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

Кафедра вычислительных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к первой лабораторной работе по дисциплине
«Моделирование»

Выполнил:
студент гр. ИВ-621
Евтушенко Д.А

Проверила:
Ассистент кафедры ВС
Петухова Я.В.

Новосибирск, 2020

Содержание

Задание на проектирование

Генерация независимых, одинаково распределенных случайных величин

● Непрерывное распределение с помощью метода отбраковки

● Дискретное распределение с возвратом

● Дискретное распределение без возврата

Теория по проектной части

● 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 $f_n(a + (b - a)\xi 1) > c\xi 2$, то выдать $a + (b - a)\xi 1$ в качестве результата. Иначе повторить Шаг 2.

● Дискретное распределение с возвратом

Если x - дискретная случайная величина, принимающая значения x1 < x2 < … < x i < … с вероятностями p1 < … < pi < …, то таблица вида

x_1	x_2	…	x_i
p_1	p_2	…	p_i

называется распределением дискретной случайной величины.

Функция распределения случайной величины, с таким распределением, имеет вид

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

● 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.

Результаты проектирования:

● Непрерывное распределение с помощью метода отбраковки

Случайная величина X задана плотностью вероятности:

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

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

$$\int_{-\infty}^{\infty} f(x)dx = \int_1^3 a(x-1)dx = a\int_1^3 (x-1)dx = a\left(\frac{1}{2}x^2 - x\right)\Big|_1^3 = a\left(\frac{1}{2}9 - 3 - \frac{1}{2} + 1\right) = 2a = 1$$

Откуда a = 0.5.

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

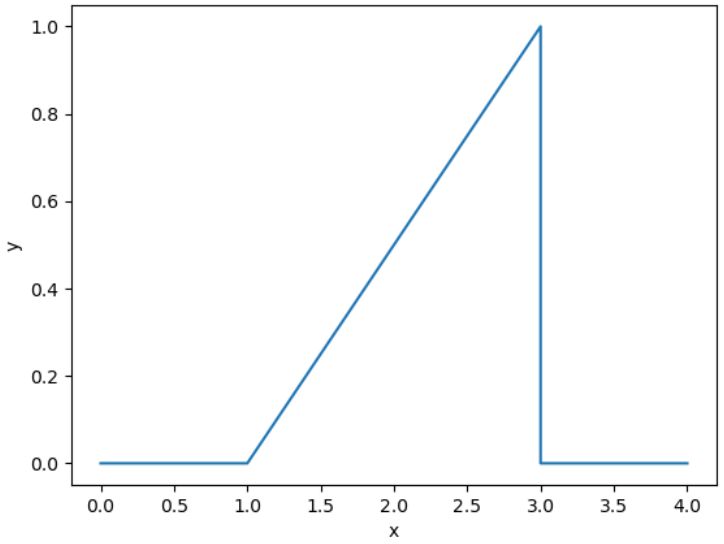

Рисунок 1 График функции $f(x)$

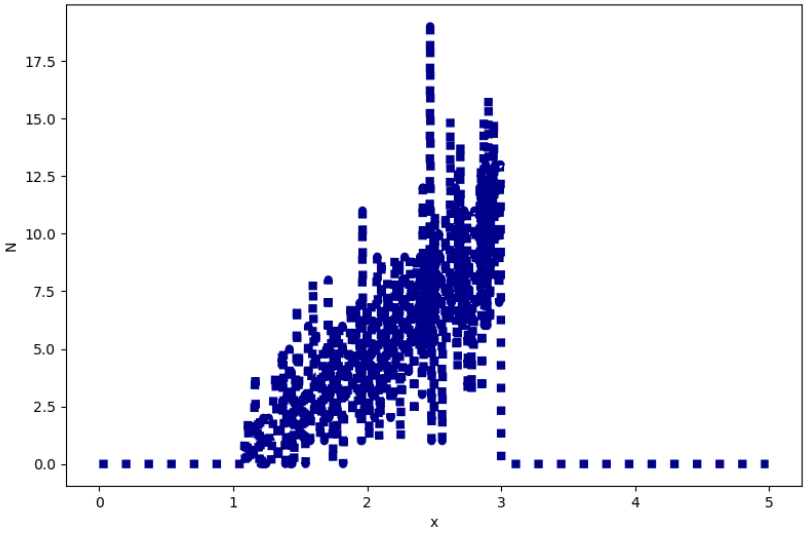

Рисунок 2 График метода отбраковки при значениях **N=10000,** кол-во точек **= 1000**

