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

з дисципліни « Методи оптимізації та планування » на тему

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

Виконав:
студент ІІ курсу ФІОТ
групи ІО – 92
Грисюк Дмитро
Номер залікової книжки: ІО - 9207

Перевірив:
ст. вик. Регіда П.Г.

Київ – 2021

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

Завдання на лабораторну роботу:

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 +1; -1;+; -; 0 для 1, 2, 3.

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.

204	15	45	15	50	15	30	3,5+6,6*x1+3,9*x2+1,8*x3+5,3*x1*x1+0,5*x2*x2+4,3*x3*x3+6,0*x1*x2+0,8*x1*x3+9,4*x2*x3+3,0*x1*x2*x3

Роздруківка тексту програми:

```
from math import sqrt
from time import process_time
from scipy.stats import f, t
from functools import partial
from random import randrange
from numpy.linalg import solve

x1, x2, x3 = [15, 45], [15, 50], [15, 30]
m, N, l = 2, 15, 1.73  # кількість повторень кожної комбінації &  кількість
повторення дослідів

x_avg = [(max(x1) + max(x2) + max(x3)) / 3, (min(x1) + min(x2) + min(x3)) / 3]  #
Xcp(max) & Xcp(min)
xo = [(min(x1) + max(x1)) / 2, (min(x2) + max(x2)) / 2, (min(x3) + max(x3)) / 2]  #
Xoi
delta_x = [max(x1) - xo[0], max(x1) - xo[1], max(x1) - xo[2]]  # delta Xi

y_range = [200 + int(max(x_avg)), 200 + int(min(x_avg))]  # Yi(max) & Yi(min)

xn = [[-1, -1, -1, -1, +1, +1, +1, +1, -1.73, 1.73, 0, 0, 0, 0, 0], # нормовані
значення факторів
      [-1, -1, +1, +1, -1, -1, +1, +1, 0, 0, -1.73, 1.73, 0, 0, 0],
      [-1, +1, -1, +1, -1, +1, -1, +1, 0, 0, 0, 0, -1.73, 1.73, 0]]

xx = [[int(x * y) for x, y in zip(xn[0], xn[1])],  # нормовані значення факторів для
```

```
ефекту взаємодії
      [int(x * y) for x, y in zip(xn[0], xn[2])],
      [int(x * y) for x, y in zip(xn[1], xn[2])]]

xxx = [int(x * y * z) for x, y, z in zip(xn[0], xn[1], xn[2])]

x_xn = [[round(xn[j][i] ** 2, 3) for i in range(N)] for j in range(3)]  # нормовані
знач. факторів для квад. членів

x = [[min(x1), min(x1), min(x1), min(x1), max(x1), max(x1), max(x1), max(x1), round(-
1 * delta_x[0] + xo[0], 3),
      round(1 * delta_x[0] + xo[0], 3), xo[0], xo[0], xo[0], xo[0], xo[0]],  #
натуральні значення факторів
     [min(x2), min(x2), max(x2), max(x2), min(x2), min(x2), max(x2), max(x2), xo[1],
xo[1],
      round(-1 * delta_x[1] + xo[1], 3), round(1 * delta_x[1] + xo[1], 3), xo[1],
xo[1], xo[1]],
     [min(x3), max(x3), min(x3), max(x3), max(x3), min(x3), max(x3), min(x3), xo[2],
xo[2], xo[2], xo[2],
      round(-1 * delta_x[2] + xo[2], 3), round(1 * delta_x[2] + xo[2], 3), xo[2]]]

xx2 = [[round(x * y, 3) for x, y in zip(x[0], x[1])],  # натуральні значення факторів
для ефекту взаємодії
       [round(x * y, 3) for x, y in zip(x[0], x[2])],
       [round(x * y, 3) for x, y in zip(x[1], x[2])]]

xxx2 = [round(x * y * z, 3) for x, y, z in zip(x[0], x[1], x[2])]

x_x = [[round(x[j][i] ** 2, 3) for i in range(N)] for j in range(3)]  # натуральні
значення факторів для квадрат. членів

while True:
    start_time = process_time()
    # формування Y
    y = [[round(3.5 + 6.6 * x[0][j] + 3.9 * x[1][j] + 1.8 * x[2][j] + 5.3 * x[0][j] *
x[0][j] + 0.5 * x[1][j] * x[1][j] +
                4.3 * x[2][j] * x[2][j] + 6.0 * x[0][j] * x[1][j] + 0.8 * x[0][j] *
x[2][j] + 9.4 * x[1][j] * x[2][j] +
                3.0 * x[0][j] * x[1][j] * x[2][j] + randrange(0, 10) - 5, 2) for i in
range(m)] for j in range(N)]
    arr_avg = lambda arr: round(sum(arr) / len(arr), 4)
    y_avg = list(map(arr_avg, y))  # середнє значення Y

    dispersions = [sum([((y[i][j] - y_avg[i]) ** 2) / m for j in range(m)]) for i in
range(N)]  # дисперсії по рядках
    x_matrix = x + xx2 + [xxx2] + x_x  # повна матриця з натуральними значеннями
факторів
    norm_matrix = xn + xx + [xxx] + x_xn  # повна матриця з нормованими значеннями
факторів

    mx = list(map(arr_avg, x_matrix))  # середні значення x по колонкам
    my = sum(y_avg) / N  # середнє значення Y_avg

    # ======================================== Форматування таблиці
