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 наукових досліджень» на тему

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)»

Виконав:

студент ІІ курсу ФІОТ
групи ІВ-93
Трибушенко А.С.

ПЕРЕВІРИВ:

ас. Регіда П. Г.

Київ - 2021

Завдання

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.

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

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

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Варіант:

$N_{варіант} = 327$

№варіанта	x_1		x_2		x_3	
	min	max	min	max	min	max
327	-8	4	-9	7	-3	9

Роздруківка коду програми:

```
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 = ((-8, 4), (-9, 7), (-3, 9))
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(f'\nГенеруємо 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(15, 3)
```

Результат виконання коду:

```
X:
 [[   1   -8   -9   -3   72   24   27 -216   64   81    9]
 [   1    4   -9   -3  -36  -12   27  108   16   81    9]
 [   1   -8    7   -3  -56   24  -21  168   64   49    9]
 [   1    4    7   -3   28  -12  -21  -84   16   49    9]
 [   1   -8   -9    9   72  -72  -81  648   64   81   81]
 [   1    4   -9    9  -36   36  -81 -324   16   81   81]
 [   1   -8    7    9  -56  -72   63 -504   64   49   81]
 [   1    4    7    9   28   36   63  252   16   49   81]
 [   1    5   -1    1   -5    5   -1   -5   25    1    1]
 [   1   -9   -1    1    9   -9   -1    9   81    1    1]
 [   1   -2    8    1  -16   -2    8  -16    4   64    1]
 [   1   -2  -10    1   20   -2  -10   20    4  100    1]
 [   1   -2   -1    8    2  -16   -8   16    4    1   64]
 [   1   -2   -1   -6    2   12    6  -12    4    1   36]
 [   1   -2   -1    1    2   -2   -1    2    4    1    1]]
```

```
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]
```

```
Y:
 [[194. 199. 202.]
 [205. 197. 195.]
 [194. 199. 197.]
 [203. 201. 201.]
 [194. 195. 194.]
 [199. 203. 198.]
 [204. 202. 202.]
 [202. 195. 197.]
 [205. 196. 205.]
 [195. 199. 199.]
 [205. 198. 204.]
 [198. 197. 200.]
 [195. 198. 201.]
 [205. 194. 197.]
 [196. 198. 197.]]

Коефіцієнти рівняння регресії:
[199.33, 0.262, 0.137, 0.011, 0.003, -0.024, -0.001, -0.006, 0.008, 0.01, -0.016]

Результат рівняння зі знайденими коефіцієнтами:
 [198.055 199.411 197.287 202.675 194.263 199.939 202.519 198.403 200.604
 197.678 200.657 198.461 198.077 197.853 198.749]
```

```
       Перевірка рівняння:

Середнє значення y: [198.333, 199.0, 196.667, 201.667, 194.333, 200.0, 202.667, 198.0, 202.0, 197.667, 202.333, 198.333, 198.0, 198.667, 197.0]
Дисперсія y: [10.889, 18.667, 4.222, 0.889, 0.222, 4.667, 0.889, 8.667, 18.0, 3.556, 9.556, 1.556, 6.0, 21.556, 0.667]

Перевірка за критерієм Кохрена
Gp = 0.19595829204658055
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [492.895, 0.232, 0.409, 0.024, 0.991, 0.771, 0.881, 2.422, 360.12, 360.364, 359.389]

Коефіцієнти [0.262, 0.137, 0.011, 0.003, -0.024, -0.001] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [199.33, -0.006, 0.008, 0.01, -0.016]
[199.33800000000002, 199.32600000000002, 199.32600000000002, 199.33800000000002, 199.32600000000002, 199.33800000000002, 199.33800000000002, 199.32600000000002, 199.34180980000002, 199.34180980000002, 199.34476225, 199.34476225, 199.30638040000002, 199.30638040000002, 199.33]

Перевірка адекватності за критерієм Фішера
Fp = 3.23942731910992
F t = 2.164579917125473
Математична модель не адекватна експериментальним даним
```

Висновки:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії, рівняння регресії з ефектом взаємодії та рівняння регресії з квадратичними членами, складено матрицю планування експерименту, було визначено коефіцієнти рівнянь регресії (натуралізовані та нормовані), для форми з квадратичними членами - натуралізовані, виконано перевірку правильності розрахунку коефіцієнтів рівнянь регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера) для кожної форми рівняння регресії . При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів, при неадекватності і такого рівняння регресії було затосовано рівняння регресії з квадратичними членами. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості q = 0.05.