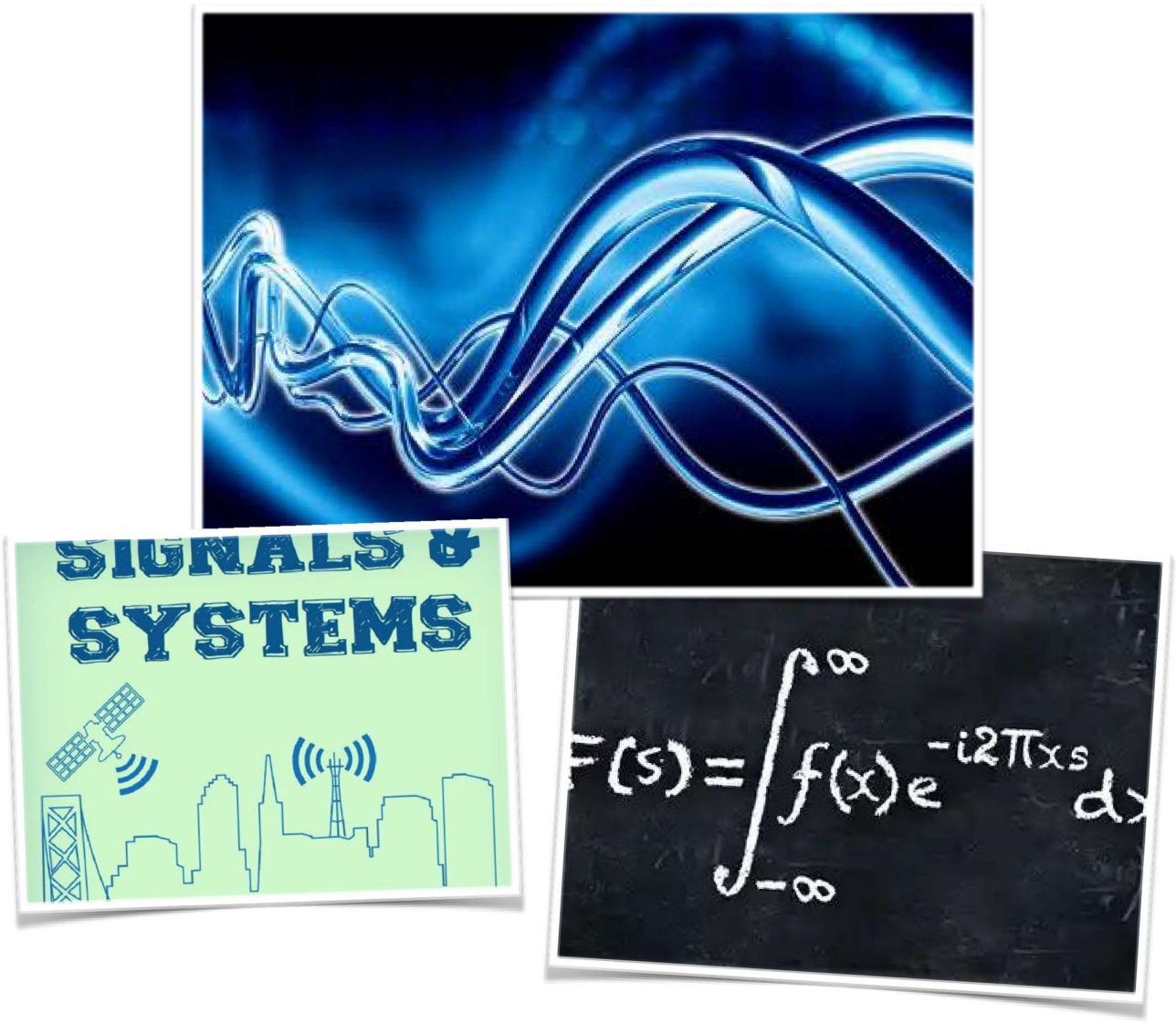
Signals and Systems



Aim	To get original signal from a given distorted signal.

