Math 521 HW3

Raj Mohanty
raj.mohanty@student.csulb.edu

2 Computing

2.1 Problem 1

Here's the problem statement:

$$f(x_m, t_\mu) = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{k}\sin[k(x_m - t_\mu)],$$

where $m = 1, \ldots, M$, with M dimension of the ambient space (size of the spatial grid), and $\mu = 1, \ldots, P$, with P the number of points in the ensemble. Let $x_m = \frac{(m-1)2\pi}{M}$ and $t_\mu = \frac{(\mu-1)2\pi}{P}$. Select an ensemble of masks $\{\mathbf{m}^{(\mu)}\}$, $\mu = 1, \ldots, P$, where 10% of the indices are selected to be zero for each mask. Each pattern in the incomplete ensemble may be written as

$$\tilde{\mathbf{x}}^{(\mu)} = \mathbf{m}^{(\mu)}.\mathbf{f}^{(\mu)},$$

where $\left(\mathbf{f}^{(\mu)}\right)_m = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{k}\sin[k(x_m - t_\mu)]$. Let $P = M = 64$ and $N = 3$.

a)

We create the original data as follows (code snippet 2.1.1):

```
M = 64;
P = 64;
N = 3;
x = (0:M-1)'/M;
t = (0:P-1)/P;
```

```
size(x)
```

```
ans =

    64     1
```

```
size(t)
```

```
ans =

     1    64
```

```
f = 0;
for k = 1:N
    f = f+1/N*1/k*sin(k*(x-t));
end
```

```
size(f)
```

```
ans =

    64    64
```

Code snippet 2.1.1 : Creation of original data

The 3-D plot of the original data is shown in figure 2.1.1

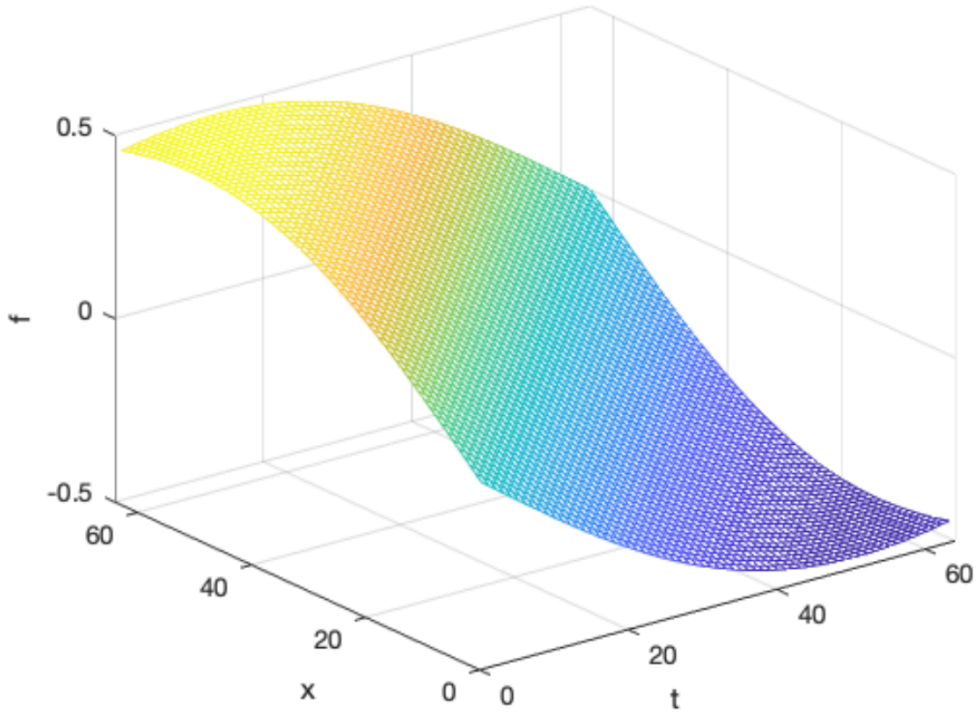


Figure 2.1.1: 3-D plot of the original data

The mask for 10% missing data is created as shown in code snippet 2.1.2

creating the mask of 10% missing

```
m = (floor(rand(M,P)*10)>0);
```

```
sum(m(:))/64^2
```

```
ans =

    0.9060
```

Code snippet 2.1.2: Mask for 10% missing data

Applying the mask to the original data creates the gappy data as shown in figure 2.1.2

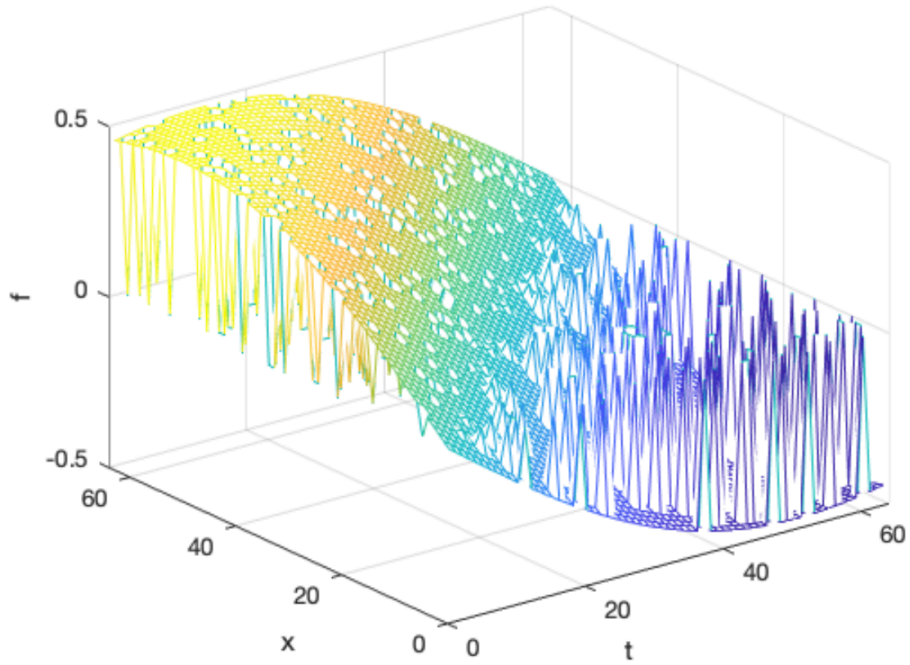


Figure 2.1.2: 3-D plot of the gappy data

We then estimate missing by taking ensemble average, bringing the missing values to non-zero values but still rough as shown in figure 2.1.3.

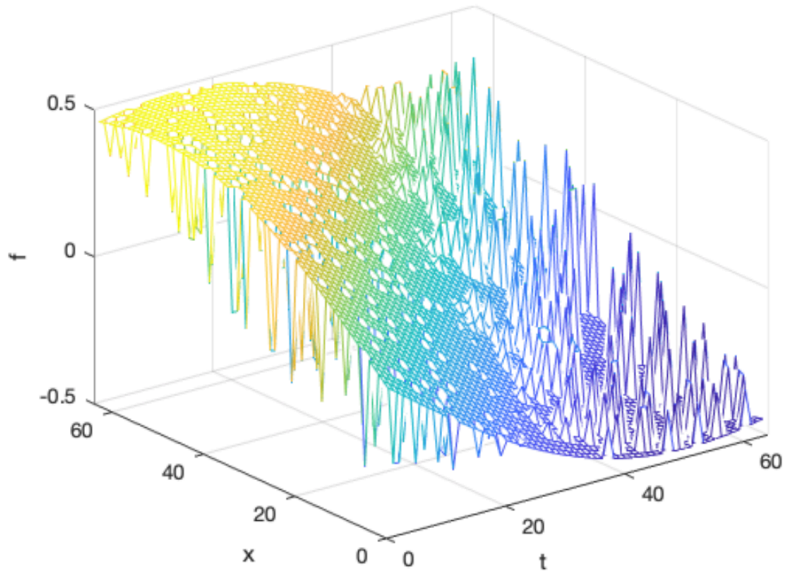


Figure 2.1.3: 3-D plot of the gappy data with missing values estimated as ensemble average

