|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |
| Институт комплексной безопасности и специального приборостроения |
| Кафедра «Цифровые технологии обработки данных» |

**ОТЧЕТ**

**по практической работе**

**«Оценка качества генераторов псевдослучайных чисел»**

**по дисциплине «Моделирование систем»**

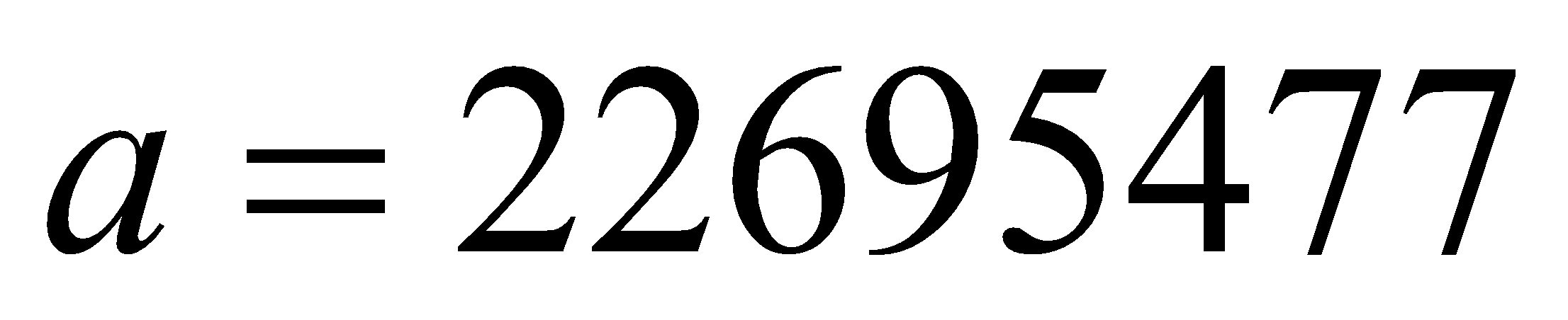
