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 №6

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

Виконав:

студент ІІ курсу ФІОТ

групи ІВ-91

Черних Богдан

Перевірив:

Регіда П.Г.

Київ – 2021

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

Завдання до лабораторної роботи:

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протоколі інтервали значень x1, x2, x3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1; +l; -l; 0; для $\overline{x_1}$, $\overline{x_2}$, $\overline{x_3}$.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

   $y_i = f(x_1, x_2, x_3) + random(10)-5,$

   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 завдання:

Варіант	X_1		X_2		X_3	
	min	max	min	max	min	max
127	-40	20	-25	10	-25	-10

$f(x_1, x_2, x_3)$
3,4+1,8*x1+3,0*x2+7,7*x3+1,6*x1*x1+0,6*x2*x2+0,7*x3*x3+3,4*x1*x2+0,4*x1*x3+8,8*x2*x3+1,8*x1*x2*x3

Роздруківка тексту програми

```
from random import randint
from numpy.linalg import det
from copy import deepcopy
from scipy.stats import t
import pandas as pd


def naturalize(matrix_of_plan, min_max_arr, flag):
```

```
    result = []
    for i in range(len(matrix_of_plan)):
        if i < 8:
            result.append(min_max_arr[1]) if matrix_of_plan[i] == 1 else
result.append(min_max_arr[0])
        else:
            x0 = (max(min_max_arr) + min(min_max_arr)) / 2
            dx = x0 - min(min_max_arr)
            value = None
            if flag == 1:
                value = matrix_of_plan[i] * dx + x0 if i == 8 or 9 else x0
            elif flag == 2:
                value = matrix_of_plan[i] * dx + x0 if i == 10 or 11 else x0
            elif flag == 3:
                value = matrix_of_plan[i] * dx + x0 if i == 12 or 13 else x0
            result.append(value)
    return result


def y_func(x, i):
    return 3.4 + 1.8 * x[0][i] + 3 * x[1][i] + 7.7 * x[2][i] + 3.4 * x[3][i]
+ 0.4 * x[4][i] + 8.8 * x[5][i] +\
           1.8 * x[6][i] + 1.6 * x[7][i] + 0.6 * x[8][i] + 0.7 * x[9][i]


def cochrans_test(y_arr, y_avg, m):
    # Перевірка однорідності дисперсії за критерієм Кохрена
    print('Перевірка однорідності дисперсії за критерієм Кохрена')
    dispersion = []
    for i in range(len(y_arr[0])):
        current_sum = 0
        for j in range(len(y_arr)):
            current_sum += (y_arr[j][i] - y_avg[i]) ** 2
        dispersion.append(current_sum / len(y_arr))

    print('dispersion:', dispersion)

    gp = max(dispersion) / sum(dispersion)
    print('Gp =', gp)

    # Рівень значимості q = 0.05
    # f1 = m - 1
    # f2 = N
    print(f'm = {m}')

    gt = 0.3346
    print(f'Gt = {gt}')
    # За таблицею Gt = 0.3346
    if gp < gt:
        print('Дисперсія однорідна')
        return dispersion

    print('Дисперсія неоднорідна')
    return None


def students_test(x0_plan1, x1_plan1, x2_plan1, x3_plan1, y_avg_arr,
dispersion, m):
    # Оцінка значимості коефіцієнтів регресії згідно критерію Стьюдента
    print('Оцінка значимості коефіцієнтів регресії згідно критерію
Стьюдента')
    s2b = sum(dispersion) / N
    s2bs_avg = s2b / N * m
    sb = s2bs_avg ** 0.5
```

```
    beta_arr = [
        sum([y_avg_arr[i] * x0_plan1[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * x1_plan1[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * x2_plan1[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * x3_plan1[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * x1_plan1[i] * x2_plan1[i] for i in range(N)]) /
N,
        sum([y_avg_arr[i] * x1_plan1[i] * x3_plan1[i] for i in range(N)]) /
N,
        sum([y_avg_arr[i] * x2_plan1[i] * x3_plan1[i] for i in range(N)]) /
N,
        sum([y_avg_arr[i] * x1_plan1[i] * x2_plan1[i] * x3_plan1[i] for i in
range(N)]) / N,
        sum([y_avg_arr[i] * x1_plan1[i] ** 2 for i in range(N)]) / N,
        sum([y_avg_arr[i] * x2_plan1[i] ** 2 for i in range(N)]) / N,
        sum([y_avg_arr[i] * x3_plan1[i] ** 2 for i in range(N)]) / N
    ]

    print('beta:', beta_arr)
    t_arr = [abs(i) / sb for i in beta_arr]
    print('t:', t_arr)

    # f3 = f1*f2 = 2*14 = 28
    f1 = m - 1
    f2 = N
    f3 = f1 * f2

    b_arr = []
    t_table = t.ppf(q=0.975, df=f3)
    print(f't table = {t_table}')
    count = 0
    for i in range(len(t_arr)):
        if t_arr[i] > t_table:
            b_arr.append(t_arr[i])
        else:
            print(f'Коефіцієнт b{i} приймаємо не значним')
            b_arr.append(0)
            count += 1

    if not count:
        print('Усі коефіцієнти рівняння значимі')

    return b_arr, s2b


def fishers_test(b_arr, s2b, y_avg, y_res, m):
    # Критерій Фішера
    print('Перевірка адекватності моделі за критерієм Фішера')