Submitted to:
Dr. Manish Narwaria

Operations:

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].

3. Then, compare x1[n] and x2[n] with [n] .

Given Kernel : h[n] = [1/16, 4/16, 6/16, 4/16, 1/16]

Programming Language Used: Python3

Required Libraries :

- Matplotlib - for plotting the graphs
- Numpy - many operations required
- CSV - for minor uses only

Here,

>>Y[n] = X[n]*h[n] + d[n]

>> Y2[n] = X[n]*h[n]

>> F[Y] = F[X] x F[h]

>> F[X] = F[Y] / F[h]

>> X[n] = InvFT (F[X])

Main Idea :

Here we are going to use very simple basic concepts learned in this course like 2-D signals, Convolution , Fourier Transform , Inverse Fourier Transform and Sampling etc. to solve this problem.

As we know distorted signal is combination of blurred and noisy signal. In this problem

Y[n] = X[n]*h[n] + d[n]

Where h[n] is given kernel which has been used for blurring and d[n] is additive noise.

Now clearly we can first denoise the given signal using convolution technique or by averaging the consecutive terms of the signal. For our purpose we'll use averaging here. Then once we have this denoised signal we have to find such signal that gives Y[n] when convolved with h[n].

Clearly it is problem of deconvolution and as we know convolution is just multiplication in frequency

domain we'll first calculate Fourier transforms of Y[n] and h[n] and then by dividing them we get corresponding

Fourier transform of X[n]. But here we have different dimensions for Y[n] and h[n] and thus we can not directly divide their Fourier transforms so we sample both signals to a certain value and then by dividing both Fourier transforms we find F[X] . Then we can easily find X[n] by using Inverse Fourier transform technique. And thus we get our estimation for original signal.Then for second part we'll just reverse the order means we'll do deconvolution first and then from that debarred signal we'll do removal of noise.

NOTE: When dividing Fourier transforms F[Y] and F[h] we can face the case where F[h] is nearly zero and therefore we clip the signal to avoid that situation.

Theory

- Convolution :

Convolution is a mathematical way of combining two signals to form a third signal. It is the single most important technique in Digital Signal processing.

Formula

$$(f*g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\tau'$$

$(f*g)(t)$ = functions that are being convoluted

t = real number variable of functions f and g

$g(\tau)$ = convolution of the function f(t)

τ' = first derivative of g(tau) function

- Fourier transform :

A Fourier transform is a mathematical transform that decomposes functions depending on space or time into functions depending on spatial or temporal frequency, such as the expression of a musical chord in terms of the volumes and frequencies of its constituent notes.

Time Duration		
Finite	Infinite	
Discrete FT (DFT)	Discrete Time FT (DTFT)	discr.
$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\omega_k n}$	$X(\omega) = \sum_{n=-\infty}^{+\infty} x(n)e^{-j\omega n}$	time
$k = 0, 1, \ldots, N-1$	$\omega \in (-\pi, +\pi)$	n
Fourier Series (FS)	Fourier Transform (FT)	cont.
$X(k) = \int_0^P x(t)e^{-j\omega_k t}dt$	$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt$	time
$k = -\infty, \ldots, +\infty$	$\omega \in (-\infty, +\infty)$	t
discrete freq. k	continuous freq. ω	

- Inverse Fourier transform :

- Fourier Transform of $x(t)$: $\mathcal{F}[x(t)]$ or $X(w)$:

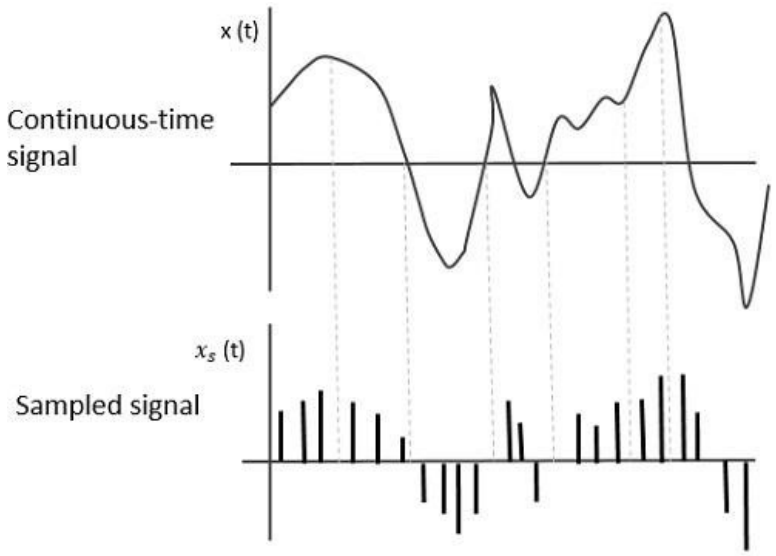
$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$$

- Inverse Fourier Transform of $X(w)$: $\mathcal{F}^{-1}[X(w)]$:

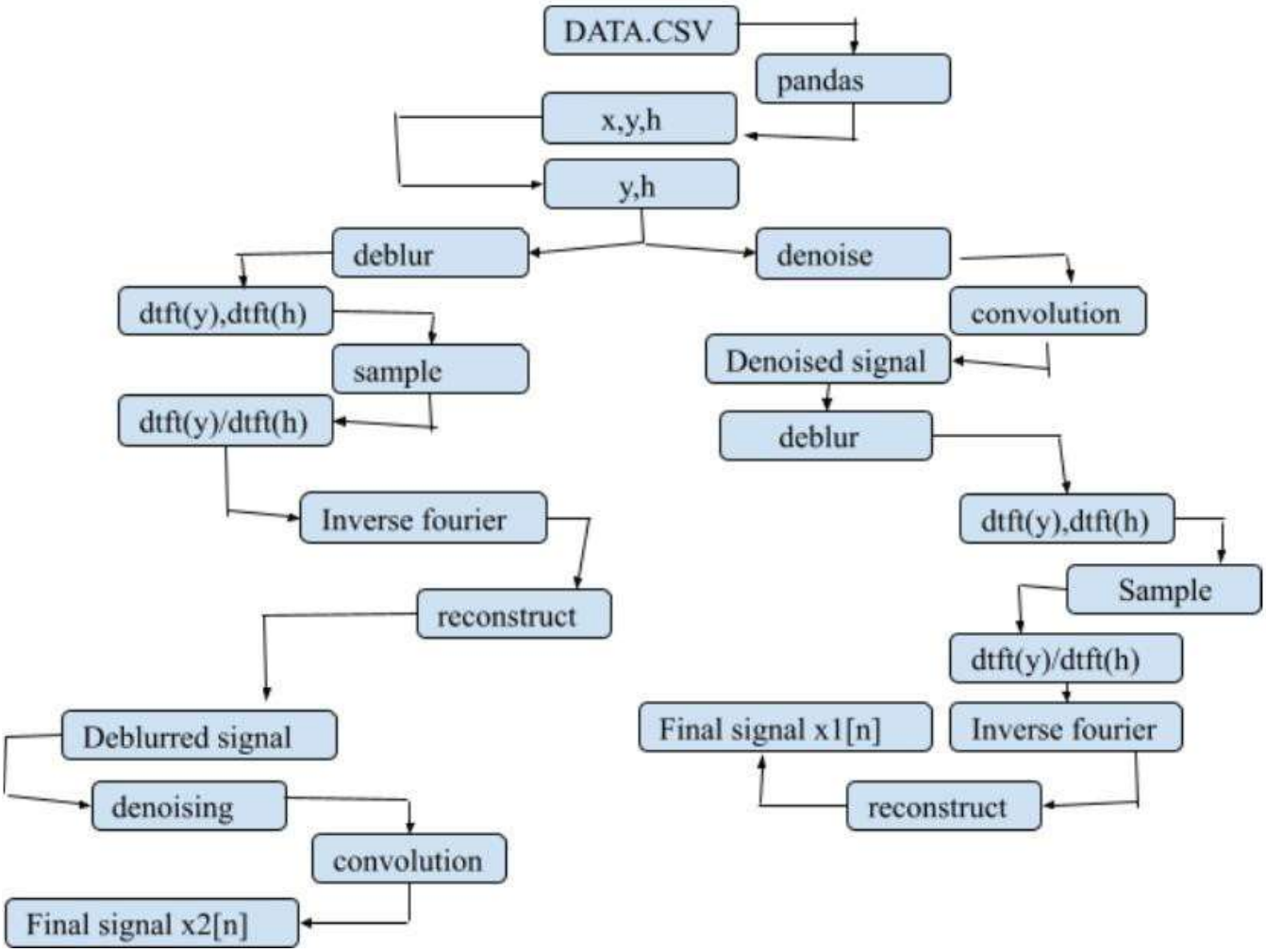
$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}d\omega$$

