Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 5

з дисципліни «Методи наукових досліджень»

на тему «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

(центральний ортогональний композиційний план)»

ВИКОНАВ:
студент 2 курсу
групи ІВ-91
Онищук Ю. І.
Залікова – 9122

ПЕРЕВІРИВ:
ас. Регіда П. Г.

Київ – 2021

Мета: Провести трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Завдання:

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.

$$y_{i\max} = 200 + x_{cp\max}$$
$$y_{i\min} = 200 + x_{cp\min}$$

где $x_{cp\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}$, $x_{cp\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Порядок виконання лабораторної роботи

1. Записати рівняння регресії з урахуванням квадратичних членів::
$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_{12}x_1x_2 + b_{13}x_1x_3 + b_{23}x_2x_3 + b_{123}x_1x_2x_3 + b_{11}x_1^2 + b_{22}x_2^2 + b_{33}x_3^2$$
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x1, x2, x3.
3. Скласти матрицю планування для ОЦКП і заповнити нормованими значеннями. Початкова кількість дослідів m = 3.
4. Провести першу статистичну перевірку - перевірку однорідності дисперсії за критерієм Кохрена (якщо дисперсія не однорідна, то збільшити m і почати з п.3).
5. Розрахувати коефіцієнти рівняння регресії, розв`язавши матричні рівняння. При розрахунку використовувати **натуральні** значення x_1, x_2 і x_3.
6. Провести другу статистичну перевірку і скорегувати рівняння регресії.
7. Провести третю статистичну перевірку.
8. Зробити висновки щодо перевірок 3-х критеріїв.

Варіанти обираються по номеру в списку в журналі викладача.

№варіанта	x_1		x_2		x_3	
	min	max	min	max	min	max
120	-10	3	-7	2	-1	6

Програмний код

```
import random as rn
from math import sqrt
from scipy.stats import f, t
import pprint
from sklearn import linear_model

def exp_row(x1_num, x2_num, x3_num, y, m = 3):
    y_gen = [y[0] + (y[1]-y[0])*rn.random() for i in range(m)]
    y_mid = sum(y_gen)/m
    sigma = 0
    for i in range(len(y_gen)):
        sigma += ((y_gen[i] - y_mid)**2)/m
    return [x1_num, x2_num, x3_num, x1_num*x2_num, x1_num*x3_num,
x2_num*x3_num, x1_num*x2_num*x3_num, x1_num*x1_num, x2_num*x2_num,
x3_num*x3_num, y_gen, y_mid, sigma]

def experiment(x1, x2, x3, y, m = 3, N = 15):
    flag = 1
    counter = 1
    while flag:
        flag = 0

        l = 1.215
        x01 = sum(x1)/2
        x02 = sum(x2)/2
        x03 = sum(x3)/2
        deltax1 = x1[1]-x01
        deltax2 = x2[1]-x02
        deltax3 = x3[1]-x03

        exp1 = exp_row(x1[0], x2[0], x3[0], y)
        exp2 = exp_row(x1[0], x2[0], x3[1], y)
        exp3 = exp_row(x1[0], x2[1], x3[0], y)
        exp4 = exp_row(x1[0], x2[1], x3[1], y)
        exp5 = exp_row(x1[1], x2[0], x3[0], y)
        exp6 = exp_row(x1[1], x2[0], x3[1], y)
        exp7 = exp_row(x1[1], x2[1], x3[0], y)
        exp8 = exp_row(x1[1], x2[1], x3[1], y)
        exp9 = exp_row(-l*deltax1 + x01, x02, x03, y)
        exp10 = exp_row(l*deltax1 + x01, x02, x03, y)
        exp11 = exp_row(x01, -l*deltax2 + x02, x03, y)
        exp12 = exp_row(x01, l*deltax2 + x02, x03, y)
        exp13 = exp_row(x01, x02, -l*deltax3 + x03, y)
        exp14 = exp_row(x01, x02, l*deltax3 + x03, y)
        exp15 = exp_row(x01, x02, x03, y)

        table = [exp1, exp2, exp3, exp4, exp5, exp6, exp7, exp8, exp9, exp10,
exp11, exp12, exp13, exp14, exp15]
        norm_table = normalize_table(table, x1, x2, x3)


        cochrane = cochrane_kriteria(table, N)
        cochrane_check = cochrane[0]
        Gp = cochrane[1]

        if not cochrane_check:
            flag = 1
            continue
        else:
```