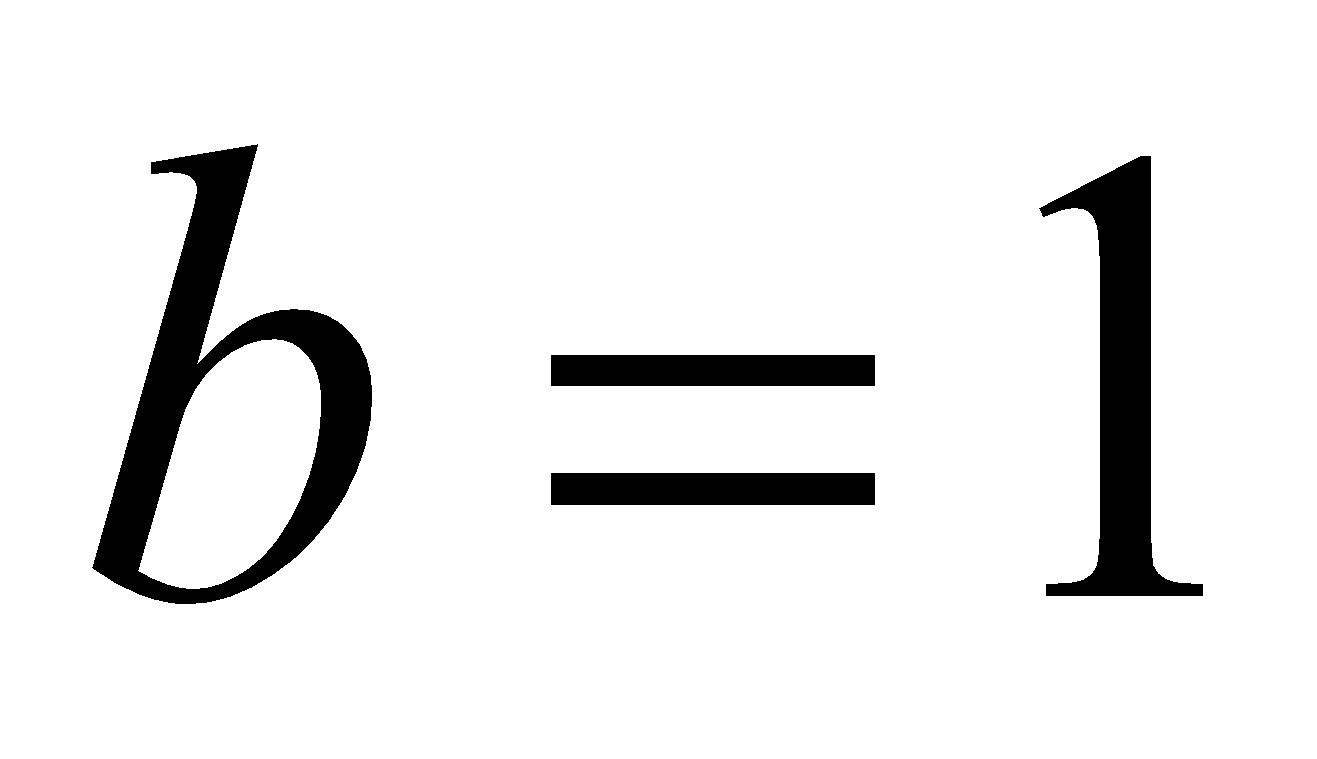
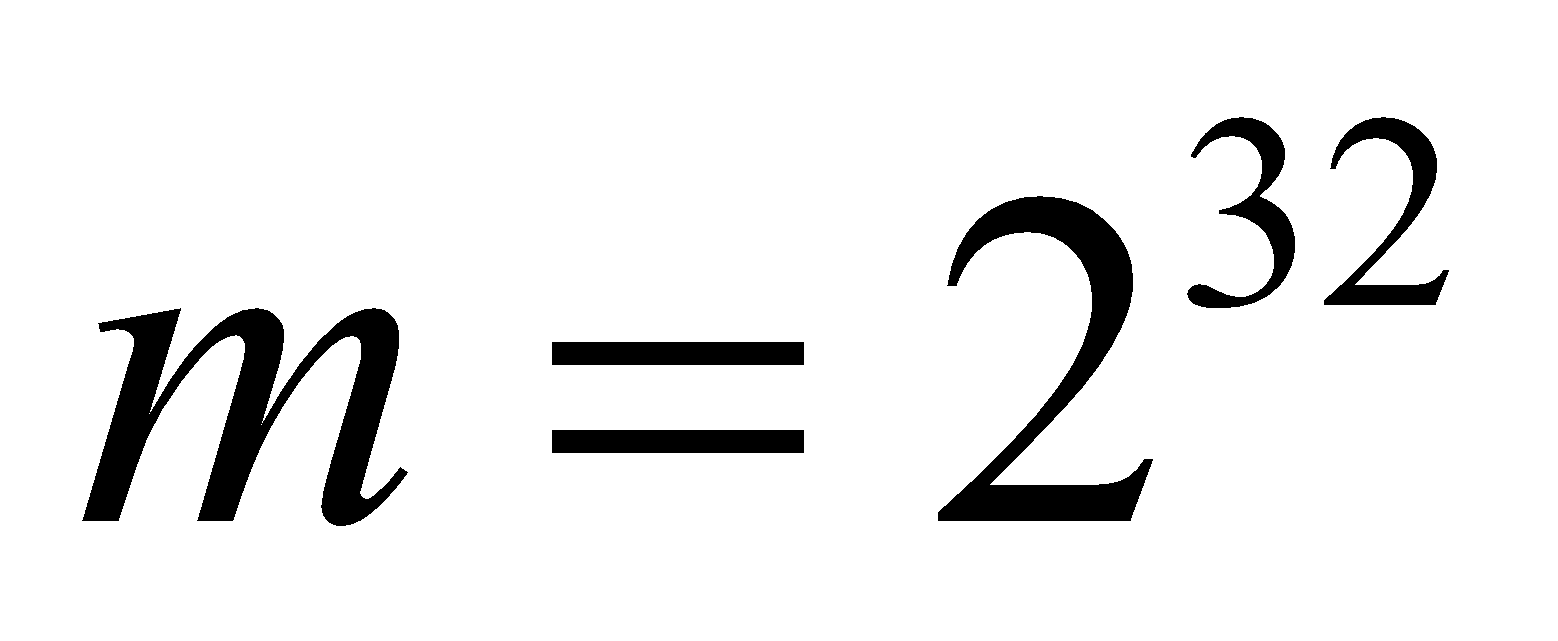
|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил |  |  | Смирнов И.А.  *фамилия, имя, отчество* |
| шифр | 21Б0700 | группа | БСБО-11-21 |
|  |  |  |  |
| Проверил |  |  | Хорсик И.А.  *фамилия, имя, отчество* |

