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»

Тема: ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ

ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ.

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-83

Мельник Р.К.

Залікова – 8319

Варіант-317

ПЕРЕВІРИВ:

Регіда П.Г.

Київ – 2020

**Лабораторна робота № 4**

**Тема:** Проведення трьохфакторного експерименту

при використанні рівняння регресії з урахуванням ефекту взаємодії.

**Мета:** Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

**Завдання на лабораторну роботу**

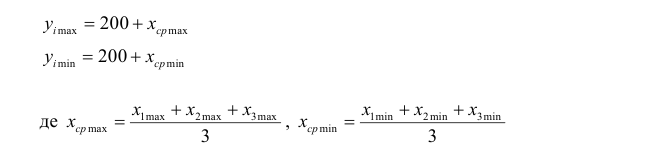
1. Скласти матрицю планування для повного трьохфакторного експерименту.

2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і

знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у

певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в

журналі викладача.



3. Знайти коефіцієнти рівняння регресії і записати його.

4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.

5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати

скореговане рівняння регресії.

6. Написати комп'ютерну програму, яка усе це моделює.

**Код програми:**

from math import \*  
from numpy.linalg import det  
from random import randrange  
  
  
class Critical\_values:  
 @staticmethod  
 def get\_cohren\_value(size\_of\_selections, qty\_of\_selections, significance):  
 from \_pydecimal import Decimal  
 from scipy.stats import f  
 size\_of\_selections += 1  
 partResult1 = significance / (size\_of\_selections - 1)  
 params = [partResult1, qty\_of\_selections, (size\_of\_selections - 1 - 1) \* qty\_of\_selections]  
 fisher = f.isf(\*params)  
 result = fisher / (fisher + (size\_of\_selections - 1 - 1))  
 return Decimal(result).quantize(Decimal('.0001')).\_\_float\_\_()  
  
 @staticmethod  
 def get\_student\_value(f3, significance):  
 from \_pydecimal import Decimal  
 from scipy.stats import t  
 return Decimal(abs(t.ppf(significance / 2, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
  
 @staticmethod  
 def get\_fisher\_value(f3, f4, significance):  
 from \_pydecimal import Decimal  
 from scipy.stats import f  
 return Decimal(abs(f.isf(significance, f4, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
  
N = 8  
m = 3  
p = 0.95  
  
matrix\_pfe = [  
 [-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],  
]  
  
x1\_min, x1\_max = -10, 50  
x2\_min, x2\_max = -20, 60  
x3\_min, x3\_max = 50, 55  
y\_min = 200 + int((x1\_min + x2\_min + x3\_min) / 3)  
y\_max = 200 + int((x1\_max + x2\_max + x3\_max) / 3)  
  
matrix\_x, matrix\_3x = [[] for x in range(N)], [[] for x in range(N)]  
  
for i in range(len(matrix\_x)):  
 x1 = x1\_min if matrix\_pfe[i][0] == -1 else x1\_max  
 x2 = x2\_min if matrix\_pfe[i][1] == -1 else x2\_max  
 x3 = x3\_min if matrix\_pfe[i][2] == -1 else x3\_max  
 matrix\_x[i] = [x1, x2, x3, x1 \* x2, x1 \* x3, x2 \* x3, x1 \* x2 \* x3]  
 matrix\_3x[i] = [x1, x2, x3]  
  
adekwatna, odnoridna = False, False  
  
while not adekwatna:  
 while not odnoridna:  
 def generate\_matrix():  
 matrix\_with\_y = [[randrange(y\_min, y\_max) for y in range(m)] for x in range(N)]  
 return matrix\_with\_y  
  
 def find\_average(lst, orientation):  
 average = []  
 if orientation == 1:  
 for rows in range(len(lst)):  
 average.append(sum(lst[rows]) / len(lst[rows]))  
 else:  
 for column in range(len(lst[0])):  
 number\_lst = []  
 for rows in range(len(lst)):  
 number\_lst.append(lst[rows][column])  
 average.append(sum(number\_lst) / len(number\_lst))  
 return average  
  
