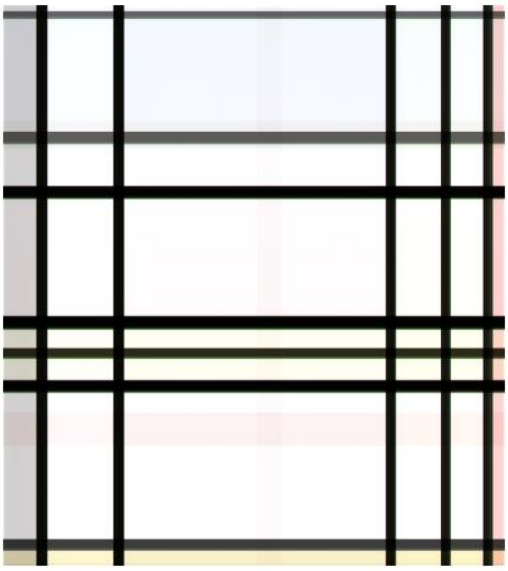
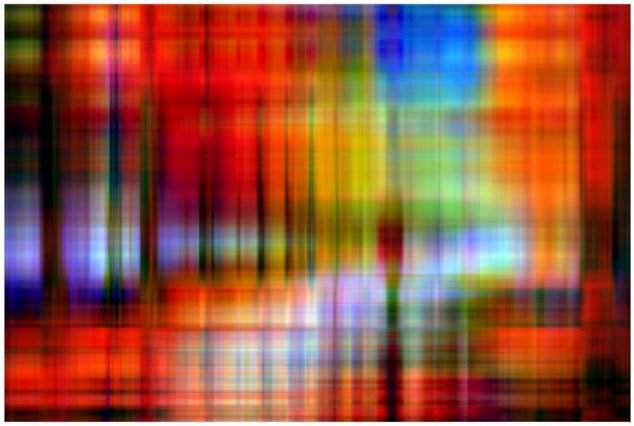
Exercise 2

SVD

1. Completed using matlab

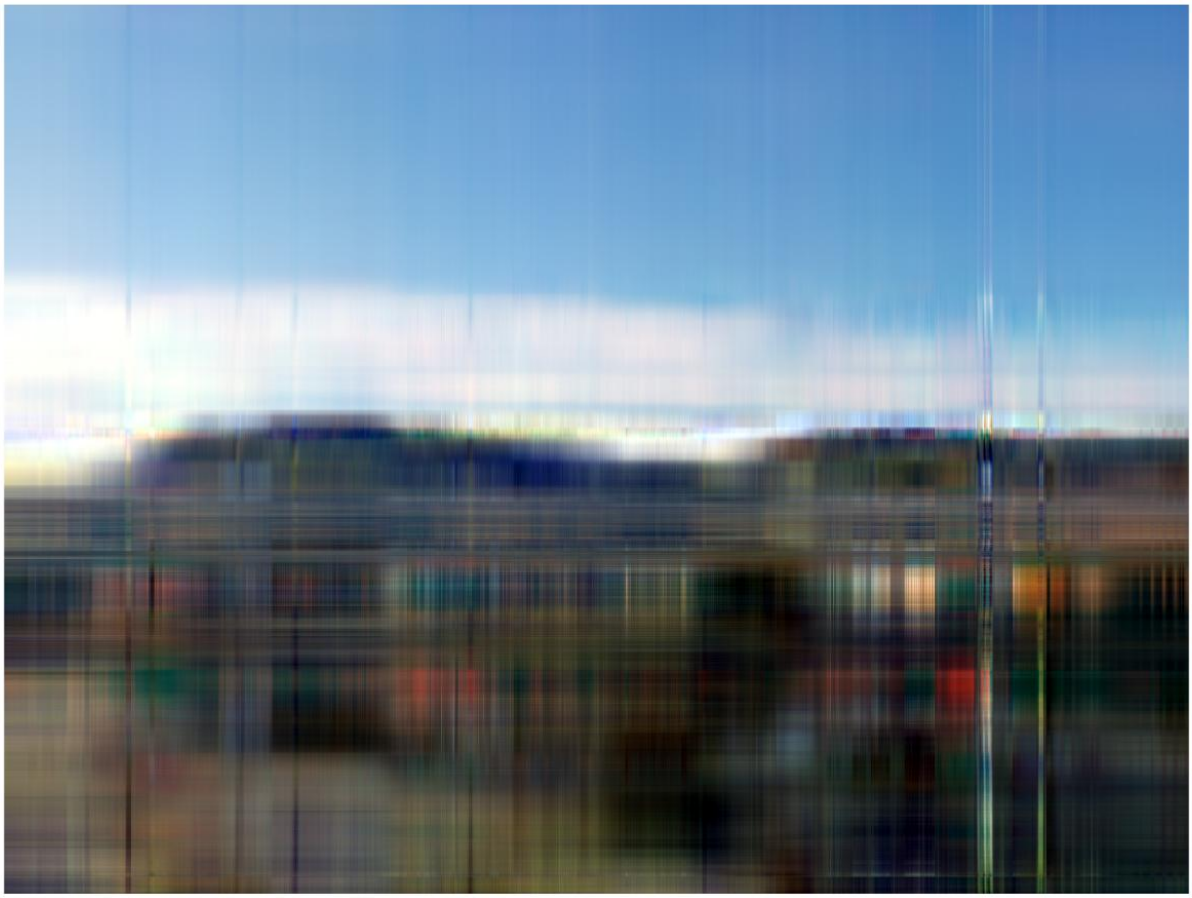
2. From observing the pictures after testing the SVD progam it appears that Walking In The Rain requires more singular components by far. Setting an arbitrary sigma threshold of e.g 100 blurs the image quite heavily while Composition10 mostly maintains shape and merely loses tone. Examples below





