Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ

Звіт
з лабораторної роботи №2
з навчальної дисципліни «Методи оптимізації та планування експерименту»
Тема: Проведення двофакторного експерименту з
використанням лінійного рівняння регресії

Виконав:
Студент 2 курсу кафедри ОТ ФІОТ
Навчальної групи ІО-91
Тарасенко Андрій

Перевірив:
Регіда П. Г.

Київ 2021

Мета: провести двофакторний експеримент, перевірити однорідність дисперсії за критерієм Романовського, отримати коефіцієнти рівняння регресії, провести натуралізацію рівняння регресії.

Завдання:

1. Записати лінійне рівняння регресії.
2. Обрати тип двофакторного експерименту і скласти матрицю планування для нього з використанням додаткового нульового фактору (x o =1).
3. Провести експеримент в усіх точках повного факторного простору (знайти значення функції відгуку y). Значення функції відгуку задати випадковим чином у відповідності до варіанту у діапазоні y min ÷ y max

122	-5	15	10	60

Лістинг програми

```
"""
 * Copyright © 2021 drewg3r
 * https://github.com/drewg3r/DOX-labs

main.py: main file to run the program.
"""

import math
import random
import numpy as np
from tabulate import tabulate

class Lab2:
    m = 5
    X = [[-1, 1, -1], [-1, -1, 1]]
    Y = []

    def __init__(self, x1min=-25, x1max=75, x2min=5, x2max=40, ymin=9, ymax=20,
test_y=False):
        print("DOX Lab2")
        self.x1min = x1min
        self.x1max = x1max
        self.x2min = x2min
        self.x2max = x2max
        self.ymin = ymin
        self.ymax = ymax
        if test_y:
            self.generate_test_y()
        else:
            self.generate_random_y()

        # Show X and Y values as a table
        table = []
        table.append([x[0] for x in self.X] + [x[0] for x in self.Y])
        table.append([x[1] for x in self.X] + [x[1] for x in self.Y])
        table.append([x[2] for x in self.X] + [x[2] for x in self.Y])
        print(tabulate(table, headers=["X1","X2","Y1","Y2","Y3","Y4","Y5"],
floatfmt=".4f", tablefmt='fancy_grid'))

        # Calculate average for every row
        self.y1avg = sum([x[0] for x in self.Y])/5
        self.y2avg = sum([x[1] for x in self.Y])/5
        self.y3avg = sum([x[2] for x in self.Y])/5

    def uniformity_of_dispersion_check(self):
        # Calculate dispersions for every row
        d1 = sum([((x[0] - self.y1avg)**2) for x in self.Y])/5
        d2 = sum([((x[1] - self.y2avg)**2) for x in self.Y])/5
        d3 = sum([((x[2] - self.y3avg)**2) for x in self.Y])/5

        # Calculate main deviation
        m_deviation = math.sqrt((2*(2*self.m-2))/(self.m*(self.m-4)))

        Fuv1 = d1/d2
        Fuv2 = d3/d1
        Fuv3 = d3/d2

        sigma_uv1 = ((self.m - 2)/self.m) * Fuv1
        sigma_uv2 = ((self.m - 2)/self.m) * Fuv2
```

```
        sigma_uv3 = ((self.m - 2)/self.m) * Fuv3

        Ruv1 = abs(sigma_uv1-1)/m_deviation
        Ruv2 = abs(sigma_uv2-1)/m_deviation
        Ruv3 = abs(sigma_uv3-1)/m_deviation

        print("\nAverage Y1: {:.3f}\nAverage Y2: {:.3f}\nAverage Y3: {:.3f}\
n".format(self.y1avg, self.y2avg, self.y3avg))
        print("DISPERSIONS:\nd1 = {:.3f}\nd2 = {:.3f}\nd3 = {:.3f}\n".format(d1, d2,
d3))
        print("Main deviation: {:.3f}\n".format(m_deviation))
        print("Fuv1 = {:.3f}\nFuv2 = {:.3f}\nFuv3 = {:.3f}\n".format(Fuv1, Fuv2, Fuv3))
        print("sigma_uv1 = {:.3f}\nsigma_uv2 = {:.3f}\nsigma_uv3 = {:.3f}\
n".format(sigma_uv1, sigma_uv2, sigma_uv3))
        print("Ruv1 = {:.3f}\nRuv2 = {:.3f}\nRuv3 = {:.3f}".format(Ruv1, Ruv2, Ruv3))
        if Ruv1 <= 2 and Ruv2 <= 2 and Ruv3 <= 2:
            print("Uniformity of dispersion check passed successfully(all R's <= 2)\n")

    def normalize_regression_factors(self):
        mx1 = sum(self.X[0])/3
        mx2 = sum(self.X[1])/3
        my = (self.y1avg + self.y2avg + self.y3avg)/3
        a1 = sum([x**2 for x in self.X[0]])/3
        a2 = ((self.X[0][0] * self.X[1][0]) + (self.X[0][1] * self.X[1][1]) + (self.X[0]
[2] * self.X[1][2]))/3
        a3 = sum([x**2 for x in self.X[1]])/3
        a11 = (self.X[0][0] * self.y1avg + self.X[0][1] * self.y2avg + self.X[0][2] *
self.y3avg)/3
        a22 = (self.X[1][0] * self.y1avg + self.X[1][1] * self.y2avg + self.X[1][2] *
self.y3avg)/3

        # Calculating determinants of matrixes
        self.b0 = np.linalg.det([[my, mx1, mx2], [a11, a1, a2], [a22, a2,
a3]])/np.linalg.det([[1, mx1, mx2], [mx1, a1, a2], [mx2, a2, a3]])
        self.b1 = np.linalg.det([[1, my, mx2], [mx1, a11, a2], [mx2, a22,
a3]])/np.linalg.det([[1, mx1, mx2], [mx1, a1, a2], [mx2, a2, a3]])
        self.b2 = np.linalg.det([[1, mx1, my], [mx1, a1, a11], [mx2, a2,
a22]])/np.linalg.det([[1, mx1, mx2], [mx1, a1, a2], [mx2, a2, a3]])

        print("Normalized regression equation: y = {:.3f} + {:.3f}x1 +
{:.3f}x2".format(self.b0, self.b1, self.b2))

    def naturalize_regression_factors(self):
        dx1 = abs(self.x1max - self.x1min)/2
        dx2 = abs(self.x2max - self.x2min)/2
        x10 = (self.x1max + self.x1min)/2
        x20 = (self.x2max + self.x2min)/2
        a0 = self.b0 - self.b1 * (x10/dx1) - self.b2 * (x20/dx2)
        a1 = self.b1/dx1
        a2 = self.b2/dx2

        print("Naturalized regression equation: y = {:.3f} + {:.3f}x1 +
{:.3f}x2".format(a0, a1, a2))


    def generate_random_y(self):
        for i in range(5):
            self.Y.append([random.randint(self.ymin, self.ymax),
random.randint(self.ymin, self.ymax), random.randint(self.ymin, self.ymax)])

    def generate_test_y(self):
```

```
        # You can use an example Y values. Just set test_y variable to True when
creating Lab2 object
        self.Y.append([9, 15, 20])
        self.Y.append([10, 14, 18])
        self.Y.append([11, 10, 12])
        self.Y.append([15, 12, 10])
        self.Y.append([9, 14, 16])

# Creating Lab2 object with variant 122 parameters(test_y=False: using random values)
lab2 = Lab2(x1min=-5, x1max=15, x2min=10, x2max=60, ymin=-20, ymax=80, test_y=False)
lab2.uniformity_of_dispersion_check()
lab2.normalize_regression_factors()
lab2.naturalize_regression_factors()
```

