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

Лабораторная работа №3

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 непрерывный характер распределения.
2. Разработка программных датчиков дискретных случайных величин.
3. 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.

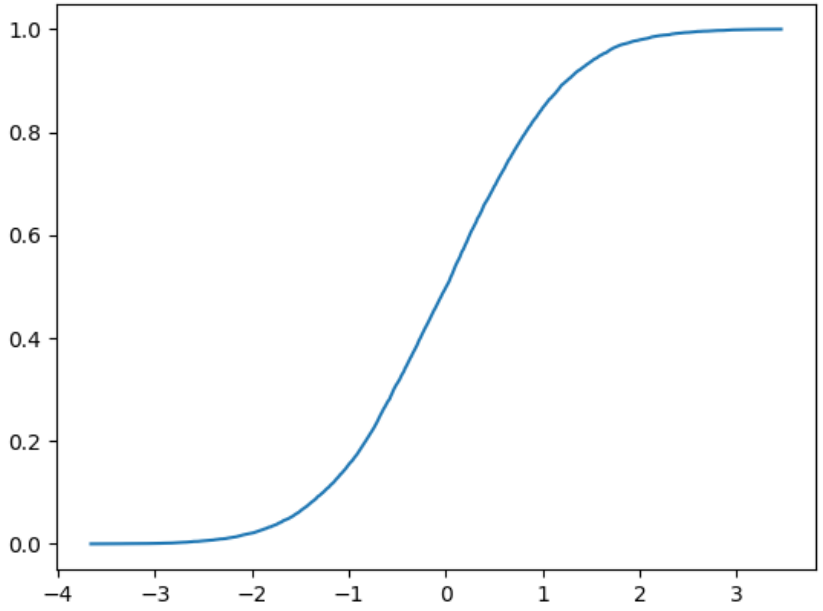
Проведение работы

Была написана программа на языке python, содержащая набор функций для генераций случайных величин, имеющих непрерывный 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.

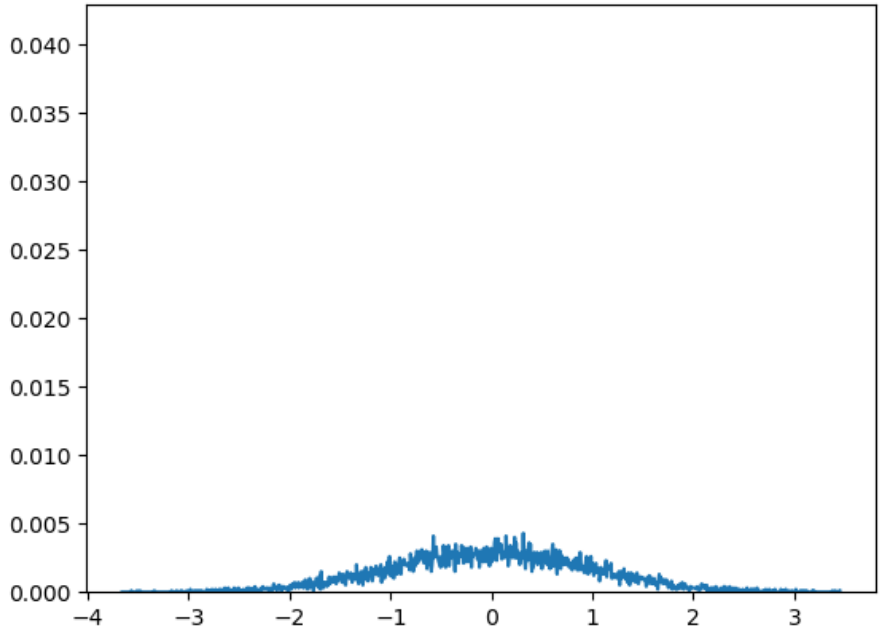
Результаты

1. Нормальное распределение

Момент	RNNRM1	RNNRM2	Теоретическое значение
@M@	-0,0064	0,0037	0.0
@D@	0,9670	0,9952	1.0