3. Below is an original photo I took on my way to school

Running the progra again with the same sigma threshold of 100 produces this:



Much like with Victor Figol's "Walking In The Rain", this image contains a lot of nuances in color and is as such quite distorted. Interestingly, the sky and clouds are relativly intact. This is probably due to the fact that the color data for these parts of the picture requires less data to reconstruct than the noisy lower part of the picture.

PCA

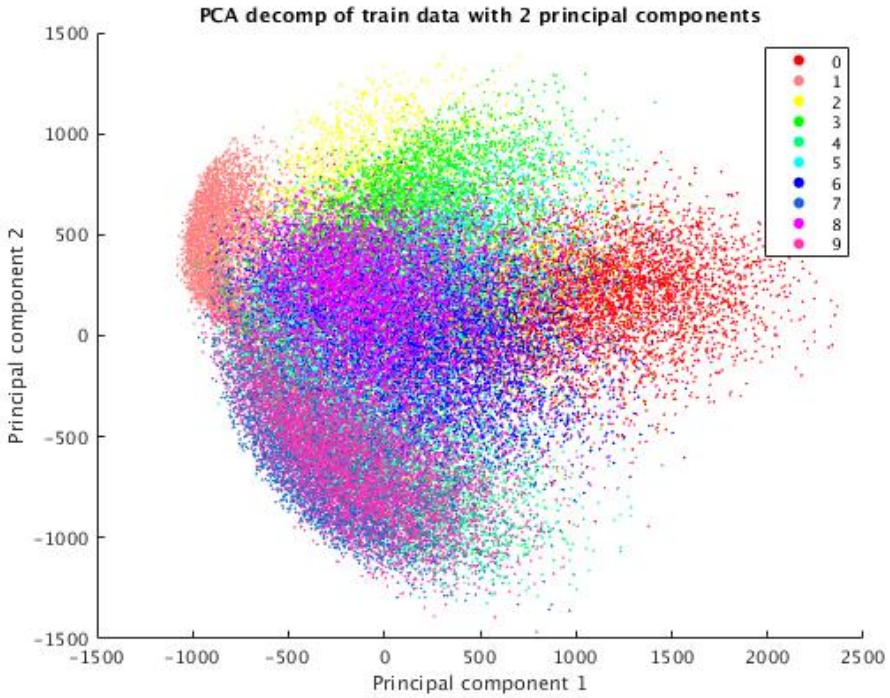
1. Solved in matlab.

