НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського”

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 6

з дисципліни "Методи оптимізації та планування експерименту" на тему:

ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З КВАДРАТИЧНИМИ ЧЛЕНАМИ

Виконала Дуплій Наталя Володимирівна

Факультет ІОТ,

Група ІВ-82

Залікова книжка № ІВ-8209

Київ – 2020 р.

**Варіант**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № вар | x1 | | x2 | | x3 | | f(x1,x2,x3) |
| min | max | min | max | min | max |
| 208 | -30 | 0 | 10 | 60 | 10 | 35 | 8,0+5,3\*x1+0,5\*x2+5.6\*x3+3.2\*x1\*x1+0.7\*x2\*x2+4.1\*x3\*x3+8.9\*x1\*x2+0.5\*x1\*x3+1.5\*x2\*x3+1.2\*x1\*x2\*x3 |

**Короткі теоретичні відомості**

Спочатку завжди береться лінійна форма рівняння регресії. Якщо при статистичних перевірках гіпотеза про адекватність рівняння регресії оригіналу не підтверджується, то необхідно додати нові члени ряду.

**Результати підготовки та виконання роботи**

Інтервали значень :

Значення, відповідні кодованим:

Складаємо матрицю планування для ОЦКП і заповнюємо таблицю кодованими значеннями .

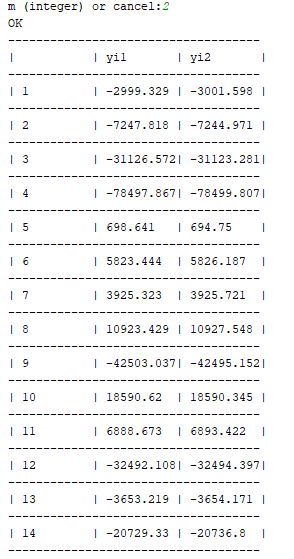
Обираємо m=2

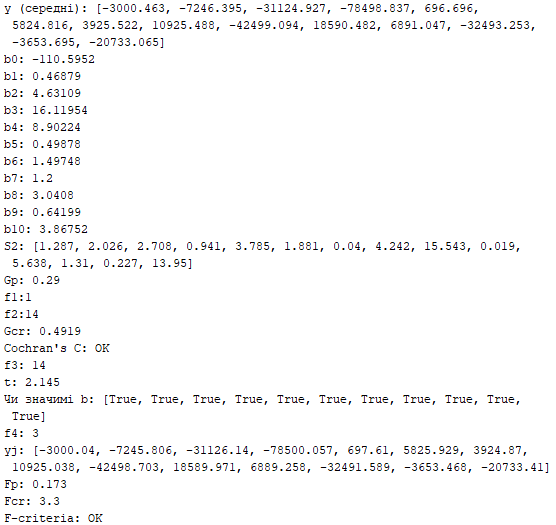
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1** | -1 | -1 | -1 | 1 | 1 | 1 | -1 | 1 | 1 | 1 | -2999.329 | -3001.598 | -3000.463 |
| **2** | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | 1 | 1 | -7247.818 | -7244.971 | -7246.395 |
| **3** | -1 | 1 | -1 | -1 | 1 | -1 | 1 | 1 | 1 | 1 | -31126.572 | -31123.281 | -31124.927 |
| **4** | -1 | 1 | 1 | -1 | -1 | 1 | -1 | 1 | 1 | 1 | -78497.867 | -78499.807 | -78498.837 |
| **5** | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 | 1 | 698.641 | 694.75 | 696.696 |
| **6** | 1 | -1 | 1 | -1 | 1 | -1 | -1 | 1 | 1 | 1 | 5823.444 | 5826.187 | 5824.816 |
| **7** | 1 | 1 | -1 | 1 | -1 | -1 | -1 | 1 | 1 | 1 | 3925.323 | 3925.721 | 3925.522 |
| **8** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 10923.429 | 10927.548 | 10925.488 |
| **9** | -1.73 | 0 | 0 | 0 | 0 | 0 | 0 | 2.9929‬ | 0 | 0 | -42503.037 | -42495.152 | -42499.094 |
| **10** | 1.73 | 0 | 0 | 0 | 0 | 0 | 0 | 2.9929‬ | 0 | 0 | 18590.62 | 18590.345 | 18590.482 |
| **11** | 0 | -1.73 | 0 | 0 | 0 | 0 | 0 | 0 | 2.9929‬ | 0 | 6888.673 | 6893.422 | 6891.047 |
| **12** | 0 | 1.73 | 0 | 0 | 0 | 0 | 0 | 0 | 2.9929‬ | 0 | -32492.108 | -32494.397 | -32493.253 |
| **13** | 0 | 0 | -1.73 | 0 | 0 | 0 | 0 | 0 | 0 | 2.9929‬ | -3653.219 | -3654.171 | -3653.695 |
| **14** | 0 | 0 | 1.73 | 0 | 0 | 0 | 0 | 0 | 0 | 2.9929‬ | -20729.33 | -20736.8 | -20733.065 |

