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 №4

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ»

Виконала:
студентка групи ІО-91
Тимошенко Діана
Варіант: 123
Перевірив Регіда П. Г.

Київ
2021 р.

Мета: провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

Завдання на лабораторну роботу

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.
   $y_{max} = 200 + x_{cp\ max}$
   $y_{min} = 200 + x_{cp\ min}$
   $x_{cp\ max} = (x_{1max} + x_{2max} + x_{2max}) / 3$
   $x_{cp\ min} = (x_{1min} + x_{2min} + x_{2min}) / 3$
3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
123	-5	15	10	60	10	20

Роздруківка тексту програми:

```
from random import randint
import numpy as np
from math import sqrt
from scipy.stats import f, t

x_range = [(-5, 15), (10, 60), (10, 20)]
xcp_min = round(sum([x_range[i][0] for i in range(len(x_range))]) / 3)
xcp_max = round(sum([x_range[i][1] for i in range(len(x_range))]) / 3)
y_min, y_max = 200 + xcp_min, 200 + xcp_max

def linear_matrix(m, n):
    x_norm = np.array([[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]])
```

```
    x_natur = np.zeros(shape=(n, len(x_norm[0]) - 1))
    for i in range(len(x_norm)):
        for j in range(len(x_norm[i]) - 1):
            if x_norm[i][j + 1] == 1:
                x_natur[i][j] = x_range[j][1]
            else:
                x_natur[i][j] = x_range[j][0]

    y = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = randint(y_min, y_max)

    linear_coef(x_natur, x_norm, y)


def linear_coef(x_natur, x_norm, y):
    y_aver = [sum(y[i]) / m for i in range(n)]
    print(x_natur)
    print("Y:", y)
    print("Середні значення функції відгуку за рядками:", y_aver)
    mx1 = sum(x_natur[i][0] for i in range(n)) / n
    mx2 = sum(x_natur[i][1] for i in range(n)) / n
    mx3 = sum(x_natur[i][2] for i in range(n)) / n
    my = sum(y_aver) / n

    a1 = sum(x_natur[i][0] * y_aver[i] for i in range(n)) / n
    a2 = sum(x_natur[i][1] * y_aver[i] for i in range(n)) / n
    a3 = sum(x_natur[i][2] * y_aver[i] for i in range(n)) / n

    a11 = sum(x_natur[i][0] * x_natur[i][0] for i in range(n)) / n
    a22 = sum(x_natur[i][1] * x_natur[i][1] for i in range(n)) / n
    a33 = sum(x_natur[i][2] * x_natur[i][2] for i in range(n)) / n

    a12 = a21 = sum(x_natur[i][0] * x_natur[i][1] for i in range(n)) / n
    a13 = a31 = sum(x_natur[i][0] * x_natur[i][2] for i in range(n)) / n
    a23 = a32 = sum(x_natur[i][1] * x_natur[i][2] for i in range(n)) / n

    matr_X = [[1, mx1, mx2, mx3],
              [mx1, a11, a21, a31],
              [mx2, a12, a22, a32],
              [mx3, a13, a23, a33]]
    matr_Y = [my, a1, a2, a3]
    b_natur = np.linalg.solve(matr_X, matr_Y)

    print("\nНатуралізоване рівняння регресії: y = {0:.2f} {1:+.2f}*x1 {2:+.2f}*x2
{3:+.2f}*x3".format(*b_natur))

    b_norm = [sum(y_aver) / n,
              sum(y_aver[i] * x_norm[i][1] for i in range(n)) / n,
              sum(y_aver[i] * x_norm[i][2] for i in range(n)) / n,
              sum(y_aver[i] * x_norm[i][3] for i in range(n)) / n]
    print("\nНормоване рівняння регресії: y = {0:.2f} {1:+.2f}*x1 {2:+.2f}*x2
{3:+.2f}*x3".format(*b_norm))
    cohren(m, y, y_aver, x_norm, x_natur, b_natur)


# ----------------- Критерій Кохрена ----------------------
def cohren(m, y, y_aver, x_norm, x_natur, b_coef):
    print("\nКритерій Кохрена")
    dispersion = []
```

```
    for i in range(n):
        z = 0
        for j in range(m):
            z += (y[i][j] - y_aver[i]) ** 2
        dispersion.append(z / m)
    print("Дисперсія:", [round(elem, 3) for elem in dispersion])

    Gp = max(dispersion) / sum(dispersion)
    f1 = m - 1
    f2 = n
    q = 0.05
    Gt = f.ppf(q=(1 - q / f1), dfn=f2, dfd=(f1 - 1) * f2)
    Gt = Gt / (Gt + f1 - 1)
    if Gp < Gt:
        print("Gp < Gt\n{0:.4f} < {1} => дисперсія однорідна".format(Gp, Gt))
        student(m, dispersion, y_aver, x_norm, x_natur, b_coef)
    else:
        print("Gp > Gt\n{0:.4f} > {1} => дисперсія неоднорідна => m+=1".format(Gp,
Gt))
        m += 1
        if flag:
            linear_matrix(m, n)
        else:
            inter_matrix(m, n)


# ---------------------- Критерій Стюдента -------------------------------
def student(m, dispersion, y_aver, x_norm, x_natur, b_coef):
