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 рівняння регресії з урахуванням ефекту взаємодії.»

Виконав:
Студент групи ІО-92
Педенко Данило Денисович
Перевірив:
ас. Регіда П. Г.

Київ
2021 р.

Лабораторна робота № 4

Тема: Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії.

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

Завдання:

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.

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

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

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

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.

Завдання відповідно за номером варіанту:

№варіанта	x_1		x_2		x_3	
	min	max	min	Max	min	max
215	-25	75	-20	60	-25	-10

Роздруківка тексту програми:

```
import numpy as np
import random as ra
import math as ma
from scipy.stats import f
import sklearn.linear_model as slm
from prettytable import PrettyTable


def cochrane(eq, n, list_y, list_av_y, list_x, koef, list_xn):
    i = 0
    f1 = eq - 1
    f2 = n
    list_g = [9065, 7679, 6841, 6287, 5892, 5598, 5365, 5175, 5017, 4884]
    list_sig = []
    for i in range(len(list_y)):
        tem = 0
        for j in range(len(list_y[i])):
            tem += pow(list_y[i][j] - list_av_y[i], 2)
```

```
        list_sig.append(tem / len(list_y[i]))
    gp = max(list_sig) / sum(list_sig)
    print("F1 = ", f1)
    print("F2 = ", f2)
    print("q = 0.05")
    print("Значення дисперсій по рядках")
    print(list_sig)
    print("\nGp = ", gp)
    for i in range(len(list_g)):
        if i == f1 - 1:
            if gp < list_g[i] / 10000:
                print("\nGp = {0} < Gt = {1}".format(gp, list_g[i] / 10000))
                print("Дисперсія однорідна\n")
                print("Оцінимо значимість коефіцієнтів регресії згідно критерію
Стьюдента")
                if student(eq, n, list_sig, list_av_y, list_x, koef, list_xn):
                    return True
                else:
                    return False
            else:
                print("Дисперсія не однорідна")


def student(eq, n, list_sig, list_av_y, list_x, koef, list_xn):
    list_t_prover = [12.71, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306, 2.262,
2.228, 2.201, 2.179, 2.160, 2.145,
                     2.131, 2.120, 2.110, 2.101, 2.093, 2.086, 2.080, 2.074, 2.069,
2.064, 2.060, 2.056, 2.052, 2.048,
                     2.045, 2.042]
    sv = sum(list_sig) / len(list_sig)
    s_sq_beta = sv / (n * eq)
    s_beta = ma.sqrt(s_sq_beta)
    list_beta = []
    new_koef = []
    no_matter_koef = []
    for i in range(len(list_x)):
        pol = 0
        for j in range(len(list_x[i])):
            pol += list_x[i][j] * list_av_y[j]
        list_beta.append(pol / len(list_av_y))
    print("m = ", eq)
    print("N = ", n)
    print("Отримані значення βi")
    print(list_beta)
    list_t = [abs(list_beta[i]) / s_beta for i in range(len(list_beta))]
    print("Отримані значення ti")
    print(list_t)
    print("\nf3 = ", (eq - 1) * n)
    print("q = 0.05")
    for i in range(len(list_t_prover)):
        if i == (eq - 1) * n - 1:
            for j in range(len(list_t)):
                if list_t[j] < list_t_prover[i]:
                    print("\nt{0} = {1} < ттабл = {2}".format(j, list_t[j],
list_t_prover[i]))
                    print("b{0} - виключається з рівняння".format(j))
                    no_matter_koef.append([j, koef[j]])
                else:
                    print("\nt{0} = {1} > ттабл = {2}".format(j, list_t[j],