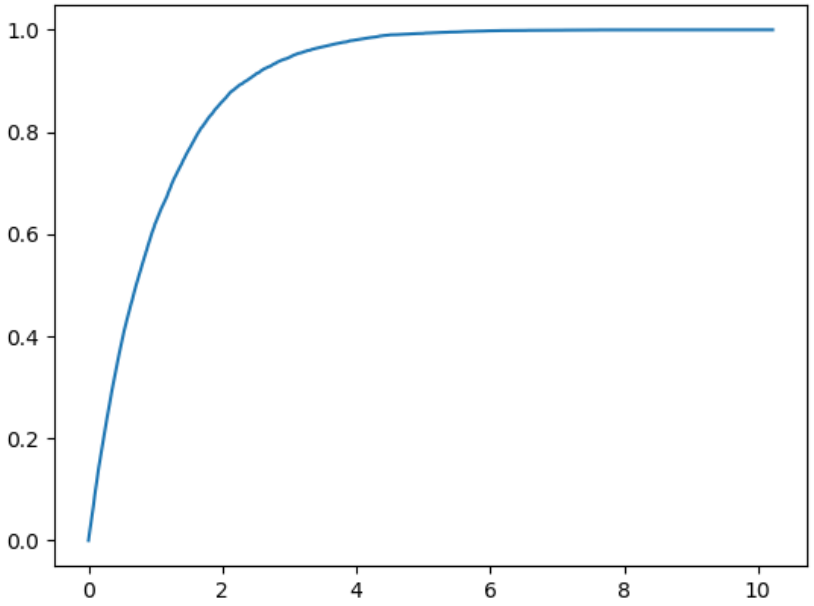
Интегральная фунцкия распределения



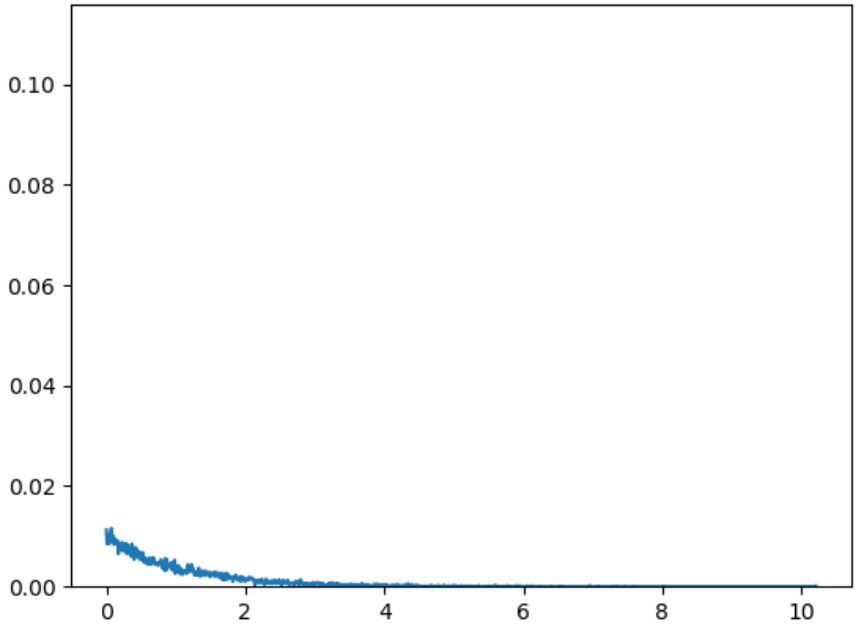
Функция плотности вероятности

2. Экспоненциальное распределение

Момент	RNEXP	Теоретическое значение
E(x)	1,021	1
V(x)	1,022	1



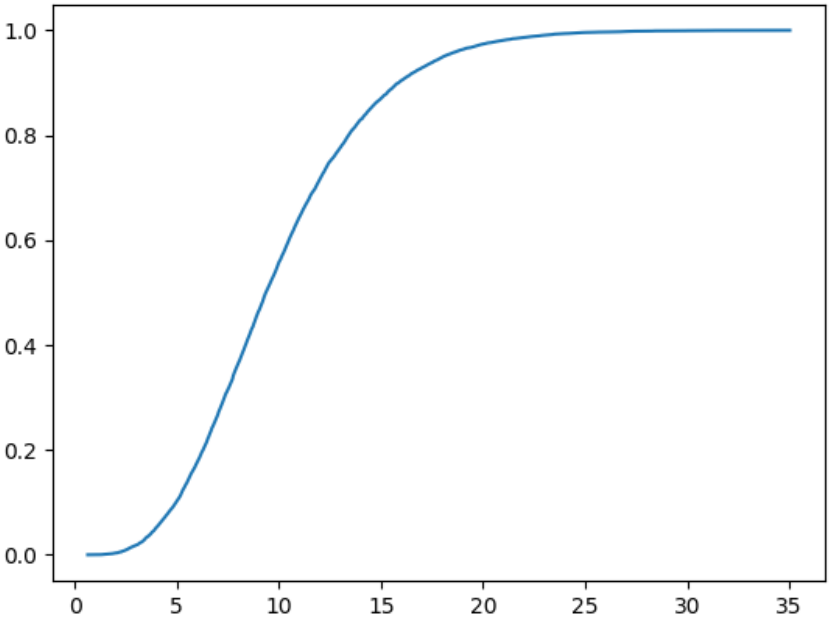
Интегральная фунцкия распределения



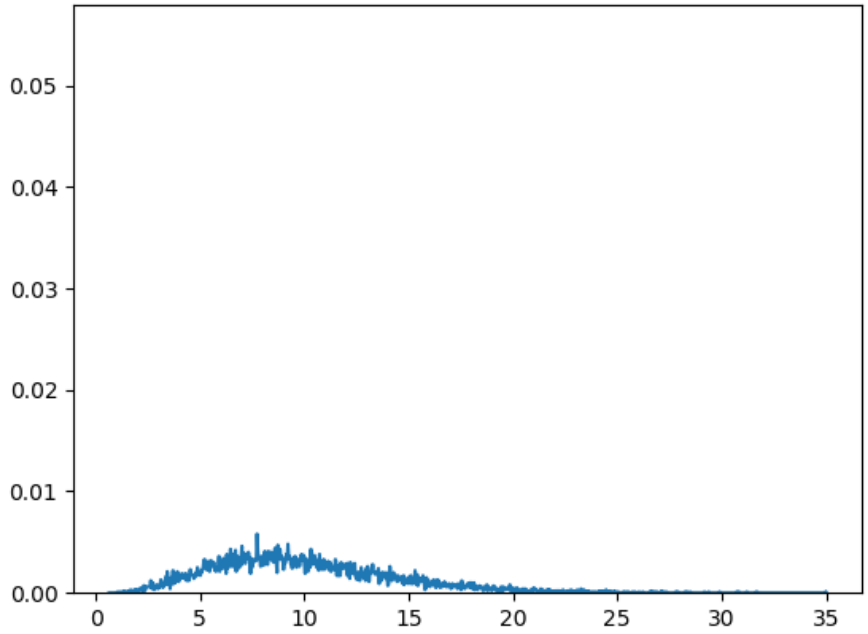
Функция плотности вероятности

3. Хи-квадрат распределение

Момент	RNCHIS	Теоретическое значение
E(x)	9,999	10
V(X)	19,171	20



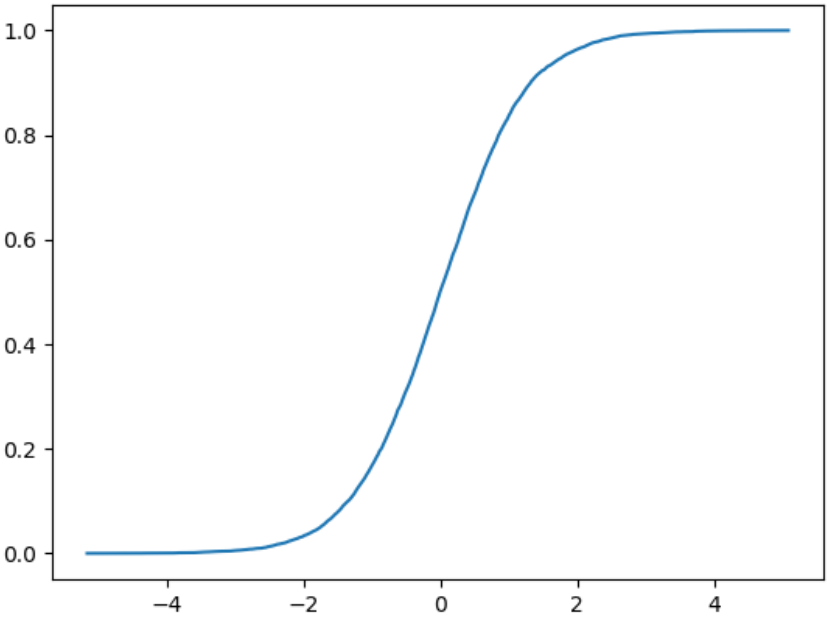
Интегральная фунцкия распределения



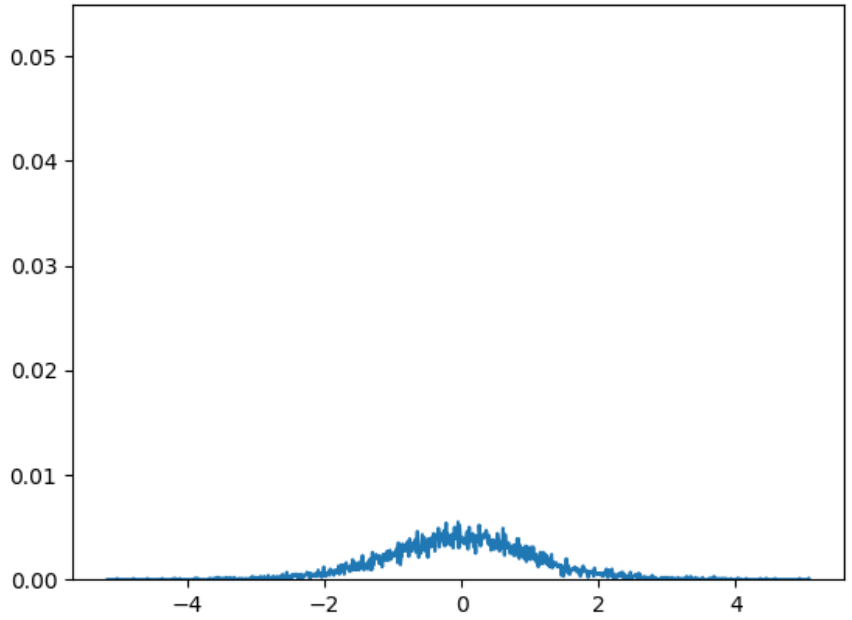
Функция плотности вероятности

4. Распределение Стюдента

Момент	RNSTUD	Погрешн	Теоретическое значение
E(t)	-0,0024	0,0024	0.0
V(t)	1,1874	0,0625	1.25



Интегральная фунцкия распределения



Функция плотности вероятности

Вывод

Были освоени практические методы получения случайных величин, имеющих непрерывный характер распределения. Были разработани программные датчики непрерывных 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

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

def RNUNI(a, b):
    """ равномерное распределние на [a, b] """
    return (b - a) * random.random() + a

