**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний Технічний Університет України**

**«Київський Політехнічний Інститут»**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №3**

# з дисципліни «Методи оптимізації та планування експерименту»

на тему: «ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

**Виконав:**

студент 2-го курсу ФІОТ

групи ІО-82

Половінкін П.О.

**Перевірив:**

Регіда П.Г.

**Київ – 2020**

**Варіант:**

C:\Users\1\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\Новый рисунок (10).bmp

**Код програми:**

**import random**

**import numpy as np**

**import copy**

**import math**

**x\_min = [-25, -20, -25]**

**x\_max = [75, 60, -10]**

**x\_average\_min = sum(x\_min) / 3**

**x\_average\_max = sum(x\_max) / 3**

**y\_min = 200 + int(x\_average\_min)**

**y\_max = 200 + int(x\_average\_max)**

**M = 3**

**N = 4**

**cohren\_values = {1: 9065, 2: 7679, 3: 6841, 4: 6287, 5: 5892, 6: 5598, 7: 5365, 8: 5175, 9: 5017,**

**10: 4884, range(11, 17): 4366, range(17, 37): 3720, range(37, 145): 3093}**

**student\_values = {1: 12.71, 2: 4.303, 3: 3.182, 4: 2.776, 5: 2.571,**

**6: 2.447, 7: 2.365, 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: 2.080, 22: 2.074, 23: 2.069, 24: 2.064, 25: 2.060,**

