

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**

ВИКОНАЛА:  
Студентка II курсу ФІОТ  
Групи ІВ-91  
Іванькова  
А.Р.

ПЕРЕВІРИВ:  
асистент  
Регіда П.Г.

Київ 2021 р.

**Мета:**

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

**Варіант завдання:**

111	-25	-5	10	60	-5	60
-----	-----	----	----	----	----	----

**Лістинг програми:**

```
from prettytable import PrettyTable
from itertools import compress
from scipy.stats import f, t
from functools import reduce
from random import randint
import numpy as np
import math

class Lab4:

    def __init__(self, n, m):

        self.n = n
        self.m = m

        self.f1 = self.m - 1
        self.f2 = self.n
        self.f3 = self.f1*self.f2

        self.p = 0.95
        self.q = 1 - self.p

        self.N = [i+1 for i in range(self.n+1)]

        self.x_min = [-25, 10, -5]
        self.x_max = [5, 60, 60]
        self.average_x_min = round(np.average(self.x_min))
        self.average_x_max = round(np.average(self.x_max))
        self.y_min = 200 + self.average_x_min
        self.y_max = 200 + self.average_x_max

        self.norm_x = [[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]]

        self.x12 = [self.norm_x[1][i]*self.norm_x[2][i] for i in range(self.n)]
        self.x13 = [self.norm_x[1][i]*self.norm_x[3][i] for i in range(self.n)]
        self.x23 = [self.norm_x[2][i]*self.norm_x[3][i] for i in range(self.n)]
        self.x123 = [self.norm_x[1][i]*self.norm_x[2][i]*self.norm_x[3][i] for i in
range(self.n)]

        self.norm_x += [self.x12, self.x13, self.x23, self.x123]
        self.norm_xt = np.array(self.norm_x)
        self.norm_xt = self.norm_xt.transpose()

        self.factors_x = [[-25,-25,5,5,-25,-25,5,5],
                        [10,60,10,60,10,60,10,60],
                        [-5,60,60,-5,60,-5,-5,60]]
```

```

        self.xf12 = [self.factors_x[0][i]*self.factors_x[1][i] for i in
range(self.n)]
        self.xf13 = [self.factors_x[0][i]*self.factors_x[2][i] for i in
range(self.n)]
        self.xf23 = [self.factors_x[1][i]*self.factors_x[2][i] for i in
range(self.n)]
        self.xf123 = [self.factors_x[0][i]*self.factors_x[1][i]*self.factors_x[2][i]
for i in range(self.n)]

        self.factors_x += [self.xf12, self.xf13, self.xf23, self.xf123]
        self.factors_xt = np.array(self.factors_x)
        self.factors_xt = self.factors_xt.transpose()

        # Матриця відгуків:
        self.y_t = np.array([[randint((self.y_min), (self.y_max)) for i in
range(self.n)] for j in range(self.m)])
        self.y = self.y_t.transpose()
        # Середнє значення функції відгуку в рядку:
        self.av_y = [round(sum(i) / len(i), 2) for i in self.y]
        # Дисперсії по рядках:
        self.S2 = [round(np.var(i),2) for i in self.y]
        self.x1 = np.array(list(zip(*self.factors_xt))[0])
        self.x2 = np.array(list(zip(*self.factors_xt))[1])
        self.x3 = np.array(list(zip(*self.factors_xt))[2])

        self.natural_bi = self.Natural(self.n, self.x1, self.x2, self.x3, self.av_y)

        print("ŷ = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13*x1*x3 + b23*x2*x3 +
b123*x1*x2*x3")

        # Нормована матриця планування експерименту
        th = ["N", "x0", "x1", "x2", "x3", "x1*x2", "x1*x3", "x2*x3", "x1*x2*x3"]
        th += ["y"+str(i+1) for i in range(self.m)]
        th.append("<y>")
        th.append("S^2")
        columns = len(th)
        table = PrettyTable(th)
        table.title = "Нормована матриця планування експерименту"
        for i in range(self.n):
            td = [self.N[i], self.norm_x[0][i], self.norm_x[1][i], self.norm_x[2][i],
self.norm_x[3][i], \
                self.x12[i], self.x13[i], self.x23[i], self.x123[i]]
            td += [self.y_t[j][i] for j in range(self.m)]
            td.append(self.av_y[i])
            td.append(self.S2[i])
            td_data = td[:]
            while td_data:
                table.add_row(td_data[:columns])
                td_data = td_data[columns:]
        print(table)

        th = ["N", "x1", "x2", "x3", "x1*x2", "x1*x3", "x2*x3", "x1*x2*x3"]
        th += ["y"+str(i+1) for i in range(self.m)]
        th.append("<y>")
        th.append("S^2")
        columns = len(th)
        table = PrettyTable(th)
        table.title = "Матриця планування експерименту"

