Міністерство освіти і науки України
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»

Виконав:

Гречаний Є.О.

Перевірив:

ас. Регіда П. Г.

Київ 2021

Тема: Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами

Мета: Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

Завдання:

Завдання до лабораторної роботи:

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x_1, x_2, x_3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1;+ l; - l; 0 для $\bar{x}_1$, $\bar{x}_2$, $\bar{x}_3$.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + \text{random}(10) - 5,$$

де $f(x_1, x_2, x_3)$ вибирається по номеру в списку в журналі викладача.

4. Провести експерименти і аналізуючи значення статистичних перевірок, отримати адекватну модель рівняння регресії. При розрахунках використовувати натуральні значення факторів.
5. Зробити висновки по виконаній роботі.

Алгоритм отримання адекватної моделі рівняння регресії

1) *Вибір рівняння регресії (лінійна форма, рівняння з урахуванням ефекту взаємодії і з урахуванням квадратичних членів);*
2) *Вибір кількості повторів кожної комбінації (m = 2);*
3) *Складення матриці планування експерименту і вибір кількості рівнів (N)*
4) *Проведення експериментів;*
5) *Перевірка однорідності дисперсії. Якщо не однорідна – повертаємося на п. 2 і збільшуємо m на 1);*
6) *Розрахунок коефіцієнтів рівняння регресії. При розрахунку використовувати* ***натуральні*** *значення x_1, x_2 и x_3.*
7) *Перевірка нуль-гіпотези. Визначення значимих коефіцієнтів;*
8) *Перевірка адекватності моделі рівняння оригіналу. При неадекватності – повертаємося на п.1, змінивши при цьому рівняння регресії;*

