Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Методи оптимізації та планування експерименту

Лабораторна робота №5

**«**Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

(центральний ортогональний композиційний план)**»**

Виконав:

студент групи ІВ-83

Головенець Р.С.

Перевірив:

ас. Регіда П.Г.

Київ-2020 р.

**Лістинг програми**

**import** random, math, numpy  
**import** scipy.stats  
x1\_min,x1\_max,x2\_min,x2\_max,x3\_min,x3\_max,l=-1,2,-9,5,-10,7,1.215 *#306*y\_min = 194  
y\_max = 204  
x01 = (x1\_max + x1\_min) / 2  
xl1 = l \* (x1\_max - x01) + x01  
xl11 = -l \* (x1\_max - x01) + x01  
x02 = (x2\_max + x2\_min) / 2  
xl2 = l \* (x2\_max - x02) + x02  
xl22 = -l \* (x2\_max - x02) + x02  
x03 = (x3\_max + x3\_min) / 2  
xl3 = l \* (x3\_max - x03) + x03  
xl33 = -l \* (x3\_max - x03) + x03  
delta\_x1 = x1\_max - x01  
delta\_x2 = x2\_max - x02  
delta\_x3 = x3\_max - x03  
  
Xf = [[-1.0, -1.0, -1.0, 1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0], *# 1* [-1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0], *# 2* [-1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0], *# 3* [-1.0, 1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0], *# 4* [1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0], *# 5* [1.0, -1.0, 1.0, -1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0], *# 6* [1.0, 1.0, -1.0, 1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0], *# 7* [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0], *# 8* [-1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0, 0], *# 9* [1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0, 0], *# 10* [0, -1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0], *# 11* [0, 1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0], *# 12* [0, 0, -1.215, 0, 0, 0, 0, 0, 0, 1.47623], *# 13* [0, 0, 1.215, 0, 0, 0, 0, 0, 0, 1.47623], *# 14* [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]] *# 15*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}  
Gt2 = [(range(11, 17), 0.4366), (range(17, 37), 0.3720), (range(37, 145), 0.3093)]  
**def** get\_b(n, lmaty):  
 xnat = [[-1.0, -1.0, -1.0], *# 1* [-1.0, -1.0, 1.0], *# 2* [-1.0, 1.0, -1.0], *# 3* [-1.0, 1.0, 1.0], *# 4* [1.0, -1.0, -1.0], *# 5* [1.0, -1.0, 1.0], *# 6* [1.0, 1.0, -1.0], *# 7* [1.0, 1.0, 1.0], *# 8* [-1.215, 0, 0], *# 9* [1.215, 0, 0], *# 10* [0, -1.215, 0], *# 11* [0, 1.215, 0], *# 12* [0, 0, -1.215], *# 13* [0, 0, 1.215], *# 14* [0, 0, 0]] *# 15* xnat=[[ x1\_min , x2\_min , x3\_min ] ,  
 [ x1\_min , x2\_min , x3\_max ] ,  
 [ x1\_min , x2\_max , x3\_min ] ,  
 [ x1\_min , x2\_max , x3\_max ] ,  
 [ x1\_max , x2\_min , x3\_min ] ,  
 [ x1\_max , x2\_min , x3\_max ] ,  
 [ x1\_max , x2\_max , x3\_min ] ,  
 [ x1\_max , x2\_max , x3\_max ] ,  
 [xl11,x02,x03],  
 [xl1,x02,x03],  
 [x01,xl22,x03],  
 [x01,xl2,x03],  
 [x01,x02,xl33],  
 [x01,x02,xl3],  
 [x01,x02,x03],]  
 print(**"X: "**)  
 **for** i **in** range(len(xnat)):  
 print(xnat[i])  
 xnm = [[xnat[i][j] **for** i **in** range(15)] **for** j **in** range(3)]  
 a00 = [[],  
 [xnm[0]], [xnm[1]], [xnm[2]],  
 [xnm[0], xnm[1]],  
 [xnm[0], xnm[2]],  
 [xnm[1], xnm[2]],  
 [xnm[0], xnm[1], xnm[2]],  
