Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. 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»

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІВ-81

Федорусов Іван

Варіант: 127

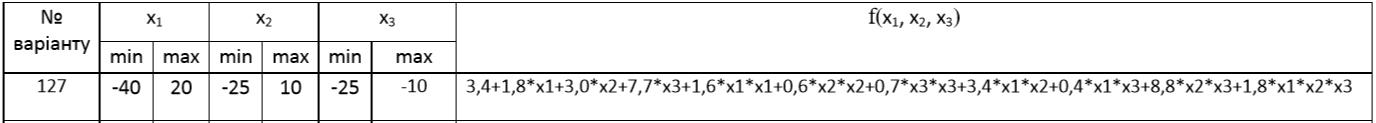
ПЕРЕВІРИВ:

Регіда П. Г.

Київ – 2020

Мета:

Провести трьохфакторний експеремент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії адекватне об'єкту.

Варіант: 

Лістинг програми:

Висновок:

**from** \_pydecimal **import** Decimal  
**from** scipy.stats **import** f, t  
**from** random **import** randrange  
**from** math **import** sqrt, fabs **as** fab  
**from** numpy.linalg **import** solve  
  
dependent, independent = 0, 0  
x1 = [-20, 30]  
x2 = [5, 40]  
x3 = [5, 40]  
x01 = (x1[1] + x1[0]) / 2  
x02 = (x2[1] + x2[0]) / 2  
x03 = (x3[1] + x3[0]) / 2  
delta\_x1 = x1[1] - x01  
delta\_x2 = x2[1] - x02  
delta\_x3 = x3[1] - x03  
N = 15  
matrix\_pe = [  
 [-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, -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],  
 [+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, +1, +1, +1, +1, +1],  
 [-1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],  
 [+1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],  
 [0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],  
 [0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],  
 [0, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929],  
 [0, 0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  
]  
  
**def** generate\_matrix():  
 **def** f(X1, X2, X3):  
 y = 0.6 + 8.0 \* X1 + 8.8 \* X2 + 9.2 \* X3 + 4.7 \* X1 \* X1 + 0.1 \* X2 \* X2 + 1.1 \* X3 \* X3 + 6.3 \* X1 \* X2 + \  
 0.4 \* X1 \* X3 + 1.3 \* X2 \* X3 + 2.9 \* X1 \* X2 \* X3 + randrange(0, 10) - 5  
 **return** y  
  
 matrix\_with\_y = [[f(matrix\_x[j][0], matrix\_x[j][1], matrix\_x[j][2]) **for** i **in** range(m)] **for** j **in** range(N)]  
 **return** matrix\_with\_y  
  
**def** average\_value(arr, orientation):  
 average = []  
 **if** orientation == 1:  
 **for** rows **in** range(len(arr)): average.append(sum(arr[rows]) / len(arr[rows]))  
 **else**:  
 **for** column **in** range(len(arr[0])):  
 number\_arr = []  
 **for** rows **in** range(len(arr)): number\_arr.append(arr[rows][column])  
 average.append(sum(number\_arr) / len(number\_arr))  
 **return** average  
  
**def** cochrane(selectionSize, qty\_of\_selections, significance):  
 selectionSize += 1  
 partResult1 = significance / (selectionSize - 1)  
 params = [partResult1, qty\_of\_selections, (selectionSize - 1 - 1) \* qty\_of\_selections]  
 fischer = f.isf(\*params)  
 result = fischer / (fischer + (selectionSize - 1 - 1))  
 **return** Decimal(result).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
**def** student(f3, significance):  
 **return** Decimal(abs(t.ppf(significance / 2, f3))).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
**def** fischer(f3, f4, significance):  
 **return** Decimal(abs(f.isf(significance, f4, f3))).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
  
**def** x(l1, l2, l3):  
 x\_1 = l1 \* delta\_x1 + x01  
 x\_2 = l2 \* delta\_x2 + x02  
 x\_3 = l3 \* delta\_x3 + x03  
 **return** [x\_1, x\_2, x\_3]  
  
  
**def** a(first, second):  
 necessary\_a = 0  
 **for** j **in** range(N): necessary\_a += matrix\_x[j][first - 1] \* matrix\_x[j][second - 1] / N  
 **return** necessary\_a  
  
  
**def** find\_known(number):  
 necessary\_a = 0  
 **for** j **in** range(N): necessary\_a += middle\_y[j] \* matrix\_x[j][number - 1] / 15  
 **return** necessary\_a  
  
