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 №3
з дисципліни «Методи оптимізації та планування експерименту»
на тему: «Проведення трьохфакторного експерименту з використанням лінійного рівняння регресії.»

Виконав:
студент групи ІО-92
Гладков Данііл
Залікова книжка № ІО-9204
Номер у списку групи - 02

Перевірив:
ас. Регіда П.Г.

Київ - 2021

Тема: «Проведення трьохфакторного експерименту з використанням лінійного рівняння регресії.»

Мета: Провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

Завдання:

1. Скласти матрицю планування для дробового трьохфакторного експерименту. Провести експеримент в усіх точках факторного простору, повторивши N експериментів, де N – кількість експериментів (рядків матриці планування) в усіх точках факторного простору – знайти значення функції відгуку У. Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі (випадковим чином).

$y_{max} = 200 + x_{cp\,max}$;

$y_{min} = 200 + x_{cp\,min}$;

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

2. Знайти коефіцієнти лінійного рівняння регресії. Записати лінійне рівняння регресії.

3. Провести 3 статистичні перевірки.

4. Написати комп'ютерну програму, яка усе це виконує.

Порядок виконання роботи:

1. Вибрати з таблиці варіантів і записати в протокол інтервали значень для x_1, x_2, x_3 Обчислити і записати для x_1, x_2, x_3 значення, відповідні кодованим +1; -1 значенням факторів $\overline{x_1}$, $\overline{x_2}$, $\overline{x_3}$

2. Скласти матрицю планування для дробового трьохфакторного експерименту з використанням додаткового (фіктивного) нульового фактору ($\overline{x_0}$=1) та заповнити таблицю нормованими значеннями $\overline{x_1}$, $\overline{x_2}$, $\overline{x_3}$.

	$\overline{x_0}$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	y_{i1}	y_{i2}	…	y_{ij}	…	y_{im}
1										
2										
3										
4										

Також скласти таблицю з натуральних значень факторів.

3. Провести експеримент в усіх точках (знайти значення функції відгуку у). Значення функції відгуку знайти у відповідності діапазону.

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

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

4. Рівняння регресії буде мати вигляд: $\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3$

Розрахувати коефіцієнти регресії для натуральних та нормованих значень.

5. Перевірити однорідність дисперсії за критерієм Кохрена, нуль-гіпотезу за критерієм Стьюдента та адекватність моделі за критерієм Фішера.

Варіант:

