EEL2010 Programming Assignment Report

Name & Roll No.	1.) B20CH013 Dev Goel 2.) B20CI008 Akshat Jain
Title	- Programming Assignment

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 by an impulse response h[n] = 1/16 [1 4 6 4 1] (assume that the centre 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.

- **Plotting Given Data**:

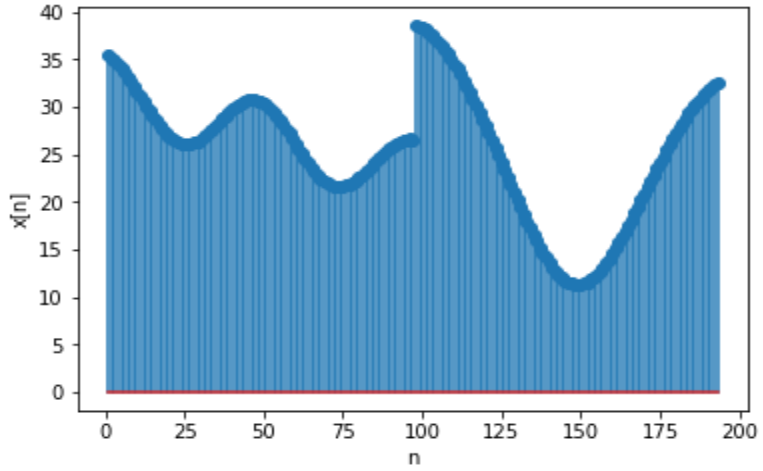

Figure A : Original Signal x[n]

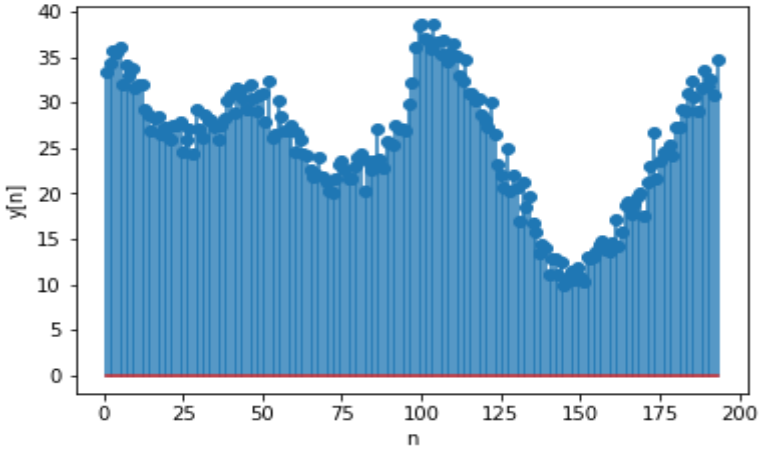

Figure B: Blurred and noise containing y[n]

METHOD - 1 Implementation:

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

Denoising.

We consider a noise present in a signal as y[n] = x[n]*h[n] + N[n], where N[n] denotes the high frequency noise component added to the original signal.

Now, from our understanding of different systems, we know that taking the average of a signal with its neighbouring values removes the high frequency component from the signal and hence denoising the signal.

Hence, we implemented the same logic in our code and got denoised signal as:

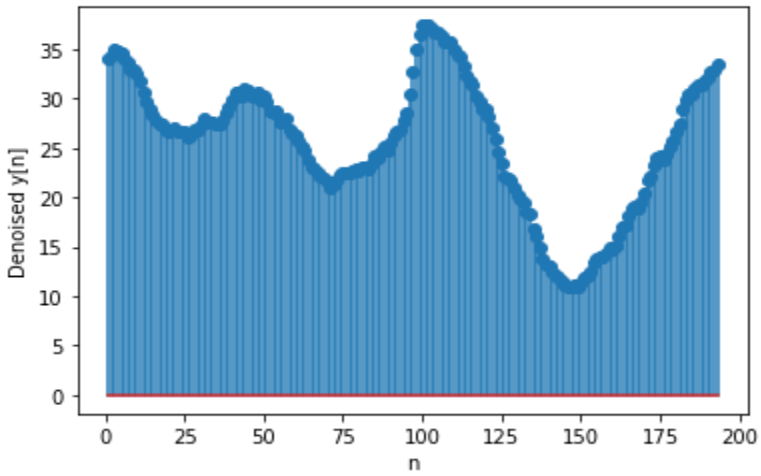

Figure 1(a): Signal after denoising y[n]

Deblurring.

