Indian Institute of technology Jodhpur

EEL2010-Signals and systems

Report of Programming Assignment

Result, Conclusion and Theoretical Explanation

We are given a signal x[n]. While storing the data in base units, the signal is getting blurred and is suffering from noise (additive).

$$\begin{tikzcd} x[n] \arrow[r] & \boxed{h[n]} \arrow[r] & {} \arrow[r, dash] & y[n] \\ & & \text{noise} \arrow[u] & \end{tikzcd}$$

So,

$$y[n] = x[n] \ast h[n] + noise$$

h[n] = (1/16)[1, 4, 6, 4, 1] , n = (-2 to 2)

Now we have to remove the noise and deblur the signal. It can be done by two approaches

Case 1:: First remove noise and then deblur the denoised y[n]

First of all, we have to define an impulse response for removing noise from y[n]

$$h'[n]= 1/5[1,1,1,1,1,1,1] \quad , \quad n=(-2 \text{ to } 2)$$

We can take an array of 3, 7 or 9 elements but an array of 5 elements is more appropriate for this signal. It is because, if we take an array of less than 5 elements then it will not remove noise efficiently. And if we take an array of length greater than 5 elements, then error in the resultant signal will increase accordingly and the signal will be getting blurred as well.

After removing the noise, we get the denoised signal y'[n] which is basically the convolution of x[n] and h[n].

$\Rightarrow \quad x[n] \ast h[n] = y'[n]$

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

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

So, for it, we computed the DTFT of h[n] and y'[n].

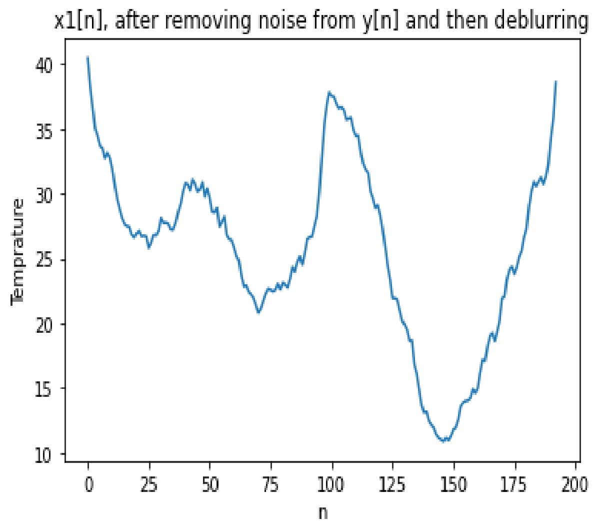
The DTFT of h[n] was tending to 0 at high frequencies, which implies that the DTFT of x[n] would be tending to infinity. It was creating a problem as we can't find the inverse transform when DTFT tends to infinity on some points. To solve this problem we defined a lower bound(0.7) for DTFT of h[n]. Wherever the DTFT goes below this lower bound then it will take value 0.7. We choose this lower bound after running the code again and again for different values of lower bound and we got this appropriate value so the error can be minimised.
After finding the DTFT $X(e^{j\Omega})$ now we have to find inverse fourier transform(IFT)

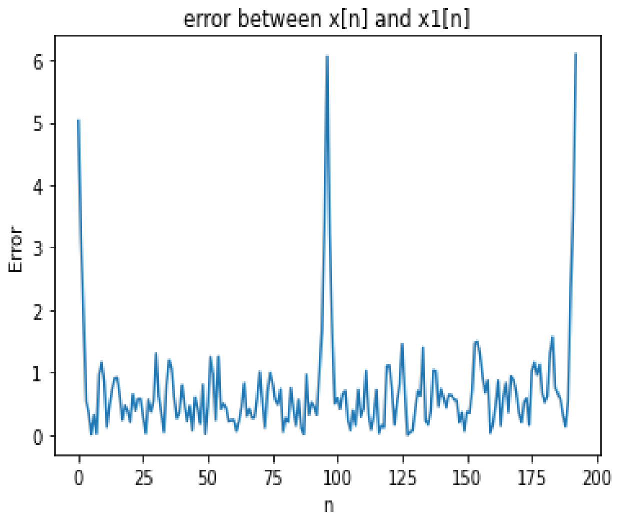
$x[n] = (1/2\pi) \int X(e^{j\Omega}).e^{j\Omega n}.d\Omega$, on some interval 2π

As we can’t implement continuous integration in software, we have to convert this integration into summation as follows

$$x[n] = (1/N)\sum_{k=0}^{k=N-1} (X(e^{j\Omega k}).e^{j\Omega kn}) \quad , \quad \text{where } \Omega k = 2\pi * k/N$$

By applying this summation we got x[n] which is basically x1[n].





