НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Лабораторна робота №1
«Моделювання випадкових величин»
з кредитного модуля «Випадкові процеси»
Варіант 11

Виконав:
студент групи КМ-93
Костенко Олександр Андрійович
Викладач:
Пашко Анатолій Олексійович

Київ — 2021

Зміст

Завдання 1

1. Згенерувати послідовність з n =1000000 псевдовипадкових чисел, що рівномірно розподілені на інтервалі (0,1) (використати вбудований генератор псевдовипадкових чисел). Побудувати графік.

1.1. Оцінити математичне сподівання та дисперсію отриманої послідовності.

1.2. Побудувати таблицю 1 (кількість L підінтервалів не менше 10), частотну таблицю вивести на екран.

Таблиця 1. Частотна таблиця

Інтервал	Кількість чисел (частота попадань), які випали в даний інтервал	Відносна частота потрапляння
Δ_1	ν_1	ν_1/n
Δ_2	ν_2	ν_2/n
…	…	…
Δ_L	ν_L	ν_L/n
	$\sum$ кіл-ть	

1.3. Перевірити гіпотезу про закон розподілу, побудувати гістограму.

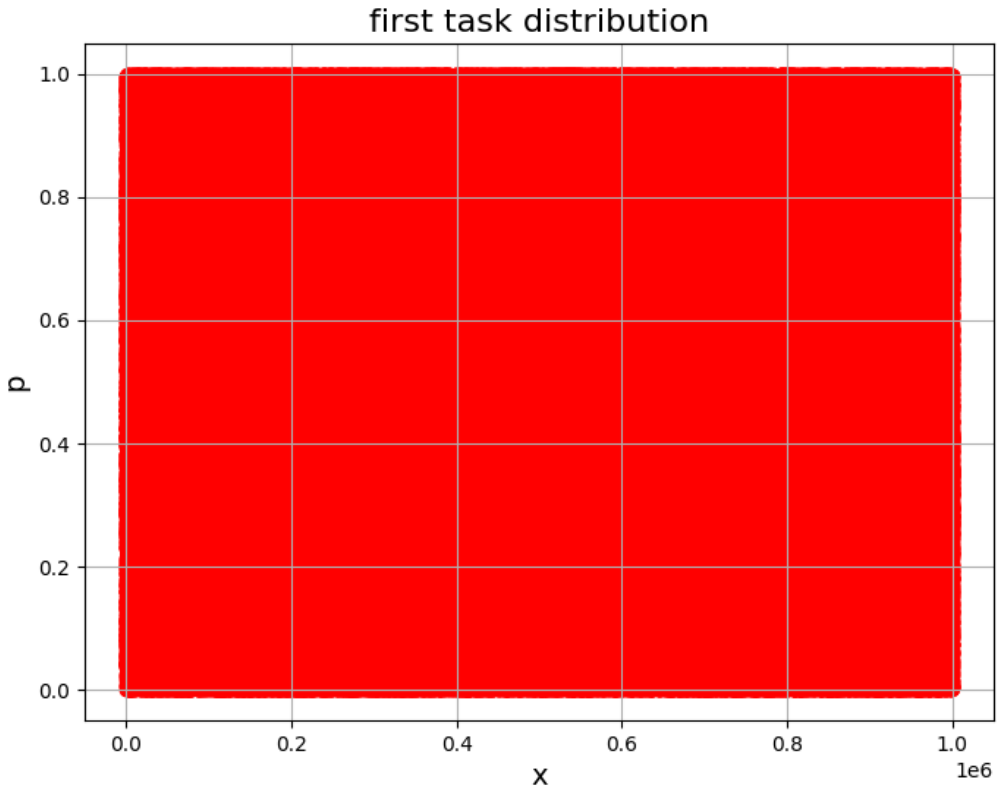
1.4. Дослідити розподіл випадкової величини $\eta = \max(\xi)$, де ξ- рівномірно розподілена на (0,1) випадкова величина, для цього:

а) змоделювати m =1000 значень величини ξ, знайти максимальне значення $\eta_1 = \max(\xi)$;

