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»

Виконала:
студентка групи ІВ-81
Дьяченко Т. С.
Перевірив:
Регіда П. Г.

Київ
2020 р.

Варіант:

112	-40	20	-35	15	20	25

Код програми:

```
      import random, math, numpy
from scipy.stats import t, f
N = 4
m = 3
p = 0.95
q = 1-p
d = 4
f1 = m-1
f2 = N
f3 = f1*f2
f4 = N-d
x1min = -40
x1max = 20
x2min = -35
x2max = 15
x3min = 20
x3max = 25
xAvmin = (x1min + x2min + x3min)/3
xAvmax = (x1max + x2max + x3max)/3
ymin = 200 + xAvmin
ymax = 200 + xAvmax
random.seed()
table_NormExperiment = [["N", "x0", "x1", "x2", "x3"],
                        [1, 1, -1, -1, -1],
                        [2, 1, -1, -1, 1],
                        [3, 1, -1, 1, -1],
                        [4, 1, -1, 1, 1],
                        [5, 1, 1, -1, -1],
                        [6, 1, 1, -1, 1],
                        [7, 1, 1, 1, -1],
                        [8, 1, 1, 1, 1]]
table_NaturExperiment = [["N", "x1", "x2", "x3"],
                         [1, -40, -35, 20],
                         [2, -40, -35, 25],
                         [3, -40, 15, 20],
                         [4, -40, 15, 25],
                         [5, 20, -35, 20],
                         [6, 20, -35, 25],
                         [7, 20, 15, 20],
                         [8, 20, 15, 25]]
fractionalExp_indexes = [0, 1, 4, 6, 7]
b = [0, 0, 0, 0]
a = []
AvYs = []
DisYs = []
def print_table(table, indexes=[0,1,2,3,4,5,6,7,8]):
    print("\n", "-" * len(table[0]) * 11, "\n")
    for i in indexes:
        print("|", end="")
        for j in range(len(table[i])):
            if i > 0 and j > table[0].index("Yi1")-1:
                print("{:.2f}".format(float(table[i][j])), end="    |")
            else:
                print(table[i][j], " "*(9-len(str(table[i][j]))), end="|")
```

```
        print("\n", "-" * len(table[0])*11, "\n")
def randomize(s, e):
    global AvYs
    AvYs = []
    global DisYs
    DisYs = []
    for i in range(9):
        sum_y = 0
        for j in range(s, e):
            if i == 0:
                table_NaturExperiment[i].append("Yi{}".format(j-2))
                table_NormExperiment[i].append("Yi{}".format(j - 2))
            else:
                y = random.uniform(ymin, ymax)
                table_NaturExperiment[i].append(y)
                table_NormExperiment[i].append(y)
                sum_y += y
        if i in fractionalExp_indexes and not i == 0:
            AvYs.append(sum_y/m)
    for i in range(1,len(fractionalExp_indexes)):
        sum_y = 0
        for j in range(3, m+3):
            sum_y += pow(table_NormExperiment[fractionalExp_indexes[i]][j]- AvYs[i-
1], 2)
        DisYs.append(sum_y/(m-1))
def cochran():
    global DisYs
    max_dispersion = max(DisYs)
    Gp = max_dispersion/sum(DisYs)
    fisher = table_fisher(p, 1, f3)
    Gt = fisher/(fisher+f2-1)
    return Gp < Gt
def table_fisher(prob, d, f3):
    x_vec = [i*0.001 for i in range(int(10/0.001))]
    for i in x_vec:
        if abs(f.cdf(i, N-d, f3)-prob) < 0.0001:
            return i
def coef():
    mx1 = 0
    mx2 = 0
    mx3 = 0
    a1 = 0
    a2 = 0
    a3 = 0
    a11 = 0
    a22 = 0
    a33 = 0
    a12 = 0
    a13 = 0
    a23 = 0
    for i in range(1, len(fractionalExp_indexes)):
        if not i == 0:
            mx1 += table_NaturExperiment[fractionalExp_indexes[i]][1]
            mx2 += table_NaturExperiment[fractionalExp_indexes[i]][2]
            mx3 += table_NaturExperiment[fractionalExp_indexes[i]][3]
            a1 += table_NaturExperiment[fractionalExp_indexes[i]][1]*AvYs[i-1]
            a2 += table_NaturExperiment[fractionalExp_indexes[i]][2] * AvYs[i - 1]
            a3 += table_NaturExperiment[fractionalExp_indexes[i]][3] * AvYs[i - 1]
            a11 += pow(table_NaturExperiment[fractionalExp_indexes[i]][1], 2)
            a22 += pow(table_NaturExperiment[fractionalExp_indexes[i]][2], 2)
            a33 += pow(table_NaturExperiment[fractionalExp_indexes[i]][3], 2)
            a12 += table_NaturExperiment[fractionalExp_indexes[i]]
[1]*table_NaturExperiment[fractionalExp_indexes[i]][2]
```

```
            a13 += table_NaturExperiment[fractionalExp_indexes[i]][1] *
table_NaturExperiment[fractionalExp_indexes[i]][3]
            a23 += table_NaturExperiment[fractionalExp_indexes[i]][3] *
table_NaturExperiment[fractionalExp_indexes[i]][2]
            b[1] += table_NormExperiment[fractionalExp_indexes[i]][2] * AvYs[i - 1]
            b[2] += table_NormExperiment[fractionalExp_indexes[i]][3] * AvYs[i - 1]
            b[3] += table_NormExperiment[fractionalExp_indexes[i]][4] * AvYs[i - 1]
    mx1 /= N
    mx2 /= N
    mx3 /= N
    a1 /= N
    a2 /= N
    a3 /= N
    a11 /= N
    a22 /= N
    a33 /= N
    a12 /= N
    a13 /= N
    a23 /= N
    my = sum(AvYs)/len(AvYs)
    denominator = numpy.linalg.det(numpy.array([[1, mx1, mx2, mx3],[mx1, a11, a12,
a13], [mx2, a12, a22, a23], [mx3, a13, a23, a33]]))
    numerator_0 = numpy.linalg.det(numpy.array([[my, mx1, mx2, mx3],[a1, a11, a12,
a13], [a2, a12, a22, a23], [a3, a13, a23, a33]]))
