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

Лабораторна робота №6
з дисципліни «Методи оптимізації та планування експерименту»

«Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

Виконав:
студент групи ІВ-92
Коптюх Н.Є
Залікова книжка № ІВ-9214
Варіант: 212
Перевірив:
Регіда П.Г.

Київ 2021

Мета: Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

Завдання:

1. Ознайомитися з теоретичними відомостями.

2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x1, x2, x3. Обчислити і записати значення, відповідні кодованим значенням

факторів

3. Значення функції відгуку знайти за допомогою підстановки в формулу:

yi = f(x1, x2, x3) + random(10)-5,

де f(x1, x2, x3) вибирається по номеру в списку в журналі викладача.

4. Провести експерименти і аналізуючи значення статистичних перевірок, отримати адекватну модель рівняння регресії. При розрахунках

використовувати натуральні значення факторів.

5. Зробити висновки по виконаній роботі.

Варіант №212:

212	-40	20	5	40	-40	-20	5,4+2,4*x1+7,3*x2+9,6*x3+2,5*x1*x1+0,2*x2*x2+8,2*x3*x3+1,7*x1*x2+0,7*x1*x3+0,6*x2*x3+9,3*x1*x2*x3

Роздруківка коду програми:

```
import random
from scipy.stats import t, f
import numpy as np
from itertools import product, combinations

np.set_printoptions(formatter={'float_kind': lambda x: "%.2f" % (x)})

gt = {12: {1: 0.5410, 2: 0.3924, 3: 0.3264, 4: 0.2880, 5: 0.2624, 6: 0.2439, 7: 0.2299,
8: 0.2187, 9: 0.2098, 10: 0.2020},
      15: {1: 0.4709, 2: 0.3346, 3: 0.2758, 4: 0.2419, 5: 0.2159, 6: 0.2034, 7: 0.1911,
8: 0.1815, 9: 0.1736, 10: 0.1671}}
tt = {24: 2.064, 30: 2.042, 32: 1.96}  # m = [3, 6]
ft = {1: 4.2, 2: 3.3, 3: 2.9, 4: 2.7, 5: 2.5, 6: 2.4}
matrix_with_min_max_x = np.array([[-40, 20], [5, 40], [-40, -20]])
m = 3

def table_student(prob, n, m):
    x_vec = [i*0.0001 for i in range(int(5/0.0001))]
    par = 0.5 + prob/0.1*0.05
    f3 = (m - 1) * n
    for i in x_vec:
        if abs(t.cdf(i, f3) - par) < 0.000005:
            return i


def table_fisher(prob, n, m, d):
    x_vec = [i*0.001 for i in range(int(10/0.001))]
    f3 = (m - 1) * n
    for i in x_vec:
        if abs(f.cdf(i, n-d, f3)-prob) < 0.0001:
```

```
            return i


def cochran_check(Y_matrix_, N):
    mean_Y = np.mean(Y_matrix, axis=1)
    dispersion_Y = np.mean((Y_matrix.T - mean_Y) ** 2, axis=0)
    Gp = np.max(dispersion_Y) / (np.sum(dispersion_Y))
    fisher = table_fisher(0.95, N, m, 1)
    Gt = fisher / (fisher + (m - 1) - 2)
    return Gp < Gt


def students_t_test(norm_matrix_, Y_matrix_, N):
    mean_Y_ = np.mean(Y_matrix_, axis=1)
    dispersion_Y = np.mean((Y_matrix_.T - mean_Y_) ** 2, axis=0)
    mean_dispersion = np.mean(dispersion_Y)
    sigma = np.sqrt(mean_dispersion / (N * m))
    betta = np.mean(norm_matrix_.T * mean_Y_, axis=1)
    t = np.abs(betta) / sigma
    if (m - 1) * N > 32:
        return np.where(t > table_student(0.95, N, m))
    return np.where(t > table_student(0.95, N, m))


def phisher_criterion(Y_matrix, d, N):
    if d == N:
        return False
    Sad = (m / (N - d)) * np.sum((check2 - mean_Y)**2)
    mean_dispersion = np.mean(np.mean((Y_matrix.T - mean_Y) ** 2, axis=0))
    Fp = Sad / mean_dispersion
    if (m-1)*N > 32:
        if N-d > 6:
            return table_fisher(0.95, N, m, d)
        return Fp < table_fisher(0.95, N, m, d)
    if N - d > 6:
        return Fp < table_fisher(0.95, N, m, d)
    return Fp < table_fisher(0.95, N, m, d)


def make_plan_matrix_from_norm_matrix(norm_matrix):
    plan_matrix = np.empty((len(norm_matrix), len(norm_matrix[0])), dtype=np.float)
    for i in range(len(norm_matrix)):
        for j in range(len(norm_matrix[i])):
            if norm_matrix[i, j] == -1:
                plan_matrix[i, j] = matrix_with_min_max_x[j-1][0]
            elif norm_matrix[i, j] == 1 and j != 0:
                plan_matrix[i, j] = matrix_with_min_max_x[j-1][1]
            elif norm_matrix[i, j] == 1 and j == 0:
                plan_matrix[i, j] = 1
            else:
                mean = np.mean(matrix_with_min_max_x[j-1])
                plan_matrix[i, j] = norm_matrix[i, j] * (matrix_with_min_max_x[j-1][1]
- mean) + mean
    return plan_matrix


def make_linear_equation():
    norm_matrix = np.array(list(product("01", repeat=3)), dtype=np.int)
    norm_matrix[norm_matrix == 0] = -1
    norm_matrix = np.insert(norm_matrix, 0, 1, axis=1)
    plan_matrix = make_plan_matrix_from_norm_matrix(norm_matrix)
    return norm_matrix, plan_matrix


