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 №4

з дисципліни

«Методи оптимізації та планування експерименту»

Тема: «Проведення трьофакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії»

Виконав:

студент групи ІО-93
Комаровський Роман Сергійович
Номер у списку: 14

Перевірив:

ас. Регіда П. Г.

Київ 2021

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

Завдання:

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. Варіанти вибираються по номеру в списку в журналі викладача.

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

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

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Варіант завдання:

314	X_1		X_2		X_3	
	-6	6	-5	5	-10	8

Лістинг програми:

```
from random import randint
from functools import reduce
import numpy as np

# Cramer's rule
def cramer(arr, ins, pos):
    matrix = np.insert(np.delete(arr, pos, 1), pos, ins, 1)

    return np.linalg.det(matrix) / np.linalg.det(arr)

# Method to get dispersion
def getDispersion(y, y_r):
    return [round(sum([(y[i][j] - y_r[i]) ** 2 for j in range(len(y[i]))]) /
3, 3) for i in range(len(y_r))]

def generate_factors_table(raw_array):
    new_list = [row + [row[0] * row[1], row[0] * row[2], row[1] * row[2],
row[0] * row[1] * row[2]]
                + list(map(lambda x: round(x ** 2, 5), row))
                for row in raw_array]
    return np.array(new_list)

def m_ij(*arrays):
```

```
    return np.average(reduce(lambda accum, el: accum * el, arrays))

# Cochran criteria
def cochran(disp, m):
    Gp = max(disp) / sum(disp)
    Gt = [.4709, .3346, .2758, .2419, .2159, .2034, .1911, .1815, .1736]

    return [round(Gp, 4), Gt[m - 2]]

# Student criteria
def student(disp, m, y_r, x_nT):
    table = {
        8: 2.306,
        16: 2.120,
        24: 2.064,
        'inf': 1.960
    }

    N = len(y_r)

    Sb = sum(disp) / len(y_r)
    Sbeta = (Sb / (m * N)) ** (1 / 2)

    beta = [sum([y_r[j] * x_nT[i][j] for j in range(N)]) / N for i in
range(N)]
    t = [abs(beta[i]) / Sbeta for i in range(len(beta))]

    f3 = N * (m - 1)

    if f3 > 30:
        t_t = table['inf']
    elif f3 > 0:
        t_t = table[f3]
    else:
        return

    result = []
    for i in t:
        if i < t_t:
            result.append(False)
        else:
            result.append(True)

    return result

# Fisher criteria
def fisher(y_r, y_st, b_det, disp, m):
    table = {
        8: [5.3, 4.5, 4.1, 3.8, 3.7, 3.6, 3.3],
        16: [4.5, 3.6, 3.2, 3.0, 2.9, 2.7, 2.4],
        24: [4.3, 3.4, 3.0, 2.8, 2.6, 2.5, 2.2],
    }

    N = len(y_r)
    Sb = sum(disp) / N
    d = 0
    for b in b_det:
        if b:
            d += 1

    f4 = N - d
    f3 = N * (m - 1)
    Sad = (m / f4) * sum([(y_st[i] - y_r[i]) ** 2 for i in range(N)])
    Fap = Sad / Sb
```

```
    Ft = table[f3][f4 - 1]

    if Fap < Ft:
        return f"\nРівняння регресії адекватно оригіналу:\nFap < Ft:
{round(Fap, 2)} < {Ft}"
    else:
        return f"\nРівняння регресії неадекватно оригіналу: \nFap > Ft:
{round(Fap, 2)} > {Ft}"

# Main function
def experiment(m, min_x1, max_x1, min_x2, max_x2, min_x3, max_x3):
    y_min = round((min_x1 + min_x2 + min_x3) / 3) + 200
    y_max = round((max_x1 + max_x2 + max_x3) / 3) + 200

    x_norm_prt = [
        [-1, -1, -1],
        [-1, -1, +1],
        [-1, +1, -1],
        [-1, +1, +1],
        [+1, -1, -1],
        [+1, -1, +1],
        [+1, +1, -1],
        [+1, +1, +1],
        [-1.215, 0, 0],
        [+1.215, 0, 0],
        [0, -1.215, 0],
        [0, +1.215, 0],
        [0, 0, -1.215],
        [0, 0, +1.215],
        [0, 0, 0]
    ]

