

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки

Лабораторна робота №4

«Проведення трьох факторного експерименту при використанні рівняння  
регресії з урахуванням ефекту взаємодії»

Виконав:  
студент групи ІО-91  
Андрейцов Я. Є.  
Залікова книжка № ІО-9101  
Варіант 1

ПЕРЕВІРИВ:  
Егіда П. Г.

Київ-2020

## Варіант 1

№ <sub>варіанта</sub>	$x_1$		$x_2$		$x_3$	
	min	max	min	Max	min	max
101	-40	20	10	60	-20	20

## Матриця планування

Матриця планування:

```

X0  X1  X2  X3  Y1  Y2  Y3
[[ 1. -40. 10. -20. 192. 214. 223. 208. 201. 217. 221. 194. 227. 196.]
 [ 1. -40. 60. 20. 191. 226. 232. 216. 217. 212. 196. 184. 198. 192.]
 [ 1. 20. 10. 20. 189. 232. 211. 199. 226. 221. 201. 232. 221. 203.]
 [ 1. 20. 60. -20. 224. 206. 217. 207. 186. 226. 193. 217. 191. 221.]
 [ 1. -40. 10. 20. 212. 220. 184. 190. 186. 203. 207. 187. 205. 199.]
 [ 1. -40. 60. -20. 194. 205. 204. 191. 229. 211. 207. 192. 232. 229.]
 [ 1. 20. 10. -20. 225. 205. 186. 231. 218. 220. 201. 217. 191. 226.]
 [ 1. 20. 60. 20. 187. 233. 228. 185. 192. 198. 220. 210. 215. 200.]]

```

## Результати виконання роботи

Рівняння регресії:

$$y = 209.36 + 0.07 \cdot x_1 + -0.01 \cdot x_2 + -0.08 \cdot x_3$$

Перевірка за критерієм Кохрена:

Розрахункове значення:  $G_p = 0.1609547237385911$

Табличне значення:  $G_t = 0.27680367243514814$

З ймовірністю 0.95 дисперсії однорідні.

Табличне значення критерій Стьюдента:

1.9934635662785827

Розрахункове значення критерій Стьюдента:

[131.102, 1.315, 0.213, 1.063]

Коефіцієнти [0.07, -0.01, -0.08] статистично незначущі.

Отримаємо значення рівня регресії для 10 дослідів:

[209.36, 209.36, 209.36, 209.36, 209.36, 209.36, 209.36, 209.36]

Перевірка адекватності за критерієм Фішера:

Розрахункове значення критерія Фішера:  $F_p = 0.9981104868598901$

Табличне значення критерія Фішера:  $F_t = 2.139655512399197$

Математична модель адекватна експериментальним даним

## Код програми

```
import random
import numpy as np
import sklearn.linear_model as lm
from scipy.stats import f, t
from numpy.linalg import solve

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

def dispersion(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 planing_matrix_interaction_effect(n, m):
    x_normalized = [[1, -1, -1, -1],
```

```

        [1, -1, 1, 1],
        [1, 1, -1, 1],
        [1, 1, 1, -1],
        [1, -1, -1, 1],
        [1, -1, 1, -1],
        [1, 1, -1, -1],
        [1, 1, 1, 1]]
y = np.zeros(shape=(n, m), dtype=np.int64)
for i in range(n):
    for j in range(m):
        y[i][j] = random.randint(y_min, y_max)

for x in x_normalized:
    x.append(x[1] * x[2])
    x.append(x[1] * x[3])
    x.append(x[2] * x[3])
    x.append(x[1] * x[2] * x[3])

x_normalized = np.array(x_normalized[:len(y)])
x = np.ones(shape=(len(x_normalized), len(x_normalized[0])),
dtype=np.int64)

for i in range(len(x_normalized)):
    for j in range(1, 4):
        if x_normalized[i][j] == -1:
            x[i][j] = x_range[j - 1][0]
        else:
            x[i][j] = x_range[j - 1][1]

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]

print(f'\nМатриця планування для n = {n}, m = {m}:')
print('\n3 кодованими значеннями факторів:')
print('\n      X0      X1      X2      X3  X1X2  X1X3  X2X3  X1X2X3      Y1      Y2
Y3')
print(np.concatenate((x, y), axis=1))
print('\nНормовані значення факторів:\n')
print(x_normalized)

return x, y, x_normalized

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)
    return B

def bs(x, y, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]
    for i in range(7):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_aver)) / n

