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 № 4**

з дисципліни «Методи оптимізації та планування експерименту» на тему

«**ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ**»

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-83

Мітячкін Денис

Залікова - 8320

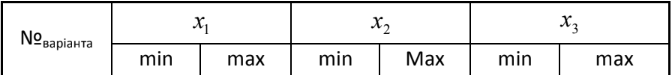
ПЕРЕВІРИВ:

ас. Регіда П. Г.

Київ – 2020

**Мета:** провести повний трьохфакторний експеремент. Знайти рівняння регресії адекватне об'єкту.

**Варіант:**



**Код програми:**

**from** math **import** sqrt, fabs **as** fab  
**from** numpy.linalg **import** det  
**from** random **import** randrange  
**from** \_pydecimal **import** Decimal  
**from** scipy.stats **import** f, t  
  
*# Значення за варіантом*m = 0  
N = 2 \*\* 3  
  
min\_x1 = -10  
max\_x1 = 50  
min\_x2 = 20  
max\_x2 = 60  
min\_x3 = -10  
max\_x3 = 10  
  
min\_y = 200 + int((min\_x1 + min\_x2 + min\_x3) / 3)  
max\_y = 200 + int((max\_x1 + max\_x2 + max\_x3) / 3)  
  
matrixExp = [  
 [-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],  
]  
  
a1, a2, a3, a11, a22, a33, a12, a13, a23 = 0, 0, 0, 0, 0, 0, 0, 0, 0  
  
**def** matrixGenerator():  
 *# Генеруємо матрицю* matrix\_y = [[randrange(min\_y, max\_y)  
 **for** y **in** range(m)] **for** x **in** range(N)]  
 **return** matrix\_y  
  
  
**def** middleValue(arr, orientation):  
 *# Функція пошуку середнього значення по колонках або по рядках* middle = []  
 **if** orientation == 1:  
 *# Середнє значення по рядку* **for** rows **in** range(len(arr)):  
 middle.append(sum(arr[rows]) / len(arr[rows]))  
 **else**:  
 *# Середнє значення по колонкі* **for** column **in** range(len(arr[0])):  
 numberArr = []  
 **for** rows **in** range(len(arr)):  
 numberArr.append(arr[rows][column])  
 middle.append(sum(numberArr) / len(numberArr))  
 **return** middle  
  
  
**def** studentTest(b\_lst, number\_x=4):  
 *# Критерій Стьюдента* dispersion\_b = sqrt(sum(dispersion\_y) / (N \* N \* m))  
 t\_lst = [0.0 **for** x **in** range(N)]  
 **for** k **in** range(number\_x):  
 **for** x **in** range(N):  
 **if** k == 0:  
 t\_lst[x] += middle\_y[x] / N  
 **else**:  
 t\_lst[x] += middle\_y[x] \* matrixExp[x][k - 1] / N  
 **for** i **in** range(len(t\_lst)):  
 t\_lst[i] = fab(t\_lst[i]) / dispersion\_b  
 tt = CritValues.studentValue(f3, q)  
 **for** i **in** range(number\_x):  
 **if** t\_lst[i] > tt:  
 **continue  
 else**:  
 t\_lst[i] = 0  
 **for** j **in** range(number\_x):  
 b\_lst[j] = 0 **if** t\_lst[j] == 0 **else** b\_lst[j]  
 **return** b\_lst  
  
  
**def** fisherTest(b\_lst, number=3):  
 *# Критерій Фішера* dispersion\_ad = 0  
 **for** i **in** range(N):  
 yj = b\_lst[0]  
 **for** j **in** range(number):  
 yj += matrix[i][j] \* b\_lst[j + 1]  
 dispersion\_ad += (middle\_y[i] - yj) \*\* 2  
 dispersion\_ad /= m / (N - d)  
 Fp = dispersion\_ad / (sqrt(sum(dispersion\_y) / (N \* N \* m)))  
 Ft = CritValues.fisherValue(f3, f4, q)  
 **return True if** Fp < Ft **else False  
  
class** CritValues:  
 *# Критичні значення* @staticmethod  
 **def** cohrenValue(selectionSize, selectionQty, significance):  
 selectionSize += 1  
 partResult1 = significance / (selectionSize - 1)  
 params = [partResult1, selectionQty, (selectionSize - 1 - 1) \* selectionQty]  
 fisher = f.isf(\*params)  
 result = fisher / (fisher + (selectionSize - 1 - 1))  
 **return** Decimal(result).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
 @staticmethod  
 **def** studentValue(f3, significance):  
 *# Значення критерію Стьюдента* **return** Decimal(abs(t.ppf(significance / 2, f3))).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
 @staticmethod  
 **def** fisherValue(f3, f4, significance):  
 *# Значення критерію Фішера* **return** Decimal(abs(f.isf(significance, f4, f3))).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
  
correctInp = **False  
while not** correctInp:  
 **try**:  
 m = int(input(**"Кількість повторень: "**))  
 p = float(input(**"Довірча ймовірність: "**))  
 correctInp = **True  
 except** ValueError:  
 **pass**matrix\_x, matrix\_3x = [[] **for** x **in** range(N)], [[] **for** x **in** range(N)]  
  
