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

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
по дисциплине «Моделирование»

Выполнил:
Студент гр. ИВ-622
Курзин А.С.

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. 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 следующий метод. Функция плотности распределения вероятностей с.в. fη(x) вписывается в прямоугольник (a, b) × (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:

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 Шаг 2

2. Дискретное распределение с возвратом

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

Если x - дискретная случайная величина, принимающая значения $x_1 < x_2 <$ … $< x_i <$ … с вероятностями $p_1 < p_2 <$ … $< p_i <$ …, то таблица вида

называется распределением дискретной случайной величины.

Функция распределения случайной величины, с таким распределением, имеет вид

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

3. **Дискретное распределение без возврата**

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

ХОД РАБОТЫ

Определим плотность распределения:

$$f(x)=\begin{cases}0, & x<0\\ a(x+5), & 0<x\le 2\\ 0, & x\ge 2\end{cases}$$

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

Расчет коэффициента:

$$\int_{-\infty}^{0} 0\,dx+\int_{0}^{2} a(x+5)\,dx+\int_{2}^{+\infty} 0\,dx=1$$

$$0+a\left(\frac{x^2}{2}+5x\right)\Bigg|_0^2+0=1$$

$$12a=1$$

$$a=\frac{1}{12}$$

Таким образом коэффициент a равен: $\frac{1}{12}$

$$f(x)=\begin{cases}0, & x<0\\ \frac{1}{12}(x+5), & 0<x\le 2\\ 0, & x\ge 2\end{cases}$$

По найденной плотности распределения можно построить график изображённый на рисунке №1.

