Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 5

з дисципліни «Методи наукових досліджень»

на тему «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

(центральний ортогональний композиційний план)»

ВИКОНАВ:

студент 2 курсу

групи ІВ-91

Желепа В. В.

Залікова – 9110

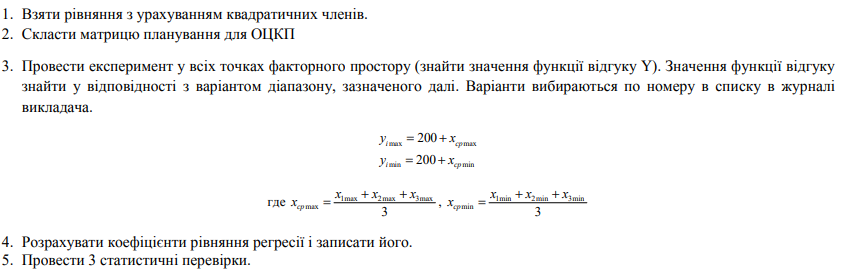
ПЕРЕВІРИВ:

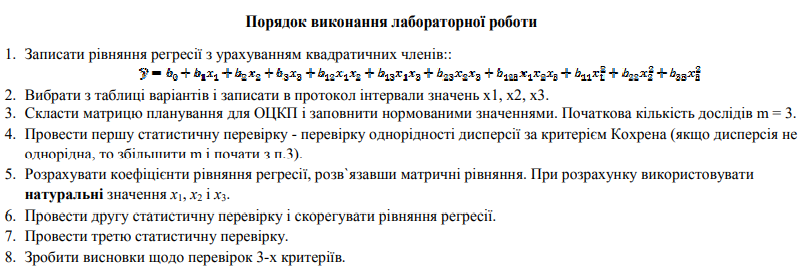
ас. Регіда П. Г.

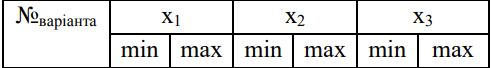
Київ – 2021

**Мета**: Провести трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

**Завдання:**





Варіанти обираються по номеру в списку в журналі викладача. 