    d = len([i for i in b_arr if i != 0])  # кількість значимих коефіцієнтів
    print(f'd = {d}')
    s2_ad = m * sum([(y_res[i] - y_avg[i]) ** 2 for i in range(N)]) / N - d
    fp = s2_ad / s2b
    print(f'Fp = {fp}')

    print(f'Ft = {3}')
    # Fт = 3
    if fp > 3:
        print('Рівняння регресії неадекватно оригіналу при рівні значимості
0.05')
    else:
        print('Рівняння регресії адекватно оригіналу при рівні значимості
0.05')
```

```python
def main(m=3):
    # Кількість факторів
    k = 3

    x1 = [-40, 20]
    x2 = [-25, 10]
    x3 = [-25, -10]

    # x1 = [10, 40]
    # x2 = [-15, 35]
    # x3 = [-15, 5]

    # Величина зоряного плеча
    l = round(k ** 0.5, 2)

    # Матриця планування з нормованих значень
    x0_plan1 = [1 for _ in range(N)]
    x1_plan1 = [-1, -1, 1, 1, -1, -1, 1, 1, l, -l, 0, 0, 0, 0]
    x2_plan1 = [-1, 1, -1, 1, -1, 1, -1, 1, 0, 0, l, -l, 0, 0]
    x3_plan1 = [-1, 1, 1, -1, 1, -1, -1, 1, 0, 0, 0, 0, l, -l]
    print('x1:', x1_plan1)
    print('x2:', x2_plan1)
    print('x3:', x3_plan1)
    print('-' * 100)

    # Матриця планування з натуралізованих значень
    x1_plan2 = naturalize(x1_plan1, x1, 1)
    x2_plan2 = naturalize(x2_plan1, x2, 2)
    x3_plan2 = naturalize(x3_plan1, x3, 3)

    # Мультиплікативні значення факторів
    x4_plan2 = [x1_plan2[i] * x2_plan2[i] for i in range(len(x1_plan2))]
    x5_plan2 = [x1_plan2[i] * x3_plan2[i] for i in range(len(x1_plan2))]
    x6_plan2 = [x2_plan2[i] * x3_plan2[i] for i in range(len(x1_plan2))]
    x7_plan2 = [x1_plan2[i] * x2_plan2[i] * x3_plan2[i] for i in
range(len(x1_plan2))]

    # Квадратичні значення факторів
    x8_plan2 = [x1_plan2[i] ** 2 for i in range(len(x1_plan2))]
    x9_plan2 = [x2_plan2[i] ** 2 for i in range(len(x1_plan2))]
    x10_plan2 = [x3_plan2[i] ** 2 for i in range(len(x1_plan2))]

    print(f'x1: {x1_plan2}')
    print(f'x2: {x2_plan2}')
    print(f'x3: {x3_plan2}')
    print(f'x4: {x4_plan2}')
    print(f'x5: {x5_plan2}')
    print(f'x6: {x6_plan2}')
    print(f'x7: {x7_plan2}')
    print(f'x8: {x8_plan2}')
    print(f'x9: {x9_plan2}')
    print(f'x10: {x10_plan2}')
    print()

    x_matrix = [x1_plan2, x2_plan2, x3_plan2, x4_plan2, x5_plan2, x6_plan2,
x7_plan2, x8_plan2, x9_plan2, x10_plan2]

    y_arr = [[y_func(x_matrix, i) + randint(0, 10) - 5 for i in range(N)] for
_ in range(m)]
    for i in range(len(y_arr)):
        print(f'y{i + 1}: {y_arr[i]}')
```

```
    y_avg = []
    for i in range(len(y_arr[0])):
        current_sum = 0
        for j in range(len(y_arr)):
            current_sum += y_arr[j][i]
        y_avg.append(current_sum / len(y_arr))
    print('y average:', y_avg)
    print('-' * 100)

    mx1 = sum(x1_plan2) / len(x1_plan2)
    mx2 = sum(x2_plan2) / len(x2_plan2)
    mx3 = sum(x3_plan2) / len(x3_plan2)
    mx4 = sum(x4_plan2) / len(x4_plan2)
    mx5 = sum(x5_plan2) / len(x5_plan2)
    mx6 = sum(x6_plan2) / len(x6_plan2)
    mx7 = sum(x7_plan2) / len(x7_plan2)
    mx8 = sum(x8_plan2) / len(x8_plan2)
    mx9 = sum(x9_plan2) / len(x9_plan2)
    mx10 = sum(x10_plan2) / len(x10_plan2)
    my = sum(y_avg) / len(y_avg)

    a1 = sum([y_avg[i] * x1_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
    a11 = mx8
    a12 = mx4
    a13 = mx5
    a14 = sum([x1_plan2[i] * x4_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
    a15 = sum([x1_plan2[i] * x5_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
    a16 = sum([x1_plan2[i] * x6_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
    a17 = sum([x1_plan2[i] * x7_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
    a18 = sum([x1_plan2[i] * x8_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
    a19 = sum([x1_plan2[i] * x9_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)

    a2 = sum([y_avg[i] * x2_plan2[i] for i in range(len(x1_plan2))]) /
len(x2_plan2)
    a21 = a12
    a22 = mx9
    a23 = mx6
    a24 = sum([x2_plan2[i] * x4_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)
    a25 = sum([x2_plan2[i] * x5_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)
    a26 = sum([x2_plan2[i] * x6_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)
    a27 = sum([x2_plan2[i] * x7_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)
    a28 = sum([x2_plan2[i] * x8_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)
    a29 = sum([x2_plan2[i] * x9_plan2[i] for i in range(len(x2_plan2))]) /
len(x2 plan2)

    a3 = sum([y_avg[i] * x3_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
    a31 = a13
    a32 = a23
    a33 = mx10
    a34 = sum([x3_plan2[i] * x4_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
```