2. I chose to utilize the pca function provided by matlab as it very handily computes scores, loadings and coefficients. While the terminology employed my mathworks is slightly different (coeff, score and

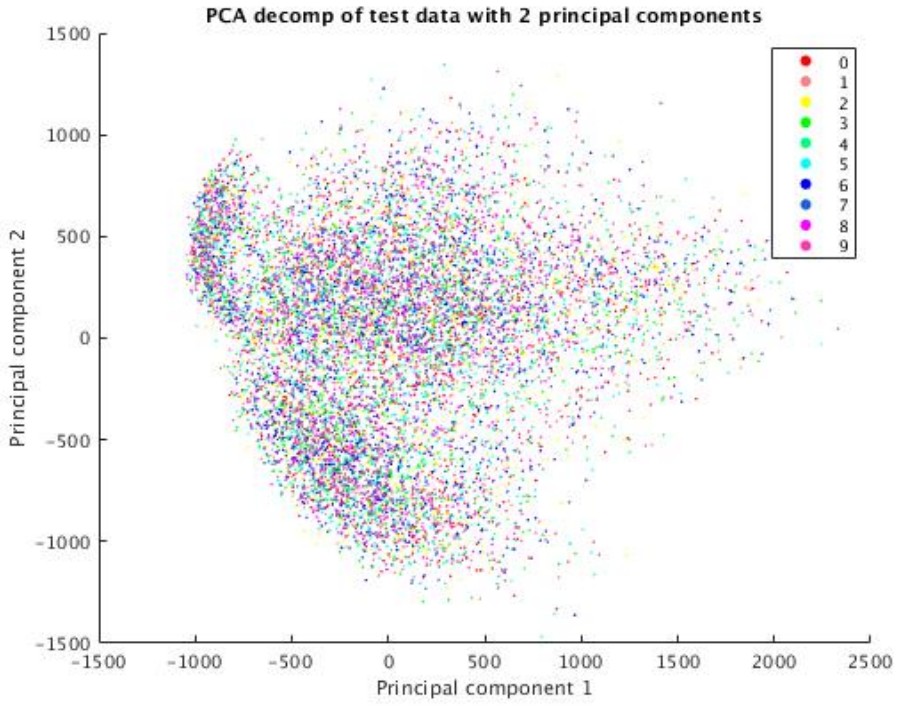
latent) they are of course the same thing. The loadings, aka coeff, provides an *nxn* matrix consisting of columns that are eigenvectors corresponding to any one principal componenet. The scores are given as an *n x m* matrix where *n* and *m* respectively are the original dimensions of the data. The columns of this matrix comprise the principal components. Lastly, the coefficients are the eigenvalues of the aforementioned eigenvectors.

According to the description of this function on the mathworks pages, all the data are by default centered by subtracting the means of each column. This is important to note because we must then remember to add back the mean of the dataset if we which to reconstruct the data (images for this assignment).

