Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №4**

*з дисципліни «Методи оптимізації та планування експерименту»*

# *на тему:* «Проведення трьохфакторного експерименту

# при використанні рівняння регресії з урахуванням ефекту взаємодії.»

**Виконав:**

студент 2-го курсу ФІОТ

групи ІВ-82

Онищук В. В.

Номер у списку: 16

Варiант:216

Перевірив:  
Регіда П.Г.

Київ – 2020

Код:

import numpy as np

import scipy.stats

def getSum(\*args):

    summa = 0

    try:

        if args[0] == "y":

            if len(args) == 1:

                summa = sum(my\_list)

            else:

                for j in range(N):

                    sum\_i\_temp = 1

                    for i in range(len(args) - 1):

                        sum\_i\_temp \*= x\_matrix[j][args[i + 1] - 1]

                    sum\_i\_temp \*= my\_list[j]

                    summa += sum\_i\_temp

        elif len(args) == 1:

            args = args[0] - 1

            for obj in x\_matrix:

                summa += obj[args]

        else:

            for obj in x\_matrix:

                sum\_i\_temp = 1

                for i in range(len(args)):

                    sum\_i\_temp \*= obj[

                        args[i] - 1]

                summa += sum\_i\_temp

    except:

        print("def error")

    return summa

X1\_MIN, X1\_MAX= 10, 50  # начальные условия

X2\_MIN, X2\_MAX = 25, 65

X3\_MIN, X3\_MAX = 50, 65

m = 3

N = 8

mx\_max = (X1\_MAX + X2\_MAX + X3\_MAX) / 3

mx\_min = (X1\_MIN + X2\_MIN + X3\_MIN) / 3

Y\_MAX = mx\_max + 200

Y\_MIN = mx\_min + 200

y\_list = np.random.randint(Y\_MIN, Y\_MAX, (N, m))

x\_matrix = [

    [X1\_MIN, X2\_MIN, X3\_MIN],

    [X1\_MIN, X2\_MIN, X3\_MAX],

    [X1\_MIN, X2\_MAX, X3\_MIN],

    [X1\_MIN, X2\_MAX, X3\_MAX],

    [X1\_MAX, X2\_MIN, X3\_MIN],

    [X1\_MAX, X2\_MIN, X3\_MAX],

    [X1\_MAX, X2\_MAX, X3\_MIN],

    [X1\_MAX, X2\_MAX, X3\_MAX]

]

while 1:  # цикл на проверку однородности дисперсии

    def y\_add\_el():  # функция увеличения m

        for obj in y\_list:

            obj.append(np.random.randint(y\_min, y\_max))

    my\_list = []

    mx1 = 0

    mx2 = 0

    mx3 = 0

    for obj in y\_list:  # создание списка my

        my\_list.append(sum(obj) / len(obj))

    for obj in x\_matrix:

        mx1 += obj[0]

        mx2 += obj[1]

        mx3 += obj[2]

    mx1 /= 8

    mx2 /= 8

    mx3 /= 8

    my = sum(my\_list) / 8

    """Coefficients"""

    m0 = [N, getSum(1), getSum(2), getSum(3), getSum(1, 2), getSum(1, 3), getSum(2, 3), getSum(1, 2, 3)]

    m1 = [getSum(1), getSum(1, 1), getSum(1, 2), getSum(1, 3), getSum(1, 1, 2), getSum(1, 1, 3),

           getSum(1, 2, 3), getSum(1, 1, 2, 3)]

    m2 = [getSum(2), getSum(1, 2), getSum(2, 2), getSum(2, 3), getSum(1, 2, 2), getSum(1, 2, 3),

           getSum(2, 2, 3), getSum(1, 2, 2, 3)]

    m3 = [getSum(3), getSum(1, 3), getSum(2, 3), getSum(3, 3), getSum(1, 2, 3), getSum(1, 3, 3),

           getSum(2, 3, 3), getSum(1, 2, 3, 3)]

    m4 = [getSum(1, 2), getSum(1, 1, 2), getSum(1, 2, 2), getSum(1, 2, 3), getSum(1, 1, 2, 2),

           getSum(1, 1, 2, 3), getSum(1, 2, 2, 3), getSum(1, 1, 2, 2, 3)]

