Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Методи оптимізації та планування експерименту**

Лабораторна робота №3:

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ

ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

Виконав:

студент групи ІО-83

Бурлака Данило Сергійович

Залікова книжка № 8305

Перевірив Регіда П. Г.

Київ 2020р.

**Лабораторна робота №3**

**Тема:** ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ

ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

**Мета:** Провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти

коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

**Виконання:**

Варіант – 305.



x1min = -30; x2min = -25; x3min = -30;

x1max = 20; x2min = 10; x3min = -15;

Ymin = 171 Ymax = 205

1. Лістинг програми:
2. import random, math  
   import numpy as np  
     
   x1\_min = -30  
   x1\_max = 20  
   x2\_min = -25  
   x2\_max = 10  
   x3\_min = -30  
   x3\_max = -15  
   y\_min = 171  
   y\_max = 205  
     
   X = [[-1.0, -1.0, -1.0],  
    [-1.0, 1.0, 1.0],  
    [1.0, -1.0, 1.0],  
    [1.0, 1.0, -1.0]]  
     
   Gt = {1: 0.9065, 2: 0.7679, 3: 0.6841, 4: 0.6287, 5: 0.5892, 6: 0.5598, 7: 0.5365, 8: 0.5175, 9: 0.5017,  
    10: 0.4884}  
   Gt2=[(range(11, 17), 0.4366), (range(17, 37), 0.3720), (range(37, 145), 0.3093)]  
     
   t\_tabl = {1: 12.71, 2: 4.303, 3: 3.182, 4: 2.776, 5: 2.571,  
    6: 2.447, 7: 2.365, 8: 2.306, 9: 2.262, 10: 2.228,  
    11: 2.201, 12: 2.179, 13: 2.160, 14: 2.145, 15: 2.131,  
    16: 2.120, 17: 2.110, 18: 2.101, 19: 2.093, 20: 2.086,  
    21: 2.080, 22: 2.074, 23: 2.069, 24: 2.064, 25: 2.060,  
    26: 2.056, 27: 2.052, 28: 2.048, 29: 2.045, 30: 2.042}  
   t\_tabl2=1.960  
     
   Ft = {8: {1: 5.3, 2: 4.5, 3: 4.1, 4: 3.8}, 9: {1: 5.1, 2: 4.3, 3: 3.9, 4: 3.6},  
    10: {1: 5.0, 2: 4.1, 3: 3.7, 4: 3.5}, 11: {1: 4.8, 2: 4.0, 3: 3.6, 4: 3.4},  
    12: {1: 4.8, 2: 3.9, 3: 3.5, 4: 3.3}, 13: {1: 4.7, 2: 3.8, 3: 3.4, 4: 3.2},  
    14: {1: 4.6, 2: 3.7, 3: 3.3, 4: 3.1}, 15: {1: 4.4, 2: 3.7, 3: 3.3, 4: 3.1},  
    16: {1: 4.5, 2: 3.6, 3: 3.2, 4: 3.0}, 17: {1: 4.5, 2: 3.6, 3: 3.2, 4: 3.0},  
    18: {1: 4.4, 2: 3.6, 3: 3.2, 4: 2.9}, 19: {1: 4.4, 2: 3.5, 3: 3.1, 4: 2.9}}  
     
   Ft2=[ (range(20, 22), {1: 4.4, 2: 3.5, 3: 3.1, 4: 2.9}), (range(22, 24), {1: 4.3, 2: 3.4, 3: 3.1, 4: 2.8}),  
    (range(24, 26), {1: 4.3, 2: 3.4, 3: 3.0, 4: 2.8}), (range(26, 28), {1: 4.2, 2: 3.4, 3: 3.0, 4: 2.7}),  
    (range(28, 30), {1: 4.2, 2: 3.3, 3: 3.0, 4: 2.7}), (range(30, 40), {1: 4.2, 2: 3.3, 3: 2.9, 4: 2.7}),  
    (range(40, 60), {1: 4.1, 2: 3.2, 3: 2.9, 4: 2.6}), (range(60, 120), {1: 4.0, 2: 3.2, 3: 2.8, 4: 2.5})]  
   Ft3={1: 3.8, 2:3.0, 3: 2.6, 4: 2.4}  
     
