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 №6

З дисципліни «Методи оптимізації та планування»

Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІО-93
Євтушок О.М. - 9311

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

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

Варіант завдання:

Варіант	X_1		X_2		X_3	
	min	max	min	max	min	max
310	-25	-5	10	60	-5	60

f(x1, x2, x3)= 0.2 + 6.7 * X1 + 6.3 * X2 + 7.3 * X3 + 5.7 * X1 * X1 + 1.0 * X2 * X2 + 3.4 * X3 * X3 + 8.3 * X1 * X2 + 0.1 * X1 * X3 + 6.7 * X2 * X3 + 8.8 * X1 * X2 * X3

Лістинг програми:

```
from math import fabs, sqrt
from time import time

m = 2
p = 0.95
N = 15
x1_min = -25
x1_max = -5
x2_min = 10
x2_max = 60
x3_min = -5
x3_max = 60
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 = 0.2 + 6.7 * X1 + 6.3 * X2 + 7.3 * X3 + 5.7 * X1 * X1 + 1.0 * X2 * X2 +
3.4 * X3 * X3 + 8.3 * X1 * X2 + \
            0.1 * X1 * X3 + 6.7 * X2 * X3 + 8.8 * 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()
```

Скріншоти роботи:

```
Отримане рівняння регресії
2.305 + 6.959 * X1 + 6.313 * X2 + 7.284 * X3 + 8.298 * X1X2 + 0.098 * X1X3 + 6.699 * X2X3+ 8.800 * X1X2X3 + 5.705 * X11^2 + 0.999 * X22^2 + 3.400 * X33^2 = ŷ
    Перевірка
ŷ1 = 12208.015 ≈ 12208.700
ŷ2 = -113969.391 ≈ -113968.800
ŷ3 = 58973.596 ≈ 58975.200
ŷ4 = -760429.310 ≈ -760427.800
ŷ5 = 1774.338 ≈ 1775.200
ŷ6 = -9875.568 ≈ -9874.800
ŷ7 = 12838.419 ≈ 12840.200
ŷ8 = -120038.488 ≈ -120036.800
ŷ9 = -266655.302 ≈ -266656.682
ŷ10 = 30868.278 ≈ 30866.488
ŷ11 = 33382.622 ≈ 33382.100
ŷ12 = -268845.427 ≈ -268848.075
ŷ13 = 126882.203 ≈ 126880.510
ŷ14 = -344589.128 ≈ -344590.605
ŷ15 = -119600.823 ≈ -119600.800
```

```
Матриця планування експеременту:
      X1          X2          X3         X1X2         X1X3         X2X3        X1X2X3         X1X1         X2X2         X3X3          Yi ->
  -25.000      10.000      -5.000     -250.000      125.000      -50.000     1250.000      625.000      100.000       25.000     12204.200    12213.200
  -25.000      10.000      60.000     -250.000    -1500.000      600.000   -15000.000      625.000      100.000     3600.000   -113972.800  -113964.800
  -25.000      60.000      -5.000    -1500.000      125.000     -300.000     7500.000      625.000     3600.000       25.000     58977.200    58973.200
  -25.000      60.000      60.000    -1500.000    -1500.000     3600.000   -90000.000      625.000     3600.000     3600.000   -760426.800  -760428.800
   -5.000      10.000      -5.000      -50.000       25.000      -50.000      250.000       25.000      100.000       25.000      1776.200     1774.200
   -5.000      10.000      60.000      -50.000     -300.000      600.000    -3000.000       25.000      100.000     3600.000     -9877.800    -9871.800
   -5.000      60.000      -5.000     -300.000       25.000     -300.000     1500.000       25.000     3600.000       25.000     12842.200    12838.200
   -5.000      60.000      60.000     -300.000     -300.000     3600.000   -18000.000       25.000     3600.000     3600.000   -120038.800  -120034.800
  -32.300      35.000      27.500    -1130.500     -888.250      962.500   -31088.750     1043.290     1225.000      756.250   -266653.182  -266660.182
    2.300      35.000      27.500       80.500       63.250      962.500     2213.750        5.290     1225.000      756.250     30863.488    30869.488
  -15.000      -8.250      27.500      123.750     -412.500     -226.875     3403.125      225.000       68.062      756.250     33382.600    33381.600
  -15.000      78.250      27.500    -1173.750     -412.500     2151.875   -32278.125      225.000     6123.062      756.250   -268848.075  -268848.075
  -15.000      35.000     -28.725     -525.000      430.875    -1005.375    15080.625      225.000     1225.000      825.126     126879.510   126881.510
  -15.000      35.000      83.725     -525.000    -1255.875     2930.375   -43955.625      225.000     1225.000     7009.876   -344591.605  -344589.605
  -15.000      35.000      27.500     -525.000     -412.500      962.500   -14437.500      225.000     1225.000      756.250   -119600.300  -119601.300
```

```
Критерій Кохрена:
Дисперсія однорідна при рівні значимості 0.05.
Отримане рівняння регресії з урахуванням критерія Стьюдента
2.305 + 6.959 * X1 + 6.313 * X2 + 7.284 * X3 + 8.298 * X1X2 + 0.098 * X1X3 + 6.699 * X2X3+ 8.800 * X1X2X3 + 5.705 * X11^2 + 0.999 * X22^2 + 3.400 * X33^2 = ŷ
    Перевірка
ŷ1 = 12208.015 ≈ 12208.700
ŷ2 = -113969.391 ≈ -113968.800
ŷ3 = 58973.596 ≈ 58975.200
ŷ4 = -760429.310 ≈ -760427.800
ŷ5 = 1774.338 ≈ 1775.200
ŷ6 = -9875.568 ≈ -9874.800
ŷ7 = 12838.419 ≈ 12840.200
ŷ8 = -120038.488 ≈ -120036.800
ŷ9 = -266655.302 ≈ -266656.682
ŷ10 = 30868.278 ≈ 30866.488
ŷ11 = 33382.622 ≈ 33382.100
ŷ12 = -268845.427 ≈ -268848.075
ŷ13 = 126882.203 ≈ 126880.510
ŷ14 = -344589.128 ≈ -344590.605
ŷ15 = -119600.823 ≈ -119600.800
Критерій Фішера
Рівняння регресії адекватне  оригіналу
```

Висновок:

В даній лабораторній роботі я провів трьохфакторний експеримент і отримав адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.