- Sampling :

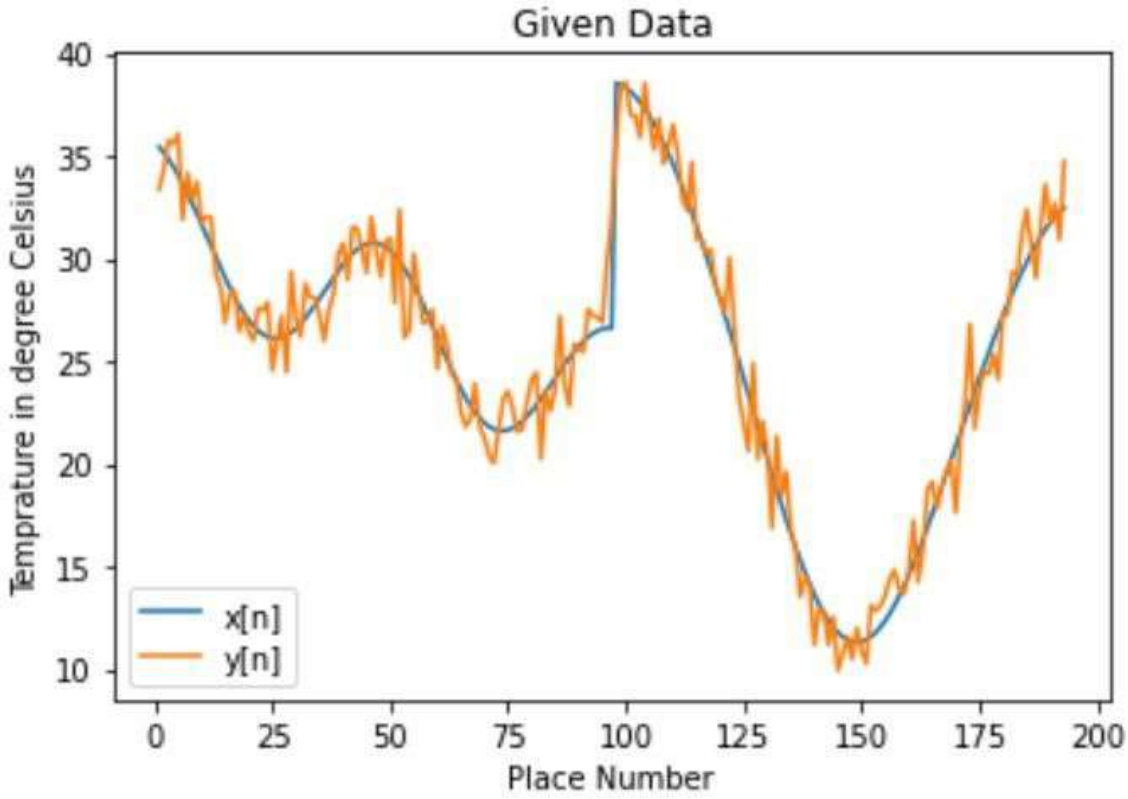
In signal processing, sampling is the reduction of a continuous-time signal to a discrete-time signal.A sample is a value or set of values at a point in time and/or space.