We then estimate the KL basis be solving the below eigenvalue problem

```
[v, e] = eig(r*r');
```

Then we estimate the coefficient a_i by doing the following

$$M_{ij} = (\phi^{(i)}, \phi^{(j)})_{\mathbf{m}}$$

$$f_i = (\tilde{\mathbf{x}}, \phi^{(i)})_{\mathbf{m}}.$$

$$\vec{a} = M^{-1} f$$

$$\mathbf{x}_D = \sum_{n=1}^{D} a_n \phi^{(n)}.$$

All of this logic is summarized in the following code snippet 2.1.3

```
r_old = r;
[v, e] = eig(r*r');
ev(it+1,:) = diag(e);

D = 2;
%KL basis
v = v(:,end-D:end);
for i = 1:P
    %Mij = (phi_i,phi_j)m
    A = (v.*m(:,i))'*(v.*m(:,i));
    %fi = (x_tilde,phi_i)m
    b = (v.*m(:,i))'*r(:,i);
    %a=M_inv f
    a = A\b;
    %XD = ai phi_i
    f(:,i) = v*a;
end
```

Code snippet 2.1.3: Steps to get estimate of missing data

We then run these steps until there is no significant improvement in recovered data over previous iteration.

b) Eigenvalues as a function of iterations
We then plot the first 8 eigenvalues as a function of the iteration

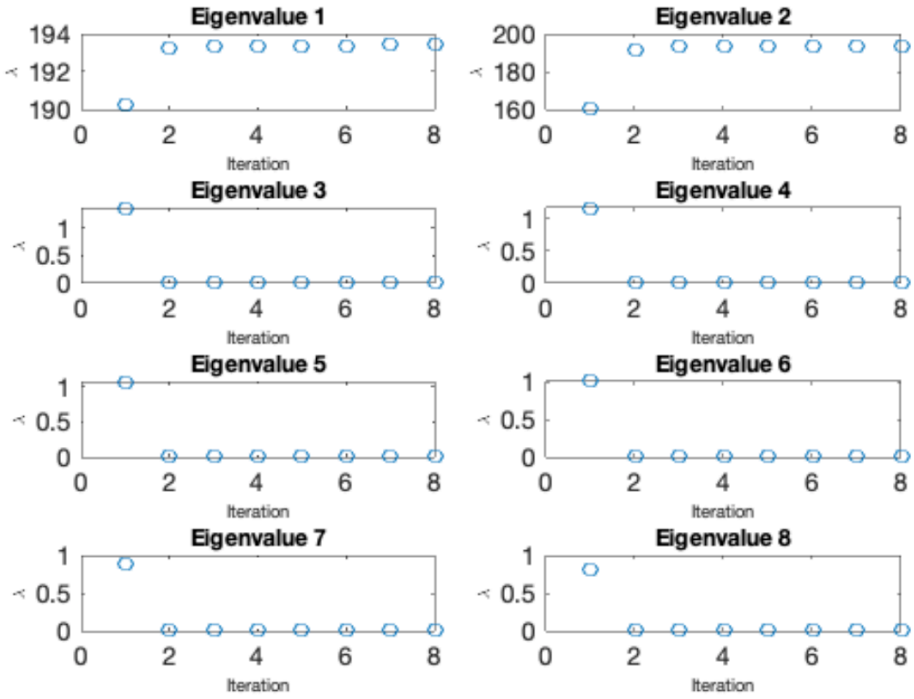


Figure 2.1.4: Eigenvalues as a function of iteration

We notice that eigenvalues 1 and 2 are the most significant across the iterations. After the algorithm has converged, we notice that we just need to keep first 3 eigenvalues.

c) Eigenfunctions of the 10 largest eigenvalues

We then plot the first 10 eigenfunctions of the largest eigenvalues after the algorithm has converged.

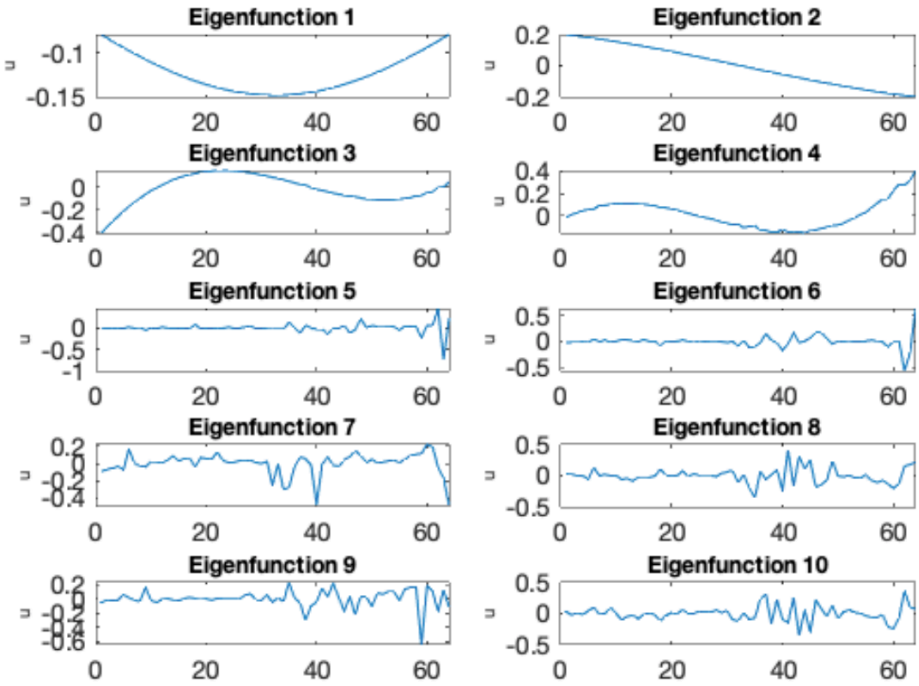


Figure 2.1.5: Eigenvalues as a function of iteration

We notice that the first 3-4 eigenfunctios are capturing some of the larger trends in the data and the later ones may be capturing short periods or noise.