**for** i **in** range(len(matrix\_x)):  
 x1 = min\_x1 **if** matrixExp[i][0] == -1 **else** max\_x1  
 x2 = min\_x2 **if** matrixExp[i][1] == -1 **else** max\_x2  
 x3 = min\_x3 **if** matrixExp[i][2] == -1 **else** max\_x3  
 matrix\_x[i] = [x1, x2, x3, x1 \* x2, x1 \* x3, x2 \* x3, x1 \* x2 \* x3]  
 matrix\_3x[i] = [x1, x2, x3]  
  
adequacy, odinority = **False**, **False***# Адекватність і однорідність по замовчуванні False***while not** adequacy:  
 **while not** odinority:  
 matrix\_y = matrixGenerator()  
 middle\_x = middleValue(arr=matrix\_3x, orientation=0)  
 middle\_y = middleValue(arr=matrix\_y, orientation=1)  
  
 **for** i **in** range(N):  
 a1 += matrix\_x[i][0] \* middle\_y[i] / N  
 a2 += matrix\_x[i][1] \* middle\_y[i] / N  
 a3 += matrix\_x[i][2] \* middle\_y[i] / N  
 a11 += matrix\_x[i][0] \*\* 2 / N  
 a22 += matrix\_x[i][1] \*\* 2 / N  
 a33 += matrix\_x[i][2] \*\* 2 / N  
 a12 += matrix\_x[i][0] \* matrix\_x[i][1] / N  
 a13 += matrix\_x[i][0] \* matrix\_x[i][2] / N  
 a23 += matrix\_x[i][1] \* matrix\_x[i][2] / N  
 a21 = a12  
 a31 = a13  
 a32 = a23  
 my = sum(middle\_y) / len(middle\_y)  
 numb0 = [[my, middle\_x[0], middle\_x[1], middle\_x[2]], [a1, a11, a12, a13], [a2, a21, a22, a23],  
 [a3, a31, a32, a33]]  
 numb1 = [[1, my, middle\_x[1], middle\_x[2]], [middle\_x[0], a1, a12, a13], [middle\_x[1], a2, a22, a23],  
 [middle\_x[2], a3, a32, a33]]  
 numb2 = [[1, middle\_x[0], my, middle\_x[2]], [middle\_x[0], a11, a1, a13], [middle\_x[1], a21, a2, a23],  
 [middle\_x[2], a31, a3, a33]]  
 numb3 = [[1, middle\_x[0], middle\_x[1], my], [middle\_x[0], a11, a12, a1], [middle\_x[1], a21, a22, a2],  
 [middle\_x[2], a31, a32, a3]]  
 dividerB = [[1, middle\_x[0], middle\_x[1], middle\_x[2]], [middle\_x[0], a11, a12, a13],  
 [middle\_x[1], a21, a22, a23], [middle\_x[2], a31, a32, a33]]  
  
 b0 = det(numb0) / det(dividerB)  
 b1 = det(numb1) / det(dividerB)  
 b2 = det(numb2) / det(dividerB)  
 b3 = det(numb3) / det(dividerB)  
  
 matrix = []  
 **for** i **in** range(N):  
 matrix.append(matrix\_3x[i] + matrix\_y[i])  
  
 print(**"\nМатриця з натуральних значень факторів"**)  
 print(**" X1 X2 X3 Y1 Y2 Y3 "**)  
 **for** i **in** range(len(matrix)):  
 print(**""**, end=**" "**)  
 **for** j **in** range(len(matrix[i])):  
 print(matrix[i][j], end=**" "**)  
 print(**""**)  
  
 print(**"\nРівняння регресії"**)  
 print(**"{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 = ŷ"**.format(b0, b1, b2, b3))  
  
