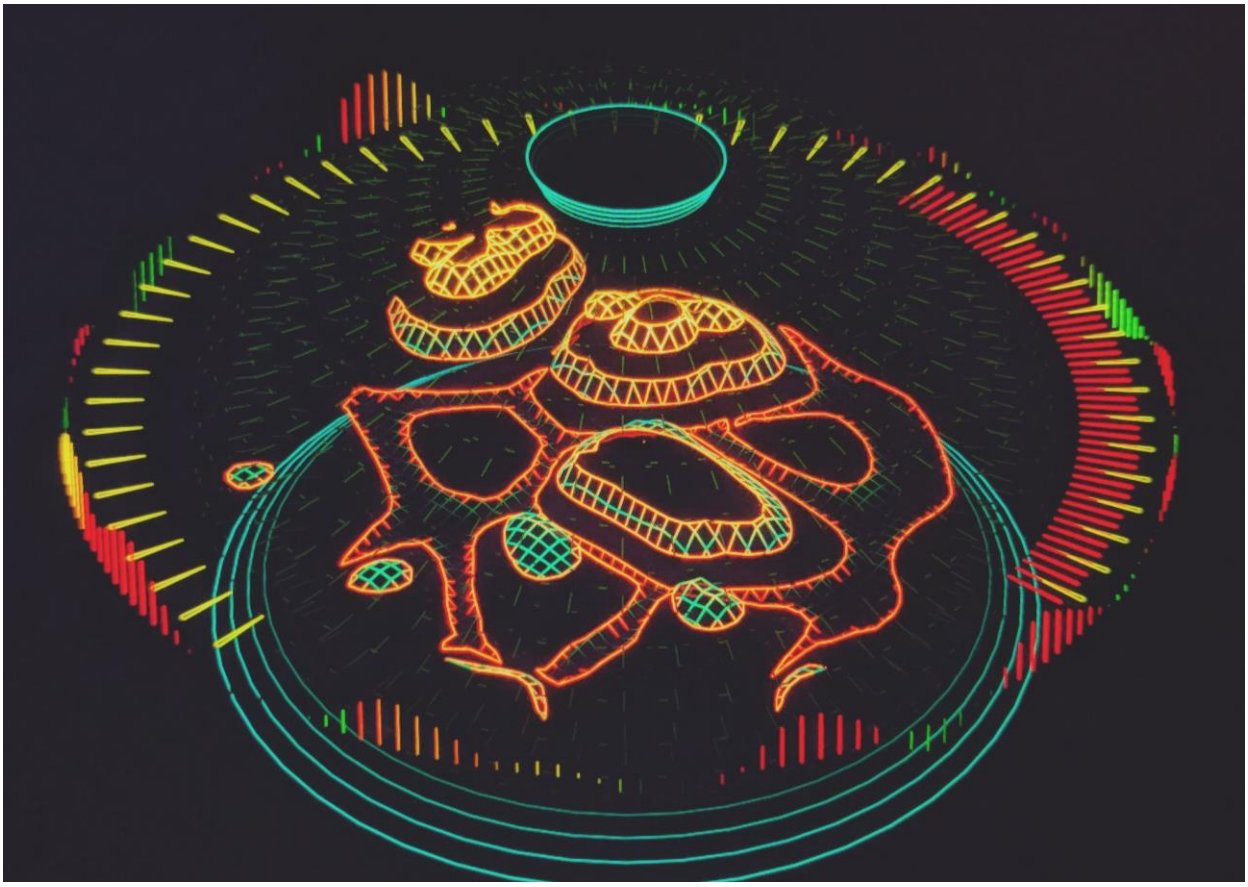
Programming assignment

EEL2010 : Signals and Systems (Summer Term 2020-21)



Gaurav Sangwan

Jul,24 2021

Indian Institute of Technology , Jodhpur

THEORY

The problem have asked us to do two different operations on the signal :

(i) Denoise

(ii) Deblur

Denoise

Denoise is the reverse of adding noise to signal, Noise is nothing but addition of high frequency, this type of noise is known as additive noise.

$$x[n] + \varepsilon_n = g[n]$$

Here, x[n] is the original signal

ε_n is a number of the noise signal

g[n] is the obtained noisy signal

Therefore, to get x[n] from g[n] I need to estimate the value of ε_n , which is impossible without any given knowledge about it. So, what I do is, I average values of near neighbours of x[n] to get the average value of x[n] , which may not be completely correct but as I increase the no. of near neighbours I decrease the error percent betIen original x[n] and avg. x[n]

In the following assignment, I have opted for an average of 3 values x[n-1],x[n] and x[n+1].And smoothen the input signal.

