МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
**«Национальный исследовательский  
Нижегородский государственный университет  
им. Н.И. Лобачевского (ННГУ)»**

**Институт информационных технологий, математики и механики**

ОТЧЕТ  
по лабораторной работе

на тему:  
«Моделирование случайных величин и проверка гипотез о  
виде распределения»

**Выполнил:** студент группы 381903-3 Кишкин В. С.

**Проверил:** преподаватель каф. ТВиАД Кудрявцев Е. В.

Нижний Новгород  
2022

Содержание

1. Условие задачи
2. Часть 1
3. Часть 2
4. Часть 3
5. Вывод
6. Приложения

Условие задачи

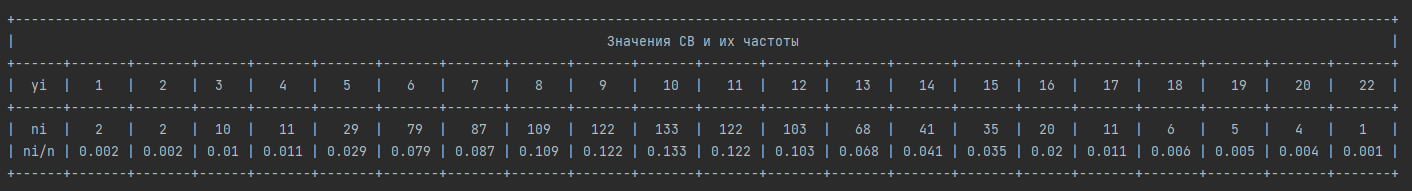
На автоматическую телефонную станцию поступает поток вызовов с интенсивностью λ. Случайная величина η – число вызовов за t минут, имеет распределение Пуассона со средним λt

Часть 1

Из условия задачи можно понять, что с.в. *η* – дискретная.

Командой np.random.uniform(0, 1, N) создаем массив размером N со значениями случайной величины с равномерным распределением на [0, 1]. После проверяется, является ли полученное значение меньше заданной вероятности p. Если меньше, то значение с.в. увеличивается, иначе – эксперимент завершается.

В итоге получаем словарь со значениями с.в. и сколько раз с.в. приняла эти значения за *n* экспериментов. После все результаты записываются в таблицу по возрастанию. Пример таблицы значений с.в. *η* для :



Часть 2

Во второй части лабораторной работы необходимо было найти теоритические и выборочные числовые характеристики

Исходя из описания задачи можно положить, что для распределения Пуассона мат.ожидание и дисперсия равны . Их выборочные аналоги, выборочное среднее и выборочную дисперсию, высчитываем с помощью библиотеки numpy и ее методов mean() и var(). Выборочную медиану так же нахожу с помощью библиотеки numpy и метода median(). Размах выборки нахожу <Последний элемент выборки> - <Первый элемент выборки>.

Таблица характеристик с.в. при :

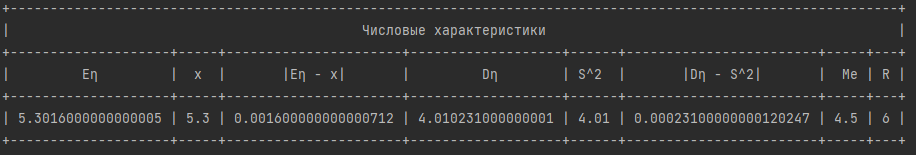
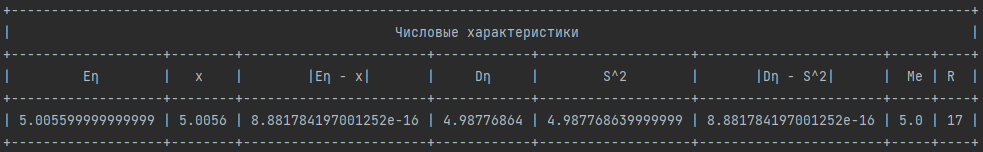