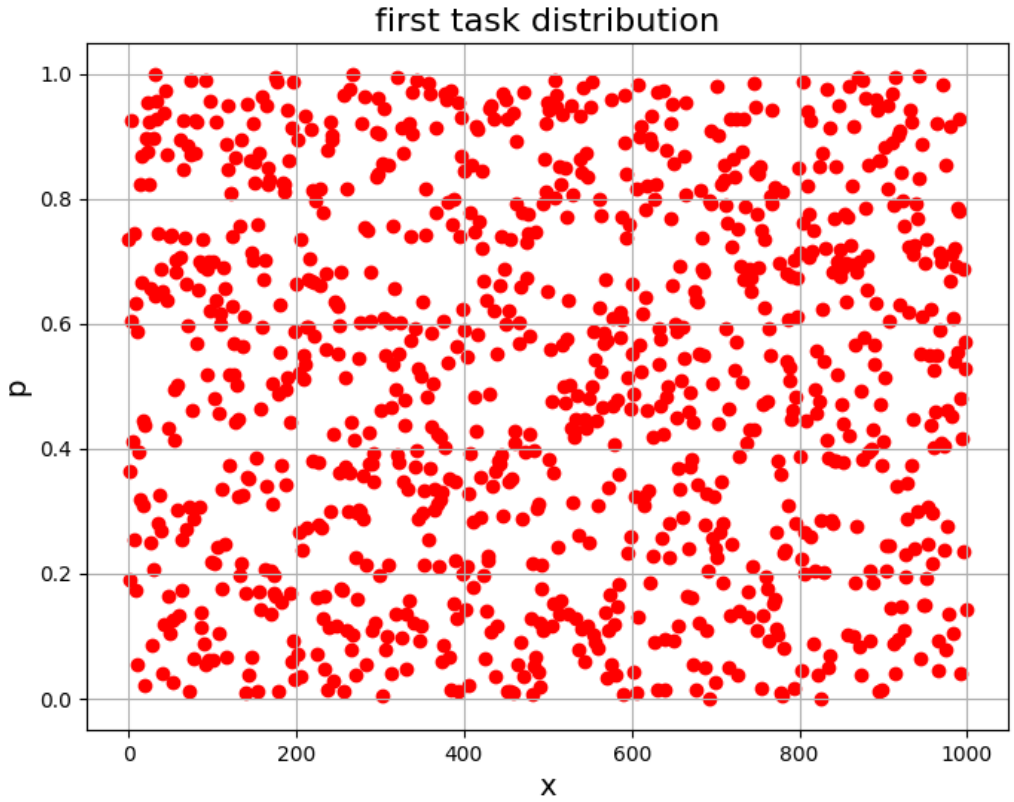
б) процедуру а) повторити n=1000000;

в) для випадкової величини $\eta = \{\eta_i, i = 1,..., n\}$ побудувати гістограму.

1. Для генерації послідовності псевдовипадкових чисел та побудови графіків було застосовано бібліотеки numpy та matplotlib відповідно. Графік при n=1000000 виглядатиме так:



Для наочності зменшимо кількість точок до n=1000:



Розподіл рівномірний.

