FILE *a = fopen("22.txt", "w");
    for (int i = 0; i < n; i++) {
        printf("%d\t%lf\n",i+1, hit_to_int[i]);
        fprintf(a, "%d\t%lf\t%lf\t%d\n", i+1, i+1.5, hit_to_int[i], max_n*100/20);
    }
    fclose(a);
}

int main() {
    srand(time(NULL));
    int max_n = 5000, n = 30;
    rejection(max_n);
    drn_with_return(max_n, n);
    drn_without_return(max_n, n);
    return 0;
}
```