REPORT – EEL2010

Programming Assignment

By: Aryan Tiwari (B20EE010)

Dhruv (B20EE016)

Problem Statement:

One of the many applications of Internet of Things (IoT) consists of continuous monitoring of temperature in an area. To that end, several temperature sensors are installed at different locations. These sensors measure and store the recorded value of temperature over time. However, due to limitations of hardware, the sensor memory needs to be cleared periodically and this is done by transmitting the stored values to a base unit. Assume that x[n] denotes the samples of the true value of temperature recorded by a sensor.

However, it is found that the received signal y[n] at the base unit suffers from blur distortions and noise (additive). Hence, the signal y[n] needs to be first processed so that we can recover x[n] from it. Assume that blur happens via a system characterized an impulse response h[n] = (1/16) [1 4 6 4 1] (Assume that the center value of 6/16 corresponds to n = 0). Then, implement the following two approaches to recover the original signal x[n] from distorted signal y[n].

1. First remove noise and then sharpen (deblur). Let the resulting signal be x1[n].

2. First sharpen (deblur) and then remove noise. Let the resulting signal be x2[n].

Now, compare x1[n] and x2[n] with x[n]. What conclusions can you draw from your observations? Also, explain your observations from a theoretical perspective if possible.

Softwares Utilized:

This question requires the knowledge of DTFT, Inverse DTFT, Sampling, Denoising, Deblurring.

We have Used **Python** to solve this assignment. The libraries used are:

- NumPy
- Pandas
- Matplotlib

We have used **Google Colaboratory** as IDE for writing our code.

Approach:

We have defined various functions that are going to work as tools in denoising and deblurring our signal.

Denoising:

This concept is used for noise removal by taking the average of a signal element along with its neighboring elements and using the average as the new value of that signal element.

Usually, this neighborhood can be of 3,5,7 kernel length. In this assignment our signals are 1D and hence, averaging will also be done in 1D.

We have defined an individual function for denoising, keeping the averaging length of 5 for every element.

For the edge elements, padding is done, i.e., for y[0], we have added two y[0] elements in the front so that new (denoised) y[0]'s value is = (y[0]+ y[0]+ y[0]+y[1]+y[2])/5. Similarly last element will be padded as well.

Deblurring:

We know that the original signal x[n] is getting convolved with h[n] to give y[n]. We have to find x[n].

Let X[k], Y[k], H[k] represent discrete time Fourier transforms of x[n], y[n], h[n] respectively. Then,

Y[k]=X[k]H[k] => X[k]=Y[k]/H[k]

We can find the DTFT of y[n] and h[n] using the definition,

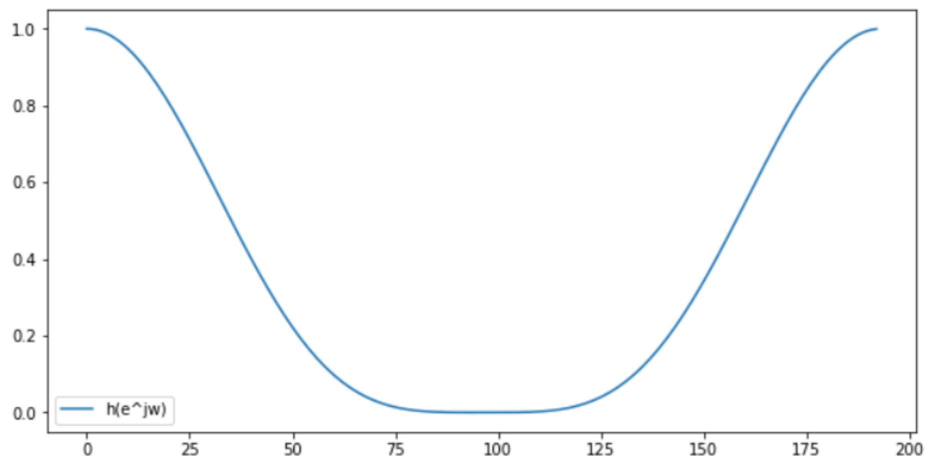
$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

and then define a function for computer to compute it.

