Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3
З дисципліни «Методи наукових досліджень»
За темою:

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІВ-91
Гришин О.С.
Номер у списку - 07

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

Мета: Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

Завдання:

Завдання на лабораторну роботу

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.

$$y_{i\max} = 200 + x_{cp\max}$$
$$y_{i\min} = 200 + x_{cp\min}$$

де $x_{cp\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}$, $x_{cp\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$

3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

107	10	40	25	45	40	45

Програмний код

Main.pt

```
import logging
import os
import sys
from pycallgraph import PyCallGraph
from pycallgraph.output import GraphvizOutput

logging.basicConfig(filename='app.log', filemode='w', format='%(levelname)s -
%(message)s')

try:
    from script import *
except ImportError:
    logging.exception('Import error')
    sys.exit()


if __name__ == '__main__':
    e = Experiment((10, 40), (25, 45), (40, 45))

    try:
        with PyCallGraph(output=GraphvizOutput()):
            e.make_experiment()
    except Exception as e:
        logging.exception(f'{e} in module '
                          f'- {os.path.split(sys.exc_info()[-
1].tb_frame.f_code.co_filename)[1]} '
                          f'- at line {sys.exc_info()[-1].tb_lineno}',
exc_info=False)
        sys.exit()
```

Script.py

```
import math
import numpy as np
from numpy import average, transpose
from numpy.linalg import solve
from prettytable import PrettyTable
from scipy.stats import f
from scipy.stats import t as t_criterium
from functools import partial
from random import randint


class Experiment:
    m, N, d = 3, 8, 8

    x0_factor = [1, 1, 1, 1, 1, 1, 1, 1]
    x1_factor = [-1, -1, 1, 1, -1, -1, 1, 1]
    x2_factor = [-1, 1, -1, 1, -1, 1, -1, 1]
    x3_factor = [-1, 1, 1, -1, 1, -1, -1, 1]
    x1x2_factor = [a * b for a, b in zip(x1_factor, x2_factor)]
    x1x3_factor = [a * b for a, b in zip(x1_factor, x3_factor)]
    x2x3_factor = [a * b for a, b in zip(x2_factor, x3_factor)]
    x1x2x3_factor = [a * b * c for a, b, c in zip(x1_factor, x2_factor,
x3_factor)]

    x1_list = []
    x2_list = []
    x3_list = []
    x1x2_list = []
    x1x3_list = []
    x2x3_list = []
    x1x2x3_list = []
    x_main_list = [x0_factor, x1_list, x2_list, x3_list, x1x2_list,
x1x3_list, x2x3_list, x1x2x3_list]
    x_factor_list = [x0_factor, x1_factor, x2_factor, x3_factor, x1x2_factor,
x1x3_factor, x2x3_factor, x1x2x3_factor]

    list_bi = []

    def __init__(self, x1, x2, x3):

        self.F1 = self.m - 1
        self.F2 = self.N
        self.F3 = self.F1 * self.F2
        self.F4 = self.N - self.d

        self.x1 = x1
        self.x2 = x2
        self.x3 = x3
        self.x_tuple = (self.x1, self.x2, self.x3)
        # print(self.x_tuple)
        self.x_max_average = average([i[1] for i in self.x_tuple])
        # print(self.x_max_average)
        self.x_min_average = average([i[0] for i in self.x_tuple])
        # print(self.x_min_average)
        self.y_max = int(200 + self.x_max_average)
        self.y_min = int(200 + self.x_min_average)
        self.y_min_max = [self.y_min, self.y_max]
        # print(self.y_max, self.y_min)
        self.mat_Y = [[randint(self.y_min_max[0], self.y_min_max[1]) for _ in
range(self.m)] for _ in range(self.N)]
        # pprint(self.mat_Y)
```