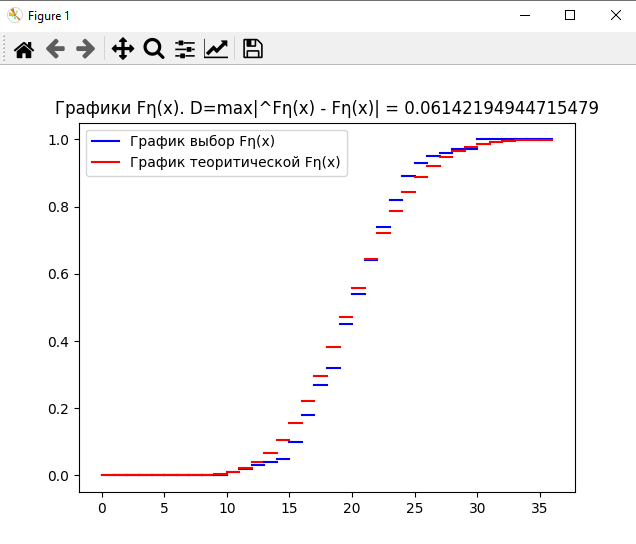
Таблица характеристик с.в. при 00:



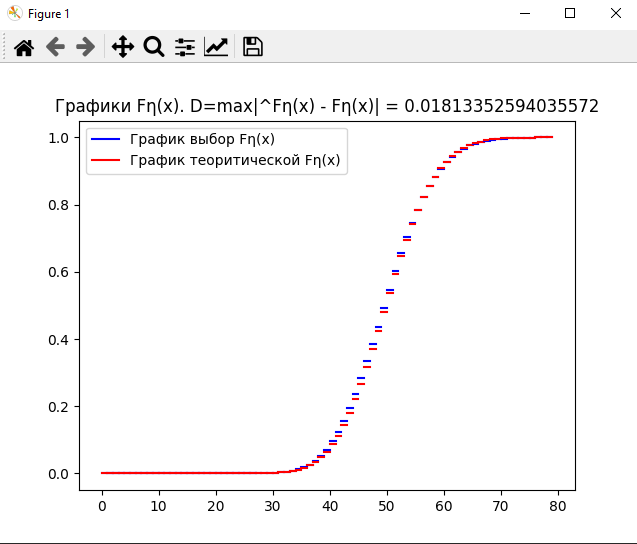
Видно, что при увеличение *n* уменьшается разность мат.ожидания и выборочного среднего, дисперсии и выборочной дисперсии

Далее строим графики теоритической и выборочной функций распределения и вычисляем меру их расхождения *D*

Графики для :



Графики для :



Видно, что мера расхождения *D* уменьшается при увеличение *n*

Так как с.в. является дискретной, то вычисляем теоритические вероятности и отклонения . После находим максимальное отклонение

Для :

Для :

Так же видно, что при увеличение *n* уменьшается максимальное отклонение частоты и теоритической вероятности

Часть 3

В третьей части лабораторной работы нужно было проверить гипотезу о виде распределения. На основаниее условия задачи была выдвинута гипотиза – геометрическое распределение

Под статистическим критерием для проверки нулевой гипотизы понимается правило, которое каждому набору выборочных значений приписывает решение «принять нулевую гипотезу» или «отклонить нулевую гипотезу»

В работе я использовал критерий : Разобьем числовую ось на интервалы . Положим

Интервалы нуно выбирать так, чтобы каждый содержал хотя бы одну точку, а вероятности были примерно одинаковые на каждом интервале, где . Так как с.в. дискретна, то вычисляется так

Пусть – число наблюдений, попавших в . В качестве статистики критерия выберем величину

При справедливости нулевой гипотезы величина имеет распределение с *k -1* степенями свободы. Не согласующиемися с нулевой гипотезой являются большие значения . Пусть – функция распределения с k -1 степенями свободы. Заметим, что функция

- невозрастающая. Значит , то есть если ) > *,* то гипотеза о виде распределения принимается, иначе – отвергается.