================================================

    table_factors_1 = ["X1", "X2", "X3"]
    table_factors_2 = ["X1X2", "X1X3", "X2X3", "X1X2X3", "X1^2", "X2^2", "X3^2"]
    table_y = ["Y{}".format(i + 1) for i in range(m)]
    other = ["#", "Y"]

    header_format_norm = "+{0:=^3}" + "+{0:=^8}" * (len(table_factors_1)) +
```

```
"+{0:=^8s}" * (len(table_factors_2))
    header_format = "+{0:=^3}" + "+{0:=^8}" * (len(table_factors_1)) + "+{0:=^10s}" *
(len(table_factors_2)) + "+{0:=^10s}" * (len(table_y)) + "+{0:=^10s}"
    row_format_norm = "|{:^3}" + "|{:^8}" * (len(table_factors_1)) + "|{:^8}" *
(len(table_factors_2))
    row_format = "|{:^3}" + "|{:^8}" * (len(table_factors_1)) + "|{:^10}" *
(len(table_factors_2)) + "|{:^10}" * (len(table_y)) + "|{:^10}"
    separator_format_norm = "+{0:-^3s}" + "+{0:-^8s}" * (len(table_factors_1)) +
"+{0:-^8s}" * (len(table_factors_2))
    separator_format = "+{0:-^3s}" + "+{0:-^8s}" * (len(table_factors_1)) + "+{0:-
^10s}" * (len(table_factors_2)) + "+{0:-^10s}" * (len(table_y)) + "+{0:-^10s}"
    my_sep_norm = "|{:^93s}|\n"
    my_sep = "|{:^140s}|\n" if m == 2 else "|{:^151s}|\n"
    # ========================================= Нормальні значення
=====================================================
    print(header_format_norm.format("=") + "+\n" + my_sep_norm.format("Матриця ПФЕ
(нормальні значення факторів)") +
          header_format_norm.format("=") + "+\n" + row_format_norm.format(other[0],
*table_factors_1, *table_factors_2)
          + "|\n" + header_format_norm.format("=") + "+")

    for i in range(N):
        print("|{:^3}|".format(i + 1), end="")
        for j in range(3): print("{:^+8}|".format(xn[j][i]), end="")
        for j in range(3): print("{:^+8}|".format(xx[j][i]), end="")
        print("{:^+8}|".format(xxx[i]), end="")
        for j in range(3): print("{:^+8}|".format(x_xn[j][i]), end="")
        print()

    print(separator_format_norm.format("-") + "+\n\n")

    # ========================================= Натуральні значення
=====================================================
    print(header_format.format("=") + "+\n" + my_sep.format("Матриця ПФЕ (натуральні
значення факторів)") +
          header_format.format("=") + "+\n" + row_format.format(other[0],
*table_factors_1, *table_factors_2, *table_y,
                                                                other[1]) + "|\n" +
header_format.format("=") + "+")

    for i in range(N):
        print("|{:^3}|".format(i + 1), end="")
        for j in range(3): print("{:^ 8}|".format(x[j][i]), end="")
        for j in range(3): print("{:^ 10}|".format(xx2[j][i]), end="")
        print("{:^ 10}|".format(xxx2[i]), end="")
        for j in range(3): print("{:^ 10}|".format(x_x[j][i]), end="")
        for j in range(m): print("{:^ 10}|".format(y[i][j]), end="")
        print("{:^10.2f}|".format(y_avg[i]))


    def a(first, second): return sum([x_matrix[first - 1][j] * x_matrix[second -
1][j] / N for j in range(N)])
    def find_a(num): return sum([y_avg[j] * x_matrix[num - 1][j] / N for j in
range(N)])
    def check(b_lst, k):
        return b_lst[0] + b_lst[1] * x_matrix[0][k] + b_lst[2] * x_matrix[1][k] +
b_lst[3] * x_matrix[2][k] + \
               b_lst[4] * x_matrix[3][k] + b_lst[5] * x_matrix[4][k] + b_lst[6] *
x_matrix[5][k] + \
               b_lst[7] * x_matrix[6][k] + b_lst[8] * x_matrix[7][k] + b_lst[9] *
x_matrix[8][k] + \
               b_lst[10] * x_matrix[9][k]
```

```
    unknown = [[1, mx[0], mx[1], mx[2], mx[3], mx[4], mx[5], mx[6], mx[7], mx[8],
mx[9]],  # ліва частина рівнянь з невідомими для пошуку коефіцієнтів b (приклад в
методі)
               [mx[0], a(1, 1), a(1, 2), a(1, 3), a(1, 4), a(1, 5), a(1, 6), a(1, 7),
a(1, 8), a(1, 9), a(1, 10)],
               [mx[1], a(2, 1), a(2, 2), a(2, 3), a(2, 4), a(2, 5), a(2, 6), a(2, 7),
a(2, 8), a(2, 9), a(2, 10)],
               [mx[2], a(3, 1), a(3, 2), a(3, 3), a(3, 4), a(3, 5), a(3, 6), a(3, 7),
a(3, 8), a(3, 9), a(3, 10)],