```

```

        for i in range(self.n):
            td = [self.N[i], self.factors_x[0][i], self.factors_x[1][i],
self.factors_x[2][i], \
                self.xf12[i], self.xf13[i], self.xf23[i], self.xf123[i]]
            td += [self.y_t[j][i] for j in range(self.m)]
            td.append(self.av_y[i])
            td.append(self.S2[i])
            td_data = td[:]
            while td_data:
                table.add_row(td_data[:columns])
                td_data = td_data[columns:]
        print(table)

        self.Kohren(self.m, self.n, self.y, self.p, self.q, self.f1, self.f2)

        self.Student(self.m, self.n, self.y, self.av_y, self.norm_xt, self.f3,
self.q)

        self.Fisher(self.m, self.n, 1, self.f3, self.q, self.factors_xt, self.y,
self.natural_bi, self.y_x)

    def Natural(self, n, x1, x2, x3, av_y):

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

        koef = [[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_vector = [m_ij(av_y), m_ij(av_y*x1), m_ij(av_y*x2), m_ij(av_y*x3),
m_ij(av_y*x1*x2), m_ij(av_y*x1*x3), m_ij(av_y*x2*x3), m_ij(av_y*x1*x2*x3)]
        natural_bi = np.linalg.solve(koef, free_vector)
        return natural_bi

# Перевірка однорідності дисперсії за критерієм Кохрена:

def Kohren(self, m, n, y, p, q, f1, f2):

    # Знайдемо дисперсії по рядках:
    S = [np.var(i) for i in y]
    # Знайдемо критерій Кохрена:
    Gp = max(S)/sum(S)
    # Табличне значення критерію Кохрена:
    q_ = q / f2
    khr = f.ppf(q=1-q_, dfn=f1, dfd=(f2 - 1) * f1)
    Gt = khr / (khr + f2 - 1)

```

```

print("Критерій Кохрена: Gr = " + str(round(Gp,3)))
# Рівень значимості прийmemo 0.05.
# Перевірка рівняння на однорідність:
if Gp < Gt:
    print("Дисперсії однорідні з вірогідністю 95%.")
    pass
else:
    print("\nДисперсії не однорідні.\nПроводимо експеримент для m+=1\n")
    Lab4(self.n, self.m+1)

def Student(self, m, n, y, av_y, norm_xt, f3, q):

    av_S = np.average(list(map(np.var, y)))
    s2_beta = av_S/n/m
    s_beta = math.sqrt(s2_beta)
    xi = np.array([el[i] for el in norm_xt] for i in range(len(norm_xt))])
    k_beta = np.array([round(np.average(av_y*xi[i]),3) for i in range(len(xi))])

    T = np.array([abs(k_beta[i])/s_beta for i in range(len(k_beta))])

    T_tabl = t.ppf(q=1-q, df=f3)

    print("\nКритерій Стюдента:")
    T_ = list(map(lambda i: "{:.2f}".format(i), T))
    for i in T_: print(str(i))
    imp = [i if i > T_tabl else 0 for i in T]
    b = ["b0", "b1", "b2", "b3", "b4", "b12", "b13", "b23", "b123"]
    index_list = [i for i, x in enumerate(imp) if x == 0]
    index_list = [b[i] for i in index_list]
    deleted_koef = ', '.join(index_list) + " - коефіцієнти рівняння регресії
приймаємо незначними, виключаємо їх з рівняння. "
    print(deleted_koef)

    self.y_x = [True if i > T_tabl else False for i in T]
    x_i = list(compress(["", "*x1", "*x2", "*x3", "*x12", "*x13", "*x23",
"*x123"], self.y_x))
    p = list(compress(k_beta, self.y_x))
    y = " ".join(["".join(i) for i in zip(list(map(lambda x: "{:.2f}".format(x),
p)),x_i)])
    print("Рівняння регресії: y = " + y)

# -----
# Перевірка однорідності дисперсії за критерієм Фішера:
# -----

def Fisher(self, m, n, d, f3, q, natural_x, y_t, b_k, imp):

    f4 = n - d
    b_k = list(compress(b_k, imp))
    y_vals = np.array([sum(map(lambda x, b: x*b, row, b_k)) for row in
natural_x])
    y_averages = np.array(list(map(np.average, y_t)))
    s_ad = m/(n-d)*(sum((y_vals-y_averages)**2))*0.001
    y_variations = np.array(list(map(np.var, y_t)))
    s_v = np.average(y_variations)

    Fp = s_ad/s_v
    print("\nКритерій Фішера: Fp = " + str(round(Fp,4)))

    F_tabl = f.isf(q,f4,f3)

