Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра прикладної математики

Лабораторна робота №3
з кредитного модуля «Випадкові процеси»
Варіант 8

Виконав:
студент групи КМ-81
Донченко Богдан Миколайович
Перевірив викладач:
Пашко Анатолій Олексійович

Київ – 2020

Завдання 1

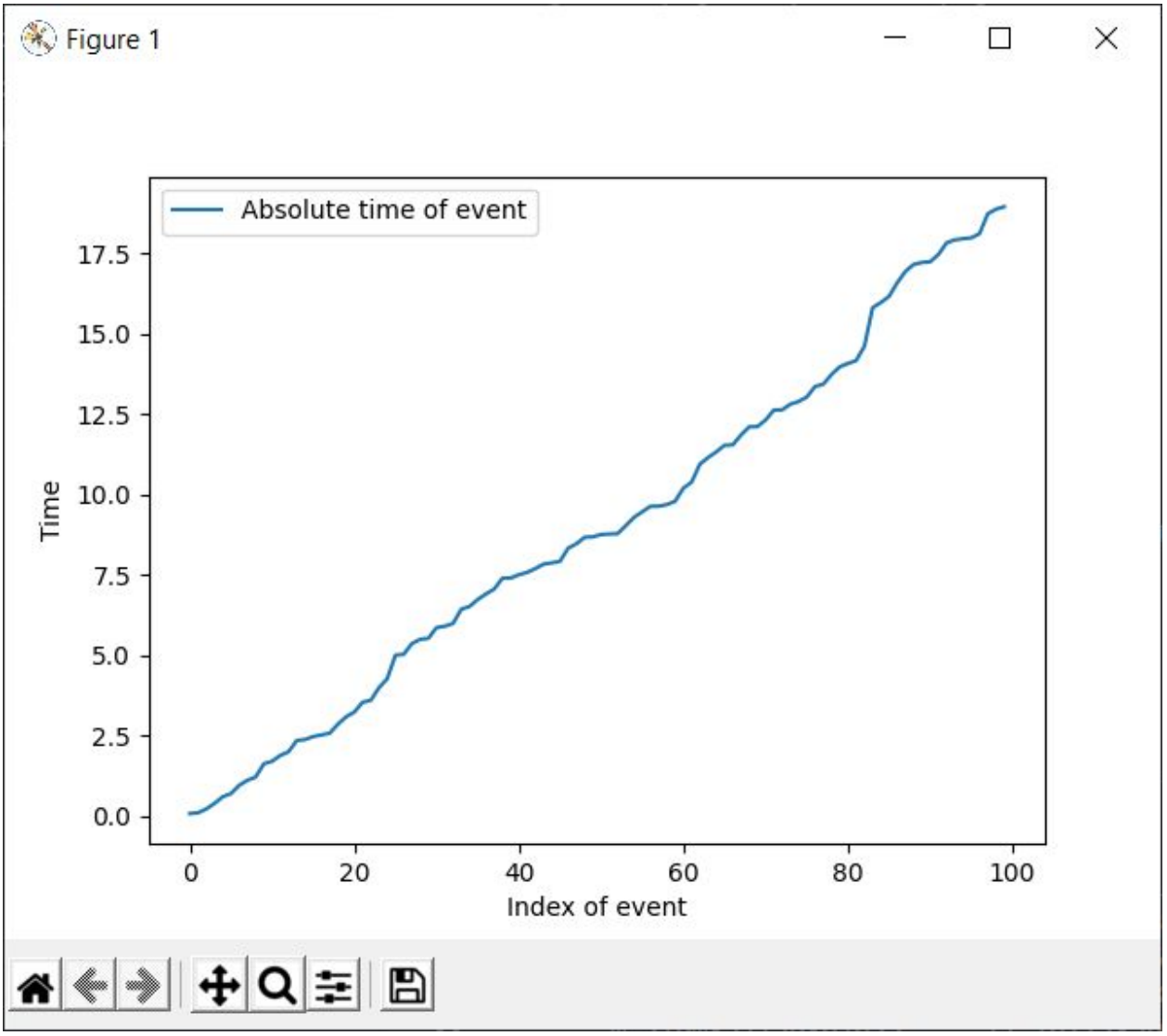
Умова(1)

1. Змоделювати Пуассонівський потік з заданою інтенсивністю. Побудувати графіки реалізацій процесу.

Побудувати гістограми розподілів:

- часу появи заданої події (перша, друга, n - та);
- інтервалу між подіями;
- появи рівно n - подій.