Таблиця з натуралізованими значеннями

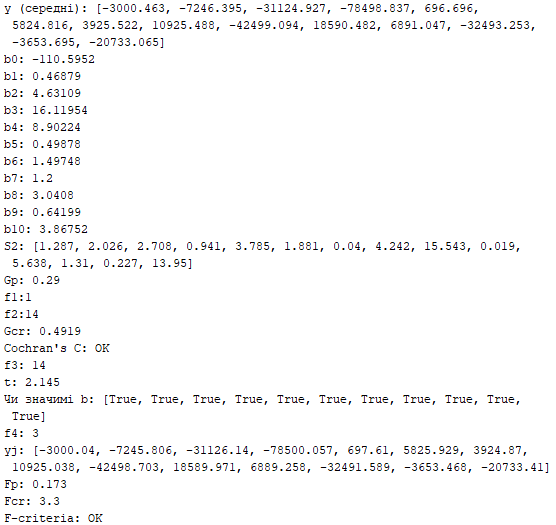
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1** | -30 | 10 | 10 | -300 | -300 | 100 | -3000 | 900 | 100 | 100 | -2999.329 | -3001.598 | -3000.463 |
| **2** | -30 | 10 | 35 | -300 | -1050 | 350 | -10500‬ | 900 | 100 | 1225 | -7247.818 | -7244.971 | -7246.395 |
| **3** | -30 | 60 | 10 | -1800 | -300 | 600 | -18000‬ | 900 | 3600 | 100 | -31126.572 | -31123.281 | -31124.927 |
| **4** | -30 | 60 | 35 | -1800 | -1050 | 2,100 | -63000 | 900 | 3600 | 1225 | -78497.867 | -78499.807 | -78498.837 |
| **5** | 0 | 10 | 10 | 0 | 0 | 100 | 0 | 0 | 100 | 100 | 698.641 | 694.75 | 696.696 |
| **6** | 0 | 10 | 35 | 0 | 0 | 350 | 0 | 0 | 100 | 1225 | 5823.444 | 5826.187 | 5824.816 |
| **7** | 0 | 60 | 10 | 0 | 0 | 600 | 0 | 0 | 3600 | 100 | 3925.323 | 3925.721 | 3925.522 |
| **8** | 0 | 60 | 35 | 0 | 0 | 2100 | 0 | 0 | 3600 | 1225 | 10923.429 | 10927.548 | 10925.488 |
| **9** | -40.95 | 35 | 22.5 | -1433.25 | -921.375‬ | 787.5‬ | -32248.125 | 1676.9025‬ | 1225 | 506.25 | -42503.037 | -42495.152 | -42499.094 |
| **10** | 10.95 | 35 | 22.5 | 383.25 | 246.375 | 787.5‬ | 8623.125‬ | 119.9025‬ | 1225 | 506.25 | 18590.62 | 18590.345 | 18590.482 |
| **11** | -15 | -8.25 | 22.5 | 123.75 | -337.5 | -185.625 | 2784.375‬ | 225 | 68.0625‬‬ | 506.25 | 6888.673 | 6893.422 | 6891.047 |
| **12** | -15 | 78.25 | 22.5 | -1173.75 | -337.5 | 1760.625 | -26409.375‬ | 225 | 6123.0625‬ | 506.25 | -32492.108 | -32494.397 | -32493.253 |
| **13** | -15 | 35 | 0.875 | -525 | -13.125‬ | 30.625‬ | -459.375 | 225 | 1225 | 0.765625‬ | -3653.219 | -3654.171 | -3653.695 |
| **14** | -15 | 35 | 44.125 | -525 | -661.875‬ | 1544.375 | -23165.625 | 225 | 1225 | 1947.015625‬‬ | -20729.33 | -20736.8 | -20733.065 |