Deblur

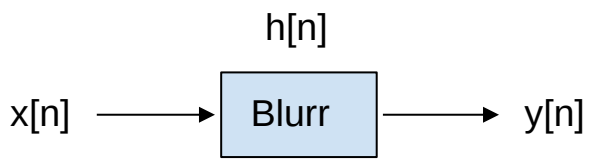
Blurring is a type of low pass filter ;it blocks out high frequency components of a signal , the way it blocks them depends on its impulse function.In the given problem the input function is

$$h[n] = (1/16) * [\ 1\ 4\ 6\ 4\ 1]$$

where , n = 0 points to 6/16

So I had the impulse response of a given blurring system, now for deblurring I just have to invert the system so that I can get the original signal (x[n]).The equation of following system is

$$y[n] = x[n] * h[n]$$



The inverse of signal I get let's say h1[n], can be obtained in 2 ways :

1. By changing h[n] to $H(e^{j\Omega})$ and then putting it in the following equation(convolution property).

$$Y(e^{j\Omega}) = X(e^{j\Omega}).H(e^{j\Omega})$$

$$\Rightarrow \quad X(e^{j\Omega}) = Y(e^{j\Omega})/H(e^{j\Omega})$$

So the required impulse response h1[n] have a fourier transform equal to $1/H(e^{j\Omega})$.

Now doing an inverse fourier transform of $1/H(e^{j\Omega})$ to get h1[n]. And then convolving it with y[n].

To get x1[n].(deblurred signal of y[n])

But, this method requires us to declare 3 different function

- Fourier transform
- Inverse Fourier transform
- Convolution

2. By changing h[n] to $H(e^{j\Omega})$ and then putting it in the following equation (convolution property).

$$Y(e^{j\Omega}) = X(e^{j\Omega}).H(e^{j\Omega})$$

$$\Rightarrow \quad X(e^{j\Omega}) = Y(e^{j\Omega})/H(e^{j\Omega})$$

Now computing RHS of the above equation by element wise division to obtain $X(e^{j\Omega})$. Then doing Inverse Fourier transformation of $X(e^{j\Omega})$.

To get x1[n] (deblurred signal of y[n])

This method requires us to declare only 2 different function :

- Fourier transform
- Inverse Fourier transform

Hence this method is better than the previous one.

DEBLURRING(designing the system)

Calculation of $H(e^{j\Omega})$:

Now, since

$$h[n] = (1/16) * [\ 1\ 4\ 6\ 4\ 1]$$

where , n = 0 points to 6/16

Using the following formula of DTFT to get its fourier transform $H(e^{j\Omega})$:

$$H(e^{j\Omega}) = \sum_{n=-\infty}^{\infty} h[n].e^{-j\Omega n}$$

I get ,

$$H(\Omega) = (1/8).[\cos(2\Omega) + 4\cos(\Omega) + 6]$$

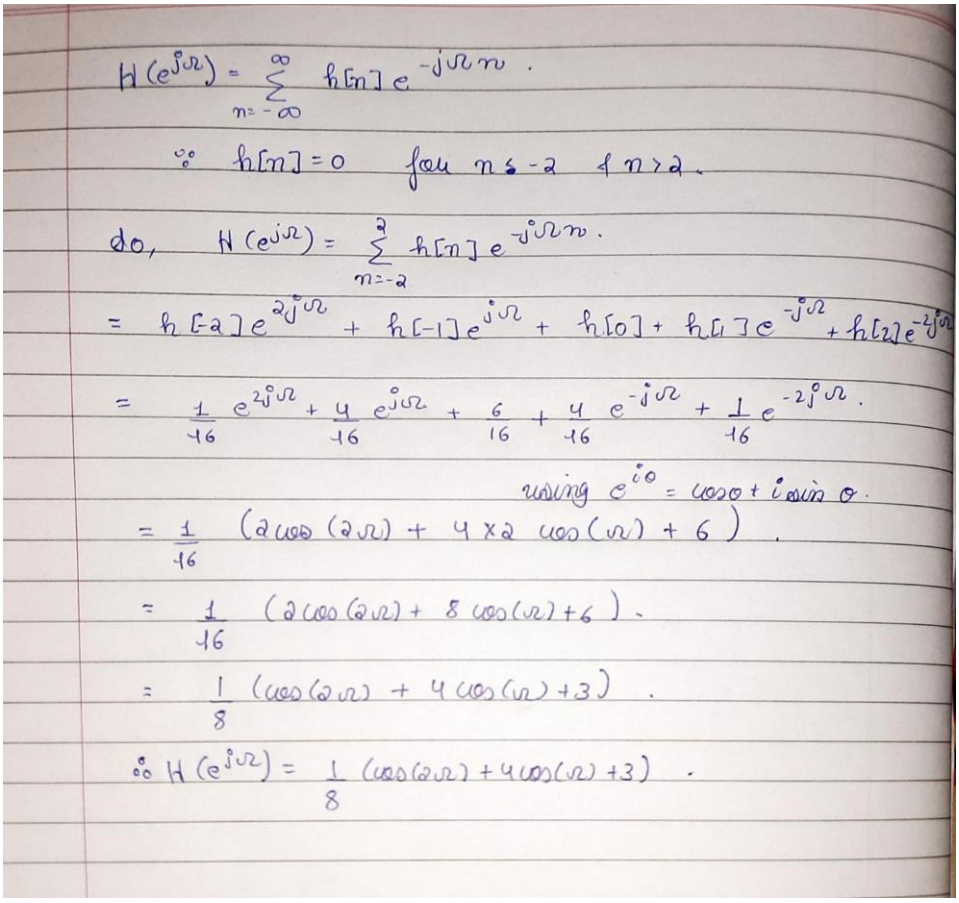
Or

$$H(e^{j\Omega}) = (1/16).[(e^{j2\Omega}) + 4\,(e^{j\Omega}) + 6 + 4(e^{-j\Omega}) + (e^{-j2\Omega})$$

(Calculation on the attached image)

Here I can conclude that $H(\Omega)$ is a real function, so I don't have to worry about its complex division while doing $Y(e^{j\Omega})/H(e^{j\Omega})$ as mentioned above.

Calculation for $H(e^{j\Omega})$(contd.):



Sampling of $H(e^{j\Omega})$:

Now as obtained above $H(e^{j\Omega})$ is a continuous and real function, but I cant process a continuous function digitally that's why I do sampling of given $H(e^{j\Omega})$ to get an array of values of $H(e^{j\Omega})$ for corresponding Ω . So there are 2 pieces of information available for sampling :

1. $H(e^{j\Omega})$ is DTFT hence it will lie in an interval of 2π .
2. Since length of array containing x[n] and y[n] is 193, so the length of Ω must also be 193

That's why I divide Ω from 0 to 2π and gap betIen 2 consecutive term is $\frac{2\pi}{192}$.

Or, as expressed in formula

$$\Omega = \left(\frac{2\pi}{N}\right) * k$$

Now I can create a new array of length 193 and store the value of $H(e^{j\Omega})$ with corresponding Ω. Thus I get $H(e^{j\Omega})$'s sampled version and I can now proceed ahead.

Getting Y($e^{j\Omega}$) :(DFT_GA function of Code)

Now to move ahead with our said approach in theory, I need to find Y($e^{j\Omega}$) from y[n].

Calculation Part :

As I know for a DTFT conversion I use following formula

$$Y(e^{j\Omega}) = \sum_{n=-\infty}^{\infty} y[n].e^{-j\Omega n}$$

Now as I did above for sampling of H($e^{j\Omega}$), I take Ω =$\left(\frac{2\pi}{N}\right) * k$ where, k have the same limits as n.

Therefore , our Y ($e^{j\Omega}$) turns into :

$$Y(e^{j\Omega}) = \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} y[n].e^{-j\left(\frac{2\pi}{N}\right)kn}$$

Or

$$Y(k) = \sum_{n=-\infty}^{\infty} y[n].e^{-j\left(\frac{2\pi}{N}\right)kn}$$

Since it is a digital system I can change n's limit to 0 to N-1

$$Y(k) = \sum_{n=0}^{N-1} y[n].e^{-j\left(\frac{2\pi}{N}\right)kn}$$

And thus I get the DTFT of y[n].

Calculating X'($e^{j\Omega}$) :

I have Y($e^{j\Omega}$) and H($e^{j\Omega}$) now using the system equation described above I get the value of X'($e^{j\Omega}$) (where ' means processed signal).

Since I have both Y($e^{j\Omega}$) and H($e^{j\Omega}$) in the form of an array I will perform an element wise division,

If A = [a b c] and B = [p q r]

Then A/B = [a/p b/q c/r]

Element wise division

But, there is one problem that arises while doing Y($e^{j\Omega}$)/H($e^{j\Omega}$) whenever the value of H($e^{j\Omega}$) goes less than a value then 1/H($e^{j\Omega}$) becomes large and it makes a high frequency component at that place or it amplifies noise . So to take care of that I assume a threshold,

let say ε and then instead of doing $Y(e^{j\Omega})/H(e^{j\Omega})$ I put that threshold in $X(e^{j\Omega})$ so that I don't end up getting a high frequency component.

