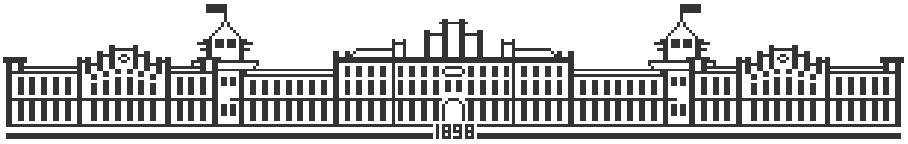

Міністерство освіти та науки України

Національний технічний університет України “Київський політехнічний інститут”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №6
з дисципліни «Методи оптимізації та планування»
на тему: «Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

Виконав:
студент 2-го курсу ФІОТ
групи ІО-92
Іванов Р. О.
Перевірив:
асистент
Регіда П. Г.

Київ – 2021

Варіант:

Варіант №209

209	-20	15	-30	45	-30	-15	5,4+3,4*x1+9,6*x2+6,8*x3+3,1*x1*x1+0,1*x2*x2+1,2*x3*x3+0,8*x1*x2+0,8*x1*x3+9,9*x2*x3+4,5*x1*x2*x3

Код програми:

```
import math
import random
from _decimal import Decimal
from scipy.stats import f, t
import numpy
from itertools import compress
from functools import reduce

xmin = [-20, -30, -30]
xmax = [15, 45, -15]
norm_plan_raw = [[-1, -1, -1],
                 [-1, +1, +1],
                 [+1, -1, +1],
                 [+1, +1, -1],
                 [-1, -1, +1],
                 [-1, +1, -1],
                 [+1, -1, -1],
                 [+1, +1, +1],
                 [-1.73, 0, 0],
                 [+1.73, 0, 0],
                 [0, -1.73, 0],
                 [0, +1.73, 0],
                 [0, 0, -1.73],
                 [0, 0, +1.73]]

x0 = [(xmax[_] + xmin[_])/2 for _ in range(3)]
dx = [xmax[_] - x0[_] for _ in range(3)]

natur_plan_raw = [[xmin[0],           xmin[1],           xmin[2]],
                  [xmin[0],           xmin[1],           xmax[2]],
                  [xmin[0],           xmax[1],           xmin[2]],
                  [xmin[0],           xmax[1],           xmax[2]],
                  [xmax[0],           xmin[1],           xmin[2]],
                  [xmax[0],           xmin[1],           xmax[2]],
                  [xmax[0],           xmax[1],           xmin[2]],
                  [xmax[0],           xmax[1],           xmax[2]],
                  [-1.73*dx[0]+x0[0], x0[1],             x0[2]],
                  [1.73*dx[0]+x0[0],  x0[1],             x0[2]],
                  [x0[0],             -1.73*dx[1]+x0[1], x0[2]],
                  [x0[0],             1.73*dx[1]+x0[1],  x0[2]],
                  [x0[0],             x0[1],             -1.73*dx[2]+x0[2]],
                  [x0[0],             x0[1],             1.73*dx[2]+x0[2]],
                  [x0[0],             x0[1],             x0[2]]]


def equation_of_regression(x1, x2, x3, cef, importance=[] * 11):
    factors_array = [1, x1, x2, x3, x1 * x2, x1 * x3, x2 * x3, x1 * x2 * x3, x1
** 2, x2 ** 2, x3 ** 2]
    return sum([el[0] * el[1] for el in compress(zip(cef, factors_array),
importance)])


def func(x1, x2, x3):
    coeffs = [5.4, 3.4, 9.6, 6.8, 3.1, 0.1, 1.2, 0.8, 0.8, 9.9, 4.5]
    return equation_of_regression(x1, x2, x3, coeffs)


def generate_factors_table(raw_array):
```

```
    raw_list = [row + [row[0] * row[1], row[0] * row[2], row[1] * row[2], row[0]
* row[1] * row[2]] + list(
        map(lambda x: x ** 2, row)) for row in raw_array]
    return list(map(lambda row: list(map(lambda el: round(el, 3), row)),
raw_list))


def generate_y(m, factors_table):
    return [[round(func(row[0], row[1], row[2]) + random.randint(-5, 5), 3) for
_ in range(m)] for row in factors_table]


def print_matrix(m, N, factors, y_vals, additional_text=":"):
    labels_table = list(map(lambda x: x.ljust(10),
                            ["x1", "x2", "x3", "x12", "x13", "x23", "x123",
"x1^2", "x2^2", "x3^2"] + [
                                "y{}".format(i + 1) for i in range(m)]))
    rows_table = [list(factors[i]) + list(y_vals[i]) for i in range(N)]
    print("\nМатриця планування" + additional_text)
    print(" ".join(labels_table))
    print("\n".join([" ".join(map(lambda j: "{:<+10}".format(j), rows_table[i]))
for i in range(len(rows_table))]))
    print("\t")


def print_equation(coeffs, importance=[True] * 11):
    x_i_names = list(compress(["", "x1", "x2", "x3", "x12", "x13", "x23",
"x123", "x1^2", "x2^2", "x3^2"], importance))
    coefficients_to_print = list(compress(coeffs, importance))
    equation = " ".join(
        ["".join(i) for i in zip(list(map(lambda x: "{:+.2f}".format(x),
coefficients_to_print)), x_i_names)])
    print("Рівняння регресії: y = " + equation)


def set_factors_table(factors_table):
    def x_i(i):
        with_null_factor = list(map(lambda x: [1] + x,
generate_factors_table(factors_table)))
        res = [row[i] for row in with_null_factor]
        return numpy.array(res)

    return x_i


def m_ij(*arrays):
