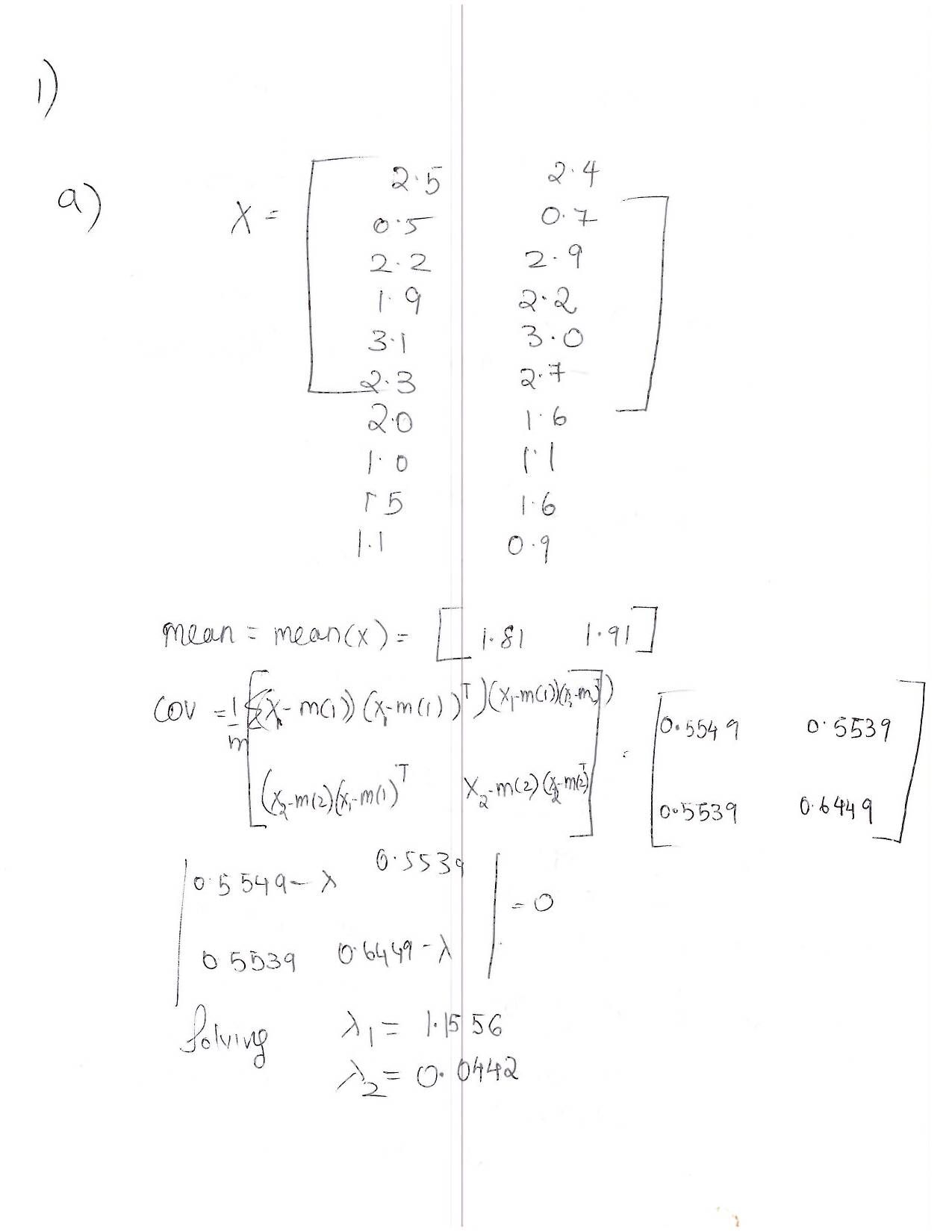
