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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

По дисциплине «Моделирование»

Выполнил:
Студент гр. ИВ-622
Евтушенко Н. А.

Проверила:
Ассистент Кафедры ВС
Петухова Я. В.

Новосибирск 2020

Содержание

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. Метод отбраковки

Законы распределения вероятности могут быть заданы различными функциями. Поэтому надо уметь превращать равномерный ГСЧ в такой генератор случайных чисел, который задан произвольным законом распределения. Один из методов для моделирования непрерывной случайной величины является метод отбраковки. Суть метода заключается в том, что функция плотности распределения вероятностей случайных величин $f_\eta(x)$ вписывается в прямоугольник $(a, b) \times (0, c)$, такой что a и b соответствует границам диапазона изменения случайных величин η, а с – максимальному значению функции плотности её распределения. Тогда очередная реализация случайных величин определяется по следующему алгоритму:

- Выбрать функцию плотности распределения случайных величин
- Получить два независимых случайных числа ξ_1 и ξ_2
- И если $f_\eta\,(a + (b - a) * \xi_1) > c * \xi_2$ то выдать $(a + (b - a) * \xi_1)$ в качестве результата, иначе повторить шаг 2.

2. *Дискретное распределение с возвратом*

Случайная величина X называется дискретной, если она может принимать дискретное множество значений ($x1, x2, x3, \dots, xn$). Дискретная случайная величина описывается с помощью таблицы (распределение случайной величины X) или в виде аналитической формулы.

$$X = \begin{pmatrix} x_1 \; x_2 \; x_3 \; \dots \; x_n \\ p_1 \; p_2 \; p_3 \; \dots \; p_n \end{pmatrix} \quad \text{, где}$$

$(x_1, x_2, x_3, \dots, x_n)$ – возможные значения величины X

$(p_1, p_2, p_3, \dots, p_n)$ – соответствующие им вероятности:

$$P(X = x_i) = p_i$$

Числа могут быть любыми, а вероятности должны удовлетворять двум условиям:

$$p_i > 0 \text{ и } \sum_{i=1}^{n} p_i = 1$$

Функция распределения принимает такой вид:

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

Наиболее общий вид построения генератора дискретных случайных величин основывается на следующем алгоритме. Пусть имеется таблица пара (xi, pi). Тогда сумму можно представить интервалом [0,1), разбитым на полуинтервалы $[\nu_{i-1}, \nu_i), \nu_0 = 0, \nu_n = 1$ длины p_i. Случайная величина X с неизбежностью принадлежит одному из этих полуинтервалов, тем самым определяя индекс дискретного значения. Номера полуинтервала определяется как:

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

3. Дискретное распределение без возврата

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

Ход работы

Определим плотность распределения:

$$f(x) = \begin{cases} 0, & x \leq 0 \\ a * \dfrac{x+1}{3}, & 0 < x \leq 2 \\ 0, & x \geq\ 2 \end{cases}$$

Найдём неизвестное a, используя свойство нормированности: $\int_{-\infty}^{\infty} f(x)dx = 1$

Расчет коэффициента:

$$\int_{-\infty}^{0} 0dx + \int_{0}^{2} a * \frac{x+1}{3} dx + \int_{2}^{\infty} 0dx = a * \frac{x+1}{3}\ \Big|_{0}^{2} = 1$$

Таким образом коэффициент *a* равен: *a* = 0.75

$$f(x) = \begin{cases} 0, & x \leq 0 \\ 0.75 * \dfrac{x+1}{3}, & 0 < x \leq 3 \\ 0, & x > 3 \end{cases}$$

По найденной плотности распределения можно построить график изображённый на рисунке:

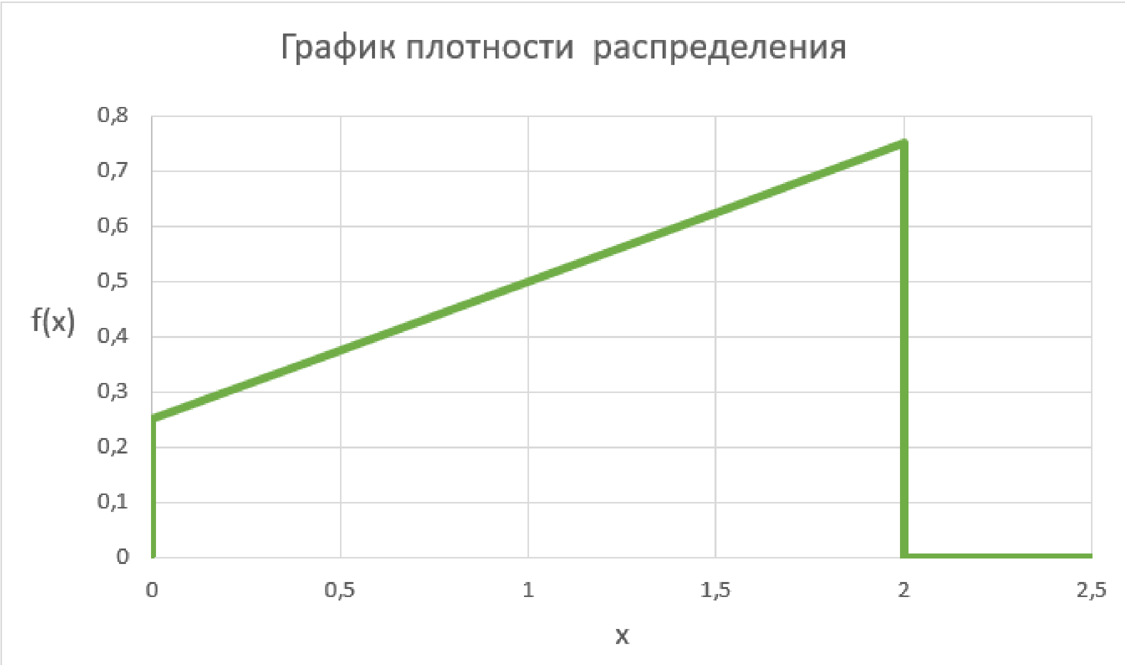

Рисунок 1 – График плотности

Рисунок 2 – Результат работы метода отбраковки

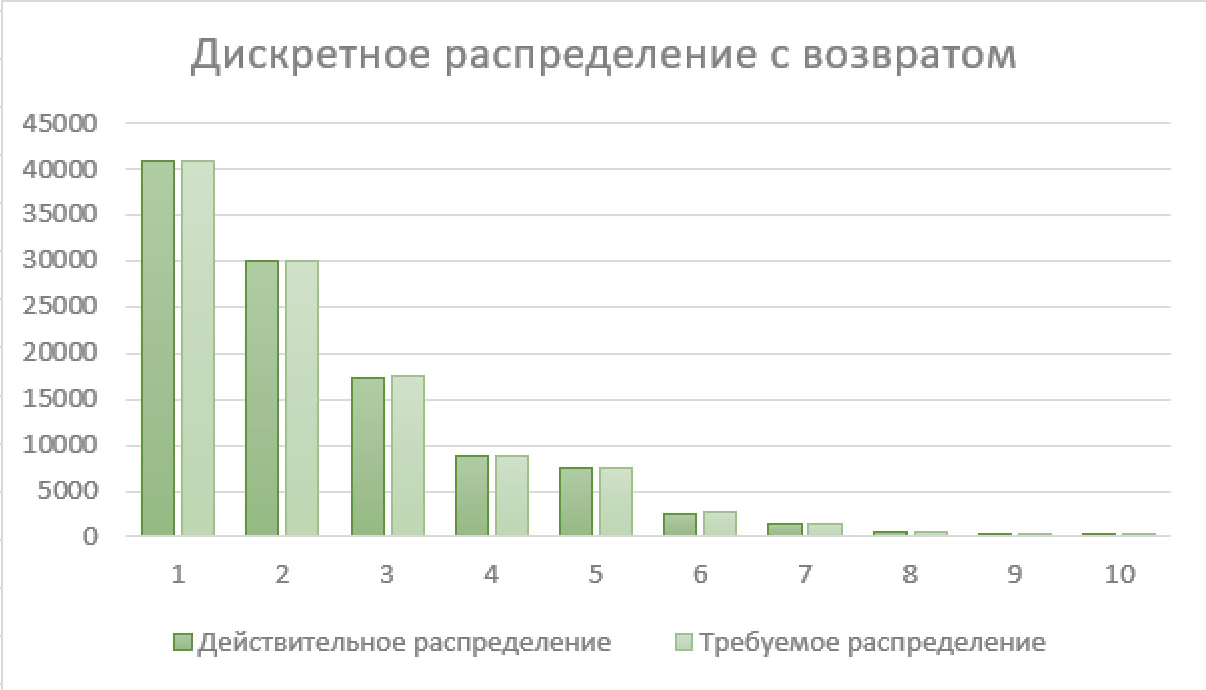