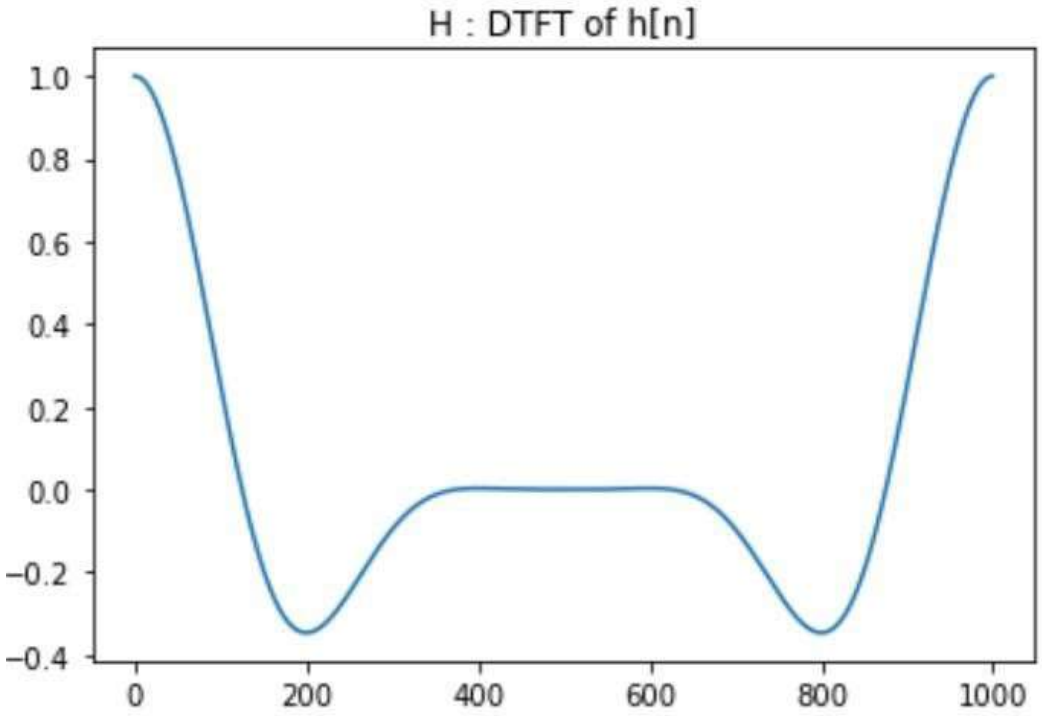


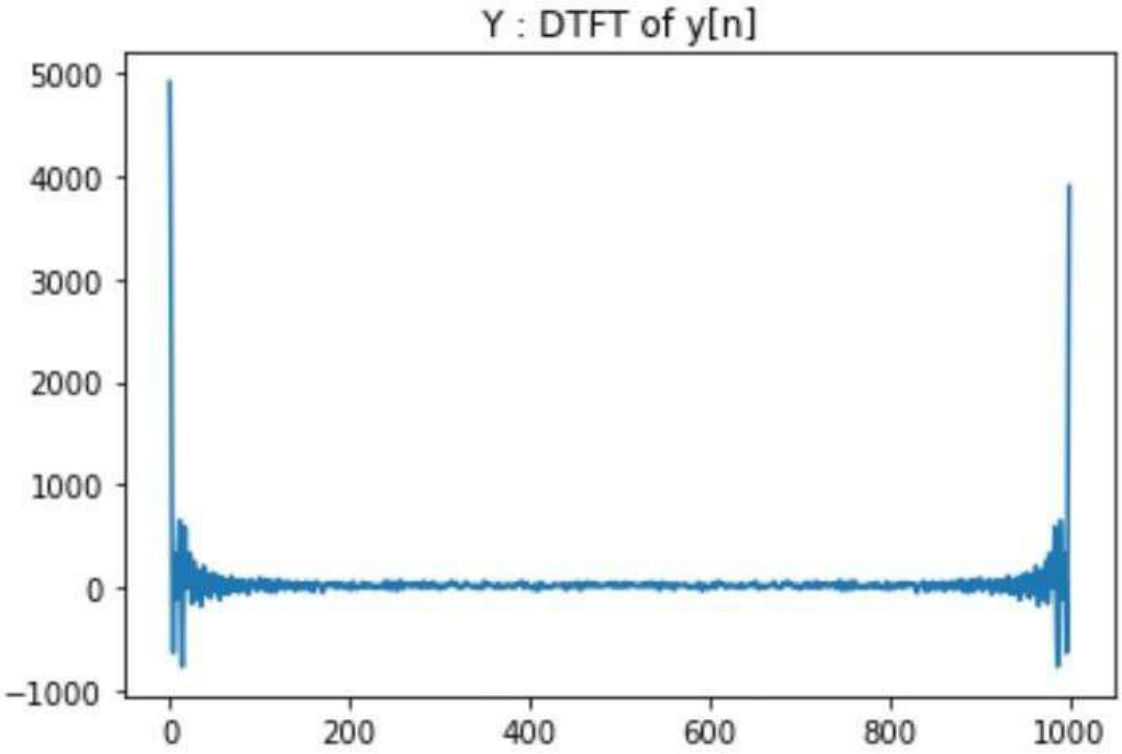
Flow Chart :

