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

Кафедра ВС

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К первой лабораторной работе по дисциплине
«Моделирование»

Выполнил:
студент гр. ИВ-621
Дьяченко Д. В.
Проверила:
Ассистент кафедры ВС
Петухова Я.В.

Новосибирск, 2020

Оглавление

Задание на проектирование

Необходимо построить нижеследующие распределения случайных величин:

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

Теория по проектной части

Для проектирования был взят генератор случайных чисел PCG64 - 128-битная имплементация перестановочного конгруэнтного генератор.

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 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, а c – максимальному значению функции плотности ее распределения. Тогда очередная генерация случайных величин сводится к следующему алгоритму:

1. Построить функцию плотности распределения вероятностей
2. Получить два независимых случайных числа

3. Если $f_\eta(a + (b - a) * \xi_1) > c * \xi_2$, то выдать $a + (b - a) * \xi_1$ в качестве результата, иначе повторить второй шаг

Дискретное распределение с возвратом.

Если x - дискретная случайная величина, которая принимает значения $x_1 < x_2 < \dots < x_i < \dots < x_n$ с вероятностями $p_1 < p_2 < \dots < p_i < \dots < p_n$, то таблица ниже показывает распределение дискретной случайной величины.

x_1	x_2	…	x_i	…	x_n
p_1	p_2	…	p_i	…	p_n

Таблица 1 – распределение дискретной случайной величины.

Функция распределения принимает следующий вид:

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

Дискретное распределение без возврата.

При данном распределении вероятность выбора каждой последующей случайной величины уменьшается, так как уменьшается и общее количество величин. Для проектирования выбиралось $^3/_4 n$ следующих величин без повторений большое количество раз и считались частоты повторений этих величин.

Результаты проектирования

Для построения плотности распределения была взята следующая функция:

$$f(x) = \begin{cases} 0, & \text{если } x \leq 3 \\ a * \sqrt{x+1}, & \text{если } 3 \leq x \leq 8 \\ 0, & \text{если } x > 8 \end{cases}$$

Несобственный интеграл от плотности — это истинное событие:

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

поэтому найдем a:

$$\int_{-\infty}^{3} 0dx + \int_{3}^{8} a\sqrt{x+1} + \int_{8}^{\infty} 0dx = 1$$

$$a\frac{2}{3}\sqrt{(x+1)^3}\bigg|_3^8 = 1$$

$$\left(a\frac{2}{3}\sqrt{9^3}\right) - \left(a\frac{2}{3}\sqrt{4^3}\right) = 1$$

$$\frac{54}{3}a - \frac{16}{3}a = 1$$

$$\frac{38}{3}a = 1$$

$$a = \frac{3}{38}$$

тогда плотность распределения:

$$f(x) = \begin{cases} 0, & \text{если } x \leq 3 \\ \frac{3}{38}\sqrt{x+1}, & \text{если } 3 \leq x \leq 8 \\ 0, & \text{если } x > 8 \end{cases}$$

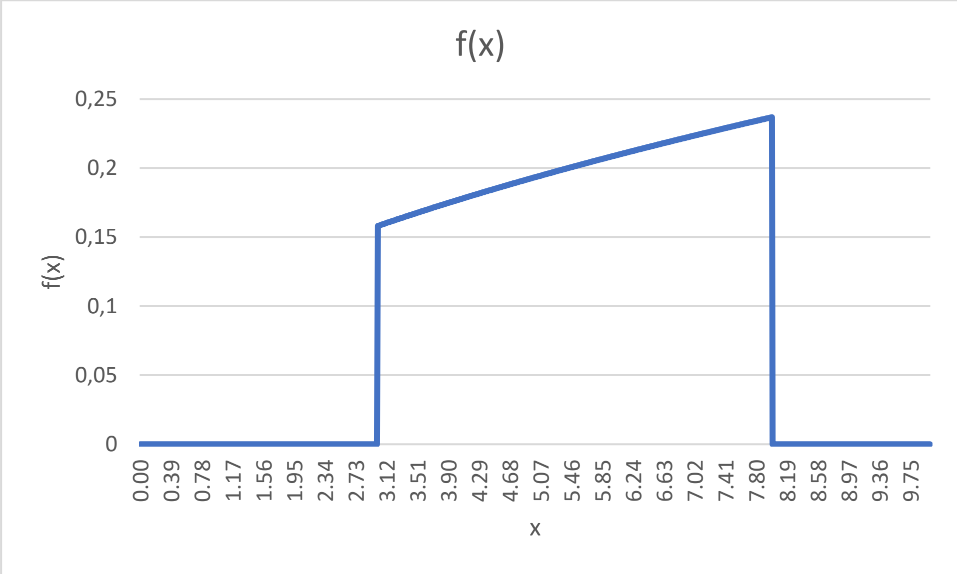

Рисунок 1 – график функции распределения.