```

```

        res.append(b)
    return res

def kriteriy_students2(x, y, y_aver, n, m):
    S_kv = dispersion(y, y_aver, n, m)
    s_kv_aver = sum(S_kv) / n
    s_Bs = (s_kv_aver / n / m) ** 0.5
    Bs = bs(x, y, y_aver, n)
    ts = [round(abs(B) / s_Bs, 3) for B in Bs]

    return ts

def kriteriy_students(x, y_average, n, m, dispersion):
    dispersion_average = sum(dispersion) / n
    s_beta_s = (dispersion_average / n / m) ** 0.5

    beta = [sum(1 * y for y in y_average) / n]
    for i in range(3):
        b = sum(j[0] * j[1] for j in zip(x[:,i], y_average)) / n
        beta.append(b)

    t = [round(abs(b) / s_beta_s, 3) for b in beta]

    return t

def kriteriy_fishera(y, y_average, y_new, n, m, d, dispersion):
    S_ad = m / (n - d) * sum([(y_new[i] - y_average[i])**2 for i in
range(len(y))])
    dispersion_average = sum(dispersion) / n

    return S_ad / dispersion_average

def check(X, Y, B, n, m, norm=False):

    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05

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

    dispersion_arr = dispersion(Y, y_aver, n, m)

    qq = (1 + 0.95) / 2
    student_cr_table = t.ppf(df=f3, q=qq)

    ts = kriteriy_students2(X[:, 1:], Y, y_aver, n, m)

    temp_cohren = f.ppf(q=(1 - q / f1), dfn=f2, dfd=(f1 - 1) * f2)
    cohren_cr_table = temp_cohren / (temp_cohren + f1 - 1)
    Gp = max(dispersion_arr) / sum(dispersion_arr)

    print('Дисперсія y:', dispersion_arr)

    print(f'Gp = {Gp}')
    if Gp < cohren_cr_table:
        print(f'З ймовірністю {1-q} дисперсії однорідні.')
    else:
        print("Необхідно збільшити кількість дослідів")

```

```

        m += 1
        with_interaction_effect(n, m)

    print('\nКритерій Ст'юдента:\n', ts)
    res = [t for t in ts if t > student_cr_table]
    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

    Fp = kriteriy_fishera(Y, y_aver, y_new, n, m, d, dispersion_arr)

    Ft = f.ppf(dfn=f4, dfd=f3, q=1 - 0.05)

    print('\nПеревірка адекватності за критерієм Фішера')
    print('Fp =', Fp)
    print('Ft =', Ft)
    if Fp < Ft:
        print('Математична модель адекватна експериментальним даним')
        return True
    else:
        print('Математична модель не адекватна експериментальним даним')
        return False

def with_interaction_effect(n, m):
    X, Y, X_norm = planing_matrix_interaction_effect(n, m)

    y_aver = [round(sum(i) / len(i), 3) for i in Y]

    B_norm = find_coef(X_norm, y_aver, norm=True)

    return check(X_norm, Y, B_norm, n, m, norm=True)

def planning_matrix_linear(n, m, x_range):
    x_normalized = np.array([[1, -1, -1, -1],
                             [1, -1, 1, 1],
                             [1, 1, -1, 1],
                             [1, 1, 1, -1],
                             [1, -1, -1, 1],
                             [1, -1, 1, -1],
                             [1, 1, -1, -1],
                             [1, 1, 1, 1]])

    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)