d)
Finally we plot the following

Gappy data – Figure 2.1.6

Recovered Data – Figure 2.1.7

Original Data – Figure 2.1.8

Error between original and gappy data – 2.1.9

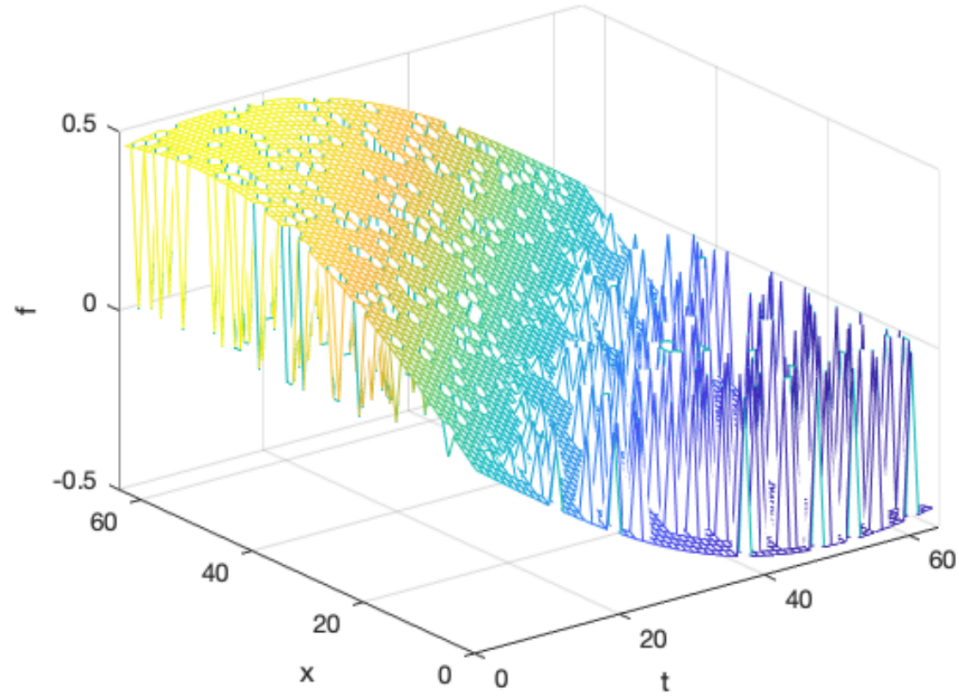


Figure 2.1.6: Gappy Data

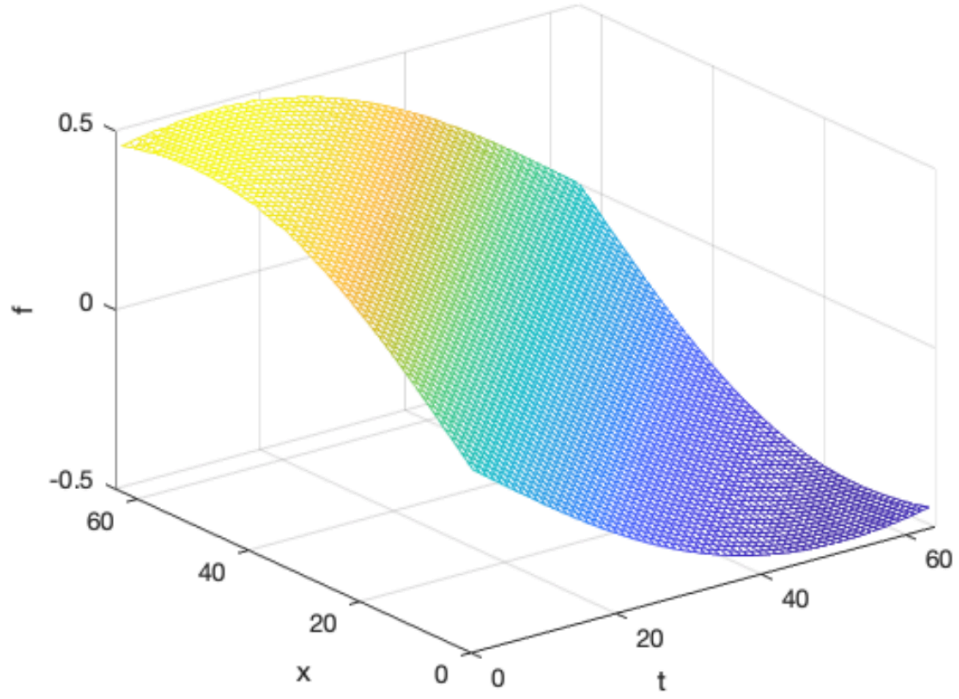


Figure 2.1.7: Recovered Data

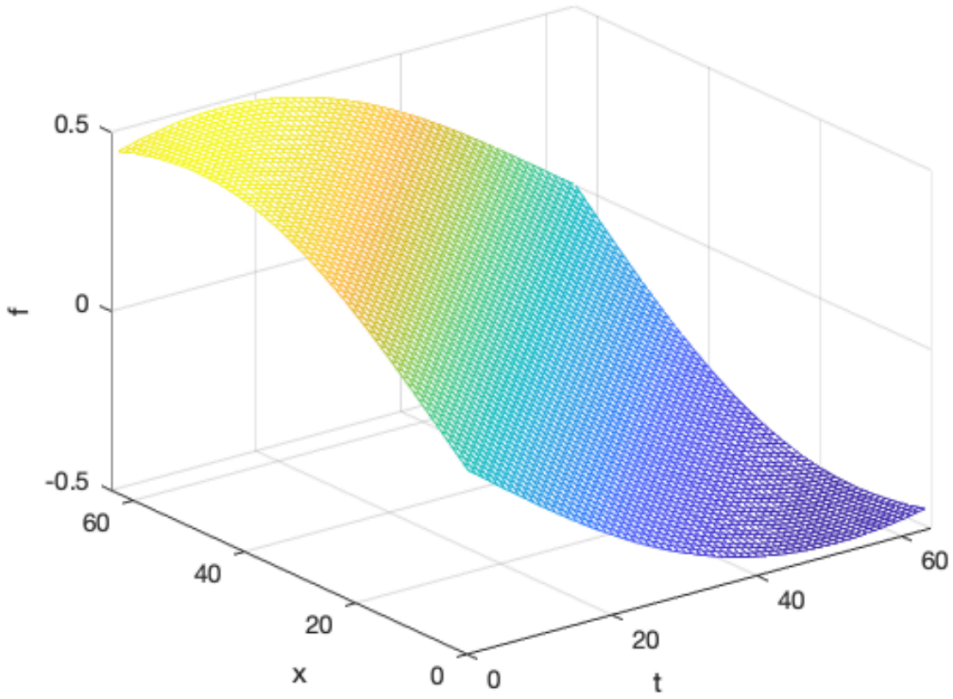


Figure 2.1.8: Original Data

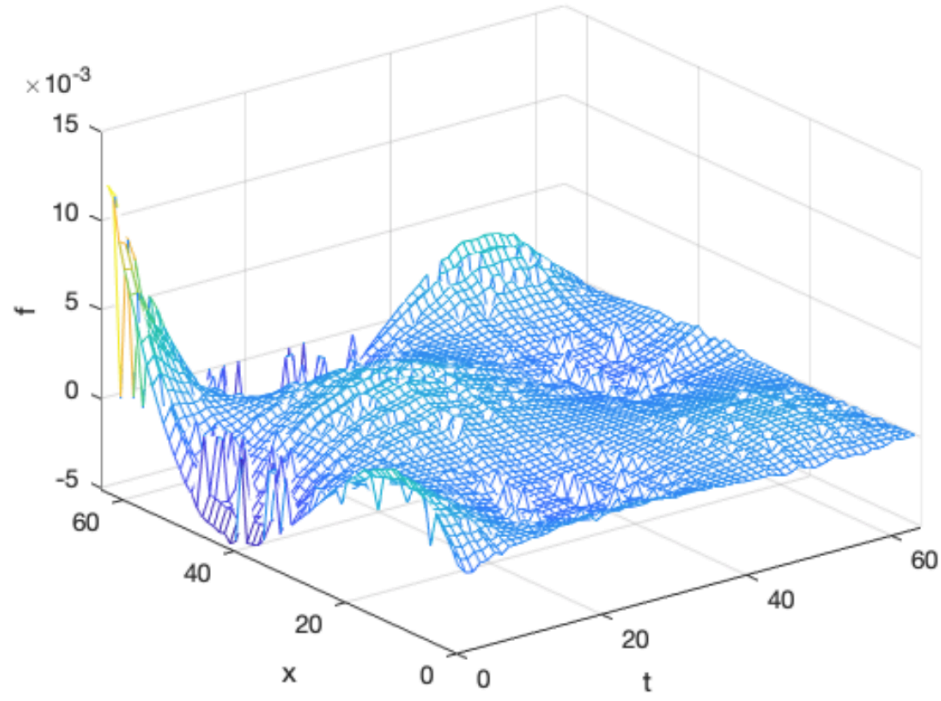


Figure 2.1.9: Error between original and gappy data

2.2 Problem 2

Since as per the fisher criterion:

$$J(w) = \frac{N(w)}{D(w)} = \frac{w^T S_B w}{w^T S_W w},$$

Which reduces to the generalized eigenvalue problem

$$S_B w = \lambda S_W w.$$

Where S_B and S_W are the between class and within class scatter matrices. Based on this problem, the size of these matrices are very big (of the order of ~20k), which makes the svd to run very very slow on a personal computer.

For that reason, we formulate an equivalent generalized eigenvalue problem as follows:

Since $\sum_{i=1}^{20} \alpha_i x_i$, then

$$J(w) = \frac{\sum_{i=1}^{20} \alpha_i x_i^T S_B \sum_{i=1}^{20} \alpha_i x_i}{\sum_{i=1}^{20} \alpha_i x_i^T S_W \sum_{i=1}^{20} \alpha_i x_i}$$

$$= \frac{\alpha^T P_B \alpha}{\alpha^T P_W \alpha}$$

Where $P_B = X^T S_B X$ and $P_W = X^T S_B X$

Here P_B and P_W are 20x20 matrices.