     
   def func(num):  
    # матриця У  
    N = 4  
    m = num  
    Y = [[random.randint(y\_min, y\_max) for y in range(m)] for x in range(N)]  
    print("~"\*40)  
    print("Y: ")  
    for i in range(len(Y)):  
    print(Y[i])  
    print("~"\*40)  
    # матриця Х  
    for i in range(len(X)):  
    if X[i][0] == -1:  
    X[i][0] = x1\_min  
    else:  
    X[i][0] = x1\_max  
    if X[i][1] == -1:  
    X[i][1] = x2\_min  
    else:  
    X[i][1] = x2\_max  
    if X[i][2] == -1:  
    X[i][2] = x3\_min  
    else:  
    X[i][2] = x3\_max  
    print("X: ")  
    for i in range(len(X)):  
    print(X[i])  
    print("~"\*40)  
    # середнє по рядках  
    Ys = []  
    for i in range(len(Y)):  
    Ys.append(sum(Y[i]) / m)  
    print("Ys: ", Ys)  
    print("~"\*40)  
    mx1 = (X[0][0] + X[1][0] + X[2][0] + X[3][0]) / N  
    mx2 = (X[0][1] + X[1][1] + X[2][1] + X[3][1]) / N  
    mx3 = (X[0][2] + X[1][2] + X[2][2] + X[3][2]) / N  
    my = sum(Ys) / N  
    print("mx1= {0} \n"  
    "mx2= {1} \n"  
    "mx3= {2} \n"  
    "my= {3} ".format(mx1, mx2, mx3, my))  
    print("~"\*40)  
    a1 = ((X[0][0] \* Ys[0]) + (X[1][0] \* Ys[1]) + (X[2][0] \* Ys[2]) + (X[3][0] \* Ys[3])) / N  
    a2 = ((X[0][1] \* Ys[0]) + (X[1][1] \* Ys[1]) + (X[2][1] \* Ys[2]) + (X[3][1] \* Ys[3])) / N  
    a3 = ((X[0][2] \* Ys[0]) + (X[1][2] \* Ys[1]) + (X[2][2] \* Ys[2]) + (X[3][2] \* Ys[3])) / N  
    print("a1= {0} \n"  
    "a2= {1} \n"  
    "a3= {2} ".format(a1, a2, a3))  
    print("~"\*40)  
    a11 = (X[0][0] \*\* 2 + X[1][0] \*\* 2 + X[2][0] \*\* 2 + X[3][0] \*\* 2) / N  
    a22 = (X[0][1] \*\* 2 + X[1][1] \*\* 2 + X[2][1] \*\* 2 + X[3][1] \*\* 2) / N  
    a33 = (X[0][2] \*\* 2 + X[1][2] \*\* 2 + X[2][2] \*\* 2 + X[3][2] \*\* 2) / N  
    print("a11= {0} \n"  
    "a22= {1} \n"  
    "a33= {2} ".format(a11, a22, a33))  
    print("~"\*40)  
    a12 = a21 = ((X[0][0] \* X[0][1]) + (X[1][0] \* X[1][1]) + (X[2][0] \* X[2][1]) + (X[3][0] \* X[3][1])) / N  
    a13 = a31 = ((X[0][0] \* X[0][2]) + (X[1][0] \* X[1][2]) + (X[2][0] \* X[2][2]) + (X[3][0] \* X[3][2])) / N  
    a23 = a32 = ((X[0][1] \* X[0][2]) + (X[1][1] \* X[1][2]) + (X[2][1] \* X[2][2]) + (X[3][1] \* X[3][2])) / N  
    print("a12=a21= {0} \n"  
    "a13=a31= {1} \n"  
    "a23=a32= {2} ".format(a12, a13, a23))  
    print("~"\*40)  
    # Розрахуємо детермінанти матриці  
    znamen = np.array([[1, mx1, mx2, mx3],  
    [mx1, a11, a12, a13],  
    [mx2, a12, a22, a32],  
    [mx3, a13, a23, a33]], dtype='float')  
    ch0 = np.array([[my, mx1, mx2, mx3],  
    [a1, a11, a12, a13],  
    [a2, a12, a22, a32],  