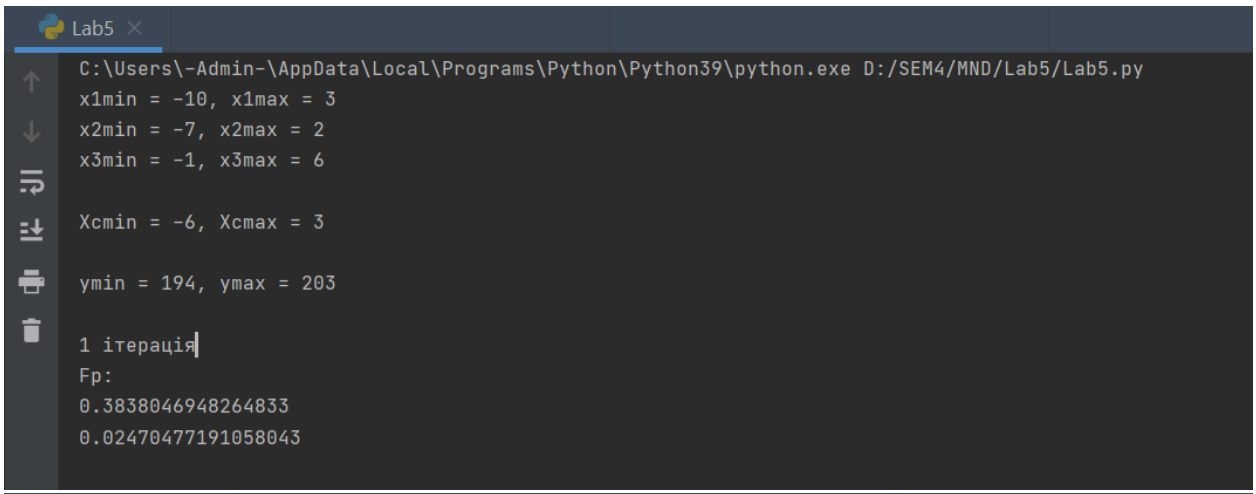
```
            aver_y = list(map(lambda x: x[-2], table))
            x_list = list(map(lambda x: x[0:10], table))
            for i in x_list:
                i.insert(0, 1)

            skm = linear_model.LinearRegression(fit_intercept=False)
            skm.fit(x_list, aver_y)
            b = list(skm.coef_)


            check_b = check(b, table)

            student = student_kriteria(table, N)
            indexes = student[0]
            SB = student[1]
            Sbeta = student[2]
            student_b = list(map(lambda x: x if b.index(x) in indexes else 0,
b))
            student_checks = check(student_b, table)

            fisher = fisher_kriteria(table, student_checks, SB)
            fisher_check = fisher[0]
            Fp = fisher[1]
            print('{} ітерація'.format(counter))
            print('Fp:')
            print(Fp)
            print(Gp*Sbeta*Fp)
            print('\n')

            if fisher_check:
                return table, b, check_b, student_b, student_checks, Fp
            else:
                flag = 1
                counter += 1
                x1 = list(map(lambda x: x/(Gp*Sbeta*Fp), x1))
                x2 = list(map(lambda x: x/(Gp*Sbeta*Fp), x2))
                x3 = list(map(lambda x: x/(Gp*Sbeta*Fp), x3))
                xcmax = int(1/3*(x1[1] + x2[1] + x3[1]))
                xcmin = int(1/3*(x1[0] + x2[0] + x3[0]))
                ymax = 200 + xcmax
                ymin = 200 + xcmin
                y = [ymax, ymin]

def normalize_table(table, x1, x2, x3):
    l = 1.215
    x01 = sum(x1)/2
    x02 = sum(x2)/2
    x03 = sum(x3)/2

    for i in range(len(table)):
        for j in range(3):
            if i < 8:
                if j == 0:
                    if table[i][j] == x1[0]:
                        table[i][j] = -1
                    else:
                        table[i][j] = 1
                elif j == 1:
                    if table[i][j] == x2[0]:
                        table[i][j] = -1
                    else:
                        table[i][j] = 1
                elif j == 2:
```

