In [3]:
```
#Author: Suryoday Basak
#suryodaybasak.info
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
plt.style.use('ggplot')
mpl.rcParams['figure.figsize'] = (10,8)
```

In [4]:
```
#Reading the data
df = pd.read_csv('../datsets/cars/mtcars.csv')
print(df)
```

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am \
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0
5	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0
6	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0
7	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0
8	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0
9	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0
10	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0
11	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0
12	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0
13	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0
14	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0
15	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0
16	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0
17	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1
18	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1
19	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1
20	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0
21	Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0
22	AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0
23	Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0
24	Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0
25	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1
26	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1
27	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	

```
  1
28       Ford Pantera L  15.8    8  351.0  264  4.22  3.170  14.50   0
  1
29         Ferrari Dino  19.7    6  145.0  175  3.62  2.770  15.50   0
  1
30        Maserati Bora  15.0    8  301.0  335  3.54  3.570  14.60   0
  1
31           Volvo 142E  21.4    4  121.0  109  4.11  2.780  18.60   1
  1

    gear  carb
0      4     4
1      4     4
2      4     1
3      3     1
4      3     2
5      3     1
6      3     4
7      4     2
8      4     2
9      4     4
10     4     4
11     3     3
12     3     3
13     3     3
14     3     4
15     3     4
16     3     4
17     4     1
18     4     2
19     4     1
20     3     1
21     3     2
22     3     2
23     3     4
24     3     2
25     4     1
26     5     2
27     5     2
28     5     4
29     5     6
30     5     8
31     4     2
```

In [5]:
```
data_mat = df.values
r,c = np.shape(data_mat)
print(np.shape(data_mat))
print(r)
ones_stub = np.ones((r,))
data_mat = np.c_[data_mat, ones_stub]
print(data_mat)
```

```
(32, 12)
32
[['Mazda RX4' 21.0 6 160.0 110 3.9 2.62 16.46 0 1 4 4 1.0]
 ['Mazda RX4 Wag' 21.0 6 160.0 110 3.9 2.875 17.02 0 1 4 4 1.0]
 ['Datsun 710' 22.8 4 108.0 93 3.85 2.32 18.61 1 1 4 1 1.0]
 ['Hornet 4 Drive' 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1 1.0]
 ['Hornet Sportabout' 18.7 8 360.0 175 3.15 3.44 17.02 0 0 3 2 1.0]
 ['Valiant' 18.1 6 225.0 105 2.76 3.46 20.22 1 0 3 1 1.0]
 ['Duster 360' 14.3 8 360.0 245 3.21 3.57 15.84 0 0 3 4 1.0]
 ['Merc 240D' 24.4 4 146.7 62 3.69 3.19 20.0 1 0 4 2 1.0]
 ['Merc 230' 22.8 4 140.8 95 3.92 3.15 22.9 1 0 4 2 1.0]
 ['Merc 280' 19.2 6 167.6 123 3.92 3.44 18.3 1 0 4 4 1.0]
 ['Merc 280C' 17.8 6 167.6 123 3.92 3.44 18.9 1 0 4 4 1.0]
 ['Merc 450SE' 16.4 8 275.8 180 3.07 4.07 17.4 0 0 3 3 1.0]
 ['Merc 450SL' 17.3 8 275.8 180 3.07 3.73 17.6 0 0 3 3 1.0]
 ['Merc 450SLC' 15.2 8 275.8 180 3.07 3.78 18.0 0 0 3 3 1.0]
 ['Cadillac Fleetwood' 10.4 8 472.0 205 2.93 5.25 17.98 0 0 3 4 1.0]
 ['Lincoln Continental' 10.4 8 460.0 215 3.0 5.4239999999999995 17.82 0
0
  3 4 1.0]
 ['Chrysler Imperial' 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4 1.0]
 ['Fiat 128' 32.4 4 78.7 66 4.08 2.2 19.47 1 1 4 1 1.0]
 ['Honda Civic' 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2 1.0]
 ['Toyota Corolla' 33.9 4 71.1 65 4.22 1.835 19.9 1 1 4 1 1.0]
 ['Toyota Corona' 21.5 4 120.1 97 3.7 2.465 20.01 1 0 3 1 1.0]
 ['Dodge Challenger' 15.5 8 318.0 150 2.76 3.52 16.87 0 0 3 2 1.0]
 ['AMC Javelin' 15.2 8 304.0 150 3.15 3.435 17.3 0 0 3 2 1.0]
 ['Camaro Z28' 13.3 8 350.0 245 3.73 3.84 15.41 0 0 3 4 1.0]
 ['Pontiac Firebird' 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2 1.0]
 ['Fiat X1-9' 27.3 4 79.0 66 4.08 1.935 18.9 1 1 4 1 1.0]
 ['Porsche 914-2' 26.0 4 120.3 91 4.43 2.14 16.7 0 1 5 2 1.0]
 ['Lotus Europa' 30.4 4 95.1 113 3.77 1.5130000000000001 16.9 1 1 5 2
1.0]
 ['Ford Pantera L' 15.8 8 351.0 264 4.22 3.17 14.5 0 1 5 4 1.0]
 ['Ferrari Dino' 19.7 6 145.0 175 3.62 2.77 15.5 0 1 5 6 1.0]
 ['Maserati Bora' 15.0 8 301.0 335 3.54 3.57 14.6 0 1 5 8 1.0]
 ['Volvo 142E' 21.4 4 121.0 109 4.11 2.78 18.6 1 1 4 2 1.0]]
```