    m5 = [getSum(1, 3), getSum(1, 1, 3), getSum(1, 2, 3), getSum(1, 3, 3), getSum(1, 1, 2, 3),

           getSum(1, 1, 3, 3), getSum(1, 2, 3, 3), getSum(1, 1, 2, 3, 3)]

    m6 = [getSum(2, 3), getSum(1, 2, 3), getSum(2, 2, 3), getSum(2, 3, 3), getSum(1, 2, 2, 3),

           getSum(1, 2, 3, 3), getSum(2, 2, 3, 3), getSum(1, 2, 2, 3, 3)]

    m7 = [getSum(1, 2, 3), getSum(1, 1, 2, 3), getSum(1, 2, 2, 3), getSum(1, 2, 3, 3), getSum(1, 1, 2, 2, 3),

           getSum(1, 1, 2, 3, 3), getSum(1, 2, 2, 3, 3), getSum(1, 1, 2, 2, 3, 3)]

    k0, k1, k2, k3, k4, k5, k6, k7 = getSum("y"), getSum("y", 1), getSum("y", 2), getSum("y", 3), \

                                     getSum("y", 1, 2), getSum("y", 1, 3), getSum("y", 2, 3), getSum("y", 1, 2, 3)

    denominator = np.linalg.det([

        m0,

        m1,

        m2,

        m3,

        m4,

        m5,

        m6,

        m7

    ])

    numerator\_b0 = np.linalg.det([

        [k0, m0[1], m0[2], m0[3], m0[4], m0[5], m0[6], m0[7]],

        [k1, m1[1], m1[2], m1[3], m1[4], m1[5], m1[6], m1[7]],

        [k2, m2[1], m2[2], m2[3], m2[4], m2[5], m2[6], m2[7]],

        [k3, m3[1], m3[2], m3[3], m3[4], m3[5], m3[6], m3[7]],

        [k4, m4[1], m4[2], m4[3], m4[4], m4[5], m4[6], m4[7]],

        [k5, m5[1], m5[2], m5[3], m5[4], m5[5], m5[6], m5[7]],

        [k6, m6[1], m6[2], m6[3], m6[4], m6[5], m6[6], m6[7]],

        [k7, m7[1], m7[2], m7[3], m7[4], m7[5], m7[6], m7[7]]

    ])

    numerator\_b1 = np.linalg.det([

        [m0[0], k0, m0[2], m0[3], m0[4], m0[5], m0[6], m0[7]],

        [m1[0], k1, m1[2], m1[3], m1[4], m1[5], m1[6], m1[7]],

        [m2[0], k2, m2[2], m2[3], m2[4], m2[5], m2[6], m2[7]],

        [m3[0], k3, m3[2], m3[3], m3[4], m3[5], m3[6], m3[7]],

        [m4[0], k4, m4[2], m4[3], m4[4], m4[5], m4[6], m4[7]],

        [m5[0], k5, m5[2], m5[3], m5[4], m5[5], m5[6], m5[7]],

        [m6[0], k6, m6[2], m6[3], m6[4], m6[5], m6[6], m6[7]],

        [m7[0], k7, m7[2], m7[3], m7[4], m7[5], m7[6], m7[7]]

    ])

    numerator\_b2 = np.linalg.det([

        [m0[0], m0[1], k0, m0[3], m0[4], m0[5], m0[6], m0[7]],

        [m1[0], m1[1], k1, m1[3], m1[4], m1[5], m1[6], m1[7]],

        [m2[0], m2[1], k2, m2[3], m2[4], m2[5], m2[6], m2[7]],

        [m3[0], m3[1], k3, m3[3], m3[4], m3[5], m3[6], m3[7]],

        [m4[0], m4[1], k4, m4[3], m4[4], m4[5], m4[6], m4[7]],

        [m5[0], m5[1], k5, m5[3], m5[4], m5[5], m5[6], m5[7]],

        [m6[0], m6[1], k6, m6[3], m6[4], m6[5], m6[6], m6[7]],

        [m7[0], m7[1], k7, m7[3], m7[4], m7[5], m7[6], m7[7]]