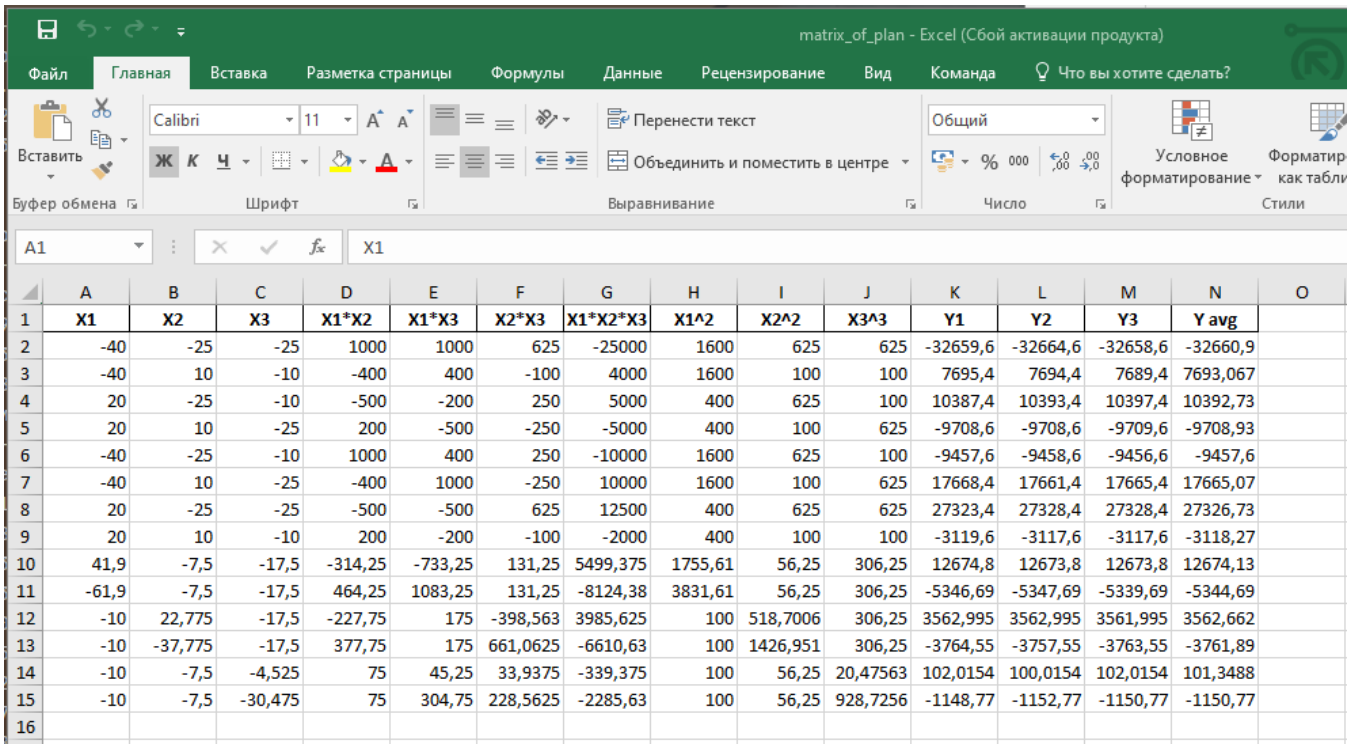
```
    a35 = sum([x3_plan2[i] * x5_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
    a36 = sum([x3_plan2[i] * x6_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
    a37 = sum([x3_plan2[i] * x7_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
    a38 = sum([x3_plan2[i] * x8_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
    a39 = sum([x3_plan2[i] * x9_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)

    a4 = sum([y_avg[i] * x4_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)
    a41 = a14
    a42 = a24
    a43 = a34
    a44 = sum([x4_plan2[i] ** 2 for i in range(len(x4_plan2))]) /
len(x4_plan2)
    a45 = sum([x4_plan2[i] * x5_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)
    a46 = sum([x4_plan2[i] * x6_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)
    a47 = sum([x4_plan2[i] * x7_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)
    a48 = sum([x4_plan2[i] * x8_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)
    a49 = sum([x4_plan2[i] * x9_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)

    a5 = sum([y_avg[i] * x5_plan2[i] for i in range(len(x5_plan2))]) /
len(x5_plan2)
    a51 = a15
    a52 = a25
    a53 = a35
    a54 = a45
    a55 = sum([x5_plan2[i] ** 2 for i in range(len(x5_plan2))]) /
len(x5_plan2)
    a56 = sum([x5_plan2[i] * x6_plan2[i] for i in range(len(x5_plan2))]) /
len(x5_plan2)
    a57 = sum([x5_plan2[i] * x7_plan2[i] for i in range(len(x5_plan2))]) /
len(x5_plan2)
    a58 = sum([x5_plan2[i] * x8_plan2[i] for i in range(len(x5_plan2))]) /
len(x5_plan2)
    a59 = sum([x5_plan2[i] * x9_plan2[i] for i in range(len(x5_plan2))]) /
len(x5_plan2)

    a6 = sum([y_avg[i] * x6_plan2[i] for i in range(len(x6_plan2))]) /
len(x6_plan2)
    a61 = a16
    a62 = a26
    a63 = a36
    a64 = a46
    a65 = a56
    a66 = sum([x6_plan2[i] ** 2 for i in range(len(x6_plan2))]) /
len(x6_plan2)
    a67 = sum([x6 plan2[i] * x7 plan2[i] for i in range(len(x6 plan2))]) /
len(x6_plan2)
    a68 = sum([x6_plan2[i] * x8_plan2[i] for i in range(len(x6_plan2))]) /
len(x6_plan2)
    a69 = sum([x6_plan2[i] * x9_plan2[i] for i in range(len(x6_plan2))]) /
