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

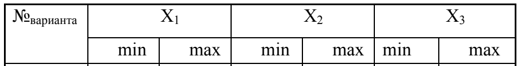
Виконала Дуплій Наталя Володимирівна

Факультет ІОТ,

Група ІВ-82

Залікова книжка № ІВ-8209

Київ – 2020 р.





Складаємо матрицю планування для дробового трьохфакторного експерименту з використанням додаткового (фіктивного) нульового фактору () та заповнюємо нормованими значеннями.

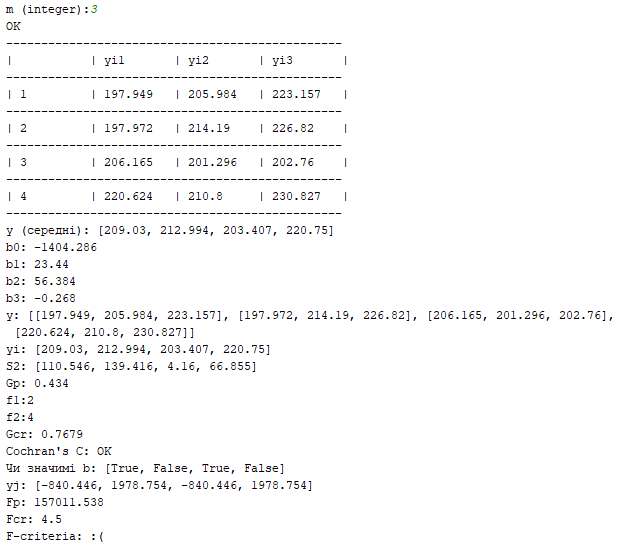
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 1 | +1 | -1 | -1 | -1 |  |  |  |  |
| 2 | +1 | -1 | +1 | +1 |  |  |  |  |
| 3 | +1 | +1 | -1 | +1 |  |  |  |  |
| 4 | +1 | +1 | +1 | -1 |  |  |  |  |

Таблиця з натуральних значень

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| 1 | -30 | 10 | 10 |  |  |  |  |
| 2 | -30 | 60 | 35 |  |  |  |  |
| 3 | 0 | 10 | 35 |  |  |  |  |
| 4 | 0 | 60 | 10 |  |  |  |  |

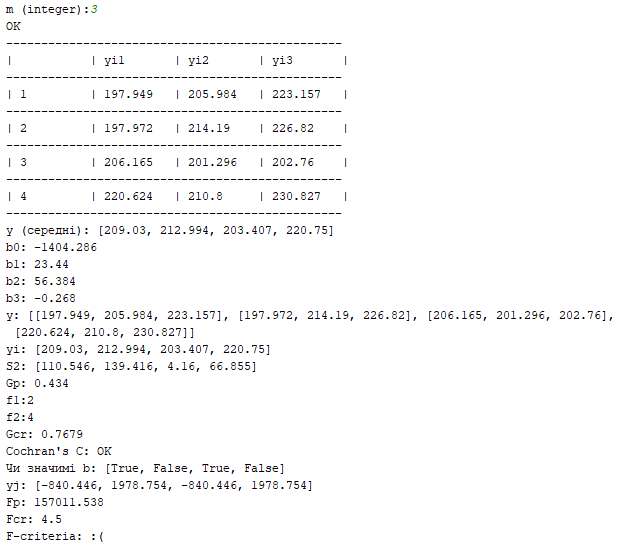
Проводимо експеримент в усіх точках плану.

Беремо m=5



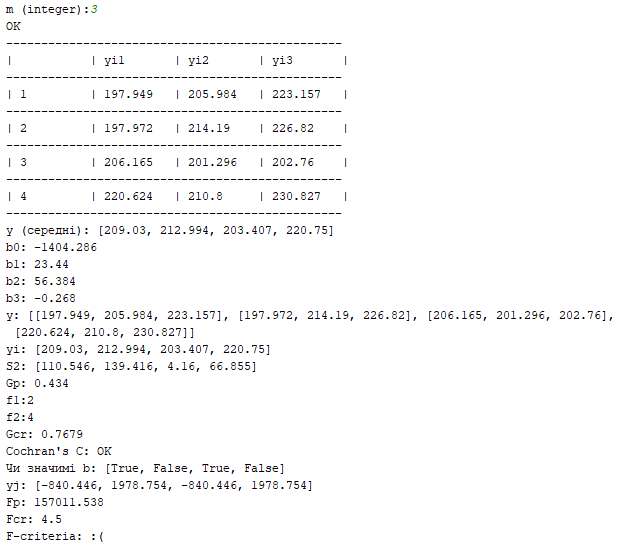
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 1 | -30 | 10 | 10 | 197.949 | 205.984 | 223.157 |
| 2 | -30 | 60 | 35 | 197.972 | 214.19 | 226.82 |
| 3 | 0 | 10 | 35 | 206.165 | 201.296 | 202.76 |
| 4 | 0 | 60 | 10 | 220.624 | 210.8 | 230.827 |

