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

Лабораторная работа №2

по дисциплине «Статистическое моделирование»

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

Руководитель Чуркин В.В.

Санкт-Петербург
2019 г.

Содержание

Цель работы

1. Практическое освоение методов получения случайных величин, имеющих дискретный характер распределения.
2. Разработка программных датчиков дискретных случайных величин.
3. Исследование характеристик моделируемых датчиков: Оценка точности моделирования: вычисление математического ожидания и дисперсии, сравнение полученных оценок с соответсвующими теоретическими значениями.
4. Графическое представление функции плотности распределения и интегральной функции распределения.

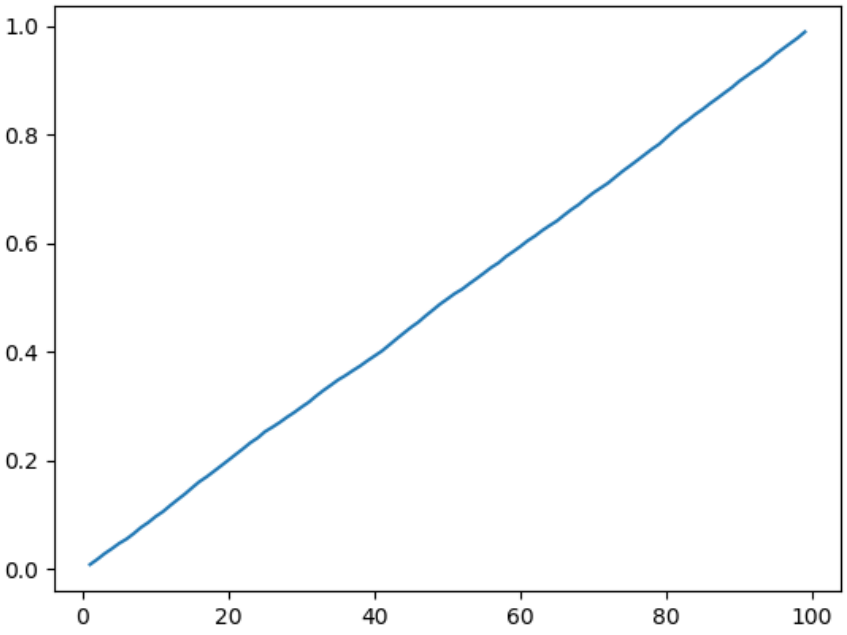
Проведение работы

Была написана программа на языке python, содержащая набор функций для генераций случайных величин, имеющих дискретный характер распределения в соответствие с разных алгоритмов моделиравния. Так же были написани функции которые получают последовательности заданного распределения используя генераторов случайных величин, далее вычисляется математическое ожидание и дисперсия полученной последовательности, а так же и соответствующие теоретические значения. Строится таблица результатов и графики функции плотности распределения и интегральной функции распределения с использованием библиотеки matplotlib.

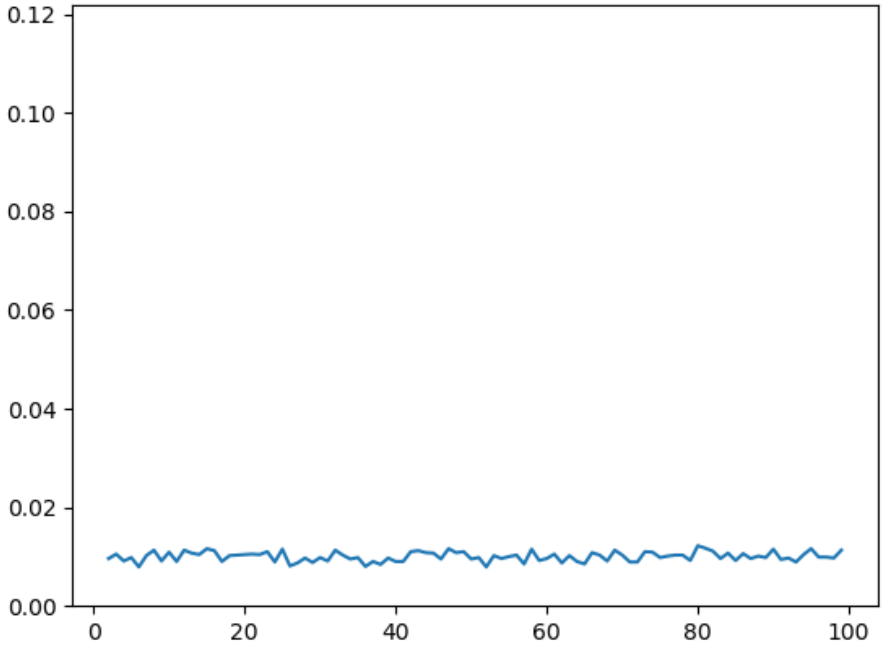
Результаты

1. Равномерное распределение

Оценка	IRNUNI	Погрешность	Теоретическое значение
@M@	50,607	0,107	50.5
@D@	836,077	2,827	833.25