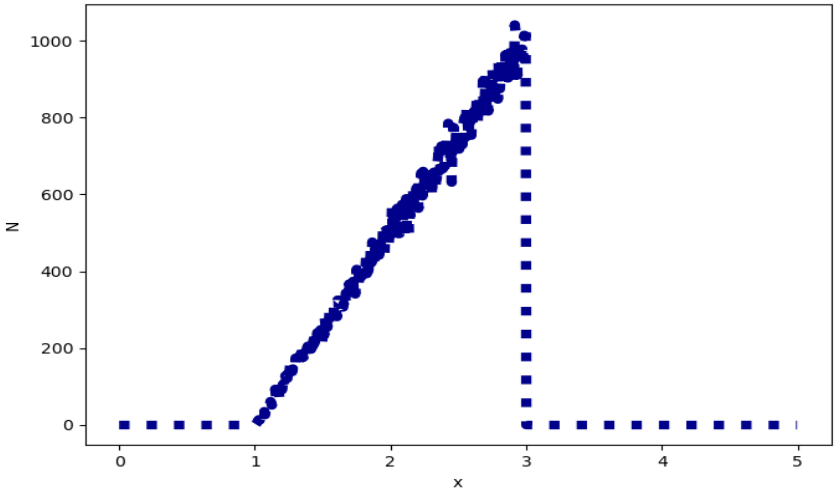

Рисунок 3 График метода отбраковки при значениях **N=1000000,** кол-во точек **= 1000**

