Назарова К.А. Группа № в5130904/30030

Расчетная работа №3. Проверка встроенного генератора случайных числе (ГСЧ) и создание на его базе собственного.

1. Проверка встроенного ГСЧ

Задание

- При помощи библиотечного ГСЧ Python 3.9 получить равномерно распределенные числа на промежутке [0;1].
- Для этих чисел подсчитать математическое ожидание и СКО.
- Сравнивнить с теоретическими значениями:

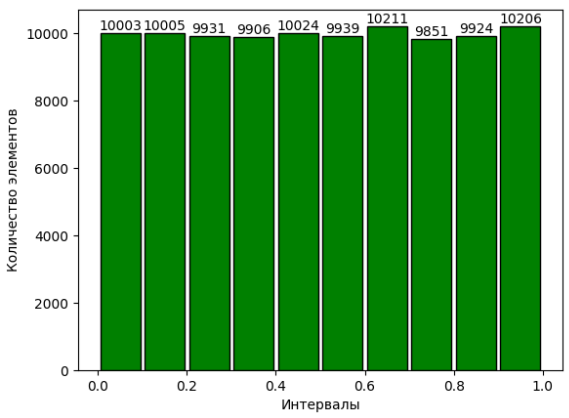
$$m_r = \frac{\sum_{i=1}^{n} r_i}{n} = 0.5$$
$$D_r = \frac{\sum_{i=1}^{n} (r_i - m_r)^2}{n} \approx 0.083$$
$$\sigma_r = \sqrt{D_r} \approx 0.288$$

- Построить частотную диаграмму.

Полученные результаты.

Промежуток нахождения случайных чисел [0,1]. Количество случайных чисел - 100 000.

	Теоритическое значение	Расчетное значение
m_r	0.5	0.4984
D_r	0.083	0.0834
σ_r	0.288	0.2888



2. **Создание собственного ГСЧ**

Задание

- На базе библиотечного ГСЧ создать свой, генерирующий случайные числа по нормальному закону с параметрами (,) Для этого можно воспользоваться методом построения по центральной предельной теореме (ЦПТ) или методом Мюллера.
- При помощи сдвига на m_x и масштабирования на σ_x преобразовать нормализованные СЧ (ряд Z) в нужный ряд X с параметрами (m_x=0, $\sigma_x = \Delta y$) :

  $$x = z\sigma_x + m_x$$

  Параметр $\sigma_x = \Delta y$ высчитывается на основе полученного y(t) из работы №1:

  $$\sigma_x = 0,05 * MAX|y(t)|$$

- При помощи созданного ГСЧ требуется получить последовательность случайных чисел.
- Высчитать реальные значения m_x=0 и σ_x.
- Построить частотную диаграмму.

Полученные результаты

$\sigma_x = \Delta y = 0.05 * 8.99997 = 0,45$

Промежуток нахождения случайных чисел [0,1].

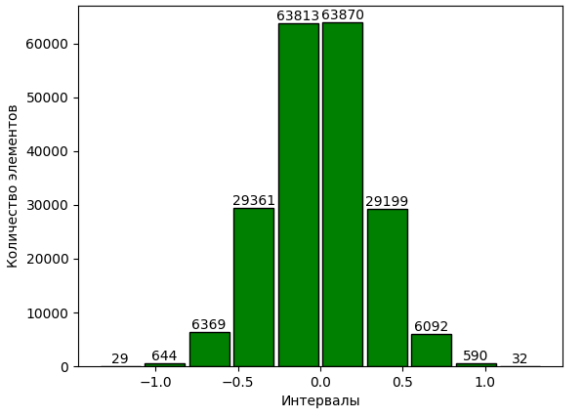
Количество случайных чисел - 100 000.

Метод Мюллера

При создании собственного ГСЧ выбран был метод Мюллера по следующим причинам:

- **Быстрая скорость генерации** - метод Мюллера является относительно быстрым и эффективным способом генерации случайных чисел, что делает его привлекательным для использования в различных вычислительных приложениях.
- **Хорошее качество случайных чисел** - метод Мюллера обеспечивает высокое качество случайных чисел, что позволяет избегать повторений и предсказуемости в генерируемой последовательности.
- **Простота реализации** - метод Мюллера не требует большого количества вычислений или сложных математических операций для своей реализации, что делает его относительно простым и легким в использовании.
- **Гибкость и настраиваемость** - метод Мюллера может быть легко настроен для генерации случайных чисел в определенном диапазоне или с определенным распределением вероятностей, что делает его удобным инструментом для различных задач.

	Теоритическое значение	Расчетное значение
m_r	0.00	0.002
σ_r	0.45	0.4192



3. Код программы

```
from cmath import sqrt
import random
from matplotlib import pyplot as plt
import math


delta_y = round (0.05 * 8.99997, 4)
print(delta_y)
iter = 100000

def calcVariables(numbers_list:list , m_calc, d_calc, sigma_calc):
    m_calc = round(sum(numbers_list) / iter , 4)
    d_calc = round(sum([(num - m_calc) **2 for num in numbers_list
       ]) / iter ,4)
    sigma_calc = round(d_calc ** 0.5, 4)

    return m_calc, d_calc, sigma_calc


def muller_generator() :
    r1 , r2 = random.uniform(0,1), random.uniform(0,1)
    z1 = math.cos(2 * math.pi * r1 ) * ( -2 * math.log10(r2)) **
       0.5
    z2 = math.sin (2 * math.pi * r1 ) * ( -2 * math.log10(r2)) **
       0.5
    x1 = z1 * delta_y
    x2 = z2 * delta_y
    return x1 , x2

def create_hist ( numbers : list , dest_graph , bins ) : counts,
   edges, bars = plt.hist(numbers, bins = bins, edgecolor ="black",
    rwidth =0.9,color='green')
    plt.bar_label(bars)
    plt.xlabel("Intervals")
    plt.ylabel("Amount of elements")
    plt.savefig(dest_graph)
    return dest_graph


```

```
35 if __name__ == "__main__":
36
37     random_num = [random.uniform(0, 1) for n in range(iter)]
38     bins = [round( z * 0.1, 1) for z in range (0, 11)]
39     createPlot = create_hist(random_num, "inner.png", bins=bins)
40     inner_m = inner_d = inner_sigma = 0
41     print (" Built-in RNG calculations: M, D, Sigma ",
           calcVariables(random_num,inner_m , inner_d, inner_sigma))
42
43
44     muller_numbers = []
45     for _ in range (iter) :
46         x1 , x2 = muller_generator()
47         muller_numbers.append(x1)
48         muller_numbers.append(x2)
49
50     muller_bins = [k * 0.27 for k in range (-5, 6) ]
51     muller_createPlot = create_hist(muller_numbers , "muller.png",
           bins = muller_bins)
52
53     muller_m = muller_d = muller_sigma = 0
54     print ("Calculations using the Muller method: M, D, Sigma ",
           calcVariables(muller_numbers, muller_m, muller_d,
           muller_sigma))
```

4. Вывод

В ходе исследования была изучена функция random() библиотеки Python 3.9, а также генерация нормализованных случайных чисел с использованием метода Мюллера.

Результаты показали, что математическое ожидание и среднеквадратичное отклонение встроенного ГСЧ близки к теоретическим значениям, отклонение от теоретических значений при использовании метода Мюллера незначительно.