МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Інститут прикладної математики та фундаментальних наук

Кафедра прикладної математики

Звіт

Про виконання лабораторної роботи № 1

з курсу “Математична Статистика”

Виконав:

Студент групи ПМ-33

Маркевич Леонід

Перевірив:

Янішевський В.С.

Львів-2023

Лабораторна робота №1
Варіант 12

Завдання:

- 1) розглянути випадкову величину задану формулою (1);
- 2) визначити чисельні характеристики випадкової величини;
- 3) здійснити розігрування випадкової величини (отримати n значень з використанням програмної реалізації);
- 4) побудувати інтервальну таблицю частот;
- 5) побудувати гістограму відносних частот;
- 6) одержати точкові оцінки для математичного очікування й дисперсії;
- 7) апроксимувати гістограму теоретичним нормальним законом розподілу;
- 8) зробити висновки про узгодження теоретичного й статистичного законів розподілів;

$$x_i=\left(\sum_{j=1}^{12} r_j - 6\right)\sigma + a \textbf{(1)}$$

де a= 12 -10= 2, σ = 3+ 12/10 = 4.2;

Хід роботи:
Реалізація виконана на мові програмування Python з використанням бібліотек pandas, numpy, math, matplotlib, scipy

1) розглянути випадкову величину задану формулою (1);

Випадкова величина задана формулою $x_i=\left(\sum_{j=1}^{12} r_j - 6\right) 4.2 + 2$ (що є нормальним розподілом) розподілена на відрізку [-23.2; 27.2]. Математичним сподіванням цієї величини є a = 2, середнє квадратичне відхилення становить 4.2.

Реалізація формули (1) програмно:

```
def x_generation():
    a = V - 10
    sigma = 3 + V / 10
    r = sum(random.uniform(0, 1) for _ in range(sum_limit))
    return (r - 6) * sigma + a
```

2) визначити чисельні характеристики випадкової величини;

M[x] = M[(12*M[r]-6)*4,2+2] = M[(6-6)*4,2+2] = 2

D[x] = 4.2^2*D[$\sum_{j=1}^{12} r_j$] = 4,2^2 = 17,64

3) Здійснити розігрування випадкової величини (отримати n значень з використанням програмної реалізації);

n = 350

Програмна реалізація:

```
def data_collector(n: int):
    data = [x_generation() for _ in range(n)]
    data.sort()
    return data
```