Since, now after denoising the signal our next target was to deblur the signal or sharpen it.

Now, as we noted above, y[n]= x[n]*h[n] + N[n], now since it has been denoised: let the new denoised signal be y'[n]. Where, y'[n]= x[n]*h[n]. (Noise N[n] is removed).

Now taking the Discrete Time Fourier Transform (DTFT) of the above y'[n], we get: y'[e^jw]= x[e^jw].h[e^jw] {By Property we know that Convolution in time domain results in a product in frequency domain}

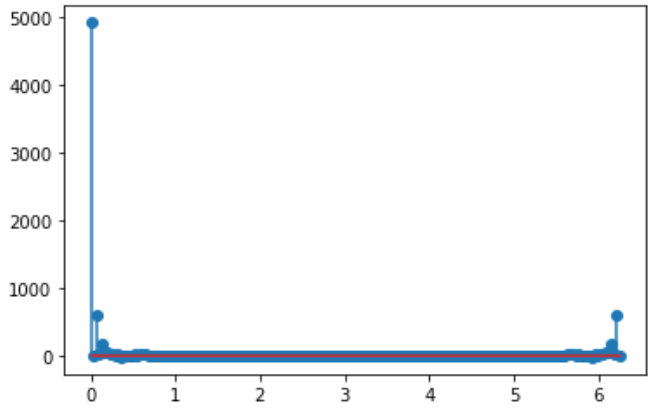

Figure 1(b): DTFT of denoised y'[n]

Here we have taken discrete values of frequency instead of the continuous plot of y'[e^jw] in the frequency domain. Such that, those discrete Wk= 2*pi*k/193 where k goes from 0 to 193 (Concept called Approximation of DTFT).

Similarly, now we take the DTFT of h[n] and obtain h[e^jw] at the same discrete values.

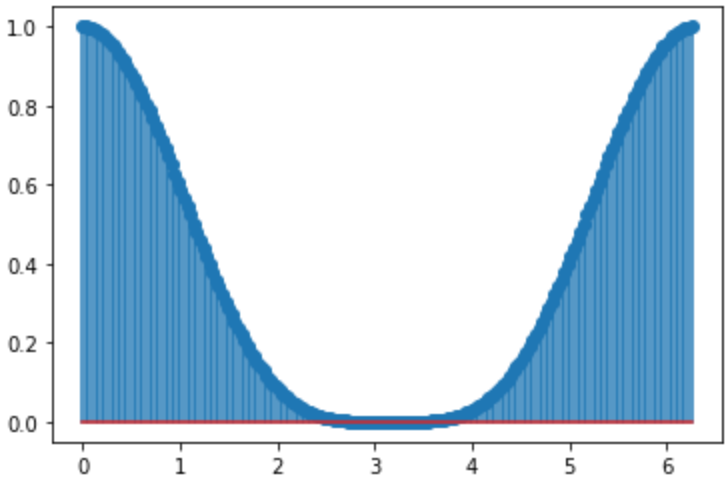

Figure 1(c): The DTFT of h[n]

Now to deblur this and obtain x1[e^jw], we have to divide y'[e^jw] by h[e^jw] at those same frequencies.

But this might create a problem. Since h[e^jw] is very small, having it in the denominator might blow up the value of the denoised signal. Hence, we have to keep a cap, below which we will divide the denoised signal by a constant.

