ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

**ОТЧЕТ**

по дисциплине “Моделирование”

на тему

**Исследование генератора случайных чисел**

|  |  |
| --- | --- |
| Выполнили студенты | Брескун И. М., Селиванов В. В. |
|  | Ф.И.О. |

|  |  |
| --- | --- |
| Группы | ИВ-022 |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работу принял |  | ст. преп. кафедры ПМиК Бублей Д. А. |
|  | подпись |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Защищена |  | Оценка |  |
|  |  |  |  |

Новосибирск – 2024

СОДЕРЖАНИЕ

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc19431)

Ход работы.........................................................................................................4

[1.1 Используемые формулы 4](#_Toc7495)

[1.2 Результат работы программы 5](#_Toc7495)

[ЗАКЛЮЧЕНИЕ 6](#_Toc3919)

[ПРИЛОЖЕНИЕ 7](#_Toc1202)

ПОСТАНОВКА ЗАДАЧИ

Написать программу, которая исследует генератор случайных чисел и провести анализ генеральной совокупности:

1. Используя критерии‌ Пирсона, при уровне значимости 0,01 проверить, согласуется ли гипотеза о равномерном распределении генеральной совокупности.

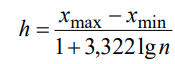
2. Используя коэффициент автокорреляции, проверить качество генеральной совокупности на независимость

Ход работы

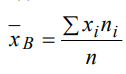
* 1. Используемые формулы

В лабораторной работе использованы следующие формулы:

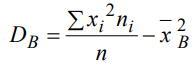
Формула Стерджеса для нахождения длины интервала варьирования



Формула для расчёта мат. ожидания:



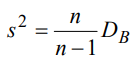
Формула для расчёта дисперсии:



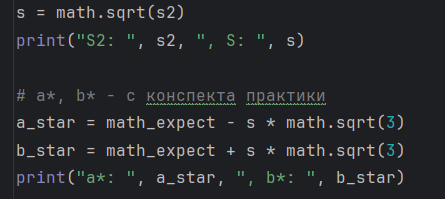
Формула для расчёта среднеквадратического отклонения:



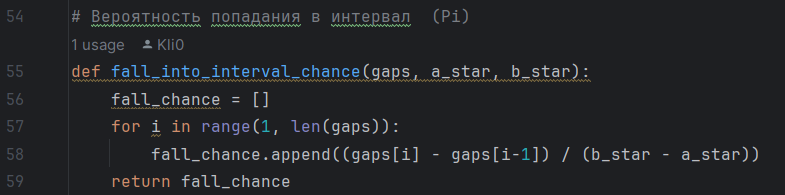
Формула для расчёта несмещенной оценки дисперсии случайной величины:



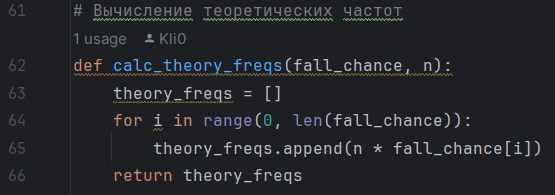
Формулы расчёта a\* и b\*:



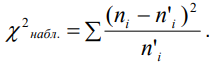
Формула расчёта вероятностей попадания в интервал:



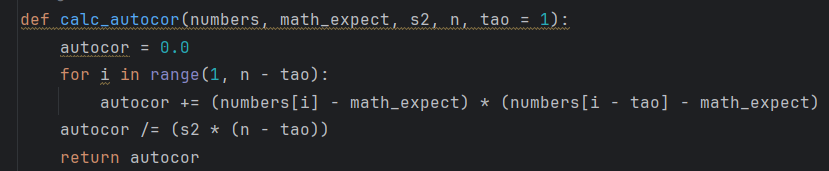
Формула расчёта теоретических частот:



Формула расчёта статистики наблюдений:



Формула расчёта коэффициента автокорреляции:



* 1. Результат работы программы

В результате работы программы построен график относительных частот:

