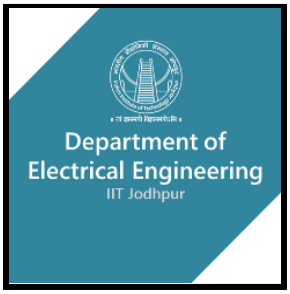
EEL2010 – Signal and Systems



Programming Assignment - Report

TEAM MEMBERS :

- HARSHITA GUPTA - B20CS018
- KRISHI PATEL - B20EE030

Bachelor of Technology
(UNDERGRADUATE - FIRST YEAR)

in

Indian Institute of Technology Jodhpur

April 2021

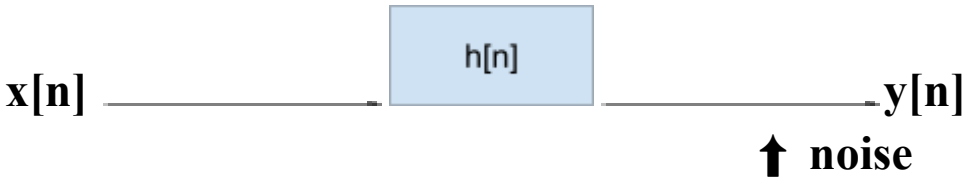
Introduction:

- One of the numerous applications of Internet of Things (IoT) comprises nonstop checking of temperature in a zone. To that conclusion, a few temperature sensors are introduced in diverse areas. These sensors measure and store the recorded esteem of temperature over time. In any case, due to restrictions of hardware, the sensor memory must be cleared occasionally, which can be done by transmitting the put-away values to a base unit.

➔ **Based upon the above process :**

➢ x[n] denotes the samples of the true value of temperature recorded by a sensor and is in the form of a continuous signal.

➢ y[n] denotes the received signal at the base unit which suffers from blur distortions and noise(additive). [Continuous time signal get converted into Discrete-Time Signal]



So y[n] = x[n] ✳ h[n] + noise

➢ Hence, the signal y[n] is being processed first so that x[n] can be recovered from it.

➢Implemented by following two approaches to recover the original signal x[n] from distorted signal y[n]:

- First, remove noise and then sharpen (deblur). Let the resulting signal be x1[n].
- First, sharpen (deblur) and then remove noise. Let the resulting signal be x2[n].

Approach:

- **Denoise:** **Denoising is any signal processing method that reconstructs a signal from a noisy one. Its goal is to remove noise and preserve useful information.**

  ➔ Denoising is done by defining a function that uses convolution(by mirror padding) to denoise a signal using the denoising kernel h'[n] = [⅕,⅕,⅕,⅕,⅕].

- **Deblur:** **The removal of blur from a digital image, or of noise from a digital signal.**

  ➔ For deblur, a function that calculates the DTFT of a sampled signal is defined that returns 193 samples.

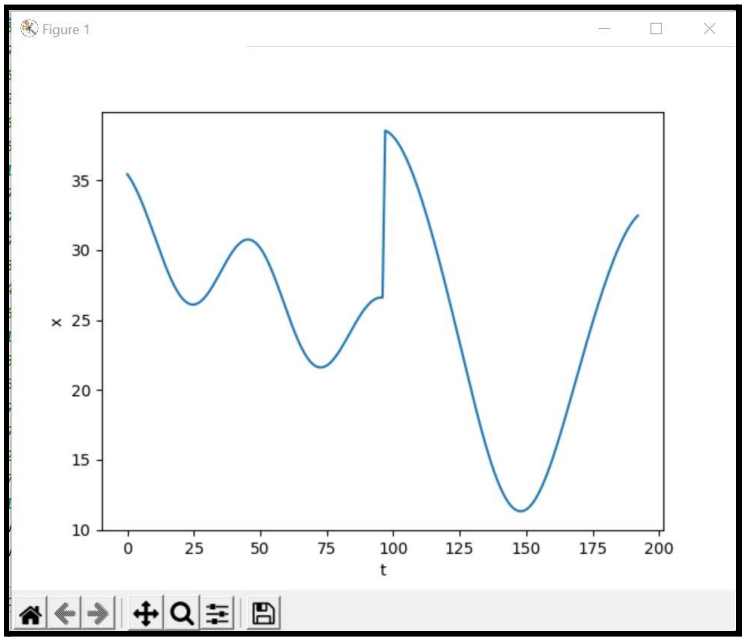
  ➔ Then a function that calculates the discrete inverse Fourier transform of a signal and returns 193 samples is defined.

  ➔ We know that h[0]=6/16 so to compensate for the timeshift we use the time shift property of DTFT. As we can see the DTFT consists of many terms that tend to be 0 so to avoid division by 0 we can clip the lower bound of the original_DTFT_of_h {it is the name of a variable used in the code to calculate DTFT(h[n])}.