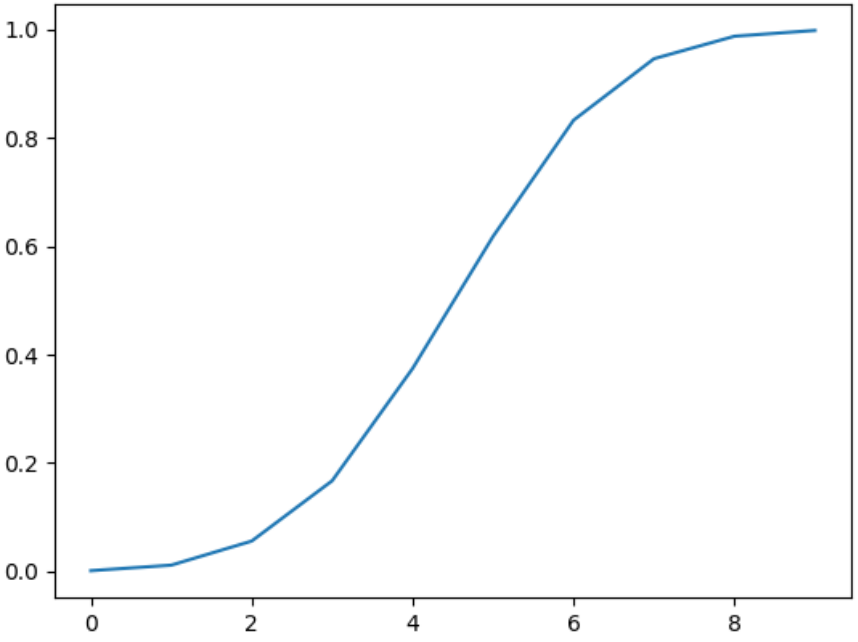
Интегральная фунцкия распределения



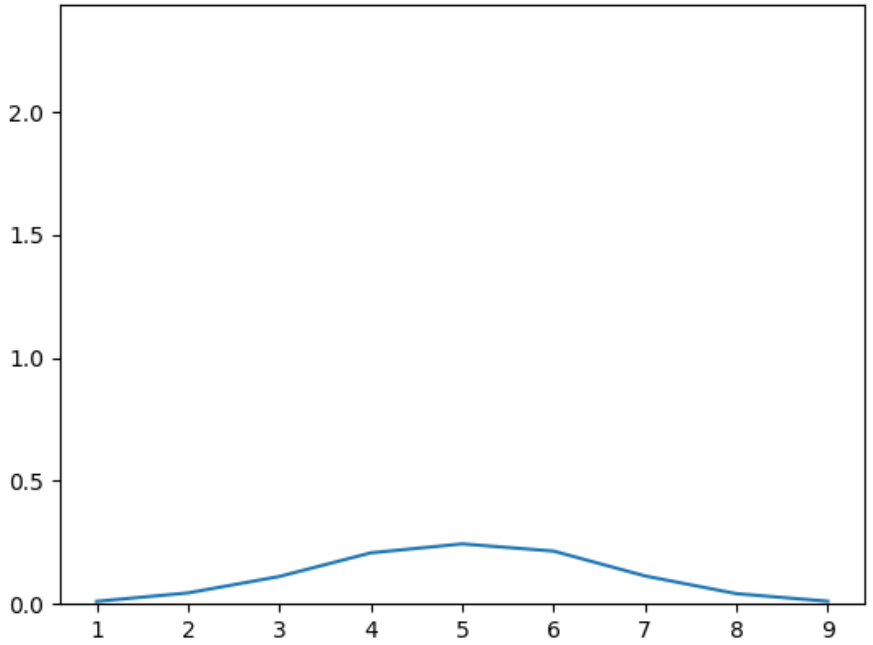
Функция плотности вероятности

2. Биномиальное распределение

Оценка	IRNBIN	IRNBNL	Теоретическое значение
@M@	4,9854	4,9851	5.0
@D@	2,5049	2,5570	2.5