**26: 2.056, 27: 2.052, 28: 2.048, 29: 2.045, 30: 2.042}**

**fisher\_values = {8: {1: 5.3, 2: 4.5, 3: 4.1, 4: 3.8}, 9: {1: 5.1, 2: 4.3, 3: 3.9, 4: 3.6},**

**10: {1: 5.0, 2: 4.1, 3: 3.7, 4: 3.5}, 11: {1: 4.8, 2: 4.0, 3: 3.6, 4: 3.4},**

**12: {1: 4.8, 2: 3.9, 3: 3.5, 4: 3.3}, 13: {1: 4.7, 2: 3.8, 3: 3.4, 4: 3.2},**

**14: {1: 4.6, 2: 3.7, 3: 3.3, 4: 3.1}, 15: {1: 4.4, 2: 3.7, 3: 3.3, 4: 3.1},**

**16: {1: 4.5, 2: 3.6, 3: 3.2, 4: 3.0}, 17: {1: 4.5, 2: 3.6, 3: 3.2, 4: 3.0},**

**18: {1: 4.4, 2: 3.6, 3: 3.2, 4: 2.9}, 19: {1: 4.4, 2: 3.5, 3: 3.1, 4: 2.9},**

**range(20, 22): {1: 4.4, 2: 3.5, 3: 3.1, 4: 2.9}, range(22, 24): {1: 4.3, 2: 3.4, 3: 3.1, 4: 2.8},**

**range(24, 26): {1: 4.3, 2: 3.4, 3: 3.0, 4: 2.8}, range(26, 28): {1: 4.2, 2: 3.4, 3: 3.0, 4: 2.7},**

**range(28, 30): {1: 4.2, 2: 3.3, 3: 3.0, 4: 2.7}, range(30, 40): {1: 4.2, 2: 3.3, 3: 2.9, 4: 2.7},**

**range(40, 60): {1: 4.1, 2: 3.2, 3: 2.9, 4: 2.6}, range(60, 120): {1: 4.0, 2: 3.2, 3: 2.8, 4: 2.5}**

**}**

**def det(arr):**

**return np.linalg.det(np.array(arr))**

**def y(k0=0, k1=0, k2=0, k3=0, x1=0, x2=0, x3=0):**

**return k0 + k1 \* x1 + k2 \* x2 + k3 \* x3**

**def get\_cohren\_criterion(f1):**

**for key in cohren\_values.keys():**

**if type(key) == int:**

**if f1 == key:**

**value = cohren\_values.get(key)**

**break**

**else:**

**if f1 in key:**

**value = cohren\_values.get(key)**

**break**

**else:**

**value = 2500**

**return value / 10000**

**def get\_student\_criterion(f3):**

**for key in student\_values.keys():**

**if f3 == key:**

**value = student\_values.get(key)**

**break**

**else:**

**value = 1.960**

**return value**

**def get\_fisher\_criterion(f3, f4):**

**value = 0**

**row\_last = {1: 3.8, 2: 3.0, 3: 2.6, 4: 2.4}**

**for key\_1 in fisher\_values:**

**if type(key\_1) == int:**

**if f3 == key\_1:**

**for key\_2 in fisher\_values.get(key\_1):**

**if f4 == key\_2:**

**value = fisher\_values.get(key\_1).get(key\_2)**

**break**

**if value:**

**break**

**else:**

**if f3 in key\_1:**

**for key\_2 in fisher\_values.get(key\_1):**

**if f4 == key\_2:**

**value = fisher\_values.get(key\_1).get(key\_2)**

**break**

**if value:**

**break**

**else:**

**for key in row\_last:**

**if f4 == key:**

**value = row\_last.get(key)**

**break**

**return value**

**matrix\_x\_code = [[-1, -1, -1],**

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

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

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

**class Calc:**

**def \_\_init\_\_(self, main\_matrix, m):**

**self.matrix\_x\_code, self.matrix\_x\_natural, self.matrix\_x\_norm, self.matrix\_y = main\_matrix, [], [], []**

**self.matrix\_x\_norm\_full = []**

**self.average\_y, self.m\_x, self.list\_a\_i, self.list\_a\_ii = None, None, None, None**

**self.amount\_of\_x, self.my, self.a12, self.a23, self.a31 = None, None, None, None, None**

**self.denominator, self.change, self.list\_b, self.list\_a = None, None, None, None**

**self.list\_delta, self.list\_x0 = [], []**

**self.m, self.p = m, 0.95**

**self.p = 0.95**

**self.dispersion, self.Coh\_coefficient, self.Coh\_criterion = None, None, None**

**self.f1, self.f2, self.f3, self.f4 = None, None, None, None**

**self.d\_reproducibility, self.d\_evaluation = None, None**

**self.list\_beta, self.list\_t, self.Stud\_criterion = [], [], None**

**self.d, self.d\_adequacy, self.Fish\_coefficient, self.Fish\_criterion = None, None, None, None**

**self.list\_y = None**

**self.main\_caption, self.average\_y\_caption = ["X1", "X2", "X3"], ["Y1", "Y2", "Y3", "Y4"]**

**self.dispersion\_caption = ["D1", "D2", "D3", "D4"]**