def make_equation_with_interaction_effect(current_norm_matrix, current_plan_matrix):
    plan_matr = current_plan_matrix
```

```
    norm_matrix = current_norm_matrix
    combination = list(combinations(range(1, 4), 2))
    for i in combination:
        plan_matr = np.append(plan_matr, np.reshape(plan_matr[:, i[0]] * plan_matr[:,
i[1]], (len(norm_matrix), 1)),axis=1)
        norm_matrix = np.append(norm_matrix, np.reshape(norm_matrix[:, i[0]] *
norm_matrix[:, i[1]], (len(norm_matrix), 1)), axis=1)
    plan_matr = np.append(plan_matr, np.reshape(plan_matr[:, 1] * plan_matr[:, 2] *
plan_matr[:, 3], (len(norm_matrix), 1)), axis=1)
    norm_matrix = np.append(norm_matrix, np.reshape(norm_matrix[:, 1] * norm_matrix[:,
2] * norm_matrix[:, 3], (len(norm_matrix), 1)), axis=1)
    return norm_matrix, plan_matr


def make_equation_with_quadratic_terms(current_norm_matrix):
    norm_matrix_second_part = np.empty((3, 7))
    key = 0
    for i in range(3):
        j = 0
        while j < 7:
            if j == key:
                norm_matrix_second_part[i][key] = -1.73
                norm_matrix_second_part[i][key + 1] = 1.73
                j += 1
            else:
                norm_matrix_second_part[i][j] = 0
            j += 1
        key += 2

    norm_matrix_second_part = np.insert(norm_matrix_second_part, 0, 1, axis=0)
    norm_matrix = np.append(current_norm_matrix, norm_matrix_second_part.T, axis=0)
    plan_matrix = make_plan_matrix_from_norm_matrix(norm_matrix)
    plan_matrix = make_equation_with_interaction_effect(norm_matrix, plan_matrix)[1]
    plan_matrix = np.append(plan_matrix, plan_matrix[:, 1:4] ** 2, axis=1)
    norm_matrix = make_equation_with_interaction_effect(norm_matrix, plan_matrix)[0]
    norm_matrix = np.append(norm_matrix, norm_matrix[:, 1:4] ** 2, axis=1)
    return norm_matrix, plan_matrix


count = 0
flag_of_model = False
while flag_of_model is False:
    norm_matrix = make_linear_equation()[0]
    plan_matr = make_linear_equation()[1]
    if count == 1:
        norm_matrix = make_equation_with_interaction_effect(norm_matrix, plan_matr)[0]
        plan_matr = make_equation_with_interaction_effect(norm_matrix, plan_matr)[1]
    elif count > 1:
        plan_matr = make_equation_with_quadratic_terms(norm_matrix)[1]
        norm_matrix = make_equation_with_quadratic_terms(norm_matrix)[0]
    plan_matr_for_calc_Y = plan_matr
    N = len(plan_matr)
    Y_matrix = []
    mean_Y = []
    indexes = []
    flag_of_dispersion = False
    while flag_of_dispersion is False:
        Y_matrix = np.array(
            [5.4 + 2.4 * plan_matr_for_calc_Y[:, 1] + 7.3 * plan_matr_for_calc_Y[:, 2]
+ 9.6 * plan_matr_for_calc_Y[:, 3] + 2.5 * plan_matr_for_calc_Y[:, 1] ** 2 +
             0.2 * plan_matr_for_calc_Y[:, 2] ** 2 + 8.2 * plan_matr_for_calc_Y[:, 3]
** 2 + 1.7 * plan_matr_for_calc_Y[:, 1] * plan_matr_for_calc_Y[:, 2] +
             0.7 * plan_matr_for_calc_Y[:, 1] * plan_matr_for_calc_Y[:, 3] + 0.6 *
plan_matr_for_calc_Y[:, 2] * plan_matr_for_calc_Y[:, 3] +
             9.3 * plan_matr_for_calc_Y[:, 1] * plan_matr_for_calc_Y[:, 2] *
plan_matr_for_calc_Y[:, 3] + random.randint(0, 100) - 50 for i in range(m)]).T
        mean_Y = np.mean(Y_matrix, axis=1)
```

```
        if cochran_check(Y_matrix, N):
            flag_of_dispersion = True
            b_natura = np.linalg.lstsq(plan_matr, mean_Y, rcond=None)[0]
            b_norm = np.linalg.lstsq(norm_matrix, mean_Y, rcond=None)[0]
            check1 = np.sum(b_natura * plan_matr, axis=1)
            indexes = students_t_test(norm_matrix, Y_matrix, N)
            check2 = np.sum(b_natura[indexes] * np.reshape(plan_matr[:, indexes], (N,
np.size(indexes))), axis=1)
            print("Матриця плану експерименту: \n", plan_matr)
            print("Нормована матриця: \n", norm_matrix)
            print("Матриця відгуків: \n", Y_matrix)
            print("Середні значення У: ", mean_Y)
            print("Натуралізовані коефіцієнти: ", b_natura)
            print("Перевірка 1: ", check1)
            print("Індекси коефіцієнтів, які задовольняють критерію Стьюдента: ",
np.array(indexes)[0])
            print("Критерій Стьюдента: ",check2)
        else:
            m += 1
            print("Дисперсія неоднорідна!")
    if phisher_criterion(Y_matrix, np.size(indexes), N):
        flag_of_model = True
        print("Рівняння регресії адекватно оригіналу.")
    else:
        count += 1
        print("Рівняння регресії неадекватно оригіналу.")
```

