­­­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-81

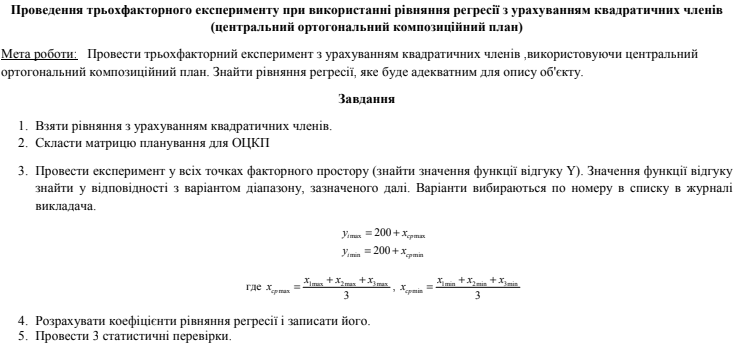
Богданенко В.Ю.

Перевірив:

ас. Регіда П. Г.

Київ

2020 р.



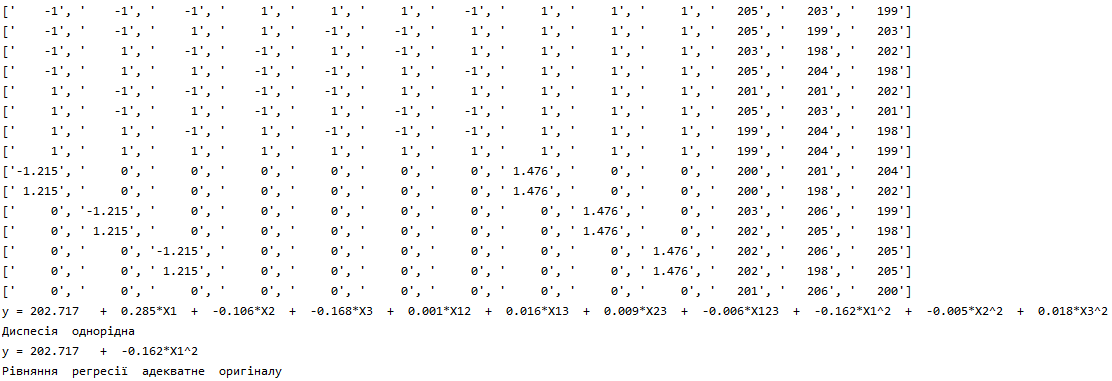
**Варіант завдання:**



**Лістинг програми:**

**import** numpy **as** np  
**from** random **import** randint  
**from** copy **import** deepcopy  
**from** math **import** sqrt  
**def** super\_turbo\_avto\_fill\_matrix(a):  
 **for** i **in** range(len(a)):  
 a[i].append(a[i][0] \* a[i][1])  
 a[i].append(a[i][0] \* a[i][2])  
 a[i].append(a[i][1] \* a[i][2])  
 a[i].append(a[i][0] \* a[i][1] \* a[i][2])  
 a[i].append(a[i][0] \*\* 2)  
 a[i].append(a[i][1] \*\* 2)  
 a[i].append(a[i][2] \*\* 2)  
 **return** a  
**def** anreal\_quick\_fill\_matrix(a, x):  
 a1 = []  
 **for** i **in** range(len(a)):  
 a1.append([])  
 **for** j **in** range(3):  
 a1[i].append(0)  
 **for** i **in** range(len(a)):  
 **for** j **in** range(3):  
 **if** a[i][j] == -1:  
 a1[i][j] = (min(x[j]))  
 **elif** a[i][j] == 1:  
 a1[i][j] = (max(x[j]))  
 **else**:  
 a1[i][j] = (x[j][0] + x[j][1]) / 2 + a[i][j] \* (x[j][1] - ((x[j][0] + x[j][1]) / 2))  
 super\_turbo\_avto\_fill\_matrix(a1)  
 **return** a1  
x1min = -1  
x1max = 4  
x2min = -3  
x2max = 6  
x3min = -1  
x3max = 9  
xmatrix = [[x1min, x1max], [x2min, x2max], [x3min, x3max]]  
ymax = 200 + (x3max + x2max + x1max) / 3  
ymin = 200 + (x3min + x2min + x1min) / 3  
matrixplan = [[-1, -1, -1],  
 [-1, -1, 1],  
 [-1, 1, -1],  
 [-1, 1, 1],  
 [1, -1, -1],  
 [1, -1, 1],  
 [1, 1, -1],  
 [1, 1, 1],  
 [-1.215, 0, 0],  
 [1.215, 0, 0],  
 [0, -1.215, 0],  
 [0, 1.215, 0],  
 [0, 0, -1.215],  
 [0, 0, 1.215],  
 [0, 0, 0]]  
matrixplan = super\_turbo\_avto\_fill\_matrix(matrixplan)  
matrixnatural = anreal\_quick\_fill\_matrix(matrixplan, xmatrix)  
  