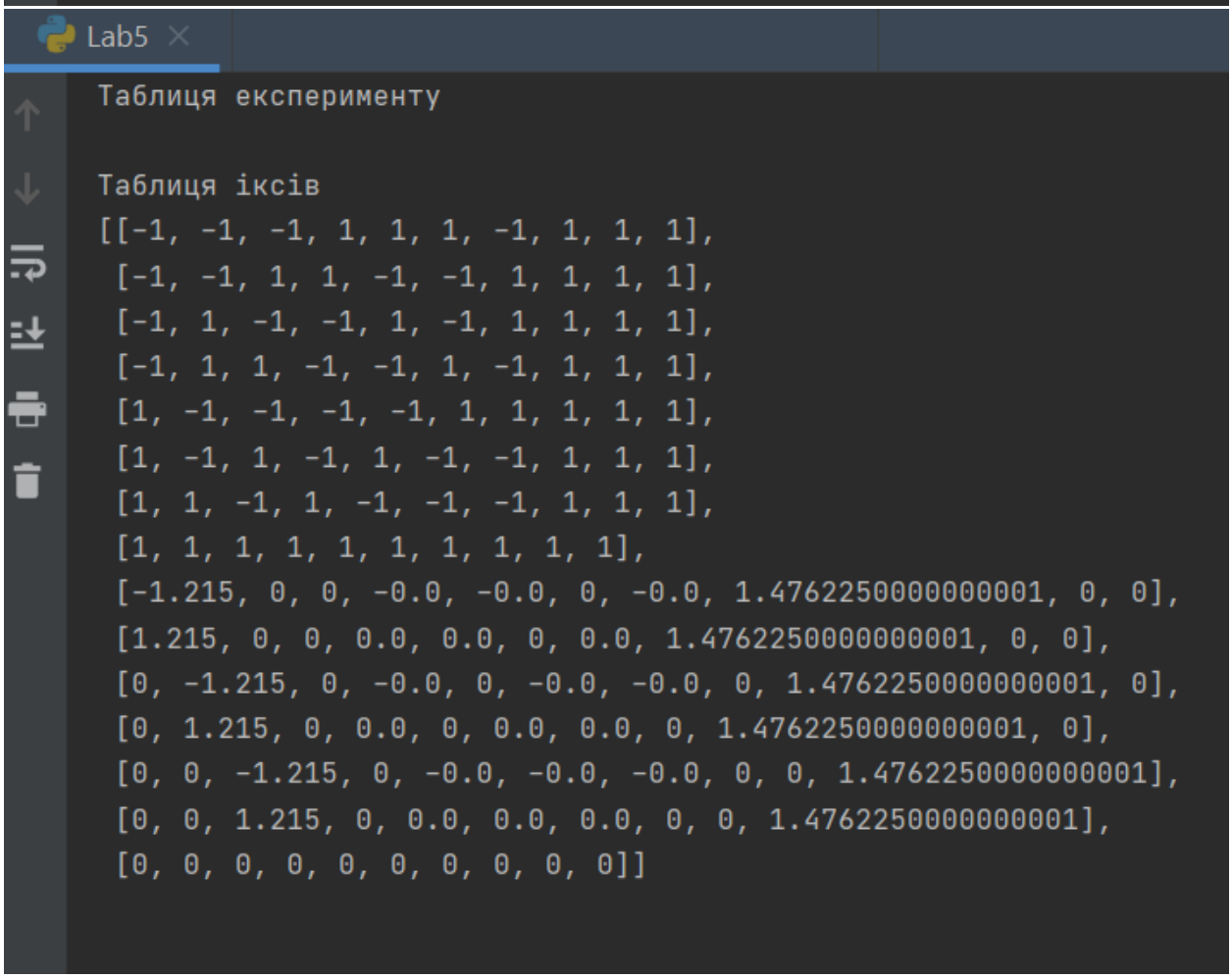
```
                    if table[i][j] == x3[0]:
                        table[i][j] = -1
                    else:
                        table[i][j] = 1
            else:
                if j == 0:
                    if table[i][j] == -l*(x1[1]-x01) + x01:
                        table[i][j] = -l
                    elif table[i][j] == l*(x1[1]-x01) + x01:
                        table[i][j] = l
                    else:
                        table[i][j] = 0
                elif j == 1:
                    if table[i][j] == -l*(x2[1]-x02) + x02:
                        table[i][j] = -l
                    elif table[i][j] == l*(x2[1]-x02) + x02:
                        table[i][j] = l
                    else:
                        table[i][j] = 0
                elif j == 2:
                    if table[i][j] == -l*(x3[1]-x03) + x03:
                        table[i][j] = -l
                    elif table[i][j] == l*(x3[1]-x03) + x03:
                        table[i][j] = l
                    else:
                        table[i][j] = 0
        table[i][3] = table[i][0]*table[i][1]
        table[i][4] = table[i][0]*table[i][2]
        table[i][5] = table[i][1]*table[i][2]
        table[i][6] = table[i][0]*table[i][1]*table[i][2]
        table[i][7] = table[i][0]*table[i][0]
        table[i][8] = table[i][1]*table[i][1]
        table[i][9] = table[i][2]*table[i][2]
    return table

def check(koeficients, table):
    checks = [koeficients[0] for i in range(15)]

    for i in range(len(table)):
        for j in range(10):
            checks[i] += koeficients[j+1]*table[i][j]
    return checks

def cochrane_kriteria(table, N = 15):
    sigma = [table[i][-1] for i in range(N)]
    Gt = 0.3346
    Gp = max(sigma)/sum(sigma)
    if Gp < Gt:
        return 1, Gp
    else:
        return 0, Gp

def student_kriteria(table, N = 15, m = 3):
    sigma = [table[i][-1] for i in range(N)]
    y_mid = [table[i][-2] for i in range(N)]
    SB = sum(sigma)/N
    Sb = SB/(N*m)
    Sbeta = sqrt(Sb)

    beta = [0]*11
    for i in range(N):
        beta[0] += 1/N*(y_mid[i]*1)
    for i in range(1, len(beta)):
        for j in range(N):
```