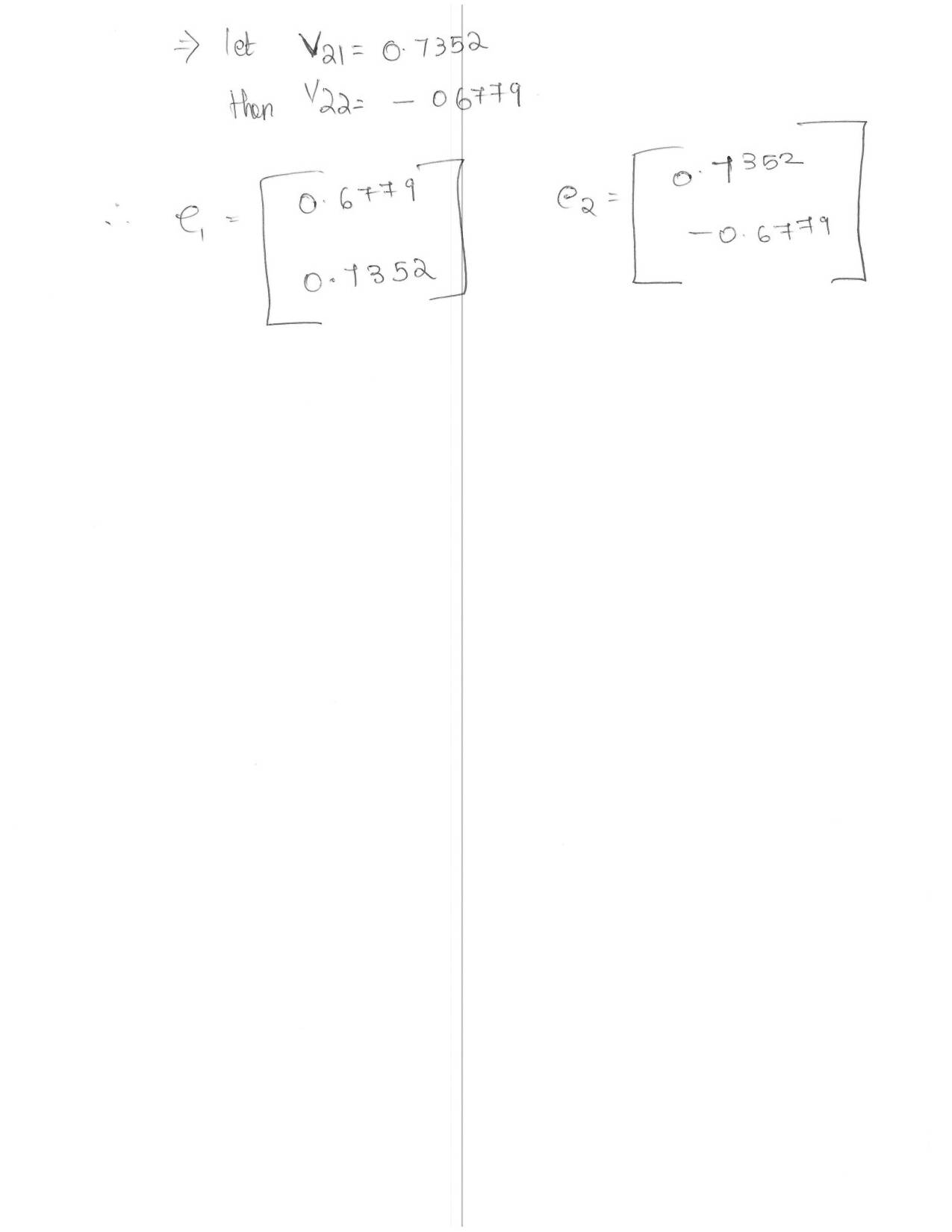
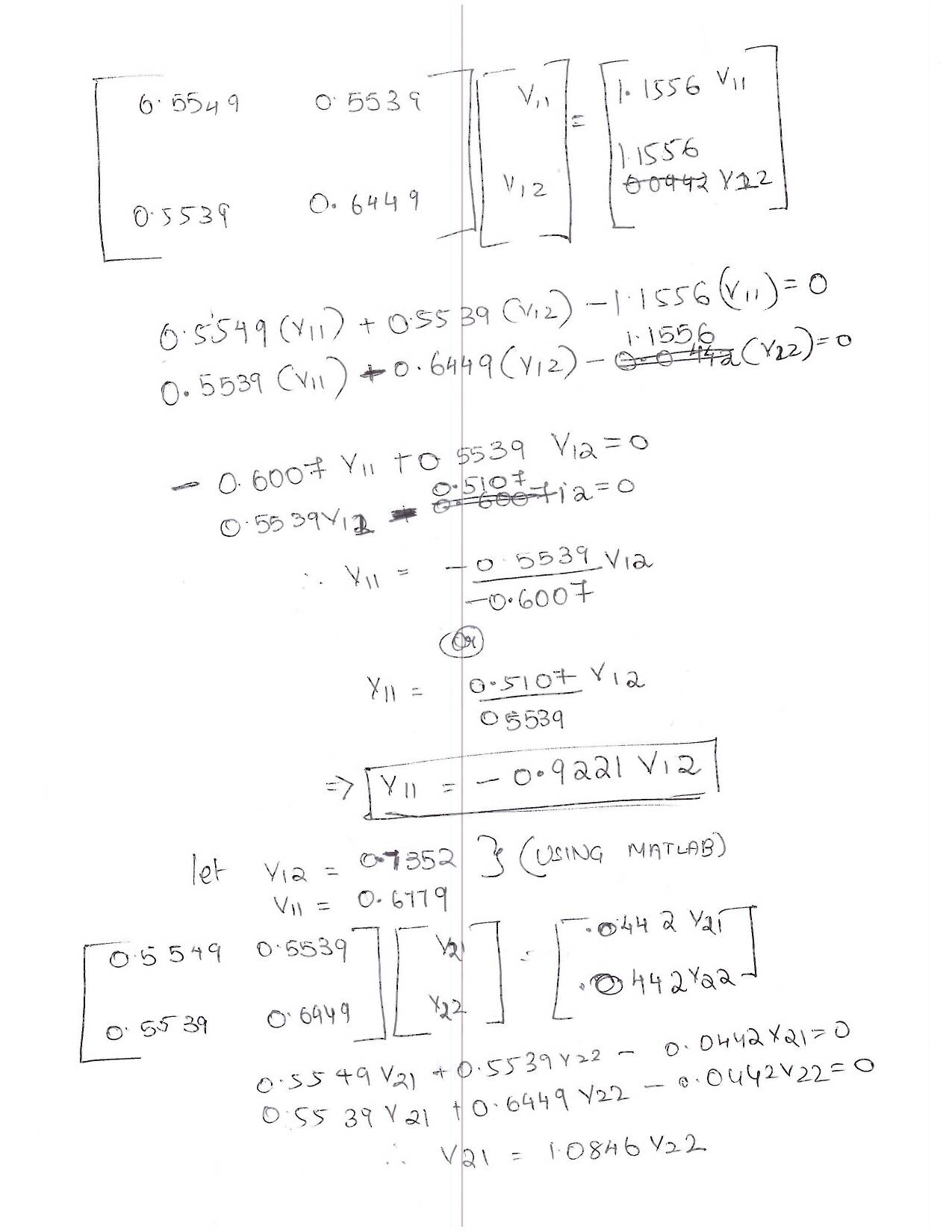
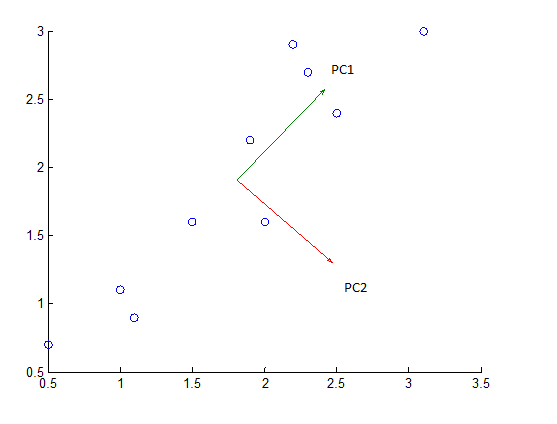
1)

