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 № 4

з дисципліни **«Методи оптимізації та планування експерименту »** на тему

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії»

Виконала:
студентка II курсу ФІОТ
групи ІО-93
Дяченко Віта
У списку групи №9

Перевірив:
Регіда П. Г.

Київ – 2021

Мета: Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

Завдання:

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.
3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

Варіант:

Варіант:	min	max	min	max	min	max
309	-30	0	-15	35	-30	35

Код програми:

```
import random as r
import numpy as np
import math
from _pydecimal import Decimal
from scipy.stats import f, t
from functools import reduce
from itertools import compress
```

```
#                        1   2   3    12  13  23     123
norm_factors_table = [[-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]]
```

```
zero_factor = [+1]*8

factors_table = [[-30, -15, -30,      450, 900, 450,          -13500],
                 [-30,  35,  35,     -1050, -1050, 1225,      36750],
                 [0,   -15, -30,      0, 0, 450,              0],
                 [0,    35,  35,      0, 0, 1225,             0],

                 [30, -15, -30,       450, 900, 450,          -13500],
                 [-30,  35, 35,      -1050, -1050, 1225,      36750],
                 [0,  -15, -30,       0, 0, 450,              0],
                 [0,   35, 35,        0, 0, 1225,             0]]

y_min = 175
y_max = 223

M = 5
N = 8
y_arr = [[r.randint(y_min, y_max) for _ in range(M)] for j in range(N)]

x1 = np.array(list(zip(*factors_table))[0])
x2 = np.array(list(zip(*factors_table))[1])
x3 = np.array(list(zip(*factors_table))[2])
yi = np.array([np.average(i) for i in y_arr])

def m_ij(*arrays):
    return np.average(reduce(lambda accum, el: accum*el, arrays))

coeffs = [[N,         m_ij(x1),     m_ij(x2),     m_ij(x3),     m_ij(x1*x2),
m_ij(x1*x3),     m_ij(x2*x3),     m_ij(x1*x2*x3)],

          [m_ij(x1), m_ij(x1**2), m_ij(x1*x2), m_ij(x1*x3), m_ij(x1**2*x2),
m_ij(x1**2*x3), m_ij(x1*x2*x3), m_ij(x1**2*x2*x3)],
          [m_ij(x2), m_ij(x1*x2), m_ij(x2**2), m_ij(x2*x3), m_ij(x1*x2**2),
m_ij(x1*x2*x3), m_ij(x2**2*x3), m_ij(x1*x2**2*x3)],
          [m_ij(x3), m_ij(x1*x3), m_ij(x2*x3), m_ij(x3**2), m_ij(x1*x2*x3),
m_ij(x1*x3**2), m_ij(x2*x3**2), m_ij(x1*x2*x3**2)],

          [m_ij(x1*x2), m_ij(x1**2*x2), m_ij(x1*x2**2), m_ij(x1*x2*x3),
m_ij(x1**2*x2**2), m_ij(x1**2*x2*x3), m_ij(x1*x2**2*x3),
m_ij(x1**2*x2**2*x3)],
          [m_ij(x1*x3), m_ij(x1**2*x3), m_ij(x1*x2*x3), m_ij(x1*x3**2),
m_ij(x1**2*x2*x3), m_ij(x1**2*x3**2), m_ij(x1*x2*x3**2),
m_ij(x1**2*x2*x3**2)],
          [m_ij(x2*x3), m_ij(x1*x2*x3), m_ij(x2**2*x3), m_ij(x2*x3**2),
m_ij(x1*x2**2*x3), m_ij(x1*x2*x3**2), m_ij(x2**2*x3**2),
m_ij(x1*x2**2*x3**2)],

          [m_ij(x1*x2*x3), m_ij(x1**2*x2*x3), m_ij(x1*x2**2*x3),
m_ij(x1*x2*x3**2), m_ij(x1**2*x2**2*x3), m_ij(x1**2*x2*x3**2),
m_ij(x1*x2**2*x3**2), m_ij(x1**2*x2**2*x3**2)]]

free_vals = [m_ij(yi), m_ij(yi*x1), m_ij(yi*x2), m_ij(yi*x3), m_ij(yi*x1*x2),
m_ij(yi*x1*x3), m_ij(yi*x2*x3), m_ij(yi*x1*x2*x3)]
```

```
natural_bi = np.linalg.solve(coeffs, free_vals)

natural_x1 =  np.array(list(zip(*norm_factors_table))[0])
natural_x2 =  np.array(list(zip(*norm_factors_table))[1])
natural_x3 =  np.array(list(zip(*norm_factors_table))[2])

norm_bi = [m_ij(yi),
           m_ij(yi*natural_x1),
           m_ij(yi*natural_x2),
           m_ij(yi*natural_x3),
           m_ij(yi*natural_x1*natural_x2),
           m_ij(yi*natural_x1*natural_x3),
           m_ij(yi*natural_x2*natural_x3),
           m_ij(yi*natural_x1*natural_x2*natural_x3)]

def cochran_criteria(m, N, y_table):
    print("Перевірка рівномірності дисперсій за критерієм Кохрена: m = {}, N =
{} для таблиці".format(m, N))
    y_variations = [np.var(i) for i in y_table]
    max_y_variation = max(y_variations)
    gp = max_y_variation/sum(y_variations)
    f1 = m - 1
    f2 = N
    p = 0.95
    q = 1-p
    gt = get_cochran_value(f1,f2, q)
    print("Gp = {}; Gt = {}; f1 = {}; f2 = {}; q = {:.2f}".format(gp, gt, f1,
f2, q))
    if gp < gt:
        print("Gp < Gt => дисперсії рівномірні")
        return True
    else:
        print("Gp > Gt => дисперсії нерівномірні - повторіть експеримент")
        return False

def student_criteria(m, N, y_table, normalized_x_table: "with zero factor!"):
    print("\nПеревірка значимості коефіцієнтів регресії за критерієм
Стьюдента: m = {}, N = {} "
          "для таблиці та нормалізованих факторів".format(m, N))