**Москва 2023г.**

**Цель работы.**

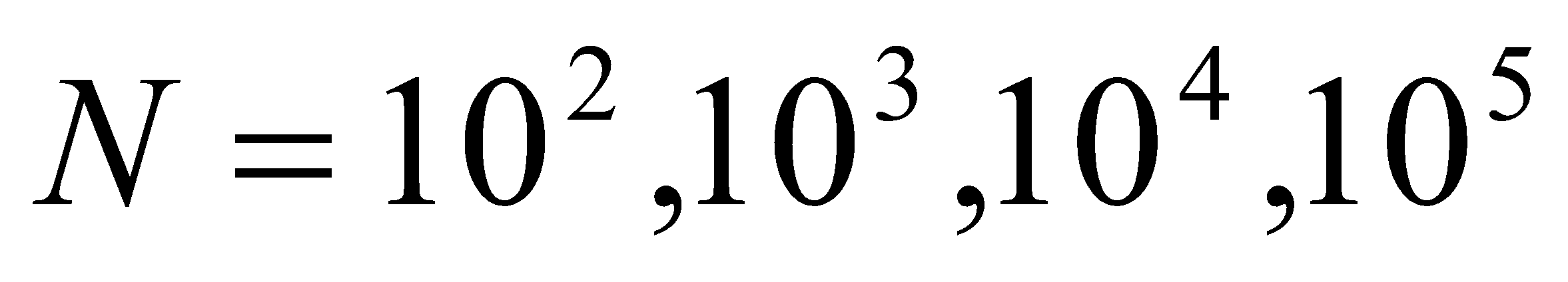
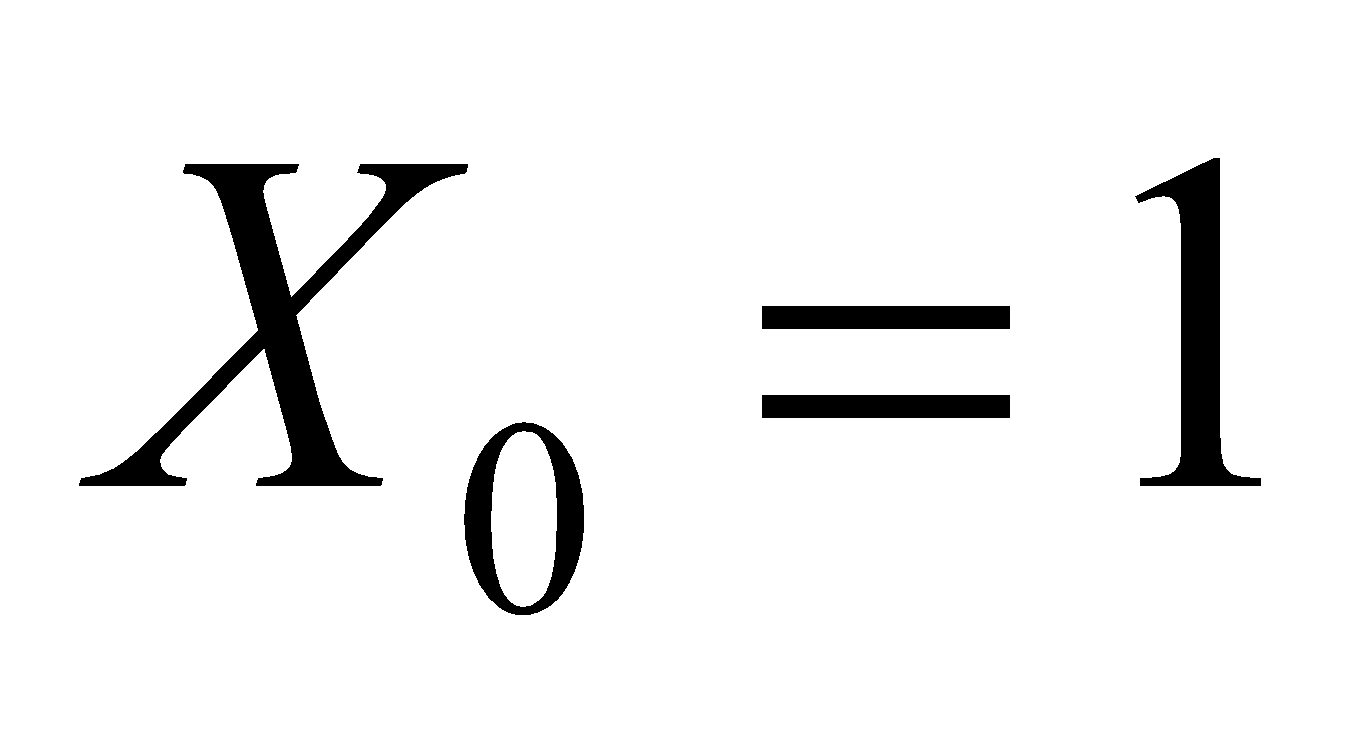
Проведение сравнительного анализа качества генераторов псевдослучайных чисел, на примере генератора на основе мультипликативного метода и встроенного генератора MathCAD.