Где

Гамма-функция Г(а) вычисляется рекурсивно с помощью соотношения Г(а)=(а-1)Г(а-1) и известных значений Г(1)=1, Г(1/2)

Интеграл посчитан численно, методом трапеций:

Где количество разбиений n взл равным 1000

Интервалы и теоритические веротяности при :



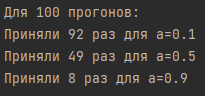
Для:

a = 0.1

F\_(R0) = 0.876246805346585

Принимаем гипотезу

Далее проводились серии экспериментов при казличных значениях



Из полученных результатов можно сделать вывод, что выдвинутая мною гипотеза о распределении случайной величины верна

Выводы

В ходе лабораторной работы было реализовано моделирование дискретной случайной величины для поставленной задачи. Были вычислены теоретические и выборочные числовые характеристики этой случайной величины, построены графики для теоретической и выборочной функции распределения и вычислена их мера расхождения. Была проверена гипотеза о виде распределения

Приложение

*""" Задача:  
На автоматическую телефонную станцию поступает поток вызовов с  
интенсивностью λ. С.в. η — число вызовов за t минут, имеет распределение Пуассона со средним λt  
---------------------------------------------------------------------------------------------------------  
Распределение Пуассона:  
P(η(t)=η) = λ^η/η! \* exp(-λ)  
"""*from math import exp, factorial, fabs, inf, sqrt, pi  
import matplotlib.pyplot as plt  
import numpy as np  
from scipy import integrate, stats  
  
import os  
from prettytable import PrettyTable  
  
  
# Чистим экран (консоль) для разных ОС  
def \_cls() -> None:  
 os.system('cls' if os.name == 'nt' else 'clear')  
  
  
def print\_title(title: str) -> None:  
 print('\n')  
 print('=====================================================================================================' \  
 '=================================================')  
 print(' ' \* (150 // 2 - len(title) // 2), title, sep='')  
 print('=====================================================================================================' \  
 '=================================================')  
 print('\n')  
  
  
def count\_poisson(liambda: float, n: int) -> float:  
 return liambda \*\* n / factorial(n) \* exp(-liambda)  
  
  
def stage1(N: int, liambda: float, print\_info=False) -> tuple:  
 *"""* ***:param*** *N: количество экспериментов* ***:param*** *liambda: интенсивность* ***:param*** *print\_info: выводить информацию или нет* ***:return****: массив св  
 """* print\_title('1 ЭТАП')  
  
 """ --------------------------------------------------Подсчет-------------------------------------------------- """  
  
 arr = []  
 U = np.random.uniform(0, 1, N)  
 for i in range(N):  
 u = U[i]  
  
 p = exp(-liambda)  
 p\_tmp = p  
 index = 0  
 while u >= p:  
 index += 1  
 p\_tmp = p\_tmp \* liambda / index  
 p += p\_tmp  
 del p\_tmp  
  
 n = index  
  
 if print\_info:  
 print('-------------------------------------')  
 print(f'Эксперимент №{i + 1}:')  
  
 content = f"""  
 Интенсивность: λ={liambda}  
 Значение случайной величины U: u={u}  
 Число вызовов за t минут: η={n}  
 """  
 print(content)  
  
 arr.append(n)  
  
 dict\_for\_table = {}  
 for elem in arr:  
 try:  
 dict\_for\_table[elem] = [elem, dict\_for\_table[elem][1] + 1, (dict\_for\_table[elem][1] + 1) / N]  
 except KeyError:  
 dict\_for\_table[elem] = [elem, 1, 1 / N]  
  
 """ --------------------------------------------------Таблица-------------------------------------------------- """  
  
 yi = ['yi']  
 ni = ['ni']  
 ni\_n = ['ni/n']  
 tmp = sorted(dict\_for\_table)  
 yi.extend(tmp)  
 ni.extend([dict\_for\_table[elem][1] for elem in tmp])  
 ni\_n.extend([dict\_for\_table[elem][2] for elem in tmp])  
 del tmp  
  
 table = PrettyTable()  
 table.title = 'Значения СВ и их частоты'  
 table.field\_names = yi  
 table.add\_rows([  
 ni,  
 ni\_n  
 ])  
 print(table)  
  
 return arr, dict\_for\_table  
  
  
def stage2(N: int, data: list, dict\_for\_table: dict, liambda: float):  
 *"""* ***:param*** *N: количество экспериментов* ***:param*** *data: массив св* ***:param*** *dict\_for\_table: словарь из 1 этапа* ***:param*** *liambda: интенсивность* ***:return****:  
 """* print\_title('2 ЭТАП')  
  
