

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет України
«Київський Політехнічний Інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни

«Методи оптимізації та планування експерименту»

**Тема: ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ
З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ.**

Виконав:

Студент 2-го курсу ФІОТ
групи ІО-ХХ

Перевірив:

Регіда П. Г.

Київ – 2021

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

Варант завдання:

328	-40	20	10	60	-20	20
-----	-----	----	----	----	-----	----

```
import random
import numpy
import scipy.stats

import tkinter.messagebox

root = tkinter.Tk()

x1_min = -40
x1_max = 20

x2_min = 10
x2_max = 60

x3_min = -20
x3_max = 20

xm_min = (x1_min + x2_min + x3_min) / 3
xm_max = (x1_max + x2_max + x3_max) / 3
y_min = 200 + xm_min
y_max = 200 + xm_max

xn = [[-1, -1, -1],
       [-1, 1, 1],
       [1, -1, 1],
       [1, 1, -1]]

x = [[10, -35, 10],
      [10, 15, 15],
      [60, -35, 15],
      [60, 15, 10]]

m = 2
y = [[random.randint(int(y_min), int(y_max)) for i in range(m)] for j in
      range(4)]

def kohren(dispersion, m):
    gt = {1: 0.9065, 2: 0.7679, 3: 0.6841, 4: 0.6287, 5: 0.5892, 6: 0.5598,
          7: 0.5365, 8: 0.5175, 9: 0.5017, 10: 0.4884}
    gp = max(dispersion) / sum(dispersion)
    return gp < gt[m - 1]

def student(dispersion_reproduction, m, y_mean, xn):
    dispersion_statistic_mark = (dispersion_reproduction / (4 * m)) ** 0.5
```

```

beta = [1 / 4 * sum(y_mean[j] for j in range(4))]
for i in range(3):
    b = 0
    for j in range(4):
        b += y_mean[j] * xn[j][i]
    beta.append(1 / 4 * b)

t = []
for i in beta:
    t.append(abs(i) / dispersion_statistic_mark)

check_st = scipy.stats.t.ppf((1 + 0.95)/2, (m-1) * 4)

return t[0] > check_st, t[1] > check_st, t[2] > check_st, t[3] > check_st

def fisher(m, d, y_mean, yo, dispersion_reproduction):
    dispersion_ad = 0
    for i in range(4):
        dispersion_ad += (yo[i] - y_mean[i]) ** 2

    dispersion_ad = dispersion_ad * m / (4 - d)

    fp = dispersion_ad / dispersion_reproduction
    check_f = scipy.stats.f.ppf(0.95, 4 - d, (m-1)*4)

    return fp < check_f

def normalized_multiplier(x, y_mean):
    mx = [0, 0, 0]
    axx = [0, 0, 0]
    ax = [0, 0, 0]
    for i in range(3):
        for j in range(4):
            mx[i] += x[j][i]
            axx[i] += x[j][i] ** 2
            ax[i] += x[j][i] * y_mean[j]
        mx[i] /= 4
        axx[i] /= 4
        ax[i] /= 4

    my = sum(y_mean) / 4

    a12 = (x[0][0] * x[0][1] + x[1][0] * x[1][1] + x[2][0] * x[2][1] +
x[3][0] * x[3][1]) / 4
    a13 = (x[0][0] * x[0][2] + x[1][0] * x[1][2] + x[2][0] * x[2][2] +
x[3][0] * x[3][2]) / 4
    a23 = (x[0][1] * x[0][2] + x[1][1] * x[1][2] + x[2][1] * x[2][2] +
x[3][1] * x[3][2]) / 4

    a = numpy.array([[1, *mx],
                     [mx[0], axx[0], a12, a13],
                     [mx[1], a12, axx[1], a23],
                     [mx[2], a13, a23, axx[2]]])
    c = numpy.array([my, *ax])
    b = numpy.linalg.solve(a, c)
    return b

def next_m(arr):
    for i in range(4):

```

```

arr[i].append(random.randint(int(y_min), int(y_max)))

while True:
    while True:
        y_mean = []
        for i in range(4):
            y_mean.append(sum(y[i]) / m)

        dispersion = []
        for i in range(len(y)):
            dispersion.append(0)
            for j in range(m):
                dispersion[i] += (y_mean[i] - y[i][j]) ** 2
            dispersion[i] /= m

        dispersion_reproduction = sum(dispersion) / 4

        if kohren(dispersion, m):
            break
        else:
            m += 1
            next_m(y)

    k = student(dispersion_reproduction, m, y_mean, xn)
    d = sum(k)

    b = normalized_multiplier(x, y_mean)
    b = [b[i] * k[i] for i in range(4)]

    yo = []
    for i in range(4):
        yo.append(b[0] + b[1] * x[i][0] + b[2] * x[i][1] + b[3] * x[i][2])

    if d == 4:
        m += 1
        next_m(y)

    elif fisher(m, d, y_mean, yo, dispersion_reproduction):
        break

    else:
        m += 1
        next_m(y)

tkinter.Label(text="x1").grid()

tkinter.Label(text="x2").grid(row=0, column=1)
tkinter.Label(text="x3").grid(row=0, column=2)
for i in range(m):
    tkinter.Label(text="yi" + str(i + 1)).grid(row=0, column=i + 3)
for i in range(len(x)):
    for j in range(len(x[i])):
        tkinter.Label(text=x[i][j]).grid(row=i + 1, column=j)
for i in range(len(y)):
    for j in range(len(y[i])):
        tkinter.Label(text=y[i][j]).grid(row=i + 1, column=j + 3)
tkinter.Label(text="Рівняння регресії:").grid(columnspan=m + 3)
text = "y = " + "{0:.2f}".format(b[0])
for i in range(3):
    if b[i + 1] != 0:
        text = text + " + {0:.2f}".format(b[i + 1]) + " * x" + str(i + 1)

```

```
tkinter.Label(text=text).grid(columnspan=m + 3)
tkinter.Label(text="Перевірка:").grid(columnspan=m + 3)

for i in range(4):
    tkinter.Label(text="yc" + str(i + 1) + " =" +
"{0:.2f}".format(y_mean[i])).grid(columnspan=m + 3)
    tkinter.Label(text="y" + str(i + 1) + " = " +
"{0:.2f}".format(yo[i])).grid(columnspan=m + 3)

root.mainloop()
```

x1	x2	x3	y1	y2	y3	y4	y5	y6	y7	y8
10	-35	10	197	186	232	212	208	211	230	207
10	15	15	186	195	227	192	208	219	214	188
60	-35	15	195	191	186	196	186	194	189	192
60	15	10	196	217	218	192	195	188	225	199

Рівняння регресії:

$$y = 233.72 + -0.19 * x1 + -1.94 * x3$$

Перевірка:

yc1 = 210.38
y1 = 212.43
yc2 = 203.62
y2 = 202.74
yc3 = 191.12
y3 = 193.18
yc4 = 203.75
y4 = 202.87