    average_variation = np.average(list(map(np.var, y_table)))

    y_averages = np.array(list(map(np.average, y_table)))
    variation_beta_s = average_variation/N/m
    standard_deviation_beta_s = math.sqrt(variation_beta_s)
    x_i = np.array([[el[i] for el in normalized_x_table] for i in
range(len(normalized_x_table))])
    coefficients_beta_s = np.array([round(np.average(y_averages*x_i[i]),3) for
i in range(len(x_i))])
    print("Оцінки коефіцієнтів βs: " + ",
".join(list(map(str,coefficients_beta_s))))
    t_i = np.array([abs(coefficients_beta_s[i])/standard_deviation_beta_s for
i in range(len(coefficients_beta_s))])
```

```
    print("Коефіцієнти ts:          " + ", ".join(list(map(lambda i:
"{:.2f}".format(i), t_i))))
    f3 = (m-1)*N
    q = 0.05

    t = get_student_value(f3, q)
    importance = [True if el > t else False for el in list(t_i)]

    # print result data
    print("f3 = {}; q = {}; tтабл = {}".format(f3, q, t))
    beta_i = ["β0", "β1", "β2", "β3", "β12", "β13", "β23", "β123"]
    importance_to_print = ["важливий" if i else "неважливий" for i in
importance]
    to_print = map(lambda x: x[0] + " " + x[1], zip(beta_i,
importance_to_print))
    x_i_names = list(compress(["", "x1", "x2", "x3", "x12", "x13", "x23",
"x123"], importance))#'[""] + list(compress(["x{}".format(i) for i in
range(N)], importance))[1:]
    betas_to_print = list(compress(coefficients_beta_s, importance))
    print(*to_print, sep = "; ")
    equation = " ".join(["".join(i) for i in zip(list(map(lambda x:
"{:+.2f}".format(x), betas_to_print)),x_i_names)])
    print("Рівняння регресії без незначимих членів: y = " + equation)
    return importance


def calculate_theoretical_y(x_table, b_coefficients, importance):
    x_table = [list(compress(row, importance)) for row in x_table]
    b_coefficients = list(compress(b_coefficients, importance))
    y_vals = np.array([sum(map(lambda x, b: x*b, row, b_coefficients)) for row
in x_table])
    return y_vals


def fisher_criteria(m, N, d, naturalized_x_table, y_table, b_coefficients,
importance):
    print("\nПеревірка адекватності моделі за критерієм Фішера: m = {}, "
          "N = {} для таблиці".format(m, N))
    f3 = (m - 1) * N
    f4 = N - d
    q = 0.05

    theoretical_y = calculate_theoretical_y(naturalized_x_table,
b_coefficients, importance)
    theoretical_values_to_print = list(zip(map(lambda x: "x1 = {0[1]}, x2 =
{0[2]}, x3 = {0[3]}".format(x),naturalized_x_table),theoretical_y))
    print("Теоретичні значення y для різних комбінацій факторів:")
    print("\n".join(["{arr[0]}: y = {arr[1]}".format(arr = el) for el in
theoretical_values_to_print]))
    y_averages = np.array(list(map(np.average, y_table)))
    s_ad = m/(N-d)*(sum((theoretical_y-y_averages)**2))
    y_variations = np.array(list(map(np.var, y_table)))
    s_v = np.average(y_variations)
```

```
    f_p = float(s_ad/s_v)
    f_t = get_fisher_value(f3, f4, q)
    print("Fp = {}, Ft = {}".format(f_p, f_t))
    print("Fp < Ft => модель адекватна" if f_p < f_t else "Fp > Ft => модель
неадекватна")
    return True if f_p < f_t else False


def m_ij(*arrays):
    return np.average(reduce(lambda accum, el: accum*el, arrays))


def get_cochran_value(f1, f2, q):
    partResult1 = q / f2 # (f2 - 1)
    params = [partResult1, f1, (f2 - 1) * f1]
    fisher = f.isf(*params)
    result = fisher/(fisher + (f2 - 1))
    return Decimal(result).quantize(Decimal('.0001')).__float__()


def get_student_value(f3, q):
    return Decimal(abs(t.ppf(q/2,f3))).quantize(Decimal('.0001')).__float__()


def get_fisher_value(f3,f4, q):
    return Decimal(abs(f.isf(q,f4,f3))).quantize(Decimal('.0001')).__float__()


while not cochran_criteria(M, 4, y_arr):
    M += 1
    y_table = [[r.randint(y_min, y_max) for _ in range(M)] for j in range(N)]
print("Матриця планування:")
labels_table = list(map(lambda x: x.ljust(6), ["x1", "x2", "x3", "x12",
"x13", "x23", "x123"] + ["y{}".format(i+1) for i in range(M)]))
rows_table = [list(factors_table[i]) + list(y_arr[i]) for i in range(N)]
rows_normalized_table = [factors_table[i] + list(y_arr[i]) for i in range(N)]
print((" ").join(labels_table))
print("\n".join([" ".join(map(lambda j: "{:<+6}".format(j), rows_table[i]))
for i in range(len(rows_table))]))
print("\t")

norm_factors_table_zero_factor = [[+1]+i for i in norm_factors_table]
importance = student_criteria(M, N, y_arr , norm_factors_table_zero_factor)

fisher_criteria(M, N, 1, factors_table, y_arr, natural_bi, importance)
```