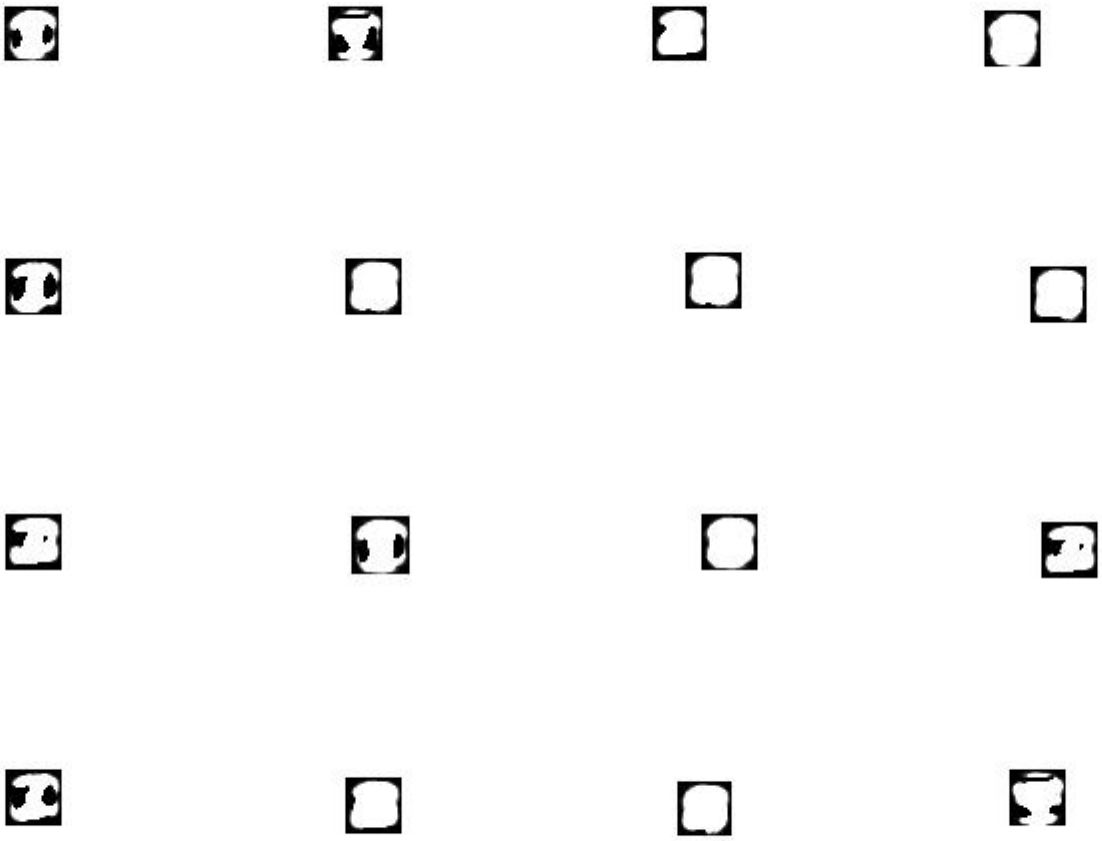
3. Solved in matlab, figure provided below



4. The main difference between train in test is that the principal comonents in test are shorter and thus provides less data. This is evident in the 2D embedding as the values are farther apart. Reconstructing the test image data is no problem: The train data contains a lot more information about what is essentially the same data set, therefore using values from train to reconstruct test results in a perfect match. Below is a figure displaying test data reconstructed from the train set.



5. Below
are 20 random images from the train data reconstructed using the first 2 principal components



Evidently, these are not good representations of the original images. It is however interesting to note recurring shapes: These likely represent the same digit with minor variances.

Selecting an appropriate amount of princiapl components boiled down to some trial and error on my part. The lowest I could go while also *occasionally* reconizing a digit was with 5 components. Example below:

Possible digit 0?

Increasing components up to 10 and 20 yielded somewhat better resul, but not consistently:

possible digit 5? 20 components

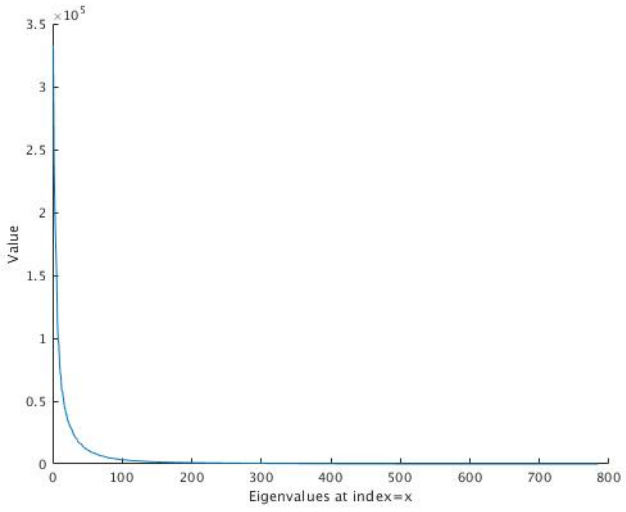
possible digit 8? 10 components

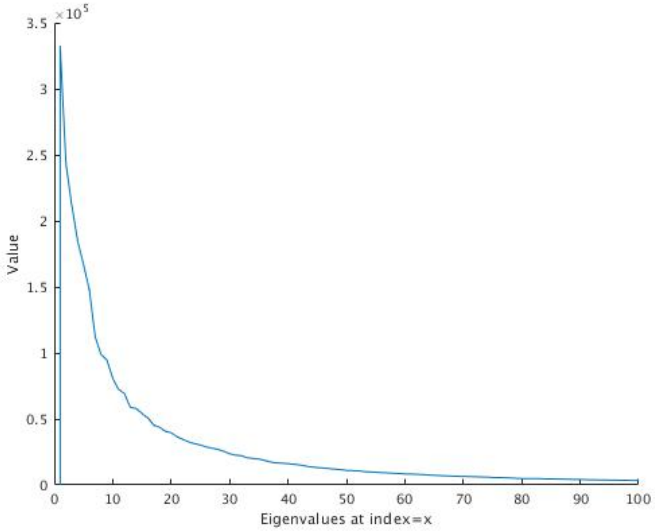
Further increasing to 100 components led to much better results, typically with discernable numbers surronded by noise. Sample below:

Quite likely digit 9, 100 components

Naturally, digits become the clearer the more components we use to reconstruct.

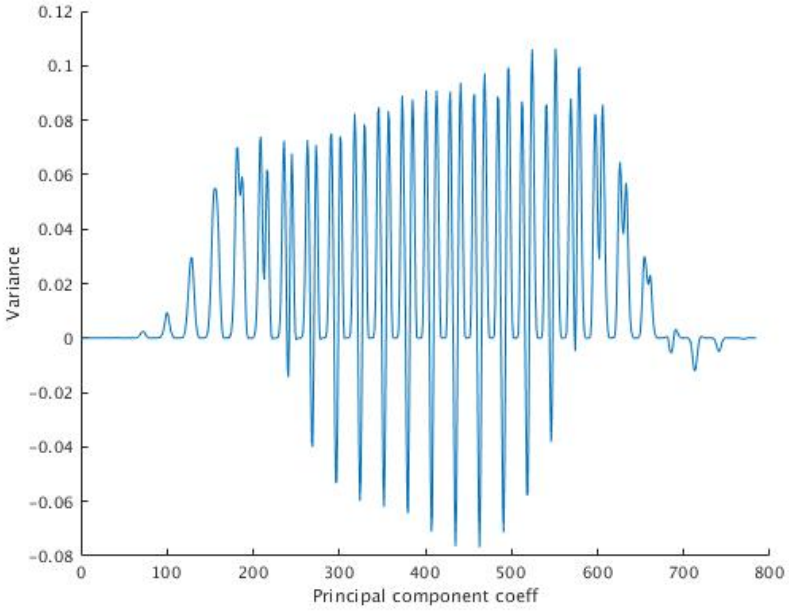
6. Below are two plots of the eigenvalues: The first one is without any scaling and the second one cuts the x axis at 100 and draws a line from x=1 to the point of the highest eigenvalue.





From this we can gather that the eigenvalues are sorted by the amount of “energy” they contain from highest to lowest. The total sum of all the eigenvalues is 3.4285e+06 and 90% of this is 3.0857e+06. To compute how many components that are required to get 90% we simply add them together from highest to lowest until we match or surpass. Using matlab I was able to determine that summing up the 87 first components yielded and energy of 3.060e+06. This correlates well to the previous task where examined numbers seemed to become clearer at around roughly 100 components.