Результати виконання програми:

```
C:\Users\Marty\AppData\Local\Programs\Python\Python37\python.exe
C:/Users/Marty/PycharmProjects/DM-Lab2/mope-6.py
Матриця плану експерименту:
 [[1.00 -40.00 5.00 -40.00]
 [1.00 -40.00 5.00 -20.00]
 [1.00 -40.00 40.00 -40.00]
 [1.00 -40.00 40.00 -20.00]
 [1.00 20.00 5.00 -40.00]
 [1.00 20.00 5.00 -20.00]
 [1.00 20.00 40.00 -40.00]
 [1.00 20.00 40.00 -20.00]]
Нормована матриця:
 [[ 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]]
Матриця відгуків:
```

[[91709.90 91775.90 91715.90]

[44361.90 44427.90 44367.90]

[609860.40 609926.40 609866.40]

[302532.40 302598.40 302538.40]

[-23916.10 -23850.10 -23910.10]

[-14624.10 -14558.10 -14618.10]

[-283395.60 -283329.60 -283389.60]

[-143483.60 -143417.60 -143477.60]]

Середні значення У: [91733.90 44385.90 609884.40 302556.40 -23892.10 -14600.10 -283371.60

-143459.60]

Натуралізовані коефіцієнти: [-129580.10 -6307.85 2771.30 -2568.40]

Перевірка 1: [239326.40 187958.40 336321.90 284953.90 -139144.60 -190512.60 -42149.10

-93517.10]

Індекси коефіцієнтів, які задовольняють критерію Стьюдента: [0 1 2 3]

Критерій Стьюдента: [239326.40 187958.40 336321.90 284953.90 -139144.60 -190512.60 -42149.10

-93517.10]

Рівняння регресії неадекватно оригіналу.