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
202	20	70	30	80	30	35

Роздруківка тексту програми:

```
from prettytable import PrettyTable as PT
from random import randint
import numpy as np
from math import *

class Laba3:
    def __init__(self):
        self.M = 3
        self.N = 4
        self.X1min, self.X2min, self.X3min = 20, 30, 30
        self.X1max, self.X2max, self.X3max = 70, 80, 35
        self.X_min, self.X_max = ((self.X1min + self.X2min + self.X3min)/3),
((self.X1max + self.X2max + self.X3max)/3)
        self.Ymin, self.Ymax = round(200 + self.X_min), round(200 + self.X_max)
        self.X = [[self.X1min, self.X2min, self.X3min],
                  [self.X1min, self.X2max, self.X3max],
                  [self.X1max, self.X2min, self.X3max],
                  [self.X1max, self.X2max, self.X3min]]
        self.Xkod = [[1, -1, -1, -1],
                     [1, -1,  1,  1],
                     [1,  1, -1,  1],
                     [1,  1,  1, -1]]
        self.main()

    def main(self):
        self.Y = [[randint(self.Ymin, self.Ymax) for i in range(self.M)] for j in
range(self.N)]
        self.table1 = PT()
        self.table1.field_names = ["X1min", "X1max", "X2min", "X2max", "X3min",
"X3max", "Ymin", "Ymax"]
        self.table1.add_rows([[self.X1min, self.X1max, self.X2min,self.X2max,
self.X3min,self.X3max, self.Ymin, self.Ymax]])
        print("Дані по варіанту:")
        print(self.table1)
        self.table2 = PT()
        self.table2.field_names = (["#", "X0", "X1", "X2", "X3"] + ["Y{}".format(i+1)
for i in range(self.M)])
        for i in range(self.N):
            self.table2.add_row([i+1] + self.Xkod[i] + self.Y[i])
        print("Матриця планування для дробового трьохфакторного експерименту:")
        print(self.table2)
        self.table3 = PT()
        self.table3.field_names = (["X1", "X2", "X3"] + ["Y{}".format(i+1) for i in
range(self.M)])
        for i in range(self.N):
            self.table3.add_row(self.X[i] + self.Y[i])
        print("Матриця планування з відповідними натуралізованими значеннями
факторів:")
        print(self.table3)

        self.Y1_, self.Y2_, self.Y3_, self.Y4_ = (sum(self.Y[0][j] for j in
range(self.M))/self.M),(sum(self.Y[1][j] for j in
range(self.M))/self.M),(sum(self.Y[2][j] for j in
```

```
range(self.M))/self.M),(sum(self.Y[3][j] for j in range(self.M))/self.M)
        self.mx1, self.mx2, self.mx3 = (sum(self.X[i][0] for i in
range(self.N))/self.N),(sum(self.X[i][1] for i in
range(self.N))/self.N),(sum(self.X[i][2] for i in range(self.N))/self.N)
        self.my = ((self.Y1_ + self.Y2_ + self.Y3_ + self.Y4_)/self.N)
        self.Y_=[self.Y1_, self.Y2_, self.Y3_, self.Y4_]
        self.a1, self.a2, self.a3 = (sum((self.X[i][0]*self.Y_[i]) for i in
range(self.N))/self.N),(sum((self.X[i][1]*self.Y_[i]) for i in
range(self.N))/self.N),(sum((self.X[i][2]*self.Y_[i]) for i in range(self.N))/self.N)
        self.a11, self.a22, self.a33 = (sum((self.X[i][0]*self.X[i][0]) for i in
range(self.N))/self.N),(sum((self.X[i][1]*self.X[i][1]) for i in
range(self.N))/self.N),(sum((self.X[i][2]*self.X[i][2]) for i in range(self.N))/self.N)
        self.a12 = self.a21 = (sum((self.X[i][0]*self.X[i][1]) for i in
range(self.N))/self.N)
        self.a13 = self.a31 = (sum((self.X[i][0]*self.X[i][2]) for i in
range(self.N))/self.N)
        self.a23 = self.a32 = (sum((self.X[i][1]*self.X[i][2]) for i in
range(self.N))/self.N)
        self.b0 = ((np.linalg.det(np.array([[self.my, self.mx1, self.mx2, self.mx3],
[self.a1, self.a11, self.a12, self.a13], [self.a2, self.a12, self.a22, self.a32],
[self.a3, self.a13, self.a23, self.a33]]))) / ((np.linalg.det(np.array([[1, self.mx1,
self.mx2, self.mx3], [self.mx1, self.a11, self.a12, self.a13], [self.mx2, self.a12,
self.a22, self.a32], [self.mx3, self.a13, self.a23, self.a33]])))))
        self.b1 = ((np.linalg.det(np.array([[1, self.my, self.mx2, self.mx3],