 def student\_test(b\_lst, number\_x=4):  
 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] += average\_y[x] / N  
 else:  
 t\_lst[x] += average\_y[x] \* matrix\_pfe[x][k - 1] / N  
 for i in range(len(t\_lst)):  
 t\_lst[i] = fabs(t\_lst[i]) / dispersion\_b  
 tt = Critical\_values.get\_student\_value(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 fisher(b\_lst, number=3):  
 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 += (average\_y[i] - yj) \*\* 2  
 dispersion\_ad /= m / (N - d)  
 Fp = dispersion\_ad / (sqrt(sum(dispersion\_y) / (N \* N \* m)))  
 Ft = Critical\_values.get\_fisher\_value(f3, f4, q)  
 return True if Fp < Ft else False  
  
 matrix\_y = generate\_matrix()  
 average\_x = find\_average(lst=matrix\_3x, orientation=0)  
 average\_y = find\_average(lst=matrix\_y, orientation=1)  
 a1, a2, a3, a11, a22, a33, a12, a13, a23 = 0, 0, 0, 0, 0, 0, 0, 0, 0  
  
 for i in range(N):  
 a1 += matrix\_x[i][0] \* average\_y[i] / N  
 a2 += matrix\_x[i][1] \* average\_y[i] / N  
 a3 += matrix\_x[i][2] \* average\_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(average\_y) / len(average\_y)  
 b0\_numerator = [[my, average\_x[0], average\_x[1], average\_x[2]], [a1, a11, a12, a13], [a2, a21, a22, a23],  
 [a3, a31, a32, a33]]  
 b1\_numerator = [[1, my, average\_x[1], average\_x[2]], [average\_x[0], a1, a12, a13], [average\_x[1], a2, a22, a23],  
 [average\_x[2], a3, a32, a33]]  
 b2\_numerator = [[1, average\_x[0], my, average\_x[2]], [average\_x[0], a11, a1, a13], [average\_x[1], a21, a2, a23],  
 [average\_x[2], a31, a3, a33]]  
 b3\_numerator = [[1, average\_x[0], average\_x[1], my], [average\_x[0], a11, a12, a1], [average\_x[1], a21, a22, a2],  
 [average\_x[2], a31, a32, a3]]  
 b\_denominator = [[1, average\_x[0], average\_x[1], average\_x[2]], [average\_x[0], a11, a12, a13],  
 [average\_x[1], a21, a22, a23], [average\_x[2], a31, a32, a33]]  
 b0 = det(b0\_numerator) / det(b\_denominator)  
 b1 = det(b1\_numerator) / det(b\_denominator)  
 b2 = det(b2\_numerator) / det(b\_denominator)  
 b3 = det(b3\_numerator) / det(b\_denominator)  
 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] - average\_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("(ಠ‿ಠ)﻿" \* 8)  
 print("—" \* 50)  
 print("Критерій Кохрена:")  
 Gt = Critical\_values.get\_cohren\_value(f2, f1, q)  
  
 if Gt > Gp or m >= 25:  
 print("Дисперсія однорідна при рівні значимості {:.2f}! Збільшувати m не потрібно.".format(q))  
 print("—" \* 50)  
 odnoridna = True  
 else:  
 print("\t\tДисперсія не однорідна при рівні значимості {:.2f}!".format(q))  
 print("—" \* 50)  
 m += 1  
 if m == 25:  
 exit()  
  
 matrix = []  
  
 for i in range(N):  
 matrix.append(matrix\_3x[i] + matrix\_y[i])  
 print("Матриця 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("—" \* 50)  
 print("Рівняння регресії:")  
 print("{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 = ŷ".format(b0, b1, b2, b3))  
 print("—" \* 50)  
 print("Критерій Стьюдента:")  
  