list_t_prover[i]))
                    new_koef.append([j, koef[j]])
    print("\nПерепишемо рівняння враховуючи вилучених коефіцієнтів")
```

```
    print(vivod(new_koef))
    print("\nРівняння з використанням незначимих коефіцієнтів")
    print(vivod(no_matter_koef))
    list_res_y = []
    print("\nПідставимо необхідні значення X")
    for i in range(len(list_xn)):
        pol = 0
        for j in range(len(new_koef)):
            if new_koef[j][0] == 0:
                pol += new_koef[j][1]
            else:
                pol += list_xn[i][new_koef[j][0] - 1] * new_koef[j][1]
        list_res_y.append(pol)
        print("y{0} = {1}".format(i, pol))
    print("\nКритерій Фішера")
    if fisher(len(new_koef), n, eq, list_res_y, list_av_y, sv):
        return True
    else:
        return False


def fisher(d, n, eq, list_res_y, list_av_y, sv):
    f4 = n - d
    f3 = (eq - 1) * n
    temp = 0
    print("d = ", d)
    print("f3 = ", f3)
    print("f4 = ", f4)
    print("q = 0.05")
    for i in range(n):
        temp += pow((list_res_y[i] - list_av_y[i]), 2)
    sad = temp * (eq / (n - d))
    fp = sad / sv
    ft = f.ppf(q=1 - 0.05, dfn=f4, dfd=f3)
    if fp > ft:
        print("\nFp = {0} > Ft = {1}".format(fp, ft))
        print("Рівняння регресії неадекватно оригіналу")
        return True
    else:
        print("\nFp = {0} < Ft = {1}".format(fp, ft))
        print("Рівняння регресії адекватно оригіналу")
        return False


def vivod(ar):
    rivn = "y = "
    for i in range(len(ar)):
        if ar[i][0] == 0:
            rivn += str(ar[i][1])
        elif i == 0:
            rivn += str(ar[i][1]) + " * x{0}".format(ar[i][0])
        else:
            rivn += " + " + str(ar[i][1]) + " * x{0}".format(ar[i][0])
    return rivn


def riv_koef(list_x, list_y):
    skm = slm.LinearRegression(fit_intercept=False)
    skm.fit(list_x, list_y)
    B = skm.coef_
    B = [round(i, 4) for i in B]
    return B
```

```
def main1(m, n):
    array_xd = [[-25, 75], [-20, 60], [-25, -10]]
    array_yd = [round(200 + (max(array_xd[0]) + max(array_xd[1]) + max(array_xd[2]))
/ len(array_xd)),
                round(200 + (min(array_xd[0]) + min(array_xd[1]) + min(array_xd[2]))
/ len(array_xd))]
    array_xp = 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]])
    array_y = [[ra.randint(array_yd[1], array_yd[0]) for j in range(m)] for i in
range(n)]
    array_aver_y = [sum(array_y[i]) / len(array_y[i]) for i in range(len(array_y))]
    array_xn = []
    array_a = []
    array_aii = []
    array_aij = []
    for i in range(len(array_xp)):
        array_xn.append([])
        for j in range(len(array_xd)):
            if array_xp[i][j + 1] == -1:
                array_xn[i].append(min(array_xd[j]))
            else:
                array_xn[i].append(max(array_xd[j]))
    trans = np.array(array_xn).transpose()
    array_mx = np.array([sum(trans[i]) / len(trans[i]) for i in range(len(trans))])
    my = sum(array_aver_y) / len(array_aver_y)
    for i in range(len(trans)):
        summa = 0
        kvad = 0
        poly = 0
        for j in range(len(trans[i])):
            summa += trans[i][j] * array_aver_y[j]
            kvad += pow(trans[i][j], 2)
            if i == 0:
                poly += trans[i][j] * trans[i + 1][j]
            elif i == 1:
                poly += trans[0][j] * trans[2][j]
            else:
                poly += trans[1][j] * trans[2][j]
        array_a.append(summa / len(trans[i]))
        array_aii.append(kvad / len(trans[i]))