Приклад роботи:

```
7.91835032180548
-9.271823416095513
-8.396314496393137
-7.3647324807094545
-7.010063080638766
-6.398701516822946
-6.0788297045543604
-5.6770507445696365
-5.45428821567903
-5.252052052013244
-5.038473080190457
-4.971201920917949
-4.7649501806118275
-4.5946406582346
-4.540502243990415
-4.534165613253391
-4.486859103435058
-4.453455363299212
-4.388998464741121
-4.310839325924104
-4.182726617402517
-4.079234086121049
-3.9452880105125283
-3.866192502027361
-3.8331459139410535
-3.801822545586968
-3.772769003067916
-3.5529746933460347
-3.545972323458691
-3.5134659487274122
-3.431403027390304
-3.430882582059228
-3.373026960688672
-3.3711105969395243
-3.347942430165449
-3.2637973481280946
-3.199786201007927
-3.199273664160277
-3.175582525810227
-3.0776134671109707
-3.0724597505003732
-3.0505165347057464
-3.0419873391778793
-2.8299666791006626
-2.778442657753004
-2.7777709728132
-2.76284940664715
-2.717861704487471
-2.6557298978910326
-2.5752943629755203
-2.548768068575182
-2.4321849592955624
-2.3548655672452137
```

```
-2.3548655672452137
-2.3185151418543564
-2.2875870714671622
-2.2676621063374407
-2.2672834036474123
-2.2250778491005496
-1.996081057570903
-1.9955012354796828
-1.9447160043492548
-1.9214427608564861
-1.8287466519434847
-1.773330365671753
-1.6721810746366716
-1.645667968575152
-1.6216435567413474
-1.604607950304545
-1.59898206807507
-1.574025737262053
-1.5712149340440416
-1.4764392690310406
-1.4587413546920964
-1.4559962847950243
-1.4024871918507267
-1.3639512132416858
-1.318479772319482
-1.173482058121433
-1.093722833357035
-1.0880196617503906
-0.992828003213301
-0.9753088638943197
-0.9671373635912097
-0.8423198721443459
-0.7195766848861718
-0.6391613779418401
-0.5287169118847777
-0.4178137928799974
-0.40540889995361296
-0.33667482826577677
-0.1799318303272024
-0.1700152245565163
-0.11514397750405037
-0.11404862590790676
0.029129538622751472
0.2130737823116149
0.2470596873473765
0.24851613480041568
0.3142437740894486
0.3199489696984674
0.338894014293319
0.38147294682158095
0.3860317321772264
0.41777365243824693
0.4284494545161184
0.44489180065983724
0.4907989935488828
0.5018872281217912
0.5255559969870431
```

```
0.5577192152686563      6.121818042566386
0.5632077811309826      6.146931684400608
0.5698202034379627      6.205149059389226
0.5703299045704535      6.438980296764238
0.5790019717277108      6.511030877690848
0.5791703422396188      6.543702925825608
0.6207149646233612      6.543870861500633
0.782481820973504       6.646994107334503
0.8017169722738935      6.76833478345057
0.8926901612964044      6.842016271082881
0.9208358847694189      7.009017933298758
0.9209563096493179      7.062075020757632
0.9229365601361845      7.152442107878302
0.9260945846488544      7.228849849766882
0.954701462660237       7.240485256513152
0.9571305650991857      7.249135852635204
0.9734291236228263      7.304662995595085
0.9883596494371558      7.350186631259151
1.025275046422444       7.394659035317295
1.0388850493965736      7.471235011309471
1.0871501856544796      7.512920413520865
1.0886922477808667      7.677438033340459
1.133144580328433       7.7363716398561495
1.1395635867653469      7.764362694487577
1.1464919568686591      7.777656778218718
1.2098316839185697      7.8013862702101635
1.213163657190989       7.802014919228019
1.220285635894184       7.806393163255611
1.2304633429534015      7.871275400041386
1.2380970210534987      7.879770942668705
1.2605499805442384      7.885414346978278
1.2693561594878808      7.947092440354883
1.2743930908430403      7.987273911980995
1.281028259700587       8.006327788072841
1.339090683289848       8.396639231520012
1.4132631798165087      8.405729642421115
1.4368331045866416      8.43463522438089
1.4476180077340817      8.445349360601282
1.457528157258377       8.557067852359712
1.4656868572490396      8.638267948068467
1.5356307701096186      8.894788425812017
1.553413286248853       9.42103657707559
1.6477226146470394      9.461855726807329
1.6816947673102647      9.479260281713621
1.6997690954441504      9.553326540021988
1.6999878673310014      10.144888138752556
1.7509114341883474      10.416363736533244
1.7554525474471707      11.570789552037509
1.7559191985364684      11.84499115886442
1.80556782054197        11.855286582900371
1.815162829639402       11.96387644783465
1.8166953442517575      12.028579332083277
1.8262676690871196      12.375746078508291
1.8640126740072822      12.704015978610697
1.884260697894287       12.70805349161376
1.885379144445642       13.047618688174829
1.8903992331155526      14.312621870090785
```

4) Побудувати інтервальну таблицю частот;

Програмна реалізація:

```
import pandas as pd

from common import data_collector, make_intervals_from_n, make_interval_labels
from const import min_value, max_value

n = 350
data = data_collector(n)

df = pd.DataFrame(data, columns=['Grades'])

bins = make_intervals_from_n(n, minimum=min_value, maximum=max_value, rounder=2)
labels = make_interval_labels(bins)
df['Grade Interval'] = pd.cut(df['Grades'], bins=bins, labels=labels)

frequency_table = df['Grade Interval'].value_counts().sort_index()

print(frequency_table)
```

Приклад роботи:

```
Grade Interval
[-23.2;-18.16]      0
[-18.16;-13.12]     0
[-13.12;-8.08]      4
[-8.08;-3.04]      38
[-3.04;2.0]       128
[2.0;7.04]        137
[7.04;12.08]       38
[12.08;17.12]       5
[17.12;22.16]       0
[22.16;27.2]        0
```