Графіки та таблиці(1)



EVENT_NUM	INTER_EVENT_T	EVENT_T
0	0.07716955300605029	0.07716955300605029
1	0.022013610619495173	0.09918316362554547
2	0.12015466255614302	0.21933782618168848
3	0.17053192094394048	0.38986974712562894
4	0.2097284423248785	0.5995981894505075
5	0.09681138618670906	0.6964095756372165
6	0.25778348801612616	0.9541930636533427
7	0.1573876720405912	1.1115807356939338
8	0.10081016357966918	1.212390899273603
9	0.41154507373891985	1.623935973012523
10	0.07731287428029324	1.7012488472928162
11	0.18335788800301847	1.8846067352958347
12	0.11338112451877025	1.997987859814605
13	0.3520762372260572	2.3500640970406623
14	0.028454998644468467	2.3785190956851308
15	0.09179968705559274	2.4703187827407236
16	0.05309159700585838	2.523410379746582
17	0.05023207546684026	2.573642455213422
18	0.28174471864099015	2.855387173854412
19	0.22989486066725343	3.0852820345216654
20	0.14939224258722855	3.234674277108894
21	0.30190496706816095	3.536579244177055
22	0.06337488835834632	3.5999541325354016
23	0.3916750815121112	3.9916292140475127
24	0.28055858942477163	4.272187803472284
25	0.7296051639677874	5.001792967440071
26	0.028733147061970082	5.030526114502042
27	0.3342602713097861	5.364786385811827
28	0.1322439629011312	5.497030348712959
29	0.030084109082031844	5.52711445779499
30	0.33630331920420414	5.863417776999195
31	0.040019080975921086	5.903436857975116
32	0.08701876842936804	5.990455626404484
33	0.4442779776049073	6.434733604009391
34	0.08293212967957422	6.517665733688966
35	0.21729059776596826	6.734956331454934
36	0.17214050972599812	6.907096841180932
37	0.15175295435511638	7.058849795536048
38	0.33954860363436484	7.3983983991704125
39	0.0036781789499549834	7.402076578120368
40	0.10484453010018369	7.506921108220551
41	0.07099719164448799	7.577918299865039
42	0.11776636662501534	7.695684666490054
43	0.1394776207544099	7.835162287244464
44	0.03938891654178714	7.874551203786251
45	0.04996352402397926	7.92451472781023
46	0.40975102468897606	8.334265752499206
47	0.1348279691301632	8.469093721629369
48	0.20498679472638578	8.674080516355755
49	0.00982527233933985	8.683905788695094
50	0.07409695309548087	8.758002741790575
51	0.01355269716151056	8.771555438952086
52	0.012108749730184065	8.78366418868227
53	0.24857940827624952	9.03224359695852
54	0.25804962943296095	9.29029322639148
55	0.17450568532860752	9.464798911720088
56	0.1705250589888226	9.63532397070891
57	0.0029066552849843744	9.638230625993895
58	0.05009036672819324	9.688320992722089
59	0.1050215414472371	9.793342534169327
60	0.4058550730856577	10.199197607254984
61	0.1867004789238233	10.385898086178807
62	0.5577152933087515	10.943613379487559
63	0.20797799647806997	11.151591375965628
64	0.1761885363773455	11.327779912342974
65	0.2051302329104984	11.532910145253473
66	0.012279773174835733	11.545189918428308
67	0.3060545279205827	11.851244446348892
68	0.25507693989873903	12.10632138624763
69	0.008156612940809313	12.11447799918844
70	0.20706835103921017	12.32154635022765
71	0.3076585420255439	12.629204892253194
72	0.005713834457513008	12.634918726710707
73	0.17700438776315935	12.811923114473865
74	0.08079407490440589	12.89271718937827
75	0.14115938059695884	13.033876569975229
76	0.32707099207475826	13.360947562049986
77	0.07143363417156601	13.432381196221552
78	0.2996822211941808	13.732063417415732
79	0.2420849987973471	13.97414841621308
80	0.10415805382757652	14.078306470040657
81	0.08856049723424188	14.166866967274899
82	0.44407458507419684	14.610941552349097
83	1.2000048065043007	15.810946358853398
84	0.1613865583940253	15.972332917247423
85	0.19228262269992627	16.16461553994735
86	0.42804752138591906	16.59266306133327
87	0.3494248227179525	16.942087884051222
88	0.2190357159367625	17.161123599987985
89	0.0591860749363833	17.22030967492437
90	0.021161346454666264	17.241471021379034
91	0.22539192657616378	17.4668629479552
92	0.3649196680089324	17.831782615964133
93	0.09144736882162113	17.923229984785756
94	0.03655756233949823	17.959787547125256
95	0.02594929439819669	17.98573684152345
96	0.13160173986224433	18.117338581385695
97	0.616786121398677	18.73412470278437
98	0.14382233845754164	18.877947041241914
99	0.07660753262198129	18.954554573863895

