Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота 3

з дисципліни «Методи оптимізації та планування експерименту»

Виконав:

студент групи ІО-92

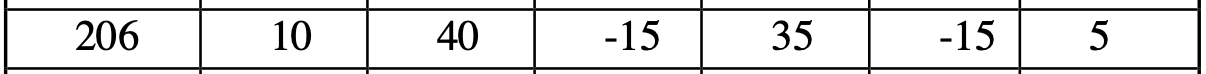
Губський Артур

Перевірив:

Регіда П.Г.

Київ 2021

# Мета: Провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

Варіант: 

**Роздруківка програми:**

import numpy as np

import random

from numpy.linalg import solve

from scipy.stats import f, t

from functools import partial

x\_range = [(10, 40), (-15, 35), (-15, 5)]

x\_aver\_max = (40 + 35 + 5)/ 3

x\_aver\_min = (10 - 15 - 15) / 3

y\_max = 200 + int(x\_aver\_max)

y\_min = 200 + int(x\_aver\_min)

def regression(x, b):

y = sum([x[i]\*b[i] for i in range(len(x))])

return y

def plan\_matrix(n, 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)

x\_norm = 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]])

x\_norm = x\_norm[:len(y)]

x = np.ones(shape=(len(x\_norm), len(x\_norm[0])))

for i in range(len(x\_norm)):

for j in range(1, len(x\_norm[i])):

if x\_norm[i][j] == -1:

x[i][j] = x\_range[j-1][0]

else:

x[i][j] = x\_range[j-1][1]

print('\nМатриця планування')

print(np.concatenate((x, y), axis=1))

return x, y, x\_norm

def find\_coefficient(x, y\_aver, n):

mx1 = sum(x[:, 1]) / n

mx2 = sum(x[:, 2]) / n

mx3 = sum(x[:, 3]) / n

my = sum(y\_aver) / 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

a1 = sum([y\_aver[i] \* x[i][1] for i in range(len(x))]) / n

a2 = sum([y\_aver[i] \* x[i][2] for i in range(len(x))]) / n

a3 = sum([y\_aver[i] \* x[i][3] for i in range(len(x))]) / 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('\nРівняння регресії')

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

return B

# 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(s)

return res

def kriteriy\_cochrena(y, y\_aver, n, m):

S\_kv = s\_kv(y, y\_aver, n, m)

Gp = max(S\_kv) / sum(S\_kv)

print('\nПеревірка за критерієм Кохрена')

return Gp

# оцінки коефіцієнтів

def bs(x, y, y\_aver, n):

res = [sum(1 \* y for y in y\_aver) / n]

for i in range(3): # 4 - ксть факторів

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

# статиcтична оцінка дисперсії

s\_Bs = (s\_kv\_aver / n / m) \*\* 0.5

Bs = bs(x, y, y\_aver, n)

ts = [abs(B) / s\_Bs 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 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)

def main(n, m):

f1 = m - 1

f2 = n

f3 = f1 \* f2

q = 0.05

### табличні значення

student = partial(t.ppf, q=1-0.025)

t\_student = student(df=f3)

G\_kr = cohren(f1, f2)

x, y, x\_norm = plan\_matrix(n, m)

y\_aver = [round(sum(i) / len(i), 2) for i in y]

B = find\_coefficient(x, y\_aver, n)

Gp = kriteriy\_cochrena(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 ксть дослідів")

m += 1

main(n, m)

ts = kriteriy\_studenta(x\_norm[:, 1:], y, y\_aver, n, m)

print('\nКритерій Стьюдента:\n', ts)

res = [t for t in ts if t > t\_student]

final\_k = [B[ts.index(i)] for i in ts if i in res]

print('Коефіцієнти {} статистично незначущі, тому ми виключаємо їх з рівняння.'.format([i for i in B if i not in final\_k]))

y\_new = []

for j in range(n):

y\_new.append(regression([x[j][ts.index(i)] for i in ts if i in res], final\_k))

print(f'\nЗначення "y" з коефіцієнтами {final\_k}')

print(y\_new)

d = len(res)

f4 = n - d

F\_p = kriteriy\_fishera(y, y\_aver, y\_new, n, m, d)

fisher = partial(f.ppf, q=1 - 0.05)

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')

if \_\_name\_\_ == '\_\_main\_\_':

main(4, 4) # 124 стрічка ми реалізуємо функцію main та передаємо їй 2 аргументи: n і m

# m – кількість вимірів y за однією й тією ж самою комбінації факторів

# n – кількість експериментів (рядків матриці планування)

# на 180 ми викликаємо цю функцію та передаємо значення цих аргументів як 4 і 4

**Контрольні запитання**

1. Що називається дробовим факторним експериментом?

Дробовим факторним експериментом називається експеримент з використанням частини повного факторного експерименту

2. Для чого потрібно розрахункове значення Кохрена?

Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій.

3. Для чого перевіряється критерій Стьюдента?

За допомогою критерію Стьюдента перевіряється значущість коефіцієнтів рівняння

4. Чим визначається критерій Фішера і як його застосовувати?

Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваного об”єкта.