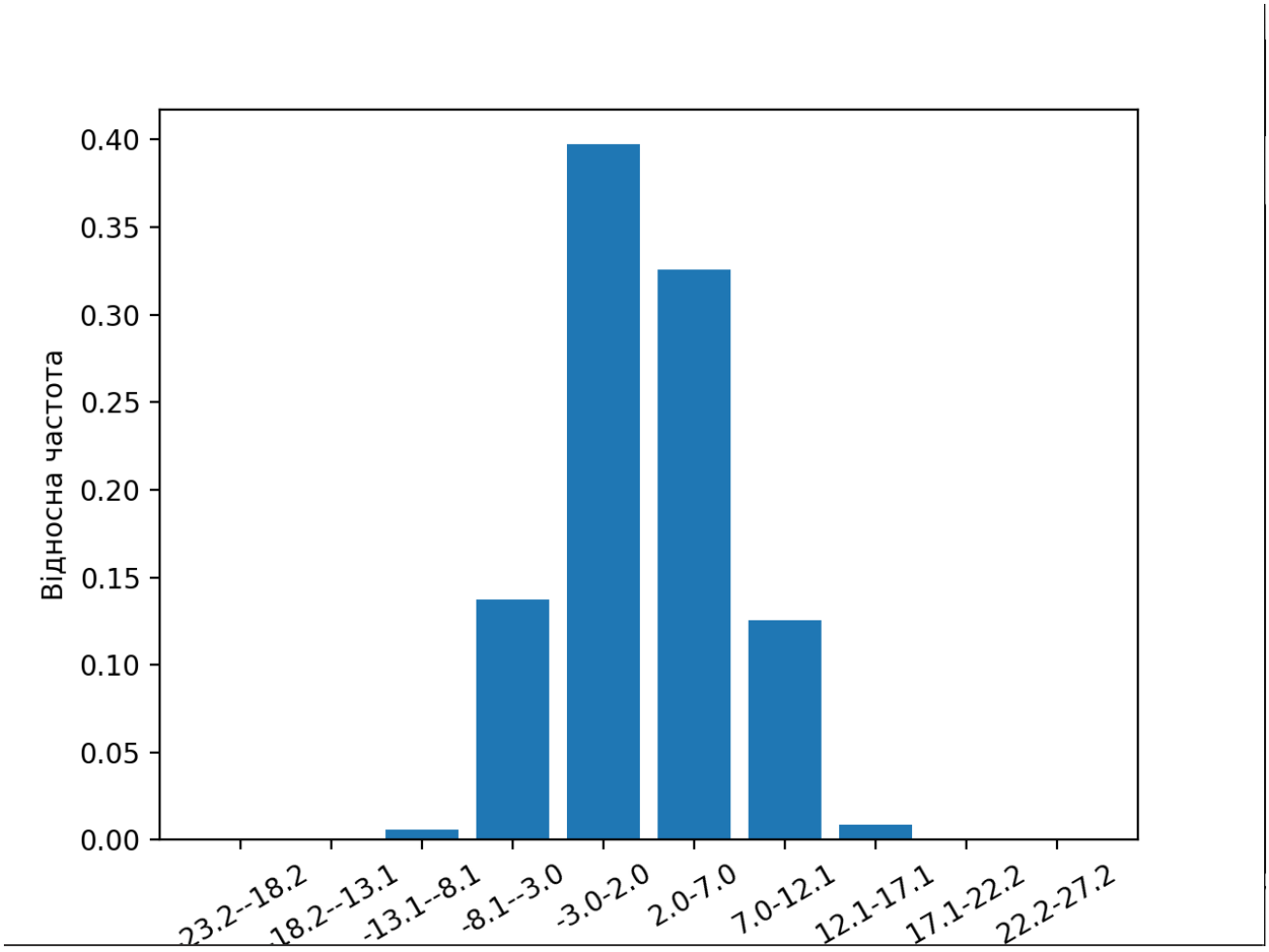
5) Побудувати гістограму відносних частот;

Програмна реалізація:

```
import matplotlib.pyplot as plt
import numpy as np

from common import data_collector, make_intervals_from_n
from const import max_value, min_value

n = 350

data = data_collector(n)

bins = make_intervals_from_n(n, maximum=max_value, minimum=min_value, rounder=1)

hist, _ = np.histogram(data, bins=bins)
relative_frequencies = hist / np.sum(hist)
```

```
plt.bar(range(len(relative_frequencies)), relative_frequencies,
        tick_label=[f"{i}-{j}" for i, j in zip(bins[:-1], bins[1:])])

plt.xlabel('Інтервали')
plt.ylabel('Відносна частота')
plt.xticks(rotation=30)

plt.show()
```

