Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Лабораторная работа №1

«Получение базового распределения»

по дисциплине «Статистическое моделирование»

Выплолнил студент
гр. 33534/5 Стойкоски Н.С.

Руководитель Чуркин В.В.

Санкт-Петербург
2019 г.

Оглавление

Цель работы

1. Получение на ЭВМ с помощью программного датчика базовой последовательности псевдослучайных чисел, имеющих равномерное распределение.

2. Освоение методов статистической оценки полученного распределения: вычисление эмпирических значений для математического ожидания и дисперсии.

3. Освоение методов оценки статистики связи: вычисление значений автокорреляционной функции и построение коррелограммы.

4. Освоение методов графического представления законов распределения: построение функции плотности распределения и интегральной функции распределения.

Проведение работы

В программу на языке python была написана функция которая как параметр принимает длину последовательности которую нужно сгенероравать. Эта функция для начало генерирует N равномерно распределёныех случайных чисел в диапазоне от 0 до 1 с использованием функции random.random() из стандарнтой библиотеки. Далее вычисляются требуемые числовые характеристики – математическое ожидание и дисперсии для ранее сгенерированная последовательность чисел. Вычисляются значений автокорреляционной функции и строится коррелограмма при использований библиотеки matplotlib. Так же создается графическое представление законов распределния – графики эмпирических функции плотности распределения и интегральная функция распределения.

Вычисление эмпирических значений:

Математическое ожидание:

@M@ = (u[1] + u[2] + ... + u[n])/n,

где @M@ - математическое ожидание по результатам наблюдений;

u[1],u[2],...,u[n] - псевдослучайные величины, вырабатываемые датчиком случайных чисел;

n - число испытаний.

Дисперсия:

@D@={(u[1]-@M@)**2 +(u[2]-@M@)**2 + ... +(u[n]-@M@)**2}/n,

@S@ = sqrt(@D@),

где @D@ - эмпирическая дисперсия;

@S@ - эмпирическое среднее квадратическое отклонение;

u[1],u[2],...,u[n] - псевдослучайные величины, вырабатываемые датчиком случайных чисел.

N	**Оценка распределения**	**RAND (эксперимент)**	**Теоретическое значение**	**Отклонение**
10	Мат. ожидание	0.418023	0.5	0.081977
	Дисперсия	0.065968	0.08333	0.017362
100	Мат. ожидание	0.512406	0.5	0.012406
	Дисперсия	0.066386	0.08333	0.016944
1000	Мат. ожидание	0.494180	0.5	0.005820
	Дисперсия	0.084936	0.08333	0.001606
10000	Мат. ожидание	0,499654	0.5	0.000346
	Дисперсия	0.082810	0.08333	0.00052

Вычисление значение автокорреляционной функции

$$K[f] = \frac{\sum_{i=1}^{n-f}(u[i]-M)*(u[i+f]-M)}{\sum_{i=1}^{n}(u[i]-m)**2}$$

где K[f] - коэффициент корреляции (f=1,2,...,n);

@M@ - математическое ожидание по результатам наблюдений;

u[1],u[2],...,u[n] - псевдослучайные величины, вырабатываемые датчиком случайных чисел;

n - число испытаний.