Програмний код

```
import math
import numpy
import random
from _decimal import Decimal
from functools import reduce
from itertools import compress
from scipy.stats import f, t

xMin = [-20, -25, -25]
xMax = [30, 10, -20]
x0 = [(xMax[_] + xMin[_]) / 2 for _ in range(3)]
dx = [xMax[_] - x0[_] for _ in range(3)]
norm_plan_raw = [[-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],
                 [+1.73, 0, 0],
                 [0, -1.73, 0],
```

```
                [0, +1.73, 0],
                [0, 0, -1.73],
                [0, 0, +1.73]]

naturalPlanRaw = [[xMin[0], xMin[1], xMin[2]],
                  [xMin[0], xMin[1], xMax[2]],
                  [xMin[0], xMax[1], xMin[2]],
                  [xMin[0], xMax[1], xMax[2]],
                  [xMax[0], xMin[1], xMin[2]],
                  [xMax[0], xMin[1], xMax[2]],
                  [xMax[0], xMax[1], xMin[2]],
                  [xMax[0], xMax[1], xMax[2]],
                  [-1.73 * dx[0] + x0[0], x0[1], x0[2]],
                  [1.73 * dx[0] + x0[0], x0[1], x0[2]],
                  [x0[0], -1.73 * dx[1] + x0[1], x0[2]],
                  [x0[0], 1.73 * dx[1] + x0[1], x0[2]],
                  [x0[0], x0[1], -1.73 * dx[2] + x0[2]],
                  [x0[0], x0[1], 1.73 * dx[2] + x0[2]],
                  [x0[0], x0[1], x0[2]]]


# Основні функції
def equationRegres(x1, x2, x3, coefficients, importance=[True] * 11):
    factors_array = [1, x1, x2, x3, x1 * x1, x2 * x2, x3 * x3, x1 * x2, x1
* x3, x2 * x3, x1 * x2 * x3]
    return sum([el[0] * el[1] for el in compress(zip(coefficients,
factors_array), importance)])


def func(x1, x2, x3):
    coefficients = [3.7, 6.3, 8.8, 5.9, 8.9, 0.4, 8.8, 4.4, 0.6, 2, 8]
    return equationRegres(x1, x2, x3, coefficients)


def generateFactorsTable(rawArr):
    raw_list = [row + [row[0] * row[1], row[0] * row[2], row[1] * row[2],
row[0] * row[1] * row[2]] + list(
        map(lambda x: x ** 2, row)) for row in rawArr]
    return list(map(lambda row: list(map(lambda el: round(el, 3), row)),
raw_list))


def generateY(m, factorsTable):
    return [[round(func(row[0], row[1], row[2]) + random.randint(-5, 5), 3)
for _ in range(m)] for row in factorsTable]


# Вивід результатів
def printMatrix(m, n, factors, valsY, additionalText=":"):
    labels_table = list(map(lambda x: x.ljust(10),
                            ["x1", "x2", "x3", "x12", "x13", "x23", "x123",
"x1^2", "x2^2", "x3^2"] + [
                                "y{}".format(i + 1) for i in range(m)]))
    rows_table = [list(factors[i]) + list(valsY[i]) for i in range(n)]
    print("\nМатриця планування" + additionalText)
    print(" ".join(labels_table))
    print("\n".join([" ".join(map(lambda j: "{:<+10}".format(j),
rows_table[i])) for i in range(len(rows_table))]))
    print("\t")


def printEquation(coefficients, importance=[True] * 11):
```

```
    XiNames = list(compress(["", "x1", "x2", "x3", "x12", "x13", "x23",
"x123", "x1^2", "x2^2", "x3^2"], importance))
    coefficientsToPrint = list(compress(coefficients, importance))
    equation = " ".join(
        ["".join(i) for i in zip(list(map(lambda x: "{:+.2f}".format(x),
coefficientsToPrint)), XiNames)])
    print("Рівняння регресії: y = " + equation)


def setFactorsTable(factorsTable):
    def x_i(i):
        withNullFactor = list(map(lambda x: [1] + x,
generateFactorsTable(factorsTable)))
        res = [row[i] for row in withNullFactor]
        return numpy.array(res)

    return x_i


def Mij(*arrays):
    return numpy.average(reduce(lambda accum, el: accum * el,
list(map(lambda el: numpy.array(el), arrays))))


def findCoeffs(factors, y_values):
    Xi = setFactorsTable(factors)
    coefficients = [[Mij(Xi(column), Xi(row)) for column in range(11)] for
row in range(11)]
    numpyY = list(map(lambda row: numpy.average(row), y_values))
    freeVal = [Mij(numpyY, Xi(i)) for i in range(11)]
    betaCoeffs = numpy.linalg.solve(coefficients, freeVal)
    return list(betaCoeffs)


# Критерії
def cochranCriteria(m, n, y_table):
    def getCochranVal(f1, f2, q):