        array_aij.append(poly / len(trans[i]))
    ta = PrettyTable()
    ta.field_names = ["X0", "X1", "X2", "X3", "Y1", "Y2", "Y3"]
    for i in range(n):
        ta.add_row(list(array_xp[i]) + array_y[i])
    ta1 = PrettyTable()
    ta1.field_names = ["X1", "X2", "X3", "Y1", "Y2", "Y3"]
    for i in range(n):
        ta1.add_row(array_xn[i] + array_y[i])
    print("Матриця планування експерименту")
    print(ta)
    print("Матриця планування експерименту для натуралізованих значень при m = 3")
    print(ta1)
```

```
    print("Середні значення функції відгуку за рядками")
    print(array_aver_y)
    res = np.linalg.solve(
        [[1, array_mx[0], array_mx[1], array_mx[2]], [array_mx[0], array_aii[0],
array_aij[0], array_aij[1]],
         [array_mx[1], array_aij[0], array_aii[1], array_aij[2]],
         [array_mx[2], array_aij[1], array_aij[2], array_aii[2]]],
        [my, array_a[0], array_a[1], array_a[2]])
    print("Значення коефіцієнтів")
    print(res)
    print("\nРівняння регресії")
    print("{0} + ({1}) * x1 + ({2}) * x2 + ({3}) * x3\n".format(round(res[0], 3),
round(res[1], 3), round(res[2], 3),
                                                                round(res[3], 3)))
    print("\nПеревірка однорідності дисперсії за критерієм Кохрена")
    if cochrane(m, 8, array_y, array_aver_y, array_xp.transpose(), res, array_xn):
        array_xp_full = []
        array_xn_full = []
        for i in range(len(array_xp)):
            array_xp_full.append(list(array_xp[i]))
            array_xp_full[i].append(array_xp[i][1] * array_xp[i][2])
            array_xp_full[i].append(array_xp[i][1] * array_xp[i][3])
            array_xp_full[i].append(array_xp[i][2] * array_xp[i][3])
            array_xp_full[i].append(array_xp[i][1] * array_xp[i][2] * array_xp[i][3])
            array_xn_full.append(list(array_xn[i]))
            array_xn_full[i].append(array_xn[i][0] * array_xn[i][1])
            array_xn_full[i].append(array_xn[i][0] * array_xn[i][2])
            array_xn_full[i].append(array_xn[i][1] * array_xn[i][2])
            array_xn_full[i].append(array_xn[i][0] * array_xn[i][1] * array_xn[i][2])
        ta_full = PrettyTable()
        ta_full.field_names = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3",
"X1X2X3", "Y1", "Y2", "Y3"]
        for i in range(n):
            ta_full.add_row(array_xp_full[i] + array_y[i])
        ta1_full = PrettyTable()
        ta1_full.field_names = ["X1", "X2", "X3", "X1X2", "X1X3", "X2X3", "X1X2X3",
"Y1", "Y2", "Y3"]
        for i in range(n):
            ta1_full.add_row(array_xn_full[i] + array_y[i])
        res_full = riv_koef(array_xp_full, array_aver_y)
        print("\nВрахуємо ефект взаємодії\n")
        print("Матриця ПФЕ")
        print(ta_full)
        print("\nМатриця ПФЕ для натуралізованих значень при m = 3")
        print(ta1_full)
        print("Значення коефіцієнтів рівняння регресії")
        print(res_full)
        print("\nРівняння регресії")
        print("{0} + ({1}) * x1 + ({2}) * x2 + ({3}) * x3 + ({3}) * x4 + ({4}) * x5 +
({5}) * x6 + ({6}) * x7\n".format(
            round(res_full[0], 3), round(res_full[1], 3), round(res_full[2], 3),
round(res_full[3], 3),
            round(res_full[4], 3), round(res_full[5], 3), round(res_full[6], 3)))
        print("\nПеревірка однорідності дисперсії за критерієм Кохрена")
        if cochrane(m, 8, array_y, array_aver_y, np.array(array_xp_full).transpose(),
res_full, array_xn_full):
            print("\nПерезапускаємо програму\n")
            main1(m, n)


main1(3, 8)
```