Mathematically ,

$$X(e^{j\Omega}) = Y(e^{j\Omega)}/H(e^{j\Omega}) \quad when\ H(e^{j\Omega}) > \varepsilon$$

$$X(e^{j\Omega}) = \varepsilon \qquad\qquad when\ H(e^{j\Omega}) < \varepsilon$$

In our program I have assumed ε to be 0.5, because if I would have gotten closer to 0 then sharpness would have dominated and if I would have gotten closer to 1 then smoothness would have dominated.Therefore to counter this tradeoff betIen smoothness and sharpness I took ε to be 0.5

Calculating x'[n] :(I-DTFT_GA function of CODE)

As I have obtained the inverse system output of y[n] in the frequency domain, its equivalent in the time domain will be x'[n]. To get x'[n] I will do Inverse :

As I know that for a continuous $X(e^{j\Omega})$ I have Inverse-DTFT as :

$$x[n] = \left(\frac{1}{2\pi}\right)\int_0^{2\pi} X(e^{j\Omega})e^{j\Omega n}d\Omega$$

Now as I did above for sampling $\Omega = \left(\frac{2\pi}{N}\right) * k$ putting this in formula and changing integration into summation using reimann's integration I get

$$x[n] = (1/N)\sum_{n=0}^{N-1} X[k].e^{j\left(\frac{2\pi}{N}\right)kn}$$

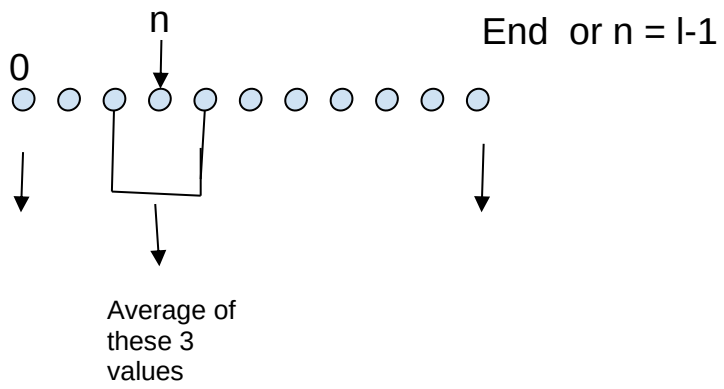
And thus I get x'[n] from X(k) or $X(e^{j\Omega})$.

And this x'[n] will be the deblurred signal of input signal y[n].

DENOISING(designing the system)

deNoise function of the code

As discussed in Theory above I will be using the averaging approach for designing the denoising system. For doing denoising of a signal , let say x[n], I will make a blank signal let say y[n] for every n in x I will compute average of x[n-1] , x[n] and x[n+1] and put that to corresponding n in y[n].



But the problem arises when I are at the first or last element , i.e n = 0 or l-1(where l is the length of the signal). The problem at these 2 points is that there is 1 missing element one on the left and one on the right so for that, I make a conditional statement where whenever it is starting of the signal I will take avg of first 3 values and whenever it is the end of signal I take the average of last 3 values , and if not both of them then I do it in the normal way

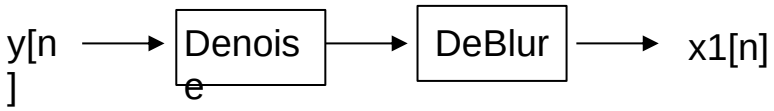
$$x[n] = (\frac{1}{3}) * (x[n-1] + x[n] + x[n+1]$$

PROCEDURE

So now the problem has asked us to use 2 different approaches to process the received signal and try to get the original signal.

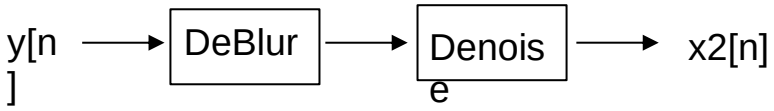
1. System A :

   First remove noise and then sharpen(Deblur) the signal to get x1[n]



2. System B :

   First sharpen(Deblur) and then remove noise from the signal to get x2[n]



Code Structure :

First I input the calculated $H(e^{j\Omega})$ and sample it . Then I Plot a Figure showing graphs of x[n] and y[n] on the same plot , h[n] and $H(e^{j\Omega})$.

After that I do Part A by first doing the Deblur system as described above and then doing denoise. Similarly then I do Part B by first denoising and then deblurring.

Then I make a new figure and make 4 subplots and I plot: x[n] and y[n]; x[n] and x1[n] ; x[n] and x2[n] ; and at last x1[n] and x2[n].