Проводимо експеримент у всіх точках плану



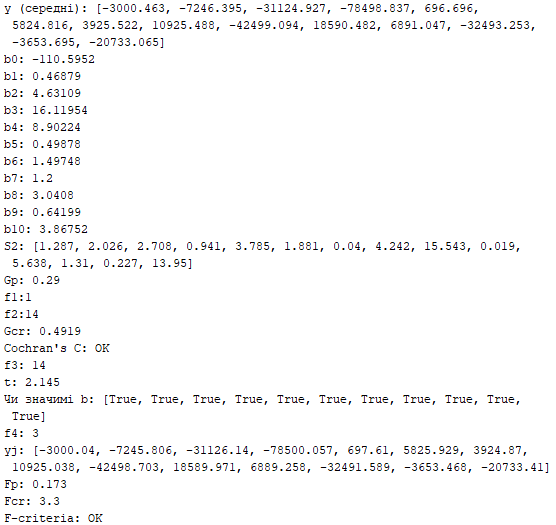


Середні значення функції відгуку та коефіцієнти рівняння регресії знайдені за допомогою програми.



Рівняння регресії:

Проводимо статистичні перевірки.



Дисперсія однорідна за критерієм Кохрена. Усі коефіцієнти значимі за критерієм Стьюдента. Модель адекватна за критерієм Фішера.

Перевірка рівнняння регресії:

**Висновки**

Проведено повний трьохфакторний експеримент і отримана адекватна модель – рівняння регресії, з використанням рототабельного композиційного плану.

**Лістинг програми**

**import** random  
**import** numpy  
**import** copy  
**from** scipy.stats **import** t  
**from** scipy.stats **import** f  
**import** math  
  
**def** cochran(f1, f2, q):  
 fish = f.isf(q/f2, f1, (f2 - 1)\*f1)  
 result = fish/(fish + f2 - 1)  
 **return** result  
  
**def** print\_line(m):  
 print(**"-"** \* 12 \* (m+1))  
  
**def** coeffs\_criterias(yi, x, y):  
 **global** b, m  
 k = len(x[0])  
 mx = [[] **for** i **in** range(len(x) + 1)]  
 mx[0].append(k)  
 **for** i **in** range(1, len(x) + 1):  
 suma = round(sum(x[i-1]), 5)  
 mx[0].append(suma)  
 mx[i].append(suma)  
 **for** j **in** range(0, len(x)):  
 mx[i].append(round(sum([round(x[i-1][l] \* x[j][l], 5) **for** l **in** range(k)]), 5))  
  
 det = numpy.linalg.det(mx)  
 delta = round(det, 5)  
  
 my = [round(sum(yi), 5)]  
 **for** i **in** range(len(x)):  
 my.append(round(sum([yi[j]\*x[i][j] **for** j **in** range(k)]), 5))  
  
 b = [copy.deepcopy(mx) **for** i **in** range(len(x) + 1)]  
 **for** i **in** range(len(x) + 1):  
 **for** j **in** range(len(x) + 1):  
 b[i][j][i] = my[j]  
 b[i] = round(numpy.linalg.det(b[i])/delta,5)  
 print(**"b"** + str(i) + **": "** + str(b[i]))  
  
  
  
 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 = k  
  
 print(**"f1:"** + str(f1))  
 print(**"f2:"** + str(f2))  
  
 alpha = 0.05  
  
 Gcr = round(cochran(f1, f2, alpha), 4)  
 print(**"Gcr: "** + str(Gcr))  
 **if** Gp < Gcr:  
 print(**"Cochran's C: OK"**)  
 **else**:  
 print(**"Cochran's C: :("**)  
 m += 1  
 **return** generate\_y(x)  
  