 beta\_1 = [b0, b1, b2, b3]  
 need\_koef = student\_test(beta\_1)  
 print("{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 = ŷ".format(need\_koef[0],  
 need\_koef[1],  
 need\_koef[2],  
 need\_koef[3]))  
 d = len(need\_koef) - need\_koef.count(0)  
 f4 = N - d  
 print("—" \* 50)  
 print("Критерій Фішера:")  
  
 if not fisher(need\_koef):  
 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(average\_y) / len(average\_y)  
 else:  
 for j in range(7):  
 beta[i] += average\_y[i] \* matrix\_pfe[i][j] / N  
 print("Рівняння 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("—" \* 50)  
 print("Критерій Кохрена:")  
 Gt = Critical\_values.get\_cohren\_value(f2, f1, q)  
  
 if Gt > Gp or m >= 25:  
 print("Дисперсія однорідна при рівні значимості {:.2f}! Збільшувати m не потрібно.".format(q))  
 odnoridna = True  
 else:  
 print("Дисперсія не однорідна при рівні значимості {:.2f}!".format(q))  
 m += 1  
 if m == 25:  
 exit()  
 need\_koef = student\_test(beta, 8)  
 print("—" \* 50)  
 print("Критерій Стьюдента:")  
 print("{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 + {:.3f} \* Х1X2 + {:.3f} \* Х1X3 + {:.3f} \* Х2X3"  
 "+ {:.3f} \* Х1Х2X3= ŷ".format(need\_koef[0], need\_koef[1],  
 need\_koef[2],  
 need\_koef[3],  
 need\_koef[4],  
 need\_koef[5],  
 need\_koef[6],  
 need\_koef[7]))  
 d = len(need\_koef) - need\_koef.count(0)  
 f4 = N - d  
  
 if student\_test(beta, 7):  
 print("Рівняння регресії адекватне стосовно оригіналу.")  
 adekwatna = True  
 else:  
 print("\tРівняння регресії адекватне стосовно оригіналу.")  
 adekwatna = True  
print("—" \* 50)  
print("(◕‿◕)♡" \* 7)

**Результат:**

(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿

——————————————————————————————————

Критерій Кохрена:

Дисперсія однорідна при рівні значимості 0.05! Збільшувати m не потрібно.

——————————————————————————————————

Матриця з натуральних значень факторів:

| X1 X2 X3 Y1 Y2 Y3 |

| -10 -20 50 214 211 235 |

| -10 -20 55 252 208 245 |

| -10 60 50 238 247 252 |

| -10 60 55 220 235 208 |

| 50 -20 50 245 229 244 |

| 50 -20 55 216 253 217 |

| 50 60 50 247 210 207 |

| 50 60 55 242 207 227 |

——————————————————————————————————

Рівняння регресії:

273.604 + -0.029 \* X1 + -0.030 \* X2 + -0.817 \* X3 = ŷ

——————————————————————————————————

Критерій Стьюдента:

273.604 + 0.000 \* X1 + 0.000 \* X2 + -0.817 \* X3 = ŷ

——————————————————————————————————

Критерій Фішера:

Рівняння регресії неадекватне стосовно оригіналу.

Ефект взаємодії!

Рівняння регресії з ефектом взаємодії:

229.542 + -29.375 \* X1 + -30.708 \* X2 + -27.625 \* X3 + -29.917 \* Х1X2 + -28.583 \* Х1X3 + -27.667 \* Х2X3+ 197.167 \* 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+ 197.167 \* Х1Х2X3= ŷ

Рівняння регресії адекватне стосовно оригіналу.

——————————————————————————————————

(◕‿◕)♡(◕‿◕)♡(◕‿◕)♡(◕‿◕)♡(◕‿◕)♡(◕‿◕)♡(◕‿◕)♡

Process finished with exit code 0

**Висновок:** У ході виконання 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.