  
**def** check\_result(arr\_b, k):  
 y\_i = arr\_b[0] + arr\_b[1] \* matrix[k][0] + arr\_b[2] \* matrix[k][1] + arr\_b[3] \* matrix[k][2] + arr\_b[4] \* matrix[k][3] + arr\_b[5] \* matrix[k][4] + arr\_b[6] \* matrix[k][5] + arr\_b[7] \* matrix[k][6] + arr\_b[8] \* matrix[k][7] + arr\_b[9] \* matrix[k][8] + arr\_b[10] \* matrix[k][9]  
 **return** y\_i  
  
  
**def** student\_test(arr\_b, number\_x=10):  
 dispersion\_b = sqrt(dispersion\_b2)  
 **for** column **in** range(number\_x + 1):  
 t\_practice = 0  
 t\_theoretical = student(f3, q)  
 **for** row **in** range(N): t\_practice += middle\_y[row] / N **if** column == 0 **else** middle\_y[row] \* matrix\_pe[row][column - 1]  
 **if** fab(t\_practice / dispersion\_b) < t\_theoretical: arr\_b[column] = 0  
 **return** arr\_b  
  
  
**def** fischer\_test():  
 dispersion\_ad = 0  
 f4 = N - d  
 **for** row **in** range(len(middle\_y)): dispersion\_ad += (m \* (middle\_y[row] - check\_result(student\_arr, row))) / (N - d)  
 f\_practice = dispersion\_ad / dispersion\_b2  
 f\_theoretical = fischer(f3, f4, q)  
 **return** f\_practice < f\_theoretical  
  
m = 3  
p = 0.95  
matrix\_x = [[] **for** x\_val **in** range(N)]  
**for** i **in** range(len(matrix\_x)):  
 **if** i < 8:  
 x\_1 = x1[0] **if** matrix\_pe[i][0] == -1 **else** x1[1]  
 x\_2 = x2[0] **if** matrix\_pe[i][1] == -1 **else** x2[1]  
 x\_3 = x3[0] **if** matrix\_pe[i][2] == -1 **else** x3[1]  
 **else**:  
 arr\_x = x(matrix\_pe[i][0], matrix\_pe[i][1], matrix\_pe[i][2])  
 x\_1, x\_2, x\_3 = arr\_x  
 matrix\_x[i] = [x\_1, x\_2, x\_3, x\_1 \* x\_2, x\_1 \* x\_3, x\_2 \* x\_3, x\_1 \* x\_2 \* x\_3, x\_1 \*\* 2, x\_2 \*\* 2, x\_3 \*\* 2]  
  
homogeneity, dispersion\_y = **False**, **None**matrix\_y = generate\_matrix()  
middle\_x = average\_value(matrix\_x, 0)  
middle\_y = average\_value(matrix\_y, 1)  
matrix = [(matrix\_x[i] + matrix\_y[i]) **for** i **in** range(N)]  
mx\_i = middle\_x  
my = sum(middle\_y) / 15  
  
values = [  
 [1, mx\_i[0], mx\_i[1], mx\_i[2], mx\_i[3], mx\_i[4], mx\_i[5], mx\_i[6], mx\_i[7], mx\_i[8], mx\_i[9]],  
 [mx\_i[0], a(1, 1), a(1, 2), a(1, 3), a(1, 4), a(1, 5), a(1, 6), a(1, 7), a(1, 8), a(1, 9), a(1, 10)],  
 [mx\_i[1], a(2, 1), a(2, 2), a(2, 3), a(2, 4), a(2, 5), a(2, 6), a(2, 7), a(2, 8), a(2, 9), a(2, 10)],  
 [mx\_i[2], a(3, 1), a(3, 2), a(3, 3), a(3, 4), a(3, 5), a(3, 6), a(3, 7), a(3, 8), a(3, 9), a(3, 10)],  
 [mx\_i[3], a(4, 1), a(4, 2), a(4, 3), a(4, 4), a(4, 5), a(4, 6), a(4, 7), a(4, 8), a(4, 9), a(4, 10)],  
 [mx\_i[4], a(5, 1), a(5, 2), a(5, 3), a(5, 4), a(5, 5), a(5, 6), a(5, 7), a(5, 8), a(5, 9), a(5, 10)],  
 [mx\_i[5], a(6, 1), a(6, 2), a(6, 3), a(6, 4), a(6, 5), a(6, 6), a(6, 7), a(6, 8), a(6, 9), a(6, 10)],  
 [mx\_i[6], a(7, 1), a(7, 2), a(7, 3), a(7, 4), a(7, 5), a(7, 6), a(7, 7), a(7, 8), a(7, 9), a(7, 10)],  
 [mx\_i[7], a(8, 1), a(8, 2), a(8, 3), a(8, 4), a(8, 5), a(8, 6), a(8, 7), a(8, 8), a(8, 9), a(8, 10)],  
 [mx\_i[8], a(9, 1), a(9, 2), a(9, 3), a(9, 4), a(9, 5), a(9, 6), a(9, 7), a(9, 8), a(9, 9), a(9, 10)],  
 [mx\_i[9], a(10, 1), a(10, 2), a(10, 3), a(10, 4), a(10, 5), a(10, 6), a(10, 7), a(10, 8), a(10, 9), a(10, 10)]  
]  
known\_values = [my, find\_known(1), find\_known(2), find\_known(3), find\_known(4), find\_known(5), find\_known(6), find\_known(7), find\_known(8), find\_known(9), find\_known(10)]  
  