```

```

x_normalized = x_normalized[:len(y)]

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

print('\nМатриця планування:')
print('\n      X0   X1   X2   X3   Y1   Y2   Y3   ')
print(np.concatenate((x, y), axis=1))

return x, y, x_normalized

def regression_equation(x, y, n):
    y_average = [round(sum(i) / len(i), 2) for i in y]

    mx1 = sum(x[:, 1]) / n
    mx2 = sum(x[:, 2]) / n
    mx3 = sum(x[:, 3]) / n

    my = sum(y_average) / n

    a1 = sum([y_average[i] * x[i][1] for i in range(len(x))]) / n
    a2 = sum([y_average[i] * x[i][2] for i in range(len(x))]) / n
    a3 = sum([y_average[i] * x[i][3] for i in range(len(x))]) / n

    a12 = sum([x[i][1] * x[i][2] for i in range(len(x))]) / n
    a13 = sum([x[i][1] * x[i][3] for i in range(len(x))]) / n
    a23 = sum([x[i][2] * x[i][3] for i in range(len(x))]) / n

    a11 = sum([i ** 2 for i in x[:, 1]]) / n
    a22 = sum([i ** 2 for i in x[:, 2]]) / n
    a33 = sum([i ** 2 for i in x[:, 3]]) / n

    X = [[1, mx1, mx2, mx3], [mx1, a11, a12, a13], [mx2, a12, a22, a23],
[mx3, a13, a23, a33]]
    Y = [my, a1, a2, a3]
    B = [round(i, 2) for i in solve(X, Y)]

    print('\nPівняння регресії:')
    print(f'y = {B[0]} + {B[1]}*x1 + {B[2]}*x2 + {B[3]}*x3')

    return y_average, B

def linear(n, m):
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05

    x, y, x_norm = planning_matrix_linear(n, m, x_range)

    y_average, B = regression_equation(x, y, n)

    dispersion_arr = dispersion(y, y_average, n, m)

    temp_cohren = f.ppf(q=(1 - q / f1), dfn=f2, dfd=(f1 - 1) * f2)
    cohren_cr_table = temp_cohren / (temp_cohren + f1 - 1)
    Gp = max(dispersion_arr) / sum(dispersion_arr)

```

```

print('\nПеревірка за критерієм Кохрена:\n')
print(f'\nРозрахункове значення: Gp = {Gp}')
    f'\nТабличне значення: Gt = {cohren_cr_table}')
if Gp < cohren_cr_table:
    print(f'З ймовірністю {1-q} дисперсії однорідні.')
else:
    print("Необхідно збільшити ксть дослідів")
    m += 1
    linear(n, m)

qq = (1 + 0.95) / 2
student_cr_table = t.ppf(df=f3, q=qq)
student_t = kriteriy_studentsa(x_norm[:,1:], y_average, n, m,
dispersion_arr)

print('\nТабличне значення критерій Стюдента:\n', student_cr_table)
print('\nРозрахункове значення критерій Стюдента:\n', student_t)
res_student_t = [temp for temp in student_t if temp > student_cr_table]
final_coefficients = [B[student_t.index(i)] for i in student_t if i in
res_student_t]
print('Коефіцієнти {} статистично незначущі.'.
    format([i for i in B if i not in final_coefficients]))

y_new = []
for j in range(n):
    y_new.append(regression([x[j][student_t.index(i)] for i in student_t
if i in res_student_t], final_coefficients))

print(f'\nОтримаємо значення рівння регресії для {m} дослідів: ')
print(y_new)

d = len(res_student_t)
f4 = n - d
Fp = kriteriy_fishera(y, y_average, y_new, n, m, d, dispersion_arr)
Ft = f.ppf(dfn=f4, dfd=f3, q=1 - 0.05)

print('\nПеревірка адекватності за критерієм Фішера:\n')
print('Розрахункове значення критерія Фішера: Fp =', Fp)
print('Табличне значення критерія Фішера: Ft =', Ft)
if Fp < Ft:
    print('Математична модель адекватна експериментальним даним')
    return True
else:
    print('Математична модель не адекватна експериментальним даним')
    return False

def main(n, m):
    main_1 = linear(n, m)
    if not main_1:
        interaction_effect = with_interaction_effect(n, m)
        if not interaction_effect:
            main(n, m)

if __name__ == '__main__':
    x_range = ((-40, 20), (10, 60), (-20, 20))

    y_max = 200 + int(sum([x[1] for x in x_range]) / 3)
    y_min = 200 + int(sum([x[0] for x in x_range]) / 3)

    main(8, 10)

```



