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 II курсу ФІОТ
групи ІО-93
Ільків Максим
Номер залікової книжки: ІО - 9313

Перевірив:
ас. Регіда П.Г.

Київ – 2021

Мета роботи: gровести 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.

Роздруківка тексту програми:

У протоколі наведена частина програми, яка стосується рівняння регресії з квадратичними членами.

```
from math import fabs, sqrt
m = 2
p = 0.95
N = 15
x1_min = 10
x1_max = 60
x2_min = -30
x2_max = 45
x3_min = -30
x3_max = 45
x01 = (x1_max + x1_min) / 2
x02 = (x2_max + x2_min) / 2
x03 = (x3_max + x3_min) / 2
delta_x1 = x1_max - x01
delta_x2 = x2_max - x02
delta_x3 = x3_max - x03

average_y = None
matrix = None
dispersion_b2 = None
student_lst = None
d = None
q = None
f3 = None
```

```
class Perevirku:
    def get_cohren_value(size_of_selections, qty_of_selections, significance):
        from _pydecimal import Decimal
        from scipy.stats import f
        size_of_selections += 1
        partResult1 = significance / (size_of_selections - 1)
        params = [partResult1, qty_of_selections, (size_of_selections - 1 - 1) *
qty_of_selections]
        fisher = f.isf(*params)
        result = fisher / (fisher + (size_of_selections - 1 - 1))
        return Decimal(result).quantize(Decimal('.0001')).__float__()

    def get_student_value(f3, significance):
        from _pydecimal import Decimal
        from scipy.stats import t
        return Decimal(abs(t.ppf(significance / 2,
f3))).quantize(Decimal('.0001')).__float__()

    def get_fisher_value(f3, f4, significance):
        from _pydecimal import Decimal
        from scipy.stats import f
        return Decimal(abs(f.isf(significance, f4,
f3))).quantize(Decimal('.0001')).__float__()


def generate_matrix():
    def f(X1, X2, X3):
        from random import randrange
        y = 0.6 + 5.3 * X1 + 1.4 * X2 + 4.9 * X3 + 4.1 * X1 * X1 + 0.6 * X2 * X2
+ 8.4 * X3 * X3 + 7.8 * X1 * X2 + \
            0.1 * X1 * X3 + 9.3 * X2 * X3 + 7.1 * X1 * X2 * X3 + randrange(0,
10) - 5
        return y

    matrix_with_y = [[f(matrix_x[j][0], matrix_x[j][1], matrix_x[j][2]) for i in
range(m)] for j in range(N)]
    return matrix_with_y


def x(l1, l2, l3):
    x_1 = l1 * delta_x1 + x01
    x_2 = l2 * delta_x2 + x02
    x_3 = l3 * delta_x3 + x03
    return [x_1, x_2, x_3]


def find_average(lst, orientation):
    average = []
    if orientation == 1:
        for rows in range(len(lst)):
            average.append(sum(lst[rows]) / len(lst[rows]))
    else:
        for column in range(len(lst[0])):
            number_lst = []
            for rows in range(len(lst)):
                number_lst.append(lst[rows][column])
            average.append(sum(number_lst) / len(number_lst))
    return average
```

```
def a(first, second):
    need_a = 0
    for j in range(N):
        need_a += matrix_x[j][first - 1] * matrix_x[j][second - 1] / N
    return need_a


def find_known(number):
    need_a = 0
    for j in range(N):
        need_a += average_y[j] * matrix_x[j][number - 1] / 15
    return need_a


def solve(lst_1, lst_2):
    from numpy.linalg import solve
    solver = solve(lst_1, lst_2)
    return solver


def check_result(b_lst, k):
    y_i = b_lst[0] + b_lst[1] * matrix[k][0] + b_lst[2] * matrix[k][1] +
b_lst[3] * matrix[k][2] + \
          b_lst[4] * matrix[k][3] + b_lst[5] * matrix[k][4] + b_lst[6] *
matrix[k][5] + b_lst[7] * matrix[k][6] + \
          b_lst[8] * matrix[k][7] + b_lst[9] * matrix[k][8] + b_lst[10] *
matrix[k][9]
    return y_i


def student_test(b_lst, number_x=10):
    dispersion_b = sqrt(dispersion_b2)
    for column in range(number_x + 1):
        t_practice = 0
        t theoretical = Perevirku.get student value(f3, q)
        for row in range(N):
            if column == 0:
                t_practice += average_y[row] / N
            else:
                t practice += average y[row] * matrix pfe[row][column - 1]
        if fabs(t_practice / dispersion_b) < t_theoretical:
            b_lst[column] = 0
    return b_lst


def fisher_test():
    dispersion_ad = 0
    f4 = N - d
    for row in range(len(average_y)):
        dispersion_ad += (m * (average_y[row] - check_result(student_lst, row)))
/ (N - d)
    F_practice = dispersion_ad / dispersion_b2
    F_theoretical = Perevirku.get_fisher_value(f3, f4, q)
    return F_practice < F_theoretical


matrix_pfe = [
    [-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, -1, -1, -1, +1, +1, +1],
    [+1, +1, +1, +1, +1, +1, +1, +1, +1, +1],
    [-1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],