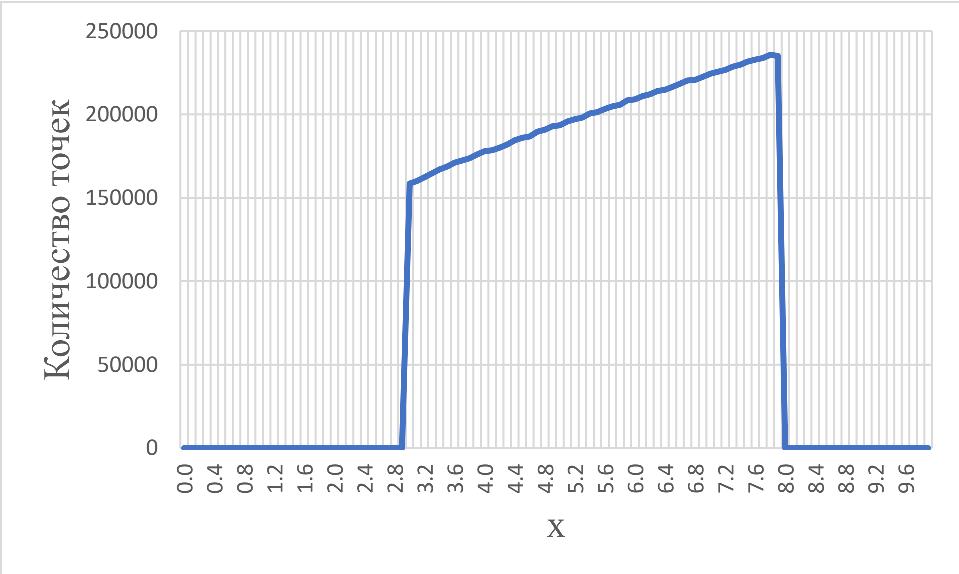

Рисунок 2 – график плотности распределения случайных величин, построенный с помощью метода отбраковки.

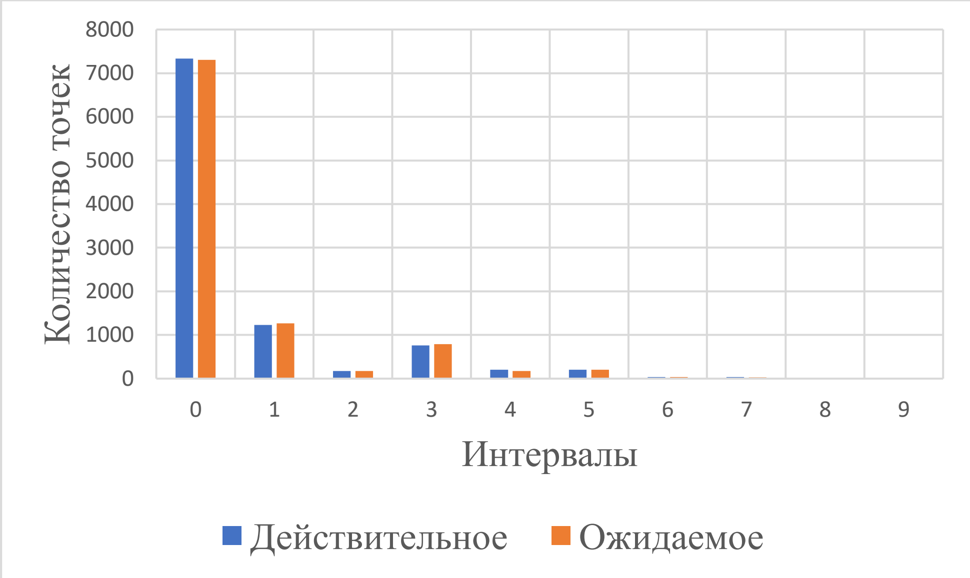

Рисунок 3 – гистограмма плотности дискретного распределения случайных величин с возвращением.

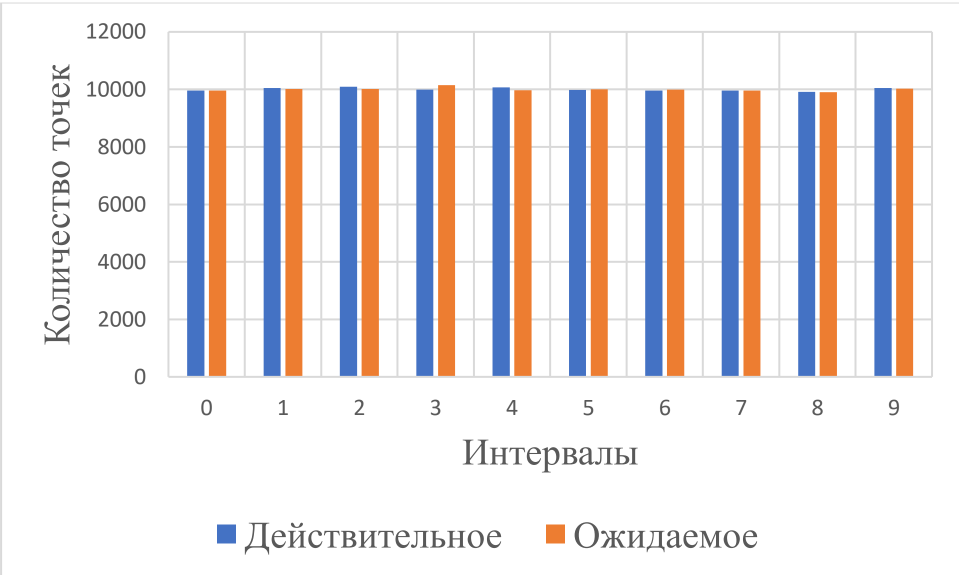

Рисунок 4 – гистограмма плотности дискретного распределения случайных величин без возвращения.

