Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Методи оптимизації та планування експерименту

Лабораторна робота №2

Виконав:
студент групи ІО-91
Дикун А.В.
Заліковакнижка № 9110
Варінт № 8
Перевірив: Регіда П. Г.

Київ 2021р

Текст програми:

```
import math
from random import randint
import numpy as np

probList = (0.99, 0.98, 0.95, 0.90)
rkrTable = {2: (1.73, 1.72, 1.71, 1.69),
            6: (2.16, 2.13, 2.10, 2.00),
            8: (2.43, 4.37, 2.27, 2.17),
            10: (2.62, 2.54, 2.41, 2.29),
            12: (2.75, 2.66, 2.52, 2.39),
            15: (2.9, 2.8, 2.64, 2.49),
            20: (3.08, 2.96, 2.78, 2.62)}
numOfExperiments = 5


def main():
    global numOfExperiments
    MIN_Y_LIMIT, MAX_Y_LIMIT = 120, 220
    X1Min, X1MinNorm = -30, -1
    X1Max, X1MaxNorm = 0, 1
    X2Min, X2MinNorm = -35, -1
    X2Max, X2MaxNorm = 10, 1

    def checkRegression():
        NORM_Y_1 = round(b0 - b1 - b2, 1)
        NORM_Y_2 = round(b0 + b1 - b2, 1)
        NORM_Y_3 = round(b0 - b1 + b2, 1)
        if NORM_Y_1 == averageYArr[0] and NORM_Y_2 == averageYArr[1] and
NORM_Y_3 == averageYArr[2]:
            return True
        else:
            return False

    def checkRegressionStr():
        if checkRegression():
            return "Значення перевірки нормаваного рівняння регресії
сходяться"
        else:
            return "Значення перевірки нормаваного рівняння регресії не
сходяться"

    def naturalizedRegression(x1, x2):
        return aa0 + aa1 * x1 + aa2 * x2

    def uniformityDispersion():
        m = min(rkrTable, key=lambda x: abs(x - numOfExperiments))
        p = 0
        for ruv in (Ruv1, Ruv2, Ruv3):
            if ruv > rkrTable[m][0]:
                raise Exception("Потрібно більше експерементів")
            for rkr in range(len(rkrTable[m])):
                if ruv < rkrTable[m][rkr]:
                    p = rkr
        return probList[p]

    def findFuv(u, v):
        if u >= v:
            return u / v
        else:
            return v / u
```

```
    def printResults():
        for i in range(3):
            print(f"Y{i + 1}: {yMatrix[i]}, Average: {averageYArr[i]}")
        print(f"\nσ² y1: {sigma2_1:.3f}\nσ² y2: {sigma2_2:.3f}\nσ² y3:
{sigma2_3:.3f}")
        print(f"σθ = {sigmaTeta:.5f}")
        print(f"\nFuv1 = {Fuv1:.5f}\nFuv2 = {Fuv2:.5f}\nFuv3 = {Fuv3:.5f}\n")
        print(f"θuv1 = {tetaUV1:.5f}\nθuv2 = {tetaUV2:.5f}\nθuv3 =
{tetaUV3:.5f}\n")
        print(f"Ruv1 = {Ruv1:.5f}\nRuv2 = {Ruv2:.5f}\nRuv3 = {Ruv3:.5f}\n")
        print(f"Однорідна дисперсія: {uniformityDispersion()}")
        print(f"\nmx1: {mX1:.5f}\nmx2: {mX2:.5f}\nmy: {mY:.3f}")
        print(f"\na1: {_a1:.5f}\na2: {_a2:.5f}\na3: {_a3:.5f}\na11:
{_a11:.5f}\na22: {_a22:.5f}")
        print(f"\nb0: {b0:.3f}\nb1: {b1:.5f}\nb2: {b2:.5f}")
        print("\nНатуралізація коефіцієнтів:")
        print(f"Δx1: {dX1}\nΔx2: {dX2}")
        print(f"x10: {x10}\nx20: {x20}")
        print(f"\na0: {aa0:.3f}\na1: {aa1:.5f}\na2: {aa2:.5f}\n")

        print(f"Натуралізоване рівняння регресії: {naturalizedRegression}")
        if naturalizedRegression == averageYArr:
            print("Коефіцієнти натуралізованого рівняння регресії вірні")
        else:
            print("Коефіцієнти натуралізованого рівняння регресії НЕ вірні")
        print(checkRegressionStr())

    yMatrix = [[randint(MIN_Y_LIMIT, MAX_Y_LIMIT) for i in
range(numOfExperiments)] for j in range(3)]

    averageYArr = [sum(yMatrix[i][j] for j in range(numOfExperiments)) /
numOfExperiments for i in range(3)]

    sigma2_1 = sum([(j - averageYArr[0]) ** 2 for j in yMatrix[0]]) /
numOfExperiments
    sigma2_2 = sum([(j - averageYArr[1]) ** 2 for j in yMatrix[1]]) /
numOfExperiments
    sigma2_3 = sum([(j - averageYArr[2]) ** 2 for j in yMatrix[2]]) /
numOfExperiments

    sigmaTeta = math.sqrt((2 * (2 * numOfExperiments - 2)) /
(numOfExperiments * (numOfExperiments - 4)))

    Fuv1 = findFuv(sigma2_1, sigma2_2)
    Fuv2 = findFuv(sigma2_3, sigma2_1)
    Fuv3 = findFuv(sigma2_3, sigma2_2)

    tetaUV1 = ((numOfExperiments - 2) / numOfExperiments) * Fuv1
    tetaUV2 = ((numOfExperiments - 2) / numOfExperiments) * Fuv2
    tetaUV3 = ((numOfExperiments - 2) / numOfExperiments) * Fuv3

    Ruv1 = abs(tetaUV1 - 1) / sigmaTeta
    Ruv2 = abs(tetaUV2 - 1) / sigmaTeta
    Ruv3 = abs(tetaUV3 - 1) / sigmaTeta

    mX1 = (X1MinNorm + X1MaxNorm + X1MinNorm) / 3
    mX2 = (X2MinNorm + X2MinNorm + X2MaxNorm) / 3
    mY = sum(averageYArr) / 3
    _a1 = (X1MinNorm ** 2 + X1MaxNorm ** 2 + X1MinNorm ** 2) / 3
    _a2 = (X1MinNorm * X2MinNorm + X1MaxNorm * X2MinNorm + X1MinNorm *
X2MaxNorm) / 3
    _a3 = (X2MinNorm ** 2 + X2MinNorm ** 2 + X2MaxNorm ** 2) / 3
    _a11 = (X1MinNorm * averageYArr[0] + X1MaxNorm * averageYArr[1] +
X1MinNorm * averageYArr[2]) / 3
```

```
    _a22 = (X2MinNorm * averageYArr[0] + X2MinNorm * averageYArr[1] +
X2MaxNorm * averageYArr[2]) / 3

    b0 = np.linalg.det(np.dot([[mY, mX1, mX2],
                               [_a11, _a1, _a2],
                               [_a22, _a2, _a3]],
                              np.linalg.inv([[1, mX1, mX2],
                                             [mX1, _a1, _a2],
                                             [mX2, _a2, _a3]])))

    b1 = np.linalg.det(np.dot([[1, mY, mX2],
                               [mX1, _a11, _a2],
                               [mX2, _a22, _a3]],
                              np.linalg.inv([[1, mX1, mX2],
                                             [mX1, _a1, _a2],
                                             [mX2, _a2, _a3]])))

    b2 = np.linalg.det(np.dot([[1, mX1, mY],
                               [mX1, _a1, _a11],
                               [mX2, _a2, _a22]],
                              np.linalg.inv([[1, mX1, mX2],
                                             [mX1, _a1, _a2],
                                             [mX2, _a2, _a3]])))

    dX1 = math.fabs(X1Max - X1Min) / 2
    dX2 = math.fabs(X2Max - X2Min) / 2
    x10 = (X1Max + X1Min) / 2
    x20 = (X2Max + X2Min) / 2

    aa0 = b0 - b1 * x10 / dX1 - b2 * x20 / dX2
    aa1 = b1 / dX1
    aa2 = b2 / dX2

    naturalizedRegression = [round(naturalizedRegression(X1Min, X2Min), 2),
                             round(naturalizedRegression(X1Max, X2Min), 2),
                             round(naturalizedRegression(X1Min, X2Max), 2)]
    try:
        uniformityDispersion()
    except Exception as e:
        print(e)
        print("Збільшуємо кількість експерементів")
        numOfExperiments += 1
        return main()
    printResults()


if __name__ == '__main__':
    main()
```