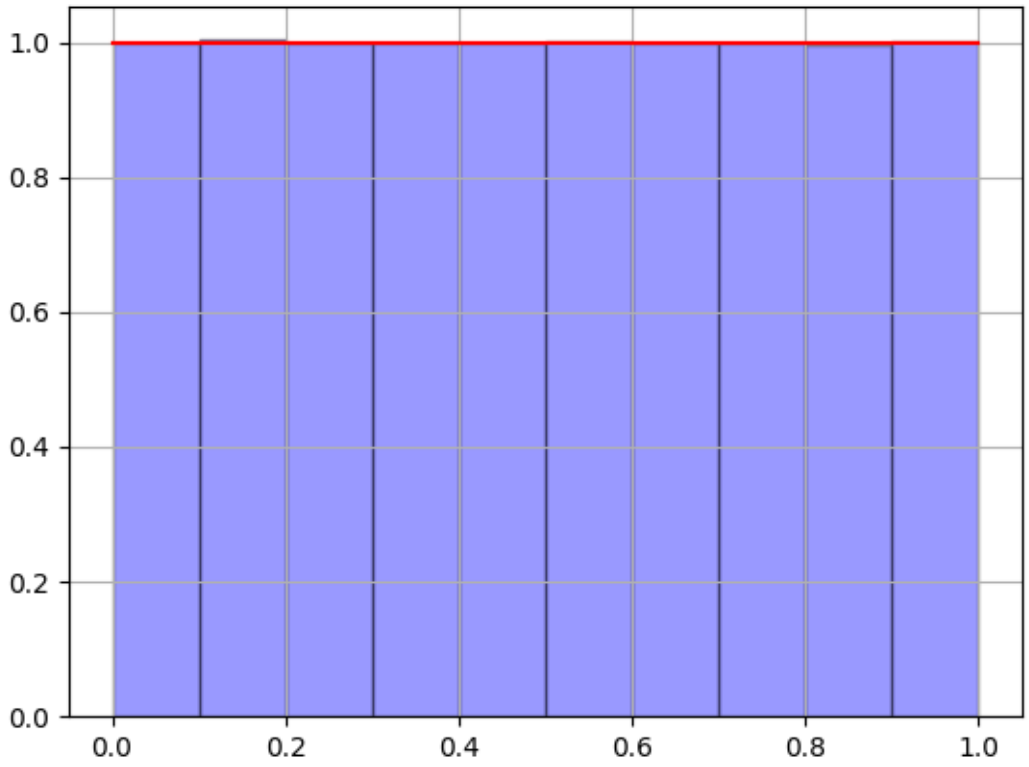
1.1 Математичне сподівання: 0.508094

Дисперсія: 0.08701

1.2

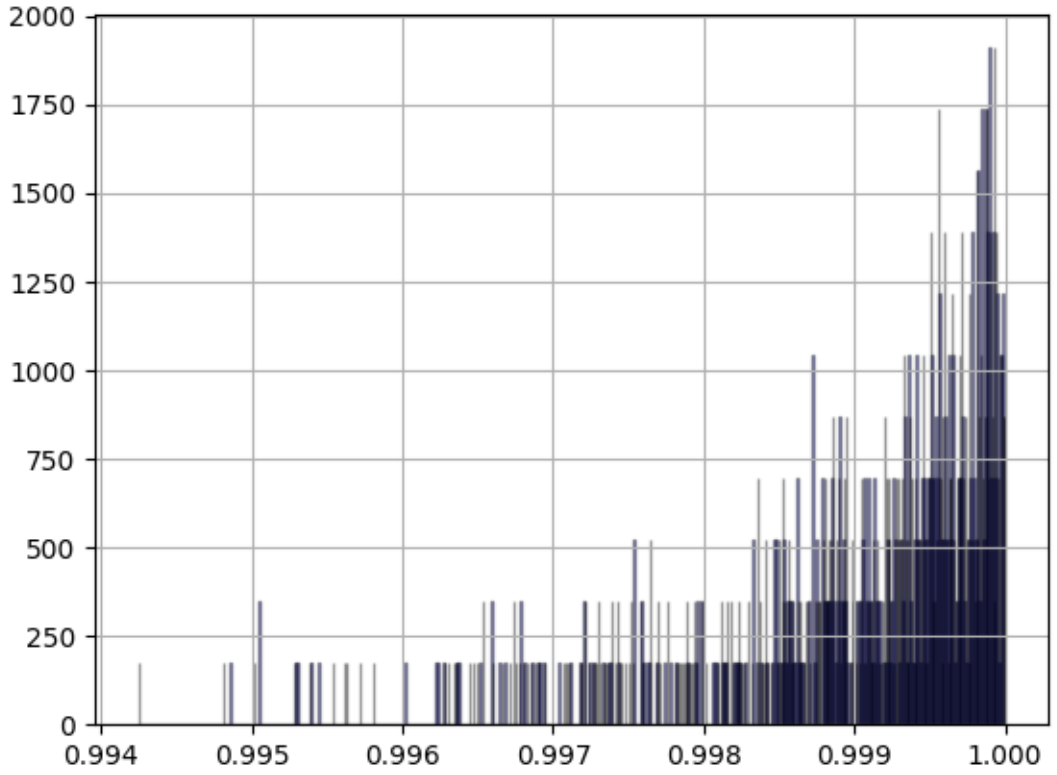
Інтервал	Кількість чисел, що випали в даний інтервал	Відносна частота потрапляння
(0;0.077]	76830	0.07683
(0.077;0.154]	77071	0.077071
(0.154;0.231]	77490	0.07749
(0.231;0.308]	76958	0.076958
(0.308;0.385]	76975	0.076975
(0.385;0.462]	76614	0.076614
(0.462;0.538]	76906	0.076906
(0.538;0.615]	76940	0.07694
(0.615;0.692]	77132	0.077132
(0.692;0.769]	76904	0.076904
(0.769;0.846]	76569	0.076569
(0.846;0.923]	76852	0.076852
(0.923;1.0]	76759	0.076759
	Загальна кількість: 1000000	

1.3



Стовпчики гістограми повторюють контур графіка щільності розподілу, який ми також побудували (y = 1), отже гіпотеза не відкидається.

1.4 Отримана наступна гістограма:



Код програми:

```
import numpy as np
from matplotlib import pyplot as plt
from prettytable import PrettyTable as pt
import random
import math

def dist(n):
    d=np.random.uniform(0,1,n)
    return d

def graph(d,n,title):
    x=range(0,n)
    fig, ax=plt.subplots(figsize=(8,6))
    ax.scatter(x,d, c='red')
    ax.set_title(title, fontsize=16)
    ax.set_xlabel("x", fontsize=14)
    ax.set_ylabel("p", fontsize=14)
    plt.grid()
    plt.show()
```

```
#___1.1___
def mean(d):
    res=[]
    res.append(np.mean(d))
    res.append(np.var(d))
    print("Математичне сподівання = ", round(res[0],6))
    print("Дисперсія = ", round(res[1],6))

#___1.2___
def table(d, l, n):
    sorted=np.sort(d)
    li=1/l
    am=0
    curr=0
    sum=0
    Table=pt(["Інтервал", "Кількість чисел, що випали в даний інтервал",
"Відносна частота потрапляння"])
    while am<n:
        curr_am=0
        while sorted[am]<curr+li:
            curr_am+=1
            am+=1
            if am==n:
                break
        Table.add_row(["("+str(round(curr,3))+";"+ str(round(curr+li,
3))+"]", curr_am, curr_am/n])
        curr+=li
        sum+=curr_am
    Table.add_row(["", "Загальна кількість: "+str(sum), ""])
    print(Table)

#___1.3___
def check(d):
    plt.hist(d, bins=10, density=True, color="blue", alpha=0.4,
edgecolor="black")
    plt.plot([0,1], [1,1], color="red")
    plt.grid()
    plt.show()

#___1.4___
def hist():
    n_maxs=[]
    for i in range(1000):
        n=np.random.uniform(0,1, 1000)
        n_max=max(n)
```

```
        n_maxs.append(n_max)
    plt.hist(n_maxs, bins=1000, density=True,  color="blue", alpha=0.4,
edgecolor="black")
    plt.grid()
    plt.show()

def first(n):
    distr=dist(n)
    graph(distr, n, "first task distribution")
    mean(distr)
    table(distr, 13,n )
    check(distr)
    hist()

n=1000000
first(n)
```

Завдання 2

2. Змоделювати дискретну випадкову величину, задану таблицею 2, побудувати графік.