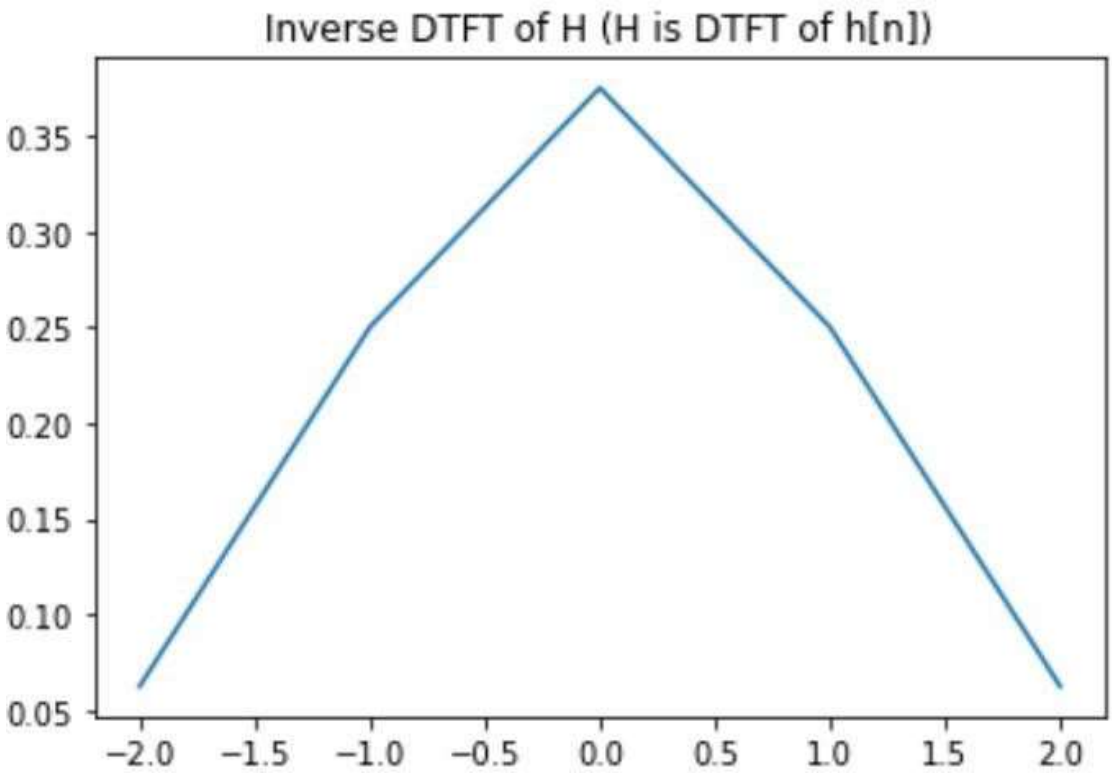

DATA.CSV
pandas
x,y,h
y,h
deblur
denoise
dtft(y),dtft(h)
sample
dtft(y)/dtft(h)
Inverse fourier
reconstruct
Deblurred signal
denoising
convolution
