МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Лабораторна робота 5

Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)

Гуменюк Святослав
4 курс
ІК-92

Викладач:
Кир'янов Артемій Юрійович

1 Лістинг програми

```
import random as rn
import statistics as stat
import math
import itertools
import numpy as np

x1min = -1
x1max = 2
x2min = -9
x2max = 6
x3min = -5
x3max = 8
xMeanMax = (x1max+x2max+x3max)/3.
xMeanMin = (x1min+x2min+x3min)/3.
ymin = 200 + int(xMeanMin)
ymax = 200 + int(xMeanMax)

def normalized_x(x):
    x0 = [0.5*(max(x[i])+min(x[i])) for i in range(len(x))]
    dx = [x0[i]-min(x[i]) for i in range(len(x))]
    return [[round((x[i][j]-x0[i])/dx[i],2) for j in range(len(x[i]))] \
    for i in range(len(x))]

def sum_of_mult(x):
    return sum([np.prod([*zip(*x)][i]) for i in range(len([*zip(*x)]))])

def B(x1,x2,x3,y):
    meanY = [stat.mean(y[i]) for i in range(len(y))]
    m00 = len(x1)
    m10 = sum(x1)
    m20 = sum(x2)
    m30 = sum(x3)
    m40 = np.dot(x1,x2)
    m50 = np.dot(x1,x3)
    m60 = np.dot(x2,x3)
    m70 = sum_of_mult([x1,x2,x3])
    m80 = sum_of_mult([x1,x1])
    m90 = sum_of_mult([x2,x2])
    m100 = sum_of_mult([x3,x3])

    m01 = m10
    m11 = np.dot(x1,x1)
    m21 = np.dot(x1,x2)
```

```
m31 = np.dot(x1,x3)
m41 = sum_of_mult([x1,x1,x2])
m51 = sum_of_mult([x1,x1,x2])
m61 = m70
m71 = sum_of_mult([x1,x1,x2,x3])
m81 = sum_of_mult([x1,x1,x1])
m91 = sum_of_mult([x1,x2,x2])
m101 = sum_of_mult([x1,x3,x3])

m02 = m20
m12 = m21
m22 = np.dot(x2,x2)
m32 = np.dot(x2,x3)
m42 = sum_of_mult([x1,x2,x2])
m52 = m70
m62 = sum_of_mult([x2,x2,x3])
m72 = sum_of_mult([x1,x2,x2,x3])
m82 = sum_of_mult([x1,x1,x2])
m92 = sum_of_mult([x2,x2,x2])
m102 = sum_of_mult([x2,x3,x3])

m03 = m30
m13 = m31
m23 = m32
m33 = np.dot(x3,x3)
m43 = m70
m53 = sum_of_mult([x1,x3,x3])
m63 = sum_of_mult([x2,x3,x3])
m73 = sum_of_mult([x1,x2,x3,x3])
m83 = sum_of_mult([x3,x1,x1])
m93 = sum_of_mult([x2,x3,x3])
m103 = sum_of_mult([x3,x3,x3])

m04 = m40
m14 = m41
m24 = m42
m34 = m43
m44 = sum_of_mult([x1,x1,x2,x2])
m54 = sum_of_mult([x1,x1,x2,x3])
m64 = sum_of_mult([x1,X2,x2,x3])
m74 = sum_of_mult([x1,x1,x2,x2,x3])
m84 = sum_of_mult([x1,x2,x1,x1])
m94 = sum_of_mult([x2,x2,x1,x2])
m104 = sum_of_mult([x3,x3,x1,x2])

m05 = m50
```

```
m15 = m51
m25 = m52
m35 = m53
m45 = m54
m55 = sum_of_mult([x1,x1,x3,x3])
m65 = sum_of_mult([x1,x2,x3,x3])
m75 = sum_of_mult([x1,x1,x2,x3,x3])
m85 = sum_of_mult([x1,x1,x1,x3])
m95 = sum_of_mult([x1,x3,x2,x2])
m105 = sum_of_mult([x1,x3,x3,x3])

m06 = m60
m16 = m61
m26 = m62
m36 = m63
m46 = m64
m56 = m65
m66 = sum_of_mult([x2,x2,x3,x3])
m76 = sum_of_mult([x1,x2,x2,x3,x3])
m86 = sum_of_mult([x2,x3,x1,x1])
m96 = sum_of_mult([x2,x3,x2,x2])
m106 = sum_of_mult([x2,x3,x3,x3])

m07 = m70
m17 = m71
m27 = m72
m37 = m73
m47 = m74
m57 = m75
m67 = m76
m77 = sum_of_mult([x1,x1,x2,x2,x3,x3])
m87 = sum_of_mult([x1,x1,x1,x2,x3])
m97 = sum_of_mult([x1,x2,x2,x2,x3])
m107 = sum_of_mult([x1,x2,x3,x3,x3])

m08 = m80
m18 = m81
m28 = m82
m38 = m83
m48 = m84
m58 = m85
m68 = m86
m78 = sum_of_mult([x1,x1,x2,x2,x3,x3])
