Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Методи планування експерименту

Лабораторна робота №5

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

Виконав:
студент ІІ курсу ФІОТ
групи ІВ-92
Накарловіч Р. Р.
номер у списку групи – 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.

Варіанти обираються по номеру в списку в журналі викладача.

Варіант завдання:

№варіанта	x_1		x_2		x_3	
	min	max	min	Max	min	max
214	-6	10	-8	3	-10	9

Лістинг програми:

```
import random
from functools import partial

import numpy
import pandas
import sklearn.linear_model as lm
from scipy.stats import f, t
from pyDOE2 import ccdesign
from tabulate import tabulate


class LaboratoryWorkN5:
    def __init__(self, x1, x2, x3):
        self.n, self.m = 15, 6
        self.x, self.y, self.x_normalized = None, None, None
        self.y_average = None
        self.b = None
        self.x_range = (x1, x2, x3)
        self.x_aver_max = numpy.average([x[1] for x in self.x_range])
        self.x_aver_min = numpy.average([x[0] for x in self.x_range])
        self.y_max = 200 + int(self.x_aver_max)
        self.y_min = 200 + int(self.x_aver_min)

    class Criteria:
        def __init__(self, x, y, n, m):
            self.x, self.y = x, y
            self.n, self.m = n, m
            self.f1, self.f2 = self.m - 1, self.n
            self.q = 0.05
            self.q1 = self.q / self.f1

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

        def cochrane_criterion(self, y_average):
            s_kv = self.s_kv(y_average)
            gp = max(s_kv) / sum(s_kv)
            print('Перевірка за критерієм Кохрена:')
            return gp

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

        def bs(self, x, y_average):
            result = [sum(y_average) / self.n]
            for i in range(len(x[0])):
                b = sum(j[0] * j[1] for j in zip(x[:, i], y_average)) /
self.n
                result.append(b)
            return result
```

```
        def student_criterion(self, x, y_average):
            s_kv = self.s_kv(y_average)
            s_kv_aver = sum(s_kv) / self.n
            s_bs = (s_kv_aver / self.n / self.m) ** 0.5
            bs = self.bs(x, y_average)
            return [round(abs(B) / s_bs, 3) for B in bs]

        def fisher_criterion(self, y_average, y_new, d):
            s_ad = self.m / (self.n - d) * sum([(y_new[i] - y_average[i])
** 2 for i in range(len(self.y))])
            s_kv = self.s_kv(y_average)
            s_kv_aver = sum(s_kv) / self.n
            return s_ad / s_kv_aver

    @staticmethod
    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

    def get_y_average(self):
        self.y_average = [round(sum(i) / len(i), 3) for i in self.y]

    @staticmethod
    def regression_equation(x, b):
        return sum([x[i] * b[i] for i in range(len(x))])

    def get_b_coefficient(self):
        skm = lm.LinearRegression(fit_intercept=False)
        skm.fit(self.x, self.y_average)
        self.b = skm.coef_
        print('Коефіцієнти рівняння регресії:')
        self.b = [round(i, 3) for i in self.b]
        print('\ty = {} +{}*x1 +{}*x2 +{}*x3 + {}*x1*x2 + {}*x1*x3 +
{}*x2*x3 + b{}*x1*x2*x3 + {}x1^2 + {}x2^2 + {}x3^2'
              .format(*self.b))
        print(f'Результат рівняння зі знайденими
коефіцієнтами:\n\t{numpy.dot(self.x, self.b)}')

    def check(self):
        criteria = self.Criteria(self.x, self.y, self.n, self.m)

        print('Перевірка рівняння:')
        f1 = self.m - 1
        f2 = self.n
        f3 = f1 * f2
        q = 0.05

        student = partial(t.ppf, q=1 - q)
        t_student = student(df=f3)
        g_kr = criteria.cochrane()
        y_average = [round(sum(i) / len(i), 3) for i in self.y]
        print(f'\tСереднє значення y: {y_average}')
        dispersion = criteria.s_kv(y_average)
```

```
        print(f'\tДисперсія y: {dispersion}')
        gp = criteria.cochrane_criterion(y_average)