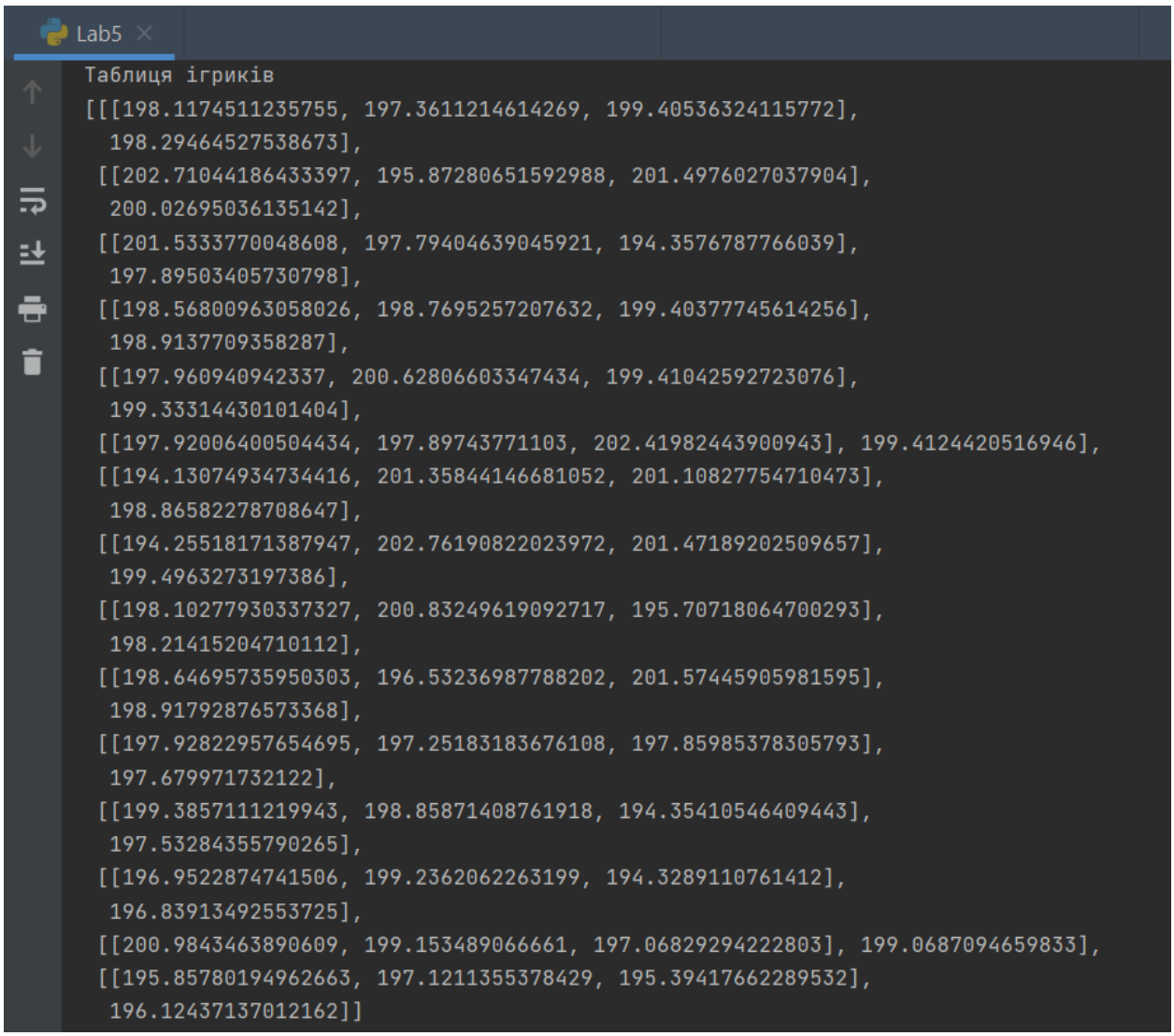
```
            beta[i] += 1/N*(y_mid[j]*table[j][i-1])

    ttab = 2.042

    t = [0]*11
    for i in range(11):
        t[i] = beta[i]/Sbeta

    indexes = []
    for i in t:
        if i > ttab:
            indexes.append(t.index(i))

    return [indexes, SB, Sbeta]

def fisher_kriteria(table, checks, Sb, N = 15, m = 3, d = 4):
    y_mid = [table[i][-2] for i in range(N)]
    Sad = 0
    for i in range(N):
        Sad += m/(N-d)*(checks[i] - y_mid[i])**2
    Ft = 2.16
    Fp = Sad/Sb
    if Fp > Ft:
        return 0, Fp
    else:
        return 1, Fp



x1 = [-10, 3]
print('x1min = {0}, x1max = {1}'.format(x1[0], x1[1]))
x2 = [-7, 2]
print('x2min = {0}, x2max = {1}'.format(x2[0], x2[1]))
x3 = [-1, 6]
print('x3min = {0}, x3max = {1}\n'.format(x3[0], x3[1]))


xcmax = int(1/3*(x1[1] + x2[1] + x3[1]))
xcmin = int(1/3*(x1[0] + x2[0] + x3[0]))
print('Xcmin = {0}, Xcmax = {1}\n'.format(xcmin, xcmax))
ymax = 200 + xcmax
ymin = 200 + xcmin
y = [ymin, ymax]
print('ymin = {0}, ymax = {1}\n'.format(ymin, ymax))

research = experiment(x1, x2, x3, y)
table = research[0]
xses = list(map(lambda x: x[0:10], table))
yeks = list(map(lambda x: x[10:-1], table))

koefs_b = research[1]
check_b = research[2]
koefs_sb = research[3]
check_sb = research[4]
Fp = research[5]


print('Таблиця експерименту\n')
print("Таблиця іксів")
pprint.pprint(xses)
print("\n")
print("Таблиця ігриків")
pprint.pprint(yeks)
#pprint.pprint(table)
```