We will also use the facts that: h[n]=0 when |n|>2 and

y[n]=0 when n does not belong to {0,193}

The graph of H[k] comes to be:



We find that H[k] has very small values present, values lower than 0.345 have been substituted by 0.345, this is to prevent the shooting up of X[k] in regions of low value of H[k].

After finding X[k] we will define an inverse Fourier transform function to convert it to x[n] and then we will have our deblurred signal.

Calculations:

Using the DTFT Formula,

$$X(e^{j\omega}) \quad = \quad \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

- While calculating DTFT of y[n], n varies from 0 to 193 since y[n]=0 when n does not belong to {0,193}
- While calculating DTFT of h[n], n varies from -2 to 2 since h[n]=0 when |n|>2
- Substituting, $\omega = \frac{2\pi k}{N}$ where N is no. of divisions we are dividing ω into (as DTFT is periodic with F.P. = 2π, we are actually dividing the period of 0 to 2π into N divisions)

For Inverse Fourier Transform,

$$x[n] = \frac{1}{2\pi}\int_{2\pi} X(e^{j\omega})e^{j\omega n}d\omega.$$

Since we can't integrate directly in python, we will define this formula as a Reimann sum.

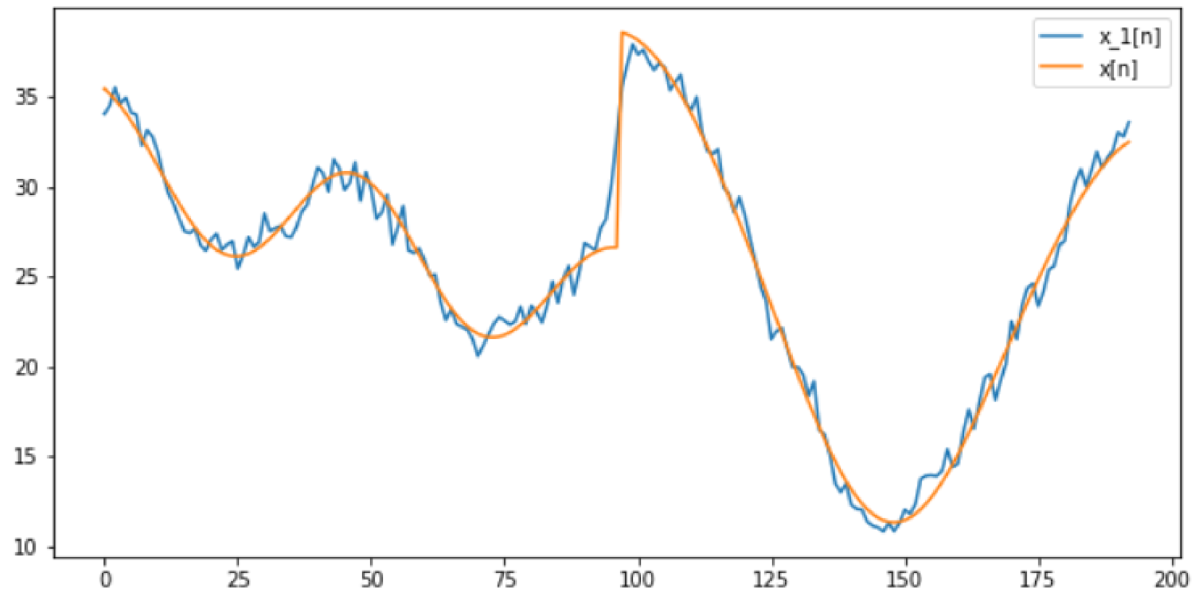
$$x[n] = \frac{1}{N} \int_N X(e^{j\frac{2\pi k}{N}}) e^{j\frac{2\pi k}{N}n} dk \rightarrow x[n] = \frac{1}{N} \sum_{k=0}^{N} X(e^{j\frac{2\pi k}{N}}) e^{j\frac{2\pi k}{N}n}$$