len(x6_plan2)

    a7 = sum([y_avg[i] * x7_plan2[i] for i in range(len(x7_plan2))]) /
len(x7_plan2)
```

```
    a71 = a17
    a72 = a27
    a73 = a37
    a74 = a47
    a75 = a57
    a76 = a67
    a77 = sum([x7_plan2[i] ** 2 for i in range(len(x7_plan2))]) /
len(x7_plan2)
    a78 = sum([x7_plan2[i] * x8_plan2[i] for i in range(len(x7_plan2))]) /
len(x7_plan2)
    a79 = sum([x7_plan2[i] * x9_plan2[i] for i in range(len(x7_plan2))]) /
len(x7_plan2)

    a8 = sum([y_avg[i] * x8_plan2[i] for i in range(len(x8_plan2))]) /
len(x8_plan2)
    a81 = a18
    a82 = a28
    a83 = a38
    a84 = a48
    a85 = a58
    a86 = a68
    a87 = a78
    a88 = sum([x8_plan2[i] ** 2 for i in range(len(x8_plan2))]) /
len(x8_plan2)
    a89 = sum([x8_plan2[i] * x9_plan2[i] for i in range(len(x8_plan2))]) /
len(x8_plan2)

    a9 = sum([y_avg[i] * x9_plan2[i] for i in range(len(x9_plan2))]) /
len(x9_plan2)
    a91 = a19
    a92 = a29
    a93 = a39
    a94 = a49
    a95 = a59
    a96 = a69
    a97 = a79
    a98 = a89
    a99 = sum([x9_plan2[i] ** 2 for i in range(len(x9_plan2))]) /
len(x9_plan2)

    a10 = sum([y_avg[i] * x10_plan2[i] for i in range(len(x10_plan2))]) /
len(x10_plan2)
    a101 = sum([x10_plan2[i] * x1_plan2[i] for i in range(len(x10_plan2))]) /
len(x10_plan2)
    a102 = sum([x10_plan2[i] * x2_plan2[i] for i in range(len(x10_plan2))]) /
len(x10_plan2)
    a103 = sum([x10_plan2[i] * x3_plan2[i] for i in range(len(x10_plan2))]) /
len(x10_plan2)
    a104 = sum([x10_plan2[i] * x4_plan2[i] for i in range(len(x10_plan2))]) /
len(x10_plan2)
    a105 = sum([x10_plan2[i] * x5_plan2[i] for i in range(len(x10_plan2))]) /
len(x10_plan2)
    a106 = sum([x10_plan2[i] * x6_plan2[i] for i in range(len(x10_plan2))]) /
len(x10_plan2)
    a107 = sum([x10_plan2[i] * x7_plan2[i] for i in range(len(x10_plan2))]) /
len(x10 plan2)
    a108 = sum([x10_plan2[i] * x8_plan2[i] for i in range(len(x10_plan2))]) /
len(x10_plan2)
    a109 = sum([x10_plan2[i] * x9_plan2[i] for i in range(len(x10_plan2))]) /
len(x10_plan2)
    a1010 = sum([x10_plan2[i] ** 2 for i in range(len(x10_plan2))]) /
len(x10_plan2)

    main_matrix = [[1, mx1, mx2, mx3, mx4, mx5, mx6, mx7, mx8, mx9, mx10],
```