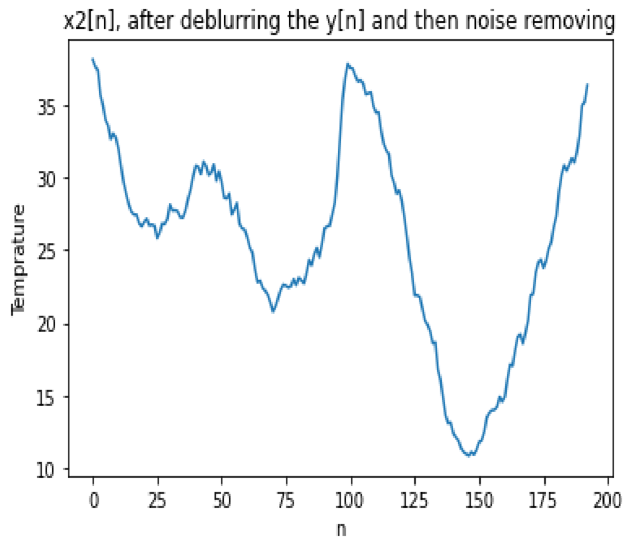
x1[n] is coming out almost the same as x[n] but the original signal is not completely recovered because noise is not removed completely. Average error between actual x[n] and computed x1[n] is 0.7143. Error at the starting and end points is maximum. This can be explained by looking at the convolution of y[n] with h’[n] (noise removing). This convolution is basically averaging the nearby 5 values at any particular ‘n’ (identical from both sides). But for example at the starting point n=0, we only have values at n=0 and on the right side of n=0, so it will take an average of only 3 values instead of 5 values. Same problem arises with the end point, where we only have values at n=192 and on the left side of n=192.

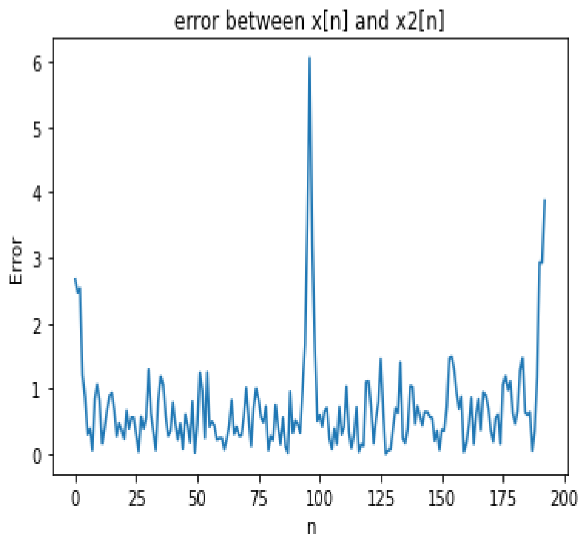
Also, we are getting a comparatively larger error at the point where there is a sharp jump in signal x[n]. This also can be explained by convolution of y[n] with h’[n]. It is because after the convolution, the resultant signal will pass from almost the middle point of the sharp jump, therefore, it will create error between actual signal and resultant signal.

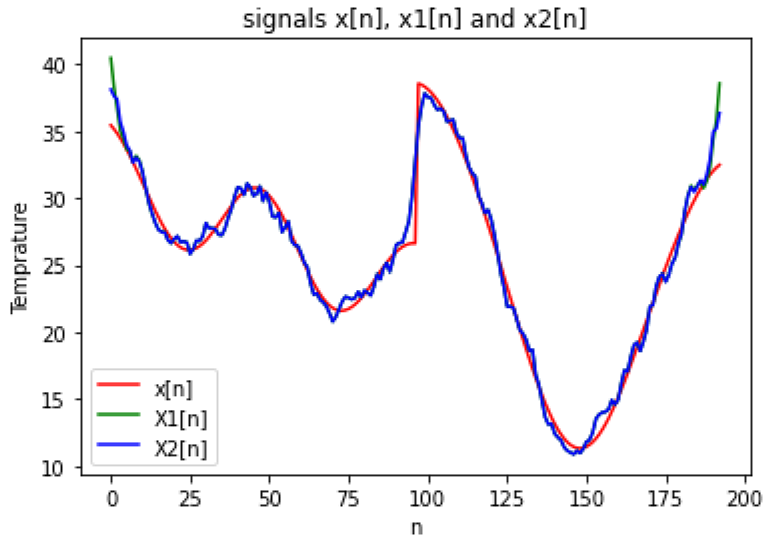
Case 2:: First deblure the y[n] and then remove noise from deblurred y[n]

For deblurring the y[n] we have applied the same process like above.