● Дискретное распределение с возвратом

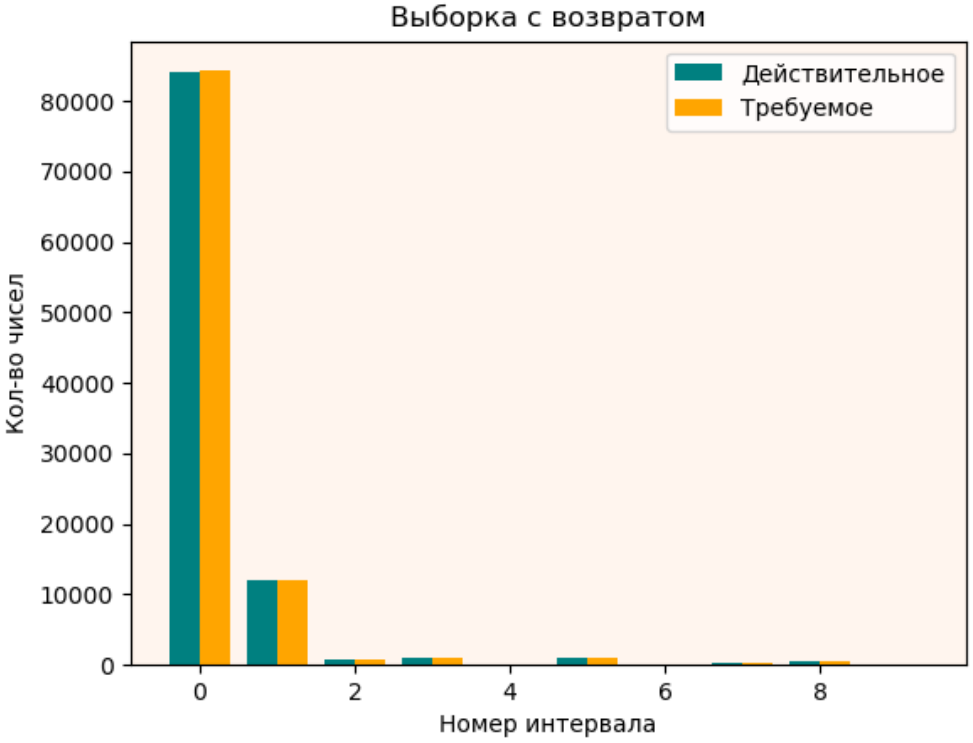

Рисунок 4 Дискретное распределение с возвратом

●Дискретное распределение без возврата

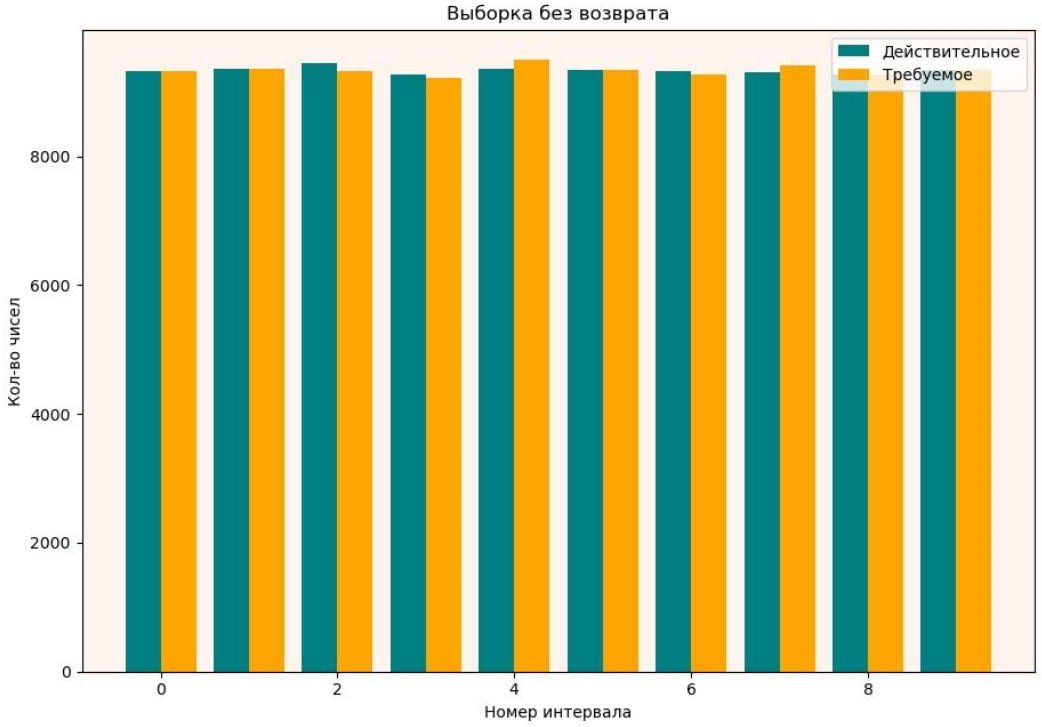

Рисунок 5 Дискретное распределение без возврата

Выводы по результатам проектирования

В ходе работы были изучены методы моделирования случайных величин, реализованы соответствующие алгоритмы. Для реализации алгоритмов был использован язык Python 3.7 с библиотеками для линейной алгебры NumPy (стандартным генератор случайных чисел Mersenne Twister из этой же библиотеки), для визуализации использована библиотека matplotlib, для структурирования данных в виде таблиц pandas.

По результатам проектирования метод отбраковки показал свою эффективность, полученные графики сопоставимы с исходным графиком плотности распределения. Анализируя графики метода отбраковки можно сделать вывод, что точность метода отбраковки при увеличении объема выборки будет выше. Неэффективностью метода отбраковки являются избыточные вычисления тех точек, которые попали выше требуемой области, так как они отбрасываются как не нужные. Метод применим только для аналитических функций плотности вероятности. Практические значения, полученные для дискретного распределения с возвратом и без возврата в сравнении с теоритическими значениями имеют достаточно малое отклонение. Из этого можно сделать вывод, что ГСЧ выдает распределение близкое к равномерному с малой долей погрешности.