 """ --------------------------------------------------Подсчет-------------------------------------------------- """  
  
 en = 0 # мат. ожидание  
 x = 0 # выборочное среднее  
 for yi in dict\_for\_table.keys():  
 ni = dict\_for\_table[yi][1]  
 ni\_n = dict\_for\_table[yi][2]  
 en += yi \* ni\_n  
 x += yi \* ni  
 x /= N  
 en\_x = fabs(en - x) # разница мат. ожидания и выборочного среднего  
  
 dn = 0 # дисперсия  
 s2 = 0 # выборочная дисперсия  
 for yi in dict\_for\_table.keys():  
 ni = dict\_for\_table[yi][1]  
 ni\_n = dict\_for\_table[yi][2]  
 dn += (yi - en) \*\* 2 \* ni\_n  
 s2 += (yi - x) \*\* 2 \* ni  
 s2 /= N  
 dn\_s2 = fabs(dn - s2) # разница дисперсии и выборочной дисперсии  
  
 # выборочная медиана  
 if N in [1, 2]:  
 if N == 1:  
 me = data[0]  
 else:  
 me = data[1]  
 else:  
 if N % 2 == 1:  
 me = data[N // 2 + 1]  
 else:  
 me = (data[N // 2] + data[N // 2 + 1]) / 2  
 r = data[-1] - data[0] # размах выборки  
  
 """ --------------------------------------------------График-------------------------------------------------- """  
  
 # Вычисляем точки выборочной функции распределения  
 y\_tmp = []  
 x\_tmp = []  
 y\_tmp\_theory = []  
 sum\_p = 0  
 for x\_i in sorted(dict\_for\_table):  
 sum\_p += dict\_for\_table[x\_i][2]  
 y\_tmp.append(round(sum\_p, 5))  
 x\_tmp.append(x\_i)  
 del x\_i  
  
 # Вычисляем точки теоритической функции распределения  
 sum\_p\_theory = 0  
 for i in range(max(x\_tmp) + 2):  
 sum\_p\_theory += P[i]  
 y\_tmp\_theory.append(sum\_p\_theory)  
 del sum\_p\_theory  
  
 # Ищем D  
 y\_tmp2 = []  
 j = 0  
 for i in range(len(y\_tmp\_theory) - 1):  
 x\_index = 0  
 while i > x\_tmp[x\_index]:  
 x\_index += 1  
 if x\_index + 1 == len(x\_tmp):  
 break  
 if i >= x\_tmp[0]:  
 j += 1  
 if j != 0:  
 y\_tmp2.append(y\_tmp[x\_index])  
 else:  
 y\_tmp2.append(0.0)  
 y\_tmp2.append(1.0)  
  
 tmp = []  
 for i in range(len(y\_tmp2)):  
 fn\_ = y\_tmp2[i]  
 fn = y\_tmp\_theory[i]  
 tmp.append(fabs(fn\_ - fn))  
 D = max(tmp)  
  
 print(f'Выборка: {x\_tmp}')  
 print(f'Значения по оси Y для выборочной ф. распределения: {y\_tmp2}')  
 print(f'Значения по оси Y для теоритической ф. распределения: {y\_tmp\_theory}')  
 print(f'Массив разности функций распределений: {tmp}')  
  
 length = len(y\_tmp2)  
 for i in range(length):  
 if i == 0:  
 plt.plot([0, x\_tmp[0]], [0, 0], 'b-', label='График выбор Fη(x)')  
 plt.plot([0, 1], [0, 0], 'r-', label='График теоритической Fη(x)')  
 else:  
 plt.plot([i, i + 1], [y\_tmp2[i], y\_tmp2[i]], 'b-')  
 plt.plot([i, i + 1], [y\_tmp\_theory[i], y\_tmp\_theory[i]], 'r-')  
 # plt.title(f'Графики Fη(x). D=max|^Fη(x) - Fη(x)| = {D} для x={tmp.index(D)}')  
 plt.title(f'Графики Fη(x). D=max|^Fη(x) - Fη(x)| = {D}')  
 plt.legend()  
 plt.show()  
  