Задание №1.

Создать функцию случайной последовательности чисел со следующими параметрами: , , .

# Task 1  
def rand(n, a=22695477, b=1, m=2 \*\* 32, x0=1):  
 random\_numbers = []  
 x = x0  
  
 for \_ in range(n):  
 x = (a \* x + b) % m  
 normal\_number = x / m \* (B - A) + A  
 random\_numbers.append(normal\_number)  
  
 return random\_numbers

Задание №2.

Генерация и сохранение последовательностей для каждого N Сгенерировать с помощью функции, реализующей мультипликативный метод, равномерно распределенные последовательности случайных чисел в интервале от *A=0* до *B=10* длинной . В качестве начального значения для генератора использовать .

A = 0  
B = 10  
  
list\_numbers = []   
  
N\_values = [10\*\*2, 10\*\*3, 10\*\*4, 10\*\*5]   
  
for N in N\_values:  
 random\_numbers = RAND(n=N) # задание 1  
 #random\_numbers = [random.uniform(A, B) for \_ in range(N)] # задание 7  
 list\_numbers.append(random\_numbers)  
  
#№ вывод последовательностей  
#for idx, random\_numbers in enumerate(list\_numbers):  
# print(f'N={N\_values[idx]}: Интервал [{random\_numbers}]')

Задание №3.

Рассчитать для сгенерированных последовательностей математическое ожидание и дисперсию. Сравнить полученные значения с математическим ожиданием и дисперсией теоретической равномерно распределенной случайной величиной.

# Task 3  
def mat\_wait(random\_numbers):  
 return sum(random\_numbers) / len(random\_numbers)  
  
def dispers(random\_numbers, mean):  
 return sum((x - mean) \*\* 2 for x in random\_numbers) / (len(random\_numbers) - 1)  
  
theor\_mat = (A + B) / 2  
theor\_dispers = (B - A) \*\* 2 / 12  
print(f'Теоретическое математическое ожидание: {theor\_mat}')  
print(f'Теоретическая дисперсия: {theor\_dispers}')  
  
for idx, N in enumerate(N\_values):  
 random\_numbers = list\_numbers[idx]  
 mat\_value = mat\_wait(random\_numbers)  
 dispers\_value = dispers(random\_numbers, mat\_value)  
  
 error\_mat = abs((mat\_value - theor\_mat) / theor\_mat) \* 100  
 error\_dispers = abs((dispers\_value - theor\_dispers) / theor\_dispers) \* 100  
  
 print(f'N={N}:')  
 print(f' Математическое ожидание: {mat\_value}, Погрешность: {error\_mat}')  
 print(f' Дисперсия: {dispers\_value}, Погрешность: {error\_dispers}')  
 print('------------------------------')

Результат:

Теоретическое математическое ожидание: 5.0

Теоретическая дисперсия: 8.333333333333334

N=100:

Математическое ожидание: 4.844686777098104, Погрешность: 3.106264458037913

Дисперсия: 8.186038407097204, Погрешность: 1.7675391148335606

------------------------------

N=1000:

Математическое ожидание: 5.082843668749556, Погрешность: 1.6568733749911146

Дисперсия: 8.127648959809106, Погрешность: 2.4682124822907383

------------------------------

N=10000:

Математическое ожидание: 4.997016315227374, Погрешность: 0.05967369545251344

Дисперсия: 8.30805367255168, Погрешность: 0.30335592937985467

------------------------------

N=100000:

Математическое ожидание: 4.992300205504522, Погрешность: 0.15399588990955948

Дисперсия: 8.344333893779716, Погрешность: 0.13200672535658242

------------------------------

Задание №4.