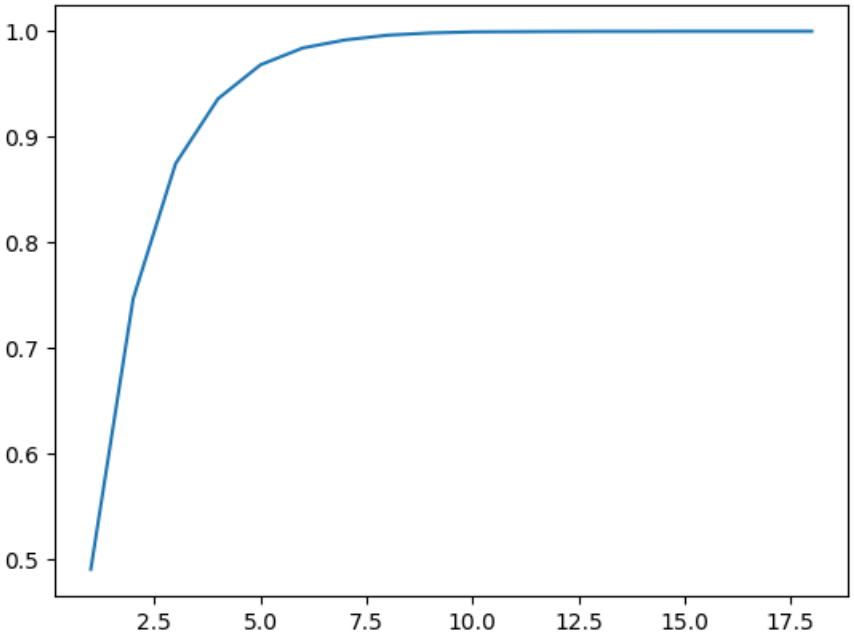
Интегральная фунцкия распределения



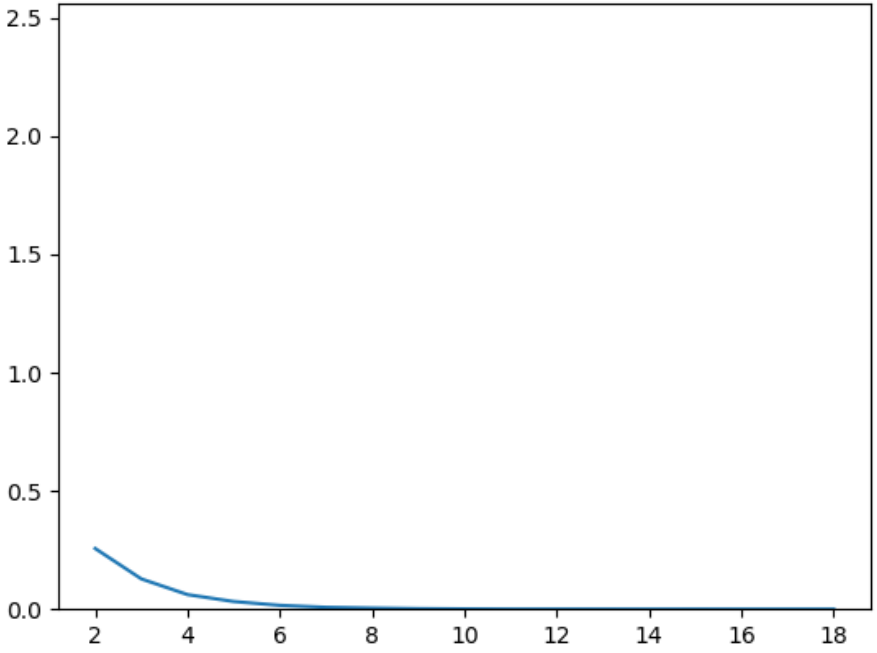
Функция плотности вероятности

3. Геометрическое распределение

Момент	IRNGEO_1	IRNGEO_2	IRNGEO_3	Теоретическое значение
@M@	1,998	2,003	1,990	2.0
@D@	1,952	2,039	1,998	2.0



