<u>Name</u> – Deepesh Kumar Singh
<u>CSULB ID</u> – 016149963
<u>Assignment Partner</u> – Sumeet Sunil Gadkari

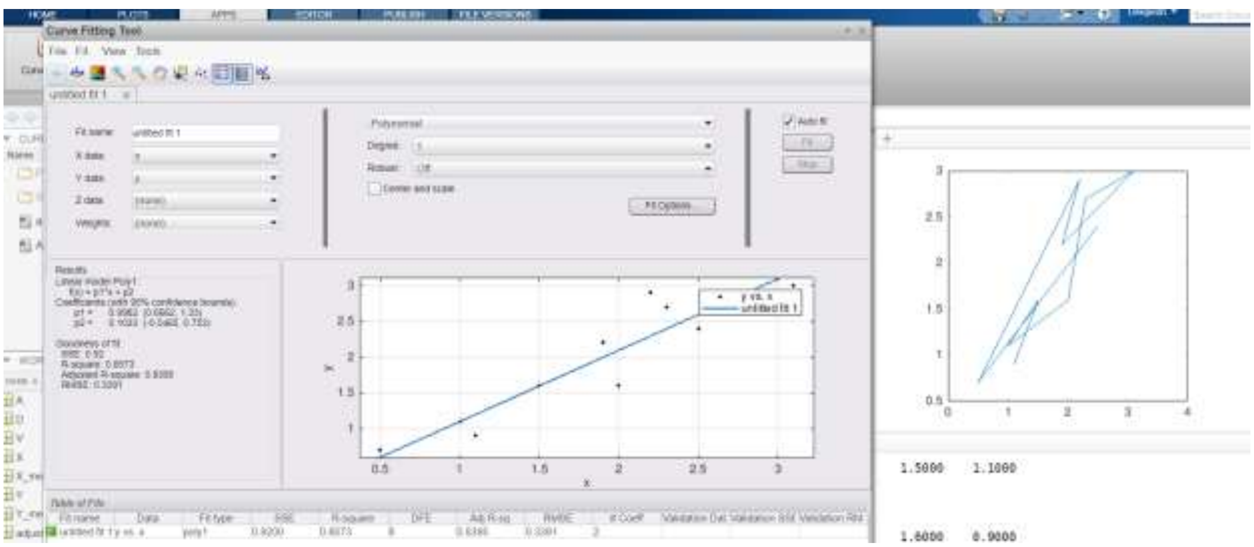
Assignment 1

1. Given is the following dataset (the training data):

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

2. Plot the graph y(x).

```
x = [ 2.5 0.5 2.2 1.9 3.1 2.3 2 1 1.5 1.1]
y = [2.4 0.7 2.9 2.2 3.0 2.7 1.6 1.1 1.6 0.9]
plot(x,y)
```



3. Find the mean values of the both x, y.

```
x1 = mean(X)
y1 = mean(Y)
```

```
x1 = 1.8100
y1 = 1.9100
```

Name – Deepesh Kumar Singh
CSULB ID – 016149963
Assignment Partner – Sumeet Sunil Gadkari

4. Calculate the covariance (2x2) matrix.

```
x = [ 2.5 0.5 2.2 1.9 3.1 2.3 2 1 1.5 1.1];
y = [2.4 0.7 2.9 2.2 3.0 2.7 1.6 1.1 1.6 0.9];
x1 = x - mean(X);
y1 = y - mean(Y);
adjustedMatrix = [x1;y1]
covMatrix = cov(x1,y1)

% We subtract mean so that the data has zero mean
```

```
adjustedMatrix =

    0.6900   -1.3100    0.3900    0.0900    1.2900    0.4900    0.1900   -
0.8100   -0.3100   -0.7100
    0.4900   -1.2100    0.9900    0.2900    1.0900    0.7900   -0.3100   -
0.8100   -0.3100   -1.0100


covMatrix =

    0.6166    0.6154
    0.6154    0.7166
```

5. Find the eigenvalues and eigenvectors of the covariance matrix.

```
x = [ 2.5 0.5 2.2 1.9 3.1 2.3 2 1 1.5 1.1];
y = [2.4 0.7 2.9 2.2 3.0 2.7 1.6 1.1 1.6 0.9];
x1 = x - mean(X);
y1 = y - mean(Y);
adjustedMatrix = [x1;y1]
covMatrix = cov(x1,y1)
[V,D] = eig(covMatrix)

% Eigen vectors of covariance matrix gives us most significant dimensions
containing maximum variance. We order eigen vectors in the decreasing order of
their eigen values.
```

```
V =

   -0.7352    0.6779
    0.6779    0.7352
```