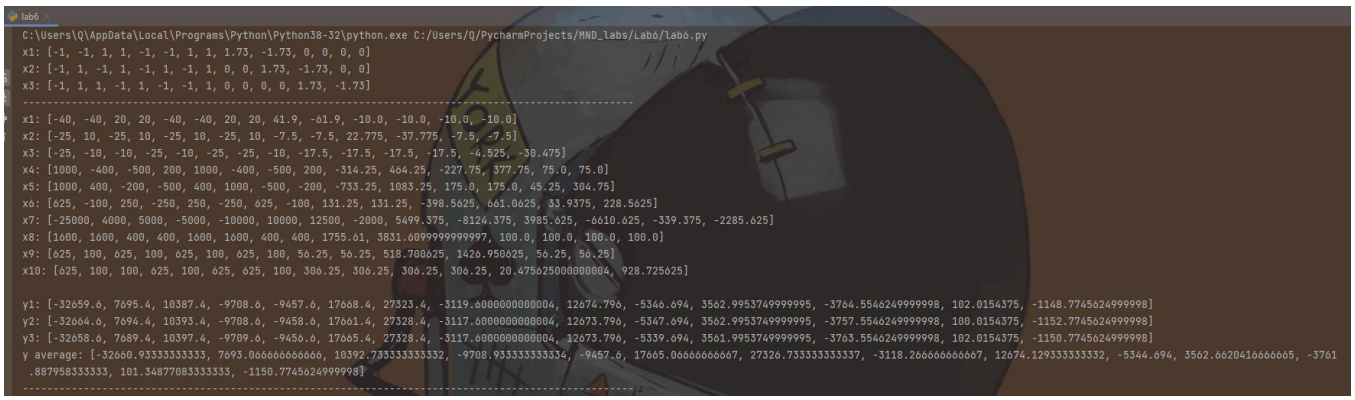
```
                  [mx1, a11, a21, a31, a41, a51, a61, a71, a81, a91, a101],
                  [mx2, a12, a22, a32, a42, a52, a62, a72, a82, a92, a102],
                  [mx3, a13, a23, a33, a43, a53, a63, a73, a83, a93, a103],
                  [mx4, a14, a24, a34, a44, a54, a64, a74, a84, a94, a104],
                  [mx5, a15, a25, a35, a45, a55, a65, a75, a85, a95, a105],
                  [mx6, a16, a26, a36, a46, a56, a66, a76, a86, a96, a106],
                  [mx7, a17, a27, a37, a47, a57, a67, a77, a87, a97, a107],
                  [mx8, a18, a28, a38, a48, a58, a68, a78, a88, a98, a108],
                  [mx9, a19, a29, a39, a49, a59, a69, a79, a89, a99, a109],
                  [mx10, a101, a102, a103, a104, a105, a106, a107, a108,
a109, a1010]]

    column_to_change = [my, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10]
    main_determinant = det(main_matrix)

    matrices = []
    for i in range(len(main_matrix[0])):
        new_matrix = deepcopy(main_matrix)
        for j in range(len(main_matrix)):
            new_matrix[j][i] = column_to_change[j]
        matrices.append(new_matrix)

    print('Знаходження коефіцієнтів рівняння регресії')
    b_list = []
    for i in range(len(matrices)):
        b_list.append(det(matrices[i]) / main_determinant)
    print(f'b: {b_list}')

    print('Підстановка отриманих коефіцієнтів у рівняння регресії')
    y_list = []
    for i in range(len(x1_plan2)):
        y = b_list[0] + b_list[1] * x1_plan2[i] + b_list[2] * x2_plan2[i] +
b_list[3] * x3_plan2[i] + \
            b_list[4] * x4_plan2[i] + b_list[5] * x5_plan2[i] + b_list[6] *
x6_plan2[i] + b_list[7] * x7_plan2[i] + \
            b_list[8] * x8_plan2[i] + b_list[9] * x9_plan2[i] + b_list[10] *
x10_plan2[i]
        y_list.append(y)
        print(f'y = {y}; y avg = {y_avg[i]}')
    print('-' * 100)

    # Запис до файлу матриці планування
    headers = ['X1', 'X2', 'X3', 'X1*X2', 'X1*X3', 'X2*X3', 'X1*X2*X3',
'X1^2', 'X2^2', 'X3^3']
    for i in range(m):
        headers.append(f'Y{i+1}')
    headers.append('Y avg')

    matrix_of_experiment = [*x_matrix, *y_arr, y_avg]
    data = {header: array for header, array in list(zip(headers,
matrix_of_experiment))}
    df = pd.DataFrame(data)
    df.to_excel('matrix_of_plan.xlsx', index=False)

    dispersion = cochrans_test(y_arr, y_avg, m)
    print('-' * 100)
    if dispersion:
        t_arr, s2b = students_test(x0_plan1, x1_plan1, x2_plan1, x3_plan1,
y_avg, dispersion, m)

        b_arr = []
        for i in range(len(b_list)):
            b = b_list[i] if t_arr[i] != 0 else 0
            b_arr.append(b)
```