m88 = sum_of_mult([x1,x1,x1,x1])
m98 = sum_of_mult([x1,x1,x2,x2])
m108 = sum_of_mult([x1,x1,x3,x3])
```

```
m09 = m90
m19 = m91
m29 = m92
m39 = m93
m49 = m94
m59 = m95
m69 = m96
m79 = sum_of_mult([x1,x1,x2,x2,x3,x3])
m89 = sum_of_mult([x1,x1,x2,x2])
m99 = sum_of_mult([x2,x2,x2,x2])
m109 = sum_of_mult([x2,x2,x3,x3])

m010 = m100
m110 = m101
m210 = m102
m310 = m103
m410 = m104
m510 = m105
m610 = m106
m710 = sum_of_mult([x1,x1,x2,x2,x3,x3])
m810 = sum_of_mult([x1,x1,x3,x3])
m910 = sum_of_mult([x2,x2,x3,x3])
m1010 = sum_of_mult([x3,x3,x3,x3])

M = np.array([[m00,m10,m20,m30,m40,m50,m60,m70,m80,m90,m100],\
[m01,m10,m21,m31,m41,m51,m61,m71,m81,m91,m101],\
[m02,m12,m22,m32,m42,m52,m62,m72,m82,m92,m102],\
[m03,m13,m23,m33,m43,m53,m63,m73,m83,m93,m103],\
[m04,m14,m24,m34,m44,m54,m64,m74,m84,m94,m104],\
[m05,m15,m25,m35,m45,m55,m65,m75,m85,m95,m105],\
[m06,m16,m26,m36,m46,m56,m66,m76,m86,m96,m106],\
[m07,m17,m27,m37,m47,m57,m67,m77,m87,m97,m107],\
[m08,m18,m28,m38,m48,m58,m68,m78,m88,m98,m108],\
[m09,m19,m29,m39,m49,m59,m69,m79,m89,m99,m109],\
[m010,m110,m210,m310,m410,m510,m610,m710,m810,m910,m1010]])

c = []
c.append(sum(meanY))
c.append(np.dot(x1,meanY))
c.append(np.dot(x2,meanY))
c.append(np.dot(x3,meanY))
c.append(sum_of_mult([meanY,x1,x2]))
c.append(sum_of_mult([meanY,x1,x3]))
c.append(sum_of_mult([meanY,x2,x3]))
c.append(sum_of_mult([meanY,x1,x2,x3]))
```

```
    c.append(sum_of_mult([meanY,x1,x1]))
    c.append(sum_of_mult([meanY,x2,x2]))
    c.append(sum_of_mult([meanY,x3,x3]))

    return np.linalg.inv(M).dot(c)

def A(x1,x2,x3,y):
    meanY = [stat.mean(y[i]) for i in range(len(y))]
    a0 = stat.mean(meanY)
    a1 = sum_of_mult([x1,meanY])/len(x1)
    a2 = sum_of_mult([x2,meanY])/len(x2)
    a3 = sum_of_mult([x3,meanY])/len(x3)
    a4 = sum_of_mult([x1,x2,meanY])/len(x1)
    a5 = sum_of_mult([x1,x3,meanY])/len(x1)
    a6 = sum_of_mult([x2,x3,meanY])/len(x3)
    a7 = sum_of_mult([x1,x2,x3,meanY])/len(x3)
    a8 = sum_of_mult([x1,x1,meanY])/len(x1)
    a9 = sum_of_mult([x2,x2,meanY])/len(x2)
    a10 = sum_of_mult([x3,x3,meanY])/len(x3)

    return [a0,a1,a2,a3,a4,a5,a5,a7,a8,a9,a10]

def kohren_criteria(y):
    s = [stat.pvariance(y[i]) for i in range(len(y))]
    return max(s)/sum(s)

def student_criteria(x,y, text=False):
    tCr = 2.120
    meanY = [stat.mean(y[i]) for i in range(len(y))]
    s = [stat.pvariance(y[i]) for i in range(len(y))]
    sb = stat.mean(s)
    sBetaS = sb/len(y[0])/len(s)
    sBetaS = math.sqrt(sBetaS)
    t0 = abs(sum_of_mult([meanY,x[0]])/len(x[0]))/sBetaS
    t1 = abs(sum_of_mult([meanY,x[1]])/len(x[1]))/sBetaS
    t2 = abs(sum_of_mult([meanY,x[2]])/len(x[2]))/sBetaS
    t3 = abs(sum_of_mult([meanY,x[3]])/len(x[3]))/sBetaS
    t4 = abs(sum_of_mult([meanY,x[1],x[2]])/len(x[1]))/sBetaS
    t5 = abs(sum_of_mult([meanY,x[1],x[3]])/len(x[1]))/sBetaS
    t6 = abs(sum_of_mult([meanY,x[2],x[3]])/len(x[2]))/sBetaS
    t7 = abs(sum_of_mult([meanY,x[1],x[2],x[3]])/len(x[1]))/sBetaS
    t8 = abs(sum_of_mult([meanY,x[1],x[1]])/len(x[1]))/sBetaS