Определить период сгенерированной последовательности случайных чисел.

Создаем функцию *random\_period*, определяющую период сгенерированной последовательности случайных чисел.

#task 4  
def rand\_period(random\_numbers):  
 n = len(random\_numbers)  
 period = 0  
 first\_pos = 1  
 second\_pos = 2  
  
 for i in range(n):  
 element = random\_numbers[i]  
 for j in range(i, n):  
 if element == random\_numbers[j] and i != j:  
 period = j - i  
 first\_pos = i  
 second\_pos = j  
 return period, first\_pos, second\_pos  
  
 period = -1  
 first\_pos = -1  
 second\_pos = -1  
 return period, first\_pos, second\_pos

С помощью разработанной функции *random\_period* проводим тестирование полученных последовательностей случайных чисел на периодичность.

for idx, random\_numbers in enumerate(list\_numbers):  
 period, first\_pos, second\_pos = rand\_period(random\_numbers)  
 print(f'Период: {period}, Первое вхождение: {first\_pos}, Второе вхождение: {second\_pos}')

Результат:

Период: -1, Первое вхождение: -1, Второе вхождение: -1

Период: -1, Первое вхождение: -1, Второе вхождение: -1

Период: -1, Первое вхождение: -1, Второе вхождение: -1

Период: -1, Первое вхождение: -1, Второе вхождение: -1

Задание №5.

Реализуем функцию определения относительных частот случайных

чисел по известной выборке (см. задание 2). Входными параметрами

функции являются: выборка случайных чисел RParamsArr, левая и правая

граница выборки A и B, и количество участков IntervalsCount. Выходным

параметром – относительные частоты для заданных участков.

#Task 5-6  
  
def get\_freq\_distr(X, A, B, IntervalsCount, dY):  
 Freq = [0] \* IntervalsCount  
  
 for Yc in X:  
 fN = math.floor((Yc - A) / dY)  
 if fN == IntervalsCount:  
 fN -= 1  
 Freq[fN] += 1  
  
 for i in range(IntervalsCount):  
 Freq[i] = Freq[i] / (len(X) \* dY)  
  
 x\_values = [A + (dY \* (0.5 + i)) for i in range(IntervalsCount)]  
  
 return x\_values, Freq  
  
  
for idx, N in enumerate(N\_values):  
 random\_numbers = list\_numbers[idx]  
 dY = (10 - 0) / 10  
 x\_values, frequencies = get\_freq\_distr(random\_numbers, 0, 10, 10, dY)  
  
 plt.bar(x\_values, frequencies, width=dY, align='edge')  
 plt.xlabel('Интервалы')  
 plt.ylabel('Относительные частоты')  
 plt.title(f'Гистограмма для N={N}')  
 plt.show()  
  
 Pearson = 0  
 for k in range(len(frequencies)):  
 Pearson += (1 / 10 - frequencies[k]) \*\* 2 / frequencies[k]  
 print(f'Значение критерия Пирсона для N={N}: {Pearson}')

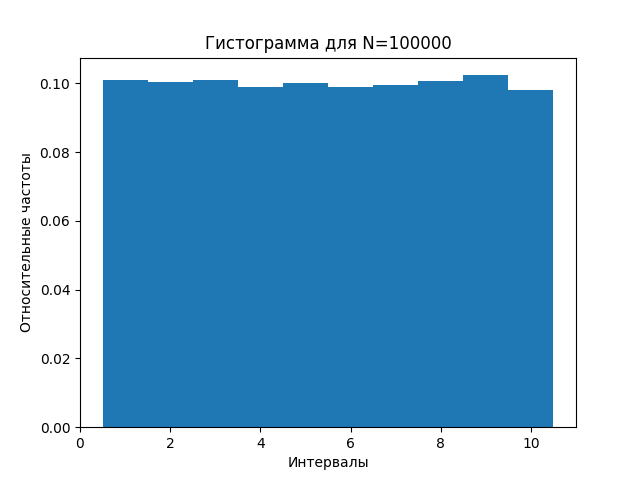
Результат:

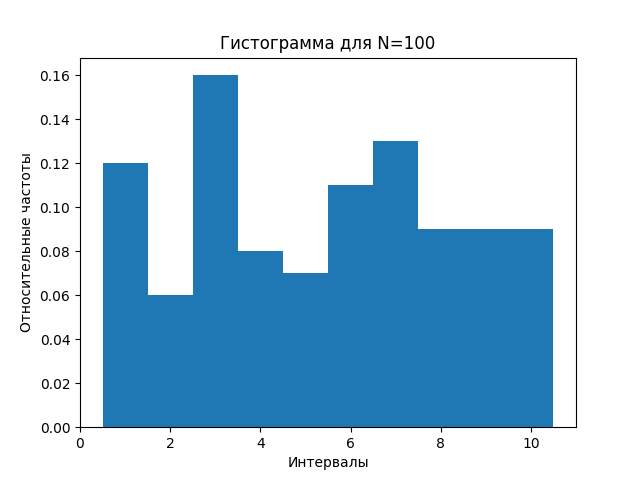
Значение критерия Пирсона для N=100: 0.08152264402264406

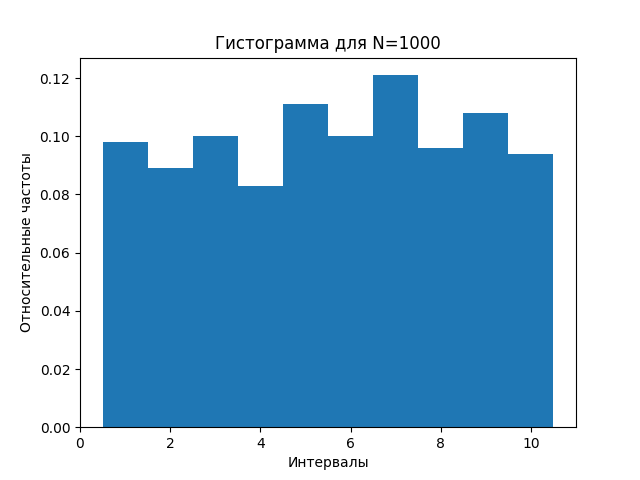
Значение критерия Пирсона для N=1000: 0.010759250771098895

Значение критерия Пирсона для N=10000: 0.0005887585715413947

Значение критерия Пирсона для N=100000: 0.00015121504213031582



****

****

**Задание №7.**

1. Используя функцию MathCAD *rnd*, реализующую встроенный генератор случайных чисел, аналогичным образом повторяем проведенный анализ (см. задания 1-6).

for N in N\_values:  
 # random\_numbers = rand(n=N) # задание 1  
 random\_numbers = [random.uniform(A, B) for \_ in range(N)] # задание 7  
 list\_numbers.append(random\_numbers)

Результат:

Теоретическое математическое ожидание: 5.0

Теоретическая дисперсия: 8.333333333333334

N=100:

Математическое ожидание: 4.727387336431338, Погрешность: 5.4522532713732375

Дисперсия: 8.91915741991003, Погрешность: 7.029889038920359

------------------------------

N=1000:

Математическое ожидание: 5.06809504534767, Погрешность: 1.3619009069534016

Дисперсия: 8.439303647564591, Погрешность: 1.2716437707750856

------------------------------

N=10000:

Математическое ожидание: 5.017028453996858, Погрешность: 0.34056907993715413

Дисперсия: 8.36367672466356, Погрешность: 0.36412069596270413

------------------------------

N=100000:

Математическое ожидание: 4.997617375822399, Погрешность: 0.0476524835520209

Дисперсия: 8.307158281364803, Погрешность: 0.3141006236223731

------------------------------

Период: -1, Первое вхождение: -1, Второе вхождение: -1

Период: -1, Первое вхождение: -1, Второе вхождение: -1

Период: -1, Первое вхождение: -1, Второе вхождение: -1

Период: -1, Первое вхождение: -1, Второе вхождение: -1

Значение критерия Пирсона для N=100: 0.04015151515151516

Значение критерия Пирсона для N=1000: 0.01288692995430771

Значение критерия Пирсона для N=10000: 0.00036271436138986035

Значение критерия Пирсона для N=100000: 9.574283372414851e-05