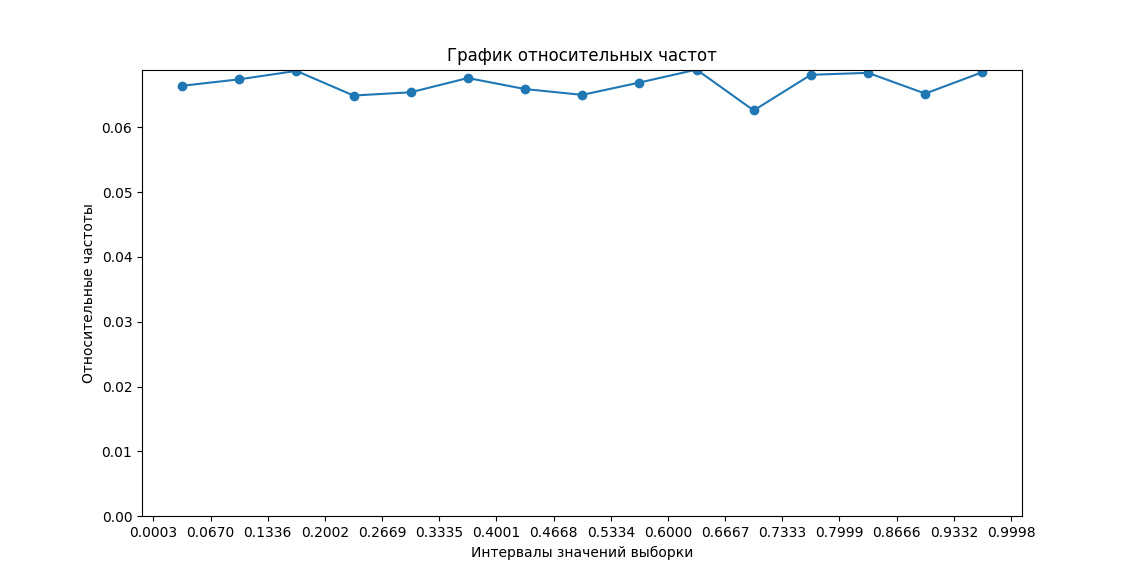


Рисунок 1 – График относительных частот

Также найдены статистика наблюдений, критическая оценка и коэффициент автокорреляции:



ЗАКЛЮЧЕНИЕ

На основе результатов вычислений можно сделать вывод о том, что нет оснований отвергнуть гипотезу о том, что эмпирические и теоретические частоты различаются незначимо. Также нет оснований для утверждения о наличии зависимостей при генерации выборочных данных

ПРИЛОЖЕНИЕ

Код программы

import numpy as np

import matplotlib.pyplot as plt

import math

# Количество генерируемых точек

N = 10000

# Уровень значимости

ALPHA = 0.01

# Число степеней свободы

R = 2

# Взял значения таблицы распределения X^2 для ALPHA == 0.01 (стр. 78 пособия)

X2 = [6.635, 9.210, 11.345, 13.237, 15.086, 16.812, 18.475, 20.090, 21.666, 23.209, 24.795, 24.217, 27.688, 29.141, 30.578, 32.000, 32.409, 34.805, 36.191, 37.566, 38.932, 40.289, 41.638, 42.980, 42.314, 45.642, 46.963, 48.278, 49.588, 50.892]

def in\_gap\_filter(num, left, right, is\_last):

if num >= left:

if is\_last:

if num <= right:

return True

else:

if num < right:

return True

return False

# Рассчет середин отрезков

def calc\_mids(gaps, k):

mids = []

for i in range(1, k + 1):

mids.append((gaps[i - 1] + gaps[i]) / 2)

return mids

# Вычисление мат. ожидания

def calc\_math\_expect(mids, freqs, n) -> float:

math\_expect = 0.0

for i in range(0, len(mids)):

math\_expect += (mids[i] \* freqs[i])

math\_expect /= n

return math\_expect

# Вычисление дисперсии

def calc\_variance(mids, freqs, math\_expect, n):

variance = 0.0

for i in range(0, len(mids)):

variance += (pow(mids[i], 2) \* freqs[i])

variance = (variance / n) - pow(math\_expect, 2)

return variance

# Среднеквадратичное отклонение

def calc\_diviation(variance):

return math.sqrt(variance)

# Вычисление несмещённой оценки дисперсии случайной величины

def calc\_s2(variance, n):

s2 = (n / (n - 1)) \* variance

return s2

# Вероятность попадания в интервал (Pi)

def fall\_into\_interval\_chance(gaps, a\_star, b\_star):

fall\_chance = []

for i in range(1, len(gaps)):

fall\_chance.append((gaps[i] - gaps[i-1]) / (b\_star - a\_star))

return fall\_chance

# Вычисление теоретических частот

def calc\_theory\_freqs(fall\_chance, n):

theory\_freqs = []

for i in range(0, len(fall\_chance)):

theory\_freqs.append(n \* fall\_chance[i])

return theory\_freqs

# Рассчет статистики наблюдений

def calc\_stats(freqs, theory\_freqs):

x\_stat = 0.0

for i in range (0, len(freqs)):

x\_stat += pow((freqs[i] - theory\_freqs[i]), 2) / theory\_freqs[i]

return x\_stat

def check\_hypotesis(x\_stat, x2):

if x2 <= x\_stat:

print("Гипотеза отвергается. Xstat = {} >= Xcrit = {}".format(x\_stat, x2))

else:

print("Гипотеза не отвергается. Xstat = {} < Xcrit = {}".format(x\_stat, x2))

