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 №3

З дисципліни «Методи оптимізації та планування»

Загальні принципи організації експериментів з довільними значеннями факторів

ВИКОНАВ: Ралло Ігор
Студент 2 курсу
ФІОТ гр. ІО-92

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

Мета:

Вивчити основні поняття, визначення, принципи теорії планування експерименту, на основі яких вивчити побудову формалізованих алгоритмів проведення експерименту і отримання формалізованої моделі об’єкта. Закріпити отримані знання практичним їх використанням при написанні програми, що реалізує завдання на лабораторну роботу.

Варіант завдання:

216	-10	50	25	65	-10	15

Лістинг програми:

```
import random
import numpy


def cochran(dispersion, m, G_table):
    return max(dispersion) / sum(dispersion) < G_table[m -
1]


def student(general_dispersion, m, y_mean, xn):
    statistic_dispersion = (general_dispersion / (N * m))
** 0.5

    beta = [1 / N * sum(y_mean[j] for j in range(N))]
    for i in range(3):
        b = 0
        for j in range(N):
            b += y_mean[j] * xn[j][i]
        beta.append(1 / N * b)

    t = [abs(i) / statistic_dispersion for i in beta]

    return t[0] > t_table[(m - 1)*N], t[1] > t_table[(m -
1)*N], t[2] > t_table[(m - 1)*N], t[3] > t_table[(m -
1)*N]


def phisher(m, d, y_mean, yo, dispersion_reproducibility,
```

```
F_table):
    dispersion_adequacy = 0
    for i in range(N): dispersion_adequacy += (yo[i] -
y_mean[i]) ** 2

    dispersion_adequacy = dispersion_adequacy * m / (N -
d)
    fp = dispersion_adequacy / dispersion_reproducibility

    return fp < F_table[N - d][(m - 1) * N]


def coef(x, y_mean):
    mx1, mx2, mx3 = sum([x[i][0] for i in range(N)]) / N,
sum([x[i][1] for i in range(N)]) / N, \
                    sum([x[i][2] for i in range(N)]) / N

    my = sum(y_mean) / N

    a11, a22, a33 = sum([x[i][0] ** 2 for i in range(N)])
/ N, sum([x[i][1] ** 2 for i in range(N)]) / N, \
                    sum([x[i][2] ** 2 for i in range(N)])
/ N

    a12, a13, a23 = sum([x[i][0] * x[i][1] for i in
range(N)]) / N, sum([x[i][0] * x[i][2] for i in range(N)])
/ N, \
                    sum([x[i][1] * x[i][2] for i in
range(N)]) / N

    a1, a2, a3 = sum([x[i][0] * y_mean[i] for i in
range(N)]) / N, sum([x[i][1] * y_mean[i] for i in
range(N)]) / N, \
                  sum([x[i][2] * y_mean[i] for i in
range(N)]) / N

    a = numpy.array([[1, mx1, mx2, mx3], [mx1, a11, a12,
a13], [mx2, a12, a22, a23], [mx3, a13, a23, a33]])
    c = numpy.array([my, a1, a2, a3])
    b = numpy.linalg.solve(a, c)
    return b
```

```
x1_min, x1_max = -10, 50
x2_min, x2_max = 25, 65
x3_min, x3_max = -10, 15

x_avg_min, x_avg_max = (x1_min + x2_min + x3_min) / 3,
(x1_max + x2_max + x3_max) / 3
y_min, y_max = 200 + x_avg_min, 200 + x_avg_max

xn = [[-1, -1, -1],
      [-1, +1, +1],
      [+1, -1, +1],
      [+1, +1, -1]]

x = [[x1_min, x2_min, x3_min],
     [x1_min, x2_max, x3_max],
     [x1_max, x2_min, x3_max],
     [x1_max, x2_max, x3_min]]

m = 3
N = 4

y = [[random.randint(int(y_min), int(y_max)) for i in
range(m)] for j in range(4)]

G_table = {1: 0.9065, 2: 0.7679, 3: 0.6841, 4: 0.6287, 5:
0.5892, 6: 0.5598, 7: 0.5365, 8: 0.5175, 9: 0.5017, 10:
0.488}
t_table = {4: 2.776, 8: 2.306, 12: 2.179, 16: 2.120, 20:
2.086, 24: 2.064, 28: 2.048}
F_table = {1: {4: 7.7, 8: 5.3, 12: 4.8, 16: 4.5, 20: 4.4,
24: 4.3, 28: 4.2},
           2: {4: 6.9, 8: 4.5, 12: 3.9, 16: 3.6, 20: 3.5,
24: 3.4, 28: 3.3},
           3: {4: 6.6, 8: 4.1, 12: 3.5, 16: 3.2, 20: 3.1,
24: 3.0, 28: 3.0},
           4: {4: 6.4, 8: 3.8, 12: 3.3, 16: 3.0, 20: 2.9,
24: 2.8, 28: 2.7},
           5: {4: 6.3, 8: 3.7, 12: 3.1, 16: 2.9, 20: 2.7,
```

```
24: 2.6, 28: 2.6},
           6: {4: 6.2, 8: 3.6, 12: 3.0, 16: 2.7, 20: 2.6,