CODE

The following code is also present in the **PA_MT21_BB03.m** file; it is directly copy-pasted from the file.

```
%clearing command window and making output format compact
clc

format compact
%main body
%importing data from data.csv as data.mat
%by running this syntax in command line load("/MATLAB Drive/data/data.mat")
%or using UI interface
%declaring the impulse response of the blurring system(i.e h[n] , given)
hn = (1/16)*[1 4 6 4 1]
%making 193 samples of w
w = 0:(2*pi/192):2*pi
%as calculated in report
H = (1/16)*(exp(2*1i*w)+4*exp(1i*w)+6+4*exp(-1i*w)+exp(-2*1i*w))
%plotting all this information on a figure
figure;
subplot(3,1,1);
title("Plot of y[n] and x[n] ");hold on;
xlabel("n"); hold on;
plot(data.xn);hold
on;plot(data.yn);legend({"x[n]","y[n]"},"location","southeast");hold on;
subplot(3,1,2);
title("Plot of h[n] ");hold on;
xlabel("n");hold on;
stem(-2:2,hn);hold on;
%plotting H(w) to get a better view of H(w) overe -pi to pi I declare a
%new variable w1
w1 = -pi:(2*pi/192):pi
HH = (1/16)*(exp(2*1i*w1)+4*exp(1i*w1)+6+4*exp(-1i*w1)+exp(-2*1i*w1))
subplot(3,1,3);
title("Plot of H(w)");hold on;
xlabel("w");hold on;
plot(w1,HH)
%converting y[n] from a column vector to row vector
%below in the program there would many instances of using transpose because
%while doing the calculations I need to have every vector as a row vector
%It is to make every column vector a row vector.


y = (data.yn)'
```

```
Y = dft_ga(y)
Y = Y'
%
%
% Part A denoise and then deblur
%
%
%denoise
y1 = deNoise(y)
%deblur
Y1 = dft_ga(y1)
Y1 = Y1'
Y11 = []
for k = 0:length(H)-1
   if abs(H(k+1)) < 0.5
       Y11=[Y11 0.5]
   else
       temp = (Y1(k+1))/(H(k+1))
       Y11 = [Y11 temp]
   end
end
xn1 = idft_ga(Y11)
xn1 = (xn1)'
%
%
% Part B deblur then denoise
%
%
%deblur
Y = dft_ga(y)
Y = Y'
X =[]
for k = 0:length(H)-1
   if abs(H(k+1)) < 0.5
       X=[X 0.5]
   else
       temp = (Y(k+1))/(H(k+1))
       X = [X temp]
   end
end
x1 = idft_ga(X)
%denoise
xn2 = deNoise(x1)
xn2 = (xn2)'
%plotting every processed signal for observation
figure;
subplot(2,2,1);
```

```
title("Plot of y[n] and x[n]");hold on;
xlabel("n");hold on;
plot(data.yn);hold on;plot(data.xn);legend({"y[n]","x[n]"},"location","southeast");
subplot(2,2,2);
title("Plot of x[n] and x1[n]");hold on;
xlabel("n");hold on;
plot(data.xn);hold
on;plot(abs(xn1));legend({"x[n]","x1[n]"},"location","southeast");
subplot(2,2,3);
title("Plot of x[n] and x2[n]");hold on;
xlabel("n");hold on;
plot(data.xn);hold
on;plot(abs(xn2));legend({"x[n]","x2[n]"},"location","southeast");
subplot(2,2,4);
title("Plot of x1[n] and x2[n]");hold on;
xlabel("n");hold on;
plot(abs(xn1));hold
on;plot(abs(xn2));legend({"x1[n]","x2[n]"},"location","southeast");
%plotting x1[n] vs x2[n] on a new figure to zoom in and take screenshot for
%observation
figure;
title("Plot of x1[n] and x2[n]");hold on;
xlabel("n");hold on;
plot(abs(xn1));hold
on;plot(abs(xn2));legend({"x1[n]","x2[n]"},"location","southeast");
%Functions
%DtFT
function [ X ] = dft_ga( x )
N=length(x);
n=0:N-1;
k=0:N-1;
W=exp((-1i*2*pi*k'*n)/N);
X=W*x';
end
%IDFT
function [ x1 ]= idft_ga( X )
N=length(X);
n=0:N-1;
k=0:N-1;
w=exp((1i*2*pi*k'*n)/N);
x1=w*X'/N;
end
%Denoise
function [yden] = deNoise(x)
%I will take average of 3 surrounding valeus (reason in report)
yden = []
l = length(x)
```