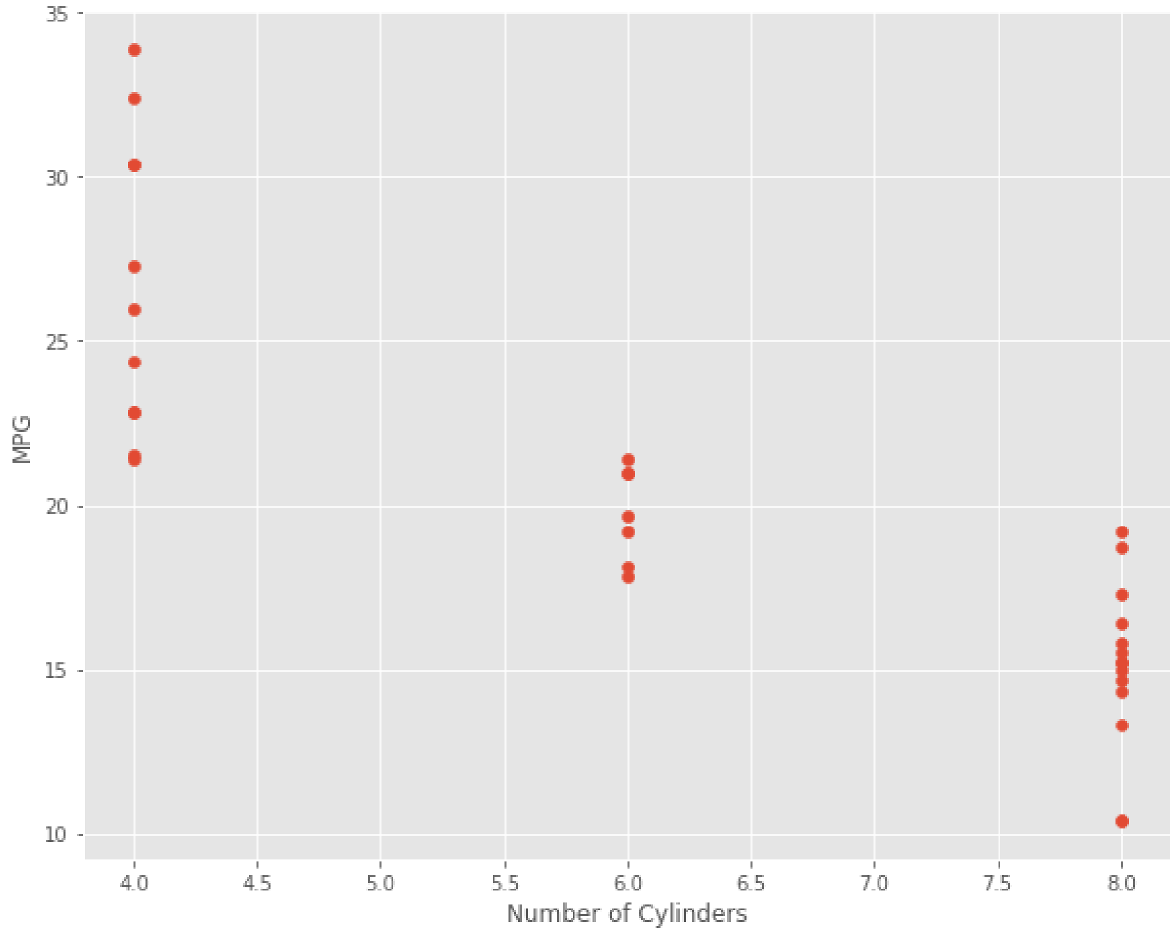
In [6]:
```
#Define x and y
x = data_mat[:,2:]
y = data_mat[:,1]

print('The x is:')
print(x)
print('')
print('The y is:')
print(y)
```

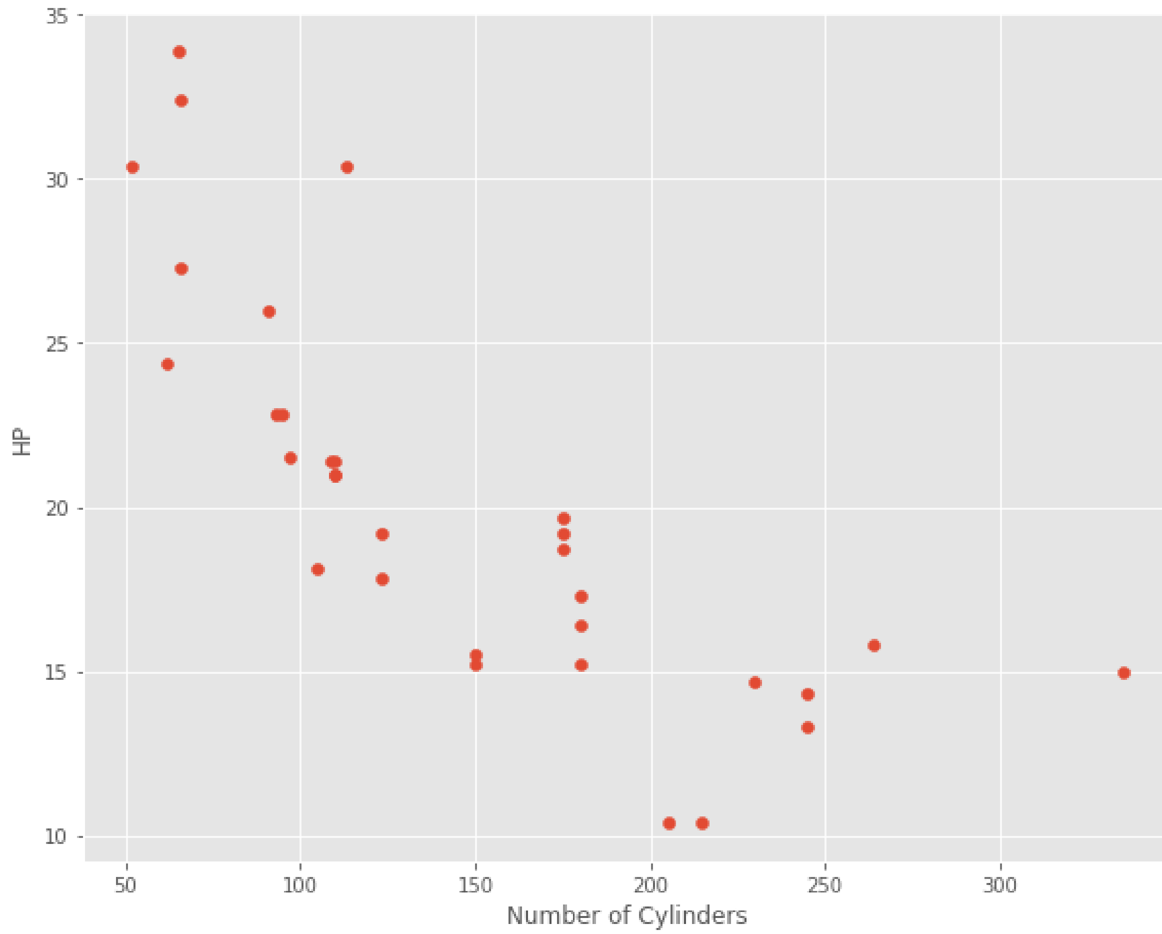
```
The x is:
[[6 160.0 110 3.9 2.62 16.46 0 1 4 4 1.0]
 [6 160.0 110 3.9 2.875 17.02 0 1 4 4 1.0]
 [4 108.0 93 3.85 2.32 18.61 1 1 4 1 1.0]
 [6 258.0 110 3.08 3.215 19.44 1 0 3 1 1.0]
 [8 360.0 175 3.15 3.44 17.02 0 0 3 2 1.0]
 [6 225.0 105 2.76 3.46 20.22 1 0 3 1 1.0]
 [8 360.0 245 3.21 3.57 15.84 0 0 3 4 1.0]
 [4 146.7 62 3.69 3.19 20.0 1 0 4 2 1.0]
 [4 140.8 95 3.92 3.15 22.9 1 0 4 2 1.0]
 [6 167.6 123 3.92 3.44 18.3 1 0 4 4 1.0]
 [6 167.6 123 3.92 3.44 18.9 1 0 4 4 1.0]
 [8 275.8 180 3.07 4.07 17.4 0 0 3 3 1.0]
 [8 275.8 180 3.07 3.73 17.6 0 0 3 3 1.0]
 [8 275.8 180 3.07 3.78 18.0 0 0 3 3 1.0]
 [8 472.0 205 2.93 5.25 17.98 0 0 3 4 1.0]
 [8 460.0 215 3.0 5.4239999999999995 17.82 0 0 3 4 1.0]