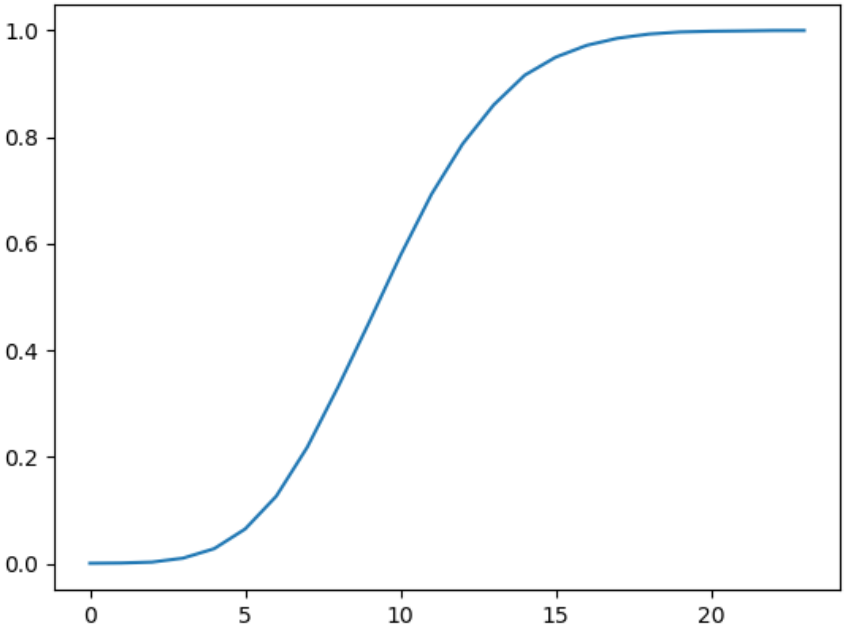
Интегральная фунцкия распределения



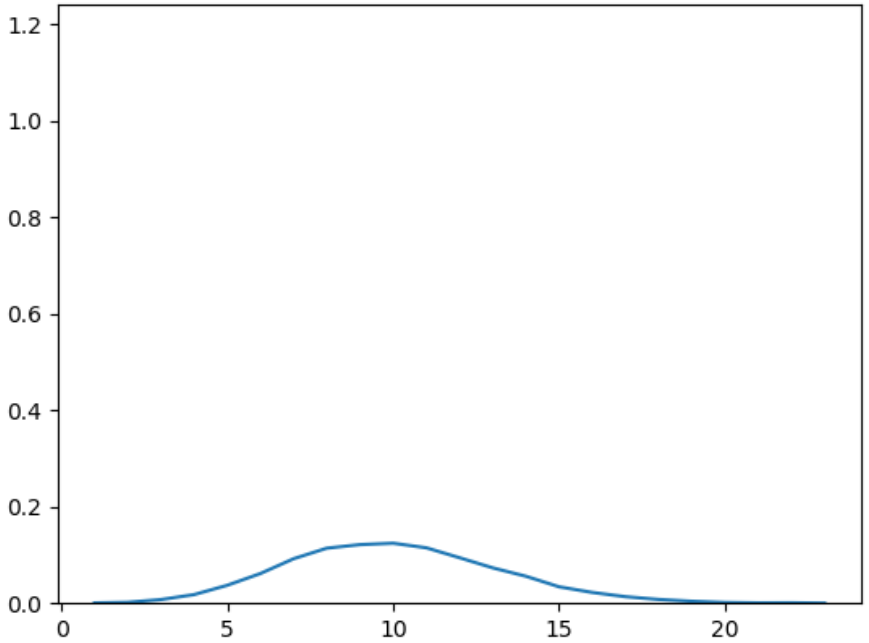
Функция плотности вероятности

4. Распределение Пуассона

Момент	IRNPOI	IRNPSN	Теоретическое значение
@M@	9,978	10,009	10.0
@D@	10,269	10,030	10.0