2.1.Оцінити математичне сподівання та дисперсію отриманої дискретної випадкової величини.

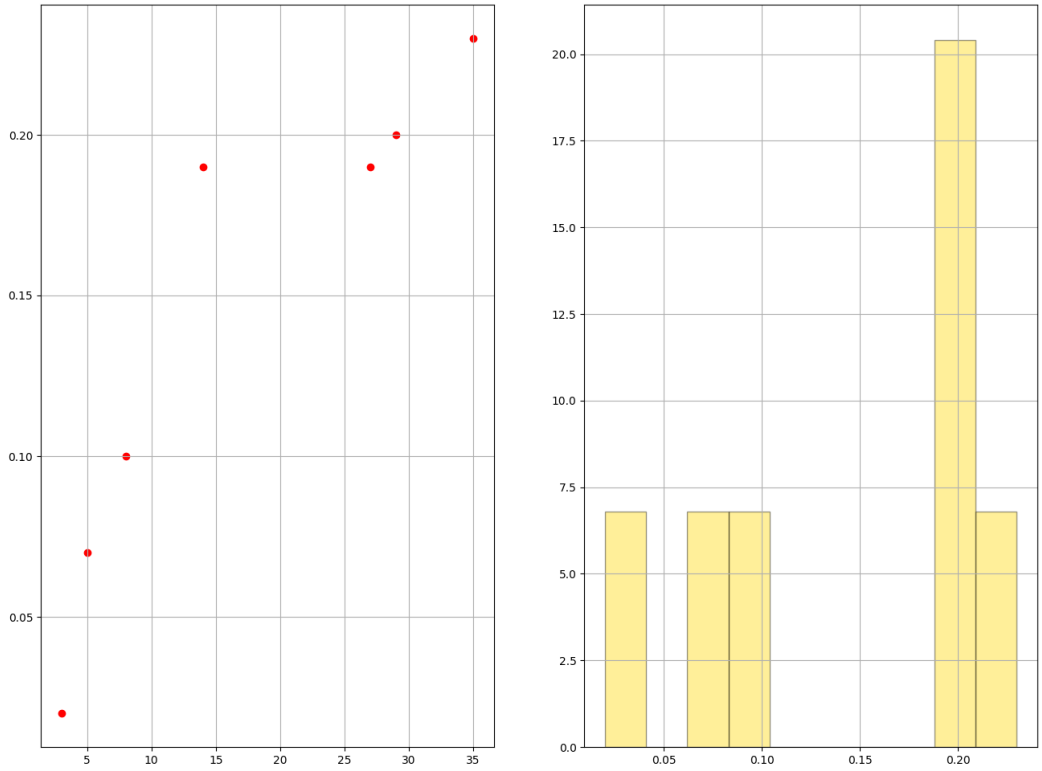
2.2. Побудувати частотну таблицю.

2.3. Перевірити гіпотезу про закон розподілу, побудувати гістограму.

Таблиця 2. Таблиця розподілів

11	x_i	3	5	8	14	27	29	35
	p_i	0.02	0.07	0.1	0.19	0.19	0.2	0.23

```
Розподіл:{3: 0.02, 5: 0.07, 8: 0.1, 14: 0.19, 27: 0.19, 29: 0.2, 35: 0.23}
Математичне сподівання =  -108.76125
Дисперсія =  17321.521127
+----------+---------------------------------------------+------------------------------+
| Інтервал | Кількість чисел, що випали в даний інтервал | Відносна частота потрапляння |
+----------+---------------------------------------------+------------------------------+
|    1     |                      3                      |     0.024793388429752067     |
|    2     |                      5                      |     0.04132231404958678      |
|    3     |                      8                      |     0.06611570247933884      |
|    4     |                      14                     |     0.11570247933884298      |
|    5     |                      27                     |      0.2231404958677686      |
|    6     |                      29                     |      0.2396694214876033      |
|    7     |                      35                     |      0.2892561983471074      |
+----------+---------------------------------------------+------------------------------+
```



Червоні точки не співпадають із висотами. А це суперечить гіпотезі про закон розподілу цієї ВВ.