 [xnm[0], xnm[0]],  
 [xnm[1], xnm[1]],  
 [xnm[2], xnm[2]]]  
 **def** calcxi(n, listx):  
 sumxi = 0  
 **for** i **in** range(n):  
 lsumxi = 1  
 **for** j **in** range(len(listx)):  
 lsumxi \*= listx[j][i]  
 sumxi += lsumxi  
 **return** sumxi  
 a0 = [15]  
 **for** i **in** range(10):  
 a0.append(calcxi(n, a00[i + 1]))  
 a1 = [calcxi(n, a00[i] + a00[1]) **for** i **in** range(len(a00))]  
 a2 = [calcxi(n, a00[i] + a00[2]) **for** i **in** range(len(a00))]  
 a3 = [calcxi(n, a00[i] + a00[3]) **for** i **in** range(len(a00))]  
 a4 = [calcxi(n, a00[i] + a00[4]) **for** i **in** range(len(a00))]  
 a5 = [calcxi(n, a00[i] + a00[5]) **for** i **in** range(len(a00))]  
 a6 = [calcxi(n, a00[i] + a00[6]) **for** i **in** range(len(a00))]  
 a7 = [calcxi(n, a00[i] + a00[7]) **for** i **in** range(len(a00))]  
 a8 = [calcxi(n, a00[i] + a00[8]) **for** i **in** range(len(a00))]  
 a9 = [calcxi(n, a00[i] + a00[9]) **for** i **in** range(len(a00))]  
 a10 = [calcxi(n, a00[i] + a00[10]) **for** i **in** range(len(a00))]  
 a = numpy.array([[a0[0], a0[1], a0[2], a0[3], a0[4], a0[5],a0[6], a0[7], a0[8], a0[9], a0[10]],  
 [a1[0], a1[1], a1[2], a1[3], a1[4], a1[5],a1[6], a1[7], a1[8], a1[9], a1[10]],  
 [a2[0], a2[1], a2[2], a2[3], a2[4], a2[5],a2[6], a2[7], a2[8], a2[9], a2[10]],  
 [a3[0], a3[1], a3[2], a3[3], a3[4], a3[5],a3[6], a3[7], a3[8], a3[9], a3[10]],  
 [a4[0], a4[1], a4[2], a4[3], a4[4], a4[5],a4[6], a4[7], a4[8], a4[9], a4[10]],  
 [a5[0], a5[1], a5[2], a5[3], a5[4], a5[5],a5[6], a5[7], a5[8], a5[9], a5[10]],  
 [a6[0], a6[1], a6[2], a6[3], a6[4], a6[5],a6[6], a6[7], a6[8], a6[9], a6[10]],  
 [a7[0], a7[1], a7[2], a7[3], a7[4], a7[5],a7[6], a7[7], a7[8], a7[9], a7[10]],  
 [a8[0], a8[1], a8[2], a8[3], a8[4], a8[5],a8[6], a8[7], a8[8], a8[9], a8[10]],  
 [a9[0], a9[1], a9[2], a9[3], a9[4], a9[5],a9[6], a9[7], a9[8], a9[9], a9[10]],  
 [a10[0], a10[1], a10[2], a10[3], a10[4], a10[5],a10[6], a10[7], a10[8], a10[9], a10[10]]])  
 c0 = [calcxi(n, [lmaty])]  
 **for** i **in** range(len(a00) - 1):  
 c0.append(calcxi(n, a00[i + 1] + [lmaty]))  
 c = numpy.array([c0[0], c0[1], c0[2], c0[3], c0[4], c0[5],  
 c0[6], c0[7], c0[8], c0[9], c0[10]])  
 b = numpy.linalg.solve(a, c)  
 **return** b  
**def** func(num):  
 N = 15  
 m = num  
 Y = [[random.randint(y\_min, y\_max) **for** y **in** range(m)] **for** x **in** range(N)]  
 print(**"Y: "**)  
 **for** i **in** range(len(Y)):  
 print(Y[i])  
 Ys = [sum(Y[i]) / m **for** i **in** range(N)]  
 b\_arr = get\_b(N, Ys)  
 b0 = b\_arr[0]  
 b1 = b\_arr[1]  
 b2 = b\_arr[2]  
 b3 = b\_arr[3]  
 b12 = b\_arr[4]  
 b13 = b\_arr[5]  
 b23 = b\_arr[6]  
 b123 = b\_arr[7]  
 b11 = b\_arr[8]  
 b22 = b\_arr[9]  
 b33 = b\_arr[10]  
 print(**"Y={} + {}\*x1 + {}\*x2 + {}\*x3 +{}\*x1x2 + {}\*x1x3 + {}\*x2x3 + {}\*x1x2x3 \n"  
 "+ {}\*x1^2 + {}\*x2^2+ {}\*x3^2"**.format(round(b0,3), round(b1,3),round(b2,3), round(b3,3), round(b12,3),  
 round(b13,3),round(b23,3), round(b123,3), round(b11,3),  
 round(b22,3), round(b33,3)))  
 print(**"Перевірка: "**)  
 z = []  
 s = 0  
 **for** i **in** range(len(Xf)):  
 **for** j **in** range(10):  
 s += b\_arr[1:][j] \* Xf[i][j]  
 z.append(s)  