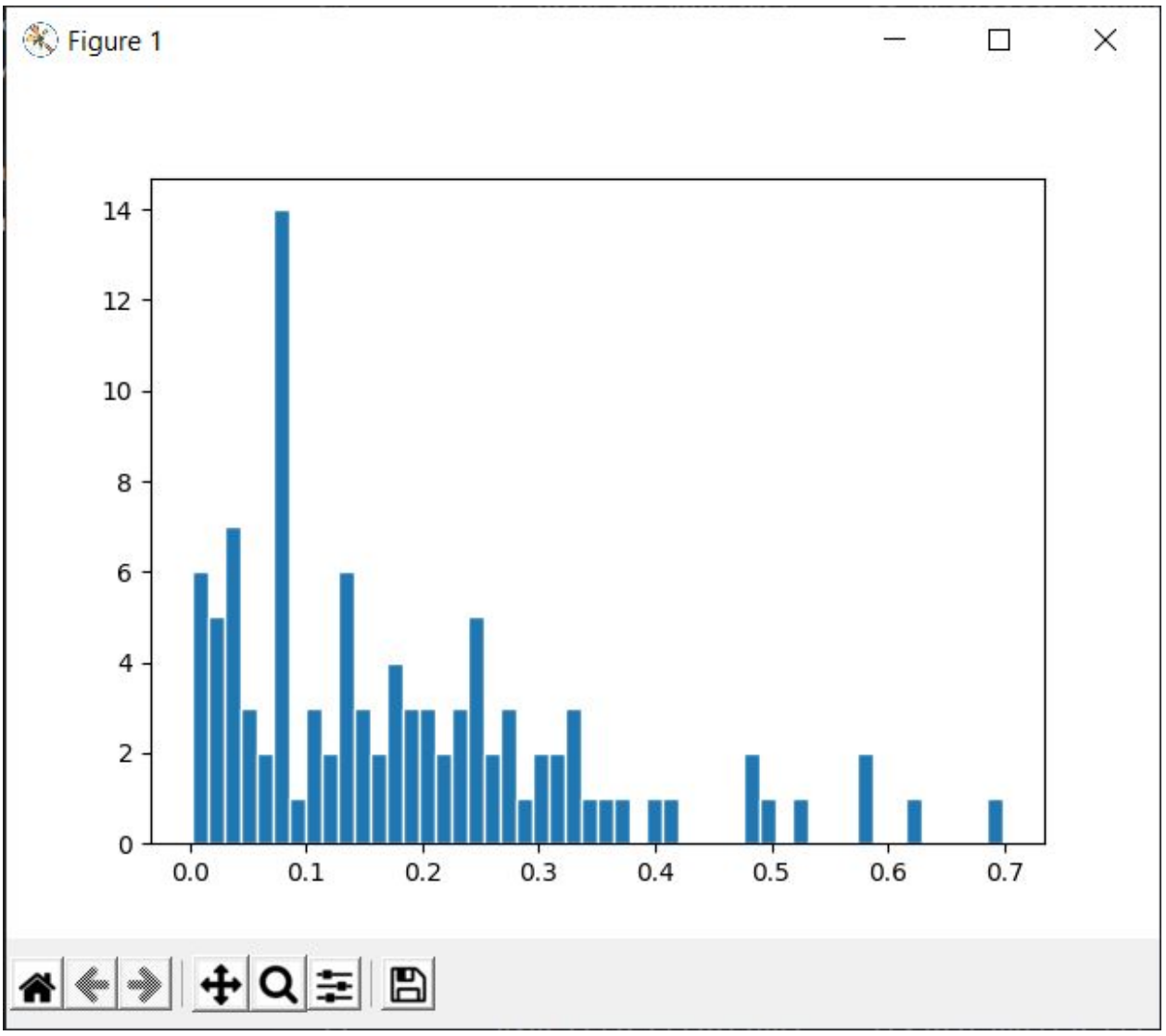
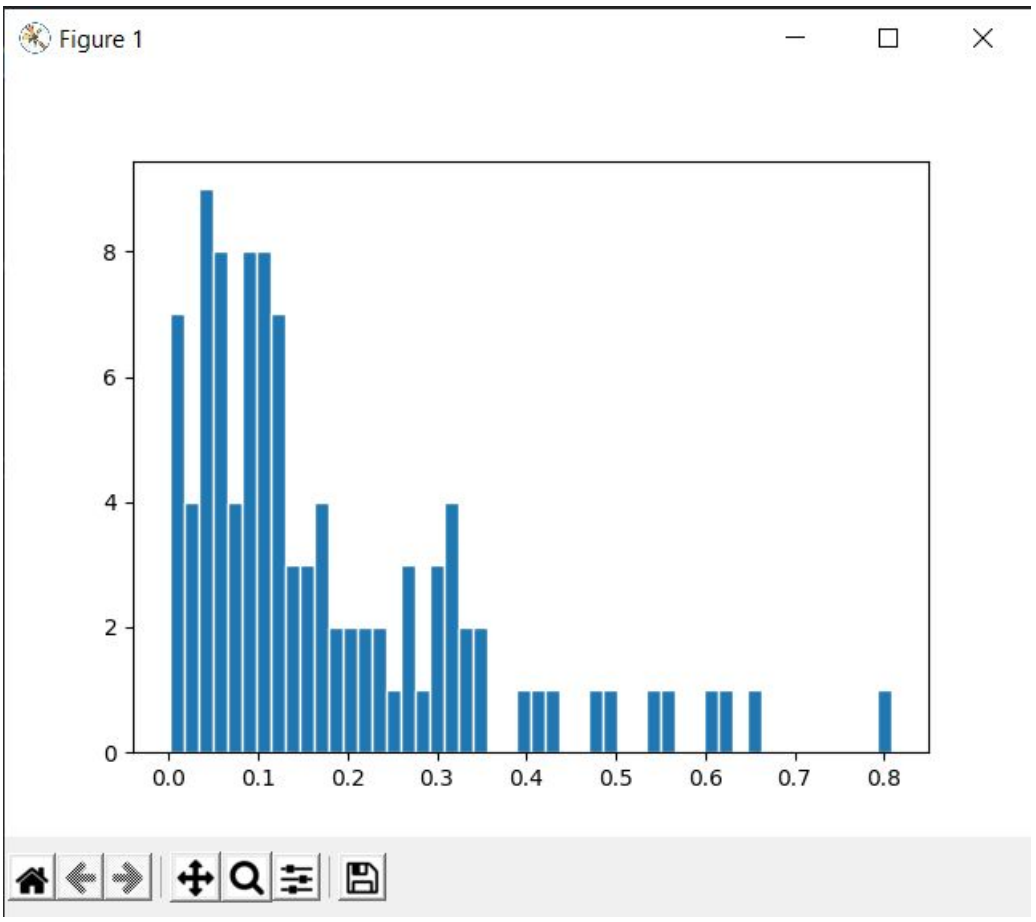