Матриця плану експерименту:

[[1.00 -40.00 5.00 -40.00 -200.00 1600.00 -200.00 8000.00]

[1.00 -40.00 5.00 -20.00 -200.00 800.00 -100.00 4000.00]

[1.00 -40.00 40.00 -40.00 -1600.00 1600.00 -1600.00 64000.00]

[1.00 -40.00 40.00 -20.00 -1600.00 800.00 -800.00 32000.00]

[1.00 20.00 5.00 -40.00 100.00 -800.00 -200.00 -4000.00]

[1.00 20.00 5.00 -20.00 100.00 -400.00 -100.00 -2000.00]

[1.00 20.00 40.00 -40.00 800.00 -800.00 -1600.00 -32000.00]

[1.00 20.00 40.00 -20.00 800.00 -400.00 -800.00 -16000.00]]

Нормована матриця:

[[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]

[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]]

Матриця відгуків:

[[91709.90 91744.90 91698.90]

[44361.90 44396.90 44350.90]

[609860.40 609895.40 609849.40]

[302532.40 302567.40 302521.40]

[-23916.10 -23881.10 -23927.10]

[-14624.10 -14589.10 -14635.10]

[-283395.60 -283360.60 -283406.60]

[-143483.60 -143448.60 -143494.60]]

Середні значення У: [91717.90 44369.90 609868.40 302540.40 -23908.10 -14616.10 -283387.60

-143475.60]

Натуралізовані коефіцієнти: [-4623.60 -47.60 16.30 -482.40 1.70 0.70 0.60 9.30]

Перевірка 1: [91717.90 44369.90 609868.40 302540.40 -23908.10 -14616.10 -283387.60

-143475.60]

Індекси коефіцієнтів, які задовольняють критерію Стьюдента: [0 1 2 3 4 5 6 7]

Критерій Стьюдента: [91717.90 44369.90 609868.40 302540.40 -23908.10 -14616.10 -283387.60

-143475.60]

Рівняння регресії неадекватно оригіналу.

Матриця плану експерименту:

[[1.00 -40.00 5.00 -40.00 -200.00 1600.00 -200.00 8000.00 1600.00 25.00

1600.00]

[1.00 -40.00 5.00 -20.00 -200.00 800.00 -100.00 4000.00 1600.00 25.00

400.00]

[1.00 -40.00 40.00 -40.00 -1600.00 1600.00 -1600.00 64000.00 1600.00

1600.00 1600.00]

[1.00 -40.00 40.00 -20.00 -1600.00 800.00 -800.00 32000.00 1600.00

1600.00 400.00]

[1.00 20.00 5.00 -40.00 100.00 -800.00 -200.00 -4000.00 400.00 25.00

1600.00]

[1.00 20.00 5.00 -20.00 100.00 -400.00 -100.00 -2000.00 400.00 25.00

400.00]

[1.00 20.00 40.00 -40.00 800.00 -800.00 -1600.00 -32000.00 400.00

1600.00 1600.00]

[1.00 20.00 40.00 -20.00 800.00 -400.00 -800.00 -16000.00 400.00 1600.00

400.00]

[1.00 -61.90 22.50 -30.00 -1392.75 1857.00 -675.00 41782.50 3831.61

506.25 900.00]

[1.00 41.90 22.50 -30.00 942.75 -1257.00 -675.00 -28282.50 1755.61

506.25 900.00]

[1.00 -10.00 -7.77 -30.00 77.75 300.00 233.25 -2332.50 100.00 60.45

900.00]

[1.00 -10.00 52.77 -30.00 -527.75 300.00 -1583.25 15832.50 100.00

2785.20 900.00]

