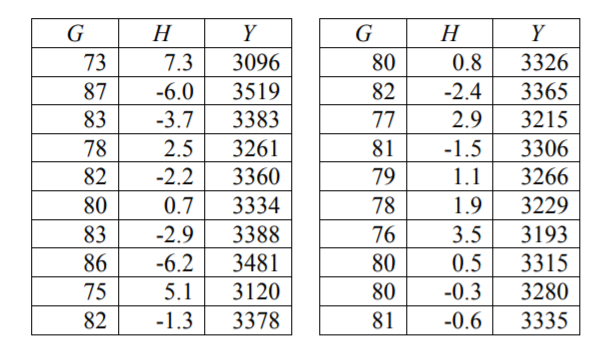
**Data analysis tools for Experimental research/DOE and data analysis Class work\_1**

|  |  |
| --- | --- |
|  |  |

**Multiple regression/ model selection**:

Suppose you have performed an experiment to collect the response variable Y while varying two factors H and G (two predictor variables).



[Paper and pencil]

1. Perform multiple regression assuming a model Y= β0+β1G+β2H. Show the formulation of the objective function and derive the equations that need to be solved to minimize the sum of squares. Calculate the regression coefficients from β0, β1, and β2 from the data. Provide a surface plot for Y as a function of G and H using the equation.
2. Also, perform the regression assuming an interaction between G and H. Y= β0+β1G+β2H+β12GH2. Provide a surface plot for Y as a function of G and H.

Show the calculations (paper and pencil) and verify the answers using regression function from sklearn

Code hints:

**Load the data from excel file**

data = pd.read\_excel('regression.xlsx', header=0)

G = data.G

H = data.H

Y = data.Y

**Regression Code:**

one\_col = np.ones(np.size(G))

X = np.column\_stack((one\_col,G,H))

Or

X = np.column\_stack((one\_col, G, H, G\*H))

def fit(X, Y):

        coeff = np.linalg.inv(X.T.dot(X)).dot(X.T.dot(Y))

        return coeff

# Refer Gauss-Markov theorem and OLS regression to understand the formula

**Plotting Surface Plot:**

#To plot surface get some equally spaced values of G and H

G1 = np.linspace(min(G), max(G), 30)

H1 = np.linspace(min(H), max(H), 30)

#This is required for plotting surface

G1, H1 = np.meshgrid(G1, H1)

#Get the values of B0, B1, B2 from the fit function as mentioned above

B0 =

B1 =

B2 =

print(B0, B1, B2)

Z = B0 + B1\*G1 + B2\*H1

%matplotlib widget

#Here we are plotting the points in 3D

plt.figure(figsize=(4, 4))

ax = plt.axes(projection='3d')

ax.scatter3D(G, H, Y, color='blue')

#We are plotting the surface as shown in figure

ax.plot\_surface(G1, H1, Z, alpha=0.5, cmap='viridis', linewidth=0)

ax.set\_xlabel('G')

ax.set\_ylabel('H')

ax.set\_zlabel('Y')

plt.show()

**Built in regression function:**

from sklearn.linear\_model import LinearRegression as regression

X = np.column\_stack((G,H))

model = regression().fit(X,Y)

print(model.intercept\_)

print(model.coef\_)