Середні значення функції відгуку та коефіцієнти рівняння регресії знайдені за допомогою програми.



Рівняння регресії:

Проводимо статистичні перевірки.



Дисперсія однорідна за критерієм Кохрена. Коефіцієнти b0 і b2 значимі, b1 і b3 незначимі за критерієм Стьюдента. Модель неадекватна за критерієм Фішера.

# Лістинг програми

**import** random  
**import** math  
**import** numpy  
**import** copy  
Gcr = {1:  
 {2: 9985, 3: 9669, 4: 9065,  
 5: 8412, 6: 7808, 7: 7271,  
 8: 6798, 9: 6385, 10: 6020},  
 2:  
 {2: 9750, 3: 8709, 4: 7679,  
 5: 6838, 6: 6161, 7: 5612,  
 8: 5157, 9: 4775, 10: 4450},  
 3:  
 {2: 9392, 3: 7977, 4: 6841,  
 5: 5981, 6: 5321, 7: 4800,  
 8: 4377, 9: 4027, 10: 3733},  
 4:  
 {2: 9057, 3: 7457, 4: 6287,  
 5: 5440, 6: 4803, 7: 4307,  
 8: 3910, 9: 3584, 10: 3311},  
 5:  
 {2: 8772, 3: 7071, 4: 5892,  
 5: 5063, 6: 4447, 7: 3974,  
 8: 3595, 9: 3286, 10: 3029},  
 6:  
 {2: 8534, 3: 6771, 4: 5598,  
 5: 4783, 6: 4184, 7: 3726,  
 8: 3362, 9: 3067, 10: 2823},  
 7:  
 {2: 8332, 3: 6530, 4: 5365,  
 5: 4564, 6: 3980, 7: 3535,  
 8: 3185, 9: 2901, 10: 2666},  
 8:  
 {2: 8159, 3: 6333, 4: 5175,  
 5: 4387, 6: 3817, 7: 3384,  
 8: 3043, 9: 2768, 10: 2541},  
 9:  
 {2: 8010, 3: 6167, 4: 5017,  
 5: 4241, 6: 3682, 7: 3259,  
 8: 2926, 9: 2659, 10: 2439},  
 10:  
 {2: 7880, 3: 6025, 4: 4884,  
 5: 4118, 6: 3568, 7: 3154,  
 8: 2829, 9: 2568, 10: 2353}}  
  
