Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №5

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

Виконав:

студент II курсу ФІОТ

групи ІО-93

Сукач Артем

Перевірив:

Регіда П.Г.

Київ – 2021

Мета роботи: Провести трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Завдання на лабораторну роботу:

1. Взяти рівняння з урахуванням квадратичних членів.

2. Скласти матрицю планування для ОЦКП

3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі.

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

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

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

4. Розрахувати коефіцієнти рівняння регресії і записати його.

5. Провести 3 статистичні перевірки.

Варіант завдання:

Варіант	X_1		X_2		X_3	
	min	max	min	max	min	max
325	-5	9	-8	10	-2	6

Довірча ймовірність дорівнює 0.95, а рівень значимості q = 0.05.

Роздруківка тексту програми:

```
import random

import numpy as np

import sklearn.linear_model as lm
```

```
from scipy.stats import f, t
from functools import partial
from pyDOE2 import *

x_range = ((-5, 9), (-8, 10), (-2, 6))

x_aver_max = sum([x[1] for x in x_range]) / 3
x_aver_min = sum([x[0] for x in x_range]) / 3

y_max = 200 + int(x_aver_max)
y_min = 200 + int(x_aver_min)


def regression(x, b):
    y = sum([x[i] * b[i] for i in range(len(x))])
    return y


def s_kv(y, y_aver, n, m):
    res = []
    for i in range(n):
        s = sum([(y_aver[i] - y[i][j]) ** 2 for j in range(m)]) / m
        res.append(round(s, 3))
    return res


def plan_matrix5(n, m):
    print('\nЛабораторна 5')
    print("\nРівняння регресії з урахуванням квадратичних членів:")
    print(
```

```
    "ŷ = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13*x1*x3 + b23*x2*x3 + b123*x1*x2*x3 + b11x1^2 + b22x2^2 + b33x3^2\n")

  print(f'\nГереруємо матрицю планування для n = {n}, m = {m}')


  y = np.zeros(shape=(n, m))

  for i in range(n):

    for j in range(m):

      y[i][j] = random.randint(y_min, y_max)


  if n > 14:

    no = n - 14

  else:

    no = 1

  x_norm = ccdesign(3, center=(0, no))

  x_norm = np.insert(x_norm, 0, 1, axis=1)


  for i in range(4, 11):

    x_norm = np.insert(x_norm, i, 0, axis=1)


  l = 1.215


  for i in range(len(x_norm)):

    for j in range(len(x_norm[i])):

      if x_norm[i][j] < -1 or x_norm[i][j] > 1:

        if x_norm[i][j] < 0:

          x_norm[i][j] = -l

        else:

          x_norm[i][j] = l


  def add_sq_nums(x):
```

```
    for i in range(len(x)):
        x[i][4] = x[i][1] * x[i][2]
        x[i][5] = x[i][1] * x[i][3]
        x[i][6] = x[i][2] * x[i][3]
        x[i][7] = x[i][1] * x[i][3] * x[i][2]
        x[i][8] = x[i][1] ** 2
        x[i][9] = x[i][2] ** 2
        x[i][10] = x[i][3] ** 2
    return x


x_norm = add_sq_nums(x_norm)


x = np.ones(shape=(len(x_norm), len(x_norm[0])), dtype=np.int64)
for i in range(8):
    for j in range(1, 4):
        if x_norm[i][j] == -1:
            x[i][j] = x_range[j - 1][0]
        else:
            x[i][j] = x_range[j - 1][1]


for i in range(8, len(x)):
    for j in range(1, 3):
        x[i][j] = (x_range[j - 1][0] + x_range[j - 1][1]) / 2


dx = [x_range[i][1] - (x_range[i][0] + x_range[i][1]) / 2 for i in range(3)]


x[8][1] = l * dx[0] + x[9][1]
x[9][1] = -l * dx[0] + x[9][1]
x[10][2] = l * dx[1] + x[9][2]
x[11][2] = -l * dx[1] + x[9][2]
```

```
    x[12][3] = l * dx[2] + x[9][3]
    x[13][3] = -l * dx[2] + x[9][3]

    x = add_sq_nums(x)

    print('\nX:\n', x)
    print('\nX нормоване:\n')
    for i in x_norm:
        print([round(x, 2) for x in i])
    print('\nY:\n', y)

    return x, y, x_norm


def find_coef(X, Y, norm=False):
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(X, Y)
    B = skm.coef_

    if norm == 1:
        print('\nКоефіцієнти рівняння регресії з нормованими X:')
    else:
        print('\nКоефіцієнти рівняння регресії:')
    B = [round(i, 3) for i in B]
    print(B)
    print('\nРезультат рівняння зі знайденими коефіцієнтами:\n', np.dot(X, B))
    return B


def kriteriy_cochrana(y, y_aver, n, m):
```

```
    f1 = m - 1
    f2 = n
    q = 0.05
    S_kv = s_kv(y, y_aver, n, m)
    Gp = max(S_kv) / sum(S_kv)
    print('\nПеревірка за критерієм Кохрена')
    return Gp


def cohren(f1, f2, q=0.05):
    q1 = q / f1
    fisher_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
    return fisher_value / (fisher_value + f1 - 1)


# оцінки коефіцієнтів
def bs(x, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]

    for i in range(len(x[0])):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_aver)) / n
        res.append(b)
    return res


def kriteriy_studenta(x, y, y_aver, n, m):
    S_kv = s_kv(y, y_aver, n, m)
    s_kv_aver = sum(S_kv) / n

    # статистична оцінка дисперсії
```

```
    s_Bs = (s_kv_aver / n / m) ** 0.5  # статистична оцінка дисперсії
    Bs = bs(x, y_aver, n)
    ts = [round(abs(B) / s_Bs, 3) for B in Bs]

    return ts


def kriteriy_fishera(y, y_aver, y_new, n, m, d):
    S_ad = m / (n - d) * sum([(y_new[i] - y_aver[i]) ** 2 for i in range(len(y))])
    S_kv = s_kv(y, y_aver, n, m)
    S_kv_aver = sum(S_kv) / n

    return S_ad / S_kv_aver


def check(X, Y, B, n, m):
    print('\n\tПеревірка рівняння:')
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05

    ### табличні значення
    student = partial(t.ppf, q=1 - q)
    t_student = student(df=f3)

    G_kr = cohren(f1, f2)
    ###

    y_aver = [round(sum(i) / len(i), 3) for i in Y]
```

```
print('\nСереднє значення y:', y_aver)


disp = s_kv(Y, y_aver, n, m)
print('Дисперсія y:', disp)


Gp = kriteriy_cochrana(Y, y_aver, n, m)
print(f'Gp = {Gp}')
if Gp < G_kr:
    print(f'З ймовірністю {1 - q} дисперсії однорідні.')
else:
    print("Необхідно збільшити кількість дослідів")
    m += 1
    main(n, m)


ts = kriteriy_studenta(X[:, 1:], Y, y_aver, n, m)
print('\nКритерій Стьюдента:\n', ts)
res = [t for t in ts if t > t_student]
final_k = [B[i] for i in range(len(ts)) if ts[i] in res]
print('\nКоефіцієнти {} статистично незначущі, тому ми виключаємо їх з рівняння.'.format(
    [round(i, 3) for i in B if i not in final_k]))


y_new = []
for j in range(n):
    y_new.append(regression([X[j][i] for i in range(len(ts)) if ts[i] in res], final_k))


print(f'\nЗначення "y" з коефіцієнтами {final_k}')
print(y_new)


d = len(res)
if d >= n:
```

```
        print('\nF4 <= 0')
        print('')
        return
    f4 = n - d

    F_p = kriteriy_fishera(Y, y_aver, y_new, n, m, d)

    fisher = partial(f.ppf, q=0.95)
    f_t = fisher(dfn=f4, dfd=f3)  # табличне знач
    print('\nПеревірка адекватності за критерієм Фішера')
    print('Fp =', F_p)
    print('F_t =', f_t)
    if F_p < f_t:
        print('Математична модель адекватна експериментальним даним')
    else:
        print('Математична модель не адекватна експериментальним даним\nНеобхідно збільшити кількість дослідів')


def main(n, m):
    X5, Y5, X5_norm = plan_matrix5(n, m)

    y5_aver = [round(sum(i) / len(i), 3) for i in Y5]
    B5 = find_coef(X5, y5_aver)

    check(X5_norm, Y5, B5, n, m)


if __name__ == '__main__':
    main(15, 3)
```