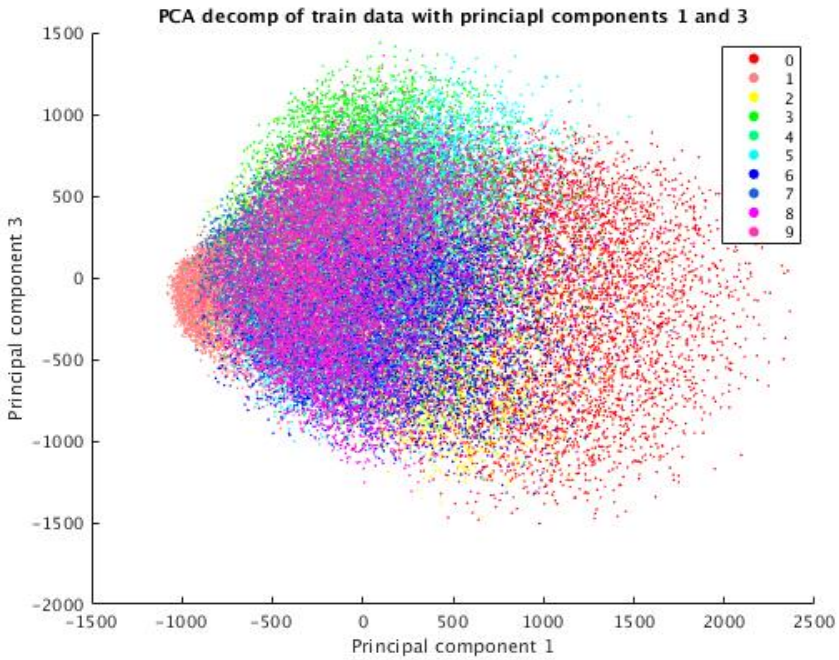
7. Below is a plot of the eigenvector corresponding to the biggest eigenvalue