[self.mx1, self.a1, self.a12, self.a13], [self.mx2, self.a2, self.a22, self.a32],
[self.mx3, self.a3, self.a23, self.a33]]))) / ((np.linalg.det(np.array([[1, self.mx1,
self.mx2, self.mx3], [self.mx1, self.a11, self.a12, self.a13], [self.mx2, self.a12,
self.a22, self.a32], [self.mx3, self.a13, self.a23, self.a33]])))))
        self.b2 = ((np.linalg.det(np.array([[1, self.mx1, self.my, self.mx3],
[self.mx1, self.a11, self.a1, self.a13], [self.mx2, self.a12, self.a2, self.a32],
[self.mx3, self.a13, self.a3, self.a33]]))) / ((np.linalg.det(np.array([[1, self.mx1,
self.mx2, self.mx3], [self.mx1, self.a11, self.a12, self.a13], [self.mx2, self.a12,
self.a22, self.a32], [self.mx3, self.a13, self.a23, self.a33]])))))
        self.b3 = ((np.linalg.det(np.array([[1, self.mx1, self.mx2, self.my],
[self.mx1, self.a11, self.a12, self.a1], [self.mx2, self.a12, self.a22, self.a2],
[self.mx3, self.a13, self.a23, self.a3]]))) / ((np.linalg.det(np.array([[1, self.mx1,
self.mx2, self.mx3], [self.mx1, self.a11, self.a12, self.a13], [self.mx2, self.a12,
self.a22, self.a32], [self.mx3, self.a13, self.a23, self.a33]])))))
        print("Рівняння регресії: y =
{0:.2f}+({1:.2f})*X1+({2:.2f})*X2+({3:.2f})*X3".format(self.b0, self.b1, self.b2,
self.b3))
        print("Перевірка:")
        print("{0:.2f}+({1:.2f})*({2:.2f})+({3:.2f})*({4:.2f})+({5:.2f})*({6:.2f}) =
{7:.2f} = Y1cep = {8:.2f}".format(self.b0, self.b1, self.X1min, self.b2, self.X2min,
self.b3, self.X3min, (self.b0 + (self.b1 * self.X1min) + (self.b2 * self.X2min) +
(self.b3 * self.X3min)), self.Y1_))
        print("{0:.2f}+({1:.2f})*({2:.2f})+({3:.2f})*({4:.2f})+({5:.2f})*({6:.2f}) =
{7:.2f} = Y2cep = {8:.2f}".format(self.b0, self.b1, self.X1min, self.b2, self.X2max,
self.b3, self.X3max, (self.b0 + (self.b1 * self.X1min) + (self.b2 * self.X2max) +
(self.b3 * self.X3max)), self.Y2_))
        print("{0:.2f}+({1:.2f})*({2:.2f})+({3:.2f})*({4:.2f})+({5:.2f})*({6:.2f}) =
{7:.2f} = Y3cep = {8:.2f}".format(self.b0, self.b1, self.X1max, self.b2, self.X2min,
self.b3, self.X3max, (self.b0 + (self.b1 * self.X1max) + (self.b2 * self.X2min) +
(self.b3 * self.X3max)), self.Y3_))
        print("{0:.2f}+({1:.2f})*({2:.2f})+({3:.2f})*({4:.2f})+({5:.2f})*({6:.2f}) =
{7:.2f} = Y4cep = {8:.2f}".format(self.b0, self.b1, self.X1max, self.b2, self.X2max,
self.b3, self.X3min, (self.b0 + (self.b1 * self.X1max) + (self.b2 * self.X2max) +
(self.b3 * self.X3min)), self.Y4_))
        self.cochrane()

    def cochrane(self):
        print("\nПеревірка рівномірності дисперсій за критерієм Кохрена (M = {0}, N =
```

```
{1}):".format(self.M, self.N))
        self.cochraneTable = {1: 9065, 2: 7679, 3: 6841, 4: 6287, 5: 5892, 6: 5598, 7:
5365, 8: 5175, 9: 5017, 10: 4884}
        self.Ydisp = [np.var(i) for i in self.Y]
        self.GP = max(self.Ydisp)/sum(self.Ydisp)
        self.F1 = self.M - 1
        self.F2 = self.N
        self.q = 0.05
        print("F1 = M - 1 = {0} - 1 = {1} \nF2 = N = {2} \nq = {3}".format(self.M,
self.F1,self.F2, self.q))
        self.cochraneValues, self.cochraneKeys = list(self.cochraneTable.values()),
list(self.cochraneTable.keys())
        for keys in range(len(self.cochraneKeys)):
            if (self.cochraneKeys[keys] == self.F1):
                self.GT = self.cochraneValues[keys]/pow(10,4)
        if (self.GP < self.GT):
            print("GP = {0:.4f} < GT = {1:.4f} - Дисперсія
однорідна!".format(self.GP,self.GT))
            self.student()
        else:
            print("GP = {0:.4f} > GT = {1} - Дисперсія неоднорідна! Змінимо M на
M=M+1".format(self.GP, self.GT))
            self.M = self.M + 1
            self.main()

    def student(self):
        print("\nПеревірка значимості коефіцієнтів регресії згідно критерію Стьюдента
(M = {0}, N = {1}):".format(self.M, self.N))
        self.studentTable = {1: 12.71, 2: 4.303, 3: 3.182, 4: 2.776, 5: 2.571, 6:
2.447, 7: 2.365, 8: 2.306, 9: 2.262, 10: 2.228,
                             11: 2.201, 12: 2.179, 13: 2.160, 14: 2.145, 15: 2.131, 16:
2.120, 17: 2.110, 18: 2.101, 19: 2.093, 20: 2.086,
                             21: 2.080, 22: 2.074, 23: 2.069, 24: 2.064, 25: 2.060, 26:
2.056, 27: 2.052, 28: 2.048, 29: 2.045, 30: 2.042}
        self.Sb=(float(sum(self.Ydisp))/self.N)
        self.Sbs=sqrt(((self.Sb)/(self.N*self.M)))
        self.xis = np.array([[x[i] for x in self.Xkod] for i in range(len(self.Xkod))])
        self.Beta = np.array([np.average(self.Y_*self.xis[i]) for i in
range(len(self.xis))])
        print("Оцінки коефіцієнтів Bs: B1={0:.2f}, B2={1:.2f}, B3={2:.2f},
B4={3:.2f}".format(self.Beta[0],self.Beta[1],self.Beta[2],self.Beta[3]))
        self.t = np.array([((fabs(self.Beta[i]))/self.Sbs) for i in range(self.N)])
        print("Коефіцієнти ts: t1={0:.2f}, t2={1:.2f}, t3={2:.2f},
t4={3:.2f}".format(self.t[0],self.t[1],self.t[2],self.t[3]))
        self.F3=self.F1*self.F2
        print("F3 = F1*F2 = {0}*{1} = {2} \nq = {3}".format(self.F1, self.F2, self.F3,
self.q))
        self.studentValues, self.studentKeys = list(self.studentTable.values()),
list(self.studentTable.keys())
        for keys in range(len(self.studentKeys)):
            if (self.studentKeys[keys] == self.F3):
                self.Ttab = self.studentValues[keys]
        print("t табличне = {0}".format(self.Ttab))
        self.ZO={}
        for i in range(len(self.t)):
            if ((self.t[i])>self.Ttab):
                self.ZO[i] = 1
            if ((self.t[i]) < self.Ttab):
                self.ZO[i] = 0
        print("Рівняння регресії: y =
{0:.2f}*({1})+({2:.2f})*({3})*X1+({4:.2f})*({5})*X2+({6:.2f})*({7})*X3".format(self.b0,
self.ZO[0],self.b1,self.ZO[1],self.b2,self.ZO[2],self.b3,self.ZO[3]))
```

```
self.Y1v=self.b0*(self.ZO[0])+self.b1*(self.ZO[1])*self.X[0][0]+self.b2*(self.ZO[2])*se
lf.X[0][1]+self.b3*(self.ZO[3])*self.X[0][2]

self.Y2v=self.b0*(self.ZO[0])+self.b1*(self.ZO[1])*self.X[1][0]+self.b2*(self.ZO[2])*se
lf.X[1][1]+self.b3*(self.ZO[3])*self.X[1][2]

self.Y3v=self.b0*(self.ZO[0])+self.b1*(self.ZO[1])*self.X[2][0]+self.b2*(self.ZO[2])*se
lf.X[2][1]+self.b3*(self.ZO[3])*self.X[2][2]

self.Y4v=self.b0*(self.ZO[0])+self.b1*(self.ZO[1])*self.X[3][0]+self.b2*(self.ZO[2])*se
lf.X[3][1]+self.b3*(self.ZO[3])*self.X[3][2]
        self.Yv=[self.Y1v, self.Y2v, self.Y3v, self.Y4v]
        self.fisher()

    def fisher(self):
        print("\nПеревірка адекватності за критерієм Фішера (M = {0}, N =
{1}):".format(self.M, self.N))
        self.fisherTable = {
            8:  [5.3, 4.5, 4.1, 3.8, 3.7, 3.6],
            12: [4.8, 3.9, 3.5, 3.3, 3.1, 3.0],
            16: [4.5, 3.6, 3.2, 3.0, 2.9, 2.7],
            20: [4.4, 3.5, 3.1, 2.9, 2.7, 2.6],
            24: [4.3, 3.4, 3.0, 2.8, 2.6, 2.5],
            28: [4.2, 3.3, 3.0, 2.7, 2.6, 2.4]}
        self.d=0
        for i in range(len(self.ZO)):
            if (self.ZO[i]==1):
                self.d+=1
        print("Кількість значимих коефіцієнтів d={0}".format(self.d))
        self.Yrazn=0
        for i in range(self.N):
            self.Yrazn+=pow((self.Yv[i]-self.Y_[i]),2)
        self.Sad=((self.M/(self.N-self.d))*self.Yrazn)
        print("Sad = {0:.2f}".format(self.Sad))
        self.FP=(self.Sad/self.Sb)
        print("FP = {0:.2f}".format(self.FP))
        self.F4=self.N-self.d
        print("F4 = N - d = {0} - {1} = {2} \nq = {3}".format(self.N, self.d, self.F4,
self.q))
        self.FT= self.fisherTable[self.F3][self.F4-1]
        print("FT = {0}".format(self.FT))
        if (self.FT>self.FP):
            print("FT = {0:.2f} > FP = {1:.2f} - рівняння регресії адекватно
оригіналу".format(self.FT,self.FP))
        if (self.FP>self.FT):
            print("FP = {0:.2f} > FT = {1:.2f} - рівняння регресії неадекватно
оригіналу".format(self.FP,self.FT))
Laba3()
```