beta = solve(values, known\_values)  
print(**"\nРівняння регресії:"**)  
print(**"{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 + {:.3f} \* Х1X2 + {:.3f} \* Х1X3 + {:.3f} \* Х2X3 + {:.3f} \* Х1Х2X3 + {:.3f} \* X11^2 + {:.3f} \* X22^2 + {:.3f} \* X33^2 = ŷ\n\nПеревірка"**.format(beta[0], beta[1], beta[2], beta[3], beta[4], beta[5], beta[6], beta[7], beta[8], beta[9], beta[10]))  
**for** i **in** range(N): print(**"ŷ{} = {:.3f} ≈ {:.3f}"**.format((i + 1), check\_result(beta, i), middle\_y[i]))  
  
**while not** homogeneity:  
 print(**"\n"** + **" "** \* 65 + **"Матриця планування"** + **" "** \* 65 + **" X1 X2 X3 X1X2 X1X3 X2X3 X1X2X3 X1X1 X2X2 X3X3 Yi ..."**)  
 **for** row **in** range(N):  
 print(**" "**, end=**' '**)  
 **for** column **in** range(len(matrix[0])): print(**"{:^12.3f}"**.format(matrix[row][column]), end=**' '**)  
 print(**" "**)  
 print(**"\n"**)  
 dispersion\_y = [0.0 **for** x **in** range(N)]  
 **for** i **in** range(N):  
 dispersion\_i = 0  
 **for** j **in** range(m): dispersion\_i += (matrix\_y[i][j] - middle\_y[i]) \*\* 2  
 dispersion\_y.append(dispersion\_i / (m - 1))  
 f1 = m - 1  
 f2 = N  
 f3 = f1 \* f2  
 q = 1 - p  
 Gp = max(dispersion\_y) / sum(dispersion\_y)  
 print(**"Критерій Кохрена"**)  
 Gt = cochrane(f2, f1, q)  
 **if** Gt > Gp **and** m < 25:  
 print(**"Дисперсія однорідна з q = {:.2f}!\n"**.format(q))  
 homogeneity = **True  
 else**:  
 print(**"Дисперсія не є однорідною. Повторна спроба при m += 1..."**)  
 m += 1  
 **if** m == 25: exit()  
  
  
dispersion\_b2 = sum(dispersion\_y) / (N \* N \* m)  
student\_arr = list(student\_test(beta))  
print(**"Рівняння регресії за критерієм Стьюдента\n{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 + {:.3f} \* Х1X2 + {:.3f} \* Х1X3 + {:.3f} \* Х2X3 + {:.3f} \* Х1Х2X3 + {:.3f} \* X11^2 + {:.3f} \* X22^2 + {:.3f} \* X33^2 = ŷ\n\nПеревірка"**.format(student\_arr[0], student\_arr[1], student\_arr[2], student\_arr[3], student\_arr[4], student\_arr[5], student\_arr[6], student\_arr[7], student\_arr[8], student\_arr[9], student\_arr[10]))  
**for** i **in** range(N): print(**"ŷ{} = {:.3f} ≈ {:.3f}"**.format((i + 1), check\_result(student\_arr, i), middle\_y[i]))  
d = 11 - student\_arr.count(0)  
dependent += N - d  
independent += d  
print(**'Математична модель адекватна експериментальним даним.'**) **if** fischer\_test() **else** print(**'Математична модель не є адекватною експериментальним даним.'**)

В процесі виконання лабораторної роботи № 6 проведено трьохфакторний експеримент з використанням рівняння з квадратичними членами. Складено матрицю планування, знайшдено коефіцієнти рівняння регресії. Розрахунки пiдтвержденi 3-ма статичними перевiрками. Написана робоча тестова програма на мовi програмування Python. Результати співпадають iз очiкуваним результатом.