Роздруківка результатів роботи програми:

```
Матриця планування експерименту
+----+----+----+----+-----+-----+-----+
| X0 | X1 | X2 | X3 |  Y1 |  Y2 |  Y3 |
+----+----+----+----+-----+-----+-----+
| 1  | -1 | -1 | -1 | 201 | 226 | 181 |
| 1  | -1 | 1  | 1  | 216 | 205 | 221 |
| 1  | 1  | -1 | 1  | 186 | 227 | 184 |
| 1  | 1  | 1  | -1 | 230 | 222 | 178 |
| 1  | -1 | -1 | 1  | 190 | 195 | 213 |
| 1  | -1 | 1  | -1 | 242 | 226 | 238 |
| 1  | 1  | -1 | -1 | 216 | 182 | 219 |
| 1  | 1  | 1  | 1  | 183 | 177 | 209 |
+----+----+----+----+-----+-----+-----+
Матриця планування експерименту для натуралізованих значень при m = 3
+-----+-----+-----+-----+-----+-----+
|  X1 |  X2 |  X3 |  Y1 |  Y2 |  Y3 |
+-----+-----+-----+-----+-----+-----+
| -25 | -20 | -25 | 201 | 226 | 181 |
| -25 |  60 | -10 | 216 | 205 | 221 |
|  75 | -20 | -10 | 186 | 227 | 184 |
|  75 |  60 | -25 | 230 | 222 | 178 |
| -25 | -20 | -10 | 190 | 195 | 213 |
| -25 |  60 | -25 | 242 | 226 | 238 |
|  75 | -20 | -25 | 216 | 182 | 219 |
|  75 |  60 | -10 | 183 | 177 | 209 |
+-----+-----+-----+-----+-----+-----+
Середні значення функції відгуку за рядками
[202.66666666666666, 214.0, 199.0, 210.0, 199.33333333333334, 235.33333333333334, 205.66666666666666, 189.66666666666666]
Значення коефіцієнтів
[ 1.92180556e+02 -1.17500000e-01  1.32291667e-01 -8.61111111e-01]

Рівняння регресії
192.181 + (-0.117) * x1 + (0.132) * x2 + (-0.861) * x3


Перевірка однорідності дисперсії за критерієм Кохрена
F1 =  2
F2 =  8
q = 0.05
Значення дисперсій по рядках
[338.8888888888889, 44.666666666666664, 392.6666666666667, 522.6666666666666, 97.55555555555554, 46.22222222222222, 281.55555555555554, 192.8888888888889]

Gp =  0.2726324330590008
```

```
Gp = 0.2726324330590008 < Gt = 0.7679
Дисперсія однорідна

Оцінимо значимість коефіцієнтів регресії згідно критерію Стьюдента
m =  3
N =  8
Отримані значення βi
[206.95833333333334, -5.875, 5.291666666666668, -6.458333333333332]
Отримані значення ti
[65.49526301361553, 1.8592373837164864, 1.674632253418396, 2.043842514014577]

f3 =  16
q = 0.05

t0 = 65.49526301361553 > ттабл = 2.12

t1 = 1.8592373837164864 < ттабл = 2.12
b1 - виключається з рівняння

t2 = 1.674632253418396 < ттабл = 2.12
b2 - виключається з рівняння

t3 = 2.043842514014577 < ттабл = 2.12
b3 - виключається з рівняння

Перепишемо рівняння враховуючи вилучених коефіцієнтів
y = 192.18055555555551

Рівняння з використанням незначимих коефіцієнтів
y = -0.1174999999999997 * x1 + 0.13229166666666695 * x2 + -0.8611111111111136 * x3

Підставимо необхідні значення X
y0 = 192.18055555555551
y1 = 192.18055555555551
y2 = 192.18055555555551
y3 = 192.18055555555551
y4 = 192.18055555555551
y5 = 192.18055555555551
y6 = 192.18055555555551
y7 = 192.18055555555551

Критерій Фішера
d =  1
f3 =  16
```

```
f4 =  7
q = 0.05

Fp = 5.457503298062028 > Ft = 2.6571966002210865
Рівняння регресії неадекватно оригіналу

Врахуємо ефект взаємодії

Матриця ПФЕ
+----+----+----+----+------+------+------+--------+-----+-----+-----+
| X0 | X1 | X2 | X3 | X1X2 | X1X3 | X2X3 | X1X2X3 |  Y1 |  Y2 |  Y3 |
+----+----+----+----+------+------+------+--------+-----+-----+-----+
| 1  | -1 | -1 | -1 |  1   |  1   |  1   |   -1   | 201 | 226 | 181 |
| 1  | -1 | 1  | 1  |  -1  |  -1  |  1   |   -1   | 216 | 205 | 221 |
| 1  | 1  | -1 | 1  |  -1  |  1   |  -1  |   -1   | 186 | 227 | 184 |
| 1  | 1  | 1  | -1 |  1   |  -1  |  -1  |   -1   | 230 | 222 | 178 |
| 1  | -1 | -1 | 1  |  1   |  -1  |  -1  |   1    | 190 | 195 | 213 |