    ])

    numerator\_b3 = np.linalg.det([

        [m0[0], m0[1], m0[2], k0, m0[4], m0[5], m0[6], m0[7]],

        [m1[0], m1[1], m1[2], k1, m1[4], m1[5], m1[6], m1[7]],

        [m2[0], m2[1], m2[2], k2, m2[4], m2[5], m2[6], m2[7]],

        [m3[0], m3[1], m3[2], k3, m3[4], m3[5], m3[6], m3[7]],

        [m4[0], m4[1], m4[2], k4, m4[4], m4[5], m4[6], m4[7]],

        [m5[0], m5[1], m5[2], k5, m5[4], m5[5], m5[6], m5[7]],

        [m6[0], m6[1], m6[2], k6, m6[4], m6[5], m6[6], m6[7]],

        [m7[0], m7[1], m7[2], k7, m7[4], m7[5], m7[6], m7[7]]

    ])

    numerator\_b12 = np.linalg.det([

        [m0[0], m0[1], m0[2], m0[3], k0, m0[5], m0[6], m0[7]],

        [m1[0], m1[1], m1[2], m1[3], k1, m1[5], m1[6], m1[7]],

        [m2[0], m2[1], m2[2], m2[3], k2, m2[5], m2[6], m2[7]],

        [m3[0], m3[1], m3[2], m3[3], k3, m3[5], m3[6], m3[7]],

        [m4[0], m4[1], m4[2], m4[3], k4, m4[5], m4[6], m4[7]],

        [m5[0], m5[1], m5[2], m5[3], k5, m5[5], m5[6], m5[7]],

        [m6[0], m6[1], m6[2], m6[3], k6, m6[5], m6[6], m6[7]],

        [m7[0], m7[1], m7[2], m7[3], k7, m7[5], m7[6], m7[7]]

    ])

    numerator\_b13 = np.linalg.det([

        [m0[0], m0[1], m0[2], m0[3], m0[4], k0, m0[6], m0[7]],

        [m1[0], m1[1], m1[2], m1[3], m1[4], k1, m1[6], m1[7]],

        [m2[0], m2[1], m2[2], m2[3], m2[4], k2, m2[6], m2[7]],

        [m3[0], m3[1], m3[2], m3[3], m3[4], k3, m3[6], m3[7]],

        [m4[0], m4[1], m4[2], m4[3], m4[4], k4, m4[6], m4[7]],

        [m5[0], m5[1], m5[2], m5[3], m5[4], k5, m5[6], m5[7]],

        [m6[0], m6[1], m6[2], m6[3], m6[4], k6, m6[6], m6[7]],

        [m7[0], m7[1], m7[2], m7[3], m7[4], k7, m7[6], m7[7]]

    ])

    numerator\_b23 = np.linalg.det([

        [m0[0], m0[1], m0[2], m0[3], m0[4], m0[5], k0, m0[7]],

        [m1[0], m1[1], m1[2], m1[3], m1[4], m1[5], k1, m1[7]],

        [m2[0], m2[1], m2[2], m2[3], m2[4], m2[5], k2, m2[7]],

        [m3[0], m3[1], m3[2], m3[3], m3[4], m3[5], k3, m3[7]],

        [m4[0], m4[1], m4[2], m4[3], m4[4], m4[5], k4, m4[7]],

        [m5[0], m5[1], m5[2], m5[3], m5[4], m5[5], k5, m5[7]],

        [m6[0], m6[1], m6[2], m6[3], m6[4], m6[5], k6, m6[7]],

        [m7[0], m7[1], m7[2], m7[3], m7[4], m7[5], k7, m7[7]]

    ])

    numerator\_b123 = np.linalg.det([

        [m0[0], m0[1], m0[2], m0[3], m0[4], m0[5], m0[6], k0],

        [m1[0], m1[1], m1[2], m1[3], m1[4], m1[5], m1[6], k1],

        [m2[0], m2[1], m2[2], m2[3], m2[4], m2[5], m2[6], k2],

        [m3[0], m3[1], m3[2], m3[3], m3[4], m3[5], m3[6], k3],

        [m4[0], m4[1], m4[2], m4[3], m4[4], m4[5], m4[6], k4],

        [m5[0], m5[1], m5[2], m5[3], m5[4], m5[5], m5[6], k5],

        [m6[0], m6[1], m6[2], m6[3], m6[4], m6[5], m6[6], k6],

        [m7[0], m7[1], m7[2], m7[3], m7[4], m7[5], m7[6], k7]