    t9 = abs(sum_of_mult([meanY,x[2],x[2]])/len(x[2]))/sBetaS
    t10 = abs(sum_of_mult([meanY,x[3],x[3]])/len(x[3]))/sBetaS
    if text==True:
        print('Student criteria =', *[t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10])
```

```
    return [1 if t>tCr else 0 for t in [t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10]]

def fisher_criteria(y,B,x1,x2,x3,sign,text=False):
    d = 0
    for i in sign:
        d+=i
    meanY = [stat.mean(y[i]) for i in range(len(y))]
    newY = [B[0]*sign[0]+B[1]*sign[1]*x1[i]+B[2]*sign[2]*x2[i]\
    +B[3]*sign[3]*x3[i]+B[4]*sign[4]*x1[i]*x2[i]+B[5]*sign[5]*x1[i]*x3[i]\
    +B[6]*sign[6]*x2[i]*x3[i]+B[7]*sign[7]*x1[i]*x2[i]*x3[i]\
    +B[8]*sign[8]*x1[i]*x1[i]+B[9]*sign[9]*x2[i]*x2[i]\
    +B[10]*sign[10]*x3[i]*x3[i] for i in range(len(x1))]
    diff = [(newY[i]-meanY[i])**2 for i in range(len(newY))]

    try:
        Sad = len(y[0])/(len(y)-d)*sum(diff)
    except ZeroDivisionError:
        return False
    s = [stat.pvariance(y[i]) for i in range(len(y))]
    sb = stat.mean(s)
    if Sad/sb < 4.5:
        if text==True:
            print("Fisher criteria =",Sad/sb)
        return True
    else:
        return False


if __name__ == '__main__':
    xmin = [x1min, x2min, x3min]
    xmax = [x1max, x2max, x3max]
    x0 = [0.5*(xmin[i]+xmax[i]) for i in range(len(xmin))]
    dx = [x0[i]-xmin[i] for i in range(len(xmin))]

    X0 = [1 for k in range(15)]
    X1 = [x1min,x1min,x1min,x1min,x1max,x1max,x1max,x1max,\
    -1.73*dx[0]+x0[0],1.73*dx[0]+x0[0],x0[0],x0[0],x0[0],x0[0],x0[0]]
    X2 = [x2min,x2min,x2max,x2max,x2min,x2min,x2max,x2max,\
    x0[1],x0[1],-1.73*dx[1]+x0[1],1.73*dx[1]+x0[1],x0[1],x0[1],x0[1]]
    X3 = [x3min,x3max,x3min,x3max,x3min,x3max,x3min,x3max,\
    x0[2],x0[2],x0[2],x0[2],-1.73*dx[2]+x0[2],1.73*dx[2]+x0[2],x0[2]]
    XN = normalized_x([X1,X2,X3])

    Seed = 1
    while True:
        rn.seed(Seed)
```

```
    m = 3
    Gt = 0.3346
    while True:
        Yy = []
        for i in range(m):
            Yy.append([rn.choice(range(ymin,ymax)) for k in range(15)])
        if kohren_criteria([*zip(*Yy)])<Gt:
            break
        m+=1
    if fisher_criteria([*zip(*Yy)],B(X1,X2,X3,[*zip(*Yy)]),\
    XN[0],XN[1],XN[2],student_criteria([X0,*XN],[*zip(*Yy)])):
        break
    Seed+=1

print("Kohren criteria =",kohren_criteria([*zip(*Yy)]))
print("b_i =",*B(X1,X2,X3,[*zip(*Yy)]))
print("a_i =",*A(*XN,[*zip(*Yy)]))
print(student_criteria([X0,*XN],[*zip(*Yy)],True))
print(fisher_criteria([*zip(*Yy)],B(X1,X2,X3,[*zip(*Yy)]),\
XN[0],XN[1],XN[2],student_criteria([X0,*XN],[*zip(*Yy)]),True))

XN.append([XN[0][i]*XN[1][i] for i in range(len(X1))])
XN.append([XN[0][i]*XN[2][i] for i in range(len(X1))])
XN.append([XN[2][i]*XN[1][i] for i in range(len(X2))])
XN.append([XN[0][i]*XN[1][i]*XN[2][i] for i in range(len(X1))])
XN.append([XN[0][i]*XN[0][i] for i in range(len(X1))])
XN.append([XN[1][i]*XN[1][i] for i in range(len(X2))])
XN.append([XN[2][i]*XN[2][i] for i in range(len(X3))])

for row in zip(*(XN+Yy)):
    print(' & '.join(map(str,row)), end='\\\\\n')
```

