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 №5

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів(центральний ортогональний композиційний план)»

Виконала:
студентка групи ІВ-81
Базова Л.Г.
Номер залікової книжки № 8101
Перевірив: Регіда П.Г.

Київ 2020 р.

Код програми:

```
import numpy as np
from scipy.stats import f, t
from tabulate import tabulate
import sklearn.linear_model as lm


def mult(x1, x2, x3 = np.ones(15)):
    x = np.ones(15)
    for i in range(15):
        x[i] *= x1[i] * x2[i] * x3[i];
    return x
def find_x(x_min, x_max):
  #величини для значень матриці планування
  x01 = (x_max[0] + x_min[0]) / 2
  x02 = (x_max[1] + x_min[1]) / 2
  x03 = (x_max[2] + x_min[2]) / 2
  delta_x1 = x_max[0] - x01
  delta_x2 = x_max[1] - x02
  delta_x3 = x_max[2] - x03
  X1 = np.array([x_min[0], x_min[0], x_min[0], x_min[0], x_max[0], x_max[0], x_max[0],
x_max[0], -l*delta_x1+x01, l*delta_x1+x01, x01, x01, x01, x01, x01])
  X2 = np.array([x_min[1], x_min[1], x_max[1], x_max[1], x_min[1], x_min[1], x_max[1],
x_max[1], x02, x02, -l*delta_x2+x02, l*delta_x2+x02, x02, x02, x02])
  X3 = np.array([x_min[2], x_max[2], x_min[2], x_max[2], x_min[2], x_max[2], x_min[2],
x_max[2], x03, x03, x03, x03, -l*delta_x3+x03, l*delta_x3+x03, x03])
  return np.array(list(zip(X1, X2, X3, mult(X1, X2), mult(X1, X3), mult(X2, X3),
mult(X1, X2, X3), mult(X1, X1), mult(X2, X2), mult(X3, X3))))
def find_y(x_min, x_max):
    # визначаємо ymax, ymin
    y_max = int(200 + x_max.mean())
    y_min = int(200 + x_min.mean())
    return np.random.randint(y_min, y_max, size=(N, m))
def find_b(X, y):
    x = list(X)
    for i in range(len(x)):
        x[i] = np.array([1, ] + list(x[i]))
    X = np.array(x)
    model = lm.LinearRegression(fit_intercept=False)
    model.fit(X, y)
    coefs = model.coef_
    print("Рівняння регресії")
    print(f"{coefs[0]:.3f} + x1 * {coefs[1]:.3f} + x2 * {coefs[2]:.3f} + x3 *
{coefs[3]:.3f}"
          f"+ x1x2 * {coefs[4]:.3f} + x1x3 * {coefs[5]:.3f} + x2x3 * {coefs[6]:.3f}"
          f"+ x1x2x3 * {coefs[7]:.4f} + x1^2 * {coefs[8]:.4f} + x2^2 * {coefs[9]:.4f} +
x3^2 * {coefs[10]:.4f}")
    return coefs
def find_disper(y, y_mean):
    disper = np.zeros(N)
    for i in range(N):
        for j in range(m):
            disper[i] += (y[i][j] - y_mean[i]) ** 2
        disper[i] /= m
    return disper
def matrix_print(y, x_list, y_mean, disper):
    global header_table
    header_table = ["№", "x1", "x2", "x3", "x1x2", "x1x3", "x2x3", "x1x2x3", "x1^2",
"x2^2", "x3^2"]
    table = []
```

```
    for i in range(N):
        table.append([i + 1])
    for i in range(N):
        for _ in range(len(x_list[0])):
            table[i].append(x_list[i][_])
        for j in range(m):
            table[i].append(y[i][j])
        table[i].append(y_mean[i])
        table[i].append(disper[i])
    for i in range(m):
        header_table.append("Y" + str(i + 1))
    header_table.append("Y")
    header_table.append("S^2")
    print(tabulate(table, headers=header_table, tablefmt="fancy_grid"))
def kohren_check(disper):
