Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Методи оптимізації та планування експерименту**

Лабораторна робота №3:

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ

ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

Виконав:

студент групи ІО-83

Веселовський Андрій

Залікова книжка № 8304

Перевірив Регіда П. Г.

Київ 2020р.

**Лабораторна робота №3**

**Тема:** ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ

ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

**Мета:** провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти

коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

**Виконання:**

Варіант – 306.

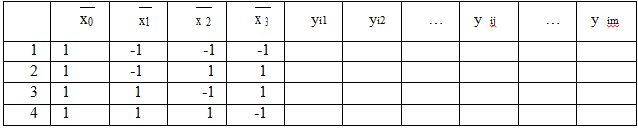
C:\Users\HP\Desktop\3.png

x1min = 10; x2min = 15; x3min = 10;

x1max = 40; x2max = 50; x3max = 30;

Ymin = 212 Ymax = 240

Нормована матриця планування:



1. Лістинг програми:
2. from \_pydecimal import Decimal  
   from scipy.stats import f, t  
   from random import randrange  
   from math import sqrt, fabs  
     
     
   def det(a):  
    from numpy.linalg import det  
    return det(a)  
     
     
   class Critical\_values:  
    @staticmethod  
    def get\_cohren\_value(size\_of\_selections, qty\_of\_selections, significance):  
    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):  
    return Decimal(abs(t.ppf(significance / 2, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
     
    @staticmethod  
    def get\_fisher\_value(f3, f4, significance):  
    return Decimal(abs(f.isf(significance, f4, f3))).quantize(Decimal('.0001')).\_\_float\_\_()  
     
     
   print("--Рівняння регресії матиме вигляд: ŷ = b0 + b1\*X1 + b2\*X2 + b3\*X3--")  
   print("--Матриця планування експеременту--")  
   matrix\_pfe = [[1, -1, -1, -1], [1, -1, 1, 1], [1, 1, -1, 1], [1, 1, 1, -1]]  
   for i in range(len(matrix\_pfe)):  
    print("|", end=" ")  
    for j in range(len(matrix\_pfe[i])):  
    print(matrix\_pfe[i][j], end=" ")  
    print("|")  
   x1\_min = 10  
   x1\_max = 40  
   x2\_min = 15  
   x2\_max = 50  
   x3\_min = 10  
   x3\_max = 30  
   y\_min = int((x1\_min + x2\_min + x3\_min) / 3)  
   y\_max = int((x1\_max + x2\_max + x3\_max) / 3)  
   matrix = [[x1\_min, x2\_min, x3\_min], [x1\_min, x2\_max, x3\_max],  
    [x1\_max, x2\_min, x3\_max], [x1\_max, x2\_max, x3\_min]]  
   for i in range(len(matrix)):  
    for j in range(3):  
    matrix[i].append(randrange(y\_min, y\_max))  
     
   print("--Матриця з натуральних значень факторів--")  
   for i in range(len(matrix)):  
    print("|", end=" ")  
    for j in range(len(matrix[i])):  
    print(matrix[i][j], end=" ")  
    print("|")  
     