# Вычисление коэффициента автокорреляции, tao (τ) - смещение

# Верхняя граница n - tao должна включаться? [] или [)?

def calc\_autocor(numbers, math\_expect, s2, n, tao = 1):

autocor = 0.0

for i in range(1, n - tao):

autocor += (numbers[i] - math\_expect) \* (numbers[i - tao] - math\_expect)

autocor /= (s2 \* (n - tao))

return autocor

def check\_dependency(autocor):

if abs(autocor) < 1:

print("Нет зависимости при генерации: |r(τ)| < 1 (|r(τ)| = {})".format(abs(autocor)))

else:

print("Есть зависимость при генерации: |r(τ)| >= 1 (|r(τ)| = {})".format(abs(autocor)))

# если диапазон от 0 до 1 - надо сгенерировать 100 чифек и сложить в массив

numbers = np.random.uniform(low=0.0, high=1.0, size=(N,))

min = (min(numbers))

max = (max(numbers))

print("MIN: {}".format(min))

print("MAX: {}".format(max))

# кол-во гистограмм

k = math.ceil(1 + math.log(N, 2))

h = ((max - min) / k)

print("k: {}".format(k))

print("h: {}".format(h))

gaps = [min]

for i in range(1, k + 1):

gaps.append((gaps[i - 1] + h))

print("GAPS: ", gaps)

freqs = [0] \* k

for num in numbers:

for j in range(1, k + 1):

if in\_gap\_filter(num, gaps[j - 1], gaps[j], True if (j == k) else False):

freqs[j - 1] += 1

break

print(freqs)

rel\_freqs = [0.0] \* k

for i in range(0, len(freqs)):

rel\_freqs[i] = freqs[i] / N

print(rel\_freqs)

mids = calc\_mids(gaps, k)

print("Middles: ", mids)

math\_expect = calc\_math\_expect(mids, freqs, N)

print("Math expectation: ", math\_expect)

variance = calc\_variance(mids, freqs, math\_expect, N)

print("Variance: ", variance)

deviation = calc\_diviation(variance)

print("Diviation: ", deviation)

s2 = calc\_s2(variance, N)

s = math.sqrt(s2)

print("S2: ", s2, ", S: ", s)

# a\*, b\* - с конспекта практики

a\_star = math\_expect - s \* math.sqrt(3)

b\_star = math\_expect + s \* math.sqrt(3)

print("a\*: ", a\_star, ", b\*: ", b\_star)

# Используя критерии‌ Пирсона, при уровне значимости 0,01 проверить, согласуется ли гипотеза о равномерном распределении генеральнои‌ совокупности

# Плотность распределения

dencity = 1.0 / (b\_star - a\_star)

print("Dencity: ", dencity)

# Оценка вероятности попадания в интервал

fall\_chance = fall\_into\_interval\_chance(gaps, a\_star, b\_star)

print("Fall into intervals chances: ", fall\_chance)

# Вычисление n`

theory\_freqs = calc\_theory\_freqs(fall\_chance, N)

print("Theoretical frequencies: ", theory\_freqs)

# Статистика X^2

x\_stat = calc\_stats(freqs, theory\_freqs)

print("X^2: ", x\_stat)

# По критерию Пирсона при уровне значимости alpha == 0,01 требуется проверить, значимо или нет различие в частотах ni и ni'

# Дополнительный -1 для отображения данных из таблицы на логику массива

x2 = X2[k - R - 1 - 1]

check\_hypotesis(x\_stat, x2)

# Используя коэффициент автокореляции, проверить касисьво генеральной совокупности на независимость

autocor = calc\_autocor(numbers, math\_expect, s2, N)

check\_dependency(autocor)

#create graphic

ax = plt.gca()

plt.plot(mids, rel\_freqs, marker='o', linestyle='-')

plt.xticks(gaps)

plt.ylim(0.0, np.max(rel\_freqs))

plt.title('График относительных частот')

plt.xlabel('Интервалы значений выборки')

plt.ylabel('Относительные частоты')

plt.show()