Результат роботи:

```
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 5, N = 4 для таблиці
Gp = 0.1578506161455277; Gt = 0.6287; f1 = 4; f2 = 4; q = 0.05
Gp < Gt => дисперсії рівномірні
Матриця планування:
x1    x2    x3    x12   x13   x23   x123   y1    y2    y3    y4    y5
-30   -15   -30   +450  +900  +450  -13500 +179  +193  +213  +192  +180
-30   +35   +35   -1050 -1050 +1225 +36750 +215  +219  +212  +189  +205
+0    -15   -30   +0    +0    +450  +0     +184  +189  +190  +207  +184
+0    +35   +35   +0    +0    +1225 +0     +188  +200  +197  +178  +208
+30   -15   -30   +450  +900  +450  -13500 +177  +201  +205  +200  +194
-30   +35   +35   -1050 -1050 +1225 +36750 +200  +209  +184  +217  +214
+0    -15   -30   +0    +0    +450  +0     +188  +210  +217  +186  +198
+0    +35   +35   +0    +0    +1225 +0     +192  +195  +223  +203  +212


Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 5, N = 8 для таблиці та нормалізованих факторів
Оцінки коефіцієнтів βs: 198.675, -1.225, 4.325, 1.125, -2.175, -0.675, 2.375, 2.575
Коефіцієнти ts:         115.05, 0.71, 2.50, 0.65, 1.26, 0.39, 1.38, 1.49
f3 = 32; q = 0.05; tтабл = 2.0369
β0 важливий; β1 неважливий; β2 важливий; β3 неважливий; β12 неважливий; β13 неважливий; β23 неважливий; β123 неважливий
Рівняння регресії без незначимих членів: y = +198.68 +4.33x2
```

```
Перевірка адекватності моделі за критерієм Фішера: m = 5, N = 8 для таблиці
Теоретичні значення y для різних комбінацій факторів:
x1 = -15, x2 = -30, x3 = 450: y = -193.20005265071308
x1 = 35, x2 = 35, x3 = -1050: y = 225.40006142583204
x1 = -15, x2 = -30, x3 = 0: y = -193.20005265071313
x1 = 35, x2 = 35, x3 = 0: y = 225.40006142583198
x1 = -15, x2 = -30, x3 = 450: y = -193.2000526507132
x1 = 35, x2 = 35, x3 = -1050: y = 225.40006142583204
x1 = -15, x2 = -30, x3 = 0: y = -193.20005265071313
x1 = 35, x2 = 35, x3 = 0: y = 225.40006142583198
Fp = 3610.3434262589753, Ft = 2.3127
Fp > Ft => модель неадекватна
```

<u>Висновок:</u> В даній роботі було проведено повний трьохфакторний експеримент при використанні лінійного рівняння регресії та рівняння регресії з ефектом взаємодії, було визначено коефіцієнти рівняння регресії, проведено три статистичні перевірки.