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
Шаврин К.С.

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 шаг 2.

● Дискретное распределение с возвратом

Если x - дискретная случайная величина, принимающая значения $x_1 < x_2 <$ ... $< x_i <$ ... с вероятностями $p_1 <$ ... $< p_i <$ ..., 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 3n / 4 следующих величин без повторений, проделываем это большое количество раз и считаем частоты этих значений.

Результаты

Для того, чтобы построить функцию распределения необходимо вычислить значение a функции плотности распределения. Сама наша плотность распределения f(x) выглядит следующим образом:

$$f(x) = \begin{cases} 0, & \text{при } x \le 1 \\ a * x^2, & \text{при } 1 < x \le 4 \\ 0, & \text{при } x > 4 \end{cases}$$

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

$$\int_{-\infty}^{+\infty} f(x)dx = \int_1^4 a * x^2 dx = a\int_1^4 x^2 dx = a * \frac{x^3}{3}dx\Big|_1^4 = a\left(\frac{4^3}{3} - \frac{1^3}{3}\right) = 21a = 1$$

a = $\frac{1}{21}$

Тогда:

$$f(x) = \begin{cases} 0, & \text{при } x \le 1 \\ \frac{1}{21} * x^2, & \text{при } 1 < x \le 4 \\ 0, & \text{при } x > 4 \end{cases}$$

Далее продемонстрирован график функции плотности распределения непрерывных случайных величин:

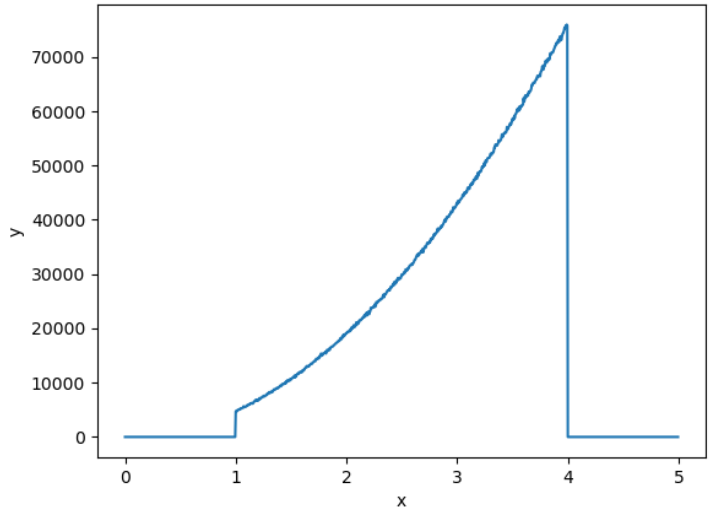

Рис.1. График значений $f(x)$ случайных величин

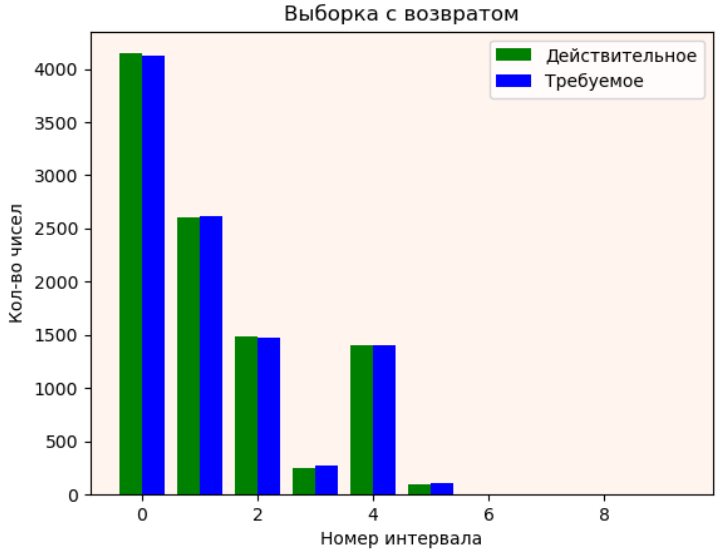

Рис.2. График дискретного распределения с возвратом

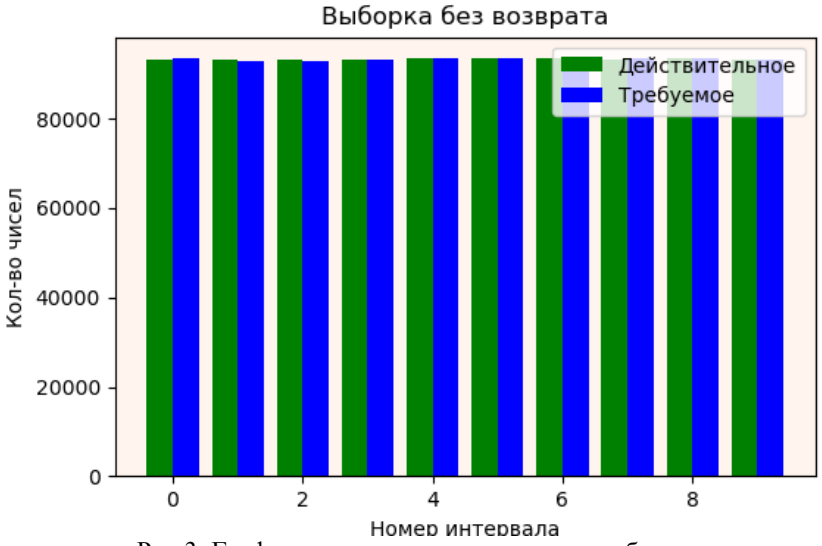

Рис.3. График дискретного распределения без возврата

Выводы по результатам проектирования

В ходе выполнения лабораторной работы были изучены методы моделирования случайных величин, реализованы алгоритмы с непрерывным распределением методом отбраковки и с дискретным распределением с возвратом и без возврата.

По результатам работы мы выяснили, что при генерации непрерывным распределением по методу отбраковки, значения соответствуют заданному закону распределения. 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.

При генерации дискретным распределением с возвратом и без него происходит расчет ряда вероятностей. Практические значения, полученные для дискретного распределения с возвратом и без возврата, в сравнении с теоретическими значениями имеют достаточно малое отклонение. Из этого можно сделать вывод, что ГСЧ выдает распределение близкое к равномерному.

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 = 1/21
	if (1.0 <= x and x <=4.0):
		y = a * x * x
	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 ):
		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):
		massx.append(i/1000)
		massy.append(mass[i])
	plt.plot(massx, massy,)
	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)
		mass[i] = probability
		residue_chance -= probability
	mass[N-1] = residue_chance
	return mass
def repeat(n, max):
	distribution = get_dist(n)
	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
	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='green', label='Действительное')
        ax1.bar(np.arange(0, n) + 0.2, res1, width=0.4, color='blue', 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
                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)
        fig, ax1 = plt.subplots()
        res = pn.Series(mass)
        x = np.random.randint(1, 11, size=max)
        uniq, counts = np.unique(x, return_counts=True)
        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='green', label='Действительное')
        ax1.bar(np.arange(0, n) + 0.2, res1, width=0.4, color='blue', label='Требуемое')
        ax1.set(title='Выборка без возврата', xlabel='Номер интервала', ylabel='Кол-во чисел')
        ax1.legend()
        ax1.set_facecolor('seashell')
        plt.show()
```