               [mx[3], a(4, 1), a(4, 2), a(4, 3), a(4, 4), a(4, 5), a(4, 6), a(4, 7),
a(4, 8), a(4, 9), a(4, 10)],
               [mx[4], a(5, 1), a(5, 2), a(5, 3), a(5, 4), a(5, 5), a(5, 6), a(5, 7),
a(5, 8), a(5, 9), a(5, 10)],
               [mx[5], a(6, 1), a(6, 2), a(6, 3), a(6, 4), a(6, 5), a(6, 6), a(6, 7),
a(6, 8), a(6, 9), a(6, 10)],
               [mx[6], a(7, 1), a(7, 2), a(7, 3), a(7, 4), a(7, 5), a(7, 6), a(7, 7),
a(7, 8), a(7, 9), a(7, 10)],
               [mx[7], a(8, 1), a(8, 2), a(8, 3), a(8, 4), a(8, 5), a(8, 6), a(8, 7),
a(8, 8), a(8, 9), a(8, 10)],
               [mx[8], a(9, 1), a(9, 2), a(9, 3), a(9, 4), a(9, 5), a(9, 6), a(9, 7),
a(9, 8), a(9, 9), a(9, 10)],
               [mx[9], a(10, 1), a(10, 2), a(10, 3), a(10, 4), a(10, 5), a(10, 6),
a(10, 7), a(10, 8), a(10, 9), a(10, 10)]]
    known = [my, find_a(1), find_a(2), find_a(3), find_a(4), find_a(5), find_a(6),
find_a(7), find_a(8), find_a(9), find_a(10)]  # знаходення відомих значень a1, a2,
...

    b = solve(unknown, known)
    print(separator_format.format("-") + f"+\n\n\tОтримане рівняння регресії при
m={m}:\n"
                                         f"ŷ = {b[0]:.3f} + {b[1]:.3f}*X1 +
{b[2]:.3f}*X2 + "
                                         f"{b[3]:.3f}*X3 + {b[4]:.3f}*X1X2 +
{b[5]:.3f}*X1X3 + "
                                         f"{b[6]:.3f}*X2X3 + {b[7]:.3f}*X1X2X3 +
{b[8]:.3f}*X11^2 + "
                                         f"{b[9]:.3f}*X22^2 +
{b[10]:.3f}*X33^2\n\n\tПеревірка:")
    for i in range(N): print("ŷ{} = {:.3f} ≈ {:.3f}".format((i + 1), check(b, i),
y_avg[i]))

    # ========================================= Критерій Кохрена
============================================================
    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

    f1, f2 = m - 1, N
    f3 = f1 * f2
    fisher = table_fisher(0.95, N, m, 1)
    Gp = max(dispersions) / sum(dispersions)
    Gt = fisher / (fisher + (m - 1) - 2)

    print("\nОднорідність дисперсії (критерій Кохрена): ")
    print(f"Gp = {Gp}\nGt = {Gt}")
    if Gp < Gt:
        print("\nДисперсія однорідна (Gp < Gt)")
```

```
        D_beta = sum(dispersions) / (N * N * m)
        Sb = sqrt(abs(D_beta))
        beta = [sum([(y_avg[j] * norm_matrix[i][j]) / N for j in range(N)]) for i in
range(len(norm_matrix))]

        t_list = [abs(i) / Sb for i in beta]
        student = partial(t.ppf, q=1-0.025)
        d, T = 0, student(df=f3)
        print("\nt табличне = ", T)

        for i in range(len(t_list)):
            if t_list[i] < T:
                b[i] = 0
                print("\tt{} = {} => коефіцієнт незначимий, його слід виключити з
рів-ня регресії".format(i, t_list[i]))
            else:
                print("\tt{} = {} => коефіцієнт значимий".format(i, t_list[i]))
                d += 1

        print("\nОтже, кіл-ть значимих коеф. d =", d, "\n\n\tРів-ня регресії з
урахуванням критерія Стьюдента:\nŷ = ", end="")
        print("{:.3f}".format(b[0]), end="") if b[0] != 0 else None
        for i in range(1, 10):
            print(" + {:.3f}*{}".format(b[i], (table_factors_1 +
table_factors_2)[i]), end="") if b[i] != 0 else None
        print("\n\n\tПеревірка при підстановці в спрощене рів-ня регресії:")
        for i in range(N): print("y`{} = {:.3f} ≈ {:.3f}".format((i + 1), check(b,
i), y_avg[i]))

        f4 = N - d
        fisher_sum = sum([(check(b, i) - y_avg[i]) ** 2 for i in range(N)])
        D_ad = (m / f4) * fisher_sum

        fisher = partial(f.ppf, q=1-0.05)
        Fp = D_ad / sum(dispersions) / N
        Ft = fisher(dfn=f4, dfd=f3)
        print("\nКритерій Фішера:")
        if Fp > Ft:
            print("\tРівняння регресії неадекватне (Ft < Fp).")
            break
        else:
            print("\tРівняння регресії адекватне (Ft > Fp)!")
            print("\n\n--- Час виконання програми: %s секунд ---" % (process_time() -
start_time))
            break

    else:
        print("Дисперсія неоднорідна (Gp > Gt), збільшуємо m, повторюємо операції")
        m += 1
```