 """ --------------------------------------------------Таблицы-------------------------------------------------- """  
  
 table = PrettyTable()  
 table.title = 'Числовые характеристики'  
 table.field\_names = ['Чем посчитано', 'Eη', 'x', '|Eη − x|', 'Dη', 'S^2', '|Dη − S^2|', 'Me', 'R']  
 table.add\_row(['Мной', en, x, en\_x, dn, s2, dn\_s2, me, r])  
 table.add\_row(['Numpy и стандартными способами\*', liambda, np.mean(data), fabs(liambda - np.mean(data)), liambda, np.var(data),  
 fabs(liambda-np.var(data)), np.median(data), data[-1] - data[0]])  
 print(table)  
 print('\*Стандартными способами - для пуассоновского распределения мат. ожидание и дисперсия равны лямбде')  
  
 yj = ['yj']  
 pj = ['P({η=yj})']  
 nj\_n = ['nj/n']  
  
 tmp = sorted(dict\_for\_table)  
 yj.extend(tmp)  
 pj.extend([P[elem] for elem in tmp])  
 nj\_n.extend([dict\_for\_table[elem][2] for elem in tmp])  
  
 table = PrettyTable()  
 table.title = 'Теоритические вероятности'  
 table.field\_names = yj  
 table.add\_rows([  
 pj,  
 nj\_n  
 ])  
 print(table)  
 print('max|nj/n - P({η=yj})| =', max([fabs(nj\_tmp - pj\_tmp) for nj\_tmp, pj\_tmp in zip(nj\_n[1:], pj[1:])]))  
  
  
def stage3(n: int, data: dict, a, k, auto, print\_info=False) -> int:  
 *"""* ***:param*** *n: кол-во экспериментов* ***:param*** *data: выборка и инфо о ней [число выборки, кол-во выпадения, вероятность выпадения]* ***:param*** *print\_info: выводить информацию или нет* ***:return****: None  
 """* print\_title('3 ЭТАП')  
  
 # Ввод границ z  
 zs = [0]  
 if auto == '-':  
 for i in range(k - 1):  
 while True:  
 try:  
 z = float(input(f'Введите границу: z\_{i + 1} = '))  
 if z <= zs[i]:  
 \_cls()  
 print(f'Граница z\_{i + 1} должна быть больше предыдущей ({zs[i]}). Попробуйте снова')  
 continue  
 if z == -inf or z == inf:  
 \_cls()  
 print(f'Граница z\_{i + 1} должна быть в интервале (-inf, inf). Попробуйте снова')  
 continue  
 zs.append(z)  
 break  
 except ValueError:  
 \_cls()  
 print('Вы ввели не число. Попробуйте снова')  
 else:  
 # Вычисляем точки выборочной функции распределения  
 x\_tmp = []  
 y\_tmp\_theory = []  
 sum\_p = 0  
 for x\_i in sorted(dict\_for\_table):  
 sum\_p += dict\_for\_table[x\_i][2]  
 x\_tmp.append(x\_i)  
 del x\_i  
  
 # Вычисляем точки теоритической функции распределения  
 sum\_p\_theory = 0  
 for i in range(max(x\_tmp) + 50):  
 sum\_p\_theory += P[i]  
 y\_tmp\_theory.append(sum\_p\_theory)  
 del sum\_p\_theory  
  
 tmp = 1 / k  
 intervals = [round(elem \* tmp, 6) for elem in range(1, k + 1)]  
 del tmp  
 j = 0  
 for i, elem in enumerate(y\_tmp\_theory):  
 try:  
 if elem >= intervals[j]:  
 j += 1  
 zs.append(i)  
 except IndexError:  
 break  
 zs.append(inf)  
 print('Границы:', zs)  
  
 """ --------------------------------------------------Таблица-------------------------------------------------- """  
  
 qs = []  
 for j in range(1, len(zs)):  
 q = 0  
 for elem in sorted(P):  
 if zs[j - 1] <= elem < zs[j]:  
 q += P[elem]  
 qs.append(q)  
 if print\_info:  
 content = """  
 Информация (Вероятности С.В. η):\n"""  
 for elem in sorted(P):  
 content += f' {elem}: {P[elem]}\n'  
 print(content)  
  
 intervals = ['Интервалы']  
 intervals.extend([f'[{zs[j - 1]}, {zs[j]})' for j in range(1, len(zs))])  
 qk = ['qk']  
 qk.extend(qs)  
 table = PrettyTable()  
 table.title = 'Отображение гипотезы в виде теоретических вероятностей'  
 table.field\_names = intervals  
 table.add\_row(qk)  
 print(table)  
 del intervals  
  