    x0i = []
    xi = []

    for i in [[min_x1, max_x1], [min_x2, max_x2], [min_x3, max_x3]]:
        x0 = round((i[1] + i[0]) / 2, 3)
        x0i.append(x0)
        xi.append([round(1.215 * (i[1] - x0) + x0, 3), round(-1.215 * (i[1] -
x0) + x0, 3)])

    x_prt = [
        [-min_x1, -min_x2, -min_x3],
        [-min_x1, -max_x2, max_x3],
        [-max_x1, min_x2, -max_x3],
        [-max_x1, max_x2, min_x3],
        [min_x1, -min_x2, -max_x3],
        [min_x1, -max_x2, min_x3],
        [max_x1, min_x2, -min_x3],
        [max_x1, max_x2, max_x3],
        [xi[0][0], x0i[1], x0i[2]],
        [xi[0][1], x0i[1], x0i[2]],
        [x0i[0], xi[1][0], x0i[2]],
        [x0i[0], xi[1][1], x0i[2]],
        [x0i[0], x0i[1], xi[2][0]],
        [x0i[0], x0i[1], xi[2][1]],
        [x0i[0], x0i[1], x0i[2]]
    ]

    x_norm = generate_factors_table(x_norm_prt)
    x = generate_factors_table(x_prt)
    # for i in x:
    #     print(i)
```

```
    # print()
    x_normT = x_norm.T
    xT = x.T

    N = len(x)
    y = [[randint(y_min, y_max) for _ in range(m)] for _ in range(N)]
    y_r = [round(sum(y[i]) / len(y[i]), 2) for i in range(N)]

    disp = getDispersion(y, y_r)
    cochran_cr = cochran(disp, m)

    # Get coefficients
    x1 = xT[0]
    x2 = xT[1]
    x3 = xT[2]
    yi = np.array(y_r)

    x_tmp = [N, sum(x1), sum(x2), sum(x3), sum(x1 * x2), sum(x1 * x3), sum(x2
* x3), sum(x1 * x2 * x3),
             sum(x1**2), sum(x2**2), sum(x3**2)]

    x_i = [x_tmp] + [[x_tmp[i]*x_tmp[j] for j in range(len(x_tmp))] for i in
range(1, len(x_tmp))]
    y_free = [round(sum(yi), 3)] + [round(sum(yi) * x_tmp[i], 3) for i in
range(1, len(x_tmp))]
    beta = np.linalg.solve(x_i, y_free)

    x1_norm = x_normT[0]
    x2_norm = x_normT[1]
    x3_norm = x_normT[2]

    b_norm = [sum(yi) / N, sum(yi * x1_norm / N), sum(yi * x2_norm) / N,
sum(yi * x3_norm) / N,
              sum(yi * x1_norm * x2_norm) / N, sum(yi * x1_norm * x3_norm) /
N, sum(yi * x2_norm * x3_norm) / N,
              sum(yi * x1 * x2 * x3_norm) / N]

    b_det = student(disp, m, y_r, x_normT)
    b_cut = b.copy()

    # Simplified equations
    if b_det is None:
        return
    else:
        for i in range(N):
            if not b_det[i]:
                b_cut[i] = 0

        y_st = [round(sum([b_cut[0]] + [x[i][j] * b_cut[j + 1] for j in
range(N - 1)]), 2) for i in range(N)]

    # Calculate F-test
    fisher_cr = fisher(y_r, y_st, b_det, disp, m)

    # Print out results
    print(f"\nМатриця планування для m = {m}:")
    for i in range(m):
        print(f"Y{i + 1} - {np.array(y).T[i]}")

    print(f"\nСередні значення функції відгуку за рядками:\nY_R: {y_r}")
    print(f"\nКоефіцієнти рівняння регресії:")
    for i in range(len(b)):
        print(f"b{i} = {round(b[i], 3)}")
```

```
    if cochran_cr[0] < cochran_cr[1]:
        print(f"\nЗа критерієм Кохрена дисперсія однорідна:\nGp < Gt -
{cochran_cr[0]} < {cochran_cr[1]}")
    else:
        print(f"\nЗа критерієм Кохрена дисперсія неоднорідна:\nGp > Gt -
{cochran_cr[0]} > {cochran_cr[1]}"
              f"\nСпробуйте збільшити кількість експериментів.")
        return

    print(f"\nЗа критерієм Стьюдента коефіцієнти ", end="")
    for i in range(len(b_det)):
        if not b_det[i]:
            print(f"b{i} ", end="")
    print("приймаємо незначними")

    print(f"\nОтримані функції відгуку зі спрощеними коефіцієнтами:\nY_St -
{y_st}")
    print(fisher_cr)

    return True

if __name__ == '__main__':
    Min_x1, Max_x1 = -6, 6
    Min_x2, Max_x2 = -5, 5
    Min_x3, Max_x3 = -10, 8

    M = 3

    experiment(M, Min_x1, Max_x1, Min_x2, Max_x2, Min_x3, Max_x3)
```