Код програми:

```
def mean(d):
    res=[]
    res.append(np.mean(d))
    res.append(np.var(d))
    print("Математичне сподівання = ", round(res[0],6))
    print("Дисперсія = ", round(res[1],6))


distr={3:0.02, 5:0.07, 8:0.1, 14:0.19, 27:0.19, 29:0.2, 35:0.23}
arr=[]
m=0
m_sqrt=0
num=0
for i in distr:
    m+=i*distr[i]
    num+=i
    m_sqrt=i*i*distr[i]
```

```
arr.append(m)
var=m_sqrt-m*m
arr.append(var)
print("Розподіл:"+str(distr))
mean(arr)
Table=pt(["Інтервал", "Кількість чисел, що випали в даний інтервал",
"Відносна частота потрапляння"])
c=1
for i in distr:
    if len(str(i))==1:
        k=f'{i}'
    else:
        k=i
    Table.add_row([c, k, i/num])
    c+=1

print(Table)

fig, ax = plt.subplots(1, 2, figsize = (16, 12))
ax[0].grid()
ax[1].grid()
ax[0].scatter(distr.keys(),distr.values(), c='red')
ax[1].hist(distr.values(), bins=10, density=True, color = "gold", alpha =
0.4, edgecolor = "black")
plt.show()
```

Завдання 3

3. Змоделювати неперервні випадкові величини, що мають закони розподілу нормальний (гауссовий), Вейбулла, Релея, логнормальний, Коші. Параметри розподілів задати в режимі діалогу. Побудувати графіки отриманих реалізацій.

3.1. Оцінити математичне сподівання та дисперсію отриманих випадкових величин.

3.2. Перевірити гіпотезу про закон розподілу, побудувати гістограму.

Код програми:

```
def weibull_distr(k,l):
    f=[]
    x=np.random.uniform(0,1,10000)
    for i in x:
        f.append(1*math.pow(-math.log(1-x),1/k))
    return np.array(f)
def weibull_mean(k,l):
```

```
    result=[]
    result.append(1*math.gamma(1+1/k))
    result.append(math.pow(l,2)*(math.gamma(1+2/k)-
math.pow((math.gamma(1+1/k)),2)))
    return result

def log_distr(m,s,n):
    f=[]
    for i in range(1000):
        f.append(math.exp(m+s*math.sqrt(12/n)*(np.sum(np.random.random(n)
)-n/2)))
    return np.array(f)
def log_mean(m,s):
    result=[]
    result.append(math.exp(m+s/2))
    result.append((math.exp(s)-1)*math.exp(2*m+s))
    print (result[0])
    print (result[1])


def couchi_distr(g,xo):
    f=[]
    x_massive=np.random.uniform(-1,1,10000)
    for x in x_massive:
        f.append(xo+g*math.tan(math.pi*(x-1/2)))
    return np.array(f)

def rayleigh_distr(s):
    f=[]
    x=np.random.uniform(0,1,10000)
    for i in x:
        f.append(math.sqrt(2*s*s*(-math.log(1-x))))
    return np.array(f)
def rayleigh_mean(m,s):
    result=[]
    result.append(math.pow((s*math.pi/2),1/2))
    result.append((2-math.pi/2)*s)
    print (result[0])
    print(result[1])



def four_l():
    distr=log_distr(0,1,12)
    title="Log distr"
    log_mean()
```

```
    fig, ax=plt.subplots(1,3,figsize=(16,12))
    ax[0].grid()
    ax[1].grid()
    ax[2].grid()
    ax[0].set_title(title+"graph")
    ax[1].set_title(title+"hypothsis")
    ax[2].set_title(title+"histogram")
    x=range(0, len(distr))

    ax[0].plot(x, distr, c="blue")
    ax[0].set_xlabel('x', fontsize=14)
    ax[0].set_ylabel('p', fontsize=14)

    ax[1].hist(distr, bins=10, density=True, color="blue", alpha=0.4)
    ax[1].plot([0,1], [1,1], color ="red")
    n_maxs=[]
    for i in range(len(distr)):
        y=log_distr(0,1,12)
        n_max=max(y)
        n_maxs.append(n_max)
    ax[2].hist(n_maxs, bins=100, density=True, coloe="blue", alpha=0.4)
    plt.show()

def four_c():
    distr=couchi_distr(1,0)
    title="Cauchi distr"
    mean(distr)
    fig, ax=plt.subplots(1,3,figsize=(16,12))
    ax[0].grid()
    ax[1].grid()
    ax[2].grid()
    ax[0].set_title(title+"graph")
    ax[1].set_title(title+"hypothsis")
    ax[2].set_title(title+"histogram")
    x=range(0, len(distr))

    ax[0].plot(x, distr, c="green")
    ax[0].set_xlabel('x', fontsize=14)
    ax[0].set_ylabel('p', fontsize=14)

    ax[1].hist(distr, bins=10, density=True, color="green", alpha=0.4)
    ax[1].plot([0,1], [1,1], color ="red")
    n_maxs=[]
    for i in range(len(distr)):
        y=couchi_distr(1,0)
```

```
        n_max=max(y)
        n_maxs.append(n_max)
    ax[2].hist(n_maxs, bins=100, density=True, coloe="green", alpha=0.4)
    plt.show()

def four_r():
    distr=rayleigh_distr(1)
    title="Rayleigh distr"
    rayleigh_mean(1)
    fig, ax=plt.subplots(1,3,figsize=(16,12))
    ax[0].grid()
    ax[1].grid()
    ax[2].grid()
    ax[0].set_title(title+"graph")
    ax[1].set_title(title+"hypothsis")
    ax[2].set_title(title+"histogram")
    x=range(0, len(distr))

    ax[0].plot(x, distr, c="pink")
    ax[0].set_xlabel('x', fontsize=14)
    ax[0].set_ylabel('p', fontsize=14)

    ax[1].hist(distr, bins=10, density=True, color="pink", alpha=0.4)
    ax[1].plot([0,1], [1,1], color ="blue")
    n_maxs=[]
    for i in range(len(distr)):
        y=rayleigh_distr(0,1)
        n_max=max(y)
        n_maxs.append(n_max)
    ax[2].hist(n_maxs, bins=100, density=True, coloe="pink", alpha=0.4)
    plt.show()

def four_w():
    distr=weibull_distr(3,2)
    title="Weibull distr"
    weibull_mean(3,2)
    fig, ax=plt.subplots(1,3,figsize=(16,12))
    ax[0].grid()
    ax[1].grid()
    ax[2].grid()
    ax[0].set_title(title+"graph")
    ax[1].set_title(title+"hypothsis")
    ax[2].set_title(title+"histogram")
    x=range(0, len(distr))
```

```
    ax[0].plot(x, distr, c="aqua")
    ax[0].set_xlabel('x', fontsize=14)
    ax[0].set_ylabel('p', fontsize=14)

    ax[1].hist(distr, bins=10, density=True, color="aqua", alpha=0.4)
    ax[1].plot([0,1], [1,1], color ="red")
    n_maxs=[]
    for i in range(len(distr)):
        y=weibull_distr(3,2)
        n_max=max(y)
        n_maxs.append(n_max)
    ax[2].hist(n_maxs, bins=100, density=True, coloe="aqua", alpha=0.4)
    plt.show()

four_c()
four_l()
four_r()
four_w()
```