A)







b)

|  |  |
| --- | --- |
| a1 | a2 |
| 0.82797 | 0.175115 |
| -1.77758 | -0.14286 |
| 0.992197 | -0.38437 |
| 0.27421 | -0.13042 |
| 1.675801 | 0.209498 |
| 0.912949 | -0.17528 |
| -0.09911 | 0.349825 |
| -1.14457 | -0.04642 |
| -0.43805 | -0.01776 |
| -1.22382 | 0.162675 |



c) New representation:

2.37125896400000 2.51870600832217

0.605025583745627 0.603160886338143

2.48258428755000 2.63944241997847

1.99587994658902 2.11159364495307

2.94598120291464 3.14201343391850

2.42886391124136 2.58118069424077

1.74281634877673 1.83713685698813

1.03412497746524 1.06853497544495

1.51306017656077 1.58795783010856

0.980404601156605 1.01027324970724



Range/distance= 3.4534

2)

a)

Let X=

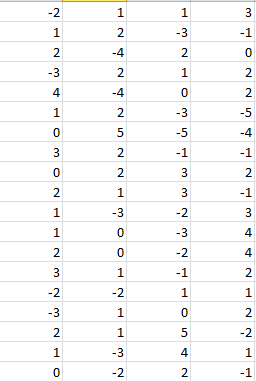
[-2 1 2 -3 4 1 0 3 0 2 1 1 2 3 -2 -3 2 1 0

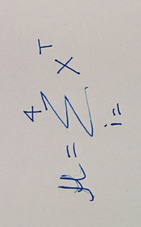
1 2 -4 2 -4 2 5 2 2 1 -3 0 0 1 -2 1 1 -3 -2

1 -3 2 1 0 -3 -5 -1 3 3 -2 -3 -2 -1 1 0 5 4 2

3 -1 0 2 2 -5 -4 -1 2 -1 3 4 4 2 1 2 -2 1 -1]

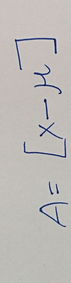
Rearranging the matrix (ie XT)





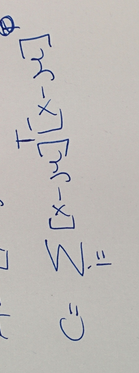
Mean= [0.75 -0.25 0 0.5 0.5 -1.25 -1 0.75 1.75 1.25 -0.25 0.5 1 1.25 -0.5 0 1.5 0.75 -0.25]

To Find the A matrix we subtract mean to each of the X values





Hence C=AT x A



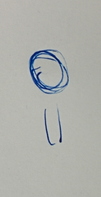
Hence the **SORTED** 4X4 inner product matrix in **descending order based on Eigen value is**

**[69.8750000000000 -18.8750000000000 -26.3750000000000 -24.6250000000000**

**-18.8750000000000 121.375000000000 -53.1250000000000 -49.3750000000000**

**-26.3750000000000 -53.1250000000000 98.3750000000000 -18.8750000000000**

**-24.6250000000000 -49.3750000000000 -18.8750000000000 92.8750000000000]**

After sorting the minimum squared error representations of these samples in 3D space is nothing but the eigen values by multiplying the new einvectors with the x data-the mean.

|69.875000**-ƛ** -18.875000 -26.37500 -24.6250

-18.875000 121.37500**-ƛ** -53.12500 -49.3750

-26.375000 -53.125000 98.37500-**ƛ** -18.8750

-24.625000 -49.37500 -18.8750 92.8750-**ƛ** |

Solving Eigen vectors and Eigen values:

**Eigen Vectors**

[0.500000000000000 0.862499590592227 -0.0412458686483948 -0.0662814796732877

0.500000000000000 -0.347332077693905 0.0682250237569957 -0.790383308236069

0.500000000000000 -0.220461654810414 0.691237386257788 0.472850435759295

0.500000000000000 -0.294705858087909 -0.718216541366389 0.383814352150061]