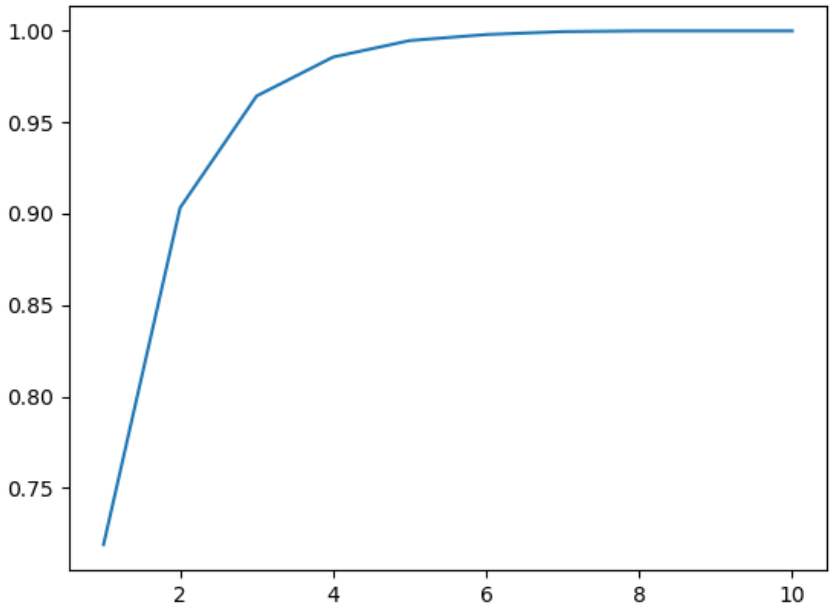
Интегральная фунцкия распределения



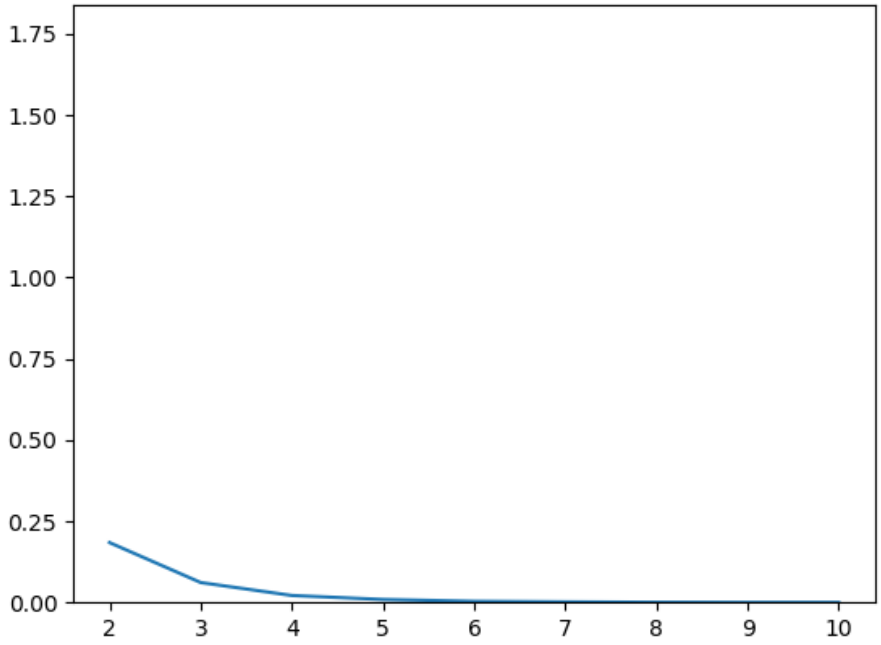
Функция плотности вероятности

5. Логарифмическое распределение

Момент	IRNLOG	Отклон	Теоретическое значение
@M@	1,422	0,020	1.44270
@D@	0,771	0,032	0.80402



Интегральная фунцкия распределения



Функция плотности вероятности

Вывод

Были освоени практические методы получения случайных величин, имеющих дискретных характер распределения. Были разработани программные датчики дискретных случайных величин в соответствие с разных алгоритмов моделирования и были исследованы их характеристик – математическое ожидание и дисперсия, а так же и сравнение полученных оценок с соответствующими теоретическими значениями. С использованием библиотеки matplotlib были построены графические представления фунцкии плотности распреелений и интегральной фунцкии распределений.

