Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Методи оптимізації та планування експерименту

Лабораторна робота №5

“ ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ КВАДРАТИЧНИХ ЧЛЕНІВ

(ЦЕНТРАЛЬНИЙ ОРТОГОНАЛЬНИЙ КОМПОЗИЦІЙНИЙ ПЛАН)”

Виконав:

студент групи ІВ-83

Гераciмов С. С.

Перевірив:

ас. Регіда П.Г.

Київ

2020 р.

**Мета:** Провести трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Номер у списку: 5

Варіант завдання: 305



**Лістинг програми:**

from copy import deepcopy

from math import sqrt

import numpy as np

from prettytable import PrettyTable

x1\_min = -10

x1\_max = 10

x2\_min = -2

x2\_max = 2

x3\_min = -1

x3\_max = 4

x\_average\_max = (x1\_max + x2\_max + x3\_max) / 3

x\_average\_min = (x1\_min + x2\_min + x3\_min) / 3

y\_max = 200 + x\_average\_max

y\_min = 200 + x\_average\_min

def replace\_column(list\_: list, column, list\_replace):

list\_ = deepcopy(list\_)

for i in range(len(list\_)):

list\_[i][column] = list\_replace[i]

return list\_

def append\_to\_list\_x(x: list, variant: int):

if variant == 1:

for i in range(len(x)):

x[i].append(x[i][1] \* x[i][2])

x[i].append(x[i][1] \* x[i][3])

x[i].append(x[i][2] \* x[i][3])

x[i].append(x[i][1] \* x[i][2] \* x[i][3])

if variant == 2:

for i in range(len(x)):

x[i].append(x[i][1] \* x[i][2])

x[i].append(x[i][1] \* x[i][3])

x[i].append(x[i][2] \* x[i][3])

x[i].append(x[i][1] \* x[i][2] \* x[i][3])

x[i].append(x[i][1] \* x[i][1])

x[i].append(x[i][2] \* x[i][2])

x[i].append(x[i][3] \* x[i][3])

for i in range(len(x)):

for j in range(len(x[i])):

if round(x[i][j], 3) == 0:

x[i][j] = 0

x[i][j] = round(x[i][j], 3)

def get\_value(table: dict, key: int):

value = table.get(key)

if value is not None:

return value

for i in table:

if type(i) == range and key in i:

return table.get(i)

def main(m, n):

if n == 15:

const\_l = 1.215

print(

'ŷ = b0 + b1 \* x1 + b2 \* x2 + b3 \* x3 + b12 \* x1 \* x2 + b13 \* x1 \* x3 + b23 \* x2 \* x3 + b123 \* x1 \* x2 \* '

'x3 + b11 \* x1 \* x1 + b22 \* x2 \* x2 + b33 \* x3 \* x3')

norm\_x = [

[+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],

[+1, -const\_l, 0, 0],

[+1, const\_l, 0, 0],

[+1, 0, -const\_l, 0],

[+1, 0, const\_l, 0],

[+1, 0, 0, -const\_l],

[+1, 0, 0, const\_l],

[+1, 0, 0, 0]

]

delta\_x1 = (x1\_max - x1\_min) / 2

delta\_x2 = (x2\_max - x2\_min) / 2

delta\_x3 = (x2\_max - x3\_min) / 2

x01 = (x1\_min + x1\_max) / 2

x02 = (x2\_min + x2\_max) / 2

x03 = (x3\_min + x3\_max) / 2

x = [

[1, x1\_min, x2\_min, x3\_min],

[1, x1\_min, x2\_max, x3\_max],

[1, x1\_max, x2\_min, x3\_max],

[1, x1\_max, x2\_max, x3\_min],

[1, x1\_min, x2\_min, x3\_max],

[1, x1\_min, x2\_max, x3\_min],

[1, x1\_max, x2\_min, x3\_min],

[1, x1\_max, x2\_max, x3\_max],

[1, -const\_l \* delta\_x1 + x01, x02, x03],

[1, const\_l \* delta\_x1 + x01, x02, x03],

[1, x01, -const\_l \* delta\_x2 + x02, x03],

[1, x01, const\_l \* delta\_x2 + x02, x03],

[1, x01, x02, -const\_l \* delta\_x3 + x03],

[1, x01, x02, const\_l \* delta\_x3 + x03],

[1, x01, x02, x03]

]

append\_to\_list\_x(norm\_x, variant=2)

append\_to\_list\_x(x, variant=2)

if n == 8:

print(

'ŷ = b0 + b1 \* x1 + b2 \* x2 + b3 \* x3 + b12 \* x1 \* x2 + b13 \* x1 \* x3 + b23 \* x2 \* x3 + b123 \* x1 \* x2 \* x3'

)

norm\_x = [

[+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]