Результати роботи програми:

Матриця ПФЕ (нормальні значення факторів)

#	X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1^2	X2^2	X3^2
1	-1	-1	-1	+1	+1	+1	-1	+1	+1	+1
2	-1	-1	+1	+1	-1	-1	+1	+1	+1	+1
3	-1	+1	-1	-1	+1	-1	+1	+1	+1	+1
4	-1	+1	+1	-1	-1	+1	-1	+1	+1	+1
5	+1	-1	-1	-1	-1	+1	+1	+1	+1	+1
6	+1	-1	+1	-1	+1	-1	-1	+1	+1	+1
7	+1	+1	-1	+1	-1	-1	-1	+1	+1	+1
8	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1
9	-1.73	+0	+0	+0	+0	+0	+0	+2.993	+0	+0
10	+1.73	+0	+0	+0	+0	+0	+0	+2.993	+0	+0
11	+0	-1.73	+0	+0	+0	+0	+0	+0	+2.993	+0
12	+0	+1.73	+0	+0	+0	+0	+0	+0	+2.993	+0
13	+0	+0	-1.73	+0	+0	+0	+0	+0	+0	+2.993
14	+0	+0	+1.73	+0	+0	+0	+0	+0	+0	+2.993
15	+0	+0	+0	+0	+0	+0	+0	+0	+0	+0

Матриця ПФЕ (натуральні значення факторів)

#	X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1^2	X2^2	X3^2	Y1	Y2	Y
1	15	15	15	225	225	225	3375	225	225	225	16232.5	16228.5	16230.50
2	15	15	30	225	450	450	6750	225	225	900	31578.0	31579.0	31578.50
3	15	50	15	750	225	750	11250	225	2500	225	49210.5	49216.5	49213.50
4	15	50	30	750	450	1500	22500	225	2500	900	93120.0	93121.0	93120.50
5	45	15	30	675	1350	450	20250	2025	225	900	85234.0	85239.0	85236.50
6	45	15	15	675	675	225	10125	2025	225	225	49279.5	49277.5	49278.50
7	45	50	30	2250	1350	1500	67500	2025	2500	900	247577.0	247580.0	247578.50
8	45	50	15	2250	675	750	33750	2025	2500	225	135807.5	135816.5	135812.00
9	4.05	32.5	22.5	131.625	91.125	731.25	2961.562	16.402	1056.25	506.25	19606.5	19605.5	19606.00
10	55.95	32.5	22.5	1818.375	1258.875	731.25	40913.438	3130.403	1056.25	506.25	161363.57	161365.57	161364.57
11	30.0	10.875	22.5	326.25	675.0	244.688	7340.625	900.0	118.266	506.25	34112.86	34107.86	34110.36
12	30.0	54.125	22.5	1623.75	675.0	1217.812	36534.375	900.0	2929.516	506.25	140197.78	140200.78	140199.28
13	30.0	32.5	-16.425	975.0	-492.75	-533.812	-16014.375	900.0	1056.25	269.781	-40845.3	-40851.3	-40848.30
14	30.0	32.5	61.425	975.0	1842.75	1996.312	59889.375	900.0	1056.25	3773.031	227719.63	227716.63	227718.13
15	30.0	32.5	22.5	975.0	675.0	731.25	21937.5	900.0	1056.25	506.25	86918.0	86917.0	86917.50