Текст программы

```
import random
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import math

def plot_raspr(u, n):
    u.sort()
    intervals = np.array(range(int(np.min(u)), int(np.max(u)), 1))
    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.plot(intervals, cumulativeFreq)

    plt.figure()
    plt.ylim(0, max(frequencies) * 10)
    plt.plot(intervals[1:], frequencies)
    plt.show()
def calc_m_d(list):
    m = np.sum(list) / len(list)
    d = np.sum([(val - m)**2 for val in list]) / len(list)
    return m, d

def IRNUNI(ILOW, IUP):
    return int((IUP - ILOW + 1)*random.random() + ILOW)

def RNNORM(N, p):
    r = random.random()
    temp_p = (1-p)**N
    sum = temp_p
    for i in range(N):
        if(sum > r):
            return i
```

```
        temp_p *= (float(N-i) / float(i+1))*(p / (1-p))
        sum += temp_p
    return N

def IRNGEO_1(p):
    pr = np.random.uniform(0, p)
    otrezok = 0
    while(p > pr):
        otrezok += 1
        pr /= p
    return otrezok
def IRNGEO_2(p):
    r = np.random.random()
    count = 1
    while(r > p):
        count+=1
        r = np.random.random()
    return count
def IRNGEO_3(p):
    u = random.random()
    k = int(math.log(u) / math.log(1-p)) + 1
    return k
def IRNPSN(mu):
    r = random.random()
    k = 0
    while(r >= math.exp(-mu)):
        r *= random.random()
        k += 1
    return k
def IRNUNI_TEST(ILOW, IUP, N):
    r = [IRNUNI(ILOW, IUP) for _ in range(N)]
    m, d = calc_m_d(r)
    t_m = (ILOW + IUP) / 2
    t_d = ((IUP - ILOW + 1) ** 2 - 1) / 12

    print(pd.DataFrame(data={'Оценка': ['@M@', '@D@'], 'IRNUNI': [m, d],
                             'Погрешность': [abs(m - t_m), abs(d - t_d)],
                             'Теоретическое значение': [t_m, t_d]}))
    plot_raspr(r, N)

def RNNORM_TEST(N, p, size):
    r1 = [RNNORM(N, p) for _ in range(size)]
    #r1 = [int(np.random.binomial(N, p)) for _ in range(size)]
    r2 = [int(np.random.normal(N * p, np.sqrt(N * p * (1 - p))) + 0.5) for _ in
range(size)]
    m1, m2 = np.sum(r1) / size, np.sum(r2) / size
    d1, d2 = np.sum([(x - m1)**2 for x in r1]) / size, np.sum([(x-m2)**2 for x in r2]) /
size
    t_m = N*p
    t_d = N*p*(1-p)

    print(pd.DataFrame(data = {'Оценка':['@M@', '@D@'],
        'IRNBIN':[m1, d1], 'IRNBNL': [m2, d2],
        'Теоретическое значение':[t_m, t_d]}))
    plot_raspr(r1, size)
    plot_raspr(r2, size)

def IRNGEO_TEST(p, size):
```

```
    r1 = [IRNGEO_1(p) for _ in range(size)]
    r2 = [IRNGEO_2(p) for _ in range(size)]
    r3 = [IRNGEO_3(p) for _ in range(size)]
    m1, d1 = calc_m_d(r1)
    m2, d2 = calc_m_d(r2)
    m3, d3 = calc_m_d(r3)
    t_m = 1/p
    t_d = (1-p)/(p**2)
    print(pd.DataFrame(data = {'Оценка':['@M@', '@D@'],
        'IRNGEO_1':[m1, d1], 'IRNGEO_2':[m2, d2], 'IRNGEO_3': [m3, d3],
        'Теоретическое значение':[t_m, t_d]}))
    plot_raspr(r1, size)
    plot_raspr(r2, size)
    plot_raspr(r3, size)

def IRNPSN_TEST(mu, size):
    r1 = np.random.poisson(mu, size)
    r2 = [IRNPSN(mu) for _ in range(size)]
    m1, d1 = calc_m_d(r1)
    m2, d2 = calc_m_d(r2)
    t_m = mu
    t_d = mu
    print(pd.DataFrame(data = {'Оценка':['@M@', '@D@'],
        'IRNPOI':[m1, d1], 'IRNPSN':[m2, d2],
        'Теоретическое значение':[t_m, t_d]}))
    plot_raspr(r1, size)
    plot_raspr(r2, size)

def IRNLOG_TEST(q, size):
    p = 1 - q
    alpha = 1 / math.log(p)
    r = np.random.logseries(p, size)
    m, d = calc_m_d(r)
    t_m = -alpha*q/p
    t_d = -alpha*q*(1+alpha*q)/p**2

    print(pd.DataFrame(data = {'Оценка':['@M@', '@D@'],
        'IRNLOG':[m, d], 'Отклон':[abs(m - t_m), abs(d - t_d)],
        'Теоретическое значение':[t_m, t_d]}))
    plot_raspr(r, size)

while True:
    inp = input("\nВыберите тип распределения: \n1 - равномерное\n2 - биномиальное\n"
                "3 - геометрическое\n4 - пуассона\n5 - логарифмическое\n > ")
    if inp == '1':
        IRNUNI_TEST(1, 100, 10**4)
    elif inp == '2':
        RNNORM_TEST(10, 0.5, 10**4)
    elif inp == '3':
        IRNGEO_TEST(0.5, 10**4)
    elif inp == '4':
        IRNPSN_TEST(10, 10**4)
    elif inp == '5':
        IRNLOG_TEST(0.5, 10**4)

    if input("продолжить? [1 - да, 0 - нет] \n > ") == '0':
        break
```