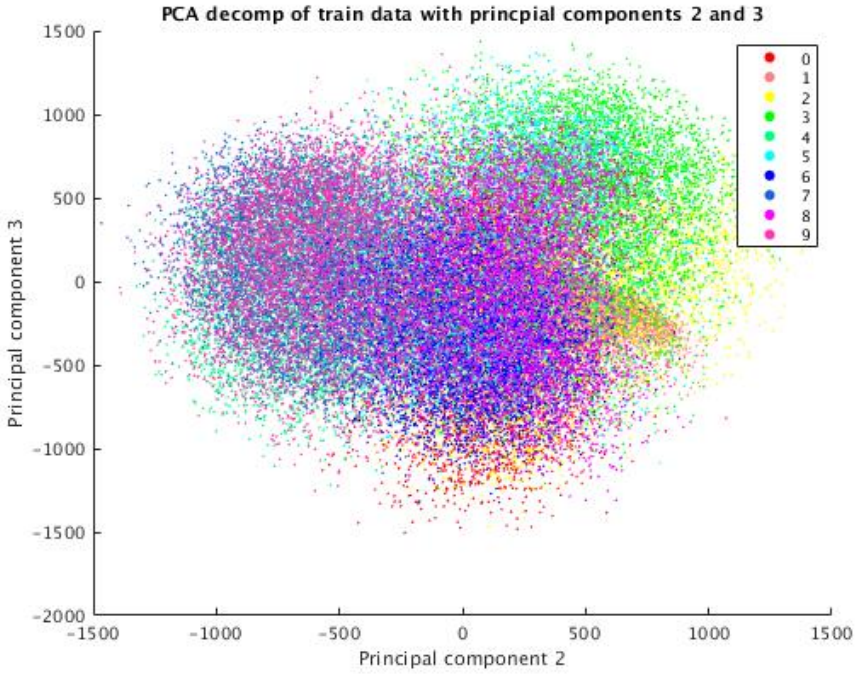


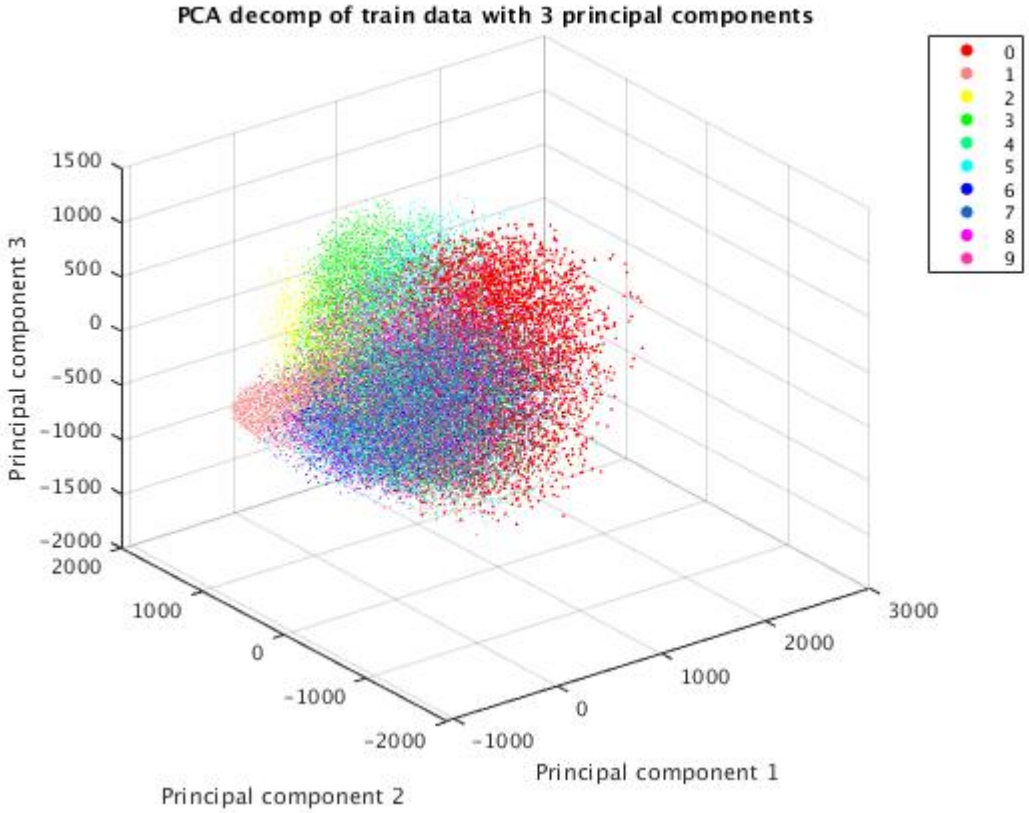
It describes the component variance of the first principal component

8. Below, in the listed order, are three plots:

    1. 2D-embedding of train data with principal components 1 and 3
    2. 2D-embedding of train data with principal componenst 2 and 3
    3. 3D scatter plot of train data with principal components 1, 2 and 3