Figure 1
14
12
10
8
6
4
2
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7



INTERVAL_NUM	NUM_EVENTS
1	7
2	6
3	6
4	5
5	1
6	8
7	4
8	9
9	7
10	7
11	3
12	5
13	7
14	5
15	3
16	2
17	3
18	8

Mean: 5.333333333333333
Var: 4.888888888888889

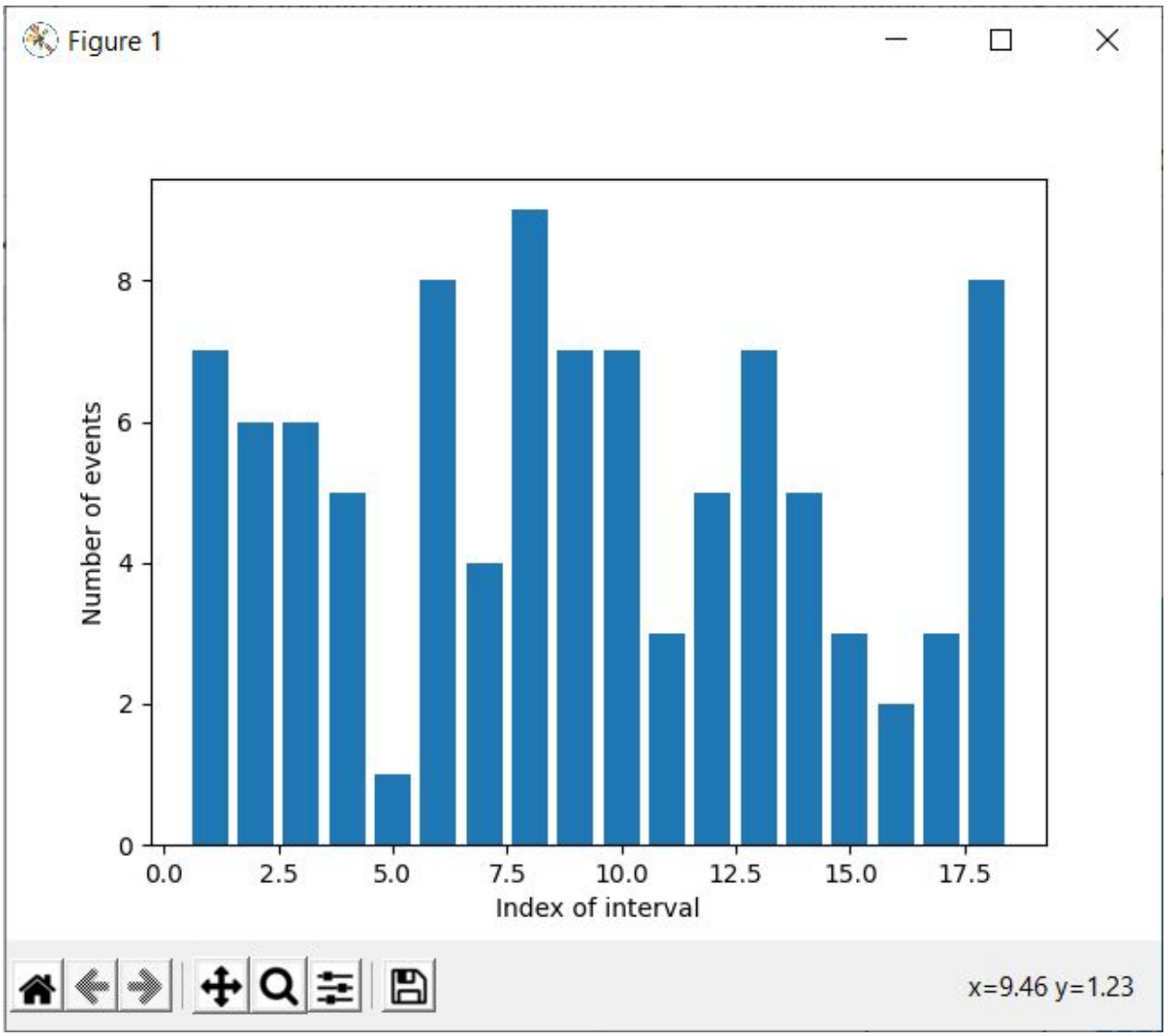


Figure 1
Number of events
0
2
4
6
8
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Index of interval
x=9.46 y=1.23

Код(1)

```
import numpy as np

import matplotlib.pyplot as plt

from tabulate import tabulate


lamb = 5

num_events = 100

event_num = []

inter_event_times = []

event_times = []

event_time = 0



def plot_graph(a, b, name):

    fig = plt.figure()

    plot, = plt.plot(a, b, label=name)

    plt.legend(handles=[plot])

    plt.xlabel('Index of event')

    plt.ylabel('Time')

    plt.show()


def bar_graph(a, b):

    fig = plt.figure()

    plt.bar(a, b)

    plt.xlabel('Index of interval')

    plt.ylabel('Number of events')

    plt.show()
```

```
def s_formula(lamb, size):
    array = np.random.rand(size)
    result = -np.log(1.0 - array) / lamb
    return result

header_1 = ["EVENT_NUM", "INTER_EVENT_T", "EVENT_T"]
table_1 = [header_1]

for i in range(num_events):
    event_num.append(i)
    n = np.random.rand()
    inter_event_time = -np.log(1.0 - n) / lamb
    inter_event_times.append(inter_event_time)
    event_time = event_time + inter_event_time
    event_times.append(event_time)
    table_1.append([str(i), str(inter_event_time), str(event_time)])
print(tabulate(table_1))

plot_graph(event_num, event_times, 'Absolute time of event')

hist = np.array([np.sum(s_formula(lamb, 1)) for i in range(num_events)])
plt.hist(hist, 50, edgecolor = "white")
plt.show()

plt.hist(s_formula(lamb, num_events), 50, edgecolor = "white")
plt.show()

interval_nums = []
```

```
num_events_in_interval = []

interval_num = 1

num_events = 0




header_2 = ['INTERVAL_NUM', 'NUM_EVENTS']

table_2 = [header_2]

for i in range(len(event_times)):

    event_time = event_times[i]

    if event_time <= interval_num:

        num_events += 1

    else:

        interval_nums.append(interval_num)

        num_events_in_interval.append(num_events)

        table_2.append([str(interval_num), str(num_events)])

        interval_num += 1

        num_events = 1


print(tabulate(table_2))

print("Mean: ",np.mean(num_events_in_interval))

print("Var: ",np.var(num_events_in_interval))


bar_graph(interval_nums, num_events_in_interval)
```