Листинг программы

```
import random
import math
import _random
import pandas as pn
import numpy as np
import time
import matplotlib.pyplot as plt

def func(x):
    y = 0.0
    a = 0.5
    if (1.0 <=  x and x <=3.0):
         y = a * (x - 1)
    else:
        y = 0.0

    return y

def method_reject(a, b, c):
    while True:
        x1 = random.random()
        x2 = random.random()
        if (func(a + (b-a) * x1)) >= c*x2:
            break

    y = a + (b - a) * x1

    return y


def continuous_distribution(N):
    mass = [0] * N * 100
    for i in range(100 * N ):
        print(i)
        p = method_reject(0.0, 5.0, 1.0)
        it = int(p*1000)
        mass[it] += 1

    print(mass)
    massx = []
    massy = []
    for i in range(0, 5000, 5):
        print(i/1000)
        massx.append(i/1000)
        print("\t", mass[i])
        massy.append(mass[i])
```

```
    x = [0,1,2,3, 3, 4]
    y = [0,0,0.5,1,0, 0]
    plt.plot(x, y,)

    plt.xlabel('x')
    plt.ylabel('y')

    plt.show()

def get_dist(N):
    mass = [0] * N
    residue_chance = 1

    for i in range(N-1):
        probability = abs(np.random.rand() % residue_chance)
        # print(random.random() % residue_chance)
        # print(probability)
        mass[i] = probability
        residue_chance -= probability

    mass[N-1] = residue_chance
    return mass

def repeat(n, max):
    distribution = get_dist(n)
    print(distribution)
    chance = 0.0
    mass = [0] * n
    for i in range(max):
        chance = np.random.rand()

        sum = 0
        for j in range(n):
            sum += distribution[j]
            if chance < sum:
                mass[j] += 1
                break
    for k in range(n):
        print("sdfa",mass[k])

    fig, ax1 = plt.subplots()
    res = pn.Series(mass)
    nedd = np.multiply(distribution, max)
    res1 = pn.Series(nedd)
    ax1.bar(np.arange(0, n) - 0.2, res, width=0.4, color='teal',
label='Действительное')
    ax1.bar(np.arange(0, n) + 0.2, res1, width=0.4,
color='orange', label='Требуемое')
```

```
    ax1.set(title='Выборка с возвратом', xlabel='Номер
интервала', ylabel='Кол-во чисел')
    ax1.legend()
    ax1.set_facecolor('seashell')
    plt.show()

def no_repeat(n, max):

    mass = [0] * n
    k = (3 * n) / 4
    part = max / k + 1
    treb = int(part) * int(k)
    for j in range(int(part)):
        mass_temp = np.arange(0, n)
        if j == int(part) - 1:
            k = max % k
        # print(k)
        for p in range(int(k)):
            distribution_unit = 1.0 / (n - p)
            probability = np.random.rand()
            ratio = probability / distribution_unit

            it = int(ratio)
            mass[mass_temp[it]] += 1

            mass_temp = np.delete(mass_temp, it)

    print(np.sum(mass))
    fig, ax1 = plt.subplots()
    res = pn.Series(mass)
    x = np.random.randint(1, 11, size=max)
    print(x)
    uniq, counts = np.unique(x, return_counts=True)
    print(counts)
    pi = np.round(counts / max, 10)
    nedd = np.multiply(pi, treb)
    res1 = pn.Series(nedd)
    ax1.bar(np.arange(0, n) - 0.2, res, width=0.4, color='teal',
label='Действительное')
    ax1.bar(np.arange(0, n) + 0.2, res1, width=0.4,
color='orange', label='Требуемое')
    ax1.set(title='Выборка без возврата', xlabel='Номер
интервала', ylabel='Кол-во чисел')
    ax1.legend()

    ax1.set_facecolor('seashell')
    plt.show()
```

Список использованной литературы

1. http://statistica.ru/theory/raspredeleniya-veroyatnostey/
2. https://csc.sibsutis.ru/sites/csc.sibsutis.ru/files/courses/model/labs.pdf
3. https://www.statmod.ru/wiki/_media/books:vv:simulation_v4.pdf
4. http://stratum.ac.ru/education/textbooks/modelir/lection24.html