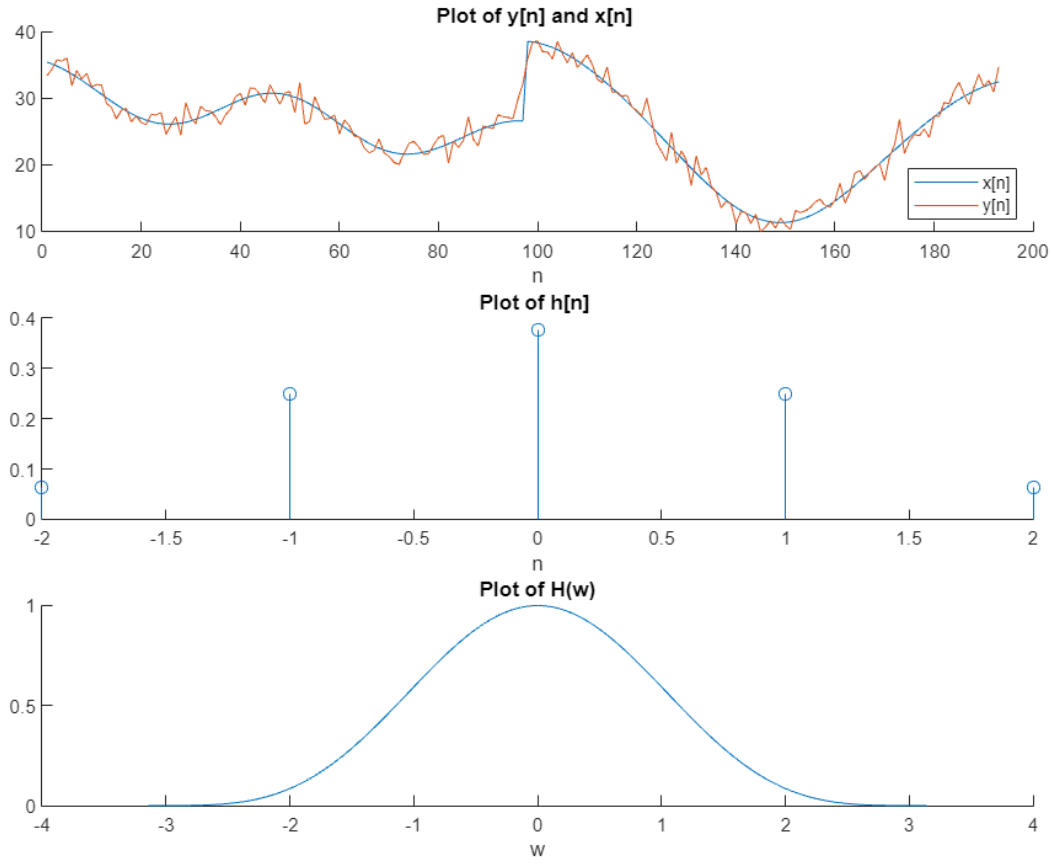
```
for i = 0:l-1
   if i == 0
       temp = (x(i+1)+x(i+2)+x(i+3))/3
       yden = [yden temp]
   elseif i == l-1
       %temp = (x(i+1)+x(i)+x(i-1)/3)
       temp = x(i+1)
       yden = [yden temp]
   else
       temp = (x(i)+x(i+1)+x(i+2))/3
       yden = [yden temp]
   end
end
end
```

P.S. How to execute the code and complete procedure can be found in “Readme.txt” attached.

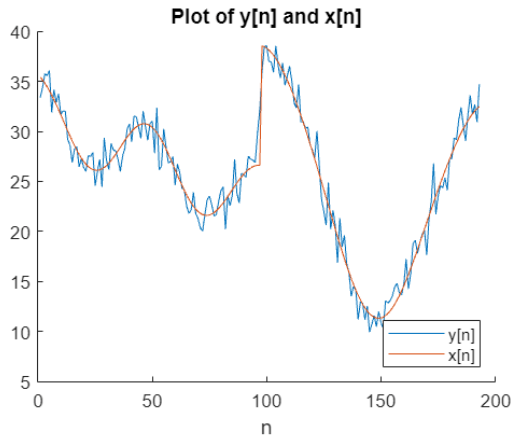
OUTPUT OF THE PROGRAM a.k.a RESULTS

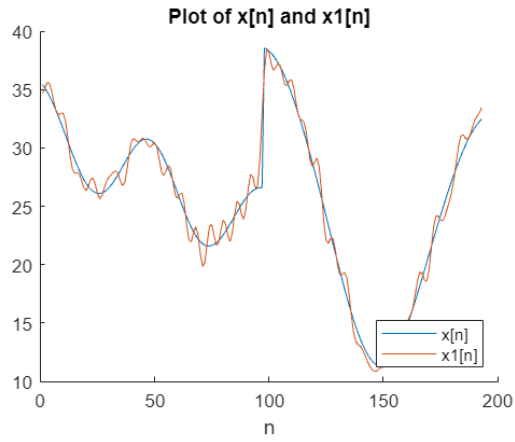
The output obtained from running the above said program by importing data.csv is as follows, I have used graphical representation to get more clarity in the differences betIen the plots.