 print(b0 + z[i], **"=="**, Ys[i])  
 print(**"Результат збігається з середніми значеннями"**)  
 print(**"Критерій Кохрена"**)  
 D = []  
 Summa = 0  
 **for** i **in** range(N):  
 **for** j **in** range(m):  
 Summa += pow((Y[i][j] - Ys[i]), 2)  
 D.append(1 / m \* Summa)  
 Summa = 0  
 Gp = max(D) / sum(D)  
 print(**"Gp= "**, Gp)  
 f1 = m - 1  
 f2 = N  
 q = 0.05  
 **if** m >= 11:  
 **for** i **in** range(len(Gt2)):  
 **if** m **in** Gt2[i][0]:  
 crit = Gt2[i][1]  
 **break  
 else**:  
 crit = Gt[f1]  
 **if** Gp <= crit:  
 print(**"Дисперсія однорідна"**)  
 print(Gp, **"<="**, crit)  
 **else**:  
 print(**"Дисперсія не однорідна"**)  
 m += 1  
 print(**"M:"**, m)  
 **return** func(m)  
 print(**"Критерій Стьюдента"**)  
 S2\_b = sum(D) / N  
 S2\_betta = S2\_b / (N \* m)  
 S\_betta = math.sqrt(S2\_betta)  
 Xs = [[1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0], *# 1* [1.0, -1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0], *# 2* [1.0, -1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0], *# 3* [1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0], *# 4* [1.0, 1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0], *# 5* [1.0, 1.0, -1.0, 1.0, -1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0], *# 6* [1.0, 1.0, 1.0, -1.0, 1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0], *# 7* [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0], *# 8* [1.0, -1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0, 0], *# 9* [1.0, 1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0, 0], *# 10* [1.0, 0, -1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0], *# 11* [1.0, 0, 1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0], *# 12* [1.0, 0, 0, -1.215, 0, 0, 0, 0, 0, 0, 1.47623], *# 13* [1.0, 0, 0, 1.215, 0, 0, 0, 0, 0, 0, 1.47623], *# 14* [1.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]] *# 15* betta = []  
 **for** i **in** range(N):  
 s = 0  
 **for** j **in** range(11):  
 s += Ys[i] \* Xs[i][j]  
 betta.append(s / N)  
 t = []  
 **for** i **in** range(len(betta)):  
 t.append(abs(betta[i]) / S\_betta)  
 f3 = f1 \* f2  
 print(**"f3="**, f3)  
 t\_tabl = scipy.stats.t.ppf((1 + (1 - q)) / 2, f3)  
 **if** t[i] < t\_tabl:  
 b\_arr[i] = 0  
 print(t[i], **"<"**, t\_tabl)  
 y = []  
 **for** i **in** range(len(z)):  
 y.append(b0 + z[i])  
  
 **for** i **in** range(len(y)):  
 print(y[i], **"=="**, Ys[i])  
 print(**"Нуль гіпотеза виконується"**)  
 print(**"Критерій Фішера"**)  
 d = 0  
 **for** i **in** range(len(b\_arr)):  
 **if** b\_arr[i] != 0:  
 d += 1  
 print(**"d="**, d)  
 f4 = N - d  
 Sum = 0  
 **for** i **in** range(len(y)):  
 Sum += pow((y[i] - Ys[i]), 2)  
 S\_ad = (m / (N - d)) \* Sum  
 Fp = S\_ad / S2\_b  
 print(**"Fp= {0} \n"  
 "f3= {1} \n"  
 "f4= {2}"**.format(Fp, f3, f4))  
 Ft = scipy.stats.f.ppf(1 - q, f4, f3)  
 print(**"Значення критерію Ft -"**,Ft)  
 **while True**:  
 **if** (Fp < Ft):  
 print(**"Рівняння регресії адекватно оригіналу при рівні значимості 0.05"**)  
 **break  
 else**:  
 print(**"Рівняння регресії неадекватно оригіналу при рівні значимості 0.05"**)  
 **return** (func(m))  
func(3)

**Результати роботи програми:**