 [8 440.0 230 3.23 5.345 17.42 0 0 3 4 1.0]
 [4 78.7 66 4.08 2.2 19.47 1 1 4 1 1.0]
 [4 75.7 52 4.93 1.615 18.52 1 1 4 2 1.0]
 [4 71.1 65 4.22 1.835 19.9 1 1 4 1 1.0]
 [4 120.1 97 3.7 2.465 20.01 1 0 3 1 1.0]
 [8 318.0 150 2.76 3.52 16.87 0 0 3 2 1.0]
 [8 304.0 150 3.15 3.435 17.3 0 0 3 2 1.0]
 [8 350.0 245 3.73 3.84 15.41 0 0 3 4 1.0]
 [8 400.0 175 3.08 3.845 17.05 0 0 3 2 1.0]
 [4 79.0 66 4.08 1.935 18.9 1 1 4 1 1.0]
 [4 120.3 91 4.43 2.14 16.7 0 1 5 2 1.0]
 [4 95.1 113 3.77 1.5130000000000001 16.9 1 1 5 2 1.0]
 [8 351.0 264 4.22 3.17 14.5 0 1 5 4 1.0]
 [6 145.0 175 3.62 2.77 15.5 0 1 5 6 1.0]
 [8 301.0 335 3.54 3.57 14.6 0 1 5 8 1.0]
 [4 121.0 109 4.11 2.78 18.6 1 1 4 2 1.0]]

The y is:
[21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
 15.8 19.7 15.0 21.4]
```