**Програмний код**

*from* random *import* \*  
*from* pprint *import* pprint  
*from* math *import* sqrt  
*from* scipy.stats *import* f  
*from* scipy.stats *import* t *as* t\_check  
*import* sklearn.linear\_model *as* lm  
N, d, l = 15, 8, 1.215  
*print*("y` = b0 + b1\*x1 + b2\*x2 + b3\*x3 + b12\*x1\*x2 + b13x1\*x3 + b23\*x2\*x3 + b123\*x1\*x2\*x3 + b11\*x1\*\*2 + b22\*x2\*\*2 + b33\*x3\*\*2")  
*def* create\_mat(*N*, *m*):  
 mat\_sX = [[-4, 3], [-6, 10], [0, 3]]  
 mat\_1X = [[-1, -1, -1], [-1, -1, 1], [-1, 1, -1], [-1, 1, 1], [1, -1, -1], [1, -1, 1], [1, 1, -1], [1, 1, 1], [-1.215, 0, 0], [1.215, 0, 0], [0, -1.215, 0], [0, 1.215, 0], [0, 0, -1.215], [0, 0, 1.215], [0, 0, 0]]  
 mat\_2X = [[-1, -1, -1], [-1, -1, 1], [-1, 1, -1], [-1, 1, 1], [1, -1, -1], [1, -1, 1], [1, 1, -1], [1, 1, 1],  
 [-1.215, 0, 0], [1.215, 0, 0], [0, -1.215, 0], [0, 1.215, 0], [0, 0, -1.215], [0, 0, 1.215], [0, 0, 0]]  
 x\_min\_max = [*sum*(mat\_sX[i][k] *for* i *in range*(3))/3 *for* k *in range*(2)]  
 y\_min\_max = [*int*(200 + x\_min\_max[i]) *for* i *in range*(2)]  
 *print*('Задана матриця Х:\n', mat\_sX)  
 *print*('Xср min and max:\n', x\_min\_max, '\nY min and max:\n', y\_min\_max, '\nМатриця Y:')  
 mat\_Y = [[randint(y\_min\_max[0], y\_min\_max[1]) *for* i *in range*(*m*)] *for* k *in range*(*N*)]  
 pprint(mat\_Y)  
 mat\_serY = [*sum*(mat\_Y[k1])/*m for* k1 *in range*(*N*)]  
 *print*('Середні значення Y:\n', mat\_serY, '\nНормована матриця Х:')  
 mat\_X0i = [*sum*(mat\_sX[i]) / 2 *for* i *in range*(3)]  
 mat\_dX = [mat\_sX[i][1] - mat\_X0i[i] *for* i *in range*(3)]  
 mat\_X = [[], [], [], [], [], [], [], [], [], [], [], [], [], [], []]  
 *for* i *in range*(15):  
 *for* j *in range*(3, 10):  
 mat\_2X[i].append(mat\_2X[i][0] \* mat\_2X[i][1])  
 mat\_2X[i].append(mat\_2X[i][0] \* mat\_2X[i][2])  
 mat\_2X[i].append(mat\_2X[i][1] \* mat\_2X[i][2])  
 mat\_2X[i].append(mat\_2X[i][0] \* mat\_2X[i][1] \* mat\_2X[i][2])  
 mat\_2X[i].append(mat\_2X[i][0] \*\* 2)  
 mat\_2X[i].append(mat\_2X[i][1] \*\* 2)  
 mat\_2X[i].append(mat\_2X[i][2] \*\* 2)  
 *break  
 for* i *in range*(15):  
 mat\_2X[i].insert(0, 1)  
 pprint(mat\_2X)  
 *for* i *in range*(8):  
 *for* j *in range*(10):  
 *if* j < 3:  
 *if* mat\_1X[i][j] == -1:  
 mat\_X[i].append(mat\_sX[j][0])  
 *else*:  
 mat\_X[i].append(mat\_sX[j][1])  
 *if* j > 3:  
 mat\_X[i].append(mat\_X[i][0] \* mat\_X[i][1])  
 mat\_X[i].append(mat\_X[i][0] \* mat\_X[i][2])  
 mat\_X[i].append(mat\_X[i][1] \* mat\_X[i][2])  
 mat\_X[i].append(mat\_X[i][0] \* mat\_X[i][1] \* mat\_X[i][2])  
 mat\_X[i].append(mat\_X[i][0] \*\* 2)  
 mat\_X[i].append(mat\_X[i][1] \*\* 2)  
 mat\_X[i].append(mat\_X[i][2] \*\* 2)  
 *break  
  
 for* i *in range*(8, 15):  
 *for* j *in range*(10):  
 *if* j < 3:  
 *if* mat\_1X[i][j] == 0:  
 mat\_X[i].append(mat\_X0i[j])  
 *else*:  
 mat\_X[i].append(mat\_1X[i][j] \* mat\_dX[j] + mat\_X0i[j])  
 *else*:  
 mat\_X[i].append(mat\_X[i][0] \* mat\_X[i][1])  
 mat\_X[i].append(mat\_X[i][0] \* mat\_X[i][2])  
 mat\_X[i].append(mat\_X[i][1] \* mat\_X[i][2])  
 mat\_X[i].append(mat\_X[i][0] \* mat\_X[i][1] \* mat\_X[i][2])  
 mat\_X[i].append(mat\_X[i][0] \*\* 2)  
 mat\_X[i].append(mat\_X[i][1] \*\* 2)  
 mat\_X[i].append(mat\_X[i][2] \*\* 2)  
 *break  
  
 return* (mat\_X, mat\_Y, mat\_serY, mat\_2X)  
  
*def* coef\_b(*x*, *y*):  
 *for* i *in range*(15):  
 *x*[i].insert(0, 1)  
 *print*('Нормалізована матриця Х:')  
 pprint(*x*, *width*=150)  
 skm = lm.LinearRegression(*fit\_intercept*=*False*)  
 skm.fit(*x*, *y*)  
 b = skm.coef\_  
 b = [*round*(i, 3) *for* i *in* b]  
 *print*(b)  
  