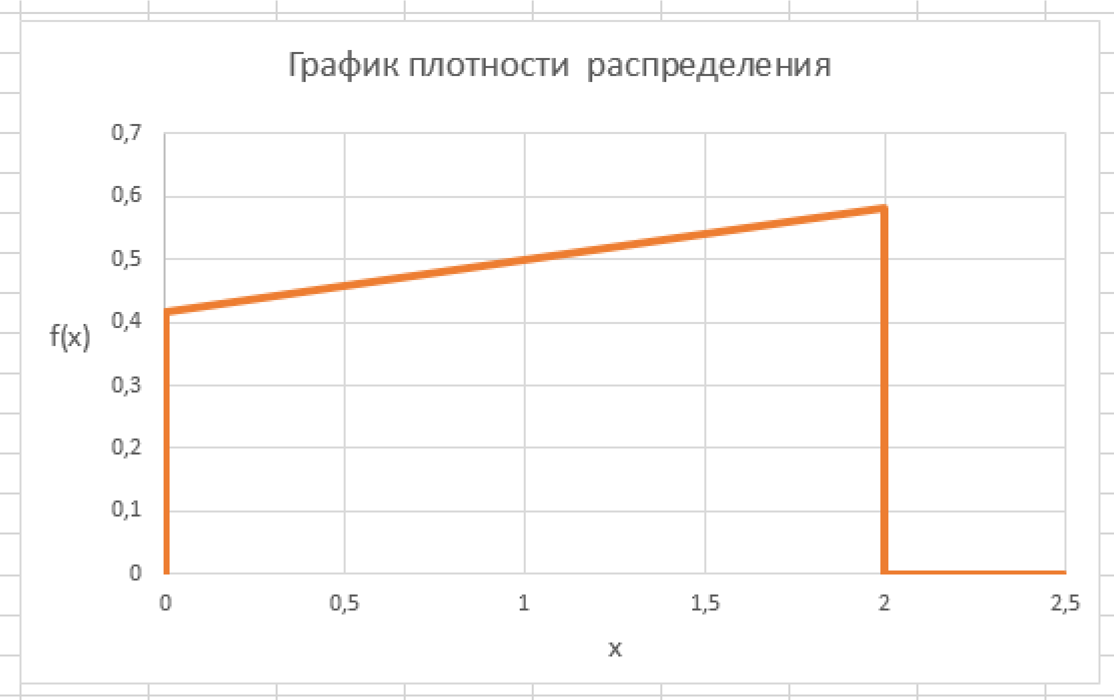

Рисунок № 1 - График плотности исходного распределения

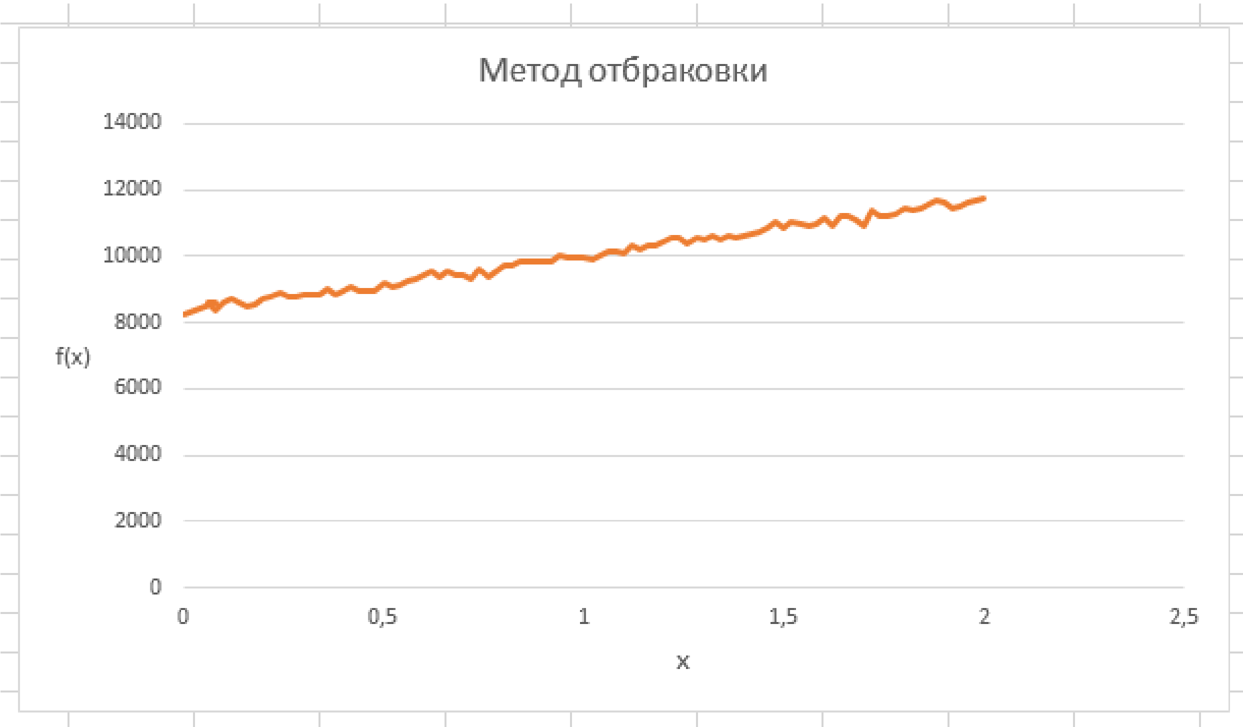

Рисунок № 2 - Результаты работы метода отбраковки

Рисунок № 3 - Дискретное распределение с возвратом

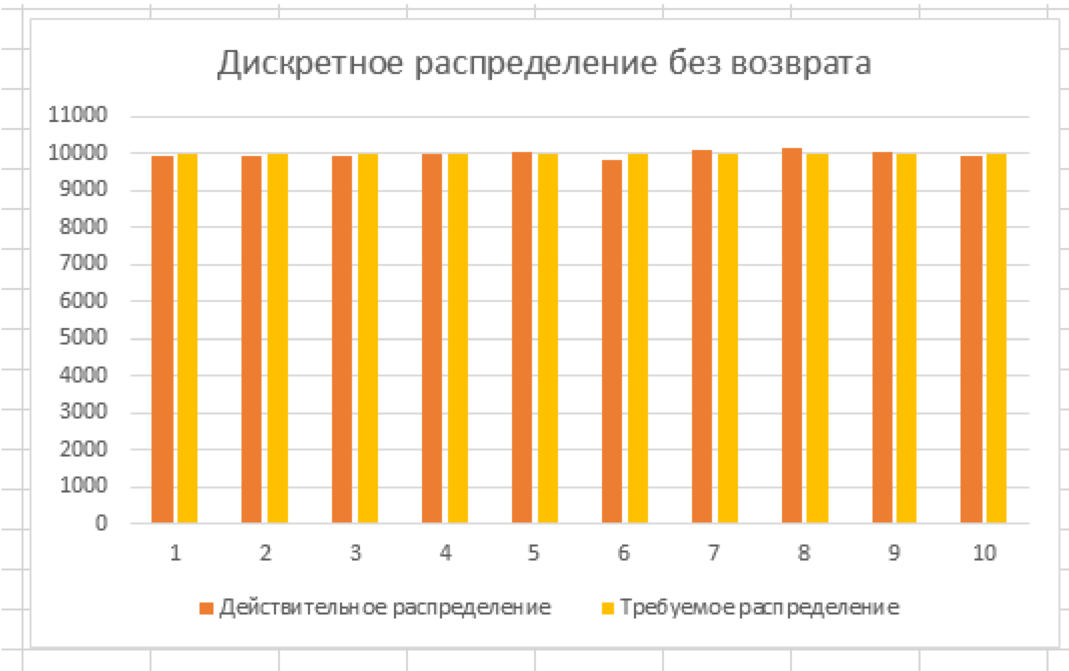

Рисунок № 4 - Дискретное распределение без возврата

ЗАКЛЮЧЕНИЕ

В ходе данной лабораторной работы были изучены такие методы моднлирования случайных величин: метод отбраковки, дискретное распределение с возвратом и дискретное распределение без возврата. В качестве генератора случайных чисел был выбран стандартный генератор языка Python.

По результатам нашей работы можно сделать вывод, что метода отбраковки текущую функцию плотности смог смоделировать достаточно точно, правда есть некоторые недостатки:

1. Точки, не попавшие на кривую распределения плотности вероятности, отбрасываются, но на их генерацию затрачивается время. Что не может не сказаться на общем времени 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.

Исходя из результатов моделирования дискретных случайных величин реализации выборки с возвратом и без возврата, очевидны следующие выводы: так как действительное распределение приближенно к требуемому, исследуемый генератор случайных чисел выдает распределение близкое к равномерному.

ЛИСТИНГ

```
import random
import typing as tp


N = 1000000
K = 100
a_ = 1 / 12


def f(x: float) -> float:
	if x > 0 and x <= 2:
		return a_ * (x + 5)
	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 = []
	m = 2 / K
	for i in range(N):
	    rand = method_rejections()
	    x.append(rand)
	    inter = rand / m
	    hit[int(inter)] += 1

	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]
    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

    return 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)

    return hit



print(distribution_density())
```