

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

Лабораторна робота №5

З дисципліни

«Методи оптимізації та планування експерименту»

На тему:

**«Проведення трьохфакторного експерименту при використанні
рівняння регресії з урахуванням квадратичних членів»**

ВИКОНАВ:
Студент II курсу ФІОТ
Групи ІВ-93
Красулін Є.С. – 9316
Номер в списку: 12

ПЕРЕВІРИВ:
ас. Регіда П.Г.

Київ 2021 р.

Мета:

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

Варіант завдання:

Варіант	X ₁		X ₂		X ₃	
	min	max	min	max	min	max
312	0	2	-5	9	-6	10

$$X_{\text{cp min}} = (0 - 5 - 6) / 3 = -3.67$$

$$X_{\text{cp max}} = (2 + 9 + 10) / 3 = 7$$

$$Y_{\text{imin}} = 200 - 3.67 = 196.33$$

$$Y_{\text{imax}} = 200 + 7 = 207$$

Роздруківка коду програми:

```
import random
import sklearn.linear_model as lm
from scipy.stats import f, t
from functools import partial
from pyDOE2 import *

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

x_range = ((0, 2), (-5, 9), (-6, 10))

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)

# квадратна дисперсія
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(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] = -1
            else:
                x_norm[i][j] = 1

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] = 1 * dx[0] + x[9][1]
x[9][1] = -1 * dx[0] + x[9][1]
x[10][2] = 1 * dx[1] + x[9][2]
x[11][2] = -1 * dx[1] + x[9][2]
x[12][3] = 1 * dx[2] + x[9][3]
x[13][3] = -1 * 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_studentsa(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_students(X[:, 1:], Y, y_aver, n, m)
    print('\nКритерій Стюдента:\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(df=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')

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)
```

Скріншоти результату виконання роботи::

Лабораторна 5

Генеруємо матрицю планування для $n = 15$, $m = 3$

X:

```
[[ 1 0 -5 -6 0 0 30 0 0 25 36]
 [ 1 2 -5 -6 -10 -12 30 60 4 25 36]
 [ 1 0 9 -6 0 0 -54 0 0 81 36]
 [ 1 2 9 -6 18 -12 -54 -108 4 81 36]
 [ 1 0 -5 10 0 0 -50 0 0 25 100]
 [ 1 2 -5 10 -10 20 -50 -100 4 25 100]
 [ 1 0 9 10 0 0 90 0 0 81 100]
 [ 1 2 9 10 18 20 90 180 4 81 100]
 [ 1 2 2 1 4 2 2 4 4 4 1]
 [ 1 0 2 1 0 0 2 0 0 4 1]
 [ 1 1 10 1 10 1 10 10 1 100 1]
 [ 1 1 -6 1 -6 1 -6 -6 1 36 1]
 [ 1 1 2 10 2 10 20 20 1 4 100]
 [ 1 1 2 -8 2 -8 -16 -16 1 4 64]
 [ 1 1 2 1 2 1 2 2 1 4 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:

```
[[205. 197. 201.]
 [198. 206. 203.]
 [201. 197. 204.]
 [200. 202. 200.]
 [207. 199. 207.]
 [200. 203. 200.]
 [197. 198. 202.]
 [204. 207. 200.]
 [203. 207. 201.]
 [197. 203. 199.]
 [204. 198. 201.]
 [197. 201. 197.]
 [203. 197. 202.]
 [206. 197. 201.]
 [202. 203. 200.]]
```

Коефіцієнти рівняння регресії:

```
[201.308, -1.626, -0.034, 0.062, 0.077, -0.047, -0.023, 0.021, 1.073, -0.015, 0.006]
```

Результат рівняння зі знайденими коефіцієнтами:

```
[200.257 202.351 200.873 201.595 203.473 200.703 198.937 204.203 202.54
 201.202 199.686 199.99 201.491 201.077 200.798]
```

Перевірка рівняння:

Середнє значення у: [201.0, 202.333, 200.667, 200.667, 204.333, 201.0, 199.0, 203.667, 203.667, 199.667, 201.0, 198.333, 200.667, 201.333, 201.667]
Дисперсія у: [10.667, 10.889, 8.222, 0.889, 14.222, 2.0, 4.667, 8.222, 6.222, 6.222, 6.0, 3.556, 6.889, 13.556, 1.556]

Перевірка за критерієм Кохрена

$G_r = 0.13704121257672555$

З ймовірністю 0.95 дисперсії однорідні.

Критерій Стюдента:

```
[513.297, 0.373, 1.344, 0.704, 1.134, 0.0, 0.113, 1.587, 375.423, 374.418, 375.888]
```

Коефіцієнти [-1.626, -0.034, 0.062, 0.077, -0.047, -0.023, 0.021] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "у" з коефіцієнтами [201.308, 1.073, -0.015, 0.006]

```
[202.372, 202.372, 202.372, 202.372, 202.372, 202.372, 202.372, 202.372, 202.372, 202.372, 202.372, 202.372, 202.372, 202.372, 202.372]
```

Перевірка адекватності за критерієм Фішера

$F_p = 1.845893662186694$

$F_t = 2.125558760875511$

Математична модель адекватна експериментальним даним

Висновок:

В даній лабораторній роботі я провів трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайшов рівняння регресії, яке буде адекватним для опису об'єкту.