```

```

print("Табличне значення критерія Фішера: Ft = "+ str(round(F_tabl,4)))

if Fp < F_tabl:
    print("\nPівняння регресії адекватно оригіналу.")
    pass
else:
    print("\nPівняння регресії НЕ адекватно оригіналу. >>> m+=1\n")
    Lab4(self.n, self.m+1)

Lab4(8, 3)

```

## Результат виконання роботи:

F:\anakkonda\envs\And\python.exe C:/Users/User/PycharmProjects/And/MND/lb4.py  
 $\hat{y} = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + b_3 \cdot x_3 + b_{12} \cdot x_1 \cdot x_2 + b_{13} \cdot x_1 \cdot x_3 + b_{23} \cdot x_2 \cdot x_3 + b_{123} \cdot x_1 \cdot x_2 \cdot x_3$

Нормована матриця планування експерименту													
N	x0	x1	x2	x3	x1*x2	x1*x3	x2*x3	x1*x2*x3	y1	y2	y3	<y>	S^2
1	1	-1	-1	-1	1	1	1	-1	236	216	213	221.67	104.22
2	1	-1	1	1	-1	-1	1	-1	223	218	205	215.33	57.56
3	1	1	-1	1	-1	1	-1	-1	228	234	212	224.67	86.22
4	1	1	1	-1	1	-1	-1	-1	236	229	228	231.0	12.67
5	1	-1	-1	1	1	-1	1	1	230	222	203	218.33	128.22
6	1	-1	1	-1	-1	1	-1	1	194	231	202	209.0	252.67
7	1	1	-1	-1	-1	-1	1	1	214	226	207	215.67	61.56
8	1	1	1	1	1	1	1	1	218	207	230	218.33	88.22

Матриця планування експерименту													
N	x1	x2	x3	x1*x2	x1*x3	x2*x3	x1*x2*x3	y1	y2	y3	<y>	S^2	
1	-25	10	-5	-250	125	-50	1250	236	216	213	221.67	104.22	
2	-25	60	60	-1500	-1500	3600	-90000	223	218	205	215.33	57.56	
3	5	10	60	50	300	600	3000	228	234	212	224.67	86.22	
4	5	60	-5	300	-25	-300	-1500	236	229	228	231.0	12.67	
5	-25	10	60	-250	-1500	600	-15000	230	222	203	218.33	128.22	
6	-25	60	-5	-1500	125	-300	7500	194	231	202	209.0	252.67	
7	5	10	-5	50	-25	-50	-250	214	226	207	215.67	61.56	
8	5	60	60	300	300	3600	18000	218	207	230	218.33	88.22	

Критерій Кохрена: Gr = 0.319  
Дисперсії однорідні з вірогідністю 95%.

Критерій Стюдента:  
108.00  
1.56  
0.41  
0.04  
1.52  
0.41  
0.74  
1.93  
b1, b2, b3, b4, b12, b13 - коефіцієнти рівняння регресії приймаємо незначними, виключаємо їх з рівняння.  
Рівняння регресії:  $y = +219.25 - 3.92 \cdot x_{123}$

Критерій Фішера: Fp = 2.597  
Табличне значення критерія Фішера: Ft = 2.6572

Рівняння регресії адекватно оригіналу.

Process finished with exit code 0

## Висновок:

В даній лабораторній роботі я провела повний трьохфакторний експеримент з трьома статистичними.