Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №2

з дисципліни МОПЕ

на тему:

# «ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

Виконав:

студент групи ІВ-82

Захарчук Даниїл Сергійович

Залікова книжка № 8210

Варіант: 209

Перевірив:

Регіда П.Г.

Київ 2020

**Варіант:**

**X1min = -20 X2min = -30**

**X1max = 15 X2max = 45**

**Ymin = (20 - N) \* 10 = (20 - 9) \* 10 = 110 Ymax = (30 - N) \* 10 = (30 - 9) \* 10 = 210**

**Код програми:**

import numpy as np

import sys

from typing import List

np.set\_printoptions(precision=3)

class FactorExperiment:

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

X1\_range: List[int],

X2\_range: List[int],

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

self.m = m

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.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)))

self.check\_r()

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 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) -> None:

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 > 2.0:

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

sys.exit()

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('Plan matrix:\n', self.plan\_matrix)

print('Norm matrix:\n', self.norm\_matrix)

print('Y matrix:\n', self.y\_matrix)

print('\nNormalized koefs: ', self.b)

print('Naturalized koefs:', self.a)

print('\nY mean: ', self.y\_mean)

print('Check b: ', self.check\_b)

print('Check a: ', self.check\_a)

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

m = 5

X1\_range = [-20, 15]

X2\_range = [-30, 45]

Y\_range = [110, 210]

lab\_2 = FactorExperiment(X1\_range, X2\_range, Y\_range, m)

lab\_2.check\_results()