Роздруківка результатів виконання програми:

```
D:\Programming\Anaconda\python.exe "C:/Users/Daniil/Desktop/КПИ/4 СЕМЕСТР/МОПЕ (4)/Лабораторні роботи/Laba3МОРЕ/Laba3МОРЕ.py"
Дані по варіанту:
+-------+-------+-------+-------+-------+-------+------+------+
| X1min | X1max | X2min | X2max | X3min | X3max | Ymin | Ymax |
+-------+-------+-------+-------+-------+-------+------+------+
|   20  |   70  |   30  |   80  |   30  |   35  | 227  | 262  |
+-------+-------+-------+-------+-------+-------+------+------+
Матриця планування для дробового трьохфакторного експерименту:
+---+----+----+----+----+-----+-----+-----+
| # | X0 | X1 | X2 | X3 |  Y1 |  Y2 |  Y3 |
+---+----+----+----+----+-----+-----+-----+
| 1 | 1  | -1 | -1 | -1 | 241 | 261 | 233 |
| 2 | 1  | -1 | 1  | 1  | 258 | 259 | 254 |
| 3 | 1  | 1  | -1 | 1  | 236 | 245 | 236 |
| 4 | 1  | 1  | 1  | -1 | 242 | 248 | 259 |
+---+----+----+----+----+-----+-----+-----+
Матриця планування з відповідними натуралізованими значеннями факторів:
+----+----+----+-----+-----+-----+
| X1 | X2 | X3 |  Y1 |  Y2 |  Y3 |
+----+----+----+-----+-----+-----+
| 20 | 30 | 30 | 241 | 261 | 233 |
| 20 | 80 | 35 | 258 | 259 | 254 |
| 70 | 30 | 35 | 236 | 245 | 236 |
| 70 | 80 | 30 | 242 | 248 | 259 |
+----+----+----+-----+-----+-----+
Рівняння регресії: y = 236.87+(-0.13)*X1+(0.23)*X2+(0.13)*X3
Перевірка:
236.87+(-0.13)*(20.00)+(0.23)*(30.00)+(0.13)*(30.00) = 245.00 = Y1сер = 245.00
236.87+(-0.13)*(20.00)+(0.23)*(80.00)+(0.13)*(35.00) = 257.00 = Y2сер = 257.00
236.87+(-0.13)*(70.00)+(0.23)*(30.00)+(0.13)*(35.00) = 239.00 = Y3сер = 239.00
236.87+(-0.13)*(70.00)+(0.23)*(80.00)+(0.13)*(30.00) = 249.67 = Y4сер = 249.67

Перевірка рівномірності дисперсій за критерієм Кохрена (M = 3, N = 4):
F1 = M - 1 = 3 - 1 = 2
F2 = N = 4
q = 0.05
GP = 0.6575 < GT = 0.7679 - Дисперсія однорідна!

Перевірка значимості коефіцієнтів регресії згідно критерію Стьюдента (M = 3, N = 4):
Оцінки коефіцієнтів Bs: B1=247.67, B2=-3.33, B3=5.67, B4=0.33
Коефіцієнти ts: t1=118.16, t2=1.59, t3=2.70, t4=0.16
F3 = F1*F2 = 2*4 = 8
q = 0.05
t табличне = 2.306
Рівняння регресії: y = 236.87*(1)+(-0.13)*(0)*X1+(0.23)*(1)*X2+(0.13)*(0)*X3

Перевірка адекватності за критерієм Фішера (M = 3, N = 4):
Кількість значимих коефіцієнтів d=2
Sad = 84.00
FP = 1.59
F4 = N - d = 4 - 2 = 2
q = 0.05
FT = 4.5
FT = 4.50 > FP = 1.59 - рівняння регресії адекватно оригіналу

Process finished with exit code 0
```

Відповіді на контрольні запитання:

1. Що називається дробовим факторним експериментом?

Дробовий факторний експеримент – це скорочений експеримент від повного факторного експерименту (його частина).

2. Для чого потрібно розрахункове значення Кохрена?

Розрахункове значення Кохрена необхідно для перевірки дисперсії.

3. Для чого перевіряється критерій Стьюдента?

Критерій Стьюдента перевіряється для перевірки значущості коефіцієнтів рівняння регресії.

4. Чим визначається критерій Фішера і як його застосовувати?

Критерій Фішера застосовується для перевірки адекватності моделі оригіналу. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності.

Висновки:

У ході виконання лабораторної роботи я провів дробовий трьохфакторний експеримент, склав матрицю планування, знайшов коефіцієнти рівняння регресії, провів 3 статистичні перевірки. Закріпив отримані знання практичним їх використанням при написанні програми, що реалізує завдання лабораторної роботи. Мета лабораторної роботи досягнута.