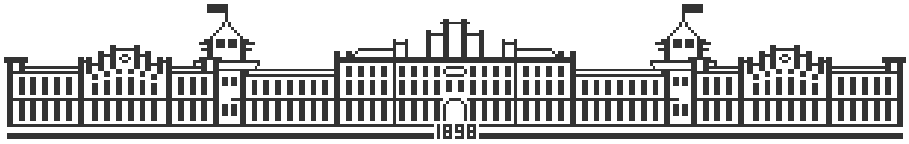

Міністерство освіти та науки України

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 ІВ-71
Мазан Я. В.

Перевірив:
асистент
Регіда П. Г.

Київ – 2019

Варіант:

Варіант №109

109	-20	15	10	60	15	35	3,2+9,1*x1+4,1*x2+3,7*x3+2,1*x1*x1+0,8*x2*x2+4,9*x3*x3+3,7*x1*x2+0,5*x1*x3+1,0*x2*x3+0,3*x1*x2*x3

Код програми:

```
import math
import random
from _decimal import Decimal
from itertools import compress
from scipy.stats import f, t
import numpy
from functools import reduce
def regression_equation(x1, x2, x3, coefficients, importance = [True]*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(coefficients, factors_array), importance)])
def func(x1,x2,x3):
    coefficients = [3.2, 9.1, 4.1, 3.7, 3.7, 0.5, 1.0, 0.3, 2.1, 0.8, 4.9]
    return regression_equation(x1, x2, x3, coefficients)
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]]
natur_plan_raw =  [[-20, +10, 15],
                   [-20,    +60,    35],
                   [+15,    +10,    35],
                   [+15,    +60,    15],
                   [-20,    +10,    35],
                   [-20,    +60,    15],
                   [+15,    +10,    15],
                   [+15,    +60,    35],
                   [-32.775, +35,    25],
                   [+27.775, +35,    25],
                   [-2.5,    -18.25, 25],
                   [-2.5,    +78.25, 25],
                   [-2.5,    +35,    7.7],
                   [-2.5,    +35,    42.3],
                   [-2.5,    +35,    25]]
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(coefficients, 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(coefficients, 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)
    coefficients = [[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(coefficients, 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 рівномірності дисперсій 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 - треба ще експериментів")
        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 значимості коефіцієнтів регресії за критерієм Стьюдента: m = {}, N = {} ".format(m, N))
    average_variation = numpy.average(list(map(numpy.var, y_table)))
    x_i = set_factors_table(natural_plan)
    variation_beta_s = average_variation/N/m
    standard_deviation_beta_s = math.sqrt(variation_beta_s)
    t_i = numpy.array([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([regression_equation(row[0], row[1], row[2], b_coefficients) for row in x_table])
    # print(theoretical_y)
    average_y = numpy.array(list(map(lambda el: numpy.average(el), y_table)))
    s_ad = m/(N-d) * sum((theoretical_y - average_y)**2)
    # print(s_ad)
    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 адекватності моделі 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:

```
/home/yan/PycharmProjects/Optimisation&PlannningLab6/venv/bin/python /home/yan/PycharmProjects/Optimisation&PlannningLab6/main.py
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15
Gp = 0.161504424778761; 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        +10        +15        -200       -300       +150       -3000      +400       +100       +225       +297.2     +300.2     +304.2
-20        +60        +35        -1200      -700       +2100      -42000     +400       +3600      +1225      -5371.8    -5367.8    -5371.8
+15        +10        +35        +150       +525       +350       +5250      +225       +100       +1225      +9612.7    +9603.7    +9604.7
+15        +60        +15        +900       +225       +900       +13500     +225       +3600      +225       +13284.7   +13286.7   +13289.7
-20        +10        +35        -200       -700       +350       -7000      +400       +100       +1225      +4079.2    +4071.2    +4076.2
-20        +60        +15        -1200      -300       +900       -18000     +400       +3600      +225       -4146.8    -4143.8    -4144.8
+15        +10        +15        +150       +225       +150       +2250      +225       +100       +225       +3386.7    +3385.7    +3379.7
+15        +60        +35        +900       +525       +2100      +31500     +225       +3600      +1225      +25011.7   +25016.7   +25011.7
-32.775    +35        +25        -1147.125  -819.375   +875       -28678.125 +1074.201  +1225      +625       -6144.219  -6142.219  -6146.219
+27.775    +35        +25        +972.125   +694.375   +875       +24303.125 +771.451   +1225      +625       +18263.486 +18269.486 +18268.486
-2.5        -18.25     +25        +45.625    -62.5      -456.25    +1140.625  +6.25      +333.062   +625       +3361.7    +3364.7    +3362.7
-2.5        +78.25     +25        -195.625   -62.5      +1956.25   -4890.625  +6.25      +6123.062  +625       +8097.85   +8104.85   +8098.85
-2.5        +35        +7.7       -87.5      -19.25     +269.5     -673.75    +6.25      +1225      +59.29     +1169.086  +1175.086  +1167.086
-2.5        +35        +42.3      -87.5      -105.75    +1480.5    -3701.25   +6.25      +1225      +1789.29   +10032.606 +10031.606 +10037.606
-2.5        +35        +25        -87.5      -62.5      +875       -2187.5    +6.25      +1225      +625       +4138.825  +4139.825  +4136.825

Рівняння регресії: y = +0.03 +9.19x1 +4.15x2 +4.11x3 +3.70x12 +0.50x13 +1.00x23 +0.30x123 +2.10x1^2 +0.80x2^2 +4.89x3^2

Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 0.027, 9.192, 4.147, 4.107, 3.698, 0.496, 1.002, 0.3, 2.098, 0.798, 4.891
Коефіцієнти ts:         0.07, 23.83, 10.75, 10.65, 9.59, 1.29, 2.60, 0.78, 5.44, 2.07, 12.68
f3 = 30; q = 0.05; tтабл = 2.0423
β0 неважливий; β1 важливий; β2 важливий; β3 важливий; β12 важливий; β13 неважливий; β23 важливий; β123 неважливий; β11 важливий; β22 важливий; β33 важливий
Рівняння регресії: y = +9.19x1 +4.15x2 +4.11x3 +3.70x12 +1.00x23 +2.10x1^2 +0.80x2^2 +4.89x3^2

Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = 10         x2 = 15         x3 = -200      : y = 300.365028198491
x1 = 60         x2 = 35         x3 = -1200     : y = -5371.457163167608
x1 = 10         x2 = 35         x3 = 150       : y = 9607.52474495849
x1 = 60         x2 = 15         x3 = 900       : y = 13287.839328035117
x1 = 10         x2 = 35         x3 = -200      : y = 4074.796640976838
x1 = 60         x2 = 15         x3 = -1200     : y = -4145.5554426131785
x1 = 10         x2 = 15         x3 = 150       : y = 3385.093132180186
x1 = 60         x2 = 35         x3 = 900       : y = 25013.604274147394
x1 = 35         x2 = 25         x3 = -1147.125 : y = -6142.846324541145
x1 = 35         x2 = 25         x3 = 972.125   : y = 18265.68579534713
x1 = -18.25     x2 = 25         x3 = 45.625    : y = 3362.78639159351
x1 = 78.25      x2 = 25         x3 = -195.625  : y = 8100.799374738629
x1 = 35         x2 = 7.7        x3 = -87.5     : y = 1169.7159271132066
x1 = 35         x2 = 42.3       x3 = -87.5     : y = 10034.550117220226
x1 = 35         x2 = 25         x3 = -87.5     : y = 4138.364727483119
Fp = 0.5671693860790351, Ft = 2.3343
Fp < Ft => модель адекватна

Process finished with exit code 0
```