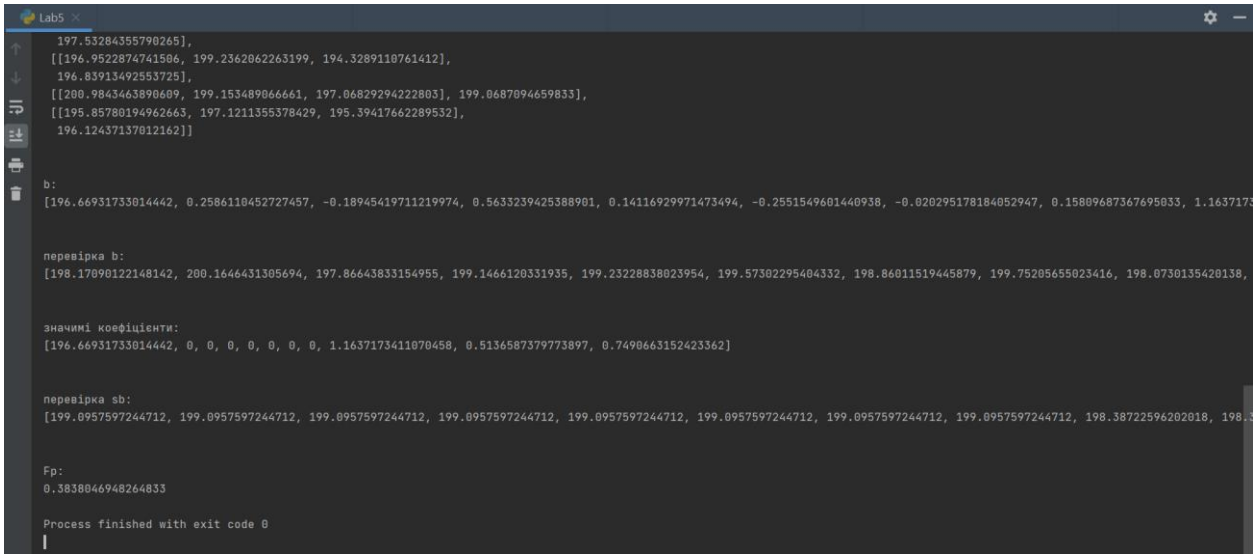
```
print('\n')
print('b:')
print(koefs_b)
print('\n')
print('перевірка b:')
print(check_b)
print('\n')
print('значимі коефіцієнти:')
print(koefs_sb)
print('\n')
print('перевірка sb:')
print(check_sb)
print('\n')
print('Fp:')
print(Fp)
```