        part_result1 = q / f2
        params = [part_result1, f1, (f2 - 1) * f1]
        fisher = f.isf(*params)
        result = fisher / (fisher + (f2 - 1))
        return Decimal(result).quantize(Decimal('.0001')).__float__()

    print("Перевірка рівномірності дисперсій за критерієм Кохрена: m = {},
N = {}".format(m, n))
    variationsY = [numpy.var(i) for i in y_table]
    variationMaxY = max(variationsY)
    gp = variationMaxY / sum(variationsY)
    f1 = m - 1
    f2 = n
    p = 0.95
    q = 1 - p
    gt = getCochranVal(f1, f2, q)
    print("Gp = {}; Gt = {}; f1 = {}; f2 = {}; q = {:.2f}".format(gp, gt,
f1, f2, q))
    if gp < gt:
        print("Gp < Gt => дисперсії рівномірні - все правильно")
        return True
    else:
        print("Gp > Gt => дисперсії нерівномірні - треба ще експериментів")
        return False
```

```
def studentCriteria(m, n, y_table, betaCoeffs):
    def getStudentVal(f3, q):
        return Decimal(abs(t.ppf(q / 2,
f3))).quantize(Decimal('.0001')).__float__()

    print("\nПеревірка значимості коефіцієнтів регресії за критерієм
Стьюдента: m = {}, N = {} ".format(m, n))
    averageVariation = numpy.average(list(map(numpy.var, y_table)))
    variationBetaS = averageVariation / n / m
    standardDeviationBetaS = math.sqrt(variationBetaS)
    Ti = [abs(betaCoeffs[i]) / standardDeviationBetaS for i in
range(len(betaCoeffs))]
    f3 = (m - 1) * n
    q = 0.05
    ourT = getStudentVal(f3, q)
    importance = [True if el > ourT else False for el in list(Ti)]
    # print result data
    print("Оцінки коефіцієнтів βs: " + ", ".join(list(map(lambda x:
str(round(float(x), 3)), betaCoeffs))))
    print("Коефіцієнти ts: " + ", ".join(list(map(lambda i:
"{:.2f}".format(i), Ti))))
    print("f3 = {}; q = {}; tтабл = {}".format(f3, q, ourT))
    betaI = ["β0", "β1", "β2", "β3", "β12", "β13", "β23", "β123", "β11",
"β22", "β33"]
    importanceToPrint = ["важливий" if i else "неважливий" for i in
importance]
    toPrint = map(lambda x: x[0] + " " + x[1], zip(betaI,
importanceToPrint))
    print(*toPrint, sep="; ")
    printEquation(betaCoeffs, importance)
    return importance


def fisherCriteria(m, N, d, tableX, tableY, coeffsB, importance):
    def getFisherVal(f3, f4, q):
        return Decimal(abs(f.isf(q, f4,
f3))).quantize(Decimal('.0001')).__float__()

    f3 = (m - 1) * N
    f4 = N - d
    q = 0.05
    theorY = numpy.array([equationRegres(row[0], row[1], row[2], coeffsB)
for row in tableX])
    avgY = numpy.array(list(map(lambda el: numpy.average(el), tableY)))
    Sad = m / (N - d) * sum((theorY - avgY) ** 2)
    variationsY = numpy.array(list(map(numpy.var, tableY)))
    Sv = numpy.average(variationsY)
    Fp = float(Sad / Sv)
    Ft = getFisherVal(f3, f4, q)
    theoreticalValsToPrint = list(
        zip(map(lambda x: "x1 = {0[1]:<10} x2 = {0[2]:<10} x3 =
{0[3]:<10}".format(x), tableX), theorY))
    print("\nПеревірка адекватності моделі за критерієм Фішера: m = {}, N =
{} для таблиці y_table".format(m, N))
    print("Теоретичні значення y для різних комбінацій факторів:")
    print("\n".join(["{arr[0]}: y = {arr[1]}".format(arr=el) for el in
theoreticalValsToPrint]))
    print("Fp = {}, Ft = {}".format(Fp, Ft))
    print("Fp < Ft => модель адекватна" if Fp < Ft else "Fp > Ft => модель
неадекватна")
    return True if Fp < Ft else False
```

```python
def main(m, n):
    naturalPlan = generateFactorsTable(naturalPlanRaw)
    arrY = generateY(m, naturalPlanRaw)
    while not cochranCriteria(m, n, arrY):
        m += 1
        arrY = generateY(m, naturalPlan)

    printMatrix(m, n, naturalPlan, arrY, " для натуралізованих факторів:")
    coefficients = findCoeffs(naturalPlan, arrY)
    printEquation(coefficients)
    importance = studentCriteria(m, n, arrY, coefficients)
    d = len(list(filter(None, importance)))
    fisherCriteria(m, n, d, naturalPlan, arrY, coefficients, importance)


if __name__ == "__main__":
    m = 3
    n = 15
    main(m, n)
    print("Виконав студент шрупи ІО-92, Гречаний Євгеній")
```