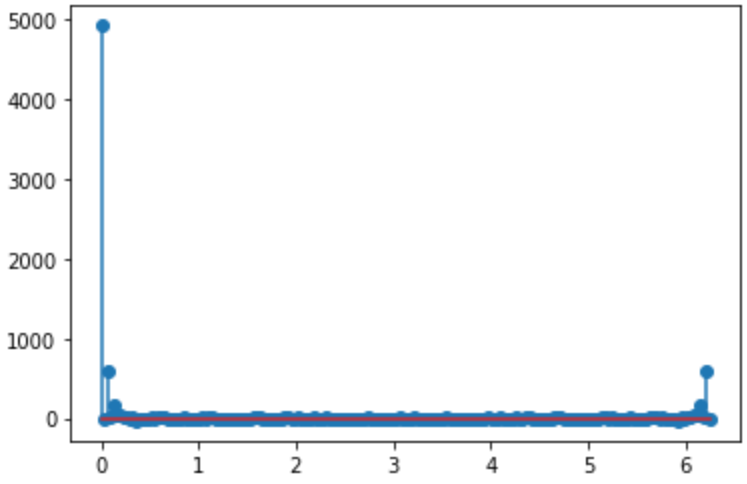

Figure 1(d): Plotting the values of y'[e^jw]/ h[e^jw]

Now finally, to obtain x1[n] from x1[e^jw] we will take the Inverse DTFT of the obtained signal, using the formula:

$X[n] = \Sigma X[e^{jw}].e^{(jwn)}]/N$

Where N= length of the signal (i.e. 193)

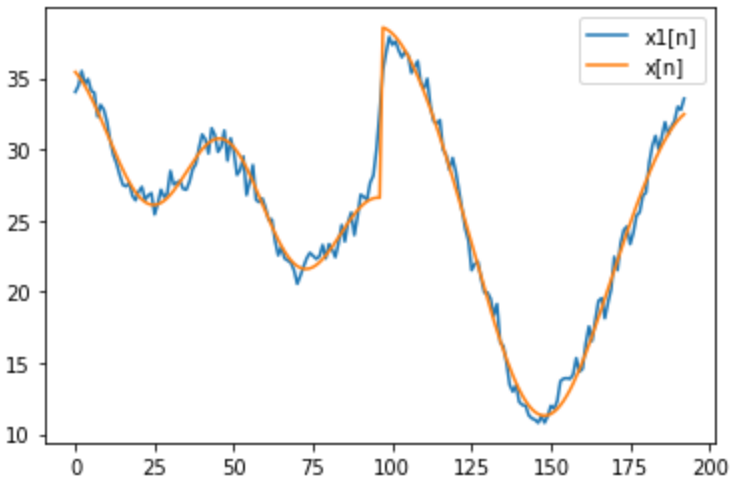

Figure 1(e): Simultaneous Plot of x[n] and x1[n]

METHOD - 2 Implementation:

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

Deblurring

Using the above mentioned Concepts, we now first deblur y[n] instead of denoising it.

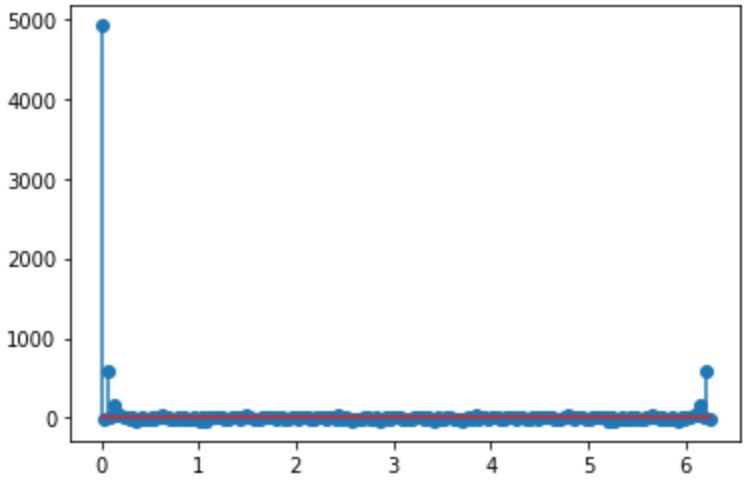

Figure 2(a): Taking the DTFT of y[n]