**def** lab(m, good\_plan, natural, ymax, ymin):  
 **if** m - 1 == 2:  
 Gt = 0.3346  
 **elif** m - 1 == 3:  
 Gt = 0.2758  
 **elif** m - 1 == 4:  
 Gt = 0.2419  
 **elif** m - 1 == 5:  
 Gt = 0.2159  
 **elif** m - 1 == 6:  
 Gt = 0.2034  
 **elif** m - 1 == 7:  
 Gt = 0.1911  
 **elif** m - 1 == 8:  
 Gt = 0.1815  
 **elif** m - 1 == 9:  
 Gt = 0.1736  
 **elif** (((m - 1) > 10) **and** ((m - 1) < 16)):  
 Gt = 0.1671  
 **elif** (((m - 1) > 16) **and** ((m - 1) < 36)):  
 Gt = 0.1429  
 **elif** (((m - 1) > 36) **and** ((m - 1) < 144)):  
 Gt = 0.1144  
 **elif** (m - 1) > 144:  
 Gt = 0.0889  
 **for** j **in** range(len(good\_plan)):  
 **for** i **in** range(len(good\_plan[14]), m + 10):  
 natural[j].append(randint(int(ymin), int(ymax)))  
 good\_plan[j].append(randint(int(ymin), int(ymax)))  
 ysplist = []  
 **for** i **in** range(len(good\_plan)):  
 ysp = 0  
 **for** j **in** range(10, len(good\_plan[0])):  
 ysp = ysp + good\_plan[i][j]  
 ysp = ysp / m  
 ysplist.append(ysp)  
 S2ylist = []  
 S2ysum = 0  
 **for** i **in** range(len(good\_plan)):  
 S2y = 0  
 **for** j **in** range(10, len(good\_plan[0])):  
 S2y = S2y + (good\_plan[i][j] - ysplist[i]) \*\* 2  
 S2y = S2y / m  
 S2ylist.append(S2y)  
 S2ysum = S2ysum + S2y  
 Gp = max(S2ylist) / S2ysum  
 **if** Gp > Gt:  
 m = m + 1  
 lab((m, good\_plan, natural, ymax, ymin))  
 **else**:  
 deepcool\_natural = deepcopy(natural)  
 **for** i **in** range(len(deepcool\_natural)):  
 deepcool\_natural[i].insert(0, 1)  
 rl = []  
 **for** z **in** range(11):  
 k0l = []  
 **for** u **in** range(11):  
 k0 = 0  
 **for** i **in** range(15):  
 k0 = k0 + deepcool\_natural[i][z] \* deepcool\_natural[i][u]  
 k0 = k0  
 k0l.append(k0)  
 rl.append(k0l)  
 det0 = np.linalg.det(rl)  
 yklist = []  
 **for** j **in** range(11):  
 yk = 0  
 **for** i **in** range(15):  
 yk = yk + ysplist[i] \* deepcool\_natural[i][j]  
 yklist.append(yk)  
 detlist = []  
 **for** j **in** range(11):  
 v = deepcopy(rl)  
 **for** i **in** range(11):  
 v[i][j] = yklist[i]  
 detlist.append(np.linalg.det(v))  
 blist = []  
 **for** i **in** range(len(detlist)):  
 blist.append(detlist[i] / det0)  
 S2B = S2ysum / 15  
 S2b = S2B / (15 \* m)  
 Sb = sqrt(S2b)  
 very\_good\_plan = deepcopy(good\_plan)  