    global Gp, Gt, f1, f2
    print("Критерій Кохрена")
    Gp = max(disper) / sum(disper)
    f1 = m - 1
    f2 = N
    fisher = f.isf(*[q / f2, f1, (f2 - 1) * f1])
    Gt = round(fisher / (fisher + (f2 - 1)), 4)
    print("Gp = " + str(Gp) + ", Gt = " + str(Gt))
def student_check():
    global sb, d, f3, yy, t_exp
    d = len(x_code[0])
    print("Критерій Стьюдента")
    f3 = f1 * f2
    sb = sum(disper) / N
    ssbs = sb / N * m
    sbs = ssbs ** 0.5
    beta = np.zeros(d)
    t_exp = []
    for j in range(d):
        for i in range(N):
            if (j == 0):
                beta[j] += y_mean[i]
            else:
                beta[j] += y_mean[i] * x_code[i][j]
        beta[j] /= N
        t_exp.append(abs(beta[j]) / sbs)

    ttabl = round(abs(t.ppf(q / 2, f3)), 4)

    string_eq = f"y = {b[0]:.7f}"
    for i in range(len(t_exp)):
        if (t_exp[i] < ttabl):
            print(f"Коефіцієнт t{i:} = {t_exp[i]:.7f} не значимий")
            b[i] = 0
            d = d - 1
        else:
            print(f"Коефіцієнт t{i:} = {t_exp[i]:.7f} значимий")
            if(i != 0): string_eq += f" + {b[i]:.7f} * " + header_table[i]
    print("Значимих коефіцієнтів: d = ", d, "\nРівняння регресії після виключення
коефіцієнтів:\n", string_eq)
    yy = np.zeros(N)
    for row in range(len(x_list)):
        yy[row] += b[0]
        for el in range(len(x_list[row])):
            yy[row] += b[el + 1] * x_list[row][el]
```

```
def fisher_check():
    print("Критерій Фішера")
    f4 = N - d
    sad = 0
    for i in range(N):
        sad += (yy[i] - y_mean[i]) ** 2
    sad *= (m / (N - d))
    Fp = sad / sb
    print("Fp=", round(Fp, 2))
    Ft = round(abs(f.isf(q, f4, f3)), 4)
    print("Fp = " + str(round(Fp, 2)) + ", Ft = " + str(Ft))
    if Fp > Ft:
        print("Fp > Ft -> Рівняння неадекватне оригіналу")
    else:
        print("Fp < Ft -> Рівняння адекватне оригіналу")


#величини за варіантом:
x_min = np.array([-5, -5, -2])
x_max = np.array([8, 8, 6])
#рівняння з урахуванням квадратичних членів
print("y=b0+b1*x1+b2*x2+b3*x3+b12*x1*x2+b13*x1*x3+b23*x2*x3+b123*x1*x2*x3+b11*x1^2+b22*x
2^2+b33*x3^2+\n")
#константи для початкових умов
m = 3
k = 3 #const
p = 0
N = 15 #2^(k - p)+2k + N0
l = 1.215 #за формулою
q = 0.05
#матриця планування
x_list = find_x(x_min, x_max)
y = find_y(x_min, x_max)
x_code = find_x(np.array([-1, -1, -1]), np.array([1, 1, 1]))
while 1:
    if(m > 3):
        next_int = np.random.randint(y_min, y_max, size=(N, 1))
        y = np.append(y, next_int, axis=1)
    y_mean = np.sum(y, axis=1) / m
    disper = find_disper(y, y_mean)
    print("Матриця планування:")
    matrix_print(y, x_code, y_mean, disper)
    print("Натуралізована матриця:")
    matrix_print(y, x_list, y_mean, disper)

    b = find_b(x_list, y_mean)
    kohren_check(disper)         #find Gp, Gt, f1, f2
    if Gp > Gt:
        print("Дисперсія  неоднорідна , потрібно збільшити  m")
        m = m + 1
        continue
    print("Gp < Gt -> Дисперсія однорідна\n")
    student_check()
    fisher_check()
    break
```