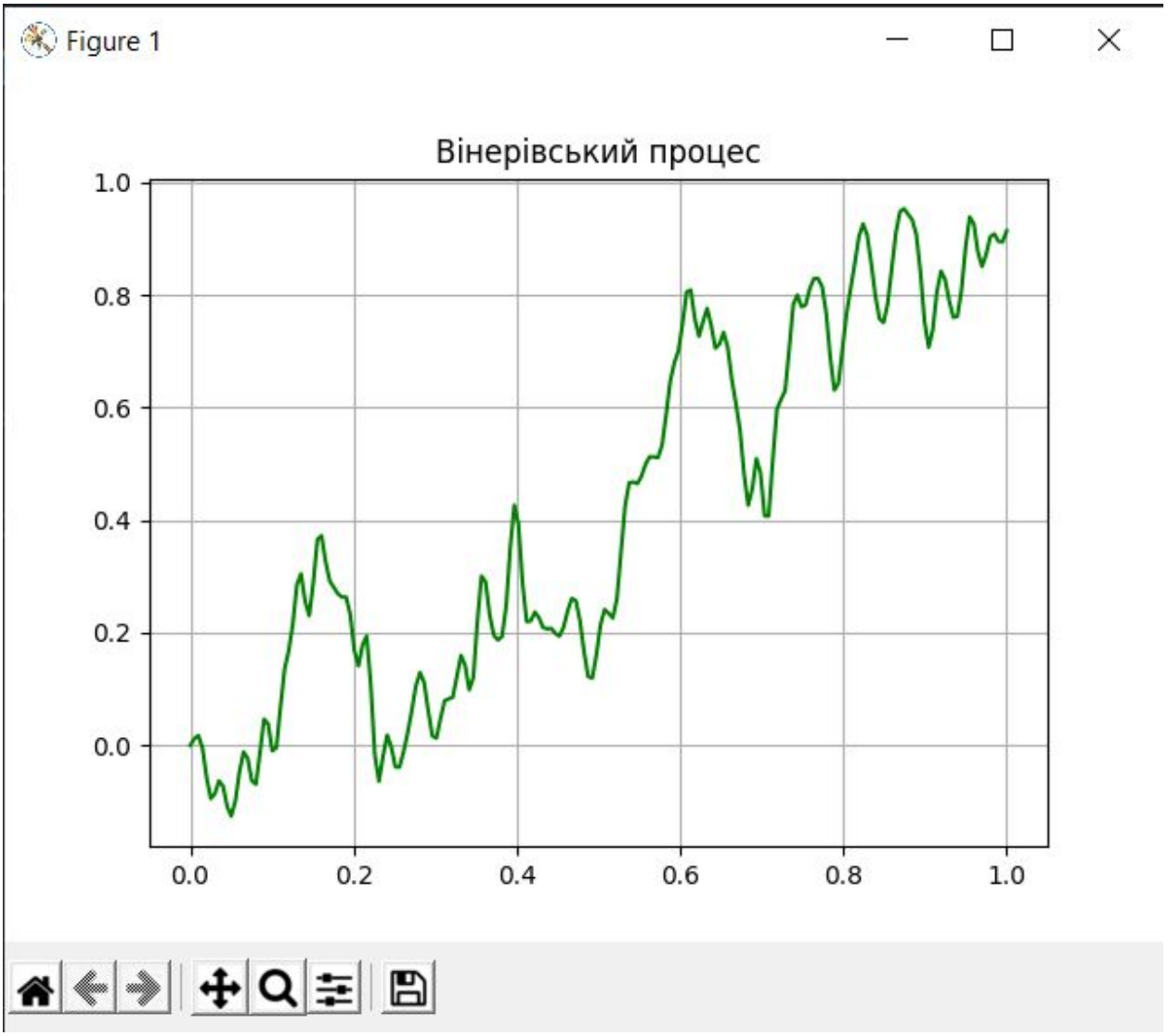
Завдання 2

Умова(2)

Завдання 2.

1. Змоделювати неперервний вінерівський випадковий процес.
2. За реалізаціями (кількість реалізацій > 100) оцінити середнє значення та дисперсію.
3. Знайти емпіричний закон розподілу ймовірностей часу першого виходу вінерівського процесу за заданий рівень.

Графіки та таблиці(2)