So we can now solve the generalized eigenvalue problem:

$$P_B \alpha = \lambda P_W \alpha$$

We can get P_B by just getting

$X^T(m_2 - m_1)$ and multiplying with its transpose which is a much lower dimensional matrix. Similary we can get P_W

We then first read in the data and compute centers m1 and m2.

We then calculate P_B and P_W as follows:

We calculate M as follows

```
M = data'*(m1-m2);
```
and then P_B

```
Pb = M*(M');
```

```
size(Pb)
```

```
ans =
    20    20
```

Followed by P_W

```
M = data'*[data(:,1:10)-m1, data(:,11:20)-m2];
```

```
size(M)
```

```
ans =
    20    20
```

```
Pw = M*(M');
```

```
size(Pw)
```

```
ans =
    20    20
```

Finally we solve the generalized eigenvalue problem:

```
[V,D] = eig(Sb,Sw);
```

Figure 2.2.1 shows the eigenvalues

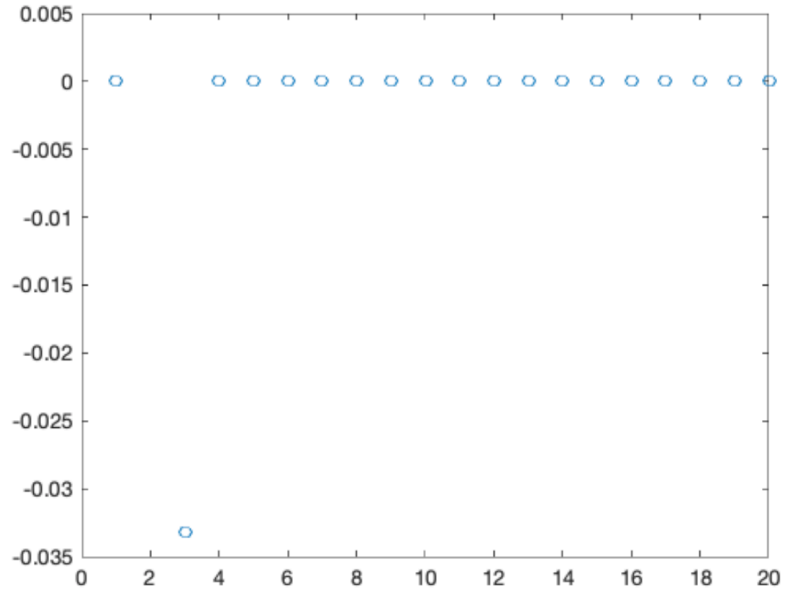


Figure 2.2.1: Eigenvalues

We notice that the 2nd eigen values has most of the energy. Therefore we project the data on the 2nd eigenvector (line).

Figure 2.2.2 shows that the classes are well separated

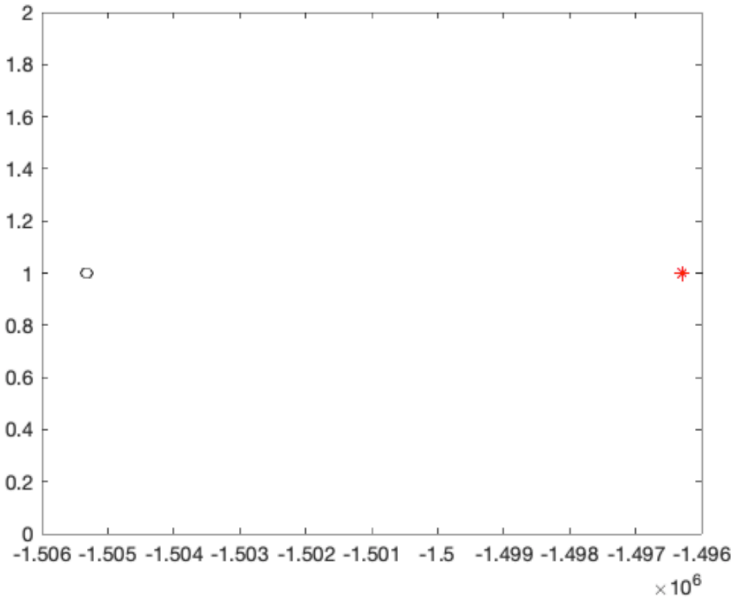


Figure 2.2.2: Classes are well separated after projection

We notice that the classes are well separated and hence this projection successfully linearly sepearated both classes using the feature set.

2.3 Problem 3

We create a 250 dim vector and from that create 10 signals as follows:

```
for i = 1:10
    coef = (rand(10,1)-0.5)*2;
    Xe(:,i) = sin(x*2*pi*(1:10))*coef;
end
```

```
size(Xe)
```

```
ans =

   250    10
```

Code snippet 2.3.1: Create 10 signals of length 250

The 10 original signals are plotted in figure 2.3.1

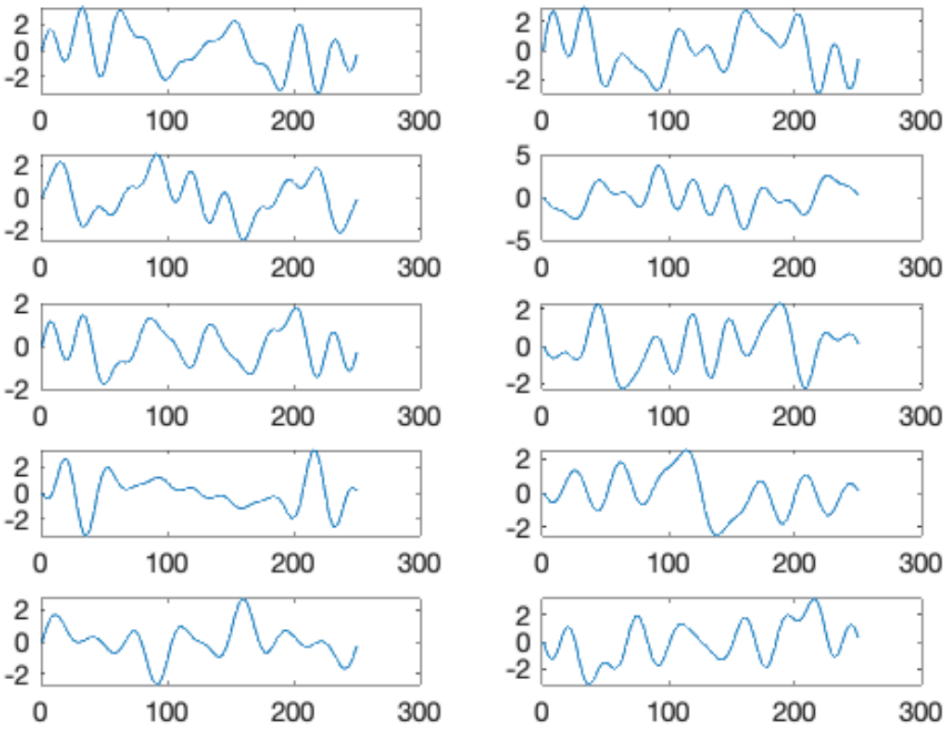


Figure 2.3.1: Original signal

We then create correlated signal of rank 3 and add to original signal as follows:

```
X = Xe+(rand(length(x),3)-0.5)*2*rand(3,10);
```

```
size(X)
```

```
ans =

   250    10
```