    print("\nКритерій Стюдента")
    sb = sum(dispersion) / n
    s_beta = sqrt(sb / (n * m))
    k = len(x_norm[0])
    beta = [sum(y_aver[i] * x_norm[i][j] for i in range(n)) / n for j in range(k)]

    t_t = [abs(beta[i]) / s_beta for i in range(k)]

    f3 = (m - 1) * n
    qq = (1 + 0.95) / 2
    t_table = t.ppf(df=f3, q=qq)

    b_impor = []
    for i in range(k):
        if t_t[i] > t_table:
            b_impor.append(b_coef[i])
        else:
            b_impor.append(0)
    print("Незначні коефіцієнти регресії")
    for i in range(k):
        if b_coef[i] not in b_impor:
            print("b{0} = {1:.2f}".format(i, b_coef[i]))

    if flag:
        y_impor = [b_impor[0] + b_impor[1] * x_norm[i][1] + b_impor[2] * x_norm[i][2]
+ b_impor[3] * x_norm[i][3] +
                   b_impor[4] * x_norm[i][4] + b_impor[5] * x_norm[i][5] + b_impor[6]
* x_norm[i][6] +
                   b_impor[7] * x_norm[i][7] for i in range(n)]
    else:
        y_impor = [b_impor[0] + b_impor[1] * x_natur[i][0] + b_impor[2] *
x_natur[i][1] + b_impor[3] * x_natur[i][2] for
                   i in range(n)]
```

```
    print("Значення функції відгуку зі значущими коефіцієнтами", [round(elem, 3) for
elem in y_impor])
    fisher(m, y_aver, b_impor, y_impor, sb)


# ---------------------- Критерій Фішера -------------------------------
def fisher(m, y_aver, b_impor, y_impor, sb):
    print("\nКритерій Фішера")
    d = 0
    for i in b_impor:
        if i:
            d += 1
    f3 = (m - 1) * n
    f4 = n - d
    s_ad = sum((y_impor[i] - y_aver[i]) ** 2 for i in range(n)) * m / f4
    Fp = s_ad / sb
    Ft = f.ppf(dfn=f4, dfd=f3, q=1 - 0.05)

    if Fp < Ft:
        print("Fp < Ft => {0:.2f} < {1}".format(Fp, Ft))
        print("Отримана математична модель при рівні значимості 0.05 адекватна
експериментальним даним")
    else:
        print("Fp > Ft => {0:.2f} > {1}".format(Fp, Ft))
        print("Рівняння регресії неадекватно оригіналу при рівні значимості 0.05")
        inter_matrix(m, n)

def inter_matrix(m, n):
    global flag
    flag = True
    print("\n---------------------------")
    x_norm = [[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]]
    for i in range(n):
        x_norm[i].append(x_norm[i][1] * x_norm[i][2])
        x_norm[i].append(x_norm[i][1] * x_norm[i][3])
        x_norm[i].append(x_norm[i][2] * x_norm[i][3])
        x_norm[i].append(x_norm[i][1] * x_norm[i][2] * x_norm[i][3])

    x_natur = np.zeros(shape=(n, len(x_norm[0]) - 1))
    for i in range(len(x_norm)):
        for j in range(3):
            if x_norm[i][j + 1] == 1:
                x_natur[i][j] = x_range[j][1]
            else:
                x_natur[i][j] = x_range[j][0]
    for i in range(n):
        x_natur[i][3] = x_natur[i][0] * x_natur[i][1]
        x_natur[i][4] = x_natur[i][0] * x_natur[i][2]
        x_natur[i][5] = x_natur[i][1] * x_natur[i][2]
        x_natur[i][6] = x_natur[i][0] * x_natur[i][1] * x_natur[i][2]

    y = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = randint(y_min, y_max)
```

```
    inter_coef(x_natur, x_norm, y)

def inter_coef(x_natur, x_norm, y):
    y_aver = [sum(y[i]) / m for i in range(n)]
    print("Натуралізована матриця X\n", x_natur)
    print("\nМатриця Y\n", y)
    print("Середні значення функції відгуку за рядками:", [round(elem, 3) for elem in
y_aver])

    b0 = sum(y_aver) / n
    b1 = sum(x_norm[i][1] * y_aver[i] for i in range(n)) / n
    b2 = sum(x_norm[i][2] * y_aver[i] for i in range(n)) / n
    b3 = sum(x_norm[i][3] * y_aver[i] for i in range(n)) / n
    b12 = sum(x_norm[i][4] * y_aver[i] for i in range(n)) / n
    b13 = sum(x_norm[i][5] * y_aver[i] for i in range(n)) / n
    b23 = sum(x_norm[i][6] * y_aver[i] for i in range(n)) / n
    b123 = sum(x_norm[i][7] * y_aver[i] for i in range(n)) / n

    b_norm = [b0, b1, b2, b3, b12, b13, b23, b123]

    print("\nНормоване рівняння регресії: y = {0:.2f} {1:+.2f}*x1 {2:+.2f}*x2
{3:+.2f}*x3 {4:+.2f}*x12 "
          "{5:+.2f}*x13 {6:+.2f}*x23 {7:+.2f}*x123".format(*b_norm))

    cohren(m, y, y_aver, x_norm, x_natur, b_norm)


if __name__ == '__main__':
    # False => linear
    flag = False
    n = 8
    m = 3
    linear_matrix(m, n)
```