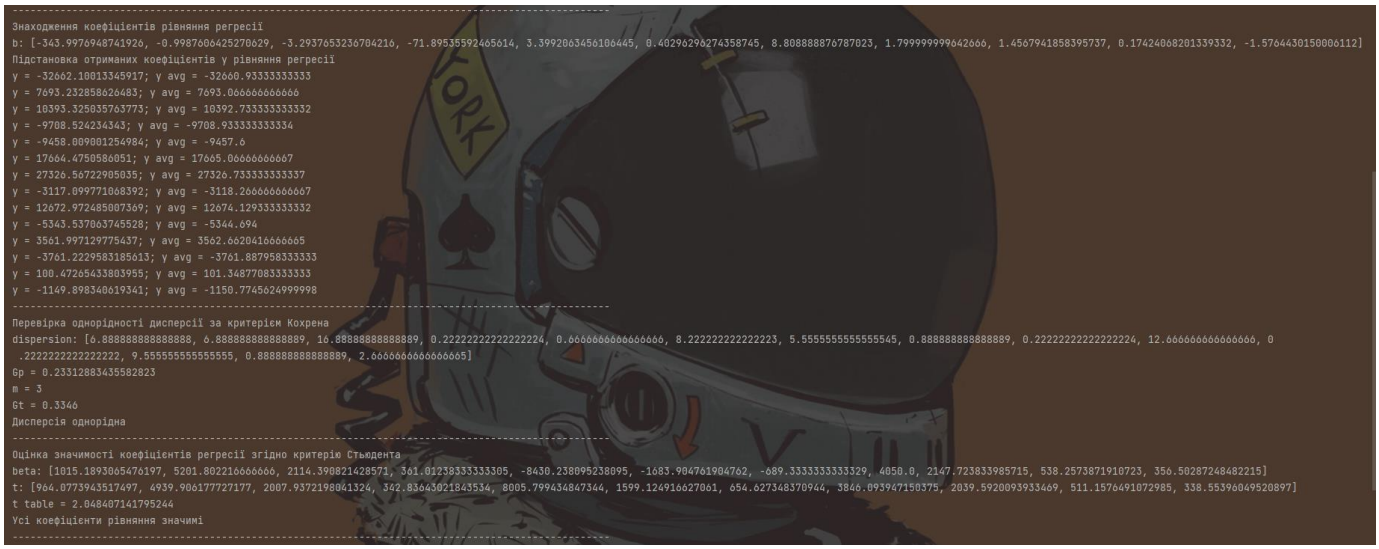
```
        print('-' * 100)

        print('Підстановка коефіцієнтів у спрощене рівняння регресії')
        y_res = []
        for i in range(N):
            y = b_arr[0] + b_arr[1] * x1_plan2[i] + b_arr[2] * x2_plan2[i] +
b_arr[3] * x3_plan2[i] + \
                b_arr[4] * x4_plan2[i] + b_arr[5] * x5_plan2[i] + b_arr[6] *
x6_plan2[i] + b_arr[7] * x7_plan2[i] +\
                b_arr[8] * x8_plan2[i] + b_arr[9] * x9_plan2[i] + b_arr[10] *
x10_plan2[i]
            print(f'y = {y}; y avg = {y_avg[i]}')
            y_res.append(y)

        print('-' * 100)
        fishers_test(b_arr, s2b, y_avg, y_res, m)
    else:
        print('Збільшуємо кількість дослідів')
        main(m+1)
        exit()


if __name__ == '__main__':
    N = 14
    main()
```