Приклад роботи:



6) Одержати точкові оцінки для математичного очікування й дисперсії;

Програмна реалізація:

```
from math import sqrt
from common import data_collector
n = 350
```

```
data = data_collector(n)
Xv = sum(data) / n
s2 = sum([(data[i] - Xv) ** 2 for i in range(n)]) / n - 1
s = sqrt(s2)
print(f"Xv = {Xv}")
print(f"s = {s}")
```

Приклад роботи:

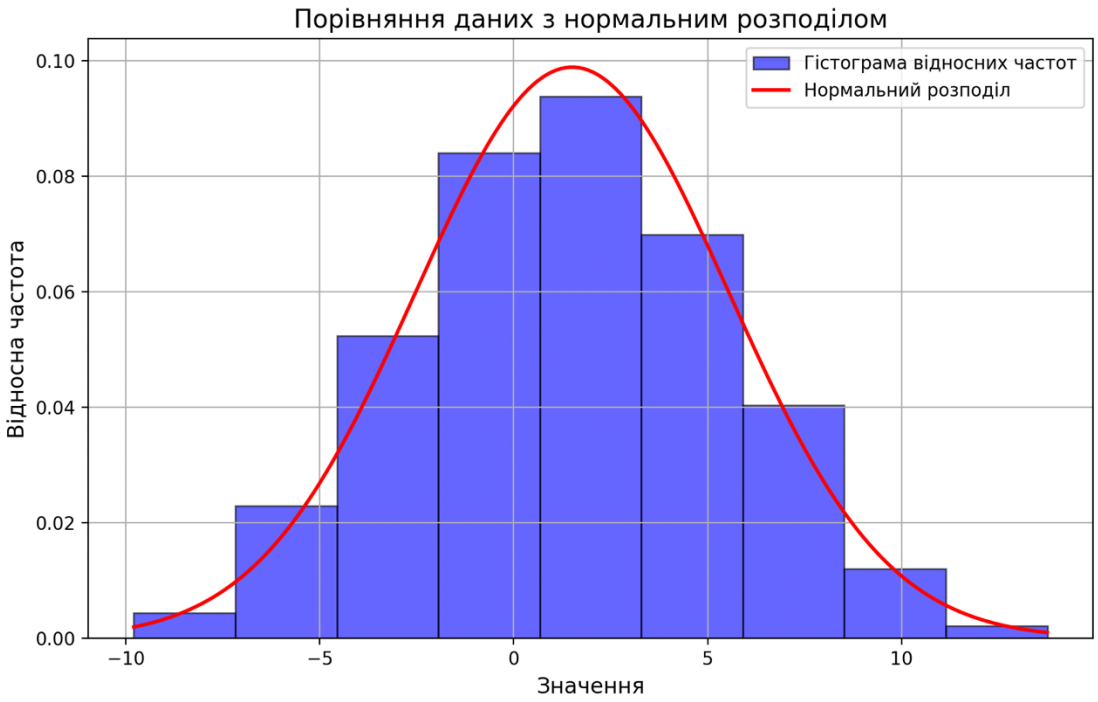
```
Xv = 2.0241787778345937
s = 4.077990053732482
```

7) Апроксимувати гістограму теоретичним нормальним законом розподілу;

Програмна реалізація:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

from common import data_collector

n = 350
data = data_collector(n)

bins = np.linspace(min(data), max(data), 10)

mean = np.mean(data)
std = np.std(data)

x = np.linspace(min(data), max(data), 100)
pdf = norm.pdf(x, mean, std)

plt.hist(data, bins, density=True, alpha=0.6, label='Гістограма відносних частот')
plt.plot(x, pdf, '-o', label='Нормальний розподіл')

plt.xlabel('Значення')
plt.ylabel('Відносна частота')
plt.legend()

plt.show()
```

Приклад роботи:



Висновки:

В ході виконання лабораторної роботи застосував на практиці теоретичні знання з теорії ймовірностей, порівняв теоретичні числові характеристики випадкової величини із статистичними.

Опрацьована випадкова величина X(описана формулою 1.) задана нормальним розподілом, знаходиться в межах [-23.2; 27.2]. та із числовими характеристиками M[X] = 2, D[X] = 17,64. Точкові оцінки цих величин при n $\rightarrow \infty$ прямують до теоретично визначених