def RNNRM1(m, sigma):
    """ нормальное распределение, алгоритм Бокса-Миллера """
    u1, u2 = RNUNI(0, 1), RNUNI(0, 1)
    z1 = math.sqrt(-2.0 * math.log(u1)) * math.cos(2 * math.pi * u2)
    z2 = math.sqrt(-2.0 * math.log(u1)) * math.sin(2 * math.pi * u2)
    y1, y2 = m + sigma * z1, m + sigma * z2
    return y1, y2

def RNNRM2(m, sigma):
    """ нормальное распределение, центральная предельная теорема """
    z = sum([random.random() for _ in range(12)]) - 6.0
    y = m + sigma * z
    return y

def RNEXP(beta):
    """ экспоненциальное распределение """
    return -beta * math.log(RNUNI(0, 1))

def RNCHIS(N):
    """ хи-квадрат распределение """
    return sum([RNNRM2(0, 1)**2 for _ in range(N)])

def RNSTUD(N):
    """ распределение стюдента """
    return RNNRM2(0, 1) / math.sqrt(RNCHIS(N) / float(N))
```

```
def plot_raspr(u, n):
    u.sort()
    intervals = np.linspace(np.min(u), np.max(u), int(n/10))
    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)

def calc_m_d(list):
    m = np.sum(list) / len(list)
    d = np.sum([(x - m)**2 for x in list]) / len(list)
    return m, d

def test_norm(m, sigma, n):
    r1 = []
    for _ in range(int(n/2)):
        r1.extend(RNNRM1(m, sigma))