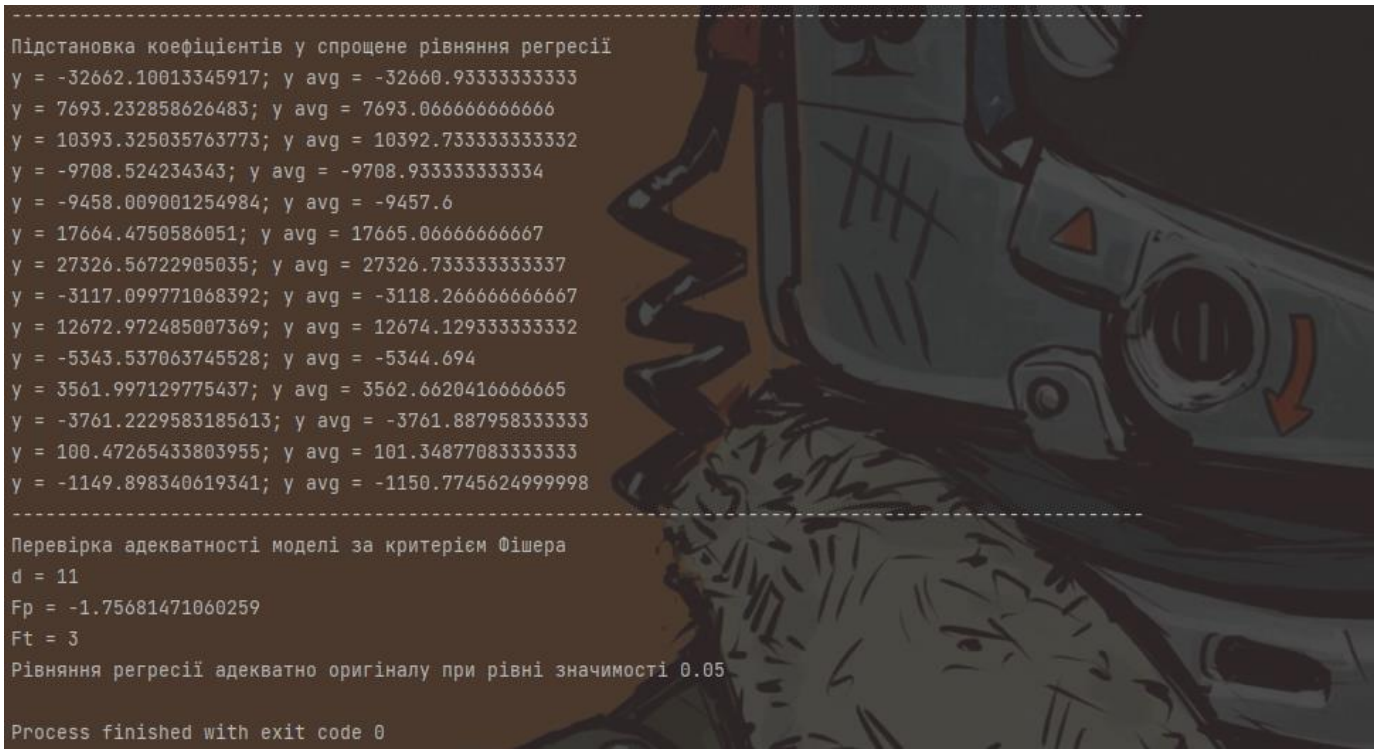
Результати роботи програми



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	X1	X2	X3	X1*X2	X1*X3	X2*X3	X1*X2*X3	X1^2	X2^2	X3^3	Y1	Y2	Y3	Y avg	
2	-40	-25	-25	1000	1000	625	-25000	1600	625	625	-32659,6	-32664,6	-32658,6	-32660,9	
3	-40	10	-10	-400	400	-100	4000	1600	100	100	7695,4	7694,4	7689,4	7693,067	
4	20	-25	-10	-500	-200	250	5000	400	625	100	10387,4	10393,4	10397,4	10392,73	
5	20	10	-25	200	-500	-250	-5000	400	100	625	-9708,6	-9708,6	-9709,6	-9708,93	
6	-40	-25	-10	1000	400	250	-10000	1600	625	100	-9457,6	-9458,6	-9456,6	-9457,6	
7	-40	10	-25	-400	1000	-250	10000	1600	100	625	17668,4	17661,4	17665,4	17665,07	
8	20	-25	-25	-500	-500	625	12500	400	625	625	27323,4	27328,4	27328,4	27326,73	
9	20	10	-10	200	-200	-100	-2000	400	100	100	-3119,6	-3117,6	-3117,6	-3118,27	
10	41,9	-7,5	-17,5	-314,25	-733,25	131,25	5499,375	1755,61	56,25	306,25	12674,8	12673,8	12673,8	12674,13	
11	-61,9	-7,5	-17,5	464,25	1083,25	131,25	-8124,38	3831,61	56,25	306,25	-5346,69	-5347,69	-5339,69	-5344,69	
12	-10	22,775	-17,5	-227,75	175	-398,563	3985,625	100	518,7006	306,25	3562,995	3562,995	3561,995	3562,662	
13	-10	-37,775	-17,5	377,75	175	661,0625	-6610,63	100	1426,951	306,25	-3764,55	-3757,55	-3763,55	-3761,89	
14	-10	-7,5	-4,525	75	45,25	33,9375	-339,375	100	56,25	20,47563	102,0154	100,0154	102,0154	101,3488	
15	-10	-7,5	-30,475	75	304,75	228,5625	-2285,63	100	56,25	928,7256	-1148,77	-1152,77	-1150,77	-1150,77	
16															


lab6
C:\Users\Q\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Q/PycharmProjects/MND_labs/Lab6/lab6.py
x1: [-1, -1, 1, 1, -1, -1, 1, 1, 1.73, -1.73, 0, 0, 0, 0]
x2: [-1, 1, -1, 1, -1, 1, -1, 1, 0, 0, 1.73, -1.73, 0, 0]
x3: [-1, 1, 1, -1, 1, -1, -1, 1, 0, 0, 0, 0, 1.73, -1.73]
x1: [-40, -40, 20, 20, -40, -40, 20, 20, 41.9, -61.9, -10.0, -10.0, -10.0, -10.0]
x2: [-25, 10, -25, 10, -25, 10, -25, 10, -7.5, -7.5, 22.775, -37.775, -7.5, -7.5]
x3: [-25, -10, -10, -25, -10, -25, -25, -10, -17.5, -17.5, -17.5, -17.5, -4.525, -30.475]
x4: [1000, -400, -500, 200, 1000, -400, -500, 200, -314.25, 464.25, -227.75, 377.75, 75.0, 75.0]
x5: [1000, 400, -200, -500, 400, 1000, -500, -200, -733.25, 1083.25, 175.0, 175.0, 45.25, 304.75]
x6: [625, -100, 250, -250, 250, -250, 625, -100, 131.25, 131.25, -398.5625, 661.0625, 33.9375, 228.5625]
x7: [-25000, 4000, 5000, -5000, -10000, 10000, 12500, -2000, 5499.375, -8124.375, 3985.625, -6610.625, -339.375, -2285.625]
x8: [1600, 1600, 400, 400, 1600, 1600, 400, 400, 1755.61, 3831.6099999999997, 100.0, 100.0, 100.0, 100.0]
x9: [625, 100, 625, 100, 625, 100, 625, 100, 56.25, 56.25, 518.700625, 1426.950625, 56.25, 56.25]
x10: [625, 100, 100, 625, 100, 625, 625, 100, 306.25, 306.25, 306.25, 306.25, 20.475625000000004, 928.725625]
y1: [-32659.6, 7695.4, 10387.4, -9708.6, -9457.6, 17668.4, 27323.4, -3119.6000000000004, 12674.796, -5346.694, 3562.9953749999995, -3764.5546249999998, 102.0154375, -1148.7745624999998]
y2: [-32664.6, 7694.4, 10393.4, -9708.6, -9458.6, 17661.4, 27328.4, -3117.6000000000004, 12673.796, -5347.694, 3562.9953749999995, -3757.5546249999998, 100.0154375, -1152.7745624999998]
y3: [-32658.6, 7689.4, 10397.4, -9709.6, -9456.6, 17665.4, 27328.4, -3117.6000000000004, 12673.796, -5339.694, 3561.9953749999995, -3763.5546249999998, 102.0154375, -1150.7745624999998]
y average: [-32660.93333333333, 7693.066666666666, 10392.733333333332, -9708.933333333334, -9457.6, 17665.06666666667, 27326.733333333337, -3118.2666666666667, 12674.129333333332, -5344.694, 3562.6620416666665, -3761.887958333333, 101.34877083333333, -1150.7745624999998]