    return numpy.average(reduce(lambda accum, el: accum * el, list(map(lambda
el: numpy.array(el), arrays))))


def find_coefficients(factors, y_vals):
    x_i = set_factors_table(factors)
    coeffs = [[m_ij(x_i(column), x_i(row)) for column in range(11)] for row in
range(11)]
    y_numpy = list(map(lambda row: numpy.average(row), y_vals))
    free_values = [m_ij(y_numpy, x_i(i)) for i in range(11)]
    beta_coefficients = numpy.linalg.solve(coeffs, free_values)
    return list(beta_coefficients)


def cochran_criteria(m, N, y_table):
    def get_cochran_value(f1, f2, q):
        partResult1 = q / f2
        params = [partResult1, f1, (f2 - 1) * f1]
        fisher = f.isf(*params)
        result = fisher / (fisher + (f2 - 1))
```

```
        return Decimal(result).quantize(Decimal('.0001')).__float__()

    print("Перевірка за критерієм Кохрена: m = {}, N = {}".format(m, N))
    y_variations = [numpy.var(i) for i in y_table]
    max_y_variation = max(y_variations)
    gp = max_y_variation / sum(y_variations)
    f1 = m - 1
    f2 = N
    p = 0.95
    q = 1 - p
    gt = get_cochran_value(f1, f2, q)
    print("Gp = {}; Gt = {}; f1 = {}; f2 = {}; q = {:.2f}".format(gp, gt, f1,
f2, q))
    if gp < gt:
        print("Gp < Gt => дисперсії рівномірні => все правильно")
        return True
    else:
        print("Gp > Gt => дисперсії нерівномірні => змінюємо значення m")
        return False


def student_criteria(m, N, y_table, beta_coefficients):
    def get_student_value(f3, q):
        return Decimal(abs(t.ppf(q / 2,
f3))).quantize(Decimal('.0001')).__float__()

    print("\nПеревірка за критерієм Стьюдента: m = {}, N = {} ".format(m, N))
    average_variation = numpy.average(list(map(numpy.var, y_table)))
    variation_beta_s = average_variation / N / m
    standard_deviation_beta_s = math.sqrt(variation_beta_s)
    t_i = [abs(beta_coefficients[i]) / standard_deviation_beta_s for i in
range(len(beta_coefficients))]
    f3 = (m - 1) * N
    q = 0.05
    t_our = get_student_value(f3, q)
    importance = [True if el > t_our else False for el in list(t_i)]
    # print result data
    print("Оцінки коефіцієнтів βs: " + ", ".join(list(map(lambda x:
str(round(float(x), 3)), beta_coefficients))))
    print("Коефіцієнти ts: " + ", ".join(list(map(lambda i: "{:.2f}".format(i),
t_i))))
    print("f3 = {}; q = {}; tтабл = {}".format(f3, q, t_our))
    beta_i = ["β0", "β1", "β2", "β3", "β12", "β13", "β23", "β123", "β11", "β22",
"β33"]
    importance_to_print = ["важливий" if i else "неважливий" for i in
importance]
    to_print = map(lambda x: x[0] + " " + x[1], zip(beta_i,
importance_to_print))
    print(*to_print, sep="; ")
    print_equation(beta_coefficients, importance)
    return importance


def fisher_criteria(m, N, d, x_table, y_table, b_coefficients, importance):
    def get_fisher_value(f3, f4, q):
        return Decimal(abs(f.isf(q, f4,
f3))).quantize(Decimal('.0001')).__float__()

    f3 = (m - 1) * N
    f4 = N - d
    q = 0.05
    theoretical_y = numpy.array([equation_of_regression(row[0], row[1], row[2],
b_coefficients) for row in x_table])
    average_y = numpy.array(list(map(lambda el: numpy.average(el), y_table)))
    s_ad = m / (N - d) * sum((theoretical_y - average_y) ** 2)
    y_variations = numpy.array(list(map(numpy.var, y_table)))
```

```
    s_v = numpy.average(y_variations)
    f_p = float(s_ad / s_v)
    f_t = get_fisher_value(f3, f4, q)
    theoretical_values_to_print = list(
        zip(map(lambda x: "x1 = {0[1]:<10} x2 = {0[2]:<10} x3 =
{0[3]:<10}".format(x), x_table), theoretical_y))
    print("\nПеревірка за критерієм Фішера: m = {}, N = {} для таблиці
y_table".format(m, N))
    print("Теоретичні значення Y для різних комбінацій факторів:")
    print("\n".join(["{arr[0]}: y = {arr[1]}".format(arr=el) for el in
theoretical_values_to_print]))
    print("Fp = {}, Ft = {}".format(f_p, f_t))
    print("Fp < Ft => модель адекватна" if f_p < f_t else "Fp > Ft => модель
неадекватна")
    return True if f_p < f_t else False


m = 3
N = 15
natural_plan = generate_factors_table(natur_plan_raw)
y_arr = generate_y(m, natur_plan_raw)
while not cochran_criteria(m, N, y_arr):
    m += 1
    y_arr = generate_y(m, natural_plan)

print_matrix(m, N, natural_plan, y_arr, " для натуралізованих факторів:")
coefficients = find_coefficients(natural_plan, y_arr)
print_equation(coefficients)
importance = student_criteria(m, N, y_arr, coefficients)
d = len(list(filter(None, importance)))
fisher_criteria(m, N, d, natural_plan, y_arr, coefficients, importance)
```