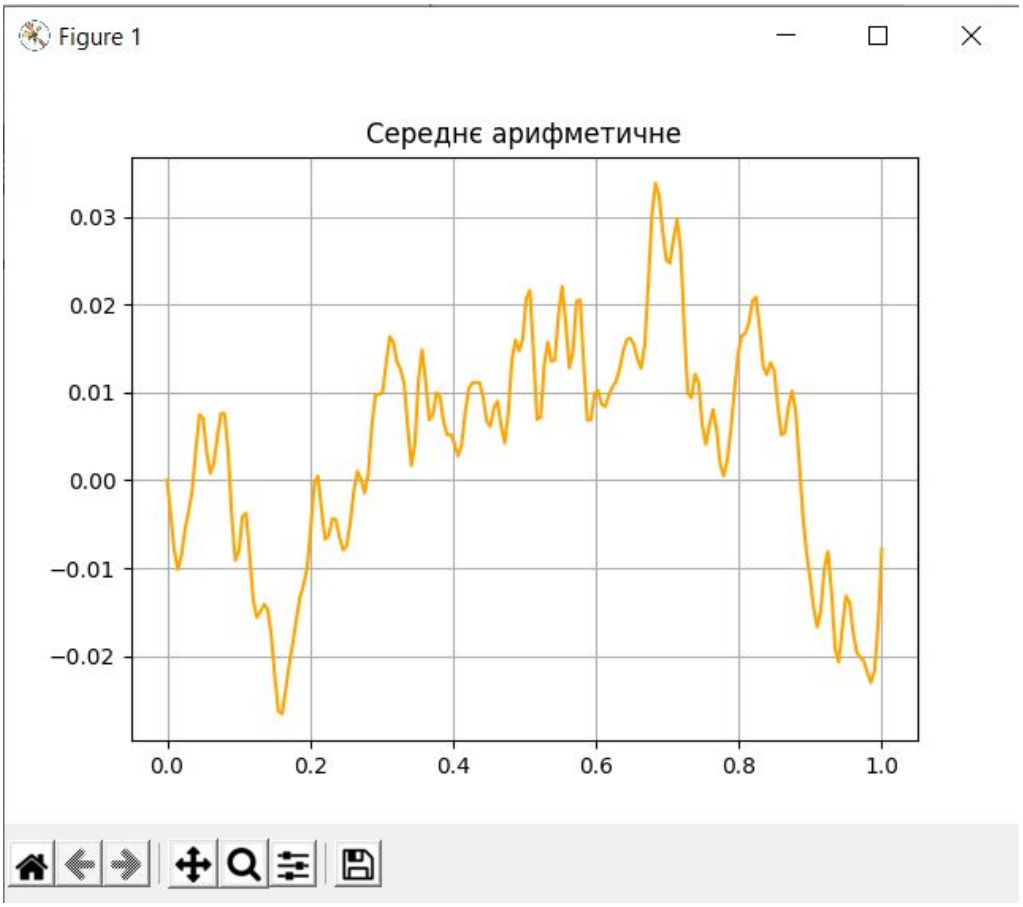


Figure 1
Середнє арифметичне
0.03
0.02
0.01
0.00
−0.01
−0.02
0.0
0.2
0.4
0.6
0.8
1.0