24: 2.5, 28: 2.4}}

while True:
    while True:
        if m > 8:
            print("Error! Поточне значення m більше ніж
максимальне значення за таблицею розподілу для критерію
Стьюдента.")
            exit(0)

        y_mean = [sum(y[i]) / m for i in range(N)]

        dispersion = []
        for i in range(len(y)):
            dispersion.append(0)
            for j in range(m):
                dispersion[i] += (y_mean[i] - y[i][j]) **
2
            dispersion[i] /= m

        dispersion_reproducibility = sum(dispersion) / N

        if cochran(dispersion, m, G_table):
            break
        else:
            m += 1
            for i in range(N):
                y[i].append(random.randint(int(y_min),
int(y_max)))

    k = student(dispersion_reproducibility, m, y_mean, xn)
    d = sum(k)

    b = coef(x, y_mean)
    b = [b[i] * k[i] for i in range(N)]

    yo = []
    for i in range(N):
```

```
        yo.append(b[0] + b[1] * x[i][0] + b[2] * x[i][1] +
b[3] * x[i][2])

    if d == N:
        m += 1
        for i in range(N):
            y[i].append(random.randint(int(y_min),
int(y_max)))

    elif phisher(m, d, y_mean, yo,
dispersion_reproducibility, F_table):
        break
    else:
        m += 1
        for i in range(N):
            y[i].append(random.randint(int(y_min),
int(y_max)))


table_values = ["#", "X1", "X2", "X3"]
for i in range(m): table_values.append("Yi{:d}".format(i +
1))

row_format = "|{:^11}" * (len(table_values))
header_separator_format = "+{0:=^11s}" *
(len(table_values))
separator_format = "+{0:-^11s}" * (len(table_values))


print('\n\tx1 min:', x1_min, '\tx1 max:', x1_max, '\n\tx2
min:', x2_min, '\t\tx2 max:', x2_max,
      '\n\tx3 min:', x3_min, '\tx3 max:', x3_max, '\n\ty
min:', "{:.2f}".format(y_min), '\ty max:', y_max, "\n\n" +
      header_separator_format.format("=") + "+\n" +
row_format.format(*table_values) + "|\n" +
      header_separator_format.format("=") + "+")

for i in range(4):
    print("|{:^11}|{:^11}|{:^11}|{:^11}|".format(i+1,
*x[i]), end="")
```

```
    for j in y[i]: print("{:^11}|".format(j), end="")
    print()

print(separator_format.format("-") + "+" +
f"\n\n\tОтримане рівняння регресії при m={m}: Y =
{b[0]:.2f}", end='')
for i in range(1, 4):
    if b[i] != 0: print(" + {0:.2f}".format(b[i]) + "*X" +
str(i), end="")
print("\n\n\tЗа критерієм Кохрена поточна дисперсія -
однорідна."
      "\n\tЗа критерієм Стьюдента значущість: b0 - {}, b1
- {}, b2 - {}, b3 - {}.".format(*k),
      "\n\tЗа критерієм Фішера рівняння регресії адекватно
оригіналу.")

print(f'\n\tКількість значущих коефіцієнтів: {d}')

print("\n\t\t\t\t Перевірка:")
print("+{0:-^14s}+{0:-^37s}+".format("-"))
for i in range(4):
    print("| Ys{0:1d} = {1:5.2f} | b0 + b1*X1 + b2*X2 +
b3*X3 = {2:^7.2f}|\n".format(i+1, y_mean[i], b[0] + b[1] *
x[i][0] + b[2] * x[i][1]), end="")
    print("+{0:-^14s}+{0:-^37s}+".format("-"))
```

Результат виконання роботи:

```
    x1 min: -10     x1 max: 50
    x2 min: 25      x2 max: 65
    x3 min: -10     x3 max: 15
    y min: 201.67   y max: 243.33333333333334
```

#	X1	X2	X3	Yi1	Yi2	Yi3
1	-10	25	-10	220	229	221
2	-10	65	15	241	236	224
3	50	25	15	217	231	234
4	50	65	-10	236	212	202

```
    Отримане рівняння регресії при m=3: Y = 226.55

    За критерієм Кохрена поточна дисперсія - однорідна.
    За критерієм Стьюдента значущість: b0 - True, b1 - False, b2 - False, b3 - False.
    За критерієм Фішера рівняння регресії адекватно оригіналу.

    Кількість значущих коефіцієнтів: 1
```

Перевірка:

Ys1 = 223.33	b0 + b1*X1 + b2*X2 + b3*X3 = 226.55
Ys2 = 233.67	b0 + b1*X1 + b2*X2 + b3*X3 = 226.55
Ys3 = 227.33	b0 + b1*X1 + b2*X2 + b3*X3 = 226.55
Ys4 = 216.67	b0 + b1*X1 + b2*X2 + b3*X3 = 226.55