After the deblurring we have computed the inverse transform of the deblurred signal and then we find convolution of the deblurred signal with h'[n] to get the x2[n].
x2[n] is also coming out almost the same as x[n]. Average error between actual signal x[n] and computed signal x2[n] is 0.6993.



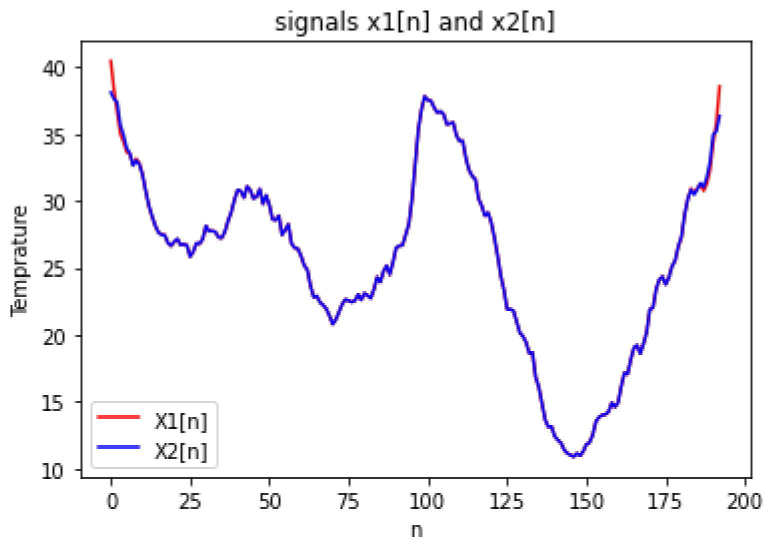




By this plot we can see that x[n], x1[n] and x2[n] are almost same

Comparison between x1[n] and x2[n]:-

We have plotted x1[n] and x2[n] on same plot as follows



We can see that x1[n] and x2[n] are overlapping each other, there is some error at the starting point and ending point of the signal, but that error is occurring because of the previously mentioned reason, that while removing noise, averaging was not done finely at starting and ending point.

So we can conclude that x1[n] and x2[n] are coming exactly same, still there is some error but that is negligible

We can explain this theoretically also

Case 1

$\Rightarrow$ $x[n] \ast h[n] + k[n] = y[n]$, *(k[n] is noise)*

$\Rightarrow$ $x[n] \ast h[n] \ast h'[n] + k[n] \ast h'[n] = y[n] \ast h'[n]$ *(removing noise)*

$\Rightarrow$ $X(e^{j\Omega}).H(e^{j\Omega}).H'(e^{j\Omega}) + K(e^{j\Omega}).H'(e^{j\Omega}) = Y(e^{j\Omega}).H'(e^{j\Omega})$

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

$\Rightarrow$ $\text{IFT}(X(e^{j\Omega}) + K(e^{j\Omega})/|H(e^{j\Omega})|) = \text{IFT}(Y(e^{j\Omega})/|H(e^{j\Omega})|) = x1[n]$

Case 2

$\Rightarrow \quad x[n] \ast h[n] + k[n] = y[n] \qquad k[n] \text{ is noise}$

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

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

$\Rightarrow \quad \mathrm{IFT}(\, X(e^{j\Omega}) + K(e^{j\Omega})/|H(e^{j\Omega})|) = \mathrm{IFT}(Y(e^{j\Omega})/|H(e^{j\Omega})|\,)$

$\Rightarrow \quad \mathrm{IFT}(\, X(e^{j\Omega}) + K(e^{j\Omega})/|H(e^{j\Omega})|) \ast h'[n] = \mathrm{IFT}(Y(e^{j\Omega})/|H(e^{j\Omega}|) \ast h'[n] = x2[n]$

As we can see that convolution of IFT($Y(e^{j\Omega})/|H(e^{j\Omega})|$) with h’[n] will not affect significantly.
This is because, convolution will mostly remove high frequencies but

IFT($Y(e^{j\Omega})/|H(e^{j\Omega})|$) (which is equal to x1[n]),

only has low frequencies. Hence x1[n] and x2[n] are same

Conclusion

So, x1[n] and x2[n] will be approximately equal, with very less average error (=0.061).
We are getting the same result for both approaches so any one of them can be used to process the signal.