    numerator_1 = numpy.linalg.det(numpy.array([[1, my, mx2, mx3],[mx1, a1, a12,
a13], [mx2, a2, a22, a23], [mx3, a3, a23, a33]]))
    numerator_2 = numpy.linalg.det(numpy.array([[1, mx1, my, mx3],[mx1, a11, a1,
a13], [mx2, a12, a2, a23], [mx3, a13, a3, a33]]))
    numerator_3 = numpy.linalg.det(numpy.array([[1, mx1, mx2, my],[mx1, a11, a12,
a1], [mx2, a12, a22, a2], [mx3, a13, a23, a3]]))
    b[0] = my
    b[1] /= N
    b[2] /= N
    b[3] /= N
    a.append(numerator_0/ denominator)
    a.append(numerator_1/ denominator)
    a.append(numerator_2 / denominator)
    a.append(numerator_3 / denominator)
    print("\nNormalized equation:\ny = {:.2f} {:+.2f}*x1 {:+.2f}*x2
{:.2f}*x3".format(b[0], b[1], b[2], b[3]))
    print("\nCheck:")
    for i in range(1, len(fractionalExp_indexes)):
        if not i == 0:
            print("{:.2f} {:+.2f} {:+.2f} {:+.2f} = {:.2f}\ny = {:.2f}\
n".format(b[0], b[1] * table_NormExperiment[fractionalExp_indexes[i]][2],
                                                                    b[2]
* table_NormExperiment[fractionalExp_indexes[i]][3],
                                                                    b[3]
* table_NormExperiment[fractionalExp_indexes[i]][4],
                                                                    b[0]
+ b[1] * table_NormExperiment[fractionalExp_indexes[i]][2] + b[2] *
table_NormExperiment[fractionalExp_indexes[i]][3] + b[3] *
table_NormExperiment[fractionalExp_indexes[i]][4],
                                                                AvYs[i -
1]))
    print("\nNaturalized equation:\ny = {:.2f} {:+.2f}*x1 {:+.2f}*x2
{:.2f}*x3".format(a[0], a[1], a[2], a[3]))
    print("\nCheck:")
    for i in range(1, len(fractionalExp_indexes)):
        if not i == 0:
            print("{:.2f} {:+.2f} {:+.2f} {:+.2f} = {:.2f}\ny = {:.2f}\
n".format(a[0], a[1] * table_NaturExperiment[
                fractionalExp_indexes[i]][1],
```

```
a[2] * table_NaturExperiment[

fractionalExp_indexes[i]][2],

a[3] * table_NaturExperiment[

fractionalExp_indexes[i]][3],

a[0] + a[1] * table_NaturExperiment[

fractionalExp_indexes[i]][1] + a[

2] * table_NaturExperiment[

fractionalExp_indexes[i]][2] + a[

3] * table_NaturExperiment[

fractionalExp_indexes[i]][3],

AvYs[i - 1]))
def student():
    global DisYs
    global AvYs
    global d
    AvDisYs = sum(DisYs)/len(DisYs)
    Sb = math.sqrt(AvDisYs/(N*m))
    t_val = []
    for x in range(4):
        new_beta = 0
        for i in range(len(fractionalExp_indexes)):
            if i > 0:
                new_beta += AvYs[i-1]*table_NormExperiment[fractionalExp_indexes[i]]
[x+1]
        t_val.append(math.fabs(new_beta/N)/Sb)
    t_cr = 0
    x_vec = [i * 0.0001 for i in range(int(5 / 0.0001))]
    par = 0.5 + p / 0.1 * 0.05
    for i in x_vec:
        if abs(t.cdf(i, f3) - par) < 0.000005:
            t_cr = i
            break
    print("According to Student's t-test these coefficients are insignificant:")
    insign = []
    for i in range(len(t_val)):
        if t_val[i] <= t_cr:
            insign.append(i)
            d -= 1
            print("t = {}\t\t\tt_cr = {}\t\t\tt < t_cr\nb{} = {:.2f} and a{} =
{:.2f}".format(t_val[1], t_cr, i, b[i], i, a[i]))
    print("\nThen the equations change:\nNormalized:\ny = ", end="")
    for i in range(len(b)):
        if not i in insign:
            if i == 0:
                print("{:.2f} ".format(b[i]), end="")
            else:
                print("{:+.2f}*x{}".format(b[i], i), end="")
    print("\n\nNaturalized:\ny = ", end="")
    for i in range(len(a)):
        if not i in insign:
            if i == 0:
                print("{:.2f} ".format(a[i]), end="")
```

```
            else:
                print("{:+.2f}*x{}".format(a[i], i), end="")
def fisher():
    AvDisYs = sum(DisYs) / len(DisYs)
    Sad = 0
    for dis in DisYs:
        Sad += dis*(m-1)
    Sad = Sad*m/(N-d)
    F_val = Sad/AvDisYs
    x_vec = [i * 0.001 for i in range(int(10 / 0.001))]
    F_cr = None
    for i in x_vec:
        if abs(f.cdf(i, N - d, f3) - p) < 0.0001:
            F_cr = i
    if not F_cr:
        print("\n\nSomething went wrong.\nUnable to calculate critical value for
Fisher's test")
    elif F_cr >= F_val:
        print("\n\nF = {}\t\t\tF_cr = {}\t\t\tF =< F_cr\nAccording to Fisher's F-test
model is adequate to the original.".format(F_val, F_cr))
    else:
        print("\n\nF = {}\t\t\tF_cr = {}\t\t\tF > F_cr\nAccording to Fisher's F-test
model is not adequate to the original.".format(F_val, F_cr))
startY = 3
endY = m + 3
randomize(startY, endY)
cochran_cond = cochran()
while not cochran_cond:
    m += 1
    startY = endY
    endY = m + 3
    randomize(startY, endY)
    cochran_cond = cochran()
print("Normalized Experiment (Full):")
print_table(table_NormExperiment)
print("Normalized Experiment (Fractional):")
print_table(table_NormExperiment, fractionalExp_indexes)
print("According to Cochran's C-test homogeneity of variance is confirmed")
print("\nNaturalized Experiment (Fractional):")
print_table(table_NaturExperiment, fractionalExp_indexes)
coef()
student()
fisher()
```