Завдання 4

4. Обчислити методом Монте-Карло визначені інтеграли

$$\int_0^1 (x^7 + x^5 + x^3)dx, \quad \int_0^\pi 2\sin(3x)\,dx, \quad \int_0^\infty \frac{1}{(x+1)\sqrt{x}}dx.$$

Моделювання проводити за алгоритмом:

4.1. Змоделювати послідовність з n =1000000 псевдовипадкових чисел, що рівномірно розподілені на інтервалі інтегрування.

4.2. Обчислити інтеграл за формулою:

$$I = \frac{b-a}{n}\sum_{i=1}^{n} f(\xi_i),$$

де a, b - межі інтегрування.

Пояснити отримані результати.

```
f(x) = x^7 + x^5 + x^3, a = 0, b = 1, n = 1000000, RESULT:  0.5419268306669546
f(x) = 2sin(3x), a = 0, b = pi, n = 1000000, RESULT:  1.3324444892706928
f(x) = 1/((x+1)*sqrt(x)), a = 0, b = 10000, n = 10000000, RESULT:  3.453272214675623
```

Код програми:

```
def f1(x):
    return x**7+x**5+x**3

def f2(x):
    return 2*np.sin(3*x)
```

```python
def f3(x):
    return 1/((x+1)*np.sqrt(x))

def monte_carlo(func, a,b,n):
    sum=0
    for i in range (n):
        x=np.random.uniform(a,b)
        sum+=func(x)
    res=(b-a)/n*sum
    return res

print("f(x) = x^7 + x^5 + x^3, a = 0, b = 1, n = 1000000, RESULT: ", monte_carlo(f1, 0, 1, 1000000))
print("f(x) = 2sin(3x), a = 0, b = pi, n = 1000000, RESULT: ", monte_carlo(f2, 0, np.pi, 1000000))
print("f(x) = 1/((x+1)*sqrt(x)), a = 0, b = 10000, n = 10000000, RESULT: ", monte_carlo(f3, 0, 10000, 1000000))
```

Завдання 5

5. Змоделювати випадкову двійкову послідовність за формулою

$$\beta_i = \begin{cases} 1, & \xi_{i+1} - \xi_i > 0 \\ 0, & \xi_{i+1} - \xi_i \le 0 \end{cases}.$$

$\{\xi_i, i = 1,\ldots, n\}$ - нормальнорозподілені випадкові величини з параметрами $N(0,1)$.

Знайти емпіричний закон розподілу двійкової послідовності.

Двійкова послідовність: [1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1,

1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0,
1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1,
0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0,
0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1,
0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,
1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0,
1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1,
1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0,
0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1,
0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0]

```
Емпіричний закон розподілу:
Xi:    0    ,    1
Pi: 0.4965  , 0.5035
```

