Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота 2

з дисципліни «Методи оптимізації та планування експерименту»

Виконав:

студент групи ІО-92

Губський Артур

Перевірив:

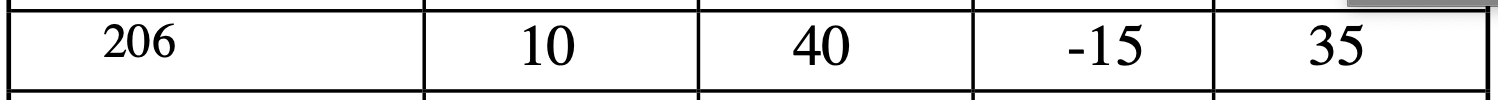
Регіда П.Г.

Київ 2021

# Мета: провести двофакторний експеримент, перевірити однорідність дисперсії за критерієм Романовського, отримати коефіцієнти рівняння регресії, провести натуралізацію рівняння регресії.

ymax = (30 - Nваріанту)\*10 = 240,

ymin = (20 - Nваріанту)\*10 = 140.

Варіант: 

**Роздруківка програми:**

import numpy as np

from typing import List

np.set\_printoptions(precision=3)

class Experiment:

def \_\_init\_\_(self,

X1\_range: List[int],

X2\_range: List[int],

Y\_range: List[int], m: int) -> None:

self.Rcritical = {

5: 2,

6: 2.16,

7: 2.3,

8: 2.43,

9: 2.5

}

self.X1\_range = X1\_range

self.X2\_range = X2\_range

self.Y\_range = Y\_range

self.plan\_matrix = np.array(

[np.random.randint(\*self.X1\_range, size=3),

np.random.randint(\*self.X2\_range, size=3)]).T

self.x0 = [np.mean(self.X1\_range), np.mean(self.X2\_range)]

self.norm\_matrix = self.make\_norm\_plan\_matrix()

self.m = m

self.experiment()

self.b = self.find\_b()

self.a = self.find\_a()

self.check\_b = self.check\_b\_koefs()

self.check\_a = self.check\_a\_koefs()

def experiment(self):

self.y\_matrix = np.random.randint(\*self.Y\_range, size=(3, self.m))

self.y\_mean = np.mean(self.y\_matrix, axis=1)

self.y\_var = np.var(self.y\_matrix, axis=1)

self.sigma = np.sqrt((2 \* (2 \* self.m - 2)) / (self.m \* (self.m - 4)))

if not self.check\_r():

print(f'\n Дісперсія неоднорідна! Змінимо m={self.m} to m={self.m+1}\n')

self.m += 1

self.experiment()

def make\_norm\_plan\_matrix(self) -> np.array:

self.N = self.plan\_matrix.shape[0]

self.k = self.plan\_matrix.shape[1]

interval\_of\_change = [self.X1\_range[1] - self.x0[0],

self.X2\_range[1] - self.x0[1]]

X\_norm = [

[(self.plan\_matrix[i, j] - self.x0[j]) / interval\_of\_change[j]

for j in range(self.k)]

for i in range(self.N)

]

return np.array(X\_norm)

def check\_r(self) -> bool:

for i in range(len(self.y\_var)):

for j in range(len(self.y\_var)):

if i > j:

if self.y\_var[i] >= self.y\_var[j]:

R = (abs((self.m - 2) \* self.y\_var[i] /

(self.m \* self.y\_var[j]) - 1) / self.sigma)

else:

R = (abs((self.m - 2) \* self.y\_var[j] /

(self.m \* self.y\_var[i]) - 1) / self.sigma)

if R > self.Rcritical[self.m]:

print('Variance isn\'t stable!')