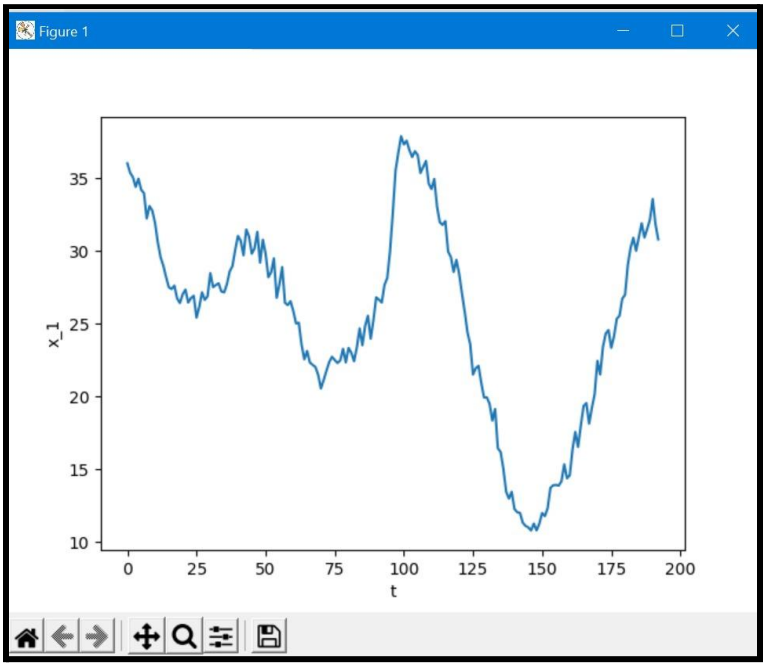
  ➔ And by hit and trial, we can find that if we clip the lower bound to 0.35 then it is sufficient to deblur the signal.

Result:

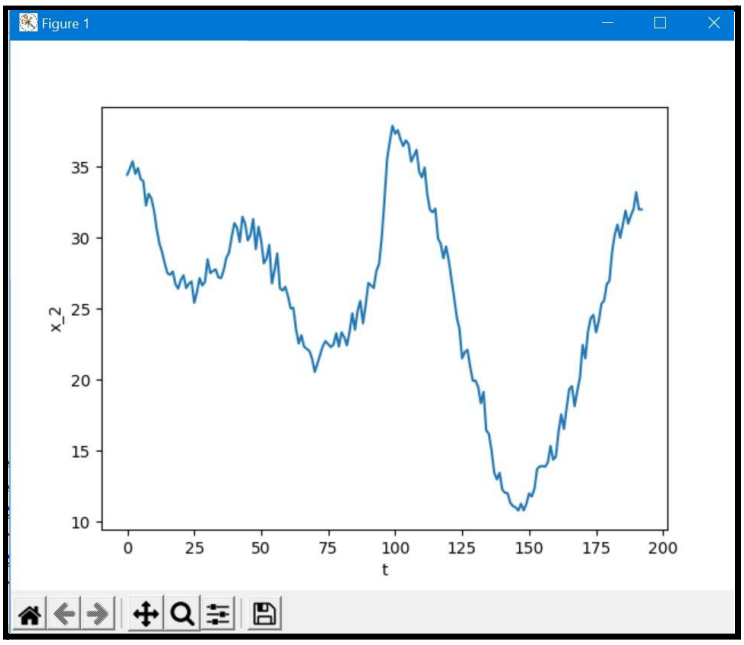
The Plot of X w.r.t time-



The plot of X_1 w.r.t time-



The plot of X_2 w.r.t time-



The average error in the signal x_1 wrt x is: 0.7199673575129534

The average error in the signal x_2 wrt x is: 0.7130113989637309

Theoretical Explanations:

Case 1:: First remove noise and then Deblur

➔First of all, we have defined an impulse response(convolution) for removing noise from y[n].

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

x[n] ✳ h[n] = y'[n]

X(e^(jΩ)).H(e^(jΩ)) = Y'(e^(jΩ))

X(e^(jΩ)) = Y'(e^(jΩ))/H(e^(jΩ))

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

➔The DTFT of h[n] was tending to 0 at mediocre frequencies, hence the DTFT of x[n] was 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(0.35) for DTFT of h[n]. Wherever the DTFT goes below this lower bound then it will take a value of 0.35. We choose this lower bound after running the code again and again for different values of the lower bound and we got this appropriate value so the error can be minimized.

➔After finding the DTFT X(e^(jΩ)) now we have to find inverse fourier transform(IFT)

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

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

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

where $\Omega k = 2\pi * k/N$

➔By applying this summation we got x[n] which is x_1[n].

➔x_1[n] is coming out almost the same as x[n]. Average error between actual x[n] and computed x_1[n] is 0.7199673575129534. Error at the starting and endpoints is maximum. This can be explained by looking at the convolution of y[n] with h'[n] (noise removing). This

happens as we encounter a jump discontinuity in the signal at both ends.

➔Also, we are getting errors comparatively large at the point where there is a sharp jump in signal x[n]. This also can be explained by the convolution of y[n] and h'[n].

Case 2:: First deblur the y[n] and then remove noise

➔For deblurring the y[n] we have applied the same process as 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 x_2[n].

➔Thus we can conclude that no matter what order we follow for deblurring and denoising, we get the same result(signal), with nearly equal average errors.

Contribution:

- ❖ Both the team members gave equal participation for this assignment.
- ❖ We conducted meets at regular intervals to come up with the approach to solve the problem and discuss new and more efficient ways to write the algorithm to do so.
- ❖ After finalizing everything KRISHI PATEL(B20EE030) wrote the code, and HARSHITA GUPTA(B20CS018) made the report and the readme file for the same.