Результат роботи програми

y=b0+b1*x1+b2*x2+b3*x3+b12*x1*x2+b13*x1*x3+b23*x2*x3+b123*x1*x2*x3+b11*x1^2+b22*x2^2+b33*x3^2+

Матриця планування:

№	x1	x2	x3	x1x2	x1x3	x2x3	x1x2x3	x1^2	x2^2	x3^2	Y1	Y2	Y3	Y	S^2
1	-1	-1	-1	1	1	1	-1	1	1	1	204	198	199	200.333	6.88889
2	-1	-1	1	1	-1	-1	1	1	1	1	201	206	199	202	8.66667
3	-1	1	-1	-1	1	-1	1	1	1	1	198	201	196	198.333	4.22222
4	-1	1	1	-1	-1	1	-1	1	1	1	203	204	200	202.333	2.88889
5	1	-1	-1	-1	-1	1	1	1	1	1	204	204	204	204	0
6	1	-1	1	-1	1	-1	-1	1	1	1	196	201	199	198.667	4.22222
7	1	1	-1	1	-1	-1	-1	1	1	1	198	198	201	199	2
8	1	1	1	1	1	1	1	1	1	1	205	201	196	200.667	13.5556
9	-1.215	0	0	-0	-0	0	-0	1.47623	0	0	196	200	204	200	10.6667
10	1.215	0	0	0	0	0	0	1.47623	0	0	202	202	204	202.667	0.888889
11	0	-1.215	0	-0	0	-0	-0	0	1.47623	0	201	204	201	202	2
12	0	1.215	0	0	0	0	0	0	1.47623	0	198	198	205	200.333	10.8889
13	0	0	-1.215	0	-0	-0	-0	0	0	1.47623	205	197	203	201.667	11.5556
14	0	0	1.215	0	0	0	0	0	0	1.47623	200	201	198	199.667	1.55556
15	0	0	0	0	0	0	0	0	0	0	204	203	198	201.667	6.88889

Натуралізована матриця:

№	x1	x2	x3	x1x2	x1x3	x2x3	x1x2x3	x1^2	x2^2	x3^2	Y1	Y2	Y3	Y	S^2
1	-5	-5	-2	25	10	10	-50	25	25	4	204	198	199	200.333	6.88889
2	-5	-5	6	25	-30	-30	150	25	25	36	201	206	199	202	8.66667
3	-5	8	-2	-40	10	-16	80	25	64	4	198	201	196	198.333	4.22222
4	-5	8	6	-40	-30	48	-240	25	64	36	203	204	200	202.333	2.88889
5	8	-5	-2	-40	-16	10	80	64	25	4	204	204	204	204	0
6	8	-5	6	-40	48	-30	-240	64	25	36	196	201	199	198.667	4.22222
7	8	8	-2	64	-16	-16	-128	64	64	4	198	198	201	199	2
8	8	8	6	64	48	48	384	64	64	36	205	201	196	200.667	13.5556
9	-6.3975	1.5	2	-9.59625	-12.795	3	-19.1925	40.928	2.25	4	196	200	204	200	10.6667
10	9.3975	1.5	2	14.0963	18.795	3	28.1925	88.313	2.25	4	202	202	204	202.667	0.888889
11	1.5	-6.3975	2	-9.59625	3	-12.795	-19.1925	2.25	40.928	4	201	204	201	202	2
12	1.5	9.3975	2	14.0963	3	18.795	28.1925	2.25	88.313	4	198	198	205	200.333	10.8889
13	1.5	1.5	-2.86	2.25	-4.29	-4.29	-6.435	2.25	2.25	8.1796	205	197	203	201.667	11.5556
14	1.5	1.5	6.86	2.25	10.29	10.29	15.435	2.25	2.25	47.0596	200	201	198	199.667	1.55556
15	1.5	1.5	2	2.25	3	3	4.5	2.25	2.25	4	204	203	198	201.667	6.88889

```
Рівняння регресії
201.459 + x1 * 0.150 + x2 * -0.152 + x3 * 0.137+ x1x2 * -0.011 + x1x3 * -0.050 + x2x3 * 0.040+ x1x2x3 * 0.0035 + x1^2 * -0.0025 + x2^2 * -0.0052 + x3^2 * -0.0349
Критерій Кохрена
Gp = 0.15601023017902815, Gt = 0.3346
Gp < Gt -> Дисперсія однорідна

Критерій Стьюдента
Коефіцієнт t0 = 186.6398714 значимий
Коефіцієнт t1 = 0.4144685 не значимий
Коефіцієнт t2 = 0.0266333 не значимий
Коефіцієнт t3 = 0.0825840 не значимий
Коефіцієнт t4 = 0.5780881 не значимий
Коефіцієнт t5 = 0.5780881 не значимий
Коефіцієнт t6 = 0.2890440 не значимий
Коефіцієнт t7 = 136.2487537 значимий
Коефіцієнт t8 = 136.2182755 значимий
Коефіцієнт t9 = 136.1268411 значимий
Значимих коефіцієнтів: d =  4
Рівняння регресії після виключення коефіцієнтів:
 y = 201.4589727 + 0.0034517 * x1x2x3 + -0.0025143 * x1^2 + -0.0051865 * x2^2
Критерій Фішера
Fp= 1.73
Fp = 1.73, Ft = 2.1256
Fp < Ft -> Рівняння адекватне оригіналу
```