Отримане рівняння регресії при m=2:

$$\hat{y} = 5.238 + 6.747*X1 + 3.528*X2 + 1.776*X3 + 6.001*X1X2 + 0.800*X1X3 + 9.399*X2X3 + 3.000*X1X2X3 + 5.297*X11^2 + 0.506*X22^2 + 4.301*X33^2$$

```
    Перевірка:
ŷ1 = 16229.437 ≈ 16230.500
ŷ2 = 31579.206 ≈ 31578.500
ŷ3 = 49213.042 ≈ 49213.500
ŷ4 = 93121.810 ≈ 93120.500
ŷ5 = 85237.522 ≈ 85236.500
ŷ6 = 49277.755 ≈ 49278.500
ŷ7 = 247580.127 ≈ 247578.500
ŷ8 = 135811.859 ≈ 135812.000
ŷ9 = 19605.989 ≈ 19606.000
ŷ10 = 161363.827 ≈ 161364.570
ŷ11 = 34110.599 ≈ 34110.360
ŷ12 = 140197.562 ≈ 140199.280
ŷ13 = -40847.661 ≈ -40848.300
ŷ14 = 227717.407 ≈ 227718.130
ŷ15 = 86917.560 ≈ 86917.500

Однорідність дисперсії (критерій Кохрена):
Gp = 0.313953488372093
Gt = 1.702247191011236

Дисперсія однорідна (Gp < Gt)
```

```
t табличне =  2.131449545559323
    t0 = 100900.5236937403 => коефіцієнт значимий
    t1 = 92787.96196626137 => коефіцієнт значимий
    t2 = 66236.39632750452 => коефіцієнт значимий
    t3 = 27179.607523362636 => коефіцієнт значимий
    t4 = 36447.05767316489 => коефіцієнт значимий
    t5 = 8320.173019686514 => коефіцієнт значимий
    t6 = 18378.09199318791 => коефіцієнт значимий
    t7 = 220058.74015103388 => коефіцієнт значимий
    t8 = 216548.1775083486 => коефіцієнт значимий
    t9 = 223167.87382688504 => коефіцієнт значимий

Отже, кіл-ть значимих коеф. d = 10

    Рів-ня регресії з урахуванням критерія Стьюдента:
ŷ = 5.238 + 6.747*X2 + 3.528*X3 + 1.776*X1X2 + 6.001*X1X3 + 0.800*X2X3 + 9.399*X1X2X3 + 3.000*X1^2 + 5.297*X2^2 + 0.506*X3^2
```

```
    Перевірка при підстановці в спрощене рів-ня регресії:
y`1 = 16229.437 ≈ 16230.500
y`2 = 31579.206 ≈ 31578.500
y`3 = 49213.042 ≈ 49213.500
y`4 = 93121.810 ≈ 93120.500
y`5 = 85237.522 ≈ 85236.500
y`6 = 49277.755 ≈ 49278.500
y`7 = 247580.127 ≈ 247578.500
y`8 = 135811.859 ≈ 135812.000
y`9 = 19605.989 ≈ 19606.000
y`10 = 161363.827 ≈ 161364.570
y`11 = 34110.599 ≈ 34110.360
y`12 = 140197.562 ≈ 140199.280
y`13 = -40847.661 ≈ -40848.300
y`14 = 227717.407 ≈ 227718.130
y`15 = 86917.560 ≈ 86917.500

Критерій Фішера:
    Рівняння регресії адекватне (Ft > Fp)!


--- Час виконання програми: 0.359375 секунд ---
```

Висновки:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії, рівняння регресії з ефектом взаємодії та рівняння регресії з квадратичними членами, складено матрицю планування експерименту, було визначено коефіцієнти рівнянь регресії (натуралізовані та нормовані), для форми з квадратичними членами - натуралізовані, виконано перевірку правильності розрахунку коефіцієнтів рівнянь регресії.