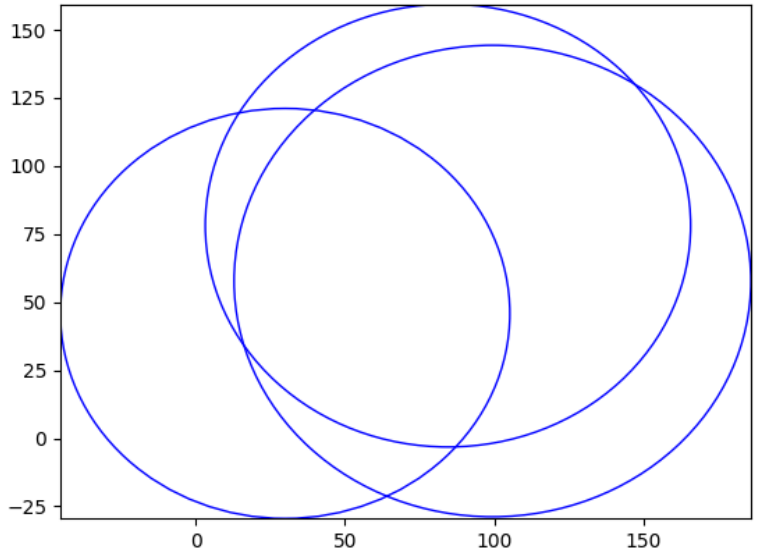
Код програми:

```python
psi=np.random.normal(0,1,1000)
beta=[]
for i in range(len(psi)-1):
    if psi[i+1]-psi[i]>0:
        beta.append(1)
    else:
        beta.append(0)
print("Двійкова послідовність: "+ str(beta))
num0=0
num1=0
for i in range(len(beta)):
    if beta[i]==0:
        num0+=1
    else:
        num1+=1
print("Емпіричний закон розподілу:")
print(f"Xi:{'0':^9},{'1':^9}")
print(f"Pi:{round(num0/len(beta), 5):^9},{round(num1/len(beta), 5):^9}")
```

Завдання 6

6. Задано N кіл (відомі координати центра та радіус). Перевірити, чи мають ці кола спільні точки. Знайти площу фігури, що утворилась в результаті їх перетину.

```
Введіть кількість кіл: 3
Перетин є
необхідна зона Є
8533.807789723936
```



Код програми:

```
def distribution(x1,y1, x2,y2):
    return math.sqrt ((x2-x1)**2+(y2-y1)**2)
def generator(n=int(input("Введіть кількість кіл: "))):
    circle=[]
    for i in range(n):
        circle.append(np.random.uniform(0,100,3))
    return circle
def c_intersection(circle):
    inter=1
    for i in range(len(circle)):
        for j in range(len(circle)):
            if distribution(circle[i][0], circle[i][1],circle[j][0],
circle[j][1])>=(circle[i][2]+circle[j][2]):
                inter=0
    if inter:
        print("Перетин є")
    else:
```

```
        print("Перетину немає")
    return inter

def area(circle):
    area_coord=[circle[0][0]-circle[0][2], circle[0][1]-circle[0][2],
circle[0][0]+circle[0][2], circle[0][1]+circle[0][2]]
    for i in range(len(circle)):
        if (circle[i][0]-circle[i][2]) < area_coord[0]:
            area_coord[0] = circle[i][0]-circle[i][2]

        if (circle[i][0]+circle[i][2]) > area_coord[2]:
            area_coord[2] = circle[i][0]+circle[i][2]

        if (circle[i][1]-circle[i][2]) < area_coord[1]:
            area_coord[1] = circle[i][1]-circle[i][2]

        if (circle[i][1]+circle[i][2]) > area_coord[3]:
            area_coord[3] = circle[i][1]+circle[i][2]
    print('необхідна зона Є')
    return area_coord

def hit(circle, dot):
    hit=1
    for i in range(len(circle)):
        if distribution(circle[i][0], circle[i][1], dot[0],
dot[1])>=circle[i][2]:
            hit=0
    return hit

def intersection(circle, area):
    s=(area[2]-area[0])*(area[3]-area[1])
    a=int((s*10)//1)
    b=0
    dots_x=np.random.uniform(area[0], area[2], a)
    dots_y=np.random.uniform(area[1], area[3], a)
    for i in range(a):
        if hit(circle, [dots_x[i], dots_y[i]]):
            b+=1
    return (s*b)/a

def draw (circle, area):
    fig, ax=plt.subplots()
    ax.set_xlim((area[0], area[2]))
    ax.set_ylim((area[1], area[3]))
    for i in range(len(circle)):
```

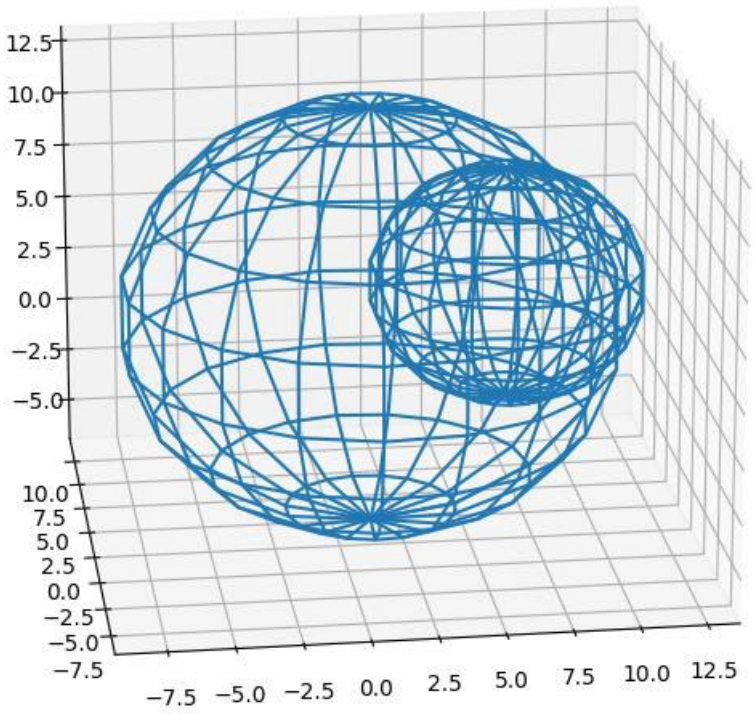
```
        c=plt.Circle((circle[i][0], circle[i][1]), circle[i][2],
color="blue", fill=False)
        ax.add_artist(c)
    plt.show()
    return


circle=generator()
if c_intersection(circle):
    zone=area(circle)
    print(intersection(circle, zone))
    draw(circle, zone)
else:
    print("Перетину немає")
```