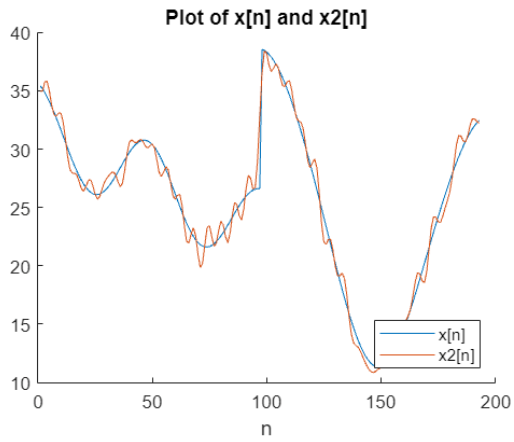
Plot of input or known signals

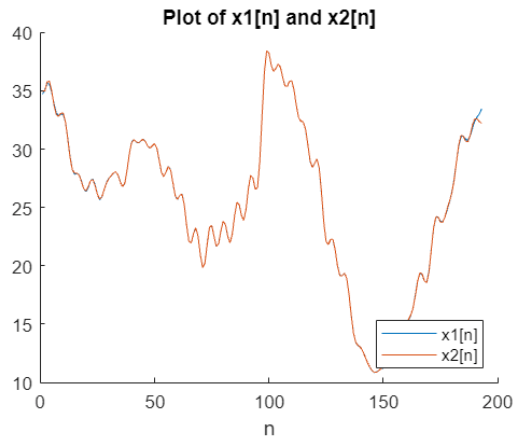


Plot of Output or processed signals



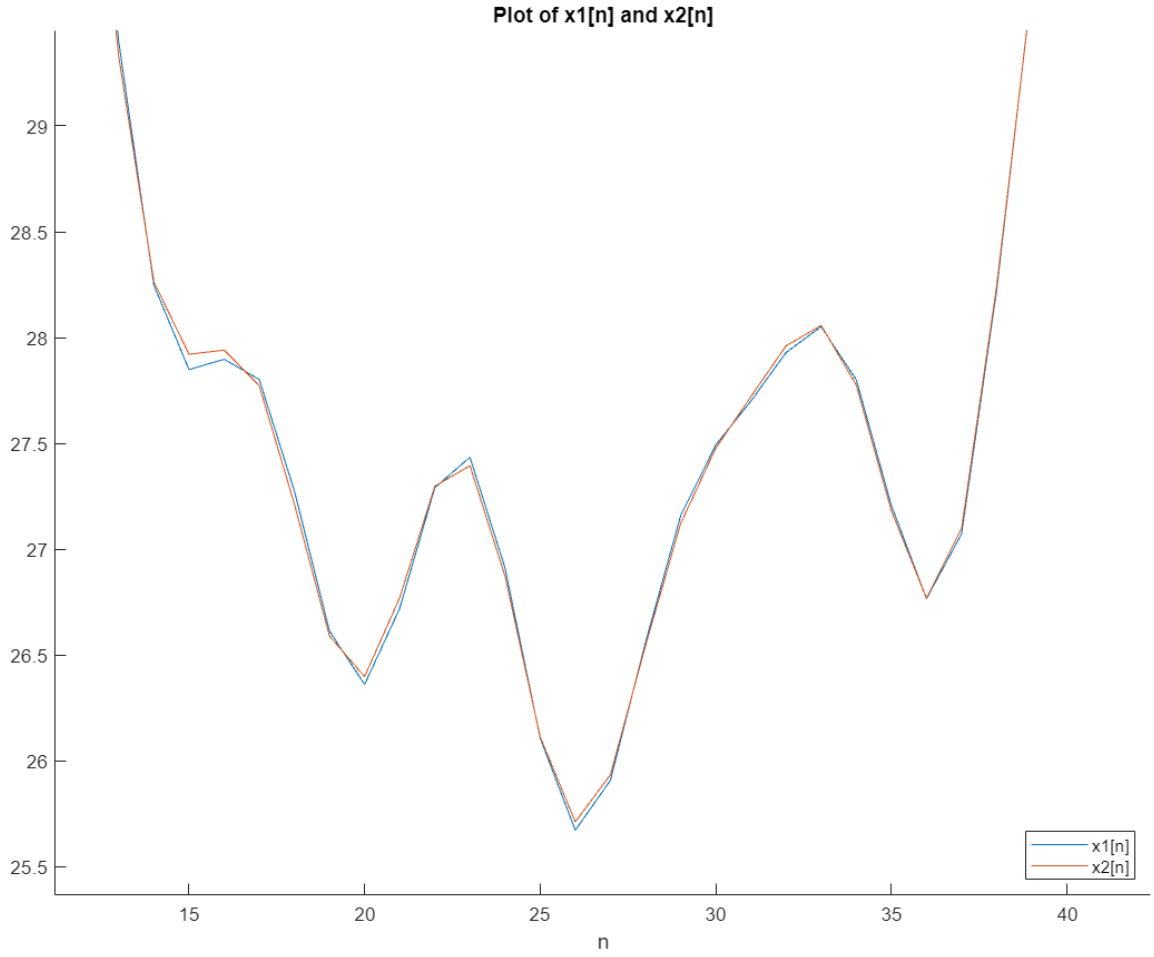






Zoomed in version of "Plot of x1[n] and x2[n]"

I zoomed in because in the zoomed out version I Ire getting that x1[n] and x2[n] are the same.



OBSERVATION

Comparing the output functions of both system A and system B (output say x1[n] and x2[n] respectively).By using RMSE I can see that both the output signals are almost the same . And the difference betIen them is very minute even on zooming in , but still x1[n] is more accurate because it is neither much sharp nor much smooth, whereas x2[n] is quite sharp because of sudden high and lows at extremities.