Code snipped 2.3.2: Creating noisy signal with noise of rank 3

We then plot the noisy signal as shown in figure 2.3.2

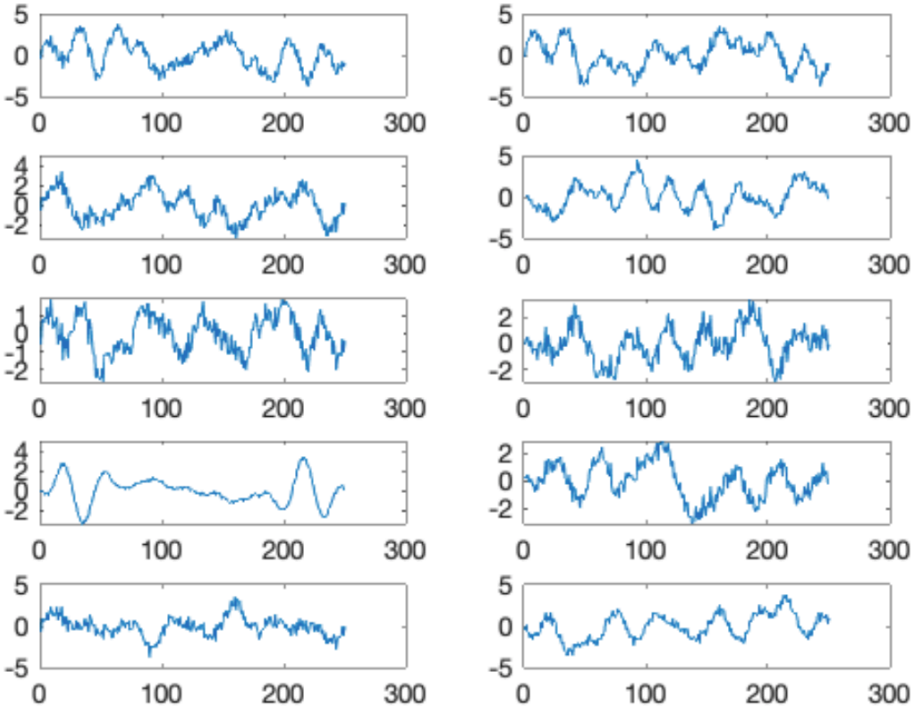


Figure 2.3.2: Noisy signal

We then estimate the noise by taking derivative as follows:

```
N = X(2:end,:)-X(1:end-1,:);
```

```
size(N)
```

```
ans =

   249    10
```

Then we use the matrix N and X to solve the generalized eigenvalue problem:

The eivenvalues are plotted in figure 2.3.3

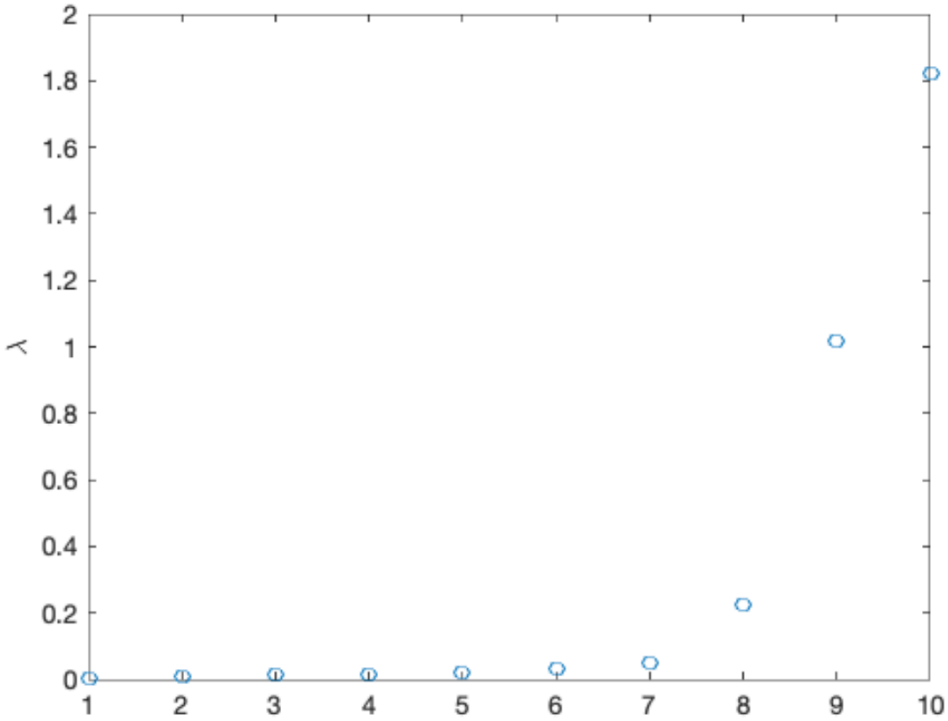


Figure 2.3.3: Eigenvalues

We notice that the first 3 large eigenvalues correspond to the signal. So to recover the signal, we just need the first 7 eigenvalues in terms of the smallest magnitude.

We then calculate ϕ as :