Результат виконання:

```
/home/mitya/.pycharm_projects/PZ_3/venv/bin/python /home/mitya/.pycharm_projects/PZ_3/main.py
Normalized Experiment (Full):

 ----------------------------------------------------------------------------------------

|N         |x0        |x1        |x2        |x3        |Yi1       |Yi2       |Yi3       |
 ----------------------------------------------------------------------------------------

|1         |1         |-1        |-1        |-1        |194.49    |208.69    |191.12    |
 ----------------------------------------------------------------------------------------

|2         |1         |-1        |-1        |1         |190.55    |212.36    |210.25    |
 ----------------------------------------------------------------------------------------

|3         |1         |-1        |1         |-1        |185.49    |219.76    |210.64    |
 ----------------------------------------------------------------------------------------

|4         |1         |-1        |1         |1         |215.39    |190.77    |203.98    |
 ----------------------------------------------------------------------------------------

|5         |1         |1         |-1        |-1        |200.06    |188.22    |196.36    |
 ----------------------------------------------------------------------------------------

|6         |1         |1         |-1        |1         |191.85    |202.35    |211.47    |
 ----------------------------------------------------------------------------------------

|7         |1         |1         |1         |-1        |195.38    |186.31    |214.52    |
 ----------------------------------------------------------------------------------------

|8         |1         |1         |1         |1         |196.03    |205.71    |202.65    |
 ----------------------------------------------------------------------------------------

Normalized Experiment (Fractional):

 ----------------------------------------------------------------------------------------

|N         |x0        |x1        |x2        |x3        |Yi1       |Yi2       |Yi3       |
 ----------------------------------------------------------------------------------------

|1         |1         |-1        |-1        |-1        |194.49    |208.69    |191.12    |
 ----------------------------------------------------------------------------------------

|4         |1         |-1        |1         |1         |215.39    |190.77    |203.98    |
 ----------------------------------------------------------------------------------------

|6         |1         |1         |-1        |1         |191.85    |202.35    |211.47    |
 ----------------------------------------------------------------------------------------

|7         |1         |1         |1         |-1        |195.38    |186.31    |214.52    |
 ----------------------------------------------------------------------------------------

According to Cochran's C-test homogeneity of variance is confirmed
```