Результат виконання програми:

```
Перевірка за критерієм Кохрена: m = 3, N = 15
Gp = 0.1653116531165312; Gt = 0.3346; f1 = 2; f2 = 15; q = 0.05
Gp < Gt => дисперсії рівномірні => все правильно

Матриця планування для натуралізованих факторів:
x1        x2        x3        x12       x13       x23       x123      x1^2      x2^2      x3^2      y1        y2        y3
-20       -30       -30       +600      +600      +900      -18000    +400      +900      +900      -4        +1        +2
-20       -30       -15       +600      +300      +450      -9000     +400      +900      +225      +0        -3        -2
-20       +45       -30       -900      +600      -1350     +27000    +400      +2025     +900      -1        +0        +1
-20       +45       -15       -900      +300      -675      +13500    +400      +2025     +225      -5        +3        +0
+15       -30       -30       -450      -450      +900      +13500    +225      +900      +900      +3        +0        +1
+15       -30       -15       -450      -225      +450      +6750     +225      +900      +225      +4        +1        -2
+15       +45       -30       +675      -450      -1350     -20250    +225      +2025     +900      -5        +1        +1
+15       +45       -15       +675      -225      -675      -10125    +225      +2025     +225      -1        +1        +4
-32.775   +7.5      -22.5     -245.812  +737.438  -168.75   +5530.781 +1074.201 +56.25     +506.25   +0        +0        -3
+27.775   +7.5      -22.5     +208.312  -624.938  -168.75   -4687.031 +771.451  +56.25     +506.25   +4        -5        -1
-2.5       -57.375   -22.5     +143.438  +56.25    +1290.938 -3227.344 +6.25     +3291.891 +506.25   -1        -1        -3
-2.5       +72.375   -22.5     -180.938  +56.25    -1628.438 +4071.094 +6.25     +5238.141 +506.25   -1        +2        -5
-2.5       +7.5      -35.475   -18.75    +88.688   -266.062  +665.156  +6.25     +56.25    +1258.476 -3        +2        -4
-2.5       +7.5      -9.525    -18.75    +23.812   -71.438   +178.594  +6.25     +56.25    +90.726   -3        +4        +2
-2.5       +7.5      -22.5     -18.75    +56.25    -168.75   +421.875  +6.25     +56.25    +506.25   +4        +1        +4



Рівняння регресії: y = -4.54 +0.09x1 +0.05x2 -0.71x3 +0.00x12 +0.00x13 +0.00x23 +0.00x123 -0.00x1^2 -0.00x2^2 -0.02x3^2

Перевірка за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: -4.536, 0.088, 0.049, -0.71, 0.001, 0.003, 0.002, 0.0, -0.004, -0.001, -0.017
Коефіцієнти ts: 13.01, 0.25, 0.14, 2.04, 0.00, 0.01, 0.00, 0.00, 0.01, 0.00, 0.05
f3 = 30; q = 0.05; tтабл = 2.0423
```