Результат роботи програми


Lab5
C:\Users\-Admin-\AppData\Local\Programs\Python\Python39\python.exe D:/SEM4/MND/Lab5/Lab5.py
x1min = -10, x1max = 3
x2min = -7, x2max = 2
x3min = -1, x3max = 6
Xcmin = -6, Xcmax = 3
ymin = 194, ymax = 203
1 ітерація
Fp:
0.3838046948264833
0.02470477191058043


Lab5
Таблиця експерименту
Таблиця іксів
[[-1, -1, -1, 1, 1, 1, -1, 1, 1, 1],
[-1, -1, 1, 1, -1, -1, 1, 1, 1, 1],
[-1, 1, -1, -1, 1, -1, 1, 1, 1, 1],
[-1, 1, 1, -1, -1, 1, -1, 1, 1, 1],
[1, -1, -1, -1, -1, 1, 1, 1, 1, 1],
[1, -1, 1, -1, 1, -1, -1, 1, 1, 1],
[1, 1, -1, 1, -1, -1, -1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
[-1.215, 0, 0, -0.0, -0.0, 0, -0.0, 1.4762250000000001, 0, 0],
[1.215, 0, 0, 0.0, 0.0, 0, 0.0, 1.4762250000000001, 0, 0],
[0, -1.215, 0, -0.0, 0, -0.0, -0.0, 0, 1.4762250000000001, 0],
[0, 1.215, 0, 0.0, 0, 0.0, 0.0, 0, 1.4762250000000001, 0],
[0, 0, -1.215, 0, -0.0, -0.0, -0.0, 0, 0, 1.4762250000000001],
[0, 0, 1.215, 0, 0.0, 0.0, 0.0, 0, 0, 1.4762250000000001],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]


Lab5
Таблиця ігриків
[[[198.1174511235755, 197.3611214614269, 199.40536324115772],
198.29464527538673],
[[202.71044186433397, 195.87280651592988, 201.4976027037904],
200.02695036135142],
[[201.5333770048608, 197.79404639045921, 194.3576787766039],
197.89503405730798],
[[198.56800963058026, 198.7695257207632, 199.40377745614256],
198.9137709358287],
[[197.960940942337, 200.62806603347434, 199.41042592723076],
199.33314430101404],
[[197.92006400504434, 197.89743771103, 202.41982443900943], 199.4124420516946],
[[194.13074934734416, 201.35844146681052, 201.10827754710473],
198.86582278708647],
[[194.25518171387947, 202.76190822023972, 201.47189202509657],
199.4963273197386],
[[198.10277930337327, 200.83249619092717, 195.70718064700293],
198.21415204710112],
[[198.64695735950303, 196.53236987788202, 201.57445905981595],
198.91792876573368],
[[197.92822957654695, 197.25183183676108, 197.85985378305793],
197.679971732122],
[[199.3857111219943, 198.85871408761918, 194.35410546409443],
197.53284355790265],
[[196.9522874741506, 199.2362062263199, 194.3289110761412],
196.83913492553725],
[[200.9843463890609, 199.153489066661, 197.06829294222803], 199.0687094659833],
[[195.85780194962663, 197.12113555378429, 195.39417662289532],
196.12437137012162]]


Lab5
197.53284355790265],
[[196.9522874741506, 199.2362062263199, 194.3289110761412],
196.83913492553725],
[[200.9843463890609, 199.153489066661, 197.06829294222803], 199.0687094659833],
[[195.85780194962663, 197.12113555378429, 195.39417662289532],
196.12437137012162]]
b:
[196.66931733014442, 0.2586110452727457, -0.18945419711219974, 0.5633239425388901, 0.14116929971473494, -0.2551549601440938, -0.020295178184052947, 0.15809687367695033, 1.1637173
перевірка b:
[198.17090122148142, 200.1646431305694, 197.86643833154955, 199.1466120331935, 199.23228838023954, 199.57302295404332, 198.86011519445879, 199.75205655023416, 198.0730135420138,
значимі коефіцієнти:
[196.66931733014442, 0, 0, 0, 0, 0, 0, 1.1637173411070458, 0.5136587379773897, 0.7490663152423362]
перевірка sb:
[199.0957597244712, 199.0957597244712, 199.0957597244712, 199.0957597244712, 199.0957597244712, 199.0957597244712, 199.0957597244712, 199.0957597244712, 198.38722596202018, 198.
Fp:
0.3838046948264833
Process finished with exit code 0