Naturalized Experiment (Fractional):

N	x1	x2	x3	Yi1	Yi2	Yi3
1	-40	-35	20	194.49	208.69	191.12
4	-40	15	25	215.39	190.77	203.98
6	20	-35	25	191.85	202.35	211.47
7	20	15	20	195.38	186.31	214.52

Normalized equation:
y = 200.53 -0.21*x1 +0.53*x2 2.11*x3

Check:
200.53 +0.21 -0.53 -2.11 = 198.10
y = 198.10

200.53 +0.21 +0.53 +2.11 = 203.38
y = 203.38

200.53 -0.21 -0.53 +2.11 = 201.89
y = 201.89

200.53 -0.21 +0.53 -2.11 = 198.74
y = 198.74

Naturalized equation:
y = 181.70 -0.01*x1 +0.02*x2 0.84*x3

Check:
181.70 +0.28 -0.75 +16.86 = 198.10
y = 198.10

181.70 +0.28 +0.32 +21.07 = 203.38
y = 203.38

181.70 -0.14 -0.75 +21.07 = 201.89
y = 201.89

181.70 -0.14 +0.32 +16.86 = 198.74
y = 198.74

```
According to Student's t-test these coefficients are insignificant:
t = 0.0036653051009452237          t_cr = 2.3059000000000003          t < t_cr
b1 = -0.21 and a1 = -0.01
t = 0.0036653051009452237          t_cr = 2.3059000000000003          t < t_cr
b2 = 0.53 and a2 = 0.02
t = 0.0036653051009452237          t_cr = 2.3059000000000003          t < t_cr
b3 = 2.11 and a3 = 0.84

Then the equations change:
Normalized:
y = 200.53

Naturalized:
y = 181.70

F = 8.0         F_cr = 4.069           F > F_cr
According to Fisher's F-test model is not adequate to the original.

Process finished with exit code 0
```