    [+1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],
    [0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],
    [0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],
    [0, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929],
    [0, 0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
]

matrix_x = [[] for x in range(N)]
for i in range(len(matrix_x)):
    if i < 8:
        x_1 = x1_min if matrix_pfe[i][0] == -1 else x1_max
        x_2 = x2_min if matrix_pfe[i][1] == -1 else x2_max
        x_3 = x3_min if matrix_pfe[i][2] == -1 else x3_max
    else:
        x_lst = x(matrix_pfe[i][0], matrix_pfe[i][1], matrix_pfe[i][2])
        x_1, x_2, x_3 = x_lst
    matrix_x[i] = [x_1, x_2, x_3, x_1 * x_2, x_1 * x_3, x_2 * x_3, x_1 * x_2 *
x_3, x_1 ** 2, x_2 ** 2, x_3 ** 2]

def run_experiment():
    adekvat = False
    odnorid = False

    global average_y
    global matrix
    global dispersion_b2
    global student lst
    global d
    global q
    global m
    global f3
    while not adekvat:
        matrix_y = generate_matrix()
        average_x = find_average(matrix_x, 0)
        average_y = find_average(matrix_y, 1)
        matrix = [(matrix_x[i] + matrix_y[i]) for i in range(N)]
        mx_i = average_x
        my = sum(average_y) / 15

        unknown = [
            [1, mx_i[0], mx_i[1], mx_i[2], mx_i[3], mx_i[4], mx_i[5], mx_i[6],
mx_i[7], mx_i[8], mx_i[9]],
            [mx_i[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_i[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_i[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_i[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_i[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_i[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_i[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_i[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_i[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_i[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_known(1), find_known(2), find_known(3), find_known(4),
find_known(5), find_known(6),
                 find_known(7),  find_known(8), find_known(9), find_known(10)]

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

        while not odnorid:
            print("Матриця планування експеременту:")
            print("      X1           X2           X3          X1X2        X1X3
X2X3         X1X2X3       X1X1"
                  "         X2X2         X3X3          Yi ->")
            for row in range(N):
                print(end=' ')
                for column in range(len(matrix[0])):
                    print("{:^12.3f}".format(matrix[row][column]), end=' ')
                print("")

            dispersion_y = [0.0 for x in range(N)]
            for i in range(N):
                dispersion_i = 0
                for j in range(m):
                    dispersion_i += (matrix_y[i][j] - average_y[i]) ** 2
                dispersion_y.append(dispersion_i / (m - 1))
            f1 = m - 1
            f2 = N
            f3 = f1 * f2
            q = 1 - p
            Gp = max(dispersion_y) / sum(dispersion_y)
            print("Критерій Кохрена:")
            Gt = Perevirku.get_cohren_value(f2, f1, q)
            if Gt > Gp:
                print("Дисперсія однорідна при рівні значимості
{:.2f}.".format(q))
                odnorid = True
            else:
                print("Дисперсія не однорідна при рівні значимості {:.2f}!
```

```
Збільшуємо m.".format(q))
                m += 1

        dispersion_b2 = sum(dispersion_y) / (N * N * m)
        student_lst = list(student_test(beta))
        print("Отримане рівняння регресії з урахуванням критерія Стьюдента")
        print("{:.3f} + {:.3f} * X1 + {:.3f} * X2 + {:.3f} * X3 + {:.3f} * X1X2
+ {:.3f} * X1X3 + {:.3f} * X2X3"
              "+ {:.3f} * X1X2X3 + {:.3f} * X11^2 + {:.3f} * X22^2 + {:.3f} *
X33^2 = ŷ\n\tПеревірка"
              .format(student_lst[0], student_lst[1], student_lst[2],
student_lst[3], student_lst[4], student_lst[5],
                      student_lst[6], student_lst[7], student_lst[8],
student_lst[9], student_lst[10]))
        for i in range(N):
            print("ŷ{} = {:.3f} ≈ {:.3f}".format((i + 1),
check_result(student_lst, i), average_y[i]))

        print("Критерій Фішера")
        d = 11 - student_lst.count(0)
        if fisher_test():
            print("Рівняння регресії адекватне  оригіналу")
            adekvat = True
        else:
            print("Рівняння регресії неадекватне  оригіналу\n\t Проводимо
експеремент повторно")
    return adekvat


if __name__ == '__main__':
    run_experiment()
```