        print(f'\tGp = {gp}')
        if gp < g_kr:
            print(f'\tЗ імовірністю {1 - q} дисперсії однорідні.')
        else:
            print('\tНеобхідно збільшити кількість дослідів!')
            self.m += 1
            new_exp = LaboratoryWorkN5(*self.x_range)
            new_exp.run(self.n, self.m)

        ts = criteria.student_criterion(self.x_normalized[:, 1:],
y_average)
        print(f'Перевірка за критерієм Стьюдента:\n\t{ts}')
        result = [element for element in ts if element > t_student]
        final_k = [self.b[i] for i in range(len(ts)) if ts[i] in result]
        print(f'\tКоефіцієнти {[round(i, 3) for i in self.b if i not in
final_k]} '
              f'статистично незначущі, тому ми виключаємо їх з
рівняння.')

        y_new = []
        for j in range(self.n):
            y_new.append(round(
                self.regression_equation([self.x[j][i] for i in
range(len(ts)) if ts[i] in result], final_k), 3))

        print(f'Значення Y з коефіцієнтами {final_k}:')
        print(f'\t{y_new}')

        d = len(result)
        if d >= self.n:
            print('F4 <= 0')

        f4 = self.n - d

        f_p = criteria.fisher_criterion(y_average, y_new, d)

        fisher = partial(f.ppf, q=0.95)
        f_t = fisher(dfn=f4, dfd=f3)
        print('Перевірка адекватності за критерієм Фішера:')
        print(f'\tFp = {f_p}')
        print(f'\tFt = {f_t}')
        if f_p < f_t:
            print('\tМатематична модель адекватна експериментальним
даним.')
        else:
            print('\tМатематична модель не адекватна експериментальним
даним.')

    def fill_y(self):
        self.y = numpy.zeros(shape=(self.n, self.m))
        for i in range(self.n):
            for j in range(self.m):
                self.y[i][j] = random.randint(self.y_min, self.y_max)

    def form_matrix(self):
        self.fill_y()
        if self.n > 14:
            no = self.n - 14
```

```
        else:
            no = 1

        self.x_normalized = ccdesign(3, center=(0, no))
        self.x_normalized = numpy.insert(self.x_normalized, 0, 1, axis=1)

        for i in range(4, 11):
            self.x_normalized = numpy.insert(self.x_normalized, i, 0,
axis=1)

        lc = 1.215

        for i in range(len(self.x_normalized)):
            for j in range(len(self.x_normalized[i])):
                if self.x_normalized[i][j] < -1 or
self.x_normalized[i][j] > 1:
                    if self.x_normalized[i][j] < 0:
                        self.x_normalized[i][j] = -lc
                    else:
                        self.x_normalized[i][j] = lc

        self.x_normalized = self.add_sq_nums(self.x_normalized)

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

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

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

        self.x[8][1] = lc * dx[0] + self.x[9][1]
        self.x[9][1] = -lc * dx[0] + self.x[9][1]
        self.x[10][2] = lc * dx[1] + self.x[9][2]
        self.x[11][2] = -lc * dx[1] + self.x[9][2]
        self.x[12][3] = lc * dx[2] + self.x[9][3]
        self.x[13][3] = -lc * dx[2] + self.x[9][3]
        self.x = self.add_sq_nums(self.x)

        show_arr = pandas.DataFrame(self.x)
        print('X:\n', tabulate(show_arr, headers='keys',
tablefmt='psql'))

        show_arr = pandas.DataFrame(self.x_normalized)
        print('Нормовані X:\n', tabulate(show_arr.round(0),
headers='keys', tablefmt='psql'))

        show_arr = pandas.DataFrame(self.y)
        print('Y:\n', tabulate(show_arr, headers='keys',
tablefmt='psql'))
```

```
    def run(self, n=None, m=None):
        if n is not None and m is not None:
            self.n = n
            self.m = m

        self.form_matrix()
        self.get_y_average()
        self.get_b_coefficient()
        self.check()


if __name__ == '__main__':
    worker = LaboratoryWorkN5((-6, 10), (-8, 3), (-10, 9))
    worker.run(15, 3)
```