(This is by substituting $\omega = \frac{2\pi k}{N}$ in the IFT formula)

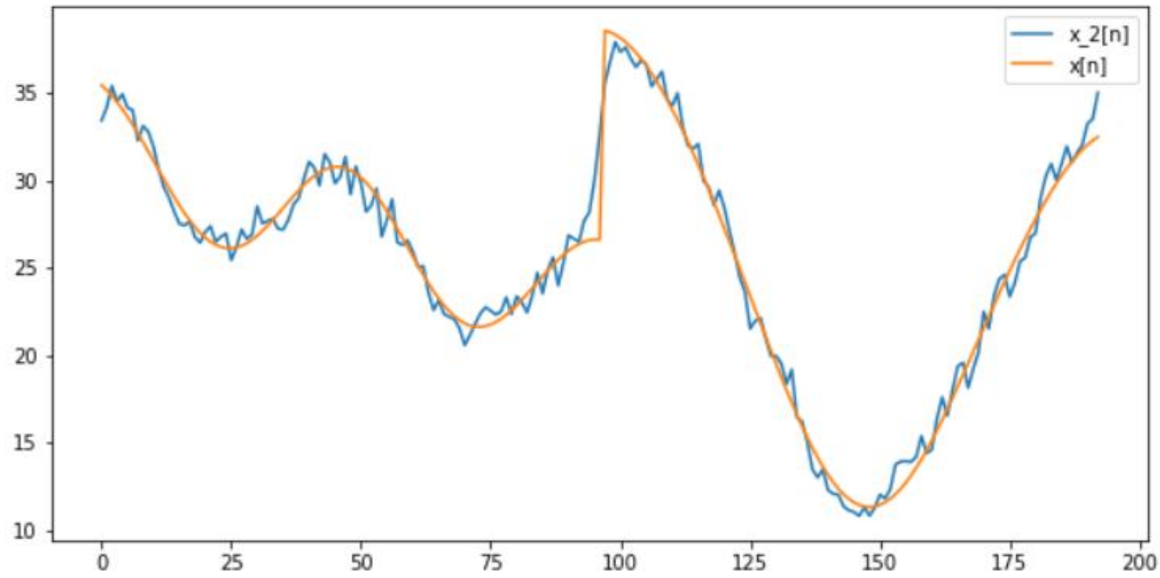
For the first part, we will denoise then deblur and call the output x_1[n] and in the second part, we will do deblur then denoise and call this output x_2[n]

Results:

x_1[n]



x_2[n]



Blue curves are x_1[n] and x_2[n] respectively, while orange is the original x[n]

On calculating the Mean Square Error of both the curves from x[n] we see that:

```
MSE(x_1[n]) = 0.9330448260500126
MSE(x_2[n]) = 0.9823025357392471
```

Conclusions:

From the Graphs we can see that y[n] has been successfully denoised and deblurred in both the given orders.

Further, we see that Mean Square Error of x_2[n] comes out more than Mean Square Error of x_1[n], which implies that x_1[n] is better resembling to x[n].

Hence, Denoising then Deblurring yields better results than deblurring first and then denoising.

Theoretical Explanation:

This can be explained as the signal first got distorted and then the noise got added, i.e.

y[n]=x[n]*h[n] + N , where N stands for noise component.

Therefore, denoising it first and then deblurring it seems more theoretically viable than deblurring it first then denoising it.

Hence, first method is better than the second method.

Contributions:

This assignment was done by equal efforts and participation by both the members along with the guidance from the professor.

Aryan Tiwari (B20EE010):

- Wrote Code for Denoising
- Report preparation
- ReadMe
- Wrote Code for DTFT

Dhruv (B20EE016):

- Wrote Code for Deblurring
- Plotted graphs
- Wrote Code for MSE
- Wrote Code for IFT

We worked together and had discussions on how we were going to approach this assignment.

During these discussions, we learnt a lot, and finally presented the best Approach for this programming assignment.

Thank you.