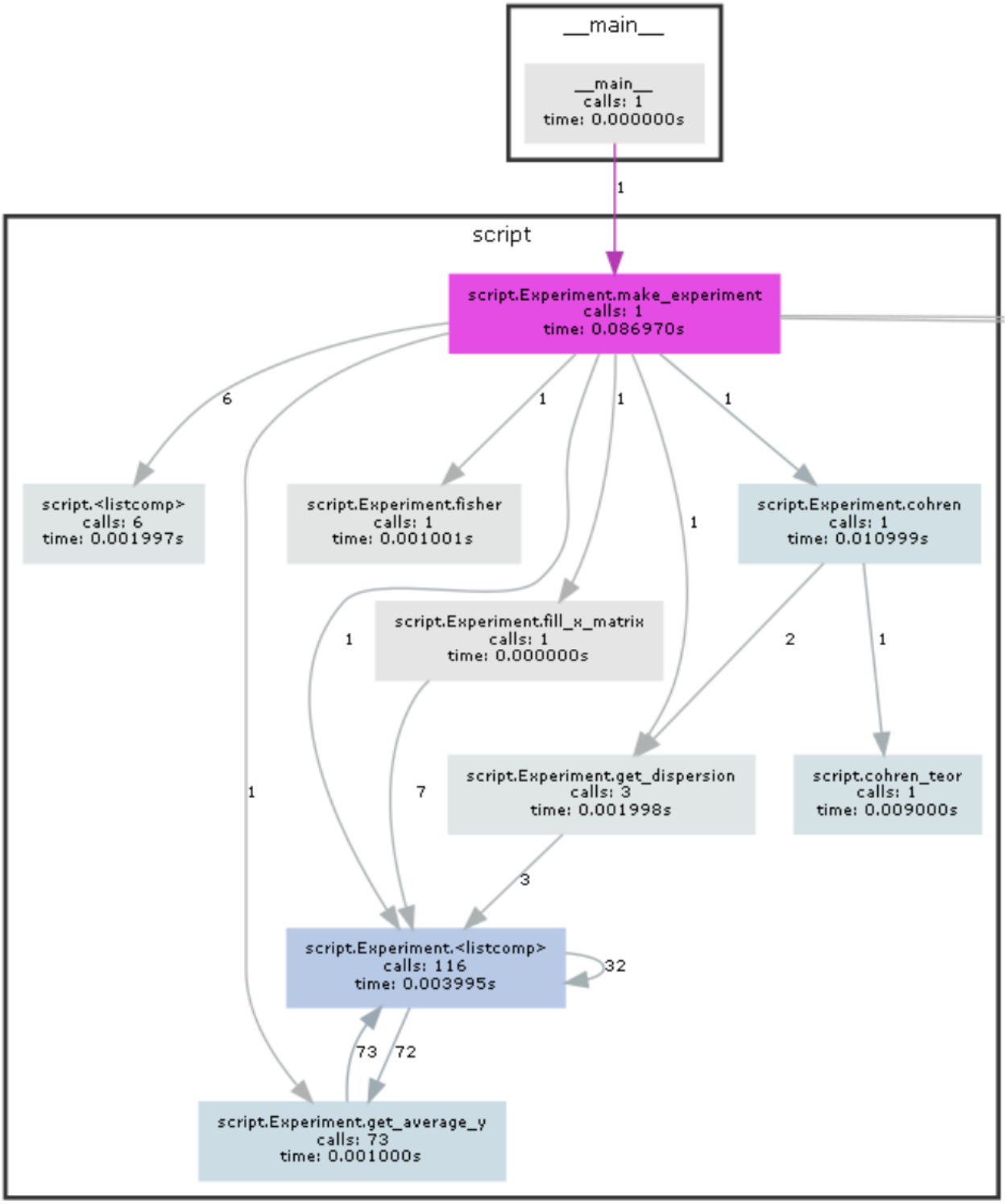
```
    def get_average_y(self):
        """середні значення функцій"""
        return [round(sum(self.mat_Y[k1]) / self.m, 3) for k1 in
range(self.N)]

    def get_dispersion(self):
        return [round(sum([((k1 - self.get_average_y()[j]) ** 2) for k1 in
self.mat_Y[j]]) / self.m, 3) for j in
                range(self.N)]

    def fill_x_matrix(self):
        """Заповнює матрицю експерименту"""
        [self.x1_list.append(self.x1[0 if i == -1 else 1]) for i in
self.x1_factor]
        [self.x2_list.append(self.x2[0 if i == -1 else 1]) for i in
self.x2_factor]
        [self.x3_list.append(self.x3[0 if i == -1 else 1]) for i in
self.x3_factor]
        [self.x1x2_list.append(a * b) for a, b in zip(self.x1_list,
self.x2_list)]
        [self.x1x3_list.append(a * b) for a, b in zip(self.x1_list,
self.x3_list)]
        [self.x2x3_list.append(a * b) for a, b in zip(self.x2_list,
self.x3_list)]
        [self.x1x2x3_list.append(a * b * c) for a, b, c in zip(self.x1_list,
self.x2_list, self.x3_list)]

    def cohren(self):

        def cohren_teor(f1, f2, q=0.05):
            q1 = q / f1
            fisher_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
            return fisher_value / (fisher_value + f1 - 1)

        Gp = max(self.get_dispersion()) / sum(self.get_dispersion())
        Gt = cohren_teor(self.F1, self.F2)
        print("\nGp = ", Gp, " Gt = ", Gt)
        return Gp < Gt

    def fisher(self):
        fisher_teor = partial(f.ppf, q=1 - 0.05)
        Ft = fisher_teor(dfn=self.F4, dfd=self.F3)
        return Ft

    def make_experiment(self):
        self.fill_x_matrix()

        dispersion = self.get_dispersion()
        sum_dispersion = sum(dispersion)
        y_average = self.get_average_y()

        column_names1 = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3",
"X1X2X3", "Y1", "Y2", "Y3", "Y", "S^2"]
        trans_y_mat = transpose(self.mat_Y).tolist()

        """При розрахунку ко-ів використовуєемо як натуральні так і нормовані
значення"""
        list_for_solve_a = list(zip(*self.x_main_list))
        list_for_solve_b = self.x_factor_list

        for k in range(self.N):
            S = 0
            for i in range(self.N):
                S += (list_for_solve_b[k][i] * y_average[i]) / self.N
```

```
            self.list_bi.append(round(S, 5))

        pt = PrettyTable()
        cols = self.x_factor_list
        [cols.extend(ls) for ls in [trans_y_mat, [y_average], [dispersion]]]
        [pt.add_column(column_names1[coll_id], cols[coll_id]) for coll_id in
range(13)]
        print(pt, "\n")
        print('Рівняння регресії з коефіцієнтами від нормованих значень
факторів')
        print("y = {} + {}*x1 + {}*x2 + {}*x3 + {}*x1x2 + {}*x1x3 + {}*x2x3 +
{}*x1x2x3 \n".format(*self.list_bi))

        pt = PrettyTable()
        cols = self.x_main_list
        [cols.extend(ls) for ls in [trans_y_mat, [y_average], [dispersion]]]
        [pt.add_column(column_names1[coll_id], cols[coll_id]) for coll_id in
range(13)]
        print(pt, "\n")

        """solve з бібліотеки numpy вирішує систему рівнянь відносно
невідомих к-тів
         рівняння регресії при використанні натуральних значень"""
        list_ai = [round(i, 5) for i in solve(list_for_solve_a, y_average)]
        print('Рівняння регресії з коефіцієнтами від натуральних значень
факторів')
        print("y = {} + {}*x1 + {}*x2 + {}*x3 + {}*x1x2 + {}*x1x3 + {}*x2x3 +
{}*x1x2x3".format(*list_ai))

        if self.cohren():
            print("Дисперсія однорідна!\n")
            Dispersion_B = sum_dispersion / self.N
            Dispersion_beta = Dispersion_B / (self.m * self.N)
            S_beta = math.sqrt(abs(Dispersion_beta))
            beta_list = np.zeros(8).tolist()
            for i in range(self.N):
                beta_list[0] += (y_average[i] * self.x0_factor[i]) / self.N
                beta_list[1] += (y_average[i] * self.x1_factor[i]) / self.N
                beta_list[2] += (y_average[i] * self.x2_factor[i]) / self.N
                beta_list[3] += (y_average[i] * self.x3_factor[i]) / self.N
                beta_list[4] += (y_average[i] * self.x1x2_factor[i]) / self.N
                beta_list[5] += (y_average[i] * self.x1x3_factor[i]) / self.N
                beta_list[6] += (y_average[i] * self.x2x3_factor[i]) / self.N
                beta_list[7] += (y_average[i] * self.x1x2x3_factor[i]) /
self.N
            t_list = [abs(beta_list[i]) / S_beta for i in range(0, self.N)]

            for i, j in enumerate(t_list):
                if j < t_criterium.ppf(q=0.975, df=self.F3):
                    print(f'незначний {beta_list[i]}')
                    beta_list[i] = 0
                    self.d -= 1
            print()
            print("y = {} + {}*x1 + {}*x2 + {}*x3 + {}*x1x2 + {}*x1x3 +
{}*x2x3 + {}*x1x2x3".format(*beta_list))

            Y_counted = [sum([beta_list[0], *[beta_list[i] *
self.x_main_list[1:][j][i] for i in range(self.N)]])
                         for j in range(self.N)]
            Dispersion_ad = 0
            for i in range(len(Y_counted)):
                Dispersion_ad += ((Y_counted[i] - y_average[i]) ** 2) *
self.m / (self.N - self.d)
            Fp = Dispersion_ad / Dispersion_beta
```

```
            Ft = self.fisher()

            if Ft > Fp:
                print("Рівняння регресії адекватне!")
            else:
                print("Рівняння регресії неадекватне.")
```