Результати виконання роботи:

X:

	0	1	2	3	4	5	6	7	8	9	10
0	1	-6	-8	-10	48	60	80	-480	36	64	100
1	1	10	-8	-10	-80	-100	80	800	100	64	100
2	1	-6	3	-10	-18	60	-30	180	36	9	100
3	1	10	3	-10	30	-100	-30	-300	100	9	100
4	1	-6	-8	9	48	-54	-72	432	36	64	81
5	1	10	-8	9	-80	90	-72	-720	100	64	81
6	1	-6	3	9	-18	-54	27	-162	36	9	81
7	1	10	3	9	30	90	27	270	100	9	81
8	1	11	-2	1	-22	11	-2	-22	121	4	1
9	1	-7	-2	1	14	-7	-2	14	49	4	1
10	1	2	4	1	8	2	4	8	4	16	1
11	1	2	-8	1	-16	2	-8	-16	4	64	1
12	1	2	-2	12	-4	24	-24	-48	4	4	144
13	1	2	-2	-10	-4	-20	20	40	4	4	100
14	1	2	-2	1	-4	2	-2	-4	4	4	1

Нормовані X:

	0	1	2	3	4	5	6	7	8	9	10
0	1	-1	-1	-1	1	1	1	-1	1	1	1
1	1	1	-1	-1	-1	-1	1	1	1	1	1
2	1	-1	1	-1	-1	1	-1	1	1	1	1
3	1	1	1	-1	1	-1	-1	-1	1	1	1
4	1	-1	-1	1	1	-1	-1	1	1	1	1
5	1	1	-1	1	-1	1	-1	-1	1	1	1
6	1	-1	1	1	-1	-1	1	-1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1
8	1	-1	0	0	-0	-0	0	-0	1	0	0
9	1	1	0	0	0	0	0	0	1	0	0
10	1	0	-1	0	-0	0	-0	-0	0	1	0
11	1	0	1	0	0	0	0	0	0	1	0
12	1	0	0	-1	0	-0	-0	-0	0	0	1
13	1	0	0	1	0	0	0	0	0	0	1
14	1	0	0	0	0	0	0	0	0	0	0

Y:

	0	1	2
0	207	204	197
1	196	196	206
2	194	203	192
3	202	205	206
4	199	195	194
5	206	195	195
6	198	200	197
7	201	194	197
8	197	201	199
9	194	202	204
10	192	193	201
11	195	197	192
12	193	207	196
13	197	206	206
14	193	202	194

```
Коефіцієнти рівняння регресії:
    y = 197.048 +0.014*x1 +-0.366*x2 +-0.117*x3 + 0.019*x1*x2 + -0.017*x1*x3 + 0.015*x2*x3 + b-0.004*x1*x2*x3 + 0.023x1^2 + -0.071x2^2 + 0.025x3^2
Результат рівняння зі знайденими коефіцієнтами:
    [202.858 199.722 197.193 204.441 196.17  197.594 198.656 197.088 199.794
 198.732 194.622 195.066 199.16  201.69  197.4  ]
Перевірка рівняння:
    Середнє значення y: [202.667, 199.333, 196.333, 204.333, 196.0, 198.667, 198.333, 197.333, 199.0, 200.0, 195.333, 194.667, 198.667, 203.0, 196.333]
    Дисперсія y: [17.556, 22.222, 22.889, 2.889, 4.667, 26.889, 1.556, 8.222, 2.667, 18.667, 16.222, 4.222, 36.222, 18.0, 16.222]
Перевірка за критерієм Кохрена:
    Gp = 0.1653127167841104
    З імовірністю 0.95 дисперсії однорідні.
Перевірка за критерієм Стьюдента:
    [348.693, 0.883, 0.134, 0.827, 0.897, 0.351, 0.273, 1.755, 255.32, 253.765, 255.781]
    Коефіцієнти [0.014, -0.366, -0.117, 0.019, -0.017, 0.015] статистично незначущі, тому ми виключаємо їх з рівняння.
Значення Y з коефіцієнтами [197.048, -0.004, 0.023, -0.071, 0.025]:
    [197.752, 194.104, 199.017, 202.409, 193.629, 199.709, 199.91, 199.654, 199.66, 197.86, 195.997, 192.685, 200.648, 199.196, 196.897]
Перевірка адекватності за критерієм Фішера:
    Fp = 2.1581558609295644
    Ft = 2.164579917125473
    Математична модель адекватна експериментальним даним.

Process finished with exit code 0
```

Висновок:

У ході виконання лабораторної роботи проведено повний трьохфакторний експеримент. В ході дослідження було розроблено відповідну програму мовою програмування Python, яка моделює проведення трьохфакторного експерименту, використовуючи центральний ортогональний композиційний план. Знайдено рівняння регресії, адекватне для опису об’єкту, проведено 3 статистичні перевірки (критерії Кохрена, Стьюдента та Фішера). Результати роботи, наведені у протоколі, підтверджують правильність виконання – кінцеву мету роботи було досягнуто.