**for i in range(self.m):**

**self.main\_caption.append("Y" + str(i + 1))**

**self.string\_main1, self.string\_main2, self.string\_y\_average1, self.string\_y\_average2 = None, None, None, None**

**self.string\_b, self.string\_a, self.matrix\_out = None, None, None**

**self.string\_regression\_caption, self.string\_regression = None, None**

**self.string\_dispersion1, self.string\_dispersion2 = None, None**

**self.string\_cohren, self.string\_cohren\_result1, self.string\_cohren\_result2 = None, None, None**

**self.string\_student\_marks, self.string\_t = None, None**

**self.string\_student\_format, self.string\_regression\_caption\_st = None, None**

**self.string\_student\_regression = None**

**self.string\_fisher\_d, self.string\_fisher = None, None**

**self.string\_fisher\_result1, self.string\_fisher\_result2, self.string\_fisher\_result3 = None, None, None**

**self.values\_regression\_st = []**

**self.initial\_calculation()**

**def initial\_calculation(self):**

**self.amount\_of\_x = len(self.matrix\_x\_code[0])**

**self.matrix\_y = [[random.randint(y\_min, y\_max + 1) for j in range(self.m)] for i in range(N)]**

**self.average\_y = [sum(self.matrix\_y[i]) / self.m for i in range(N)]**

**for i in range(self.amount\_of\_x):**

**self.list\_delta.append(abs(x\_max[i] - x\_min[i]) / 2)**

**self.list\_x0.append((x\_max[i] + x\_min[i]) / 2)**

**for i in range(N):**

**self.matrix\_x\_natural.append([])**

**self.matrix\_x\_norm.append([])**

**for j in range(self.amount\_of\_x):**

**self.matrix\_x\_natural[i].append(x\_min[j]) if matrix\_x\_code[i][j] == -1 \**

**else self.matrix\_x\_natural[i].append(x\_max[j])**

**self.matrix\_x\_norm[i].append((self.matrix\_x\_natural[i][j] - self.list\_x0[j]) /(self.list\_delta[j]))**

**self.matrix\_x\_norm\_full = copy.deepcopy(self.matrix\_x\_norm)**

**for i in range(N):**

**self.matrix\_x\_norm\_full[i].insert(0, 1)**

**self.initial\_format()**

**self.find\_coefficients()**

**def initial\_format(self):**

**self.string\_main1 = "№ " + "{:^6} " \* self.amount\_of\_x + "{:^6} " \* (**

**len(self.main\_caption) - self.amount\_of\_x)**

**self.string\_main2 = "{} " + "| {:+} | " \* self.amount\_of\_x + "|{:^5}| " \* (**

**len(self.main\_caption) - self.amount\_of\_x)**

**self.string\_y\_average1 = "\nСередні значення функції відгуку\n " + "{:^6} " \* N**

**self.string\_y\_average2 = "|{:^5}| "\*N**

**self.string\_b = "\nНатуральні значення факторів\nb0 = {:.3f}, b1 = {:.3f}, b2 = {:.3f}, b3 = {:.3f}"**

**self.string\_a = "\nНормовані значення факторів\na0 = {:.3f}, a1 = {:.3f}, a2 = {:.3f}, a3 = {:.3f}"**

**self.string\_regression\_caption = "Y = {:.3f}{:{sign}.3f}\*x1{:{sign}.3f}\*x2{:{sign}.3f}\*x3"**

**self.string\_regression = "{:.3f}{:{sign}.3f}\*({num1:{sign}})" \**

**"{:{sign}.3f}\*({num2:{sign}})" \**

**"{:{sign}.3f}\*({num3:{sign}}) = {res} / {av\_y}"**

**self.string\_dispersion1 = "\nПеревірка однорідності за критерієм Кохрена\nДисперсії\n " + "{:^6} " \* N**

**self.string\_dispersion2 = "|{:^5}| "\*N**

**self.string\_cohren = "Коефіцієнт Кохрена: {}\nСтупені свободи f1 = {}, f2 = {}\tКритерій Кохрена: {}"**

**self.string\_cohren\_result1 = "Дисперсія однорідна з ймовірністю " + str(self.p)**

**self.string\_cohren\_result2 = "Дисперсія неоднорідна з ймовірністю " + str(self.p)**

**self.string\_student\_marks = "\nПеревірка значущості коефіцієнтів за критерієм Стьюдента\n" \**

**"Оцінка генеральної дисперсії відтворюваності: {:.3f}, " \**

**"статистична оцінка дисперсії: {:.3f}"**

**self.string\_t = "Коефіцієнти Стьюдента\nt0 = {:.3f}, t1 = {:.3f}, t2 = {:.3f}, t3 = {:.3f}"**

**self.string\_student\_format = "Ступені свободи f3 = {}\tКритерій Стьюдента: {}"**

**self.string\_regression\_caption\_st = "Тепер рівняння регресії має вигляд\nY = "**

**self.string\_student\_regression = ""**

**self.string\_fisher\_d = "\nПеревірка адекватності за критерієм Фішера\n" \**

**"Кількість значущих коефіцієнтів d = {}, Дисперсія адекватності: {:.3f}"**

**self.string\_fisher = "Коефіцієнт Фішера: {}\nСтупені свободи f3 = {}, f4 = {}\tКритерій Фішера: {}"**

**self.string\_fisher\_result1 = "Модель адекватна експериментальним даним з ймовірністю " + str(self.p)**

**self.string\_fisher\_result2 = "Модель неадекватна експериментальним даним з ймовірністю " + str(self.p)**

**self.string\_fisher\_result3 = "Неможливо використовувати критерій Фішера тому, що N = d"**

**self.matrix\_out = copy.deepcopy(self.matrix\_x\_natural)**

**for i in range(N):**

**for j in range(self.m):**

**self.matrix\_out[i].append(self.matrix\_y[i][j])**

**def table\_update(self):**

**self.main\_caption.append("Y" + str(self.m))**

**self.string\_main1 += "{:^6} "**

**self.string\_main2 += "|{:^5}| "**

**def find\_coefficients(self):**

**self.out\_table()**

**self.list\_b = self.coefficients(self.matrix\_x\_natural)**

**self.out\_regression\_caption(True)**

**self.list\_a = self.coefficients(self.matrix\_x\_norm)**

**self.out\_regression\_caption(False)**

**if self.check\_cohren():**

**self.check\_student()**

**self.check\_fisher()**

**def coefficients(self, matrix):**

**reverse\_matrix = list(zip(\*matrix))**

**self.average\_y = [sum(self.matrix\_y[i]) / self.m for i in range(N)]**

**self.m\_x = [sum(column) / N for column in reverse\_matrix]**

**self.my = sum(self.average\_y) / N**

**self.list\_a\_i = [sum([reverse\_matrix[row][col] \* self.average\_y[col] for col in range(N)]) / N**

**for row in range(self.amount\_of\_x)]**

**self.list\_a\_ii = [sum([reverse\_matrix[row][col] \*\* 2 for col in range(N)]) / N for row in**

**range(self.amount\_of\_x)]**

**self.a12, self.a23, self.a31 = 0, 0, 0**

**for i in range(N):**

**self.a12 += reverse\_matrix[0][i] \* reverse\_matrix[1][i]**

**self.a23 += reverse\_matrix[1][i] \* reverse\_matrix[2][i]**

**self.a31 += reverse\_matrix[2][i] \* reverse\_matrix[0][i]**

**self.a12, self.a23, self.a31 = self.a12 / 4, self.a23 / 4, self.a31 / 4**

**self.denominator = [[1, self.m\_x[0], self.m\_x[1], self.m\_x[2]],**

**[self.m\_x[0], self.list\_a\_ii[0], self.a12, self.a31],**

**[self.m\_x[1], self.a12, self.list\_a\_ii[1], self.a23],**

**[self.m\_x[2], self.a31, self.a23, self.list\_a\_ii[2]]]**

**denominator\_det = det(self.denominator)**

**self.change = [self.my, self.list\_a\_i[0], self.list\_a\_i[1], self.list\_a\_i[2]]**

**reverse\_denominator = list(map(list, zip(\*self.denominator)))**

**list\_k = []**

**for index in range(len(reverse\_denominator)):**

**numerator = reverse\_denominator[:]**

**numerator[index] = self.change**

**list\_k.append(det(list(zip(\*numerator))) / denominator\_det)**

**return list\_k**

**def add\_y(self):**

**for i in range(N):**

**r = random.randint(y\_min, y\_max + 1)**

**self.matrix\_y[i].append(r)**

**self.matrix\_out[i].append(r)**

**self.m += 1**

**self.table\_update()**

**self.find\_coefficients()**

**def check\_cohren(self):**

**self.dispersion = [sum([(self.matrix\_y[row][col] - self.average\_y[row]) \*\* 2**

**for col in range(self.m)]) / self.m**

**for row in range(N)]**

**self.Coh\_coefficient = max(self.dispersion) / sum(self.dispersion)**

**self.f1, self.f2 = self.m - 1, N**

**self.Coh\_criterion = get\_cohren\_criterion(self.f1)**