 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(**"\nКритерій Кохрена"**)  
 Gt = CritValues.cohrenValue(f2, f1, q)  
 **if** Gt > Gp **or** m >= 25:  
 print(**"Дисперсія однорідна при рівні значимості {:.2f}!\nЗбільшувати m не потрібно."**.format(q))  
 odinority = **True  
 else**:  
 print(**"Дисперсія не однорідна при рівні значимості {:.2f}!"**.format(q))  
 m += 1  
 **if** m == 25:  
 exit()  
  
 print(**"\nКритерій Стьюдента"**)  
 beta\_1 = [b0, b1, b2, b3]  
 significant\_coefficients = studentTest(beta\_1)  
 print(**"{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 = ŷ"**.format(significant\_coefficients[0],  
 significant\_coefficients[1],  
 significant\_coefficients[2],  
 significant\_coefficients[3]))  
  
 d = len(significant\_coefficients) - significant\_coefficients.count(0)  
 f4 = N - d  
 print(**"\nКритерій Фішера"**)  
 **if not** fisherTest(significant\_coefficients):  
 print(**"Рівняння регресії неадекватне стосовно оригіналу\nЕфект взаємодії!"**)  
 beta = [0 **for** i **in** range(N)]  
 **for** i **in** range(N):  
 **if** i == 0:  
 beta[i] += sum(middle\_y) / len(middle\_y)  
 **else**:  
 **for** j **in** range(7):  
 beta[i] += middle\_y[i] \* matrixExp[i][j] / N  
 print(**"\nРівняння регресії з ефектом взаємодії"**)  
 print(**"{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 + {:.3f} \* Х1X2 + {:.3f} \* Х1X3 + {:.3f} \* Х2X3"  
 "+ {:.3f} \* Х1Х2X3= ŷ"**.format(beta[0], beta[1], beta[2], beta[3], beta[4], beta[5], beta[6], beta[7]))  
 print(**"\nКритерій Кохрена"**)  
 Gt = CritValues.cohrenValue(f2, f1, q)  
 **if** Gt > Gp **or** m >= 25:  
 print(**"Дисперсія однорідна при рівні значимості {:.2f}!\nЗбільшувати m не потрібно."**.format(q))  
 odinority = **True  
 else**:  
 print(**"Дисперсія не однорідна при рівні значимості {:.2f}!"**.format(q))  
 m += 1  
 **if** m == 25:  
 exit()  
 significant\_coefficients = studentTest(beta, 8)  
 print(**"\nКритерій Стьюдента"**)  
 print(**"{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 + {:.3f} \* Х1X2 + {:.3f} \* Х1X3 + {:.3f} \* Х2X3"  
 "+ {:.3f} \* Х1Х2X3= ŷ"**.format(significant\_coefficients[0], significant\_coefficients[1],  
 significant\_coefficients[2],  
 significant\_coefficients[3],  
 significant\_coefficients[4],  
 significant\_coefficients[5],  
 significant\_coefficients[6],  
 significant\_coefficients[7]))  
  
 d = len(significant\_coefficients) - significant\_coefficients.count(0)  
 f4 = N - d  
 **if** studentTest(beta, 7):  
 print(**"Рівняння регресії адекватне стосовно оригіналу"**)  
 adequacy = **True  
 else**:  
 print(**"Рівняння регресії адекватне стосовно оригіналу"**)  
 adequacy = **True**

**Результат:**

Кількість повторень: 3

Довірча ймовірність: 0.95

Матриця з натуральних значень факторів

X1 X2 X3 Y1 Y2 Y3

-10 20 -10 209 203 202

-10 20 10 211 234 208

-10 60 -10 204 202 220

-10 60 10 230 230 231

50 20 -10 221 231 224

50 20 10 233 229 234

50 60 -10 213 235 239

50 60 10 225 204 205

Рівняння регресії

216.931 + 0.151 \* X1 + -0.002 \* X2 + 0.296 \* X3 = ŷ

Критерій Кохрена

Дисперсія однорідна при рівні значимості 0.05!

Збільшувати m не потрібно.

Критерій Стьюдента

216.931 + 0.000 \* X1 + 0.000 \* X2 + 0.296 \* X3 = ŷ

Критерій Фішера

Рівняння регресії неадекватне стосовно оригіналу

Ефект взаємодії!

Рівняння регресії з ефектом взаємодії

219.875 + -27.208 \* X1 + -26.083 \* X2 + -28.792 \* X3 + -28.167 \* Х1X2 + -29.000 \* Х1X3 + -28.625 \* Х2X3+ 184.917 \* Х1Х2X3= ŷ

Критерій Кохрена

Дисперсія однорідна при рівні значимості 0.05!

Збільшувати m не потрібно.

Критерій Стьюдента

0.000 + 0.000 \* X1 + 0.000 \* X2 + 0.000 \* X3 + 0.000 \* Х1X2 + 0.000 \* Х1X3 + 0.000 \* Х2X3+ 184.917 \* Х1Х2X3= ŷ

Рівняння регресії адекватне стосовно оригіналу

Process finished with exit code 0

**Висновок:** Отже, у ході виконання даної лабораторної роботи ми провели повний трьохфакторний експеримент при використанні рівняння з ефектом взаємодії. Склали матрицю планування, знайшли коефіцієнти рівняння регресії, провели 3 статистичні перевірки. Була написана текстова програма, результати наведені вище. Результати співпадають із калькулятором. Кінцева мета роботи досягнута!