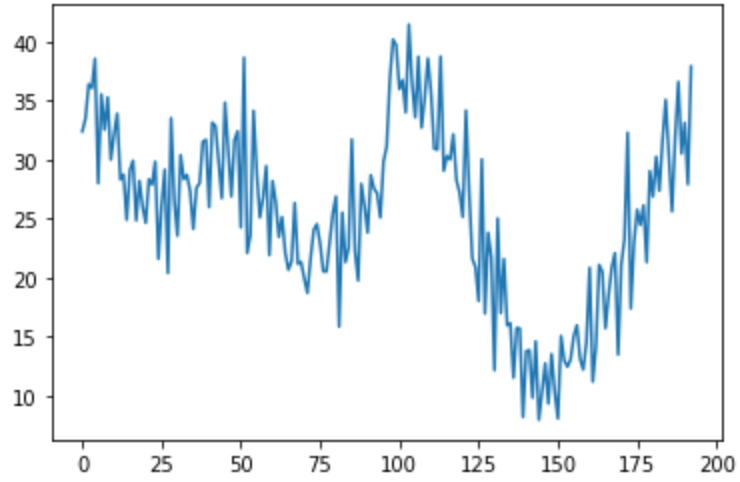

Figure 2(b): y”[n] obtained after deblurring y[n]

Denoising

Again, using the concept of denoising, we remove the high frequency component from y''[n] and thus obtain our final signal x2[n]

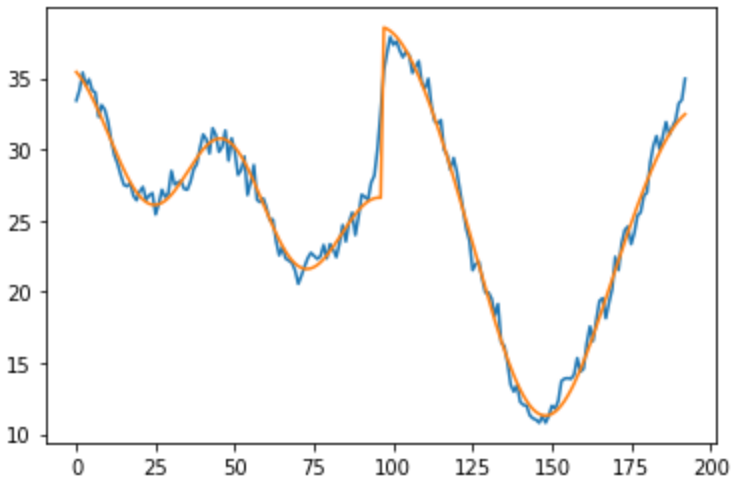

Figure 2(c): Plotting x2[n] (blue) simultaneously with x[n] (orange)

Results, Conclusions and Theoretical explanations

Here in the overall assignment, to re-obtain original signal we have used two approaches, namely:

i) Denoising and then deblurring the signal

ii) Deblurring and then denoising the signal

But which one is better?

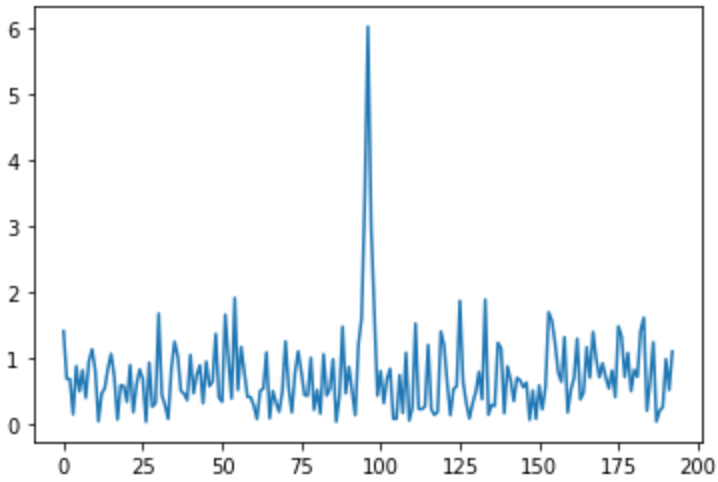

Figure 3(a): This figure represents the the error between x[n] and x1[n] i.e. |x1[n]- x[n]|