   my1 = sum(matrix[0][3:]) / 3  
   my2 = sum(matrix[1][3:]) / 3  
   my3 = sum(matrix[2][3:]) / 3  
   my4 = sum(matrix[3][3:]) / 3  
   my = (my1 + my2 + my3 + my4) / 4  
   mx1 = (matrix[0][0] + matrix[1][0] + matrix[2][0] + matrix[3][0]) / 4  
   mx2 = (matrix[0][1] + matrix[1][1] + matrix[2][1] + matrix[3][1]) / 4  
   mx3 = (matrix[0][2] + matrix[1][2] + matrix[2][2] + matrix[3][2]) / 4  
   a1 = (matrix[0][0] \* my1 + matrix[1][0] \* my2 + matrix[2][0] \* my3 + matrix[3][0] \* my4) / 4  
   a2 = (matrix[0][1] \* my1 + matrix[1][1] \* my2 + matrix[2][1] \* my3 + matrix[3][1] \* my4) / 4  
   a3 = (matrix[0][2] \* my1 + matrix[1][2] \* my2 + matrix[2][2] \* my3 + matrix[3][2] \* my4) / 4  
   a11 = (matrix[0][0] 2 + matrix[1][0] 2 + matrix[2][0] 2 + matrix[3][0] 2) / 4  
   a22 = (matrix[0][1] 2 + matrix[1][1] 2 + matrix[2][1] 2 + matrix[3][1] 2) / 4  
   a33 = (matrix[0][2] 2 + matrix[1][2] 2 + matrix[2][2] 2 + matrix[3][2] 2) / 4  
   a12 = (matrix[0][0] \* matrix[0][1] + matrix[1][0] \* matrix[1][1] + matrix[2][0] \* matrix[2][1] +  
    matrix[3][0] \* matrix[3][1]) / 4  
   a21 = a12  
   a13 = (matrix[0][0] \* matrix[0][2] + matrix[1][0] \* matrix[1][2] + matrix[2][0] \* matrix[2][2] +  
    matrix[3][0] \* matrix[3][2]) / 4  
   a31 = a13  
   a23 = (matrix[0][1] \* matrix[0][2] + matrix[1][1] \* matrix[1][2] + matrix[2][1] \* matrix[2][2] +  
    matrix[3][1] \* matrix[3][2]) / 4  
   a32 = a23  
     
   b0\_numerator = [[my, mx1, mx2, mx3], [a1, a11, a12, a13], [a2, a21, a22, a23], [a3, a31, a32, a33]]  
   b1\_numerator = [[1, my, mx2, mx3], [mx1, a1, a12, a13], [mx2, a2, a22, a23], [mx3, a3, a32, a33]]  
   b2\_numerator = [[1, mx1, my, mx3], [mx1, a11, a1, a13], [mx2, a21, a2, a23], [mx3, a31, a3, a33]]  
   b3\_numerator = [[1, mx1, mx2, my], [mx1, a11, a12, a1], [mx2, a21, a22, a2], [mx3, a31, a32, a3]]  
   b\_denominator = [[1, mx1, mx2, mx3], [mx1, a11, a12, a13], [mx2, a21, a22, a23], [mx3, 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)  
     
   print("--Рівняння регресії--")  
   print("{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 = ŷ".format(b0, b1, b2, b3))  
   print("--Перевірка--")  
   print("{:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} = ".format(b0, b1, x1\_min, b2, x2\_min, b3, x3\_min)  
    + str(b0 + b1 \* x1\_min + b2 \* x2\_min + b3 \* x3\_min))  
   print("{:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} = ".format(b0, b1, x1\_min, b2, x2\_max, b3, x3\_max)  
    + str(b0 + b1 \* x1\_min + b2 \* x2\_max + b3 \* x3\_max))  
   print("{:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} = ".format(b0, b1, x1\_max, b2, x2\_min, b3, x3\_max)  
    + str(b0 + b1 \* x1\_max + b2 \* x2\_min + b3 \* x3\_max))  
   print("{:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} = ".format(b0, b1, x1\_max, b2, x2\_max, b3, x3\_min)  
    + str(b0 + b1 \* x1\_max + b2 \* x2\_max + b3 \* x3\_min))  
   dispersion\_y1 = ((matrix[0][3] - my1) 2 + (matrix[0][4] - my1) 2 + (matrix[0][5] - my1) \*\* 2) / 3  
   dispersion\_y2 = ((matrix[1][3] - my2) 2 + (matrix[1][4] - my2) 2 + (matrix[1][5] - my2) \*\* 2) / 3  
   dispersion\_y3 = ((matrix[2][3] - my3) 2 + (matrix[2][4] - my3) 2 + (matrix[2][5] - my3) \*\* 2) / 3  
   dispersion\_y4 = ((matrix[3][3] - my4) 2 + (matrix[3][4] - my4) 2 + (matrix[3][5] - my4) \*\* 2) / 3  
   dispersion\_lst = [dispersion\_y1, dispersion\_y2, dispersion\_y3, dispersion\_y4]  
   Gp = max(dispersion\_lst) / sum(dispersion\_lst)  
   f1 = 2  
   f2 = 4  
   q = 0.05  
   print("--Критерій Кохрена--")  
   Gt = Critical\_values.get\_cohren\_value(f2, f1, q)  
   if Gt > Gp:  
    print("--Дисперсія однорідна--")  
   else:  
    print("--Дисперсія не однорідна--")  
     