β0 важливий; β1 неважливий; β2 неважливий; β3 неважливий; β12 неважливий; β13 неважливий; β23 неважливий; β123 неважливий; β11 неважливий; β22 неважливий; β33 неважливий

```
Рівняння регресії: y = -4.54

Перевірка за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення Y для різних комбінацій факторів:
x1 = -30        x2 = -30        x3 = 600        : y = 0
x1 = -30        x2 = -15        x3 = 600        : y = 0
x1 = 45         x2 = -30        x3 = -900       : y = 0
x1 = 45         x2 = -15        x3 = -900       : y = 0
x1 = -30        x2 = -30        x3 = -450       : y = 0
x1 = -30        x2 = -15        x3 = -450       : y = 0
x1 = 45         x2 = -30        x3 = 675        : y = 0
x1 = 45         x2 = -15        x3 = 675        : y = 0
x1 = 7.5        x2 = -22.5      x3 = -245.812   : y = 0
x1 = 7.5        x2 = -22.5      x3 = 208.312    : y = 0
x1 = -57.375    x2 = -22.5      x3 = 143.438    : y = 0
x1 = 72.375     x2 = -22.5      x3 = -180.938   : y = 0
x1 = 7.5        x2 = -35.475    x3 = -18.75     : y = 0
x1 = 7.5        x2 = -9.525     x3 = -18.75     : y = 0
x1 = 7.5        x2 = -22.5      x3 = -18.75     : y = 0
Fp = 1.084494773519164, Ft = 2.0374
Fp < Ft => модель адекватна
```

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.