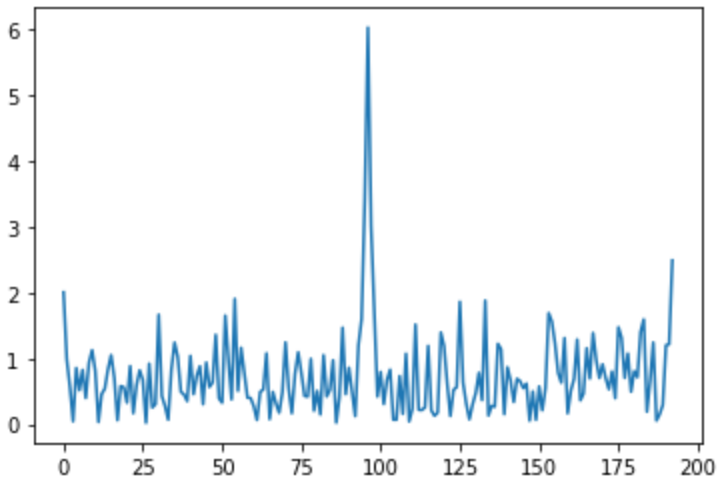

Figure 3(b): This figure represents the the error between x[n] and x2[n] i.e. |x2[n]- x[n]|

- **Here we can see from the graph that x2[n] almost consistently has a higher error**

	x[n]	y[n]	n	x1[n]	x2[n]
0	35.4312	33.3735	1	34.031982	33.428884
1	35.1511	34.3744	2	34.476859	34.160934
2	34.8284	35.7514	3	35.501016	35.391677
3	34.4656	35.5869	4	34.603344	34.510767
4	34.0656	36.0826	5	34.934034	34.920315
..	...	...	...	...	...
188	31.4198	33.5967	189	31.612465	31.570600
189	31.7396	31.7135	190	31.988088	32.031654
190	32.0228	32.6819	191	32.998526	33.220938
191	32.2673	30.9260	192	32.773135	33.481248
192	32.4714	34.7257	193	33.562125	34.957760

Table 1: Values of x[n], x1[n] and x2[n]at different values of n

We can also see from Table 1, that values of x1[n] are qualitatively more closer to x[n].

But now, let's also compare them mathematically by calculating mean square error i.e.

$\sum(xi - x)^2/N$ where, N is the length of the signal.

After computing MSE through a program, we get MSE:

- for approach 1 as 0.9267337063157449
- for approach 2 as 0.9742072885799745

  Which implies that 2nd approach has a higher error than approach 1 or approach 1 is more accurate.

Theoretical Explanation for the results:

As we assumed earlier, y[n]= x[n]*h[n] + N[n]

Or in frequency domain: y[e^jw]= x[e^jw].h[e^jw] + N[e^jw]

Now in the first approach, by denoising first we remove the N[e^jw] part and can directly obtain a close approximation of x[e^jw] by dividing y[e^jw] by h[e^jw]

But in approach 2, while deblurring if we divide the whole equation by h[e^jw] an extra term N[e^jw]/ h[e^jw] will be there, a part of which might still remain in signal when we denoise it later and thus will result in a slight error.

Contribution of each group member:

1. Akshat Jain (B20CI008):

    - Data extraction.
    - Theoretically understood the problem and figured out how to apply concepts of Inverse DTFT.
    - Code implementation and debugging of Denoising and Deblurring the signal.
    - Analysing both the methods by plotting the signals x1[n] and x2[n] with given signal x[n].

2. Dev Goel (B20CH013):

    - Understood the problem theoretically and figured out how to apply the concepts of DTFT and the approach to Denoise the signal to solve this problem.
    - Applied Denoising to the given signal.
    - Code implementation and debugging of Denoising, DTFT and Inverse DTFT.
    - Analysing both the methods by plotting the signals x1[n] and x2[n] with given signal x[n].