Результати роботи програми:

```
Рівняння регресії з урахуванням квадратичних членів:
ŷ = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13*x1*x3 + b23*x2*x3 + b123*x1*x2*x3 + b11x1^2 + b22x2^2 + b33x3^2


Гереруємо матрицю планування для n = 15, m = 3

X:
 [[   1   -5   -8   -2   40   10   16  -80   25   64    4]
 [   1    9   -8   -2  -72  -18   16  144   81   64    4]
 [   1   -5   10   -2  -50   10  -20  100   25  100    4]
 [   1    9   10   -2   90  -18  -20 -180   81  100    4]
 [   1   -5   -8    6   40  -30  -48  240   25   64   36]
 [   1    9   -8    6  -72   54  -48 -432   81   64   36]
 [   1   -5   10    6  -50  -30   60 -300   25  100   36]
 [   1    9   10    6   90   54   60  540   81  100   36]
 [   1   10    1    1   10   10    1   10  100    1    1]
 [   1   -6    1    1   -6   -6    1   -6   36    1    1]
 [   1    2   11    1   22    2   11   22    4  121    1]
 [   1    2   -9    1  -18    2   -9  -18    4   81    1]
 [   1    2    1    5    2   10    5   10    4    1   25]
 [   1    2    1   -3    2   -6   -3   -6    4    1    9]
 [   1    2    1    1    2    2    1    2    4    1    1]]
```

```
X нормоване:

[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, -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, 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, 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, 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, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, -1.22, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 1.48, 0.0, 0.0]
[1.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0, 0.0]
[1.0, 0.0, -1.22, 0.0, -0.0, 0.0, -0.0, -0.0, 0.0, 1.48, 0.0]
[1.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0]
[1.0, 0.0, 0.0, -1.22, 0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 1.48]
[1.0, 0.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Y:
 [[198. 199. 198.]
 [205. 196. 202.]
 [196. 197. 201.]
 [206. 198. 196.]
 [201. 202. 204.]
 [197. 201. 207.]
 [201. 197. 205.]
 [198. 208. 198.]
 [195. 207. 199.]
 [205. 201. 195.]
 [199. 198. 207.]
 [200. 198. 201.]
 [195. 204. 200.]
 [197. 206. 195.]
 [196. 204. 205.]]
```

```
Коефіцієнти рівняння регресії:
[199.947, 0.096, -0.008, 0.363, -0.001, -0.022, -0.005, 0.001, -0.0, 0.002, -0.023]

Результат рівняння зі знайденими коефіцієнтами:
 [198.421 200.717 198.799 200.339 202.109 201.045 201.047 201.243 201.016
 199.832 200.534 200.714 201.136 198.976 200.424]

    Перевірка рівняння:

Середнє значення y: [198.333, 201.0, 198.0, 200.0, 202.333, 201.667, 201.0, 201.333, 200.333, 200.333, 201.333, 199.667, 199.667, 199.333, 201.667]
Дисперсія y: [0.222, 14.0, 4.667, 18.667, 1.556, 16.889, 10.667, 22.222, 24.889, 16.889, 16.222, 1.556, 13.556, 22.889, 16.222]

Перевірка за критерієм Кохрена
Gp = 0.12375629621158252
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [367.138, 0.529, 0.614, 1.05, 0.041, 0.611, 0.041, 0.203, 268.104, 268.164, 267.803]

Коефіцієнти [0.096, -0.008, 0.363, -0.001, -0.022, -0.005, 0.001] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [199.947, -0.0, 0.002, -0.023]
[199.92600000000002, 199.92600000000002, 199.92600000000002, 199.92600000000002, 199.92600000000002, 199.92600000000002, 199.92600000000002, 199.92600000000002, 199.92600000000002, 199.

Перевірка адекватності за критерієм Фішера
Fp = 0.5087411278152151
F_t = 2.125558760875511
Математична модель адекватна експериментальним даним

Process finished with exit code 0
```

Висновок:

У ході лабораторної роботи я змоделював ттрьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план та знайшов рівняння регресії, яке адекватне для опису об'єкту.