Результат виконання програми:

```
Натуралізована матриця X
 [[-5. 10. 10.]
 [-5. 10. 20.]
 [-5. 60. 10.]
 [-5. 60. 20.]
 [15. 10. 10.]
 [15. 10. 20.]
 [15. 60. 10.]
 [15. 60. 20.]]

Матриця Y
 [[212. 216. 205.]
 [211. 211. 215.]
 [217. 222. 230.]
 [205. 218. 221.]
 [224. 208. 215.]
 [226. 223. 217.]
 [218. 223. 217.]
 [219. 211. 207.]]
Середні значення функції відгуку за рядками: [211.0, 212.333, 223.0, 214.667, 215.667, 222.0, 219.333, 212.333]

Натуралізоване рівняння регресії: y = 217.19 +0.10*x1 +0.04*x2 -0.19*x3

Нормоване рівняння регресії: y = 216.29 +1.04*x1 +1.04*x2 -0.96*x3

Критерій Кохрена
Дисперсія: [20.667, 3.556, 28.667, 48.222, 42.889, 14.0, 6.889, 24.889]
Gp < Gt
0.2541 < 0.815948432359917 => дисперсія однорідна
```

```
Критерій Стюдента
Незначні коефіцієнти регресії
b1 = 0.10
b2 = 0.04
b3 = -0.19
Значення функції відгуку зі значущими коефіцієнтами [217.188, 217.188, 217.188, 217.188, 217.188, 217.188, 217.188, 217.188]

Критерій Фішера
Fp > Ft => 2.81 > 2.6571966002210865
Рівняння регресії неадекватно оригіналу при рівні значимості 0.05

---------------------------
Натуралізована матриця X
 [[-5.0e+00  1.0e+01  1.0e+01 -5.0e+01 -5.0e+01  1.0e+02 -5.0e+02]
 [-5.0e+00  1.0e+01  2.0e+01 -5.0e+01 -1.0e+02  2.0e+02 -1.0e+03]
 [-5.0e+00  6.0e+01  1.0e+01 -3.0e+02 -5.0e+01  6.0e+02 -3.0e+03]
 [-5.0e+00  6.0e+01  2.0e+01 -3.0e+02 -1.0e+02  1.2e+03 -6.0e+03]
 [ 1.5e+01  1.0e+01  1.0e+01  1.5e+02  1.5e+02  1.0e+02  1.5e+03]
 [ 1.5e+01  1.0e+01  2.0e+01  1.5e+02  3.0e+02  2.0e+02  3.0e+03]
 [ 1.5e+01  6.0e+01  1.0e+01  9.0e+02  1.5e+02  6.0e+02  9.0e+03]
 [ 1.5e+01  6.0e+01  2.0e+01  9.0e+02  3.0e+02  1.2e+03  1.8e+04]]

Матриця Y
 [[223. 212. 223.]
 [229. 213. 215.]
 [221. 231. 223.]
 [232. 217. 207.]
 [219. 206. 212.]
 [207. 222. 205.]
 [213. 232. 224.]
```

```
Середні значення функції відгуку за рядками: [219.333, 219.0, 225.0, 218.667, 212.333, 211.333, 223.0, 219.333]

Нормоване рівняння регресії: y = 218.50 -2.00*x1 +3.00*x2 -1.42*x3 +1.67*x12 +0.25*x13 -1.08*x23 +0.42*x123

Критерій Кохрена
Дисперсія: [26.889, 50.667, 18.667, 105.556, 28.222, 57.556, 60.667, 20.222]
Gp < Gt
0.2865 < 0.815948432359917 => дисперсія однорідна

Критерій Стюдента
Незначні коефіцієнти регресії
b1 = -2.00
b3 = -1.42
b4 = 1.67
b5 = 0.25
b6 = -1.08
b7 = 0.42
Значення функції відгуку зі значущими коефіцієнтами [215.5, 215.5, 221.5, 221.5, 215.5, 215.5, 221.5, 221.5]

Критерій Фішера
Fp < Ft => 0.89 < 2.741310828338778
Отримана математична модель при рівні значимості 0.05 адекватна експериментальним даним
```