**def** get\_cochran\_c(f1, f2):  
 **global** Gcr  
 gcr = Gcr.get(f1)  
 gcr = gcr.get(f2)  
 gcr = round(gcr / 10000, 4)  
 **return** gcr  
Fcr = {1:  
 {1: 164.4, 2: 18.5, 3: 10.1, 4: 7.7,  
 5: 6.6, 6: 6.0, 7: 5.5, 8: 5.3,  
 9: 5.1, 10: 5.0, 11: 4.8, 12: 4.8,  
 13: 4.7, 14: 4.6, 15: 4.5, 16: 4.5},  
 2:  
 {1: 199.5, 2: 19.2, 3: 9.6, 4: 6.9,  
 5: 5.8, 6: 5.1, 7: 4.7, 8: 4.5,  
 9: 4.3, 10: 4.1, 11: 4.0, 12: 3.9,  
 13: 3.8, 14: 3.7, 15: 3.7, 16: 3.6,  
 17: 3.6, 18: 3.6, 19: 3.5, 20: 3.5,  
 22: 3.4, 24: 3.4, 26: 3.4, 28: 3.3,  
 30: 3.3, 40: 3.2, 60: 3.2, 120: 3.1},  
 3:  
 {1: 215.7, 2: 19.2, 3: 9.3, 4: 6.6,  
 5: 5.4, 6: 4.8, 7: 4.4, 8: 4.1,  
 9: 3.9, 10: 3.7, 11: 3.6, 12: 3.5,  
 13: 3.4, 14: 3.3, 15: 3.3, 16: 3.2,  
 17: 3.2, 18: 3.2, 19: 3.1, 20: 2.1,  
 22: 3.1, 24: 3.0, 26: 3.0, 28: 3.0,  
 30: 2.9, 40: 2.9, 60: 2.8, 120: 2.7},  
 4:  
 {1: 224.6, 2: 19.3, 3: 9.1, 4: 6.4,  
 5: 5.2, 6: 4.5, 7: 4.1, 8: 3.8,  
 9: 3.6, 10: 3.5, 11: 3.4, 12: 3.3,  
 13: 3.2, 14: 3.1, 15: 2.1, 16: 3.0},  
 5:  
 {1: 230.2, 2: 19.3, 3: 9.0, 4: 6.3,  
 5: 5.1, 6: 4.4, 7: 4.0, 8: 3.7,  
 9: 3.5, 10: 3.3, 11: 3.2, 12: 3.1,  
 13: 3.0, 14: 3.0, 15: 2.9, 16: 2.9},  
 6:  
 {1: 234.0, 2: 19.3, 3: 8.9, 4: 6.2,  
 5: 5.0, 6: 4.3, 7: 3.9, 8: 3.6,  
 9: 3.4, 10: 3.2, 11: 3.1},  
 12:  
 {1: 244.9, 2: 19.4, 3: 8.7, 4: 5.9,  
 5: 4.7, 6: 4.0, 7: 3.6, 8: 3.3,  
 9: 3.1, 10: 2.9, 11: 2.8}  
 }  
  
**def** get\_f\_criteria(f3, f4):  
 **global** Fcr  
 fcr = Fcr.get(f4)  
 fcr = fcr.get(f3)  
 **return** fcr  
  
Tcr = {1: 12.71, 2: 4.303, 3: 3.182, 4: 2.776, 5: 2.571,  
 6: 2.447, 7: 2.365, 8: 2.306, 9: 2.262, 10: 2.228,  
 11: 2.201, 12: 2.179, 13: 2.160, 14: 2.145, 15: 2.131,  
 16: 2.120, 17: 2.110, 18: 2.101, 19: 2.093, 20: 2.086,  
 21: 2.080, 22: 2.074, 23: 2.069, 24: 2.064, 25: 2.060,  
 26: 2.056, 27: 2.052, 28: 2.048, 29: 2.045, 30: 2.042}  
  
**def** get\_t\_criteria(f3):  
 **global** Tcr  
 tcr = Tcr.get(f3)  
 **if** isinstance(tcr, float):  
 tcr = 1.960  
 **return** tcr  
  
**def** print\_line(m):  
 print(**"-"** \* 12 \* (m+1))  
  
**def** part2(yi, x):  
 **global** b  
 k = len(x)  
 mx = []  
 **for** i **in** range(k):  
 mx.append(round(sum(x[i])/4, 3))  
 my = round(sum(yi)/4, 3)  
 a0 = []  
 **for** i **in** range(k):  
 a0.append(round(sum([x[i][j] \* yi[j] **for** j **in** range(k)])/k, 3))  
 a = []  
 **for** i **in** range(k):  
 a.append([])  
 **for** j **in** range(k):  
 a[i].append(round(sum([x[i][l] \* x[j][l] **for** l **in** range(k)])/k, 3))  
  