**Eigen Values**

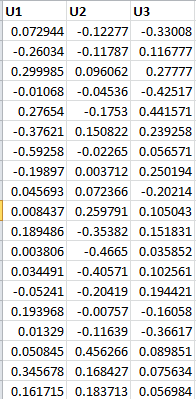
0.0000

92.6318

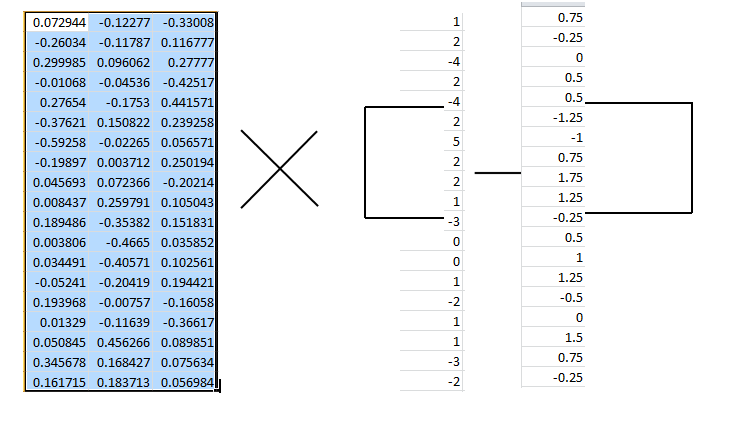
114.3171

175.5512

Now=A\*sorted\_evec; and normalizing the vectors gives us the answer



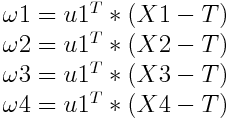
b) Multiplying the following vectors ie(UT \*X2-T) we get the vector of weights needed

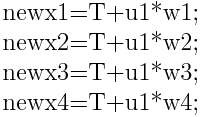


Hence the weights needed are

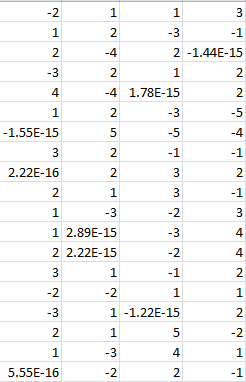
**X2= mean+-10.4722412661684(e1) + 0.729456174057581(e2) + -3.34291138878802(e3)**

c) Mean square error is calculated by reconstructing the data similar to as shown above, finding the difference between the reconstructed data and the new data, squaring it and fining the square root of the sum of the errors. Finally we can divide the errors by the number of dimensions





Therefore the newx



Hence the mean square errors for each sample are:

**4.112e-31, 2.732e-30, 9.0952e-31, 3.8580e-30**

d) Using only 2-D

**3.6268 0.5881 0.2369 0.4234**

e) The Euclidian distance to each projected image are:

**12.0711 3.2728 16.1466 15.0901**

**The answer is the 2nd sample**

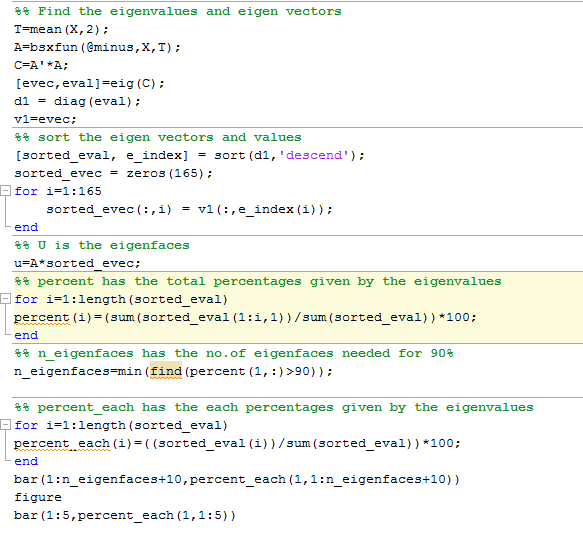
f) The new distances are

**16.6073 4.1098 18.1979 15.3917**

**The answer is still the 2nd sample**

**This does make intuitive sense because the Euclidian distance between the test vector and the 2nd sample increases. This is because using all the Eigen values gives a more accurate prediction of the mean square errors and the distance from the original values.**

3)



3)