Результати роботи програми

Метрики (повна версія із замірами залучення інших бібліотек можна знайти в репозиторії)



Результати розрахунків

X0	X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	Y1	Y2	Y3	Y	S^2
1	-1	-1	-1	1	1	1	-1	231	239	234	234.667	10.889
1	-1	1	1	-1	-1	1	-1	235	238	227	233.333	21.556
1	1	-1	1	-1	1	-1	-1	234	226	232	230.667	11.556
1	1	1	-1	1	-1	-1	-1	234	237	225	232.0	26.0
1	-1	-1	1	1	-1	-1	1	235	236	239	236.667	2.889
1	-1	1	-1	-1	1	-1	1	226	227	226	226.333	0.222
1	1	-1	-1	-1	-1	1	1	234	229	226	229.667	10.889
1	1	1	1	1	1	1	1	226	230	241	232.333	40.222

Рівняння регресії з коефіцієнтами від нормованих значень факторів

y = 231.95838 + -0.79162*x1 + -0.95863*x2 + 1.29162*x3 + 1.95838*x1x2 + -0.95838*x1x3 + 0.54162*x2x3 + -0.70838*x1x2x3

X0	X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	Y1	Y2	Y3	Y	S^2
1	10	25	40	250	400	1000	10000	231	239	234	234.667	10.889
1	10	45	45	450	450	2025	20250	235	238	227	233.333	21.556
1	40	25	45	1000	1800	1125	45000	234	226	232	230.667	11.556
1	40	45	40	1800	1600	1800	72000	234	237	225	232.0	26.0
1	10	25	45	250	450	1125	11250	235	236	239	236.667	2.889
1	10	45	40	450	400	1800	18000	226	227	226	226.333	0.222
1	40	25	40	1000	1600	1000	40000	234	229	226	229.667	10.889
1	40	45	45	1800	1800	2025	81000	226	230	241	232.333	40.222

Рівняння регресії з коефіцієнтами від натуральних значень факторів

y = 301.41908 + -2.23346*x1 + -3.35008*x2 + -1.25558*x3 + 0.09334*x1x2 + 0.04056*x1x3 + 0.06889*x2x3 + -0.00189*x1x2x3

Gp = 0.3237886703750514 Gt = 0.815948432359917

Дисперсія однорідна!

незначний -0.7916249999999962

незначний -0.9586250000000014

незначний 1.2916249999999998

незначний -0.9583750000000002

незначний 0.5416249999999962

незначний -0.7083750000000038

y = 231.95837500000002 + 0*x1 + 0*x2 + 0*x3 + 1.9583750000000002*x1x2 + 0*x1x3 + 0*x2x3 + 0*x1x2x3

Рівняння регресії неадекватне.