]

x = [

[1, x1\_min, x2\_min, x3\_min],

[1, x1\_min, x2\_max, x3\_max],

[1, x1\_max, x2\_min, x3\_max],

[1, x1\_max, x2\_max, x3\_min],

[1, x1\_min, x2\_min, x3\_max],

[1, x1\_min, x2\_max, x3\_min],

[1, x1\_max, x2\_min, x3\_min],

[1, x1\_max, x2\_max, x3\_max]

]

append\_to\_list\_x(norm\_x, variant=1)

append\_to\_list\_x(x, variant=1)

if n == 4:

print('ŷ = b0 + b1 \* x1 + b2 \* x2 + b3 \* x3')

norm\_x = [

[+1, -1, -1, -1],

[+1, -1, +1, +1],

[+1, +1, -1, +1],

[+1, +1, +1, -1],

]

x = [

[1, x1\_min, x2\_min, x3\_min],

[1, x1\_min, x2\_max, x3\_max],

[1, x1\_max, x2\_min, x3\_max],

[1, x1\_max, x2\_max, x3\_min],

]

y = np.random.randint(y\_min, y\_max, size=(n, m))

y\_av = list(np.average(y, axis=1))

for i in range(len(y\_av)):

y\_av[i] = round(y\_av[i], 3)

if n == 15:

t = PrettyTable(['N', 'norm\_x\_0', 'norm\_x\_1', 'norm\_x\_2', 'norm\_x\_3', 'norm\_x\_1\_x\_2', 'norm\_x\_1\_x\_3',

'norm\_x\_2\_x\_3', 'norm\_x\_1\_x\_2\_x\_3', 'norm\_x\_1\_x\_1', 'norm\_x\_2\_x\_2', 'norm\_x\_3\_x\_3', 'x\_0',

'x\_1', 'x\_2', 'x\_3', 'x\_1\_x\_2', 'x\_1\_x\_3', 'x\_2\_x\_3', 'x\_1\_x\_2\_x\_3', 'x\_1\_x\_1', 'x\_2\_x\_2',

'x\_3\_x\_3'] + [f'y\_{i + 1}' for i in range(m)] + ['y\_av'])

if n == 8:

t = PrettyTable(['N', 'norm\_x\_0', 'norm\_x\_1', 'norm\_x\_2', 'norm\_x\_3', 'norm\_x\_1\_x\_2', 'norm\_x\_1\_x\_3',

'norm\_x\_2\_x\_3', 'norm\_x\_1\_x\_2\_x\_3', 'x\_0', 'x\_1', 'x\_2', 'x\_3', 'x\_1\_x\_2', 'x\_1\_x\_3',

'x\_2\_x\_3', 'x\_1\_x\_2\_x\_3'] + [f'y\_{i + 1}' for i in range(m)] + ['y\_av'])

if n == 4:

t = PrettyTable(

['N', 'norm\_x\_0', 'norm\_x\_1', 'norm\_x\_2', 'norm\_x\_3', 'x\_0', 'x\_1', 'x\_2', 'x\_3'] +

[f'y\_{i + 1}' for i in range(m)] + ['y\_av'])

for i in range(n):

t.add\_row([i + 1] + list(norm\_x[i]) + list(x[i]) + list(y[i]) + [y\_av[i]])

print(t)

m\_ij = []

for i in range(len(x[0])):

m\_ij.append([round(sum([x[k][i] \* x[k][j] for k in range(len(x))]) / 15, 3) for j in range(len(x[i]))])

k\_i = []

for i in range(len(x[0])):

a = sum(y\_av[j] \* x[j][i] for j in range(len(x))) / 15

k\_i.append(a)

det = np.linalg.det(m\_ij)

det\_i = [np.linalg.det(replace\_column(m\_ij, i, k\_i)) for i in range(len(k\_i))]

b\_i = [round(i / det, 3) for i in det\_i]

if n == 15:

print(

f"\nНормоване рівняння регресії: "

f"y = {b\_i[0]:.5f} + {b\_i[1]:.5f} \* x1 + {b\_i[2]:.5f} \* x2 + "

f"{b\_i[3]:.5f} \* x3 + {b\_i[4]:.5f} \* x1 \* x2 + "

f"{b\_i[5]:.5f} \* x1 \* x3 + {b\_i[6]:.5f} \* x2 \* x3 + {b\_i[7]:.5f} \* x1 \* x2 \* x3 + {b\_i[8]:.5f} \* x1 \* x1 + "

f"{b\_i[9]:.5f} \* x2 \* x2 + {b\_i[10]:.5f} \* x3 \* x3")

if n == 8:

print(

f"\nНормоване рівняння регресії: "

f"y = {b\_i[0]:.5f} + {b\_i[1]:.5f} \* x1 + {b\_i[2]:.5f} \* x2 + "