Результати роботи програми:

У протоколі наведено приклад результатів роботи програми від початку(лінійної форми) до останнього етапу(форми з квадратичними членами):

```
C:\Users\fanfl\PycharmProjects\MOPE\venv\Scripts\python.exe C:/Users/fanfl/PycharmProjects/MOPE/Lab6/Lab6.py
Отримане рівняння регресії
1.450 + 5.346 * X1 + 1.387 * X2 + 4.920 * X3 + 7.799 * X1X2 + 0.100 * X1X3 + 9.300 * X2X3+ 7.100 * X1X2X3 + 4.099 * X11^2 + 0.599 * X22^2 + 8.399 * X33^2 =
    Перевірка
ŷ1 = 78274.674 ≈ 78274.600
ŷ2 = -92507.997 ≈ -92508.900
ŷ3 = -95773.333 ≈ -95773.400
ŷ4 = 185133.496 ≈ 185132.600
ŷ5 = 400540.788 ≈ 400541.600
ŷ6 = -568617.883 ≈ -568617.900
ŷ7 = -543010.719 ≈ -543009.900
ŷ8 = 936398.610 ≈ 936398.600
ŷ9 = -2470.680 ≈ -2469.600
ŷ10 = 62483.881 ≈ 62482.912
ŷ11 = -118958.092 ≈ -118958.028
ŷ12 = 168679.698 ≈ 168679.747
ŷ13 = -76467.379 ≈ -76468.281
ŷ14 = 191844.729 ≈ 191845.744
ŷ15 = 22338.601 ≈ 22338.600
Матриця планування експеременту:
      X1          X2          X3         X1X2        X1X3        X2X3        X1X2X3       X1X1        X2X2        X3X3         Yi ->
    10.000      -30.000     -30.000     -300.000    -300.000     900.000     9000.000     100.000     900.000     900.000     78273.600   78275.600
    10.000      -30.000      45.000     -300.000     450.000    -1350.000   -13500.000    100.000     900.000    2025.000    -92509.900  -92507.900
    10.000       45.000     -30.000      450.000    -300.000    -1350.000   -13500.000    100.000    2025.000     900.000    -95774.400  -95772.400
    10.000       45.000      45.000      450.000     450.000     2025.000    20250.000    100.000    2025.000    2025.000    185130.600  185134.600
    60.000      -30.000     -30.000    -1800.000   -1800.000     900.000     54000.000    3600.000    900.000     900.000     400543.600  400539.600
    60.000      -30.000      45.000    -1800.000    2700.000    -1350.000   -81000.000    3600.000    900.000    2025.000    -568614.900 -568620.900
    60.000       45.000     -30.000     2700.000   -1800.000    -1350.000   -81000.000    3600.000    2025.000    900.000    -543009.400 -543010.400
    60.000       45.000      45.000     2700.000    2700.000     2025.000    121500.000    3600.000    2025.000    2025.000    936400.600  936396.600
    -8.250       7.500       7.500      -61.875     -61.875      56.250      -464.062      68.062      56.250      56.250     -2471.100   -2468.100
    78.250       7.500       7.500      586.875     586.875      56.250      4401.562     6123.062     56.250      56.250      62485.912   62479.912
    35.000      -57.375      7.500     -2008.125    262.500     -430.312    -15060.938    1225.000    3291.891     56.250     -118957.028 -118959.028
    35.000       72.375      7.500      2533.125    262.500      542.812     18998.438    1225.000    5238.141     56.250      168678.747  168680.747
    35.000       7.500      -57.375     262.500    -2008.125    -430.312    -15060.938    1225.000     56.250     3291.891    -76469.281  -76467.281
    35.000       7.500       72.375     262.500     2533.125     542.812     18998.438    1225.000     56.250     5238.141    191844.744  191846.744
    35.000       7.500       7.500      262.500     262.500      56.250      1968.750     1225.000     56.250      56.250      22336.100   22341.100
Критерій Кохрена:
Дисперсія однорідна при рівні значимості 0.05.
Отримане рівняння регресії з урахуванням критерія Стьюдента
1.450 + 5.346 * X1 + 1.387 * X2 + 4.920 * X3 + 7.799 * X1X2 + 0.100 * X1X3 + 9.300 * X2X3+ 7.100 * X1X2X3 + 4.099 * X11^2 + 0.599 * X22^2 + 8.399 * X33^2 = ŷ
    Перевірка
ŷ1 = 78274.674 ≈ 78274.600
ŷ2 = -92507.997 ≈ -92508.900
ŷ3 = -95773.333 ≈ -95773.400
ŷ4 = 185133.496 ≈ 185132.600
ŷ5 = 400540.788 ≈ 400541.600
ŷ6 = -568617.883 ≈ -568617.900
ŷ7 = -543010.719 ≈ -543009.900
ŷ8 = 936398.610 ≈ 936398.600
ŷ9 = -2470.680 ≈ -2469.600
ŷ10 = 62483.881 ≈ 62482.912
ŷ11 = -118958.092 ≈ -118958.028
ŷ12 = 168679.698 ≈ 168679.747
ŷ13 = -76467.379 ≈ -76468.281
ŷ14 = 191844.729 ≈ 191845.744
ŷ15 = 22338.601 ≈ 22338.600
Критерій Фішера
Рівняння регресії адекватне  оригіналу

Process finished with exit code 0
```

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. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера) для кожної форми рівняння регресії . При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів, при неадекватності і такого рівняння регресії було затосовано рівняння регресії з квадратичними членами. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості q = 0.05.