 **for** i **in** range(len(very\_good\_plan)):  
 very\_good\_plan[i].insert(0, 1)  
 rl = []  
 **for** z **in** range(11):  
 k0l = []  
 **for** u **in** range(11):  
 k0 = 0  
 **for** i **in** range(15):  
 k0 = k0 + very\_good\_plan[i][z] \* very\_good\_plan[i][u]  
 k0 = k0  
 k0l.append(k0)  
 rl.append(k0l)  
 det0 = np.linalg.det(rl)  
 yklist = []  
 **for** j **in** range(11):  
 yk = 0  
 **for** i **in** range(15):  
 yk = yk + ysplist[i] \* very\_good\_plan[i][j]  
 yklist.append(yk)  
 detlist = []  
 **for** j **in** range(11):  
 v = deepcopy(rl)  
 **for** i **in** range(11):  
 v[i][j] = yklist[i]  
 detlist.append(np.linalg.det(v))  
 tlist = []  
 **for** i **in** range(len(detlist)):  
 tlist.append(abs(detlist[i] / det0) / Sb)  
 sumt = 0  
 bultlist = []  
 **for** i **in** range(len(tlist)):  
 **if** tlist[i] >= 2.042:  
 bultlist.append(1)  
 sumt = sumt + 1  
 **elif** tlist[i] < 2.042:  
 bultlist.append(0)  
 ynewlist = []  
 **for** j **in** range(15):  
 ynew = 0  
 **for** i **in** range(11):  
 **if** bultlist[i] == 1:  
 ynew = ynew + blist[i] \* deepcool\_natural[j][i]  
 ynewlist.append(ynew)  
 **if** (((m - 1) \* 15 >= 30) **and** ((m - 1) \* 15 < 40)):  
 **if** (15 - sumt) == 1:  
 Ft = 4.2  
 **elif** (15 - sumt) == 2:  
 Ft = 3.3  
 **elif** (15 - sumt) == 3:  
 Ft = 2.9  
 **elif** (15 - sumt) == 4:  
 Ft = 2.7  
 **elif** (15 - sumt) == 5:  
 Ft = 2.5  
 **elif** ((15 - sumt) >= 6 **and** (15 - sumt) < 12):  
 Ft = 2.4  
 **elif** ((15 - sumt) >= 12 **and** (15 - sumt) < 24):  
 Ft = 2.1  
 **elif** ((15 - sumt) >= 24):  
 Ft = 1.9  
 **elif** (((m - 1) \* 15 >= 40) **and** ((m - 1) \* 15 < 60)):  
 **if** (15 - sumt) == 1:  
 Ft = 4.1  
 **elif** (15 - sumt) == 2:  
 Ft = 3.2  
 **elif** (15 - sumt) == 3:  
 Ft = 2.9  
 **elif** (15 - sumt) == 4:  
 Ft = 2.6  
 **elif** (15 - sumt) == 5:  
 Ft = 2.5  
 **elif** ((15 - sumt) >= 6 **and** (15 - sumt) < 12):  
 Ft = 2.3  
 **elif** ((15 - sumt) >= 12 **and** (15 - sumt) < 24):  
 Ft = 2.0  
 **elif** ((15 - sumt) >= 24):  
 Ft = 1.8  
 **elif** (((m - 1) \* 15 >= 60) **and** ((m - 1) \* 15 < 120)):  
 **if** (15 - sumt) == 1:  
 Ft = 4.0  
 **elif** (15 - sumt) == 2:  
 Ft = 3.2  
 **elif** (15 - sumt) == 3:  
 Ft = 2.8  
 **elif** (15 - sumt) == 4:  
 Ft = 2.5  