    [a3, a13, a23, a33]], dtype='float')  
    ch1 = np.array([[1, my, mx2, mx3],  
    [mx1, a1, a12, a13],  
    [mx2, a2, a22, a32],  
    [mx3, a3, a23, a33]], dtype='float')  
    ch2 = np.array([[1, mx1, my, mx3],  
    [mx1, a11, a1, a13],  
    [mx2, a12, a2, a32],  
    [mx3, a13, a3, a33]], dtype='float')  
    ch3 = np.array([[1, mx1, mx2, my],  
    [mx1, a11, a12, a1],  
    [mx2, a12, a22, a2],  
    [mx3, a13, a23, a3]], dtype='float')  
    b0 = np.linalg.det(ch0) / np.linalg.det(znamen)  
    b1 = np.linalg.det(ch1) / np.linalg.det(znamen)  
    b2 = np.linalg.det(ch2) / np.linalg.det(znamen)  
    b3 = np.linalg.det(ch3) / np.linalg.det(znamen)  
    b = []  
    b.append(b0)  
    b.append(b1)  
    b.append(b2)  
    b.append(b3)  
    print("b0= {0} \n"  
    "b1= {1} \n"  
    "b2= {2} \n"  
    "b3= {3}".format(b0, b1, b2, b3))  
    print("~"\*40)  
    # Записуємо нормоване рівняння  
    print("Y={} + {}\*x1 + {}\*x2 + {}\*x3".format(b0, b1, b2, b3))  
    print("~"\*40)  
    # Виконуємо перевірку  
    print("Перевірка: ")  
    print("\n")  
    print(b0 + b1 \* X[0][0] + b2 \* X[0][1] + b3 \* X[0][2], "==", Ys[0])  
    print(b0 + b1 \* X[1][0] + b2 \* X[1][1] + b3 \* X[1][2], "==", Ys[1])  
    print(b0 + b1 \* X[2][0] + b2 \* X[2][1] + b3 \* X[2][2], "==", Ys[2])  
    print(b0 + b1 \* X[3][0] + b2 \* X[3][1] + b3 \* X[3][2], "==", Ys[3])  
    print("Результат збігається з середніми значеннями")  
    print("~"\*40)  
    # Виконуємо перевірку однорідності за Кохреном  
    # Дисперсія по рядках  
    D = []  
    Summa = 0  
    for i in range(N):  
    for j in range(m):  
    Summa += pow((Y[i][j] - Ys[i]), 2)  
    D.append(1 / m \* Summa)  
    Summa = 0  
    print("D: ", D)  
    print("~"\*40)  
    Gp = max(D) / sum(D)  
    print("Gp= ", Gp)  
    f1 = m - 1  
    f2 = N = 4  
    if m >= 11:  
    for i in range(len(Gt2)):  
    if m in Gt2[i][0]:  
    crit = Gt2[i][1]  
    break  
    else:  
    crit = Gt[f1]  
    if Gp <= crit:  
    print("Дисперсія однорідна")  
    print(Gp, "<=", crit)  
    else:  
    print("Дисперсія не однорідна")  
    m += 1  
    print("M:", m)  
    return func(m)  
    print("~"\*40)  
    # Критерій Стьюдента  
    S2\_b = sum(D) / N  
    S2\_beta = S2\_b / (N \* m)  
    S\_beta = math.sqrt(S2\_beta)  
    print("S2\_b= {0} \n"  
    "S2\_beta= {1} \n"  
    "S\_beta= {2}".format(S2\_b, S2\_beta, S\_beta))  
    print("~"\*40)  
    Xs = [[1.0, -1.0, -1.0, -1.0],  
    [1.0, -1.0, 1.0, 1.0],  
    [1.0, 1.0, -1.0, 1.0],  
    [1.0, 1.0, 1.0, -1.0]]  
    betta = []  
     