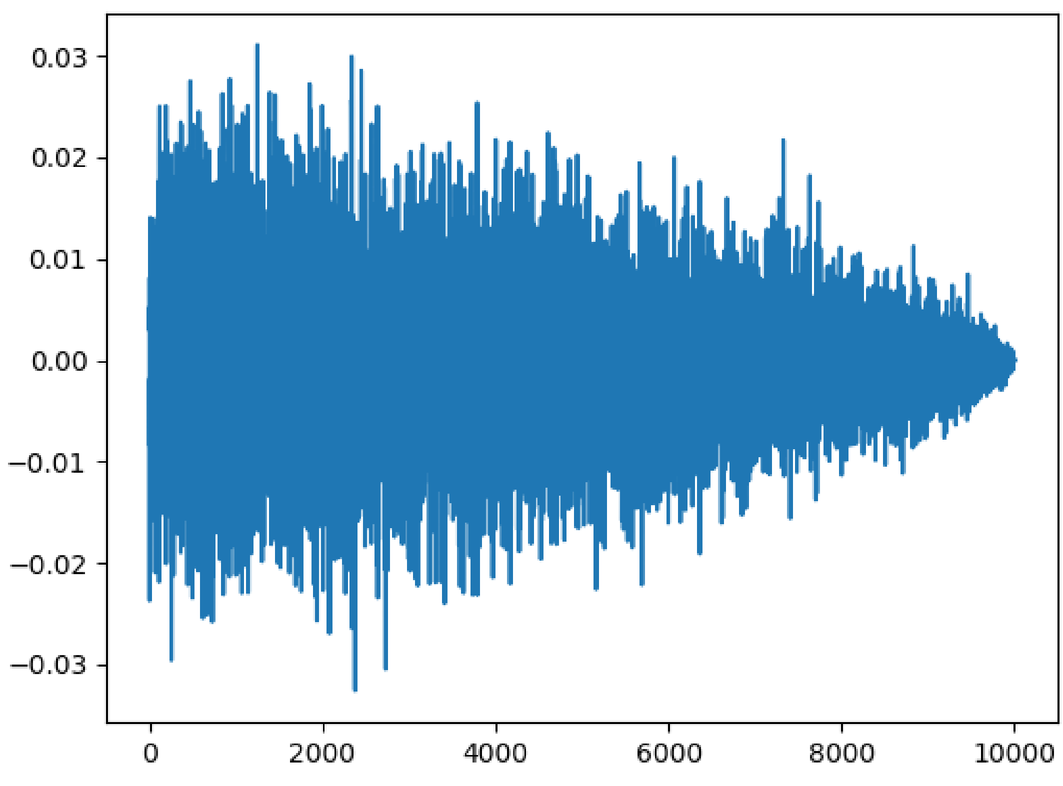

Рис.1 Коррелограмма

Графическое представление законов распределения

Сгенерированная выборка разделяется на n/10 равные по длину диапазоны. В каждом таком диапазоне считается количесво чисел попавщие туда. По данным строются графики эмпирических функции плотности и фунцкии распределения.