   print("--Критерій Стьюдента--")  
   S\_2b = (dispersion\_y1 + dispersion\_y2 + dispersion\_y3 + dispersion\_y4) / 4  
   S\_2b /= 12  
   S\_b = sqrt(S\_2b)  
   beta\_0 = (my1 + my2 + my3 + my4) / 4  
   beta\_1 = (-my1 - my2 + my3 + my4) / 4  
   beta\_2 = (-my1 + my2 - my3 + my4) / 4  
   beta\_3 = (-my1 + my2 + my3 - my4) / 4  
   t\_0 = beta\_0 / S\_b  
   t\_1 = beta\_1 / S\_b  
   t\_2 = beta\_2 / S\_b  
   t\_3 = beta\_3 / S\_b  
   tt = Critical\_values.get\_student\_value(f1 \* f2, q)  
   t\_lst = [fabs(t\_0), fabs(t\_1), fabs(t\_2), fabs(t\_3)]  
   b\_lst = [b0, b1, b2, b3]  
   for i in range(4):  
    if t\_lst[i] > tt:  
    continue  
    else:  
    t\_lst[i] = 0  
   for j in range(4):  
    if t\_lst[j] != 0:  
    continue  
    else:  
    b\_lst[j] = 0  
   print("--Перевірка значемих коефіціентів--")  
   yj1 = b\_lst[0] + b\_lst[1] \* x1\_min + b\_lst[2] \* x2\_min + b\_lst[3] \* x3\_min  
   yj2 = b\_lst[0] + b\_lst[1] \* x1\_min + b\_lst[2] \* x2\_max + b\_lst[3] \* x3\_max  
   yj3 = b\_lst[0] + b\_lst[1] \* x1\_max + b\_lst[2] \* x2\_min + b\_lst[3] \* x3\_max  
   yj4 = b\_lst[0] + b\_lst[1] \* x1\_max + b\_lst[2] \* x2\_max + b\_lst[3] \* x3\_min  
   print("{:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} = "  
    "".format(b\_lst[0], b\_lst[1], x1\_min, b\_lst[2], x2\_min, b\_lst[3],  
    x3\_min) + str(yj1))  
   print("{:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} = "  
    "".format(b\_lst[0], b\_lst[1], x1\_min, b\_lst[2], x2\_max, b\_lst[3],  
    x3\_max) + str(yj2))  
   print("{:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} = "  
    "".format(b\_lst[0], b\_lst[1], x1\_max, b\_lst[2], x2\_min, b\_lst[3],  
    x3\_max) + str(yj3))  
   print("{:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} + {:.3f} \* {:.3f} = "  
    "".format(b\_lst[0], b\_lst[1], x1\_max, b\_lst[2], x2\_max, b\_lst[3],  
    x3\_min) + str(yj4))  
   print("--Критерій Фішера--")  
   for i in range(3):  
    if b\_lst[i] == 0:  
    del b\_lst[i]  
     
   d = len(b\_lst)  
   f4 = 4 - d  
   S\_2ad = 3 \* ((yj1 - my1)\*\*2 + (yj2 - my2)\*\*2 + (yj3 - my3)\*\*2 + (yj4 - my4)\*\*2) / f4  
   Fp = S\_2ad / S\_2b  
   Ft = Critical\_values.get\_fisher\_value(f1 \* f2, f4, q)  
   if Fp > Ft:  
    print("Pівняння регресії неадекватно оригіналу при рівні значимості 0.05")  
   else:  
    print("Pівняння регресії адекватно оригіналу при рівні значимості 0.05"
3. Результат виконання роботи програми:

--Рівняння регресії матиме вигляд: ŷ = b0 + b1\*X1 + b2\*X2 + b3\*X3--  
--Матриця планування експеременту--  
| 1 -1 -1 -1 |  
| 1 -1 1 1 |  
| 1 1 -1 1 |  
| 1 1 1 -1 |  
--Матриця з натуральних значень факторів--  
| 10 15 10 22 25 23 |  
| 10 50 30 36 34 21 |  
| 40 15 30 16 28 11 |  
| 40 50 10 34 38 13 |  
--Рівняння регресії--  
21.607 + -0.117 \* X1 + 0.243 \* X2 + -0.075 \* X3 = ŷ  
--Перевірка--  
21.607 + -0.117 \* 10.000 + 0.243 \* 15.000 + -0.075 \* 10.000 = 23.333333333333304  
21.607 + -0.117 \* 10.000 + 0.243 \* 50.000 + -0.075 \* 30.000 = 30.333333333333286  
21.607 + -0.117 \* 40.000 + 0.243 \* 15.000 + -0.075 \* 30.000 = 18.33333333333328  
21.607 + -0.117 \* 40.000 + 0.243 \* 50.000 + -0.075 \* 10.000 = 28.333333333333293  
--Критерій Кохрена--  
--Дисперсія однорідна--  
--Критерій Стьюдента--  
--Перевірка значемих коефіціентів--  
21.607 + 0.000 \* 10.000 + 0.000 \* 15.000 + 0.000 \* 10.000 = 21.60714285714284  
21.607 + 0.000 \* 10.000 + 0.000 \* 50.000 + 0.000 \* 30.000 = 21.60714285714284  
21.607 + 0.000 \* 40.000 + 0.000 \* 15.000 + 0.000 \* 30.000 = 21.60714285714284  
21.607 + 0.000 \* 40.000 + 0.000 \* 50.000 + 0.000 \* 10.000 = 21.60714285714284  
--Критерій Фішера--  
Pівняння регресії неадекватно оригіналу при рівні значимості 0.05

**Контрольні запитання**

1. *Що називається дробовим факторним експериментом?*

У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Якщо буде використовуватися лінійна регресія, то можливо зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів слід скоротити, використовуючи для планування так звані регулярні дробові репліки від повного факторного експерименту, що містять відповідну кількість дослідів і зберігають основні властивості матриці планування – це означає дробовий факторний експеримент (ДФЕ).

1. *Для чого потрібно розрахункове значення Кохрена?*

Для перевірки дисперсії на однорідність.

1. *Для чого перевіряється критерій Стьюдента?*

Для перевірки значущості коефіцієнтів регресії. Тобто, Якщо виконується нерівність *ts*< *tтабл*, то приймається нуль-гіпотеза, тобто вважається, що знайдений коефіцієнт *βs* є статистично незначущим і його слід виключити з рівняння регресії. Якщо *ts* > *tтабл* то гіпотеза не підтверджується, тобто *βs* – значимий коефіцієнт і він залишається в рівнянні регресії.

1. *Чим визначається критерій Фішера і як його застосовувати?*

Отримане рівняння регресії необхідно перевірити на адекватність досліджуваному об'єкту. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності. Адекватність моделі перевіряють за F-критерієм Фішера, який дорівнює відношенню дисперсії адекватності до дисперсії відтворюваності.

**Висновок:** Отже, у ході виконання лабораторної роботи № 3 провели дробовий трьохфакторний експеримент. 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!