**if self.Coh\_coefficient <= self.Coh\_criterion:**

**flag = True**

**self.out\_cohren(flag)**

**return flag**

**else:**

**flag = False**

**self.out\_cohren(flag)**

**self.add\_y()**

**return flag**

**def check\_student(self):**

**self.d\_reproducibility = sum(self.dispersion) / N**

**self.d\_evaluation = self.d\_reproducibility /(self.m \* N)**

**self.d\_evaluation = math.sqrt(self.d\_evaluation)**

**reverse\_matrix\_norm = list(zip(\*self.matrix\_x\_norm\_full))**

**for col in range(N):**

**self.list\_beta.append(sum([self.average\_y[row] \* reverse\_matrix\_norm[col][row] for row in range(N)]) / N)**

**self.list\_t.append(abs(self.list\_beta[col]) / self.d\_evaluation)**

**self.f3 = self.f1 \* self.f2**

**self.Stud\_criterion = get\_student\_criterion(self.f3)**

**for index in range(N):**

**if self.list\_t[index] >= self.Stud\_criterion:**

**continue**

**else:**

**self.list\_b[index] = 0**

**self.list\_y = [y(\*self.list\_b, \*self.matrix\_x\_natural[row]) for row in range(N)]**

**self.out\_student()**

**def check\_fisher(self):**

**self.d = N - self.list\_b.count(0)**

**if N - self.d == 0:**

**self.out\_fisher(False)**

**else:**

**self.d\_adequacy =(self.m /(N - self.d)) \* sum(**

**[(self.list\_y[row] - self.average\_y[row]) \*\* 2 for row in range(N)])**

**self.Fish\_coefficient = self.d\_adequacy / self.d\_reproducibility**

**self.f4 = N - self.d**

**self.Fish\_criterion = get\_fisher\_criterion(self.f3, self.f4)**

**if self.Fish\_coefficient <= self.Fish\_criterion:**

**self.out\_fisher(True, True)**

**else:**

**self.out\_fisher(True, False)**

**def out\_table(self):**

**print("m =", self.m)**

**print(self.string\_main1.format(\*self.main\_caption))**

**for row in range(len(self.matrix\_out)):**

**print(self.string\_main2.format(row + 1, \*self.matrix\_out[row]))**

**print(self.string\_y\_average1.format(\*self.average\_y\_caption))**

**print(self.string\_y\_average2.format(\*list(map(round, self.average\_y))))**

**def out\_regression\_caption(self, flag):**

**if flag:**

**print(self.string\_b.format(\*self.list\_b))**

**print(self.string\_regression\_caption.format(\*self.list\_b, sign="+"))**

**self.out\_regression(flag)**

**else:**

**print(self.string\_a.format(\*self.list\_a))**

**print(self.string\_regression\_caption.format(\*self.list\_a, sign="+"))**

**self.out\_regression(flag)**

**def out\_regression(self, flag):**

**if flag:**

**matrix, list\_k = self.matrix\_x\_natural, self.list\_b**

**else:**

**matrix, list\_k = self.matrix\_x\_norm, self.list\_a**

**index = 0**

**for x1, x2, x3 in matrix:**

**print(self.string\_regression.format(\*list\_k, num1=x1, num2=x2, num3=x3, sign="+",**

**res=y(\*list\_k, x1, x2, x3), av\_y=self.average\_y[index]))**

**index += 1**

**def out\_cohren(self, flag):**

**print(self.string\_dispersion1.format(\*self.dispersion\_caption))**

**print(self.string\_dispersion2.format(\*list(map(round, self.dispersion))))**

**print(self.string\_cohren.format(self.Coh\_coefficient, self.f1, self.f2, self.Coh\_criterion))**

**if flag:**

**print(self.Coh\_coefficient, "<=", self.Coh\_criterion)**

**print(self.string\_cohren\_result1)**

**else:**

**print(self.Coh\_coefficient, ">", self.Coh\_criterion)**

**print(self.string\_cohren\_result2)**

**def out\_student(self):**

**print(self.string\_student\_marks.format(self.d\_reproducibility, self.d\_evaluation))**

**print(self.string\_t.format(\*self.list\_t))**

**print(self.string\_student\_format.format(self.f3, self.Stud\_criterion))**

**for index in range(N):**

**if self.list\_b[index]:**