Результати

```
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15
Gp = 0.14875630252100838; Gt = 0.3346; f1 = 2; f2 = 15; q = 0.05
Gp < Gt => дисперсії рівномірні - все правильно

Матриця планування для натуралізованих факторів:
x1      x2      x3      x12     x13     x23     x123    x1^2    x2^2    x3^2    y1          y2          y3
-20     -25     -25     +500    +500    +625    -12500  +400    +625    +625    -87434.8    -87427.8    -87428.8
-20     -25     -20     +500    +400    +500    -10000  +400    +625    +400    -69685.3    -69691.3    -69687.3
-20     +10     -25     -200    +500    -250    +5000   +400    +100    +625    +47838.2    +47843.2    +47838.2
-20     +10     -20     -200    +400    -200    +4000   +400    +100    +400    +37929.7    +37928.7    +37923.7
+30     -25     -25     -750    -750    +625    +18750  +900    +625    +625    +161884.2   +161898.2   +161883.2
+30     -25     -20     -750    -600    +500    +15000  +900    +625    +400    +128974.7   +128976.7   +128969.7
+30     +10     -25     +300    -750    -250    -7500   +900    +100    +625    -45943.8    -45947.8    -45949.8
+30     +10     -20     +300    -600    -200    -6000   +900    +100    +400    -35703.3    -35706.3    -35718.3
-38.25  -7.5     -22.5   +286.875  +860.625  +168.75   -6454.688  +1463.062  +56.25    +506.25   -32465.644  -32462.644  -32460.644
+48.25  -7.5     -22.5   -361.875  -1085.625 +168.75   +8142.188  +2328.062  +56.25    +506.25   +88548.556  +88542.556  +88542.556
+5.0    -37.775  -22.5   -188.875  -112.5    +849.938  +4249.688  +25.0     +1426.951 +506.25   +39628.135  +39612.135  +39613.135
+5.0    +22.775  -22.5   +113.875  -112.5    -512.438  -2562.188  +25.0     +518.701  +506.25   -16096.975  -16097.975  -16101.975
+5.0    -7.5     -26.825  -37.5     -134.125  +201.188  +1005.938  +25.0     +56.25    +719.581  +14587.642  +14596.642  +14589.642
+5.0    -7.5     -18.175  -37.5     -90.875   +136.312  +681.562   +25.0     +56.25    +330.331  +8520.477   +8522.477   +8521.477
+5.0    -7.5     -22.5    -37.5     -112.5    +168.75   +843.75    +25.0     +56.25    +506.25   +11396.45   +11390.45   +11389.45

Рівняння регресії: y = +8.13 +6.41x1 +8.57x2 +5.41x3 +4.42x12 +8.68x13 +1.99x23 +8.88x123 +8.98x1^2 +0.48x2^2 +8.79x3^2

Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 8.13, 6.407, 8.57, 5.413, 4.42, 8.683, 1.989, 8.881, 8.9, 8.399, 8.787
```

```
f3 = 30; q = 0.05; ttabn = 2.0423
β0 незначний; β1 значимий; β2 значимий; β3 значимий; β12 значимий; β13 незначимий; β23 значимий; β123 значимий; β11 значимий; β22 незначимий; β33 значимий
Рівняння регресії: y = +6.41x1 +8.57x2 +5.41x3 +4.42x12 +1.99x23 +8.88x123 +8.98x1^2 +8.79x3^2

Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = -25        x2 = -25        x3 = 500        : y = -98224.83485938124
x1 = -25        x2 = -20        x3 = 500        : y = -77617.48438165749
x1 = 10         x2 = -25        x3 = -200       : y = 49577.51109963462
x1 = 10         x2 = -20        x3 = -200       : y = 39588.28884363272
x1 = -25        x2 = -25        x3 = -750       : y = 157764.72951195878
x1 = -25        x2 = -20        x3 = -750       : y = 125600.97996520877
x1 = 10         x2 = -25        x3 = 300        : y = -64851.842564255144
x1 = 10         x2 = -20        x3 = 300        : y = -50736.41783615466
x1 = -7.5       x2 = -22.5      x3 = 286.875    : y = -39617.15389383893
x1 = -7.5       x2 = -22.5      x3 = -361.875   : y = 70587.78693449196
x1 = -37.775    x2 = -22.5      x3 = -188.875   : y = 36732.86248421093
x1 = 22.775     x2 = -22.5      x3 = 113.875    : y = -21272.25535427496
x1 = -7.5       x2 = -26.825    x3 = -37.5      : y = 8823.888959826486
x1 = -7.5       x2 = -18.175    x3 = -37.5      : y = 5605.388695959832
x1 = -7.5       x2 = -22.5      x3 = -37.5      : y = 7177.190223875564
Fp = 76636919.84K726, Ft = 2.3343
Fp > Ft => модель неадекватна
Виконав студент шрупи ІО-92, Гречаний Євгеній
```