In [7]:
```
#Visualize cyl vs mpg
plt.scatter(x[:,0], y)
plt.xlabel('Number of Cylinders')
plt.ylabel('MPG')
plt.show()
```



In [8]:
```
#Visualize hp vs mpg
plt.scatter(x[:,2], y)
plt.xlabel('Number of Cylinders')
plt.ylabel('HP')
plt.show()
```



In [12]:
```
#Find the number of features
_,n = np.shape(x)
print(n)
```

11

In [13]:
```
#Our matrix system will have n+1 entries, with the 1 addition correspo
nding to the intercept
A = np.zeros((n,n))
print(A)
```

```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

In [17]:

```
for i in range(0,n):
    for j in range(0,n):
        A[i,j] = np.sum(np.multiply(x[:,i],x[:,j]))

#Populating the last row of the matrix
for j in range(0,n):
    A[n-1,j] = np.sum(sum(x[:,j]))

print(A)
```

```
[[1.32400000e+03 5.18724000e+04 3.22040000e+04 6.91400000e+02
  6.79404000e+02 3.47556000e+03 6.40000000e+01 6.60000000e+01
  7.10000000e+02 6.04000000e+02 1.98000000e+02]
 [5.18724000e+04 2.17962747e+06 1.29136440e+06 2.50947960e+04
  2.70914888e+04 1.28801504e+05 1.85440000e+03 1.86590000e+03
  2.56503000e+04 2.32161000e+04 7.38310000e+03]
 [3.22040000e+04 1.29136440e+06 8.34278000e+05 1.63722800e+04
  1.64717440e+04 8.10921600e+04 1.27900000e+03 1.64900000e+03
  1.71120000e+04 1.57760000e+04 4.69400000e+03]
 [6.91400000e+02 2.50947960e+04 1.63722800e+04 4.22790700e+02
  3.58718960e+02 2.05691400e+03 5.40300000e+01 5.26500000e+01
  4.32950000e+02 3.21260000e+02 1.15090000e+02]
 [6.79404000e+02 2.70914888e+04 1.64717440e+04 3.58718960e+02
  3.60901070e+02 1.82809458e+03 3.65580000e+01 3.13430000e+01
  3.66582000e+02 3.10502000e+02 1.02952000e+02]
 [3.47556000e+03 1.28801504e+05 8.10921600e+04 2.05691400e+03
  1.82809458e+03 1.02934802e+04 2.70670000e+02 2.25680000e+02
  2.09746000e+03 1.54767000e+03 5.71160000e+02]
 [6.40000000e+01 1.85440000e+03 1.27900000e+03 5.40300000e+01
  3.65580000e+01 2.70670000e+02 1.40000000e+01 7.00000000e+00
  5.40000000e+01 2.50000000e+01 1.40000000e+01]
 [6.60000000e+01 1.86590000e+03 1.64900000e+03 5.26500000e+01
  3.13430000e+01 2.25680000e+02 7.00000000e+00 1.30000000e+01
  5.70000000e+01 3.80000000e+01 1.30000000e+01]
 [7.10000000e+02 2.56503000e+04 1.71120000e+04 4.32950000e+02
  3.66582000e+02 2.09746000e+03 5.40000000e+01 5.70000000e+01
  4.52000000e+02 3.42000000e+02 1.18000000e+02]
 [6.04000000e+02 2.32161000e+04 1.57760000e+04 3.21260000e+02
  3.10502000e+02 1.54767000e+03 2.50000000e+01 3.80000000e+01
  3.42000000e+02 3.34000000e+02 9.00000000e+01]
 [1.98000000e+02 7.38310000e+03 4.69400000e+03 1.15090000e+02
  1.02952000e+02 5.71160000e+02 1.40000000e+01 1.30000000e+01
  1.18000000e+02 9.00000000e+01 3.20000000e+01]]
```

In [19]:

```
#This vector will store the RHS of the matrix system
b = np.zeros((n,))

for i in range(0,n):
    b[i] = np.sum(x[:,i]*y)
#b[n] = np.sum(y)                #This term will hold sum(yi)

print(b)
```

```
[  3693.6    128705.08    84362.7      2380.277    1909.7528  11614.745
    343.8       317.1      2436.9      1641.9       642.9   ]
```

In [20]:
```
#Finding the inverse
A_inv = np.linalg.inv(A)
print(A_inv)
```