Результат виконання роботи: нормований план експерименту та функція відгуку для точки плану, що відповідає критерію оптимальності

DOX Lab2

X1	X2	Y1	Y2	Y3	Y4	Y5
-1	-1	35	26	-17	60	-12
1	-1	58	-20	11	-15	23
-1	1	53	15	-16	0	79

```
Average Y1: 18.400
Average Y2: 11.400
Average Y3: 26.200

DISPERSIONS:
d1 = 848.240
d2 = 797.840
d3 = 1219.760

Main deviation: 1.789

Fuv1 = 1.063
Fuv2 = 1.438
Fuv3 = 1.529

sigma_uv1 = 0.638
sigma_uv2 = 0.863
sigma_uv3 = 0.917

Ruv1 = 0.202
Ruv2 = 0.077
Ruv3 = 0.046
Uniformity of dispersion check passed successfully(all R's <= 2)

Normalized regression equation: y = 18.800 + -3.500x1 + 3.900x2
Naturalized regression equation: y = 15.090 + -0.350x1 + 0.156x2
```

Відповіді на контрольні запитання

Що таке регресійні поліноми і де вони застосовуються ?

В теорії планування експерименту найважливішою частиною є оцінка результатів вимірів . При цьому використовують апроксимуючі поліноми , за допомогою яких ми можемо описати нашу функцію . В ТПЕ ці поліноми отримали спеціальну назву - регресійні поліноми , а їх знаходження та аналіз - регресійний аналіз . Найчастіше в якості базисної функції використовується ряд Тейлора , який має скінченну кількість членів.

Визначення однорідності дисперсії.

Однорідність дисперсії показує відсутність грубих помилок в результатах експерименту(тобто дисперсія результатів задовільная критерій Романовського, результати експерименту мають приблизно однакові значення).

Що називається повним факторним експериментом?

Повним факторним експериментом (ПФЕ) називається такий експеримент, при реалізації якого визначається значення параметра оптимізації при всіх можливих поєднаннях рівнів варіювання факторів.

Висновки:

Під час виконання лабораторної роботи було проведено двофакторний експеримент, перевірено однорідність дисперсії за критерієм Романовського, отримано коефіцієнти рівняння регресії, проведено натуралізацію рівняння регресії, закріплено отримані знання практичним їх використанням при написанні програми, що реалізує завдання на лабораторну роботу.