[1.00 -10.00 22.50 -47.30 -225.00 473.00 -1064.25 10642.50 100.00 506.25

2237.29]

```
 [1.00 -10.00 22.50 -12.70 -225.00 127.00 -285.75 2857.50 100.00 506.25
  161.29]
 [1.00 -10.00 22.50 -30.00 -225.00 300.00 -675.00 6750.00 100.00 506.25
  900.00]]
Нормована матриця:
 [[1.00 -1.00 -1.00 -1.00 1.00 1.00 1.00 -1.00 1.00 1.00 1.00]
 [1.00 -1.00 -1.00 1.00 1.00 -1.00 -1.00 1.00 1.00 1.00 1.00]
 [1.00 -1.00 1.00 -1.00 -1.00 1.00 -1.00 1.00 1.00 1.00 1.00]
 [1.00 -1.00 1.00 1.00 -1.00 -1.00 1.00 -1.00 1.00 1.00 1.00]
 [1.00 1.00 -1.00 -1.00 -1.00 -1.00 1.00 1.00 1.00 1.00 1.00]
 [1.00 1.00 -1.00 1.00 -1.00 1.00 -1.00 -1.00 1.00 1.00 1.00]
 [1.00 1.00 1.00 -1.00 1.00 -1.00 -1.00 -1.00 1.00 1.00 1.00]
 [1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00]
 [1.00 -1.73 0.00 0.00 -0.00 -0.00 0.00 -0.00 2.99 0.00 0.00]
 [1.00 1.73 0.00 0.00 0.00 0.00 0.00 0.00 2.99 0.00 0.00]
 [1.00 0.00 -1.73 0.00 -0.00 0.00 -0.00 -0.00 0.00 2.99 0.00]
 [1.00 0.00 1.73 0.00 0.00 0.00 0.00 0.00 0.00 2.99 0.00]
 [1.00 0.00 0.00 -1.73 0.00 -0.00 -0.00 -0.00 0.00 0.00 2.99]
 [1.00 0.00 0.00 1.73 0.00 0.00 0.00 0.00 0.00 0.00 2.99]
 [1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]]
Матриця відгуків:
 [[91749.90 91772.90 91736.90]
 [44401.90 44424.90 44388.90]
 [609900.40 609923.40 609887.40]
 [302572.40 302595.40 302559.40]
 [-23876.10 -23853.10 -23889.10]
 [-14584.10 -14561.10 -14597.10]
 [-283355.60 -283332.60 -283368.60]
 [-143443.60 -143420.60 -143456.60]
 [403900.84 403923.84 403887.84]
 [-250853.99 -250830.99 -250866.99]
 [-13928.39 -13905.39 -13941.39]
 [153873.82 153896.82 153860.82]
 [116676.90 116699.90 116663.90]
 [27810.26 27833.26 27797.26]
 [69789.40 69812.40 69776.40]]
Середні значення У:  [91753.23 44405.23 609903.73 302575.73 -23872.77 -14580.77
-283352.27
 -143440.27 403904.17 -250850.66 -13925.06 153877.16 116680.23 27813.59
 69792.73]
Натуралізовані коефіцієнти:  [11.73 2.40 7.30 9.60 1.70 0.70 0.60 9.30 2.50 0.20
8.20]
```

```
Перевірка 1:   [91753.23  44405.23  609903.73  302575.73  -23872.77  -14580.77  -
283352.27
 -143440.27 403904.17 -250850.66 -13925.06 153877.16 116680.23 27813.59
 69792.73]
Індекси коефіцієнтів, які задовольняють критерію Стьюдента:  [ 0  1  2  3  4  5
6  7  8  9 10]
Критерій Стьюдента:  [91753.23 44405.23 609903.73 302575.73 -23872.77 -14580.77
-283352.27
 -143440.27 403904.17 -250850.66 -13925.06 153877.16 116680.23 27813.59
 69792.73]
Рівняння регресії адекватно оригіналу.


Process finished with exit code 0
```

Висновок: Під час виконання роботи проблем не виникало. Отримані результати збігаються з очікуваними. Необхідно рівняння з квадратичними членами, щоб модель була адекватна. Дисперсія завжди однорідна.