```
[[ 1.55487569e-01 -7.07183997e-04 -5.95395085e-04  6.70914657e-02
   3.18058726e-02  2.98592741e-02  1.00440083e-01  7.98098197e-02
   7.83313749e-02 -2.81337389e-02 -1.87424239e+00]
 [-7.07183997e-04  4.54030479e-05 -2.90503410e-05 -4.81865226e-04
  -3.70558863e-03  5.30481853e-04  5.35944692e-04  1.40983761e-04
  -2.98197687e-04  1.42126468e-03 -8.41481423e-04]
 [-5.95395085e-04 -2.90503410e-05  6.74689335e-05  4.40699441e-04
   1.41161399e-03  2.45554276e-04 -1.77604959e-03 -3.02647283e-04
  -4.14002931e-04 -1.35265847e-03 -3.78868814e-03]
 [ 6.70914657e-02 -4.81865226e-04  4.40699441e-04  3.80782830e-01
   7.42871900e-02  6.18475667e-03 -1.45767070e-02 -7.48973106e-02
  -2.60486727e-02 -3.98232345e-02 -1.84263491e+00]
 [ 3.18058726e-02 -3.70558863e-03  1.41161399e-03  7.42871900e-02
   5.10967881e-01 -9.99519610e-02  4.81275585e-02  5.23321257e-02
   7.30403152e-02 -1.55368805e-01  4.49345889e-01]
 [ 2.98592741e-02  5.30481853e-04  2.45554276e-04  6.18475667e-03
  -9.99519610e-02  7.60490849e-02 -7.99725516e-02  5.86241152e-02
   1.25790869e-02  2.31673742e-02 -1.50159397e+00]
 [ 1.00440083e-01  5.35944692e-04 -1.77604959e-03 -1.45767070e-02
   4.81275585e-02 -7.99725516e-02  6.30587107e-01  1.29246808e-01
  -1.94917069e-02  2.30188288e-02  5.19141666e-01]
 [ 7.98098197e-02  1.40983761e-04 -3.02647283e-04 -7.48973106e-02
   5.23321257e-02  5.86241152e-02  1.29246808e-01  6.02233193e-01
  -1.37190334e-01  1.50552284e-02 -1.26497276e+00]
 [ 7.83313749e-02 -2.98197687e-04 -4.14002931e-04 -2.60486727e-02
   7.30403152e-02  1.25790869e-02 -1.94917069e-02 -1.37190334e-01
   3.17478643e-01 -7.47704807e-02 -1.61711918e+00]
 [-2.81337389e-02  1.42126468e-03 -1.35265847e-03 -3.98232345e-02
  -1.55368805e-01  2.31673742e-02  2.30188288e-02  1.50552284e-02
  -7.47704807e-02  9.77897589e-02  2.58652641e-01]
 [-1.87424239e+00 -8.41481423e-04 -3.78868814e-03 -1.84263491e+00
   4.49345889e-01 -1.50159397e+00  5.19141666e-01 -1.26497276e+00
  -1.61711918e+00  2.58652641e-01  4.98835322e+01]]
```

In [21]:
```
#Computing the slopes corresponding to each variable
theta = np.matmul(A_inv,b)
print(theta)
```

```
[-0.11144048  0.01333524 -0.02148212  0.78711097 -3.71530393  0.8210407
5
  0.31776281  2.52022689  0.65541302 -0.19941925 12.30337416]
```

In [26]:
```
#Finding the error and displaying the difference between original and
 estimated
error = 0.0
r, c = np.shape(x)
for i in range(0,r):
    i_est = np.sum(x[i,:]*theta)
    error += (i_est - y[i])**2 #(Estimated - original)^2
    print('Original:\t', y[i], '\tEstimated:\t',i_est)

error/=n
print("The mean squared error is:\t\t", error)
print("The root means squared error is:\t", error**(0.5))
```

```
Original:        21.0   Estimated:       22.599505761262435
Original:        21.0   Estimated:       22.111886079356715
Original:        22.8   Estimated:       26.250644084799433
Original:        21.4   Estimated:       21.23740454667545
Original:        18.7   Estimated:       17.693434028696203
Original:        18.1   Estimated:       20.383039035671622
Original:        14.3   Estimated:       14.38625625277921
Original:        24.4   Estimated:       22.496011884889107
Original:        22.8   Estimated:       24.419089897563087
Original:        19.2   Estimated:       18.699029942339234
Original:        17.8   Estimated:       19.191654392144116
Original:        16.4   Estimated:       14.172162110442295
Original:        17.3   Estimated:       15.599573596008746
Original:        15.2   Estimated:       15.742224699462273
Original:        10.4   Estimated:       12.034013415125738
Original:        10.4   Estimated:       10.936437710853777
Original:        14.7   Estimated:       10.493629361832134
Original:        32.4   Estimated:       27.77290580777931
Original:        30.4   Estimated:       29.89673891131207
Original:        33.9   Estimated:       29.51236909573993
Original:        21.5   Estimated:       23.643103441605476
Original:        15.5   Estimated:       16.943053221175177
Original:        15.2   Estimated:       17.732181497828407
Original:        13.3   Estimated:       13.30602197619969
Original:        19.2   Estimated:       16.691678988690626
Original:        27.3   Estimated:       28.29346869344556
Original:        26.0   Estimated:       26.15295396086912
Original:        30.4   Estimated:       27.63627258279828
Original:        15.8   Estimated:       18.870040802863176
Original:        19.7   Estimated:       19.693828154474943
Original:        15.0   Estimated:       13.941118382058537
Original:        21.4   Estimated:       24.368267683244035
The mean squared error is:               13.40858454696824
The root means squared error is:         3.661773415569052
```