| 1  | -1 | 1  | -1 |  -1  |  1   |  -1  |   1    | 242 | 226 | 238 |
| 1  | 1  | -1 | -1 |  -1  |  -1  |  1   |   1    | 216 | 182 | 219 |
| 1  | 1  | 1  | 1  |  1   |  1   |  1   |   1    | 183 | 177 | 209 |
+----+----+----+----+------+------+------+--------+-----+-----+-----+

Матриця ПФЕ для натуралізованих значень при m = 3
+-----+-----+-----+-------+-------+-------+---------+-----+-----+-----+
|  X1 |  X2 |  X3 |  X1X2 |  X1X3 |  X2X3 |  X1X2X3 |  Y1 |  Y2 |  Y3 |
+-----+-----+-----+-------+-------+-------+---------+-----+-----+-----+
| -25 | -20 | -25 |  500  |  625  |  500  |  -12500 | 201 | 226 | 181 |
| -25 |  60 | -10 | -1500 |  250  |  -600 |  15000  | 216 | 205 | 221 |
|  75 | -20 | -10 | -1500 |  -750 |  200  |  15000  | 186 | 227 | 184 |
|  75 |  60 | -25 |  4500 | -1875 | -1500 | -112500 | 230 | 222 | 178 |
| -25 | -20 | -10 |  500  |  250  |  200  |  -5000  | 190 | 195 | 213 |
| -25 |  60 | -25 | -1500 |  625  | -1500 |  37500  | 242 | 226 | 238 |
|  75 | -20 | -25 | -1500 | -1875 |  500  |  37500  | 216 | 182 | 219 |
|  75 |  60 | -10 |  4500 |  -750 |  -600 |  -45000 | 183 | 177 | 209 |
+-----+-----+-----+-------+-------+-------+---------+-----+-----+-----+
Значення коефіцієнтів рівняння регресії
[206.9583, -5.875, 5.2917, -6.4583, -6.5417, -0.2917, -3.9583, 0.5417]

Рівняння регресії
206.958 + (-5.875) * x1 + (5.292) * x2 + (-6.458) * x3 + (-6.458) * x4 + (-6.542) * x5 + (-0.292) * x6 + (-3.958) * x7
```

```
Перевірка однорідності дисперсії за критерієм Кохрена
F1 =  2
F2 =  8
q = 0.05
Значення дисперсій по рядках
[338.8888888888889, 44.666666666666664, 392.6666666666667, 522.6666666666666, 97.55555555555554, 46.22222222222222, 281.55555555555554, 192.8888888888889]

Gp =  0.2726324330590008

Gp = 0.2726324330590008 < Gt = 0.7679
Дисперсія однорідна

Оцінимо значимість коефіцієнтів регресії згідно критерію Стьюдента
m =  3
N =  8
Отримані значення βi
[206.95833333333334, -5.875, 5.291666666666668, -6.458333333333332, -6.541666666666668, -0.29166666666666785, -3.958333333333343, 0.5416666666666785]
Отримані значення ti
[65.49526301361553, 1.8592373837164864, 1.674632253418396, 2.043842514014577, 2.0702146754857336, 0.0923025651490458, 1.2526776698799051, 0.1714190495625167]

f3 =  16
q = 0.05

t0 = 65.49526301361553 > ttабл = 2.12

t1 = 1.8592373837164864 < ttабл = 2.12
b1 - виключається з рівняння

t2 = 1.674632253418396 < ttабл = 2.12
b2 - виключається з рівняння

t3 = 2.043842514014577 < ttабл = 2.12
b3 - виключається з рівняння

t4 = 2.0702146754857336 < ttабл = 2.12
b4 - виключається з рівняння

t5 = 0.0923025651490458 < ttабл = 2.12
b5 - виключається з рівняння

t6 = 1.2526776698799051 < ttабл = 2.12
b6 - виключається з рівняння
```

```
t7 = 0.1714190495625167 < ттабл = 2.12
b7 - виключається з рівняння

Перепишемо рівняння враховуючи вилучених коефіцієнтів
y = 206.9583

Рівняння з використанням незначимих коефіцієнтів
y = -5.875 * x1 + 5.2917 * x2 + -6.4583 * x3 + -6.5417 * x4 + -0.2917 * x5 + -3.9583 * x6 + 0.5417 * x7

Підставимо необхідні значення X
y0 = 206.9583
y1 = 206.9583
y2 = 206.9583
y3 = 206.9583
y4 = 206.9583
y5 = 206.9583
y6 = 206.9583
y7 = 206.9583

Критерій Фішера
d =  1
f3 =  16
f4 =  7
q = 0.05

Fp = 2.333049065243009 < Ft = 2.6571966002210865
Рівняння регресії адекватно оригіналу

Process finished with exit code 0
```

Висновок:

В ході лабораторної роботи, було проведено повний трьохфакторний експеримент. Знайдено рівняння регресії адекватне об'єкту. Під час виконання лабораторної роботи проблем не виникло, що підтверджують дані наведені вище