    r2 = [RNNRM2(m, sigma) for _ in range(n)]
    tm, td = m, sigma ** 2
    r1m, r1d = calc_m_d(r1)
    r2m, r2d = calc_m_d(r2)

    print(pd.DataFrame({'Момент':['E(x)','V(x)'], 'RNNRM1':[r1m, r1d],
                        'RNNRM2':[r2m, r2d], 'Теоретич.':[tm, td]}))
    plot_raspr(r1, n)
    plot_raspr(r2, n)

def test_exp(beta, n):
    r = [RNEXP(beta) for _ in range(n)]
    tm, td = beta, beta**2
    rm, rd = calc_m_d(r)
    print(pd.DataFrame({'Момент':['E(x)','V(x)'], 'RNEXP':[rm, rd],
                        'Теоретич.':[tm, td]}))
    plot_raspr(r, n)

def test_chisq(N, n):
    r = [RNCHIS(N) for _ in range(n)]
    tm, td = N, 2*N
    rm, rd = calc_m_d(r)
    print(pd.DataFrame({'Момент':['E(x)','V(x)'], 'RNCHIS':[rm, rd],
                        'Теоретич.':[tm, td]}))
    plot_raspr(r, n)

def test_stud(N, n):
    r = [RNSTUD(N) for _ in range(n)]
    tm, td = 0, N/(N-2)
    rm, rd = calc_m_d(r)
    epsm, epsd = abs(tm-rm), abs(td - rd)
    print(pd.DataFrame({'Момент':['E(x)','V(x)'], 'RNSTUD':[rm, rd],
                        'Погрешн':[epsm, epsd], 'Теоретич.':[tm, td]}))
    plot_raspr(r, n)

test_norm(0, 1, 10**4)
test_exp(1, 10**4)
test_chisq(10, 10**4)
test_stud(10, 10**4)
plt.show()
```