 **elif** (15 - sumt) == 5:  
 Ft = 2.4  
 **elif** ((15 - sumt) >= 6 **and** (15 - sumt) < 12):  
 Ft = 2.3  
 **elif** ((15 - sumt) >= 12 **and** (15 - sumt) < 24):  
 Ft = 1.9  
 **elif** ((15 - sumt) >= 24):  
 Ft = 1.7  
 **elif** (((m - 1) \* 15 >= 120)):  
 **if** (15 - sumt) == 1:  
 Ft = 3.9  
 **elif** (15 - sumt) == 2:  
 Ft = 3.1  
 **elif** (15 - sumt) == 3:  
 Ft = 2.7  
 **elif** (15 - sumt) == 4:  
 Ft = 2.5  
 **elif** (15 - sumt) == 5:  
 Ft = 2.3  
 **elif** ((15 - sumt) >= 6 **and** (15 - sumt) < 12):  
 Ft = 2.2  
 **elif** ((15 - sumt) >= 12 **and** (15 - sumt) < 24):  
 Ft = 1.8  
 **elif** ((15 - sumt) >= 24):  
 Ft = 1.6  
 Sad = 0  
 **for** i **in** range(15):  
 Sad = Sad + ((ynewlist[i] - ysplist[i]) \*\* 2) \* m / (15 - sumt)  
 Fp = Sad / S2B  
 **for** i **in** range(len(good\_plan)):  
 **for** j **in** range(len(good\_plan[i])):  
 **if** type(good\_plan[i][j]) == float:  
 **if** good\_plan[i][j] != 0:  
 good\_plan[i][j] = **'%.3f'** % good\_plan[i][j]  
 **if** (good\_plan[i][j] == 0.0 **or** good\_plan[i][j] == -0.0):  
 good\_plan[i][j] = 0  
 good\_plan[i][j] = (**'%+6s'** % good\_plan[i][j])  
 print(good\_plan[i])  
 xlist = [**" "**, **"\*X1"**, **"\*X2"**, **"\*X3"**, **"\*X12"**, **"\*X13"**, **"\*X23"**, **"\*X123"**, **"\*X1^2"**, **"\*X2^2"**, **"\*X3^2"**]  
 text3 = **"y = "** blist1 = [str(**'%.3f'** % blist[0]), **" + "** + str(**'%.3f'** % blist[1]), **" + "** +  
 str(**'%.3f'** % blist[2]), **" + "** + str(**'%.3f'** % blist[3]),  
 **" + "** + str(**'%.3f'** % blist[4]), **" + "** + str(**'%.3f'** % blist[5]), **" + "** + str(**'%.3f'** % blist[6]),  
 **" + "** + str(**'%.3f'** % blist[7]), **" + "** + str(**'%.3f'** % blist[8]), **" + "** +  
 str(**'%.3f'** % blist[9]), **" + "** + str(**'%.3f'** % blist[10]), ]  
 **for** i **in** range(len(xlist)):  
 text3 = text3 + (blist1[i]) + xlist[i]  
 text4 = **"y = "  
 for** i **in** range(len(xlist)):  
 **if** bultlist[i] == 1:  
 text4 = text4 + (blist1[i]) + xlist[i]  
 print(text3)  
 print(**"Диспесія однорідна"**)  
 print(text4)  
 **if** Fp < Ft:  
 print(**"Рівняння регресії адекватне оригіналу"**)  
 **elif** Fp > Ft:  
 print(**"Рівняння регресії неадекватне оригіналу"**)  
m = 3  
lab(m, matrixplan, matrixnatural, ymax, ymin)

**Результат виконання роботи:**



­­­