 S2v = sum(S2) / 4  
  
 S2b = round(S2v / (4 \* m), 3)  
 Sb = round(math.sqrt(S2b), 3)  
  
 f3 = f1 \* f2  
 print(**"f3: "** + str(f3))  
 tcr = round(t.ppf(1 - alpha / 2, df=f3), 3)  
 print(**"t: "** + str(tcr))  
 bs = []  
 ts = []  
 d = 0  
 bs.append(round(sum([yi[j] **for** j **in** range(len(yi))]) / len(yi), 3))  
  
 ts.append(round(bs[0] / Sb, 3))  
 **if** ts[0] < 0:  
 ts[0] \*= -1  
 **if** ts[0] > tcr:  
 ts[0] = **True** d += 1  
 **else**:  
 ts[0] = **False  
 for** i **in** range(len(x)):  
 bs.append(round(sum([yi[j] \* x[i][j] **for** j **in** range(len(yi))]) / len(yi), 3))  
 ts.append(round(bs[i+1] / Sb, 3))  
 **if** ts[i+1] < 0:  
 ts[i+1] \*= -1  
 **if** ts[i+1] > tcr:  
 ts[i+1] = **True** d += 1  
 **else**:  
 ts[i+1] = **False** print(**"Чи значимі b: "** + str(ts))  
  
 f4 = k - d  
 print(**"f4: "** + str(f4))  
 yj = []  
 b0 = []  
 **for** i **in** range(len(b)):  
 **if** ts[i]:  
 b0.append(b[i])  
 **else**:  
 b0.append(0)  
 **for** j **in** range(k):  
 yj.append(round(b0[0] + sum([x[i-1][j] \* b0[i] **for** i **in** range(1, len(b0))]), 3))  
 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 = round(f.ppf(1 - alpha, f4, f3), 1)  
 print(**"Fcr: "** + str(Fcr))  
 **if** Fp < Fcr:  
 print(**"F-criteria: OK"**)  
 **else**:  
 print(**"F-criteria: :("**)  
 start(x)  
  
**def** generate\_y(x):  
 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 = []  
 k = len(x[0])  
 **for** j **in** range(1, k + 1):  
 y.append([])  
 print(**"| "** + **'{:<10}'**.format(j), end=**""**)  
 **for** i **in** range(1, m + 1):  
 r = round(8 + 5.3 \* x[0][j - 1] +  
 0.5 \* x[1][j - 1] +  
 5.6 \* x[2][j - 1] +  
 3.2 \* x[7][j - 1] +  
 0.7 \* x[8][j - 1] +  
 4.1 \* x[9][j - 1] +  
 8.9 \* x[3][j - 1] +  
 0.5 \* x[4][j - 1] +  
 1.5 \* x[5][j - 1] +  
 1.2 \* x[6][j - 1] +  
 random.random() \* 10 - 5, 3)  
 y[j - 1].append(r)  
 print(**"| "** + **'{:<10}'**.format(r), end=**""**)  
 print(**"|"**)  
 print\_line(m)  
  
 yi = []  
 **for** i **in** range(k):  
 yi.append(round(1 / m \* sum(y[i]), 3))  
  
 print(**"y (середні): "** + str(yi))  
  
 coeffs\_criterias(yi, x, y)  
  
**def** start(x):  
 **global** m  
 **while True**:  
 m = input(**"m (integer) or cancel:"**)  
 **if** m.isnumeric():  
 print(**"OK"**)  
 m = int(m)  
 **break  
 elif** m == **"cancel"**:  
 **return  
 else**:  
 print(**"m must be integer"**)  
  
 generate\_y(x)  
  
  
x = [  
 [-30, -30, -30, -30, 0, 0, 0, 0, -40.95, 10.95, -15, -15, -15, -15],  
 [10, 10, 60, 60, 10, 10, 60, 60, 35, 35, -8.25, 78.25, 35, 35],  
 [10, 35, 10, 35, 10, 35, 10, 35, 22.5, 22.5, 22.5, 22.5, 0.875, 44.125]  
]  
  
k = len(x[0])  
x.append([round(x[0][i] \* x[1][i], 6) **for** i **in** range(k)])  
x.append([round(x[0][i] \* x[2][i], 6) **for** i **in** range(k)])  
x.append([round(x[1][i] \* x[2][i], 6) **for** i **in** range(k)])  
x.append([round(x[0][i] \* x[1][i] \* x[2][i], 6) **for** i **in** range(k)])  
**for** j **in** range(3):  
 x.append([round(x[j][i] \*\* 2, 6) **for** i **in** range(k)])  
start(x)

**Результати виконання**