$\phi = X\psi$

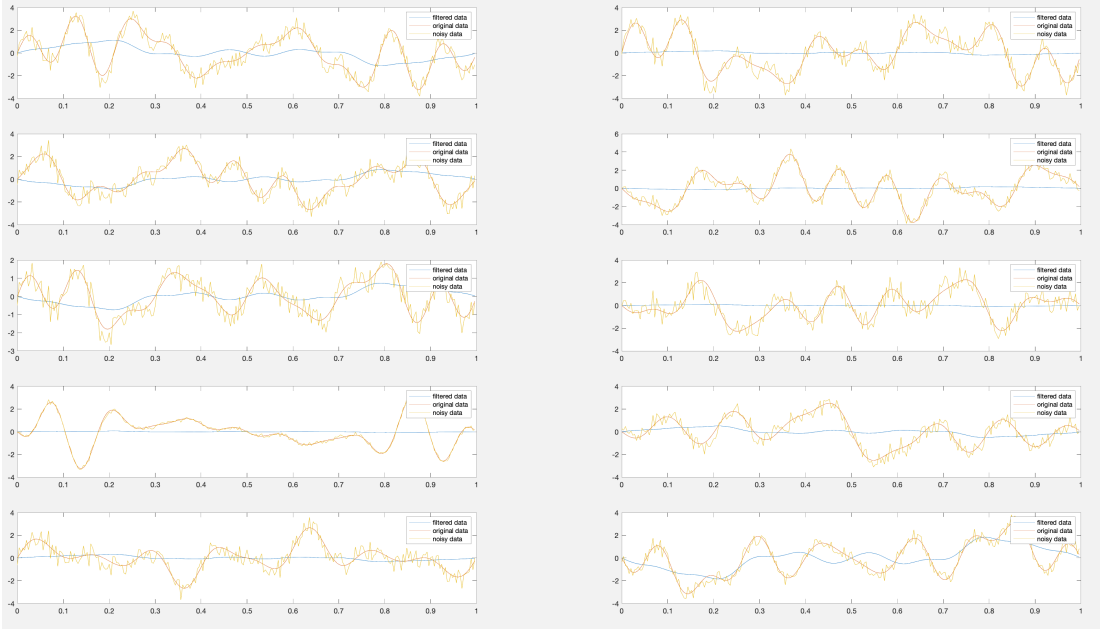
And then the recovered signal

$$Y = \phi\phi^T X$$

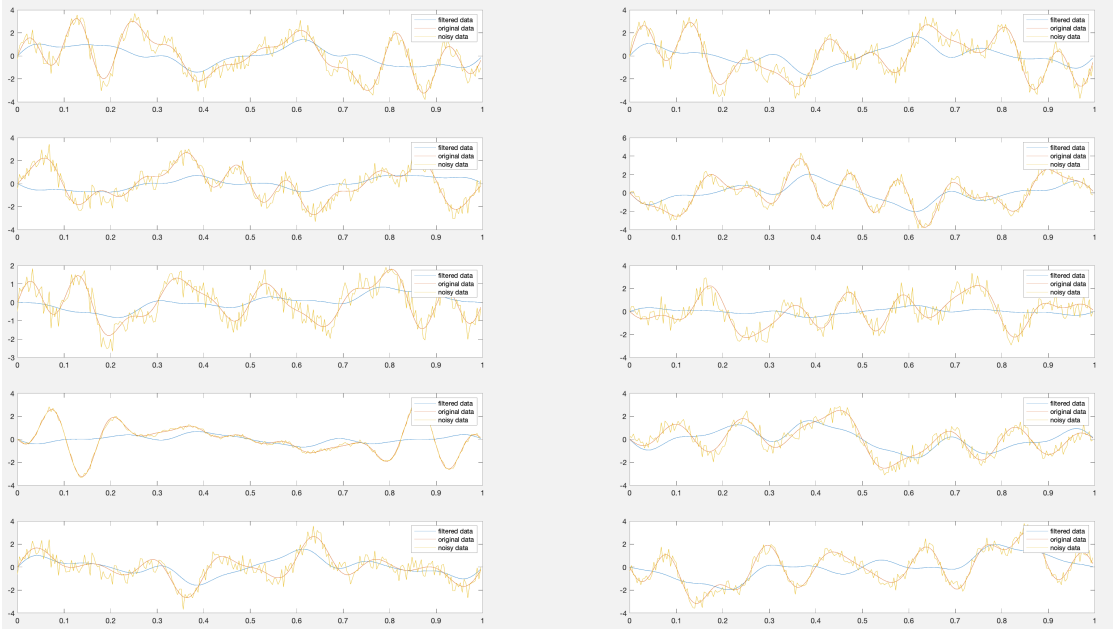
Where ϕ can be the first 7 smallest eigenvalues of the KL basis ϕ.

We then plot the filtered data, original data and noisy data for all values of D where D goes from 1 to 10.