*# delta = numpy.array()* base = [[1, mx[0], mx[1], mx[2]],  
 [mx[0], a[0][0], a[0][1], a[0][2]],  
 [mx[1], a[1][0], a[1][1], a[1][2]],  
 [mx[2], a[2][0], a[2][1], a[2][2]]]  
  
 delta = round(numpy.linalg.det(base), 3)  
  
 b = [copy.deepcopy(base) **for** i **in** range(k+1)]  
 **for** i **in** range(k+1):  
 b[i][0][i] = my  
 **for** j **in** range(k):  
 b[i][j+1][i] = a0[j]  
 b[i] = round(numpy.linalg.det(b[i])/delta, 3)  
  
 print(**"b"** + str(i) + **": "** + str(b[i]))  
  
  
max\_num = 231.667  
min\_num = 196.667  
k = 4  
**while True**:  
 m = input(**"m (integer):"**)  
 **if** m.isnumeric():  
 print(**"OK"**)  
 m = int(m)  
 **break  
 else**:  
 print(**"m must be integer"**)  
  
print\_line(m)  
print(**"| "** + **'{:<10}'**.format(**""**), end=**""**)  
**for** i **in** range(1, m+1):  
 print(**"| "** + **'{:<10}'**.format(**"yi"** + str(i)), end=**""**)  
print(**"|"**)  
print\_line(m)  
  
y = []  
  
**for** j **in** range(1, k+1):  
 y.append([])  
 print(**"| "** + **'{:<10}'**.format(j), end=**""**)  
 **for** i **in** range(1, m+1):  
 r = round(random.random() \* (max\_num - min\_num) + min\_num, 3)  
 y[j-1].append(r)  
 print(**"| "** + **'{:<10}'**.format(r), end=**""**)  
 print(**"|"**)  
 print\_line(m)  
  
yi = []  
sigma2 = []  
**for** i **in** range(k):  
 yi.append(round(1/m \* sum(y[i]), 3))  
  
print(**"y (середні): "** + str(yi))  
x = [[-30, -30, 0, 0],  
 [10, 60, 10, 60],  
 [10, 35, 35, 10]]  
part2(yi, x)  
  
x = [[1, 1, 1, 1],  
 [-1, -1, 1, 1],  
 [-1, 1, -1, 1],  
 [-1, 1, 1, -1]]  
  
print(**"y: "** + str(y))  
print(**"yi: "** + str(yi))  
  
S2 = []  
**for** i **in** range(len(y)):  
 S2.append(sum([(y[i][j] - yi[i])\*\*2 **for** j **in** range(len(y[i]))]))  
 S2[i] = round(S2[i]/len(y[i]), 3)  
print(**"S2: "** + str(S2))  
  
Gp = round(max(S2)/sum(S2), 3)  
print(**"Gp: "** + str(Gp))  
  
f1 = m - 1  
f2 = 4  
  
print(**"f1:"** + str(f1))  
print(**"f2:"** + str(f2))  
  
Gcr = get\_cochran\_c(f1, f2)  
print(**"Gcr: "** + str(Gcr))  
**if** Gp < Gcr:  
 print(**"Cochran's C: OK"**)  
**else**:  
 print(**"Cochran's C: :("**)  
 exit(0)  
  
S2v = sum(S2)/4  
  
S2b = round(S2v/(4 \* m), 3)  
Sb = round(math.sqrt(S2b), 3)  
  
f3 = f1 \* f2  
t = get\_t\_criteria(f3)  
bs = []  
ts = []  
d = 0  
**for** i **in** range(4):  
 bs.append(round(sum([yi[j] \* x[i][j] **for** j **in** range(4)])/4, 3))  
 ts.append(round(bs[i]/Sb, 3))  
 **if** ts[i] > t:  
 ts[i] = **True** d += 1  
 **else**:  
 ts[i] = **False**print(**"Чи значимі b: "** + str(ts))  
  
f4 = 4 - d  
x = [[-30, -30, 0, 0],  
 [10, 60, 10, 60],  
 [10, 35, 35, 10]]  
yj = []  
**for** i **in** range(4):  
 yj.append(0)  
 **for** j **in** range(4):  
 **if** ts[j]:  
 **if** j == 0:  
 yj[i] += b[0]  
 **else**:  
 yj[i] += b[j] \* x[j-1][i]  
print(**"yj: "** + str(yj))  
  
S2ad = round(m \* sum([(yj[i] - yi[i])\*\*2 **for** i **in** range(4)])/f4, 3)  
  
Fp = round(S2ad/S2v, 3)  
print(**"Fp: "** + str(Fp))  
Fcr = get\_f\_criteria(f3, f4)  
print(**"Fcr: "** + str(Fcr))  
**if** Fp < Fcr:  
 print(**"F-criteria: OK"**)  
**else**:  
 print(**"F-criteria: :("**)