    ])

    b0 = numerator\_b0 / denominator

    b1 = numerator\_b1 / denominator

    b2 = numerator\_b2 / denominator

    b3 = numerator\_b3 / denominator

    b12 = numerator\_b12 / denominator

    b13 = numerator\_b13 / denominator

    b123 = numerator\_b123 / denominator

    print("b\u2080:", "%.2f" % b0, " b\u2081:", "%.2f" % b1, " b\u2082:", "%.2f" % b2, " b\u2083:", "%.2f" % b3, " b\u2081\u2082:", "%.2f" % b12,

          " b\u2081\u2083:", "%.2f" % b13, " b\u2081\u2082\u2083:", "%.2f" % b123)

    print(

        f"Рівняння регресії: y = {b0:.2f}{b1:+.2f}\*x\u2081{b2:+.2f}\*x\u2082{b3:+.2f}\*x\u2083{b12:+.2f}\*x\u2081\u2082{b13:+.2f}\*x\u2081\u2083{b123:+.2f}\*x\u2081\u2082\u2083")

    # find dispersion

    S2 = []

    for i in range(len(y\_list)):

        S2.append(((y\_list[i][0] - my\_list[i]) \*\* 2 + (y\_list[i][1] - my\_list[i]) \*\* 2 + (

                    y\_list[i][2] - my\_list[i]) \*\* 2) / 3)

    """KOHREN"""

    Gp = max(S2) / sum(S2)

    m = len(y\_list[0])

    f1 = m - 1

    f2 = N  # N=8

    q = 0.05

    Gt = [None, 0.68, 0.516, 0.438, 0.391, 0.3595, 0.3365, 0.3185, 0.3043, 0.2926, 0.2829, 0.2462, 0.2022, 0.1616,

          0.1250]

    print("Gt:", Gt[f1])

    if Gp < Gt[f1]:

        print("Дисперсія однорідна")

        break

    else:

        print("Дисперсія не однорідна")

        m += 1

        y\_add\_el()

x\_matrix\_normal = [

    [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],

]

"""STUDENT"""

def getBeta(i):

    summa = 0

    for j in range(N):

        summa += my\_list[j] \* x\_matrix\_normal[j][i]

    summa /= N

    return summa

S2B = sum(S2) / N

S2beta = S2B / (N \* m)

Sbeta = np.sqrt(S2beta)

beta0 = getBeta(0)

beta1 = getBeta(1)

beta2 = getBeta(2)

beta3 = getBeta(3)

t\_criterion = []

t\_criterion.append(abs(beta0) / Sbeta, )

t\_criterion.append(abs(beta1) / Sbeta)

t\_criterion.append(abs(beta2) / Sbeta)

t\_criterion.append(abs(beta3) / Sbeta)

t0 = abs(beta0) / Sbeta

t1 = abs(beta1) / Sbeta

t2 = abs(beta2) / Sbeta

t3 = abs(beta3) / Sbeta

f3 = f1 \* f2

t\_tab = scipy.stats.t.ppf((1 + (1 - q)) / 2, f3)

print("t табличне:", t\_tab)

if t0 < t\_tab:

    b0 = 0

    print("t\u2080:", t0, " t0<t\_tab; b0=0")

if t1 < t\_tab:

    b1 = 0

    print("t\u2081:", t1, " t\u2081<t\_tab; b\u2081=0")

if t2 < t\_tab:

    b2 = 0

    print("t\u2082:", t2, " t\u2082<t\_tab; b\u2082=0")

if t3 < t\_tab:

    b3 = 0

    print("t\u2083:", t3, " t\u2083<t\_tab; b\u2083=0")

y\_hat = []

for i in range(N):

    y\_hat.append(

        b0 + b1 \* x\_matrix[i][0] + b2 \* x\_matrix[i][1] + b3 \* x\_matrix[i][2] + b12 \* x\_matrix[i][0] \* x\_matrix[i][1] +

        b13 \* x\_matrix[i][0] \* x\_matrix[i][2] + b123 \* x\_matrix[i][0] \* x\_matrix[i][1] \* x\_matrix[i][2])

    print(f"y{chr(8321+i)}^ = {b0:.2f}{b1:+.2f}\*x{chr(8321+i)}\u2081{b2:+.2f}\*x{chr(8321+i)}\u2082{b3:+.2f}\*x{chr(8321+i)}\u2083{b12:+.2f}\*x{chr(8321+i)}\u2081"

          f"\*x{chr(8321+i)}\u2082{b13:+.2f}\*x{chr(8321+i)}\u2081\*x{chr(8321+i)}\u2083{b123:+.2f}\*x{chr(8321+i)}\u2081\*x{chr(8321+i)}\u2082\*x{chr(8321+i)}\u2083 "

          f"= {y\_hat[i]:.2f}")