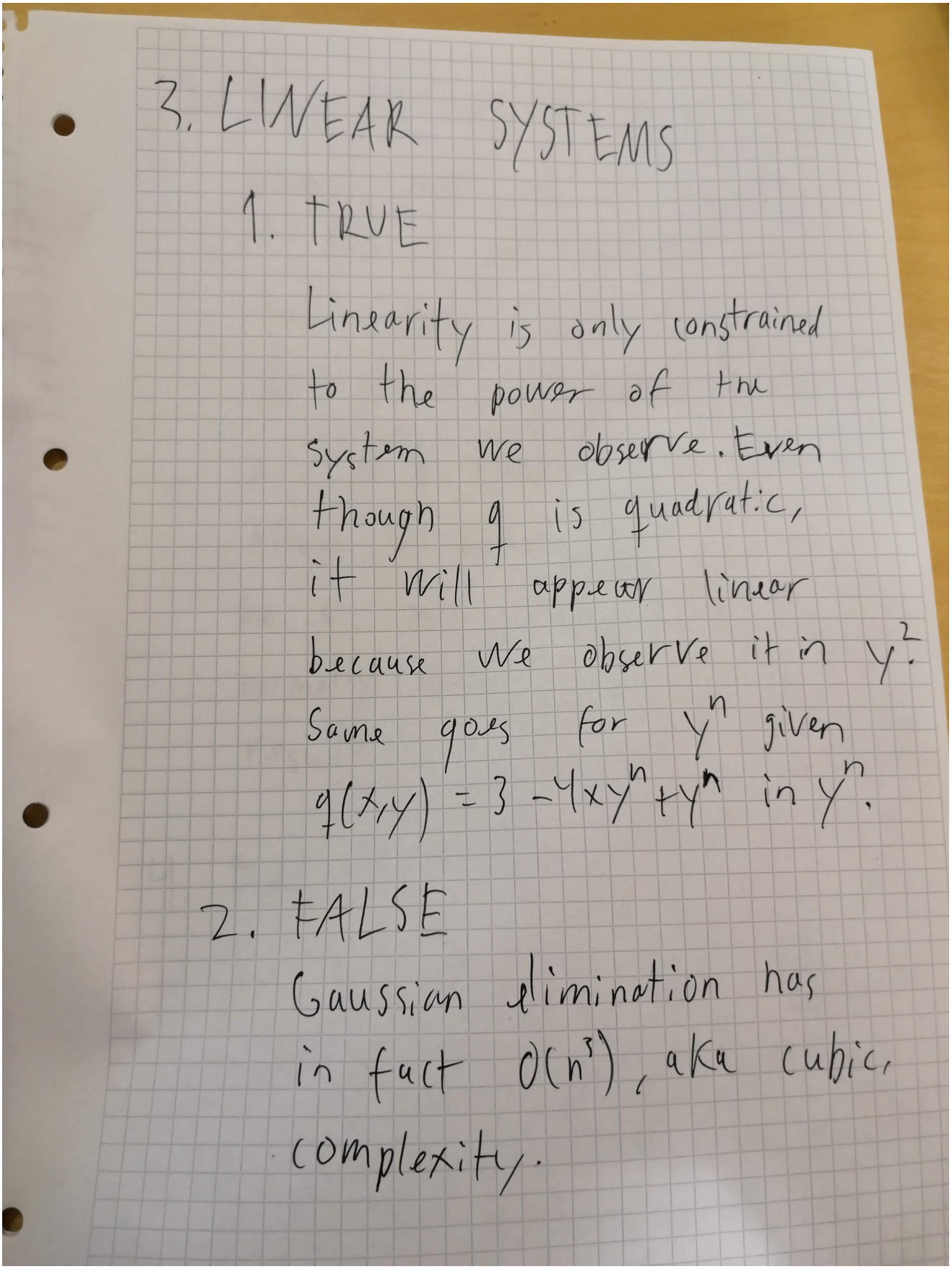




Comparing the two 2D plots against the 3D plot, one can spot clusters that only appear in one of their respecitve 2D plots now appearing together in the 3D plot. As an example, the cluster of corresponding values of 1 that formed in only the first figure can now clearly be seen in 3D figure.

3. Linear Systems

See images for solutions



3. True EXCEPT IF $b=0$

Given that A is singular it has a determinant $=0$, thus solving for A^{-1} gives $\frac{1}{0}$ which in turn invalidates $x = A^{-1}b$. However, if $b=0$ then all entries in x can be 0 and thus solving normally shows the expected result.

4. FALSE

It is in fact the opposite, only non-singular matrices may be inverted. Same point as before: $\det(A\text{-singular}) = 0$, $A\text{-singular}^{-1} \to \frac{1}{0}$ which is invalid