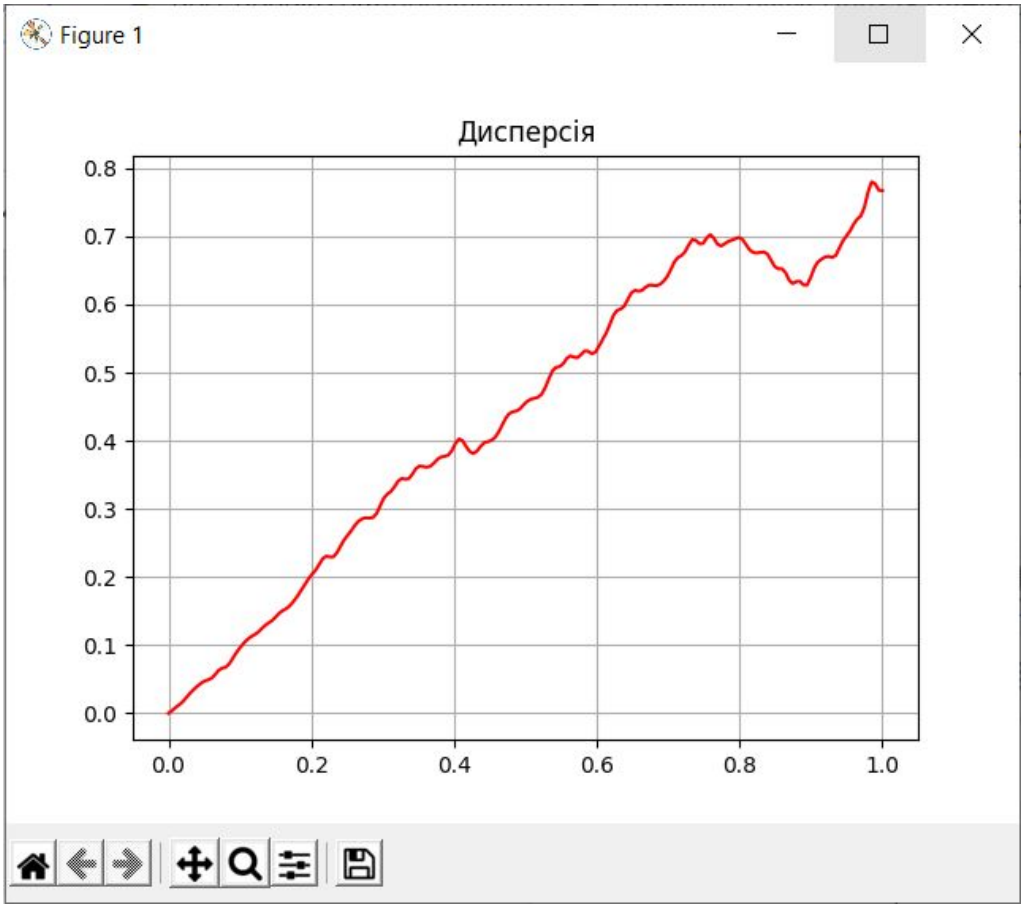


Figure 1
Дисперсія
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
1.0

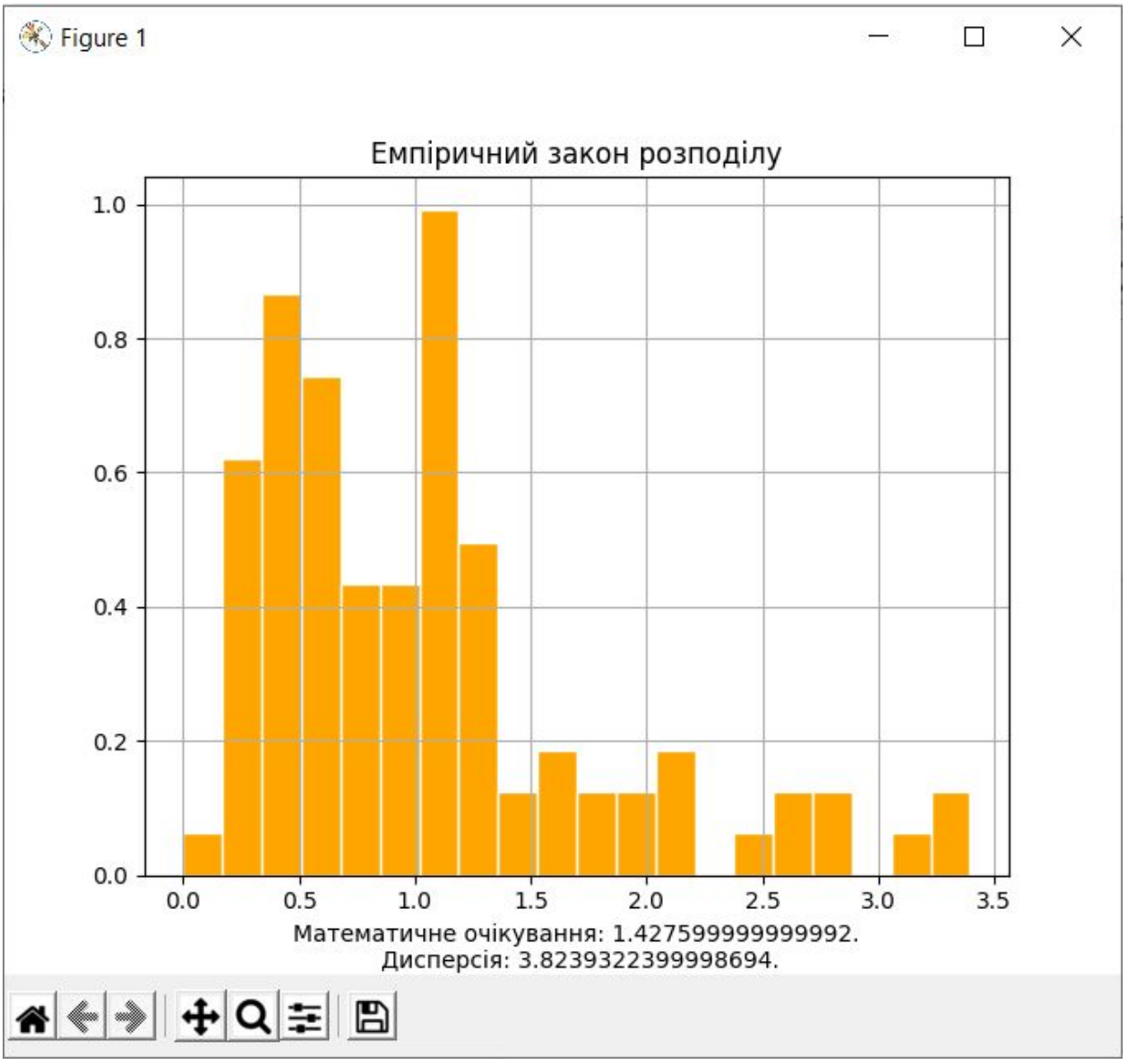


Figure 1
Емпіричний закон розподілу
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Математичне очікування: 1.427599999999992.
Дисперсія: 3.8239322399998694.

Код (2)

```
import numpy as np

import matplotlib.pyplot as plt




def formula(size = 100):

    n = np.random.standard_normal(size+1)

    result = lambda t: t*n[0] + np.sqrt(2)*np.sum([np.sin(i * np.pi *
t)/(i * np.pi)*n[i] for i in range(1, size+1)], axis=0)

    return result


def graph_plot(x, y, name, color):

    plt.grid() == True

    plt.title(name)

    plt.plot(x, y, color=color)

    plt.show()


def graph_hist(k, delta, size = 1000):

    hist_arr = np.zeros(N)

    for i in range(N):

        result = formula()

        time = 0

        current = result(0)

        while -k < current < k:

            time += delta

            current = result(time)

        hist_arr[i] = time

    plt.grid() == True
```

```
    plt.xlabel("Математичне очікування: {}.\n Дисперсія:
{}.".format(np.mean(hist_arr), np.var(hist_arr)))

    plt.title("Емпіричний закон розподілу")

    plt.hist(hist_arr, density=True, bins=20, color='orange',
edgecolor="white", range = (0, np.quantile(hist_arr, 0.95)))

    plt.show()


N = 100


result = formula()

x_arr = np.linspace(0, 1, 200)

y_arr = np.array([result(i) for i in x_arr])

graph_plot(x_arr, y_arr, "Вінерівський процес", 'green')


y_mean = np.mean([formula()(x_arr) for i in range(200)], axis = 0)

graph_plot(x_arr, y_mean, "Середнє арифметичне", 'orange')


y_var = np.var([formula()(x_arr) for i in range(200)], axis = 0)

graph_plot(x_arr, y_var, "Дисперсія", 'red')


graph_hist(1, 0.01)
```