Выводы

Исходя из полученных результатов и построенных графиков про метод отбраковки можно сказать, что он отработал корректно. Аналитически можно убедиться в схожести графиков теоретической плотности распределения $f(x)$ и полученной методой отбраковки.

Также в ходе исследования данного метода была замечена негативная для производительности особенность. Так как метод отбраковки требует попадания случайного числа в заданные границы, получается, что числа, не попавшие в границы, отбрасываются. Следовательно, получается увеличение необходимых производительных и временных ресурсов.

Исходя из результатов и гистограмм, полученных в ходе проектирования дискретного распределения с возвращением и без возвращения, можно сказать, что между ожидаемым и действительным результатами есть незначительная погрешность. Следовательно, можно сделать вывод о том, что используемый генератор случайных чисел имеет равномерное распределение.

Приложение

1. Листинг программы

```
import numpy as np


def get_rand_double():
    return np.random.random()


def get_rand_int(low=0, high=100):
    return np.random.randint(low=0, high=high)


def functor(x):
    if 3 <= x <= 8:
        return (np.sqrt(x + 1) * 3) / 38
    return 0


def func(x):
    if 3 <= x <= 8:
        return np.sqrt(x + 1)
    return 0


def reject_method(a, b, c):
    x1 = 0.0
    condition = True
    while condition:
        x1 = get_rand_double()
        x2 = get_rand_double()
        condition = functor(a + (b - a) * x1) <= c * x2

    return a + (b - a) * x1


def start_distribution(n):
    d = [0] * n
    residue_chance = 1.0
    for i in range(n-1):
        d[i] = np.absolute(np.remainder(get_rand_double(), residue_chance))
        residue_chance -= d[i]
    d[n - 1] = residue_chance
    return d


def return_method(probability, MAX, n):
    hits = [0] * n
    for i in range(MAX):
        rv = get_rand_double()
```

```
        acc = 0.0
        for j in range(n):
            acc += probability[j]
            if rv < acc:
                hits[j] += 1
                break
    return hits


def without_return(MAX, n):
    hits = [0] * n
    k = int(3 * n / 4)
    part = int(MAX / k + 1)
    for j in range(part):
        nums = np.arange(0, n)
        if j == part - 1:
            k = MAX % k
        for i in range(k):
            idx = int(get_rand_double() * (n - i))
            hits[nums[idx]] += 1
            nums = np.delete(nums, idx)
    return hits


def test_of_functor():
    N = 10
    x = 0.0
    file = open("test_functor.txt", "w")
    file.write("x f(x)\n")
    while x < N:
        file.write("{0:.2f} {1:.2f}\n".format(x, functor(x)))
        x += 0.001


def test_of_reject_method():
    file = open("test_reject_method.txt", "w")

    count = 100
    hits = [0] * count
    max = 100
    for i in range(max * 1000):
        rand_number = reject_method(3.0, 8.0, 0.24)
        hits[int(rand_number * 10)] += 1

    file.write("n\tx\n")
    for i in range(0, count, 1):
        file.write("{0:.1f} {1:d}\n".format(i / 10.0, hits[i]))


def test_of_return():
    file = open("test_of_return.txt", "w")
    n = 10
```

```
    max = 1000
    probabilities = start_distribution(n)
    hits = return_method(probabilities, max, n)

    file.write("i\tActual\tExpected\n")
    for i in range(len(hits)):
        file.write("{0:d} {1:d} {2:d}\n".format(i, hits[i], int(probabilities[i] * max)))


def test_of_without_return():
    max = 1000
    n = 10
    hits = without_return(max, n)
    counts = get_list_of_unique_numbers(max, n)

    file = open("test_of_without_return.txt", "w")

    file.write("i\tActual\tExpected\n")
    for i in range(n):
        file.write("{0:d} {1:d} {2:d}\n".format(i, hits[i], counts[i]))


def get_list_of_unique_numbers(n, max):
    counts = {i: 0 for i in range(max)}
    randlist = [get_rand_int(low=0, high=max) % max for _ in range(n)]
    for rand in randlist:
        counts[rand] += 1
    return counts


def test_of_func():
    N = 10
    x = 1.0
    file = open("test_func.txt", "w")
    file.write("x f(x)\n")
    while x < N:
        file.write("{0:.2f} {1:.2f}\n".format(x, func(x)))
        x += 0.025


if __name__ == "__main__":
    test_of_func()
    test_of_functor()
    test_of_reject_method()
    test_of_return()
    test_of_without_return()
```