 """ -------------------------------------------------Гипотезы------------------------------------------------- """  
  
 from pprint import pprint  
 print('need')  
 pprint(qs)  
 pprint(zs)  
 pprint(data)  
 # Находим статистику критерия  
 R0 = 0  
 for i in range(k):  
 print(i)  
 nj = 0  
 qj = qs[i]  
 if zs[i + 1] != inf:  
 tmp = zs[i + 1]  
 else:  
 tmp = max(sorted(dict\_for\_table))  
 for j in range(zs[i], tmp):  
 try:  
 nj += data[j][1]  
 # qj += data[j][2]  
 except KeyError:  
 pass  
 tmp = n \* qj  
 try:  
 R0 += (nj - tmp) \*\* 2 / tmp  
 except ZeroDivisionError:  
 pass  
 del tmp  
  
 def G(x) -> float:  
 if x == 1:  
 return 1.  
 if x == .5:  
 return sqrt(pi)  
  
 return (x - 1) \* G(x - 1)  
  
 def fx(x):  
 return 2 \*\* (- k / 2) \* x \*\* (k / 2 - 1) \* exp(-x / 2) \* G(k / 2) \*\* - 1  
  
 F\_, err = integrate.quad(fx, 0, R0)  
 F\_ = 1 - F\_  
 print('NEED-')  
 print(F\_)  
  
 F\_ = stats.chi2.sf(R0, df=k-1)  
 print(F\_)  
  
 print(f'Для:\n'  
 f'a = {a}\n'  
 f'F\_(R0) = {F\_}')  
 if F\_ > a:  
 print('Принимаем гипотезу')  
 return 1  
 else:  
 print('Не принимаем гипотезу')  
 return 0  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 np.random.seed()  
  
 # GLOBAL variables  
 P = {}  
 N = 100000  
 liambda = 1  
  
 while True:  
 try:  
 N = int(input('Введите количество экспериментов: N = '))  
 break  
 except ValueError:  
 \_cls()  
 print('Вы ввели не число. Попробуйте снова')  
 while True:  
 try:  
 liambda = float(input('Введите интенсивность: λ = '))  
 if liambda <= 0:  
 \_cls()  
 print('λ должна быть положительной. Попробуйте снова')  
 continue  
 break  
 except ValueError:  
 \_cls()  
 print('Вы ввели не число. Попробуйте снова')  
  
 # # Ввод a  
 # while True:  
 # try:  
 # a = float(input('Введите уровень значимости: a = '))  
 # if not (0 <= a < 1):  
 # \_cls()  
 # print('a должна быть 0 <= a < 1. Попробуйте снова')  
 # continue  
 # break  
 # except ValueError:  
 # \_cls()  
 # print('Вы ввели не число. Попробуйте снова')  
  
 # Ввод k  
 while True:  
 try:  
 k = int(input('Введите количество интервалов: k = '))  
 if k <= 0:  
 \_cls()  
 print('k должна быть положительной. Попробуйте снова')  
 continue  
 break  
 except ValueError:  
 \_cls()  
 print('Вы ввели не число. Попробуйте снова')  
 auto = input('Автоматический ввод интервалов (+ да, - нет): ')  
  
 results = []  
 arr\_a = [.1, .5, .9]  
 result = [0, 0, 0]  
 count\_for = 100  
 for index in range(count\_for):  
 array, dict\_for\_table = stage1(N, liambda, False)  
  
 array.sort()  
 for elem in range(max(sorted(dict\_for\_table)) + 50):  
 P[elem] = count\_poisson(liambda, elem)  
  
 stage2(N, array, dict\_for\_table, liambda)  
  
 result[0] += stage3(N, dict\_for\_table, arr\_a[0], k, auto, True)  
 result[1] += stage3(N, dict\_for\_table, arr\_a[1], k, auto, True)  
 result[2] += stage3(N, dict\_for\_table, arr\_a[2], k, auto, True)  
 for i, a in enumerate(arr\_a):  
 results.append(f'Приняли {result[i]} раз для a={a}')  
  
 print\_title('РЕЗУЛЬТАТЫ')  
 print(f'Для {count\_for} прогонов:')  
 for elem in results:  
 print(elem)