 *return* b  
  
  
*def* check(*mat\_serY*, *mat\_Y*, *tran1*, *blist*, *matt\_fullX*, *m*):  
 d = 11  
 d1 = 11  
 mat\_disY = [*sum*([((k1 - *mat\_serY*[j]) \*\* 2) *for* k1 *in mat\_Y*[j]]) / *m for* j *in range*(N)]  
 *print*("Дисперсії в рядках:\n", mat\_disY)  
 *print*(  
 '-------------------------------------------------------------------------\nПЕРЕВІРКА ОДНОРІДНОСТІ ДИСПЕРСІЇ ЗА КРИТЕРІЄМ КОХРЕНА:\n\n...\n..')  
 *if max*(mat\_disY) / *sum*(mat\_disY) < 0.7679:  
 *print*('Дисперсія однорідна')  
 *else*:  
 *print*('Дисперсія неоднорідна')  
 *m* += 1  
 main(N, *m*)  
 *print*(  
 '-------------------------------------------------------------------------\nПЕРЕВІРКА ЗНАЧУЩОСТІ КОЕФІЦІЄНТІВ ЗА КРИТЕРІЄМ СТЬЮДЕНТА:\n')  
 S2b = *sum*(mat\_disY) / N  
 S2bs = S2b / (*m* \* N)  
 Sbs = sqrt(S2bs)  
 *print*('Sbs:\n', Sbs)  
 bb = [*sum*(*mat\_serY*[k] \* *tran1*[i][k] *for* k *in range*(N)) / N *for* i *in range*(d)]  
 t = [*abs*(bb[i]) / Sbs *for* i *in range*(d1)]  
 *print*('bi:\n', bb, '\nti:\n', t, '\n...\n..')  
 f1, f2 = *m* - 1, N  
 f3 = f1 \* f2  
 *for* i *in range*(d1):  
 *if* t[i] < t\_check.ppf(*q*=0.975, *df*=f3):  
 *blist*[i] = 0  
 d -= 1  
 *print*('Виключаємо з рівняння коефіціент b', i)  
 y\_reg = [  
 *blist*[0] \* *matt\_fullX*[i][0] + *blist*[1] \* *matt\_fullX*[i][1] + *blist*[2] \* *matt\_fullX*[i][2] + *blist*[3] \* *matt\_fullX*[i][3] + *blist*[4] \*  
 *matt\_fullX*[i][4] + *blist*[5] \* *matt\_fullX*[i][5] + *blist*[6] \* *matt\_fullX*[i][6] + *blist*[7] \* *matt\_fullX*[i][7] + *blist*[8] \* *matt\_fullX*[i][8] + *blist*[9] \* *matt\_fullX*[i][9] + *blist*[10] \* *matt\_fullX*[i][10] *for* i  
 *in range*(N)]  
 *print*('Значення рівнянь регресій:\n', y\_reg)  
 *print*(  
 '-------------------------------------------------------------------------\nПЕРЕВІРКА АДЕКВАТНОСТІ ЗА КРИТЕРІЄМ ФІШЕРА:\n')  
 f4 = N - d  
 Sad = (*m* / (N - d)) \* *int*(*sum*(y\_reg[i] - *mat\_serY*[i] *for* i *in range*(N)) \*\* 2)  
 Fp = Sad / S2b  
 *print*('Кількість значимих коефіціентів:\n', d, '\nFp:\n', Fp, '\n...\n..')  
 *if* Fp > f.ppf(*q*=0.95, *dfn*=f4, *dfd*=f3):  
 *print*('Рівняння регресії неадекватно оригіналу при рівні значимості 0.05')  
 *else*:  
 *print*('Рівняння регресії адекватно оригіналу при рівні значимості 0.05')  
  
  
*def* main(*N*, *m*):  
 x, mat\_Y, mat\_serY, mat\_2X = create\_mat(*N*, *m*)  
 tran1 = [*list*(i) *for* i *in zip*(\*mat\_2X)]  
 b = coef\_b(x, mat\_serY)  
 check(mat\_serY, mat\_Y, tran1, b, x, *m*)  
  
  
main(15, 3)

**Результат роботи програми**