f"{b\_i[3]:.5f} \* x3 + {b\_i[4]:.5f} \* x1 \* x2 + "

f"{b\_i[5]:.5f} \* x1 \* x3 + {b\_i[6]:.5f} \* x2 \* x3 + {b\_i[7]:.5f} \* x1 \* x2 \* x3")

if n == 4:

print(

f"\nНормоване рівняння регресії: "

f"y = {b\_i[0]:.5f} + {b\_i[1]:.5f} \* x1 + {b\_i[2]:.5f} \* x2 + {b\_i[3]:.5f} \* x3\n")

check\_i = [round(sum(b\_i[j] \* i[j] for j in range(len(b\_i))), 3) for i in x]

for i in range(len(check\_i)):

print(f'ŷ{i + 1} = {check\_i[i]}, y\_av{i + 1} = {y\_av[i]}')

print("\nКритерій Кохрена:")

f\_1 = m - 1

f\_2 = n

s\_i = [sum([(i - y\_av[j]) \*\* 2 for i in y[j]]) / m for j in range(len(y))]

g\_p = max(s\_i) / sum(s\_i)

table = {3: 0.6841, 4: 0.6287, 5: 0.5892, 6: 0.5598, 7: 0.5365, 8: 0.5175, 9: 0.5017, 10: 0.4884,

range(11, 17): 0.4366, range(17, 37): 0.3720, range(37, 145): 0.3093}

g\_t = get\_value(table, m)

if g\_p < g\_t:

print(f"Дисперія однорідна: Gp = {g\_p:.5} < Gt = {g\_t}")

else:

print(f"Дисперсія не однорідна Gp = {g\_p:.5} < Gt = {g\_t}\nStart again with m = m + 1 = {m + 1}")

return main(m=m + 1, n=n)

print("\nКритерій Ст'юдента:")

s2\_b = sum(s\_i) / n

s2\_beta\_s = s2\_b / (n \* m)

s\_beta\_s = sqrt(s2\_beta\_s)

if n == 15:

beta\_i = b\_i

else:

beta\_i = [sum([norm\_x[i][j] \* y\_av[i] for i in range(len(norm\_x))]) / n for j in range(len(norm\_x[0]))]

beta\_i = [round(i, 3) for i in beta\_i]

t = [abs(i) / s\_beta\_s for i in beta\_i]

f\_3 = f\_1 \* f\_2

t\_table = {8: 2.306, 9: 2.262, 10: 2.228, 11: 2.201, 12: 2.179, 13: 2.160, 14: 2.145, 15: 2.131, 16: 2.120,

17: 2.110, 18: 2.101, 19: 2.093, 20: 2.086, 21: 2.08, 22: 2.074, 23: 2.069, 24: 2.064,

range(25, 30): 2.06, range(30, 40): 2.042, range(40, 60): 2.021, range(60, 100): 2,

range(100, 2 \*\* 100): 1.96}

d = deepcopy(n)

for i in range(len(t)):

if get\_value(t\_table, f\_3) > t[i]:

beta\_i[i] = 0

d -= 1

if n == d:

n = 8 if n == 4 else 15

print(f"n=d\nПроводимо експеримент знову з n = {n}")

return main(m=m + 1, n=n)

if n == 15:

print(

f"Нормоване спрощене рівняння регресії: "

f"y = {beta\_i[0]:.5f} + {beta\_i[1]:.5f} \* x1 + "

f"{beta\_i[2]:.5f} \* x2 + {beta\_i[3]:.5f} \* x3 + {beta\_i[4]:.5f} \* x1 \* x2 + "

f"{beta\_i[5]:.5f} \* x1 \* x3 + {beta\_i[6]:.5f} \* x2 \* x3 + {beta\_i[7]:.5f} \* x1 \* x2 \* x3 + "

f"{beta\_i[8]:.5f} \* x1 \* x1 + {beta\_i[9]:.5f} \* x2 \* x2 + {beta\_i[10]:.5f} \* x3 \* x3")

check\_i = [round(sum(beta\_i[j] \* i[j] for j in range(len(beta\_i))), 3) for i in x]

if n == 8:

print(

f"Нормоване рівняння регресії: "

f"y = {beta\_i[0]:.5f} + {beta\_i[1]:.5f} \* x1 + {beta\_i[2]:.5f} \* x2 + "

f"{beta\_i[3]:.5f} \* x3 + {beta\_i[4]:.5f} \* x1 \* x2 + "

f"{beta\_i[5]:.5f} \* x1 \* x3 + {beta\_i[6]:.5f} \* x2 \* x3 + {beta\_i[7]:.5f} \* x1 \* x2 \* x3")