Рисунок 3 – Дискретное распределение с возвратом

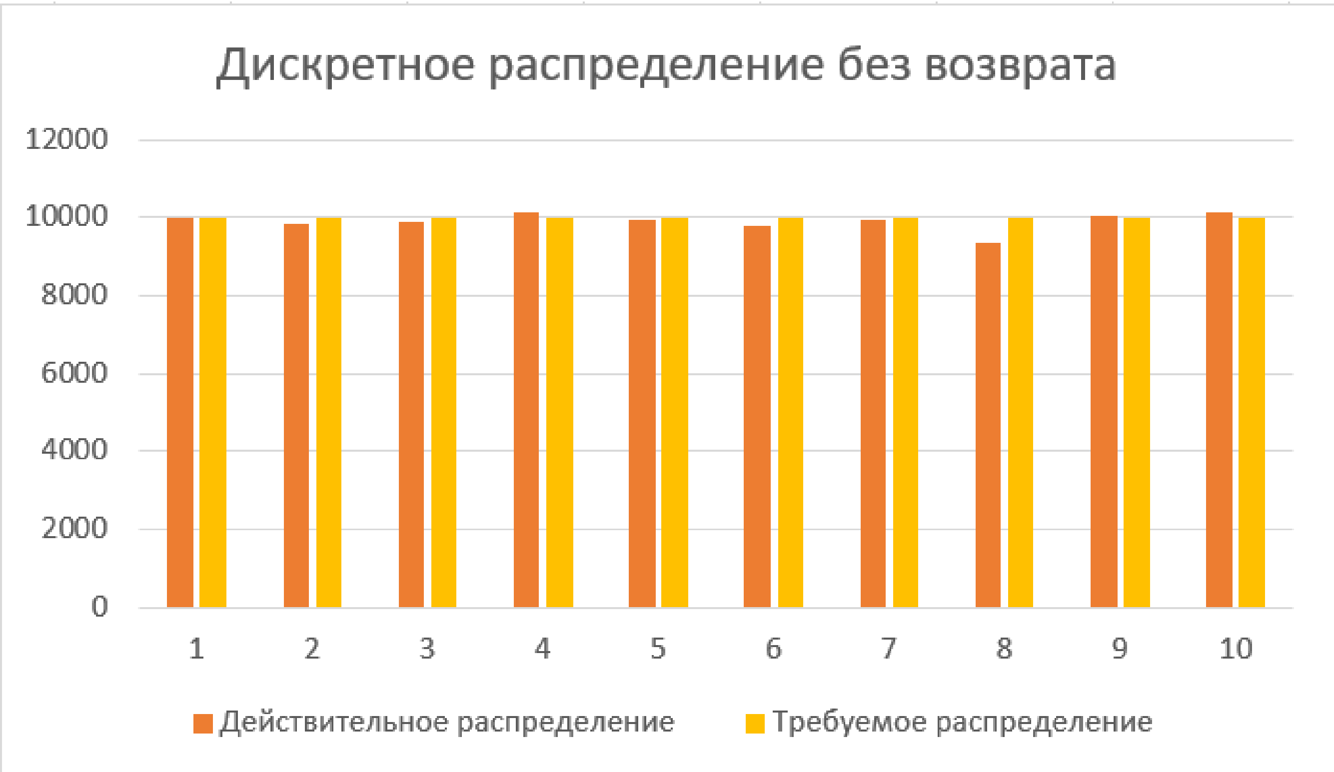

Рисунок 4 – Дискретное распределение без возврата

Заключение

В рамках лабораторной работы были изучены:

- Метод отбраковки
- Дискретное распределение без возврата
- Дискретное распределение с возвратом

В качестве генератора случайных чисел был выбран стандартный генератор языка Python.

Основываясь на результаты проделанной работы видно, что исходную функцию плотности метод смог смоделировать достаточно точно, но есть определённые недостатки:

1. Тратится большое количество времени на генерацию точек, которые не попали на кривую распределения плотности вероятности. Что сказывается на скорости работы алгоритма.
2. Метод применим только для аналитических функций. У алгоритма слабая эффективность для распределений с длинными «хвостами», так как в этом случае увеличивается частота повторных испытаний.

По результатам моделирования дискретных случайных величин с реализацией выборки с возвратом и без возврата можно следующие выводы: исследуемый генератор случайных чисел выдает распределение близкое к равномерному, так как действительное распределение приближенно к требуемому.

Листинг

lab_1.py

```
import random
import typing as tp

N = 1000000
K = 100
a_ = 0.75


def f(x: float) -> float:
    if x > 0 and x <= 2:
        return a_ * ((x + 1) / 3)
    else:
        return 0


def method_rejections() -> float:
    array = []
    a = 0
    b = 2
    c = 1
    while True:
        x1 = random.uniform(0, 1)
        x2 = random.uniform(0, 1)
        tmp = f(a + (b - a) * x1)
        if tmp > c * x2:
            return a + (b - a) * x1


def distribution_density() -> tp.List[int]:
    hit = [0] * K
    x = []
    y_1 = []
    m = 2 / K
    f = open('density.txt', 'w')
    for i in range(N):
        rand = method_rejections()
        x.append(rand)
        y = a_ * (x[i] + 5)
        y_1.append(y)
        inter = rand / m
        hit[int(inter)] += 1
        f.write(str(x[i]) + '\t' + str(y) + '\n')
    f.close()
    return hit


def repeat() -> tp.List[int]:
    residue_chance = 1.0
    chance_hit = [0] * K
    hit = [0] * K

    for i in range(K - 1):
        chance_hit[i] = random.uniform(0, 1) % residue_chance
        residue_chance -= chance_hit[i]
```

```python
    chance_hit[-1] = residue_chance

    for i in range(N):
        chance = random.uniform(0, 1)
        sum = 0.0
        for j in range(K):
            sum += chance_hit[j]
            if (chance < sum):
                hit[j] += 1
                break

    for i in range(K):
        chance_hit[i] = chance_hit[i] * 100000
    f = open('repeat.txt', 'w')
    i = 0
    while i != K:
        f.write(str(hit[i]) + '\t' + str(chance_hit[i]) + '\n')
        i += 1
    f.close()
    return hit, chance_hit


def no_repeat() -> tp.List[int]:
    k = 3 / 4 * K
    hit = [0] * K
    part = N / k + 1

    for i in range(int(part)):
        nums = [nums for nums in range(K)]
        if i == part - 1:
            k = N % k
        for j in range(int(k)):
            p = 1.0 / (K - j)
            it = int(random.uniform(0, 1) * 1.0 / p)
            hit[nums[it]] += 1
            nums.pop(it)
    f = open('norepeat.txt', 'w')
    i = 0
    while i != K:
        f.write(str(hit[i]) + '\t' + str(N / K) + '\n')
        i += 1
    f.close()
    return hit


distribution_density()
repeat()
no_repeat()
```