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»

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

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІО-92
Кушенко Сергій

ПЕРЕВІРИВ:
Регіда П.Г.

Київ 2021 р.

Завдання на лабораторну роботу

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.

$$y_{i\max} = 200 + x_{cp\max}$$
$$y_{i\min} = 200 + x_{cp\min}$$

де $x_{cp\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}$, $x_{cp\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$

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:

№213

X_1		X_2		X_3	
min	**max**	**min**	**max**	**min**	**max**
-40	**20**	**5**	**40**	**-40**	**-20**

Код програми:

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

#                      x1  x2  x3    x12 x13 x23    x123
norm_factor_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]]
```

```
var_factor_table = [[-40, 5, -40,     -200, 1600, -200,    8000],
                    [-40, 40, -20,    -1600, 800, -800,   32000],
                    [20, 5, -20,       100, -400, -100,    -2000],
                    [20, 40, -40,       800, -800, -1600,  -32000],

                    [-40, 5, -20,     -200, 800, -100,   4000],
                    [-40,  40, -40,     -1600, 1600, -1600,   64000],
                    [20,  5, -40,     100, -800, -200,        -4000],
                    [20,  40, -20,     800, -400, -800,       -16000]]

y_max = 213
y_min = 175
M = 3
N = 8
y_arr = [[randint(y_min, y_max) for _ in range(M)] for j in range(N)]

x1 = np.array(list(zip(*var_factor_table))[0])
x2 = np.array(list(zip(*var_factor_table))[1])
x3 = np.array(list(zip(*var_factor_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_factor_table))[0])
natural_x2 = np.array(list(zip(*norm_factor_table))[1])
natural_x3 = np.array(list(zip(*norm_factor_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)]
```

```python
def cochran_cr(m, N, y_table):
    print("Перевірка рівномірності дисперсій за критерієм Кохрена: ")
    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 = Cochran_val(f1,f2, q)
    print("Gp = {:.3f}; Gt = {:.3f}".format(gp, gt))
    if gp < gt:
        print("Gp < Gt => дисперсії рівномірні")
        return True
    else:
        print("Gp > Gt => дисперсії нерівномірні - Потрібно додати експерименти")
        return False

def student_criteria(m, N, y_table, normalized_x_table: "with zero factor!"):
    print("\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента: ")
    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 = Student_val(f3, q)
    importance = [True if el > t else False for el in list(t_i)]

    print("Табличне значення критерія Стьюдента: {}".format(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))
    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:")
    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]:.2f}".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 = Fisher_val(f3, f4, q)
    print("\nFp = {:.3f}, Ft = {:.3f}".format(f_p, f_t))
    print("Fp < Ft => МАТЕМАТИЧНА МОДЕЛЬ АДЕКВАТНА" if f_p < f_t else "Fp > Ft =>
МАТЕМАТИЧНА МОДЕЛЬ НЕАДЕКВАТНА")
    return True if f_p < f_t else False

print("m = {}, n = {}".format(M,N))
print("Матриця планування для повного трьохфакторного експерименту:")
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(var_factor_table[i]) + list(y_arr[i]) for i in range(N)]
rows_normalized_table = [var_factor_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")

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

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

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

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


while not cochran_cr(M, 4, y_arr):
    M += 1
    y_table = [[randint(y_min, y_max) for _ in range(M)] for j in range(N)]

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

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

Результат виконання:

```
m = 3, n = 8
Матриця планування для повного трьохфакторного експерименту:
x1     x2     x3     x12    x13    x23    x123   y1     y2     y3
-40    +5     -40    -200   +1600  -200   +8000  +198   +186   +194
-40    +40    -20    -1600  +800   -800   +32000 +203   +181   +205
+20    +5     -20    +100   -400   -100   -2000  +188   +189   +191
+20    +40    -40    +800   -800   -1600  -32000 +200   +198   +212
-40    +5     -20    -200   +800   -100   +4000  +185   +202   +213
-40    +40    -40    -1600  +1600  -1600  +64000 +185   +179   +209
+20    +5     -40    +100   -800   -200   -4000  +206   +205   +190
+20    +40    -20    +800   -400   -800   -16000 +211   +176   +193

Перевірка рівномірності дисперсій за критерієм Кохрена:
Gp = 0.275; Gt = 0.768
Gp < Gt => дисперсії рівномірні

Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента:
Оцінки коефіцієнтів βs: 195.792, 0.792, 0.208, -1.042, 1.542, -4.208, -0.125, 0.375
Коефіцієнти ts: 99.64, 0.40, 0.11, 0.53, 0.78, 2.14, 0.06, 0.19
Табличне значення критерія Стьюдента: 2.1199
β0 важливий; β1 неважливий; β2 неважливий; β3 неважливий; β12 неважливий; β13 важливий; β23 неважливий; β123 неважливий
Рівняння регресії без незначимих членів: y = +195.79 -4.21X13
```

```
Перевірка адекватності моделі за критерієм Фішера:
Теоретичні значення y для різних комбінацій факторів:
x1 = 5, x2 = -40, x3 = -200: y = -31.71
x1 = 40, x2 = -20, x3 = -1600: y = -11.04
x1 = 5, x2 = -20, x3 = 100: y = 22.74
x1 = 40, x2 = -40, x3 = 800: y = 74.40
x1 = 5, x2 = -20, x3 = -200: y = -35.15
x1 = 40, x2 = -40, x3 = -1600: y = 16.51
x1 = 5, x2 = -40, x3 = 100: y = 26.19
x1 = 40, x2 = -20, x3 = 800: y = 46.85

Fp = 1273.006, Ft = 2.657
Fp > Ft => МАТЕМАТИЧНА МОДЕЛЬ НЕАДЕКВАТНА

Process finished with exit code 0
```

Висновок:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії та рівняння регресії з ефектом взаємодії, складено матрицю планування експерименту, було визначено коефіцієнти рівняння регресії, виконано перевірку правильності розрахунку коефіцієнтів рівняння регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера). При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів.