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 № 5

з дисципліни «Методи оптимізації та планування експерименту»
натему

«ЗАГАЛЬНІ ПРИНЦИПИ ОРГАНІЗАЦІЇЕКСПЕРИМЕНТІВ З ДОВІЛЬНИМИ ЗНАЧЕННЯМИ ФАКТОРІВ»

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІО93
Ященко Євген
ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

```python
import random
import numpy as np
import sklearn.linear_model as lm
from scipy.stats import f, t
from functools import partial
from pyDOE2 import *


def regression(x, b):
    y = sum([x[i] * b[i] for i in range(len(x))])
    return y


x_range = ((10, 60), (-35, 10), (-35, 45))

x_aver_max = sum([x[1] for x in x_range]) / 3
x_aver_min = sum([x[0] for x in x_range]) / 3

y_max = 200 + int(x_aver_max)
y_min = 200 + int(x_aver_min)


# квадратна дисперсія
def s_kv(y, y_aver, n, m):
    res = []
    for i in range(n):
        s = sum([(y_aver[i] - y[i][j]) ** 2 for j in range(m)]) / m
        res.append(round(s, 3))
    return res


def plan_matrix5(n, m):
    print('\nЛабораторна 5')
    print(f'\nГереруємо матрицю планування для n = {n}, m = {m}')

    y = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = random.randint(y_min, y_max)

    if n > 14:
        no = n - 14
    else:
        no = 1
    x_norm = ccdesign(3, center=(0, no))
    x_norm = np.insert(x_norm, 0, 1, axis=1)

    for i in range(4, 11):
        x_norm = np.insert(x_norm, i, 0, axis=1)

    l = 1.215

    for i in range(len(x_norm)):
        for j in range(len(x_norm[i])):
            if x_norm[i][j] < -1 or x_norm[i][j] > 1:
                if x_norm[i][j] < 0:
                    x_norm[i][j] = -l
                else:
                    x_norm[i][j] = l

    def add_sq_nums(x):
        for i in range(len(x)):
            x[i][4] = x[i][1] * x[i][2]
            x[i][5] = x[i][1] * x[i][3]
```

```
            x[i][6] = x[i][2] * x[i][3]
            x[i][7] = x[i][1] * x[i][3] * x[i][2]
            x[i][8] = x[i][1] ** 2
            x[i][9] = x[i][2] ** 2
            x[i][10] = x[i][3] ** 2
        return x

    x_norm = add_sq_nums(x_norm)

    x = np.ones(shape=(len(x_norm), len(x_norm[0])), dtype=np.int64)
    for i in range(8):
        for j in range(1, 4):
            if x_norm[i][j] == -1:
                x[i][j] = x_range[j - 1][0]
            else:
                x[i][j] = x_range[j - 1][1]

    for i in range(8, len(x)):
        for j in range(1, 3):
            x[i][j] = (x_range[j - 1][0] + x_range[j - 1][1]) / 2

    dx = [x_range[i][1] - (x_range[i][0] + x_range[i][1]) / 2 for i in
range(3)]

    x[8][1] = l * dx[0] + x[9][1]
    x[9][1] = -l * dx[0] + x[9][1]
    x[10][2] = l * dx[1] + x[9][2]
    x[11][2] = -l * dx[1] + x[9][2]
    x[12][3] = l * dx[2] + x[9][3]
    x[13][3] = -l * dx[2] + x[9][3]

    x = add_sq_nums(x)

    print('\nX:\n', x)
    print('\nX нормоване:\n')
    for i in x_norm:
        print([round(x, 2) for x in i])
    print('\nY:\n', y)

    return x, y, x_norm


def find_coef(X, Y, norm=False):
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(X, Y)
    B = skm.coef_

    if norm == 1:
        print('\nКоефіцієнти рівняння регресії з нормованими X:')
    else:
        print('\nКоефіцієнти рівняння регресії:')
    B = [round(i, 3) for i in B]
    print(B)
    print('\nРезультат рівняння зі знайденими коефіцієнтами:\n', np.dot(X,
B))
    return B


def kriteriy_cochrana(y, y_aver, n, m):
    f1 = m - 1
    f2 = n
    q = 0.05
    S_kv = s_kv(y, y_aver, n, m)
    Gp = max(S_kv) / sum(S_kv)
```

```
    print('\nПеревірка за критерієм Кохрена')
    return Gp


def cohren(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)


# оцінки коефіцієнтів
def bs(x, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]

    for i in range(len(x[0])):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_aver)) / n
        res.append(b)
    return res


def kriteriy_studenta(x, y, y_aver, n, m):
    S_kv = s_kv(y, y_aver, n, m)
    s_kv_aver = sum(S_kv) / n

    # статистична оцінка дисперсії
    s_Bs = (s_kv_aver / n / m) ** 0.5  # статистична оцінка дисперсії
    Bs = bs(x, y_aver, n)
    ts = [round(abs(B) / s_Bs, 3) for B in Bs]

    return ts


def kriteriy_fishera(y, y_aver, y_new, n, m, d):
    S_ad = m / (n - d) * sum([(y_new[i] - y_aver[i]) ** 2 for i in
range(len(y))])
    S_kv = s_kv(y, y_aver, n, m)
    S_kv_aver = sum(S_kv) / n

    return S_ad / S_kv_aver


def check(X, Y, B, n, m):
    print('\n\tПеревірка рівняння:')
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05

    ### табличні значення
    student = partial(t.ppf, q=1 - q)
    t_student = student(df=f3)

    G_kr = cohren(f1, f2)
    ###

    y_aver = [round(sum(i) / len(i), 3) for i in Y]
    print('\nСереднє значення y:', y_aver)

    disp = s_kv(Y, y_aver, n, m)
    print('Дисперсія y:', disp)

    Gp = kriteriy_cochrana(Y, y_aver, n, m)
    print(f'Gp = {Gp}')
    if Gp < G_kr:
```

```
        print(f'З ймовірністю {1-q} дисперсії однорідні.')
    else:
        print("Необхідно збільшити кількість дослідів")
        m += 1
        main(n, m)

    ts = kriteriy_studenta(X[:, 1:], Y, y_aver, n, m)
    print('\nКритерій Стьюдента:\n', ts)
    res = [t for t in ts if t > t_student]
    final_k = [B[i] for i in range(len(ts)) if ts[i] in res]
    print('\nКоефіцієнти {} статистично незначущі, тому ми виключаємо їх з
рівняння.'.format(
        [round(i, 3) for i in B if i not in final_k]))

    y_new = []
    for j in range(n):
        y_new.append(regression([X[j][i] for i in range(len(ts)) if ts[i] in
res], final_k))

    print(f'\nЗначення "y" з коефіцієнтами {final_k}')
    print(y_new)

    d = len(res)
    if d >= n:
        print('\nF4 <= 0')
        print('')
        return
    f4 = n - d

    F_p = kriteriy_fishera(Y, y_aver, y_new, n, m, d)

    fisher = partial(f.ppf, q=0.95)
    f_t = fisher(dfn=f4, dfd=f3)  # табличне знач
    print('\nПеревірка адекватності за критерієм Фішера')
    print('Fp =', F_p)
    print('F_t =', f_t)
    if F_p < f_t:
        print('Математична модель адекватна експериментальним даним')
    else:
        print('Математична модель не адекватна експериментальним даним')


def main(n, m):
    X5, Y5, X5_norm = plan_matrix5(n, m)

    y5_aver = [round(sum(i) / len(i), 3) for i in Y5]
    B5 = find_coef(X5, y5_aver)

    check(X5_norm, Y5, B5, n, m)


if __name__ == '__main__':
    main(17, 5)
```

Лабораторна 5

Гереруємо матрицю планування для n = 17, m = 5

X:

```
[[    1    10   -35   -35  -350  -350  1225 12250   100  1225
   1225]
 [    1    60   -35   -35 -2100 -2100  1225 73500  3600  1225
   1225]
 [    1    10    10   -35   100  -350  -350 -3500   100   100
   1225]
 [    1    60    10   -35   600 -2100  -350 -21000  3600   100
   1225]
 [    1    10   -35    45  -350   450 -1575 -15750   100  1225
   2025]
 [    1    60   -35    45 -2100  2700 -1575 -94500  3600  1225
   2025]
 [    1    10    10    45   100   450   450  4500   100   100
   2025]
 [    1    60    10    45   600  2700   450 27000  3600   100
   2025]
 [    1    65   -12     1  -780    65   -12  -780  4225   144
      1]
 [    1     4   -12     1   -48     4   -12   -48    16   144
      1]
 [    1    35    15     1   525    35    15   525  1225   225
      1]
 [    1    35   -39     1 -1365    35   -39 -1365  1225  1521
      1]
 [    1    35   -12    49  -420  1715  -588 -20580  1225   144
   2401]
 [    1    35   -12   -47  -420 -1645   564  19740  1225   144
   2209]
 [    1    35   -12     1  -420    35   -12  -420  1225   144
      1]
 [    1    35   -12     1  -420    35   -12  -420  1225   144
      1]
 [    1    35   -12     1  -420    35   -12  -420  1225   144
      1]]
```

X нормоване:

X нормоване:

```
[1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, 1.0, -1.0, 1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, -1.0, 1.0, -1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, -1.22, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 1.48, 0.0, 0.0]
[1.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0, 0.0]
[1.0, 0.0, -1.22, 0.0, -0.0, 0.0, -0.0, -0.0, 0.0, 1.48, 0.0]
[1.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0]
[1.0, 0.0, 0.0, -1.22, 0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 1.48]
[1.0, 0.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

Y:

```
 [[206. 216. 219. 190. 227.]
 [220. 217. 184. 224. 222.]
 [225. 222. 229. 218. 190.]
 [203. 200. 218. 219. 203.]
 [204. 217. 196. 210. 212.]
 [180. 233. 207. 218. 211.]
 [185. 210. 185. 229. 191.]
 [195. 186. 209. 233. 183.]
 [190. 209. 237. 225. 229.]
 [217. 234. 230. 201. 207.]
 [217. 231. 182. 203. 192.]
 [188. 206. 237. 190. 230.]
 [199. 197. 200. 181. 228.]
 [226. 186. 199. 200. 201.]
 [194. 202. 215. 194. 202.]
 [213. 220. 237. 205. 233.]
 [202. 196. 183. 210. 193.]]
```

```
Коефіцієнти рівняння регресії:
[220.99, -0.786, -0.004, -0.146, -0.003, 0.002, -0.004, 0.0, 0.011, -0.001, -0.003]

Результат рівняння зі знайденими коефіцієнтами:
 [210.03  210.98  215.925 210.125 208.75  217.7   200.245 202.445 218.648
 217.977 204.956 209.762 199.544 202.808 208.088 208.088 208.088 208.088]

        Перевірка рівняння:

Середнє значення y: [211.6, 213.4, 216.8, 208.6, 207.8, 209.8, 200.0, 201.2, 218.0, 217.8, 205.0, 210.2, 201.0, 202.4, 201.4, 221.6, 196.8]
Дисперсія y: [161.84, 221.44, 192.56, 66.64, 52.16, 300.56, 294.4, 334.56, 279.2, 162.16, 304.4, 405.76, 230.0, 169.04, 59.04, 143.84, 81.36]

Перевірка за критерієм Кохрена
Gp = 0.11730693618891226
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [134.72, 0.131, 0.368, 1.137, 0.411, 0.365, 0.639, 0.35, 87.923, 86.767, 86.104]

Коефіцієнти [-0.786, -0.004, -0.146, 0.002, -0.004, 0.0] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [220.99, 0.011, -0.001, -0.003]
[220.997, 220.997, 220.997, 220.997, 220.997, 220.997, 220.997, 220.997, 221.006238475, 221.006238475, 220.988523775, 220.988523775, 220.985571325
, 220.985571325, 220.99, 220.99, 220.99]

Перевірка адекватності за критерієм Фішера
Fp = 6.720256438211878
F t = 1.8669463026594668



        Перевірка рівняння:

Середнє значення y: [211.6, 213.4, 216.8, 208.6, 207.8, 209.8, 200.0, 201.2, 218.0, 217.8, 205.0, 210.2, 201.0, 202.4, 201.4, 221.6, 196.8]
Дисперсія y: [161.84, 221.44, 192.56, 66.64, 52.16, 300.56, 294.4, 334.56, 279.2, 162.16, 304.4, 405.76, 230.0, 169.04, 59.04, 143.84, 81.36]

Перевірка за критерієм Кохрена
Gp = 0.11730693618891226
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [134.72, 0.131, 0.368, 1.137, 0.411, 0.365, 0.639, 0.35, 87.923, 86.767, 86.104]

Коефіцієнти [-0.786, -0.004, -0.146, 0.002, -0.004, 0.0] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [220.99, 0.011, -0.001, -0.003]
[220.997, 220.997, 220.997, 220.997, 220.997, 220.997, 220.997, 220.997, 221.006238475, 221.006238475, 220.988523775, 220.988523775, 220.985571325
, 220.985571325, 220.99, 220.99, 220.99]

Перевірка адекватності за критерієм Фішера
Fp = 6.720256438211878
F_t = 1.8669463026594668
Математична модель не адекватна експериментальним даним
PS C:\Users\maxma\PycharmProjects\Lab№4>
```