THEORETICAL JUSTIFICATION

As I discussed above noise is addition of high frequency and blurring is removal of high frequency components or as I call Low Pass Filter, so its inverse will be a High pass filter or addition of some high frequency components.

Now let's talk about all this in a mathematical way.

Let obtained faulty model be(as discussed in THEORY) :

$$y[n] = x[n] * h[n] + n[n]$$

Let's assume that Noise is very very small, and h[n] is invertible(say h'[n])

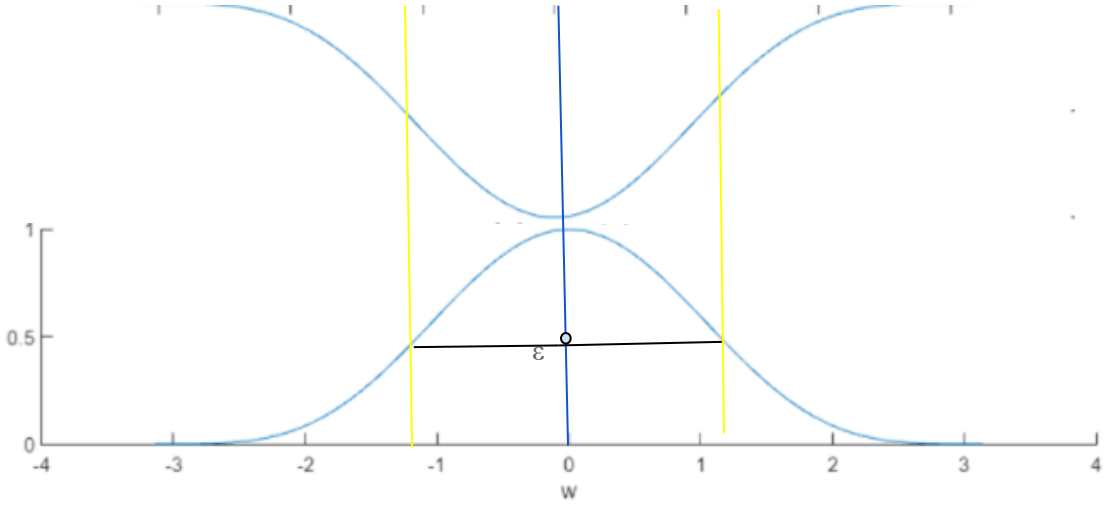
$$y'[n] = h'[n]*y[n]$$

$$y'[n] = x[n]+h'[n] * n[n]$$

In fourier domain

$$Y'(e^{j\Omega}) = X(e^{j\Omega}) + N(e^{j\Omega})/H(e^{j\Omega})$$

The term $N(e^{j\Omega})/H(e^{j\Omega})$ will exist even if the noise is small,as evident at higher frequencies the amplification of new noise (i.e $N(e^{j\Omega})/H(e^{j\Omega})$) will happen as described above.Which I countered by introducing ε as threshold.Now if ε will be close to 1 then the output will have less noise but will be very smoothed out. Which is also evident from the figure, because when it will be close to 1 its bottom part would be uniform and thus giving smoothness.But if it will be close to 0 then the output will have less noise but will be sharp. This exchange betIen smoothness(denoise) and sharpness is being portrayed by these 2 systems. System B , it deblures(adds high frequency components to already existing high frequency components) and then denoises hence making it smooth. Whereas in system A, it denoises (removes high frequencies) and then deblurs, adding a few high frequency components and making it sharper but smooth at the same time.



Therefore, I can say that more appropriate resemblance with true signal will be given by system A. Hence, supporting our observations.

CONCLUSION

Therefore, I can conclude that whenever I need to deblur and then denoise a signal as in this problem, I will follow system A which is , first denoise then deblur. It will give us a much better signal , which inturn will help us to study the true signal which in this case is Temperature recorded by sensors.

REFERENCES/PLATFORMS USED

1. Alan V. Oppenheim , Signals and Systems , 2nd edition
2. MATLAB ONLINE - code writing and executing
3. Google Docs - for writing report
4. Windows Notepad - for writing readme.txt
5. Code Blocks (Google Docs-add on) for formatting code in report
6. Most importantly, lecture videos of Mr. Manish Narwaria

APPENDIX - Readme.txt

```
HOW TO RUN PROGRAM AND OBTAIN THE FIGURE

1. Open the folder "Source code" in matlab
2. Go to Home->Variable->Importdata , a dialog box will appear ,
navigate to current folder and choose data.mat or data.csv
   (if using data.csv only choose numerical value rows)
3. Open "PA_MT21_BB03.m"
4. Go to Editor -> Run ->Run
5. You will get 3 figures. Which figure corresponds to which plot is
mentioned in the figure itself
6. Figure 3 is for understanding purpose, to do that hover over the
plot in figure 3 choose "Zoom In" option, then drag the mouse      cursor
so that the rectangle lies on left side of n=50 mark and the    curve is in
the box
7. Everything Else is in the report attached.
```