check\_i = [round(sum(beta\_i[j] \* i[j] for j in range(len(beta\_i))), 3) for i in norm\_x]

if n == 4:

print(

f"Нормоване рівняння регресії: "

f"y = {beta\_i[0]:.5f} + {beta\_i[1]:.5f} \* x1 + {beta\_i[2]:.5f} \* x2 + "

f"{beta\_i[3]:.5f} \* x3")

check\_i = [round(sum(beta\_i[j] \* i[j] for j in range(len(beta\_i))), 3) for i in norm\_x]

for i in range(len(check\_i)):

print(f'ŷ{i + 1} = {check\_i[i]}, y\_av{i + 1} = {y\_av[i]}')

print("\nКритерій Фішера:")

f\_4 = n - d

s2\_ad = m / f\_4 \* sum([(check\_i[i] - y\_av[i]) \*\* 2 for i in range(len(y\_av))])

f\_p = s2\_ad / s2\_b

f\_t = {

1: [164.4, 199.5, 215.7, 224.6, 230.2, 234, 235.8, 237.6],

2: [18.5, 19.2, 19.2, 19.3, 19.3, 19.3, 19.4, 19.4],

3: [10.1, 9.6, 9.3, 9.1, 9, 8.9, 8.8, 8.8],

4: [7.7, 6.9, 6.6, 6.4, 6.3, 6.2, 6.1, 6.1],

5: [6.6, 5.8, 5.4, 5.2, 5.1, 5, 4.9, 4.9],

6: [6, 5.1, 4.8, 4.5, 4.4, 4.3, 4.2, 4.2],

7: [5.5, 4.7, 4.4, 4.1, 4, 3.9, 3.8, 3.8],

8: [5.3, 4.5, 4.1, 3.8, 3.7, 3.6, 3.5, 3.5],

9: [5.1, 4.3, 3.9, 3.6, 3.5, 3.4, 3.3, 3.3],

10: [5, 4.1, 3.7, 3.5, 3.3, 3.2, 3.1, 3.1],

11: [4.8, 4, 3.6, 3.4, 3.2, 3.1, 3, 3],

12: [4.8, 3.9, 3.5, 3.3, 3.1, 3, 2.9, 2.9],

13: [4.7, 3.8, 3.4, 3.2, 3, 2.9, 2.8, 2.8],

14: [4.6, 3.7, 3.3, 3.1, 3, 2.9, 2.8, 2.7],

15: [4.5, 3.7, 3.3, 3.1, 2.9, 2.8, 2.7, 2.7],

16: [4.5, 3.6, 3.2, 3, 2.9, 2.7, 2.6, 2.6],

17: [4.5, 3.6, 3.2, 3, 2.8, 2.7, 2.5, 2.3],

18: [4.4, 3.6, 3.2, 2.9, 2.8, 2.7, 2.5, 2.3],

19: [4.4, 3.5, 3.1, 2.9, 2.7, 2.7, 2.4, 2.3],

range(20, 22): [4.4, 3.5, 3.1, 2.8, 2.7, 2.7, 2.4, 2.3],

range(22, 24): [4.3, 3.4, 3.1, 2.8, 2.7, 2.6, 2.4, 2.3],

range(24, 26): [4.3, 3.4, 3, 2.8, 2.6, 2.5, 2.3, 2.2],

range(26, 28): [4.2, 3.4, 3, 2.7, 2.6, 2.5, 2.3, 2.2],

range(28, 30): [4.2, 3.3, 3, 2.7, 2.6, 2.4, 2.3, 2.1],

range(30, 40): [4.2, 3.3, 3, 2.7, 2.6, 2.4, 2.3, 2.1, 2, 2, 2, 2],

range(40, 60): [4.1, 3.2, 2.9, 2.6, 2.5, 2.3, 2.2, 2, 1.9, 1.9, 1.9, 1.9],

range(60, 120): [4, 3.2, 2.8, 2.5, 2.4, 2.3, 2.1, 1.9, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8],

range(120, 2 \*\* 100): [3.8, 3, 2.6, 2.4, 2.2, 2.1, 2, 2, 1.9, 1.9, 1.9, 1.8, 1.8]

}

if f\_p > get\_value(f\_t, f\_3)[f\_4]:

n = 8 if n == 4 else 15

print(

f"fp = {f\_p} > ft = {get\_value(f\_t, f\_3)[f\_4]}.\n"

f"Математична модель не адекватна\n"

f"Проводимо експеримент знову з m = m + 1 = {m + 1} та n = {n}")

return main(m=m + 1, n=n)

else:

print(

f"fP = {f\_p} < fT = {get\_value(f\_t, f\_3)[f\_4]}.\n"

f"Математична модель адекватна\n")

main(m=3, n=15)

**Результати виконання роботи**