**print(self.list\_t[index], ">=", self.Stud\_criterion,**

**"|b" + str(index) + " - значимий з ймовірністю " + str(self.p))**

**else:**

**print(self.list\_t[index], "<", self.Stud\_criterion,**

**"|b" + str(index) + " - незначимий з ймовірністю " + str(self.p))**

**self.out\_regression\_student()**

**def out\_regression\_student(self):**

**if sum(self.list\_b) == 0:**

**print(self.string\_regression\_caption\_st + "0")**

**elif sum(self.list\_b[1:]) == 0:**

**print(self.string\_regression\_caption\_st + str(round(self.list\_b[0], 3)))**

**else:**

**if self.list\_b[0]:**

**self.string\_regression\_caption\_st += "{:.3f}"**

**self.string\_student\_regression += "{:.3f}"**

**for i in range(N):**

**self.values\_regression\_st.append([self.list\_b[0]])**

**else:**

**for i in range(N):**

**self.values\_regression\_st.append([])**

**for i in range(1, N):**

**if self.list\_b[i]:**

**self.string\_regression\_caption\_st += "{:{sign}.3f}\*x" + str(i)**

**self.string\_student\_regression +="{:{sign}.3f}\*({:{sign}})"**

**else:**

**continue**

**index = 1**

**for row in range(N):**

**for col in range(self.amount\_of\_x):**

**if self.list\_b[index]:**

**self.values\_regression\_st[row].append(self.list\_b[index])**

**self.values\_regression\_st[row].append(self.matrix\_x\_natural[row][col])**

**index+=1**

**index = 1**

**list\_out = [self.list\_b[i] for i in range(N) if self.list\_b[i] != 0]**

**self.string\_student\_regression += " = {res}"**

**print(self.string\_regression\_caption\_st.format(\*list\_out, sign="+"))**

**for row in range(N):**

**print(self.string\_student\_regression.format(\*self.values\_regression\_st[row], sign="+",**

**res=self.list\_y[row]))**

**def out\_fisher(self, check, flag=True):**

**if not check:**

**print(self.string\_fisher\_result3)**

**else:**

**print(self.string\_fisher\_d.format(self.d, self.d\_adequacy))**

**print(self.string\_fisher.format(self.Fish\_coefficient, self.f3, self.f4, self.Fish\_criterion))**

**if flag:**

**print(self.Fish\_coefficient, "<=", self.Fish\_criterion)**

**print(self.string\_fisher\_result1)**

**else:**

**print(self.Fish\_coefficient, ">", self.Fish\_criterion)**

**print(self.string\_fisher\_result2)**

**calc = Calc(matrix\_x\_code, M)**

**Контрольні запитання:**

1. *Що називається дробовим факторним експериментом?*

У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Якщо буде використовуватися лінійна регресія, то можливо зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів слід скоротити, використовуючи для планування так звані регулярні дробові репліки від повного факторного експерименту, що містять відповідну кількість дослідів і зберігають основні властивості матриці планування – це означає дробовий факторний експеримент (ДФЕ).

1. *Для чого потрібно розрахункове значення Кохрена?*

Для перевірки дисперсії на однорідність.

1. *Для чого перевіряється критерій Стьюдента?*

Для перевірки значущості коефіцієнтів регресії. Тобто, Якщо виконується нерівність *ts*< *tтабл*, то приймається нуль-гіпотеза, тобто вважається, що знайдений коефіцієнт *βs* є статистично незначущим і його слід виключити з рівняння регресії. Якщо *ts* > *tтабл* то гіпотеза не підтверджується, тобто *βs* – значимий коефіцієнт і він залишається в рівнянні регресії.

1. *Чим визначається критерій Фішера і як його застосовувати?*

Отримане рівняння регресії необхідно перевірити на адекватність досліджуваному об'єкту. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності. Адекватність моделі перевіряють за F-критерієм Фішера, який дорівнює відношенню дисперсії адекватності до дисперсії відтворюваності.