return False

return True

def find\_b(self) -> np.array:

mx1 = np.mean(self.norm\_matrix[:, 0])

mx2 = np.mean(self.norm\_matrix[:, 1])

a1 = np.mean(self.norm\_matrix[:, 0] \*\* 2)

a2 = np.mean(self.norm\_matrix[:, 0] \* self.norm\_matrix[:, 1])

a3 = np.mean(self.norm\_matrix[:, 1] \*\* 2)

my = np.mean(self.y\_mean)

a11 = np.mean(self.norm\_matrix[:, 0] \* self.y\_mean)

a22 = np.mean(self.norm\_matrix[:, 1] \* self.y\_mean)

b = np.linalg.solve([[1, mx1, mx2],

[mx1, a1, a2],

[mx2, a2, a3]],

[my, a11, a22])

return b

def find\_a(self) -> np.array:

delta\_x = [abs(self.X1\_range[1] - self.X1\_range[0]) / 2,

abs(self.X2\_range[1] - self.X2\_range[0]) / 2]

a = [(self.b[0] - self.b[1] \* self.x0[0] / delta\_x[0] -

self.b[2] \* self.x0[1] / delta\_x[1]),

self.b[1] / delta\_x[0],

self.b[2] / delta\_x[1]]

return np.array(a)

def check\_b\_koefs(self) -> np.array:

return np.array([

(self.b[0] + np.sum(self.b[1:3] \* self.norm\_matrix[i]))

for i in range(self.N)])

def check\_a\_koefs(self) -> np.array:

return np.array([

(self.a[0] + np.sum(self.a[1:3] \* self.plan\_matrix[i]))

for i in range(self.N)])

def check\_results(self) -> None:

print('Матриця планування:\n', self.plan\_matrix)

print('Нормована матриця:\n', self.norm\_matrix)

print('Матриця Y:\n', self.y\_matrix)

print('\nНормовані коефіцієнти: ', self.b)

print('Натуралізовані коефіцієнти:', self.a)

print('\nY середнє: ', self.y\_mean)

print('Перевірка нормованих коефіцієнтів: ', self.check\_b)

print('Перевірка натуралізованих коефіцієнтів: ', self.check\_a)

if \_\_name\_\_ == '\_\_main\_\_':

m = 5

X1\_range = [10, 40]

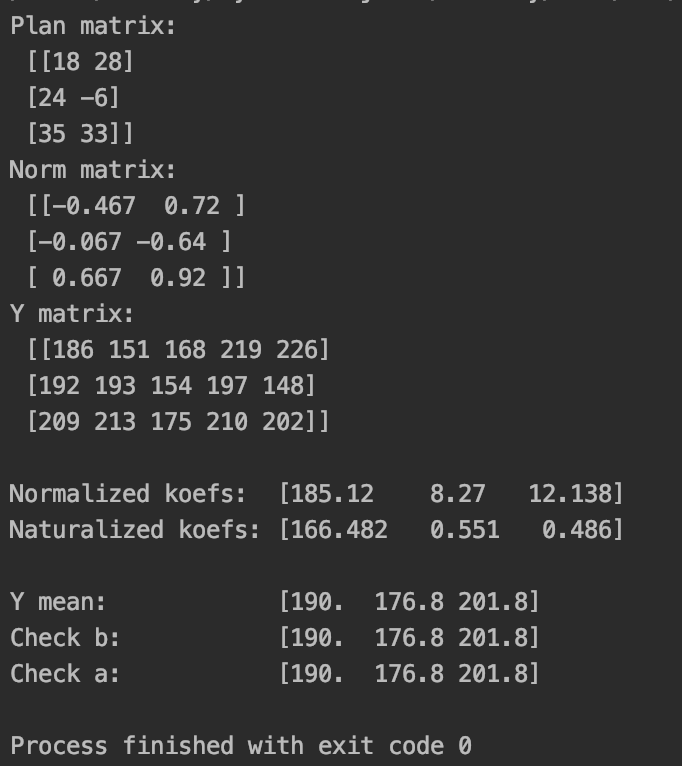
X2\_range = [-15, 35]

Y\_range = [140, 240]

res = Experiment(X1\_range, X2\_range, Y\_range, m)

res.check\_results()

**Результати роботи програми:**

****