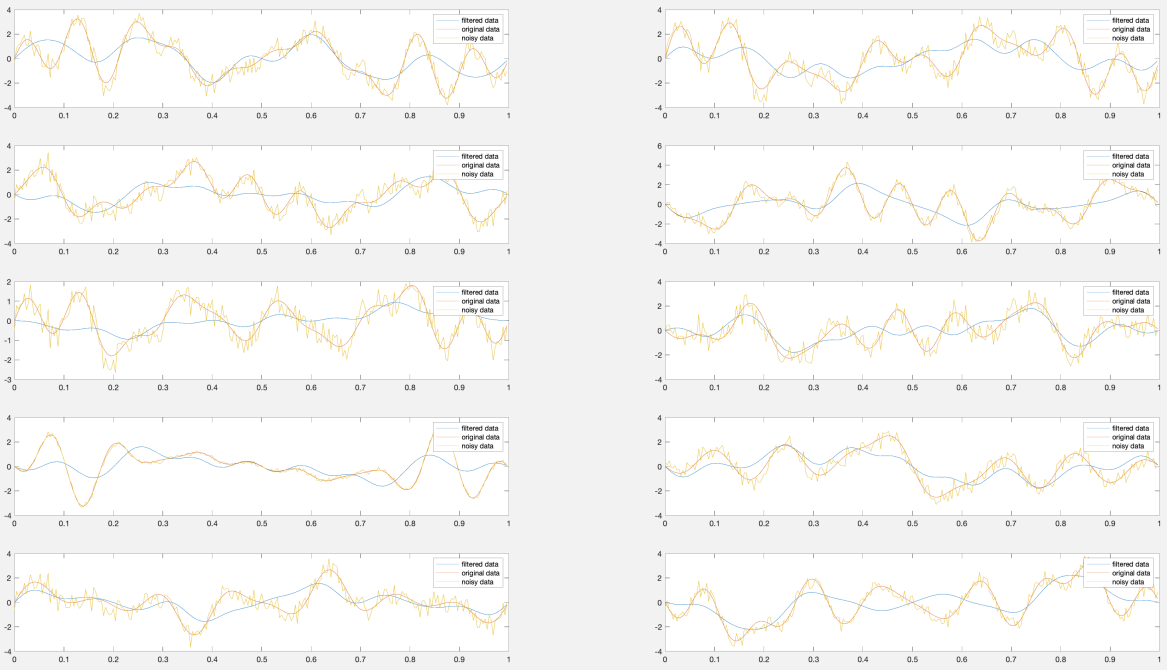
D= 1


filtered data
original data
noisy data

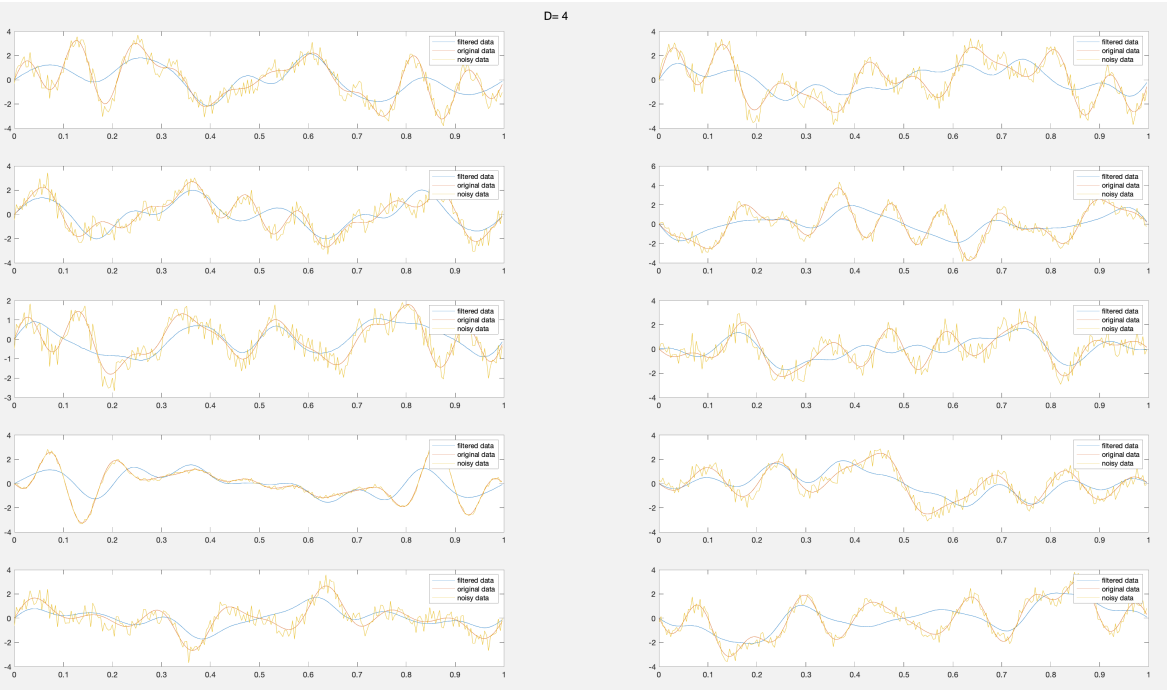
D= 2


filtered data
original data
noisy data

D= 3

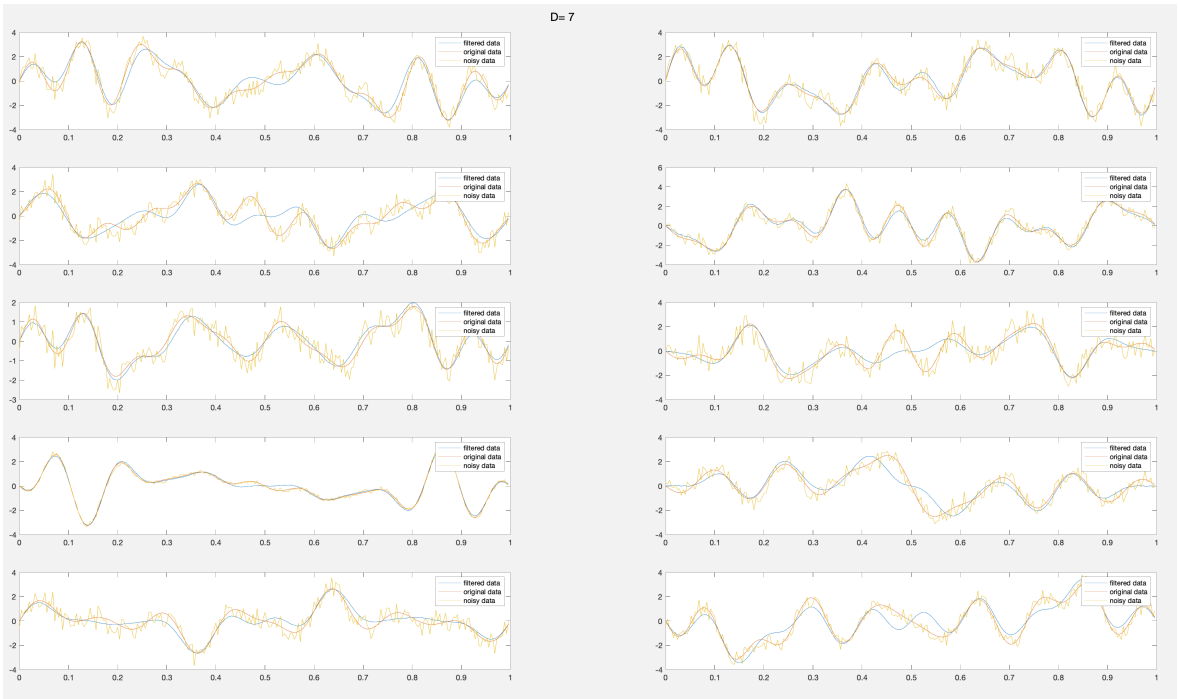

filtered data
original data
noisy data

D= 4

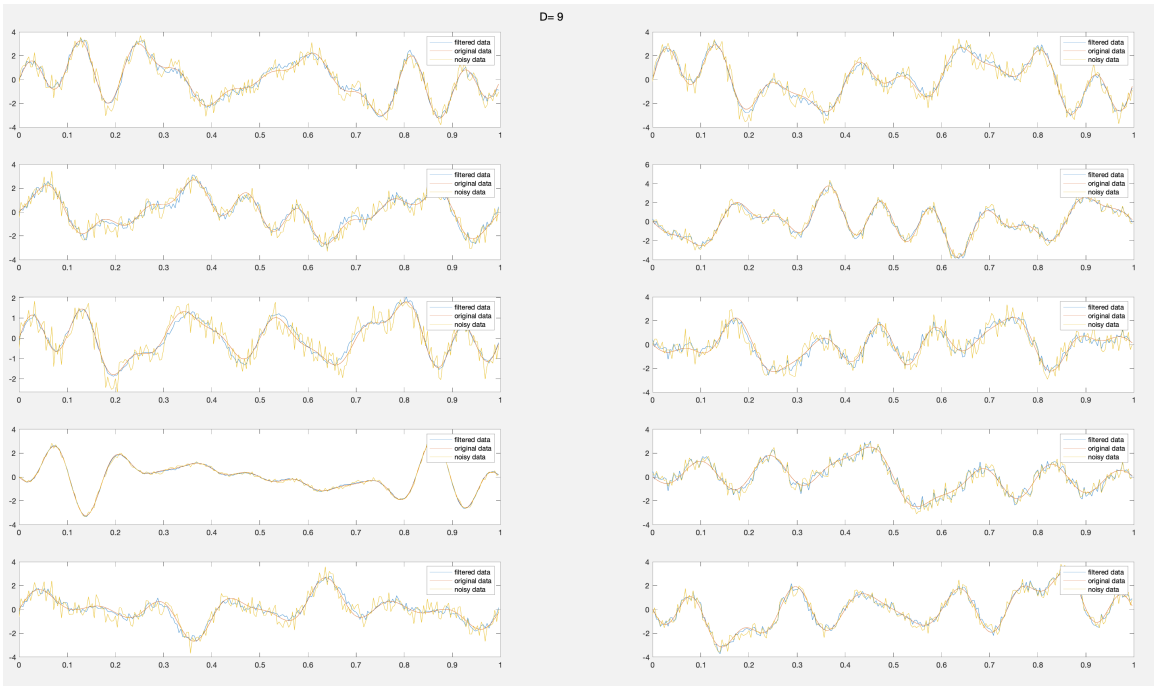

filtered data
original data
noisy data

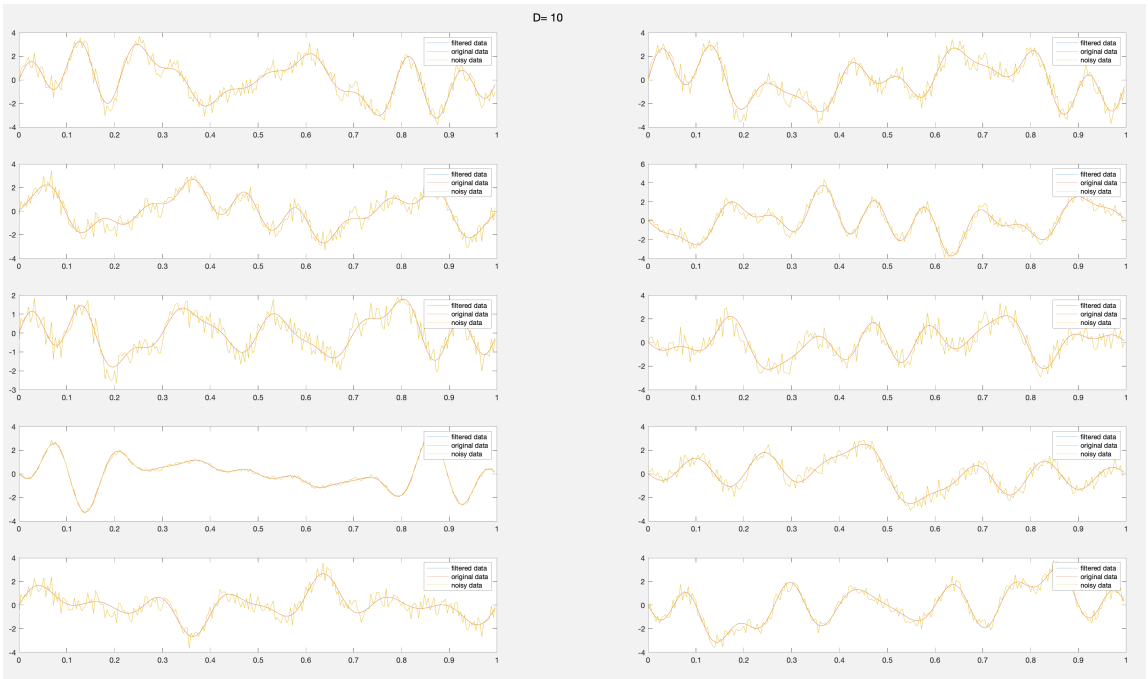

D= 5
filtered data
original data
noisy data


D= 6
filtered data
original data
noisy data


D= 7
filtered data
original data
noisy data


D= 8
filtered data
original data
noisy data





We notice at D=7, we get the best result as the filter data follows the trend of original data without much noise. Where as at D=1 there is no noise but the filtered data has a different trend and at D=10 it exactly matches the noisy data.

2.4 Problem 4

We load the image she visualize it:

```
A = imread('app-ndt-Chip-5.jpg');
```

```
imshow(A);
```

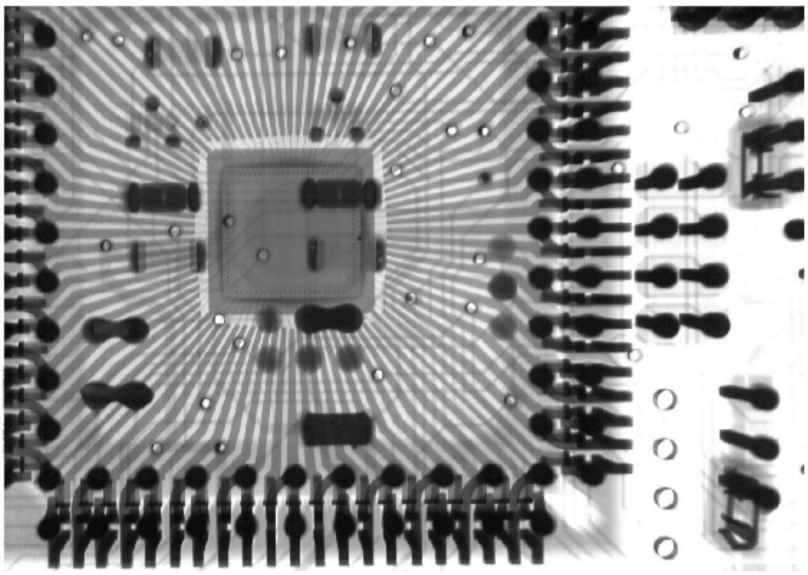


Figure 2.4.1: The original image

We then add 40% salt and pepper noise to the image:

```
A = imnoise( A ,'salt & pepper', .4);
```

```
imshow(A);
```