Завдання 7

7. Задано N сфер (відомі координати центра та радіус). Перевірити, чи мають ці сфери спільні точки. Знайти об'єм фігури, що утворилась в результаті їх перетину.

Сфери мають спільну зону.

Об'єм фігури: 5.848



Код програми:

```
plt.rcParams.update({'figure.max_open_warning': 1})

def distance(x1,y1,z1,x2,y2,z2):
    return math.sqrt((x2-x1)**2 + (y2-y1)**2 + (z2-z1)**2)

def sphere_generator(N = int(input('К-сть сфер: '))):
    spheres = []
    for i in range(N):
        spheres.append(np.random.uniform(0,100,4))
    return spheres


def check_intersection (spheres):
    intersection = 1
    for i in range(len(spheres)):
        for j in range(len(spheres)):
            if distance(spheres[i][0], spheres[i][1], spheres[i][2],
spheres[j][0], spheres[j][1], spheres[j][2]) >= (spheres[i][3] +
spheres[j][3]):
                intersection = 0
    if intersection:
        print('Перетин є')
    else:
        print('Перетину немає')
    return intersection
def search_zone (spheres):
    zone_coord = [spheres[0][0]-spheres[0][3], spheres[0][1]-
spheres[0][3], spheres[0][0]+spheres[0][3], spheres[0][1]+spheres[0][3],
spheres[0][2]-spheres[0][3], spheres[0][2]+spheres[0][3]]
    for i in range(len(spheres)):
        if (spheres[i][0]-spheres[i][3]) < zone_coord[0]:
            zone_coord[0] = spheres[i][0]-spheres[i][3]
        if (spheres[i][0]+spheres[i][3]) > zone_coord[2]:
            zone_coord[2] = spheres[i][0]+spheres[i][3]
        if (spheres[i][1]-spheres[i][3]) < zone_coord[1]:
            zone_coord[1] = spheres[i][1]-spheres[i][3]
        if (spheres[i][1]+spheres[i][3]) > zone_coord[3]:
            zone_coord[3] = spheres[i][1]+spheres[i][3]
        if (spheres[i][2]+spheres[i][3]) < zone_coord[4]:
            zone_coord[4] = spheres[i][2]-spheres[i][3]
        if (spheres[i][2]+spheres[i][3]) > zone_coord[5]:
             zone_coord[5] = spheres[i][2]+spheres[i][3]
             print('необхідна ділянка Є')
```

```
    return zone_coord

def hit (spheres, dot):
    hit = 1
    for i in range(len(spheres)):
        if distance(spheres[i][0], spheres[i][1], spheres[i][2], dot[0],
dot[1], dot[2]) >= spheres[i][2]:
            hit = 0
    return hit
def intersection (spheres, zone):
    S = (zone[2]-zone[0])*(zone[3]-zone[1])*(zone[5]-zone[4])
    A = int((S*1000)//1)
    B = 0
    print(spheres, zone)
    print(S, A)
    dots_x = np.random.uniform(zone[0], zone[2], A)
    dots_y = np.random.uniform(zone[1], zone[3], A)
    dots_z = np.random.uniform(zone[4], zone[5], A)
    for i in range(A):
        if hit(spheres, [dots_x[i], dots_y[i], dots_z[i]]):
            B += 1
    print(B)
    return (S*B)/A
def draw (spheres, zone):
    u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
    x = spheres[0][3]*np.cos(u)*np.sin(v)+spheres[0][0]
    y = spheres[0][3]*np.sin(u)*np.sin(v)+spheres[0][1]
    z = spheres[0][3]*np.cos(v)+spheres[0][2]
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_wireframe(x, y, z)
    for i in range(1, len(spheres)):
        x = spheres[i][3]*np.cos(u)*np.sin(v)+spheres[i][0]
        y = spheres[i][3]*np.sin(u)*np.sin(v)+spheres[i][1]
        z = spheres[i][3]*np.cos(v)+spheres[i][2]
        ax.plot_wireframe(x, y, z)
        plt.show()
        return

spheres = sphere_generator()
if check_intersection(spheres):
    print(intersection(spheres, search_zone(spheres)))
    draw(spheres, search_zone(spheres))
else:
    print('Перетину немає')
```