Name – Deepesh Kumar Singh
CSULB ID – 016149963
Assignment Partner – Sumeet Sunil Gadkari

```
D =

    0.0491         0
         0    1.2840
```

6. Compare the vectors to see if there is a vector that can be identified as the principal component.

Below can identified as the principal component since it has maximum eigen value

```
0.6779    0.7352
```

7. Create a learning (regression) model utilizing the principal component.

```
x = [ 2.5 0.5 2.2 1.9 3.1 2.3 2 1 1.5 1.1];
y = [2.4 0.7 2.9 2.2 3.0 2.7 1.6 1.1 1.6 0.9];
x1 = x - mean(X);
y1 = y - mean(Y);
adjustedMatrix = [x1;y1]
covMatrix = cov(x1,y1)
[V,D] = eig(covMatrix)
vec = [V(2:2,:)]
matrixM=V*adjustedMatrix
finalX = matrixM(2:2,:)
finalY = matrixM(1:1,:)
plot(finalX,finalY)
originalDataset = V*matrixM
xOriginal = originalDataset(1:1,:) + mean(X)
yOriginal = originalDataset(2:2,:) + mean(Y)
invVec = transpose(vec)
finalDataset = invVec*finalX
xNew = finalDataset(1:1,:) + mean(X)
yNew = finalDataset(2:2,:) + mean(Y)
```

```
vec =  0.6779    0.7352


matrixM =

   -0.1751    0.1429    0.3844    0.1304   -0.2095    0.1753   -0.3498    0.0464    0.0178   -
0.1627
    0.8280   -1.7776    0.9922    0.2742    1.6758    0.9129   -0.0991   -1.1446   -0.4380   -
1.2238

finalX =

    0.8280   -1.7776    0.9922    0.2742    1.6758    0.9129   -0.0991   -1.1446   -0.4380   -
1.2238

finalY =
```

Name – Deepesh Kumar Singh
CSULB ID – 016149963
Assignment Partner – Sumeet Sunil Gadkari

```
   -0.1751    0.1429    0.3844    0.1304   -0.2095    0.1753   -0.3498    0.0464    0.0178   -
0.1627

originalDataset =

    0.6900   -1.3100    0.3900    0.0900    1.2900    0.4900    0.1900   -0.8100   -0.3100   -
0.7100
    0.4900   -1.2100    0.9900    0.2900    1.0900    0.7900   -0.3100   -0.8100   -0.3100   -
1.0100

xOriginal =

    2.5000    0.5000    2.2000    1.9000    3.1000    2.3000    2.0000    1.0000    1.5000
1.1000

yOriginal =

    2.4000    0.7000    2.9000    2.2000    3.0000    2.7000    1.6000    1.1000    1.6000
0.9000


invVec =

    0.6779
    0.7352

finalDataset =

    0.5613   -1.2050    0.6726    0.1859    1.1360    0.6189   -0.0672   -0.7759   -0.2969   -
0.8296
    0.6087   -1.3068    0.7294    0.2016    1.2320    0.6712   -0.0729   -0.8415   -0.3220   -
0.8997

xNew =

    2.3713    0.6050    2.4826    1.9959    2.9460    2.4289    1.7428    1.0341    1.5131
0.9804

yNew =

    2.5187    0.6032    2.6394    2.1116    3.1420    2.5812    1.8371    1.0685    1.5880
1.0103
```

Linear Regression model Poly1:

f(x) = p1*x + p2

Coefficients (with 95% confidence bounds):

p1 = 1.085 (1.085, 1.085)

p2 = -0.05301 (-0.05301, -0.05301)

Goodness of fit:

SSE: 7.654e-30

R-square: 1

Adjusted R-square: 1

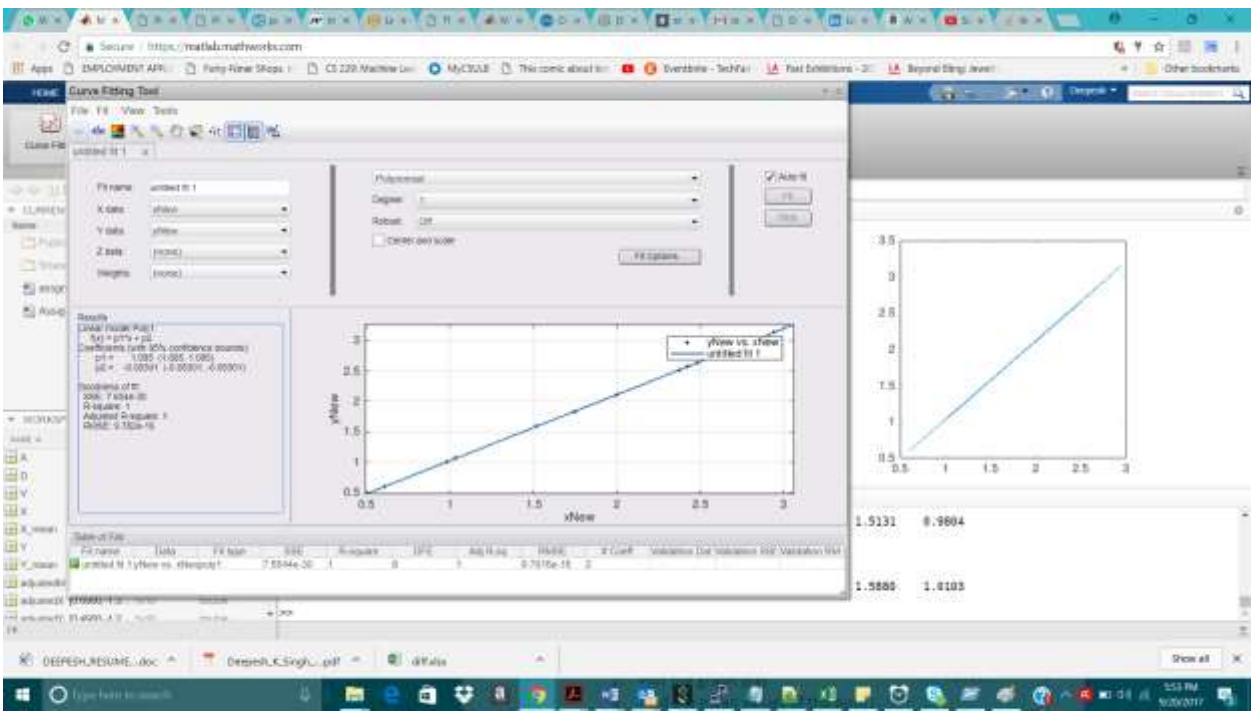
RMSE: 9.782e-16

% R-square indicates how close our model fits the points. R-square with value 1 is considered as good fit. We also got very small RMSE indicating accurate model.

<u>Name</u> – Deepesh Kumar Singh
<u>CSULB ID</u> – 016149963
<u>Assignment Partner</u> – Sumeet Sunil Gadkari

xNew	yNew
2.3712590	2.5187060
0.6050256	0.6031609
2.4825843	2.6394424
1.9958799	2.1115936
2.9459812	3.1420134
2.4288639	2.5811807
1.7428163	1.8371369
1.0341250	1.0685350
1.5130602	1.5879578
0.9804046	1.0102732

8. Plot the graph $y = f(x)$ representing this new model.



Name – Deepesh Kumar Singh
CSULB ID – 016149963
Assignment Partner – Sumeet Sunil Gadkari

9. Use the model to test it for the unused, so far, data. What output the trained model will suggest for *x* = 2.5 and 5?

f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
p1 = 1.085 (1.085, 1.085)
p2 = -0.05301 (-0.05301, -0.05301)

Putting 2.5 and 5 we get below values for f(x) –

2.65949

5.37199

Name – Deepesh Kumar Singh
CSULB ID – 016149963
Assignment Partner – Sumeet Sunil Gadkari

Complete MATLAB program to solve this assignment was –

```
x = [ 2.5 0.5 2.2 1.9 3.1 2.3 2 1 1.5 1.1];
y = [2.4 0.7 2.9 2.2 3.0 2.7 1.6 1.1 1.6 0.9];
x1 = x - mean(X);
y1 = y - mean(Y);
adjustedMatrix = [x1;y1]
covMatrix = cov(x1,y1)
[V,D] = eig(covMatrix)
vec = [V(2:2,:)]
matrixM=V*adjustedMatrix
finalX = matrixM(2:2,:)
finalY = matrixM(1:1,:)
plot(finalX,finalY)
originalDataset = V*matrixM
xOriginal = originalDataset(1:1,:) + mean(X)
yOriginal = originalDataset(2:2,:) + mean(Y)
invVec = transpose(vec)
finalDataset = invVec*finalX
xNew = finalDataset(1:1,:) + mean(X)
yNew = finalDataset(2:2,:) + mean(Y)
```