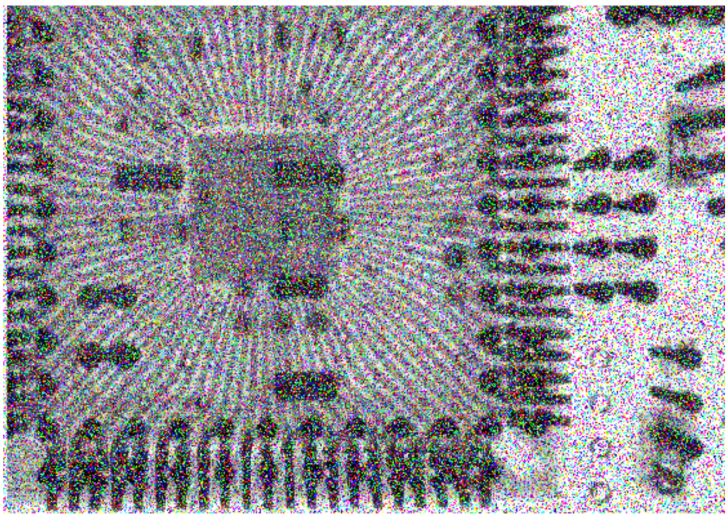


Figure 2.4.2: The image with salt and pepper noise

We then apply a 3x3 median filter as follows:

```
A1 = medfilt3(A);        % 3x3X3 Median filter output for degraded image
```

```
imshow(A1);
```

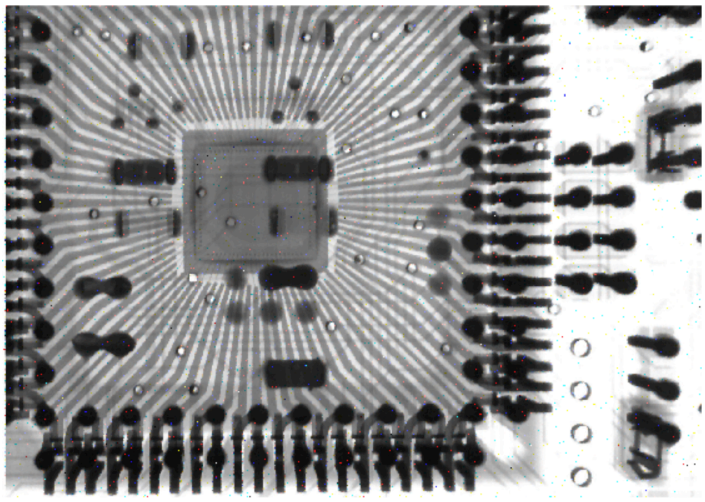


Figure 2.4.3: After applying 3x3 median filter

We notice that there are still some lingering noise.
We then apply a bigger 5x5 media filter.

```
A2 = medfilt3(A,[5 5 5]);        % 5x5 Median filter output for degraded image
```

```
imshow(A2);
```

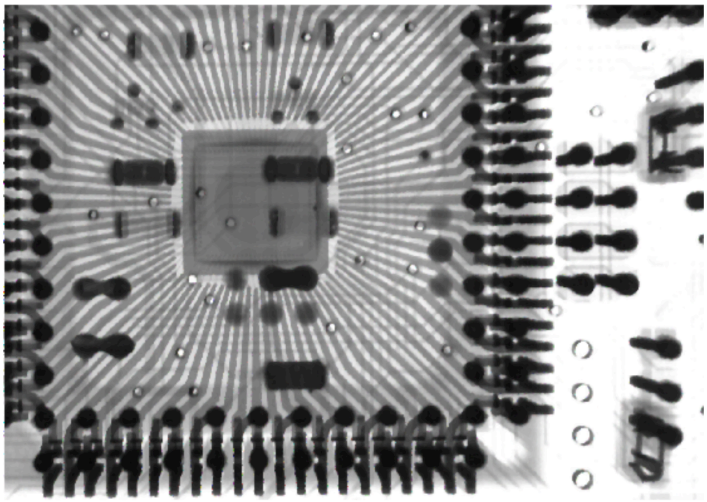


Figure 2.4.4: After applying 5x5 median filter

As expected, the noise is completely gone and the output looks like the original image.

2.5 Problem 5

We load the original image and display it.

```
A=imread('CTimage.jpg');
```

```
imshow(A);
```

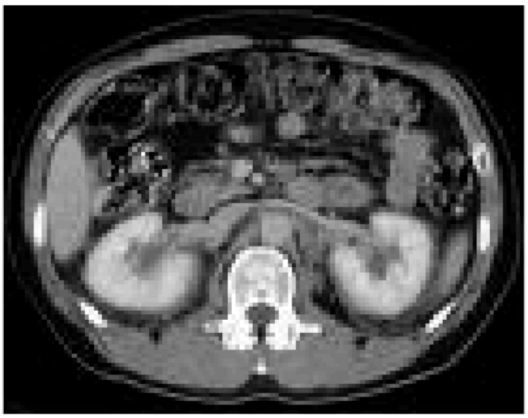


Figure 2.5.1: Original image

The image looks pixalated and not very sharp.

We then smooth the image by applying a 3x3 average filter as follows:

```
A = rgb2gray(A);
```

```
avg3 = fspecial('average');
```

```
A1 = imfilter(A,avg3);
```

```
imshow(A1);
```

Figure 2.5.2: Blurred image after 3x3 average filter

We then construct the Laplacian filter and visualize it.

```
A1=double(A1);
```

```
lapFilter = [0 -1 0; -1 4 -1; 0 -1 0];
```

```
A2 = conv2(A, lapFilter,'same');
```

```
A2=double(A2);
```

```
A3 = A1+A2;
```

```
imshow(uint8(A2));
```

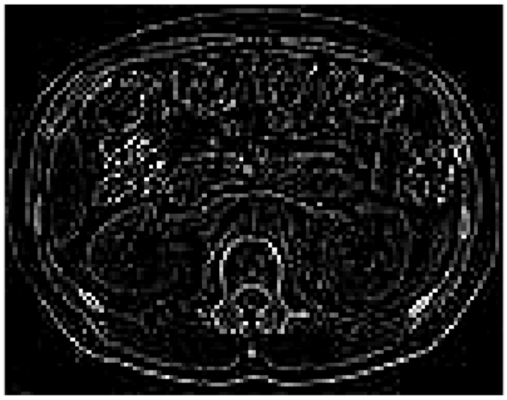


Figure 2.5.3: After applying the Laplacian filter

We then add the above image to the blurry image to sharpen the image. The image looks much more sharpened as shown below.

```
imshow(uint8(A3));
```

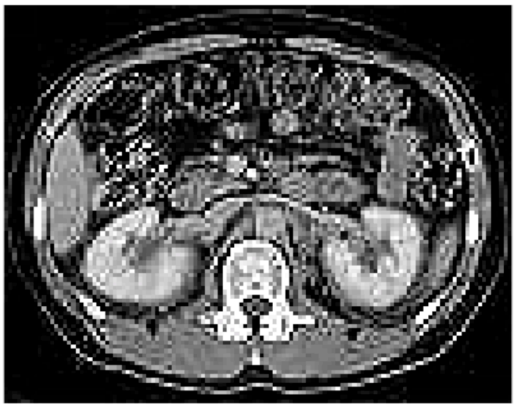


Figure 2.5.4: Sharpened image.

We notice that the resulting image looks much more sharpened than the original and hence easier to read.