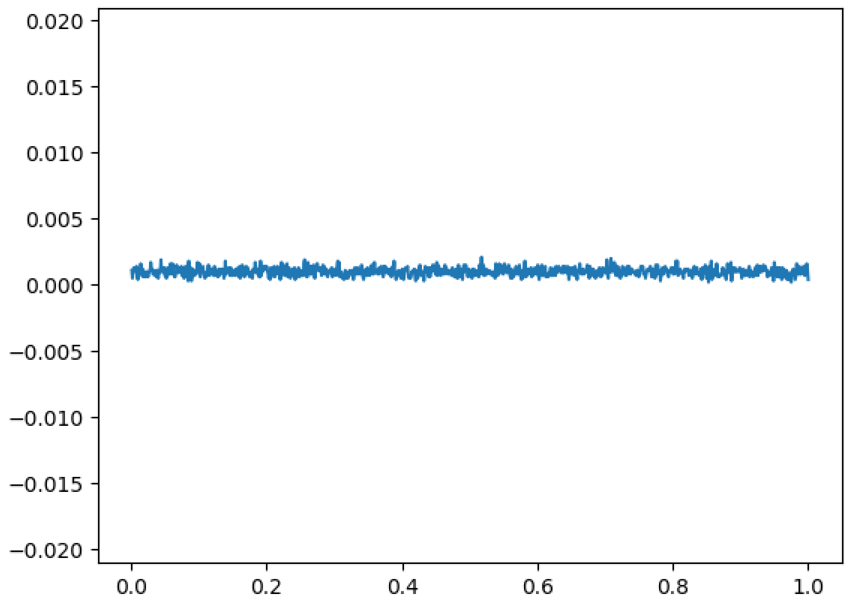

Рис.2 Эмпирическая функция плотности распределения

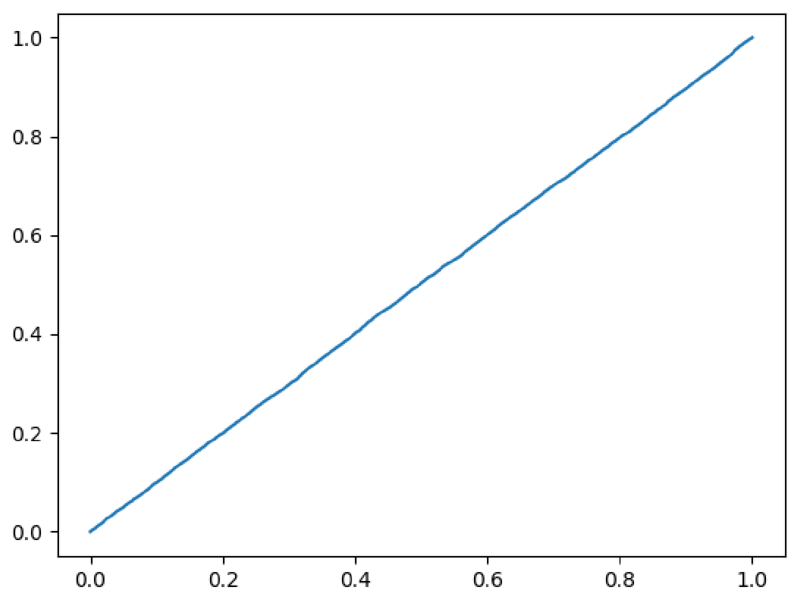

Рис.3 Эмпирическая интегральная функция распределения

Вывод

С помощю программного датчика была сгенерирована последовательность псевдослучайных чисел, имеющих равномерное распределение. Были вычислены эмпирических значений для математического ожидания и дисперсии для разных объёмов выборки. Результаты записани в виде таблички для искомые значений объёмов выборки. При возрастании объёма выборки, полученные значения стремятся к теоретические значения. Была построена коррелограмма которая отражает случайность чисел в исходной выборке. Графики функции распределения и плотности вероятности соответствуют кривым равномерного распределения.

Текст программы

```
import numpy as np
import matplotlib.pyplot as plt
import random
import math

def calc(n):
    u = [random.random() for _ in range(n)]
    m = np.sum(u) / n
    d = np.sum([ (x - m)**2 for x in u ]) / n
    s = math.sqrt(d)
    k = [ np.sum( [(u[i] - m)*(u[i+f+1] - m) for i in range(n-f-1)] ) / (d * n) for f
in range(n)]
    print(f' n = {n} \n M = {m} \n D = {d} \n S = {s}\n')

    plt.figure()
    plt.plot(range(n), k)
    plt.savefig(f'correlogram_{n}.png')

    #func raspredelenie
    u.sort()
    plt.figure()
    plt.plot(np.linspace(0, 1, n), u)
    plt.savefig(f'f_rasp_{n}.png')

    #func plotnost
    numIntervals = max(3, int(n/10))
    intervals = np.linspace(0, 1, numIntervals)
    cumulativeFreq = [(u < r).sum() / n for r in intervals]
    frequencies = [cumulativeFreq[i+1] - cumulativeFreq[i] for i in
range(len(cumulativeFreq) - 1)]
    plt.figure()
    plt.ylim(max(frequencies) * -10, max(frequencies) * 10)
    plt.plot(intervals[1:], frequencies)
    plt.savefig(f'f_plotn_{n}.png')

for n in [10, 100, 1000, 10000]:
    calc(n)
```