Результати роботи програми:

```
C:\Users\anvat\Anaconda3\envs\MOPE_labs\python.exe C:/Users/anvat/PycharmProjects/MOPE_labs/Lab2/Lab2.py
Y1: [126, 130, 172, 186, 130], Average: 148.8
Y2: [171, 164, 154, 127, 181], Average: 159.4
Y3: [175, 208, 202, 191, 130], Average: 181.2

σ² y1: 629.760
σ² y2: 340.240
σ² y3: 781.360
σθ = 1.78885

Fuv1 = 1.85093
Fuv2 = 1.24073
Fuv3 = 2.29650

θuv1 = 1.11056
θuv2 = 0.74444
θuv3 = 1.37790

Ruv1 = 0.06180
Ruv2 = 0.14286
Ruv3 = 0.21125
```

```
Однорідна дисперсія: 0.9

mx1: -0.33333
mx2: -0.33333
my: 163.133

a1: 1.00000
a2: -0.33333
a3: 1.00000
a11: -56.86667
a22: -42.33333

b0: 170.300
b1: 5.30000
b2: 16.20000

Натуралізація коефіцієнтів:
Δx1: 15.0
Δx2: 22.5
x10: -15.0
x20: -12.5

a0: 184.600
a1: 0.35333
a2: 0.72000

Натуралізоване рівняння регресії: [148.8, 159.4, 181.2]
Коефіцієнти натуралізованого рівняння регресії вірні
Значення перевірки нормаваного рівняння регресії сходяться

Process finished with exit code 0
```