Результат:

```
Y:
 [[198. 200. 204. 195.]
 [203. 200. 199. 203.]
 [204. 196. 197. 201.]
 [197. 200. 197. 197.]
 [195. 195. 202. 199.]
 [201. 196. 202. 196.]
 [198. 200. 198. 201.]
 [202. 197. 198. 199.]
 [203. 197. 204. 198.]
 [196. 203. 197. 197.]
 [204. 202. 203. 197.]
 [195. 199. 201. 198.]
 [197. 201. 201. 196.]
 [200. 197. 197. 196.]
 [198. 203. 200. 203.]]

Коефіцієнти рівняння регресії:
[198.957, 0.147, 0.201, -0.095, -0.023, 0.0, 0.041, 0.003, -0.01, -0.004, 0.002]

Результат рівняння зі знайденими коефіцієнтами:
 [199.062 201.062 199.314 197.604 197.533 199.203 199.787 200.057 199.614
 198.33  200.172 198.364 199.35  199.458 199.332]
```

```
    Перевірка рівняння:

Середнє значення y: [199.25, 201.25, 199.5, 197.75, 197.75, 198.75, 199.25, 199.0, 200.5, 198.25, 201.5, 198.25, 198.75, 197.5, 201.0]
Дисперсія y: [10.688, 3.188, 10.25, 1.688, 8.688, 7.688, 1.688, 3.5, 9.25, 7.688, 7.25, 4.688, 5.188, 2.25, 4.5]

Перевірка за критерієм Кохрена
Gp = 0.12119013062409288
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [636.403, 0.369, 1.16, 0.962, 1.065, 0.106, 1.065, 0.532, 464.515, 464.83, 463.729]

Коефіцієнти [0.147, 0.201, -0.095, -0.023, 0.0, 0.041, 0.003] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [198.957, -0.01, -0.004, 0.002]
[198.94500000000002, 198.94500000000002, 198.94500000000002, 198.94500000000002, 198.94500000000002, 198.94500000000002,


Перевірка адекватності за критерієм Фішера

Fp = 1.5429882557515084
F_t = 2.008842199095351

Математична модель адекватна
```

Висновок

У ході лабораторної роботи було досліджено трьохфакторний експеримент з рівнянням регресії з квадратичними членами, використано критерій Кохрена для перевірки дисперсій на однорідність, критерій Стьюдента для перевірки нуль-гіпотези та критерій Фішера перевірки адекватності гіпотези. Можна зробити висновок, що квадратичні члени підвищують точність апроксимації. Розглянута модель дає результати, що практично співпадають з модельованими. При моделюванні використано центральний ортогональний композиційний план, оскільки дробового та повного факторного плану недостатньо для пошуку всіх невідомих коефіцієнтів рівняння регресії.