2 Результат роботи програми

Нормована матриця планування:

	x_{N1}	x_{N2}	x_{N3}	$x_{N1} \cdot x_{N2}$	$x_{N1} \cdot x_{N3}$	$x_{N2} \cdot x_{N3}$	$x_{N1} \cdot x_{N2} \cdot x_{N3}$	x_{N1}^2	x_{N2}^2	x_{N3}^2	Y_1	Y_2	Y_3
1	-0.58	-0.58	-0.58	0.34	0.34	0.34	-0.19	0.34	0.34	0.34	198	198	195
2	-0.58	-0.58	0.58	0.34	-0.34	-0.34	0.19	0.34	0.34	0.34	204	202	199
3	-0.58	0.58	-0.58	-0.34	0.34	-0.34	0.19	0.34	0.34	0.34	203	203	195
4	-0.58	0.58	0.58	-0.34	-0.34	0.34	-0.19	0.34	0.34	0.34	197	203	199
5	0.58	-0.58	-0.58	-0.34	-0.34	0.34	0.19	0.34	0.34	0.34	200	202	202
6	0.58	-0.58	0.58	-0.34	0.34	-0.34	-0.19	0.34	0.34	0.34	204	201	204
7	0.58	0.58	-0.58	0.34	-0.34	-0.34	-0.19	0.34	0.34	0.34	202	197	201
8	0.58	0.58	0.58	0.34	0.34	0.34	0.19	0.34	0.34	0.34	204	198	201
9	-1.0	0.0	0.0	-0.0	-0.0	0.0	-0.0	1.0	0.0	0.0	196	197	201
10	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	204	203	204
11	0.0	-1.0	0.0	-0.0	0.0	-0.0	-0.0	0.0	1.0	0.0	195	201	202
12	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	202	195	197
13	0.0	0.0	-1.0	0.0	-0.0	-0.0	-0.0	0.0	0.0	1.0	199	196	200
14	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	203	197	196
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	198	204	195

Далі було знайдено нормовані коефіцієнти рівняння:

$$y = 199.87 - 3.01 \cdot x_{n1} + 0.11 \cdot x_{n2} + 0.21 \cdot x_{n3} - 0.21 \cdot x_{n1} \cdot x_{n2} - 0.23 \cdot x_{n1} \cdot x_{n3} - \\ -0.01 \cdot x_{n2} \cdot x_{n3} + 0.002 \cdot x_{n1} \cdot x_{n1} \cdot x_{n3} + 0.39 \cdot x_{n1}^2 + 0.01 \cdot x_{n2}^2 - 0.001 \cdot x_{n3}^2$$

Критерій Кохрена показав наступне значення:

$$G_p = 0.158$$

Вищезгадане значення менше за $G_T = 0.3346$, а отже дисперсія однорідна.

Критерій Стьюдента показав наступні значення:

$$t_0 = 548.2,\ t_1 = 1.7,\ t_2 = 0.5,\ t_3 = 0.7,\ t_4 = 0.3,\ t_5 = 0.1,\ t_6 = 0.4,\ t_7 = 0.2,\ t_8 = 172.1,\ t_9 = 171.3,\ t_{10} = 171.2$$

Значення $t_{1,2,3,4,5,6,7} < 2.306$, тому коефіцієнти рівняння регресії приймаємо незначними при рівні значимості 0.05

Таким чином рівняння регресії має вигляд:

$$y = 199.87 + 0.39 \cdot x_{n1}^2 + 0.01 \cdot x_{n2}^2 - 0.001 \cdot x_{n3}^2$$

Також було знайдено натуралызовані коефіцєнти рівнняння регресії:

$$y = 199.87 + 0.39 \cdot x_1^2 + 0.01 \cdot x_2^2 - 0.001 \cdot x_3^2$$

Останнім кроком була перевірка адекватносі моделі за допомогою критерію Фішера:

$$F_p = 2.23$$

Оскільки $F_p < 2.31$, отже рівняння регресії адекватно оригіналу при рівні значимості 0.05

3 Висновки

В ході даної лабораторної роботи було проведено трьохфакторний експеримент з використанням рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план). Було перевірено однорідність десперсії за критерієм Кохрена. Після цього було знайдено натуралізовані коефіцієнти, та визначено значимість коефіцієнтів за допомогою критерію Стьюдента, який показав, що один з коефіцієнтів є незначним. Адекватность рівняння оригіналу було перевірено за допомогою критерію Фішера, який показав, що рівнняння є адекватним оригіналу.