Знаходження коефіцієнтів рівняння регресії
b: [-343.9976948741926, -0.9987606425270629, -3.2937653236704216, -71.89535592465614, 3.3992063456106445, 0.40296296274358745, 8.808888876787023, 1.799999999642666, 1.4567941858395737, 0.17424068201339332, -1.5764430150006112]
Підстановка отриманих коефіцієнтів у рівняння регресії
y = -32662.10013345917; y avg = -32660.93333333333
y = 7693.232858626483; y avg = 7693.066666666666
y = 10393.325035763773; y avg = 10392.733333333332
y = -9708.524234343; y avg = -9708.933333333334
y = -9458.009001254984; y avg = -9457.6
y = 17664.4750586051; y avg = 17665.06666666667
y = 27326.56722905035; y avg = 27326.733333333337
y = -3117.099771068392; y avg = -3118.266666666667
y = 12672.972485007369; y avg = 12674.129333333332
y = -5343.537063745528; y avg = -5344.694
y = 3561.997129775437; y avg = 3562.6620416666665
y = -3761.2229583185613; y avg = -3761.887958333333
y = 100.47265433803955; y avg = 101.34877083333333
y = -1149.898340619341; y avg = -1150.7745624999998
Перевірка однорідності дисперсії за критерієм Кохрена
dispersion: [6.888888888888888, 6.888888888888889, 16.88888888888889, 0.22222222222222224, 0.6666666666666666, 8.222222222222223, 5.5555555555555545, 0.888888888888889, 0.22222222222222224, 12.666666666666666, 0.2222222222222222, 9.555555555555555, 0.888888888888889, 2.666666666666665]
Gp = 0.23312883435582823
m = 3
Gt = 0.3346
Дисперсія однорідна
Оцінка значимості коефіцієнтів регресії згідно критерію Стьюдента
beta: [1015.1893065476197, 5201.802216666666, 2114.390821428571, 361.01238333333305, -8430.238095238095, -1683.904761904762, -689.3333333333329, 4050.0, 2147.723833985715, 538.2573871910723, 356.50287248482215]
t: [964.0773943517497, 4939.906177727177, 2007.9372198041324, 342.83643021843534, 8005.799434847344, 1599.124916627061, 654.627348370944, 3846.093947150375, 2039.5920093933469, 511.1576491072985, 338.55396049520897]
t table = 2.048407141795244
Усі коефіцієнти рівняння значимі


Підстановка коефіцієнтів у спрощене рівняння регресії
y = -32662.10013345917; y avg = -32660.93333333333
y = 7693.232858626483; y avg = 7693.066666666666
y = 10393.325035763773; y avg = 10392.733333333332
y = -9708.524234343; y avg = -9708.933333333334
y = -9458.009001254984; y avg = -9457.6
y = 17664.4750586051; y avg = 17665.06666666667
y = 27326.56722905035; y avg = 27326.733333333337
y = -3117.099771068392; y avg = -3118.266666666667
y = 12672.972485007369; y avg = 12674.129333333332
y = -5343.537063745528; y avg = -5344.694
y = 3561.997129775437; y avg = 3562.6620416666665
y = -3761.2229583185613; y avg = -3761.887958333333
y = 100.47265433803955; y avg = 101.34877083333333
y = -1149.898340619341; y avg = -1150.7745624999998
Перевірка адекватності моделі за критерієм Фішера
d = 11
Fp = -1.75681471060259
Ft = 3
Рівняння регресії адекватно оригіналу при рівні значимості 0.05
Process finished with exit code 0

Висновок:

У ході лабораторної роботи я змоделював трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи рототабельний центральний композиційний план, провів 3 статистичні перевірки. По результатам перевірок, дисперсія виявилась однорідною при початковому $m = 3$, а у рівнянні регресії не знайшлося не значних коефіцієнтів, тож воно залишилось таким як і до перевірок. В решті, я отримав рівняння регресії, яке адекватне оригіналу при заданому рівні значимості для опису об'єкту.