1. **The new number of dimension= 35**



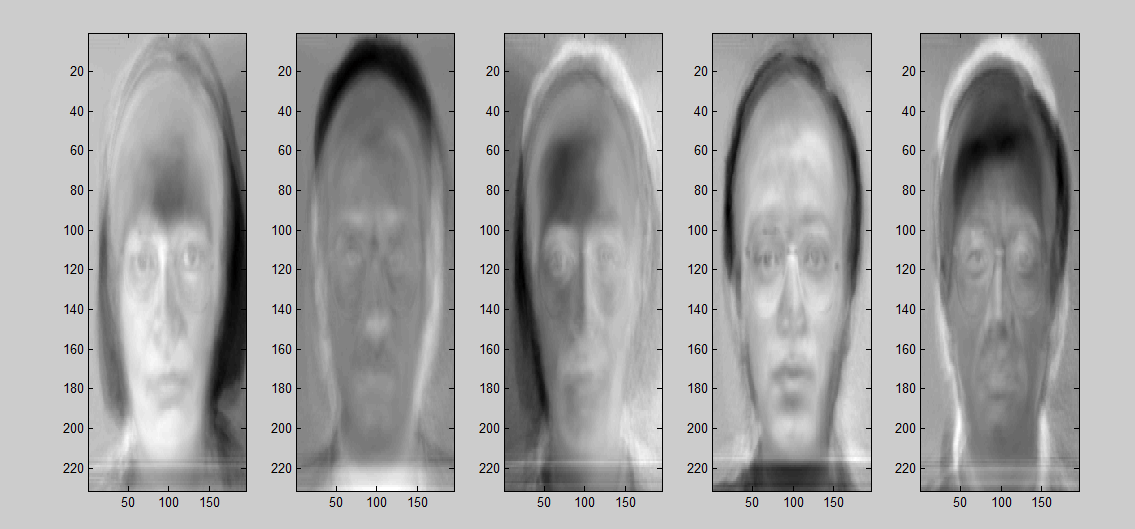
% variance explained

Each Eigen face



% variance explained

Each Eigen face



1. original image





1. accuracy=80%

|  |  |  |
| --- | --- | --- |
| Testing values | Actual values | R/W |
| 1 | 1 | TRUE |
| 4 | 1 | FALSE |
| 2 | 2 | TRUE |
| 2 | 2 | TRUE |
| 3 | 3 | TRUE |
| 7 | 3 | FALSE |
| 4 | 4 | TRUE |
| 4 | 4 | TRUE |
| 5 | 5 | TRUE |
| 5 | 5 | TRUE |
| 1 | 6 | FALSE |
| 6 | 6 | TRUE |
| 7 | 7 | TRUE |
| 7 | 7 | TRUE |
| 2 | 8 | FALSE |
| 8 | 8 | TRUE |
| 9 | 9 | TRUE |
| 9 | 9 | TRUE |
| 10 | 10 | TRUE |
| 10 | 10 | TRUE |
| 11 | 11 | TRUE |
| 11 | 11 | TRUE |
| 12 | 12 | TRUE |
| 5 | 12 | FALSE |
| 13 | 13 | TRUE |
| 13 | 13 | TRUE |
| 4 | 14 | FALSE |
| 14 | 14 | TRUE |
| 15 | 15 | TRUE |
| 15 | 15 | TRUE |

1. Accuracy increased to 83.33%

|  |  |  |
| --- | --- | --- |
| Testing values | Actual values | R/W |
| 1 | 1 | 1 |
| 4 | 1 | 0 |
| 2 | 2 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 1 |
| 15 | 3 | 0 |
| 4 | 4 | 1 |
| 4 | 4 | 1 |
| 5 | 5 | 1 |
| 5 | 5 | 1 |
| 6 | 6 | 1 |
| 6 | 6 | 1 |
| 7 | 7 | 1 |
| 7 | 7 | 1 |
| 2 | 8 | 0 |
| 8 | 8 | 1 |
| 9 | 9 | 1 |
| 9 | 9 | 1 |
| 10 | 10 | 1 |
| 10 | 10 | 1 |
| 11 | 11 | 1 |
| 11 | 11 | 1 |
| 12 | 12 | 1 |
| 5 | 12 | 0 |
| 13 | 13 | 1 |
| 13 | 13 | 1 |
| 4 | 14 | 0 |
| 14 | 14 | 1 |
| 15 | 15 | 1 |
| 15 | 15 | 1 |

Yes this is what I would expect as there is no information that is lost in the original dimension space. Hence the algorithm should perform better.