"""FISHER"""

d = 2

f4 = N - d

S2\_ad = 0

for i in range(N):

    S2\_ad += (m / (N - d) \* ((y\_hat[i] - my\_list[i]) \*\* 2))

Fp = S2\_ad / S2B

Ft = scipy.stats.f.ppf(1 - q, f4, f3)

print("Fp:", Fp)

print("Ft:", Ft)

if Fp > Ft:

    print("Рівняння регресії не адекватно оригіналу при рівні значимості 0,05")

else:

    print("Рівняння регресії адекватно оригіналу при рівні значимості 0,05")

Вивід:

E:\Vlad\МОПЕ\lab4\venv\Scripts\python.exe E:/Vlad/МОПЕ/lab4/lab4.py

b₀: 201.75 b₁: 2.52 b₂: 1.29 b₃: 0.97 b₁₂: -0.06 b₁₃: -0.05 b₁₂₃: 0.00

Рівняння регресії: y = 201.75+2.52\*x₁+1.29\*x₂+0.97\*x₃-0.06\*x₁₂-0.05\*x₁₃+0.00\*x₁₂₃

Gt: 0.516

Дисперсія однорідна

t табличне: 2.119905299221011

t₁: 1.5562094041282513 t₁<t\_tab; b₁=0

t₂: 0.39668082850328323 t₂<t\_tab; b₂=0

t₃: 0.5187364680427522 t₃<t\_tab; b₃=0

y₁^ = 201.75+0.00\*x₁₁+0.00\*x₁₂+0.00\*x₁₃-0.06\*x₁₁\*x₁₂-0.05\*x₁₁\*x₁₃+0.00\*x₁₁\*x₁₂\*x₁₃ = 176.01

y₂^ = 201.75+0.00\*x₂₁+0.00\*x₂₂+0.00\*x₂₃-0.06\*x₂₁\*x₂₂-0.05\*x₂₁\*x₂₃+0.00\*x₂₁\*x₂₂\*x₂₃ = 172.84

y₃^ = 201.75+0.00\*x₃₁+0.00\*x₃₂+0.00\*x₃₃-0.06\*x₃₁\*x₃₂-0.05\*x₃₁\*x₃₃+0.00\*x₃₁\*x₃₂\*x₃₃ = 174.76

y₄^ = 201.75+0.00\*x₄₁+0.00\*x₄₂+0.00\*x₄₃-0.06\*x₄₁\*x₄₂-0.05\*x₄₁\*x₄₃+0.00\*x₄₁\*x₄₂\*x₄₃ = 178.51

y₅^ = 201.75+0.00\*x₅₁+0.00\*x₅₂+0.00\*x₅₃-0.06\*x₅₁\*x₅₂-0.05\*x₅₁\*x₅₃+0.00\*x₅₁\*x₅₂\*x₅₃ = 73.02

y₆^ = 201.75+0.00\*x₆₁+0.00\*x₆₂+0.00\*x₆₃-0.06\*x₆₁\*x₆₂-0.05\*x₆₁\*x₆₃+0.00\*x₆₁\*x₆₂\*x₆₃ = 57.19

y₇^ = 201.75+0.00\*x₇₁+0.00\*x₇₂+0.00\*x₇₃-0.06\*x₇₁\*x₇₂-0.05\*x₇₁\*x₇₃+0.00\*x₇₁\*x₇₂\*x₇₃ = 66.77

y₈^ = 201.75+0.00\*x₈₁+0.00\*x₈₂+0.00\*x₈₃-0.06\*x₈₁\*x₈₂-0.05\*x₈₁\*x₈₃+0.00\*x₈₁\*x₈₂\*x₈₃ = 85.52

Fp: 1620.5886828448736

Ft: 2.741310828338778

Рівняння регресії не адекватно оригіналу при рівні значимості 0,05

Process finished with exit code 0