Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

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

ВИКОНАВ:
Студент II курсу
ФІОТ
Групи ІО-93
Май Тієн Ноанг
Варіант - 9318

ПЕРЕВІРИВ:
асистент Регіда
П.Г.

Київ 2021 р.

Мета:
Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

Варіант завдання:

318	20	70	-15	45	20	35	2,4+0,5*x1+6,0*x2+10,0*x3+4,2*x1*x1+0,9*x2*x2+2,2*x3*x3+0,8*x1*x2+0,9*x1*x3+7,2*x2*x3+4,7*x1*x2*x3

Лістинг 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
import matplotlib.pyplot as plot


def regression_equation(x1, x2, x3, coeffs, 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(coeffs, factors_array), imp
ortance)])


def func(x1, x2, x3):
    coeffs = [2.4, 0.5, 6.0, 10.0, 4.2, 0.9, 2.2, 0.8, 0.9, 7.2, 4.7]
    return regression_equation(x1, x2, x3, coeffs)

xmin = [20, -15, 20]
xmax = [70, 45, 35]
x0 = [(xmax[_] + xmin[_])/2 for _ in range(3)]
dx = [xmax[_] - x0[_] for _ in range(3)]
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 = [[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 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_lis
t))


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), coeffic
ients_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(fa
ctors_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 r
ange(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 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)))
    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("Оцінки коефіцієнтів βs: " + ", ".join(list(map(lambda x: str(round(fl
oat(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)
    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_c
oefficients) 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}".form
at(x), x_table), theoretical_y))
    print("\nПеревірка адекватності моделі за критерієм Фішера: m = {}, N = {} д
ля 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 theoretic
al_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 неа
декватна")
    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 роботи програми:

```
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15
Gp = 0.14855875831485588; 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       -15       +20       -300      +400      -300      -6000     +400      +225      +400      -2380.6     -2375.6     -2377.6
+20       -15       +35       -300      +700      -525      -10500    +400      +225      +1225     -2178.1     -2173.1     -2174.1
+20       +45       +20       +900      +400      +900      +18000    +400      +2025     +400      +37826.4    +37818.4    +37819.4
+20       +45       +35       +900      +700      +1575     +31500    +400      +2025     +1225     +54405.9    +54406.9    +54401.9
+70       -15       +20       -1050     +1400     -300      -21000    +4900     +225      +400      -12547.6    -12551.6    -12549.6
+70       -15       +35       -1050     +2450     -525      -36750    +4900     +225      +1225     -20677.1    -20672.1    -20672.1
+70       +45       +20       +3150     +1400     +900      +63000    +4900     +2025     +400      +88245.4    +88243.4    +88250.4
+70       +45       +35       +3150     +2450     +1575     +110250   +4900     +2025     +1225     +132505.9   +132502.9   +132509.9
+1.75     +15.0     +27.5     +26.25    +48.125   +412.5    +721.875  +3.062    +225.0    +756.25   +7181.969   +7182.969   +7183.969
+88.25    +15.0     +27.5     +1323.75  +2426.875 +412.5    +36403.125 +7788.062 +225.0    +756.25   +50372.094  +50370.094  +50366.094
+45.0     -36.9     +27.5     -1660.5   +1237.5   -1014.75  -45663.75 +2025.0   +1361.61  +756.25   -29368.833  -29365.833  -29367.833
+45.0     +66.9     +27.5     +3010.5   +1237.5   +1839.75  +82788.75 +2025.0   +4475.61  +756.25   +122335.867 +122342.867 +122333.867
+45.0     +15.0     +14.525   +675.0    +653.625  +217.875  +9804.375 +2025.0   +225.0    +210.976  +16438.323  +16436.323  +16443.323
+45.0     +15.0     +40.475   +675.0    +1821.375 +607.125  +27320.625 +2025.0  +225.0    +1638.226 +39323.223  +39330.223  +39331.223
+45.0     +15.0     +27.5     +675.0    +1237.5   +412.5    +18562.5  +2025.0   +225.0    +756.25   +27090.025  +27095.025  +27092.025

Рівняння регресії: y = +3.99 +0.52x1 +6.05x2 +9.75x3 +4.20x12 +0.90x13 +2.20x23 +0.80x123 +0.90x1^2 +7.20x2^2 +4.71x3^2
```

```
Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 3.985, 0.521, 6.046, 9.755, 4.197, 0.898, 2.199, 0.8, 0.901, 7.2, 4.706
Коефіцієнти ts: 10.34, 1.35, 15.69, 25.32, 10.89, 2.33, 5.71, 2.08, 2.34, 18.68, 12.21
f3 = 30; q = 0.05; tтабл = 2.0423
β0 важливий; β1 неважливий; β2 важливий; β3 важливий; β12 важливий; β13 важливий; β23 важливий; β123 важливий; β11 важливий; β22 важливий; β33 важливий
Рівняння регресії: y = +3.99 +6.05x2 +9.75x3 +4.20x12 +0.90x13 +2.20x23 +0.80x123 +0.90x1^2 +7.20x2^2 +4.71x3^2
β0 важливий; β1 неважливий; β2 важливий; β3 важливий; β12 важливий; β13 важливий; β23 важливий; β123 важливий; β11 важливий; β22 важливий; β33 важливий
Рівняння регресії: y = +3.99 +6.05x2 +9.75x3 +4.20x12 +0.90x13 +2.20x23 +0.80x123 +0.90x1^2 +7.20x2^2 +4.71x3^2

Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = -15        x2 = 20         x3 = -300       : y = -2378.790726876588
x1 = -15        x2 = 35         x3 = -300       : y = -2175.923995386681
x1 = 45         x2 = 20         x3 = 900        : y = 37821.37178795825
x1 = 45         x2 = 35         x3 = 900        : y = 54404.90518611713
x1 = -15        x2 = 20         x3 = -1050      : y = -12550.769522039025
x1 = -15        x2 = 35         x3 = -1050      : y = -20674.902790543187
x1 = 45         x2 = 20         x3 = 3150       : y = 88246.0596594643
x1 = 45         x2 = 35         x3 = 3150       : y = 132505.92639096014
x1 = 15.0       x2 = 27.5       x3 = 26.25      : y = 7183.3866736889995
x1 = 15.0       x2 = 27.5       x3 = 1323.75    : y = 50370.56669139702
x1 = -36.9      x2 = 27.5       x3 = -1660.5    : y = -29365.76300996546
x1 = 66.9       x2 = 27.5       x3 = 3010.5     : y = 122337.35314070141
x1 = 15.0       x2 = 14.525     x3 = 675.0      : y = 16440.13790960382
x1 = 15.0       x2 = 40.475     x3 = 675.0      : y = 39328.960688420826
x1 = 15.0       x2 = 27.5       x3 = 675.0      : y = 27092.347287850273
Fp = 0.8991770096906953, Ft = 2.5336
Fp < Ft => модель адекватна
```

Висновов: При виконання лабораторній роботі я навчився находжувати трьохфакторний експеримент і отримав адекватну модель – рівняння