    beta0 = (Ys[0] \* Xs[0][0] + Ys[1] \* Xs[1][0] + Ys[2] \* Xs[2][0] + Ys[3] \* Xs[3][0]) / N  
    beta1 = (Ys[0] \* Xs[0][1] + Ys[1] \* Xs[1][1] + Ys[2] \* Xs[2][1] + Ys[3] \* Xs[3][1]) / N  
    beta2 = (Ys[0] \* Xs[0][2] + Ys[1] \* Xs[1][2] + Ys[2] \* Xs[2][2] + Ys[3] \* Xs[3][2]) / N  
    beta3 = (Ys[0] \* Xs[0][3] + Ys[1] \* Xs[1][3] + Ys[2] \* Xs[2][3] + Ys[3] \* Xs[3][3]) / N  
    betta.append(beta0)  
    betta.append(beta1)  
    betta.append(beta2)  
    betta.append(beta3)  
    print("beta0= {0} \n"  
    "beta1= {1} \n"  
    "beta2= {2} \n"  
    "beta3= {3}".format(beta0, beta1, beta2, beta3))  
    print("~"\*40)  
    t = []  
    for i in range(len(betta)):  
    t.append(abs(betta[i]) / S\_beta)  
    print("t: ", t)  
    print("~"\*40)  
    f3 = f1 \* f2  
    print("f3=",f3)  
    if f3>30:  
    if t[i] < t\_tabl2:  
    b[i] = 0  
    print(t[i], "<", t\_tabl2)  
    else:  
    for i in range(len(t)):  
    if t[i] < t\_tabl[f3]:  
    b[i] = 0  
    print(t[i], "<", t\_tabl[f3])  
    print("~"\*40)  
    y = []  
    y.append(b0 + b1 \* X[0][0] + b2 \* X[0][1] + b3 \* X[0][2])  
    y.append(b0 + b1 \* X[1][0] + b2 \* X[1][1] + b3 \* X[1][2])  
    y.append(b0 + b1 \* X[2][0] + b2 \* X[2][1] + b3 \* X[2][2])  
    y.append(b0 + b1 \* X[3][0] + b2 \* X[3][1] + b3 \* X[3][2])  
    print("y: ", y)  
    print("\n")  
    for i in range(len(y)):  
    print(y[i], "==", Ys[i])  
    print("~"\*40)  
    # критерій Фішера  
    d = 0  
    for i in range(len(b)):  
    if b[i] != 0:  
    d += 1  
    print("d=",d)  
    f4 = N - d  
    Sum = 0  
    for i in range(len(y)):  
    Sum += pow((y[i] - Ys[i]), 2)  
    S\_ad = (m / (N - d)) \* Sum  
    print("S\_ad= ", S\_ad)  
    Fp = S\_ad / S2\_b  
    print("Fp= {0} \n"  
    "f3= {1} \n"  
    "f4= {2}".format(Fp, f3, f4))  
    print("~"\*40)  
    if f3 < 20:  
    if Fp > Ft[f3][f4]:  
    print("Рівняння регресії неадекватно оригіналу при рівні значимості 0.05")  
    print("Значення критерію = ",Ft[f3][f4])  
    else:  
    print("Рівняння регресії адекватно оригіналу при рівні значимості 0.05")  
    print("Значення критерію = ",Ft[f3][f4])  
    elif f3 >= 20 and f3 <= 120:  
    for i in range(len(Ft2)):  
    if f3 in Ft2[i][0]:  
    cr = Ft2[i][1][f4]  
    break  
    if Fp > cr:  
    print("Рівняння регресії неадекватно оригіналу при рівні значимості 0.05")  
    print("Значення критерію = ",cr)  
    else:  
    print("Рівняння регресії адекватно оригіналу при рівні значимості 0.05")  
    print("Значення критерію = ",cr)  
    else:  
    if Fp > Ft3[f4]:  
    print("Рівняння регресії неадекватно оригіналу при рівні значимості 0.05")  
    print("Значення критерію=",Ft3[f4])  
    else:  
    print("Рівняння регресії адекватно оригіналу при рівні значимості 0.05")  
    print("Значення критерію=",Ft3[f4])  
    print("~" \* 40)  
   func(3)
3. Результат виконання роботи програми:

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Y:

[180, 171, 179]

[182, 172, 173]

[192, 171, 172]

[183, 182, 178]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

X:

[-30, -25, -30]

[-30, 10, -15]

[20, -25, -15]

[20, 10, -30]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Ys: [176.66666666666666, 175.66666666666666, 178.33333333333334, 181.0]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mx1= -5.0

mx2= -7.5

mx3= -22.5

my= 177.91666666666666

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

a1= -845.8333333333333

a2= -1327.0833333333335

a3= -4010.0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

a11= 650.0

a22= 362.5

a33= 562.5

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

a12=a21= 37.5

a13=a31= 112.5

a23=a32= 168.75

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

b0= 175.69523809523787

b1= 0.0700000000000001

b2= 0.023809523809523184

b3= -0.12222222222222853

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Y=175.69523809523787 + 0.0700000000000001\*x1 + 0.023809523809523184\*x2 + -0.12222222222222853\*x3

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Перевірка:

176.66666666666666 == 176.66666666666666

175.66666666666654 == 175.66666666666666

178.33333333333323 == 178.33333333333334

180.99999999999997 == 181.0

Результат збігається з середніми значеннями

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

D: [16.222222222222218, 20.222222222222218, 93.55555555555554, 4.666666666666666]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Gp= 0.6947194719471949

Дисперсія однорідна

0.6947194719471949 <= 0.7679

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

S2\_b= 33.66666666666666

S2\_beta= 2.805555555555555

S\_beta= 1.674979270186815

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

beta0= 177.91666666666666

beta1= 1.750000000000007

beta2= 0.4166666666666643

beta3= -0.9166666666666643

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

t: [106.22022005491634, 1.044789049720493, 0.2487592975524959, 0.5472704546154926]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

f3= 8

1.044789049720493 < 2.306

0.2487592975524959 < 2.306

0.5472704546154926 < 2.306

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

y: [176.66666666666666, 175.66666666666654, 178.33333333333323, 180.99999999999997]

176.66666666666666 == 176.66666666666666

175.66666666666654 == 175.66666666666666

178.33333333333323 == 178.33333333333334

180.99999999999997 == 181.0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

d= 1

S\_ad= 2.665718770922843e-26

Fp= 7.917976547295576e-28

f3= 8

f4= 3

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Рівняння регресії адекватно оригіналу при рівні значимості 0.05

Значення критерію = 4.1

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Контрольні запитання**

1. *Що називається дробовим факторним експериментом?*

У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Якщо буде використовуватися лінійна регресія, то можливо зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів слід скоротити, використовуючи для планування так звані регулярні дробові репліки від повного факторного експерименту, що містять відповідну кількість дослідів і зберігають основні властивості матриці планування – це означає дробовий факторний експеримент (ДФЕ).

1. *Для чого потрібно розрахункове значення Кохрена?*

Для перевірки дисперсії на однорідність.

1. *Для чого перевіряється критерій Стьюдента?*

Для перевірки значущості коефіцієнтів регресії. Тобто, Якщо виконується нерівність *ts*< *tтабл*, то приймається нуль-гіпотеза, тобто вважається, що знайдений коефіцієнт *βs* є статистично незначущим і його слід виключити з рівняння регресії. Якщо *ts* > *tтабл* то гіпотеза не підтверджується, тобто *βs* – значимий коефіцієнт і він залишається в рівнянні регресії.

1. *Чим визначається критерій Фішера і як його застосовувати?*

Отримане рівняння регресії необхідно перевірити на адекватність досліджуваному об'єкту. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності. Адекватність моделі перевіряють за F-критерієм Фішера, який дорівнює відношенню дисперсії адекватності до дисперсії відтворюваності.

**Висновок:** Отже, у ході виконання лабораторної роботи № 3 провели дробовий трьохфакторний експеримент. Склали матрицю планування, знайшли

коефіцієнти рівняння регресії, провели 3 статистичні перевірки. Була написана текстова програма, результати наведені вище. Результати співпадають із калькулятором. Кінцева мета роботи досягнута.