Final signal x2[n]
convolution
Denoised signal
deblur
dtft(y),dtft(h)
Sample
dtft(y)/dtft(h)
Inverse fourier
reconstruct
Final signal x1[n]

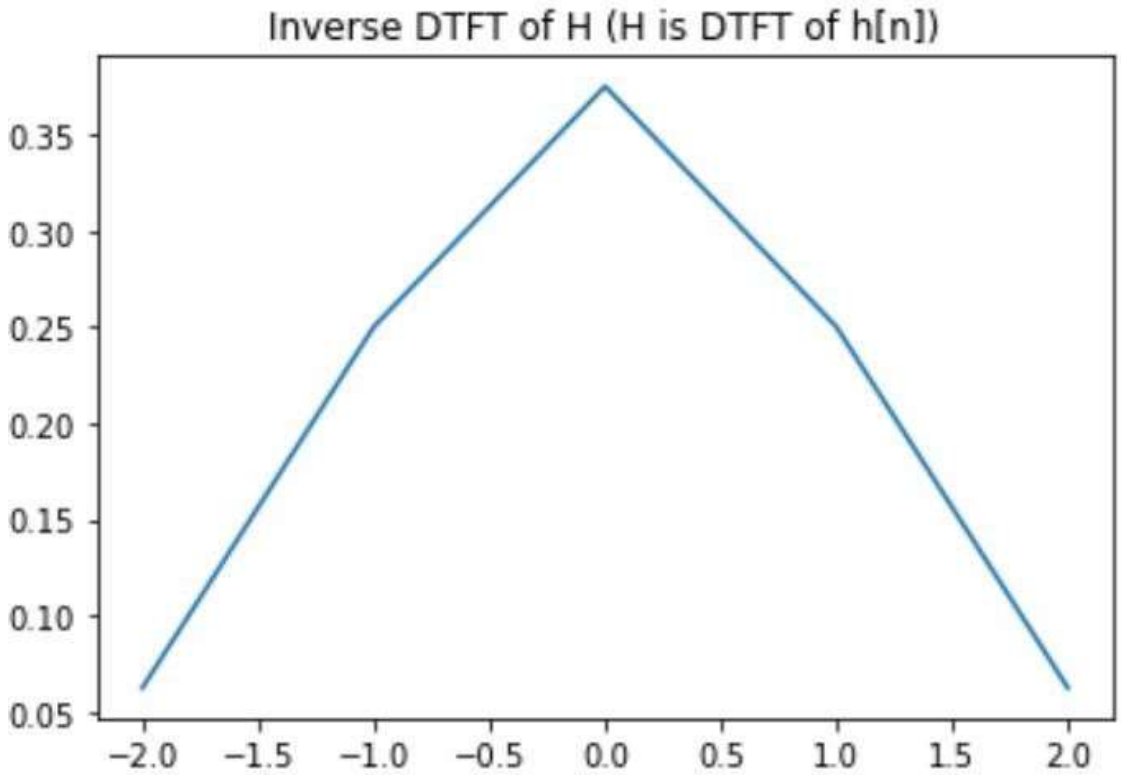
Plotting Graphs:

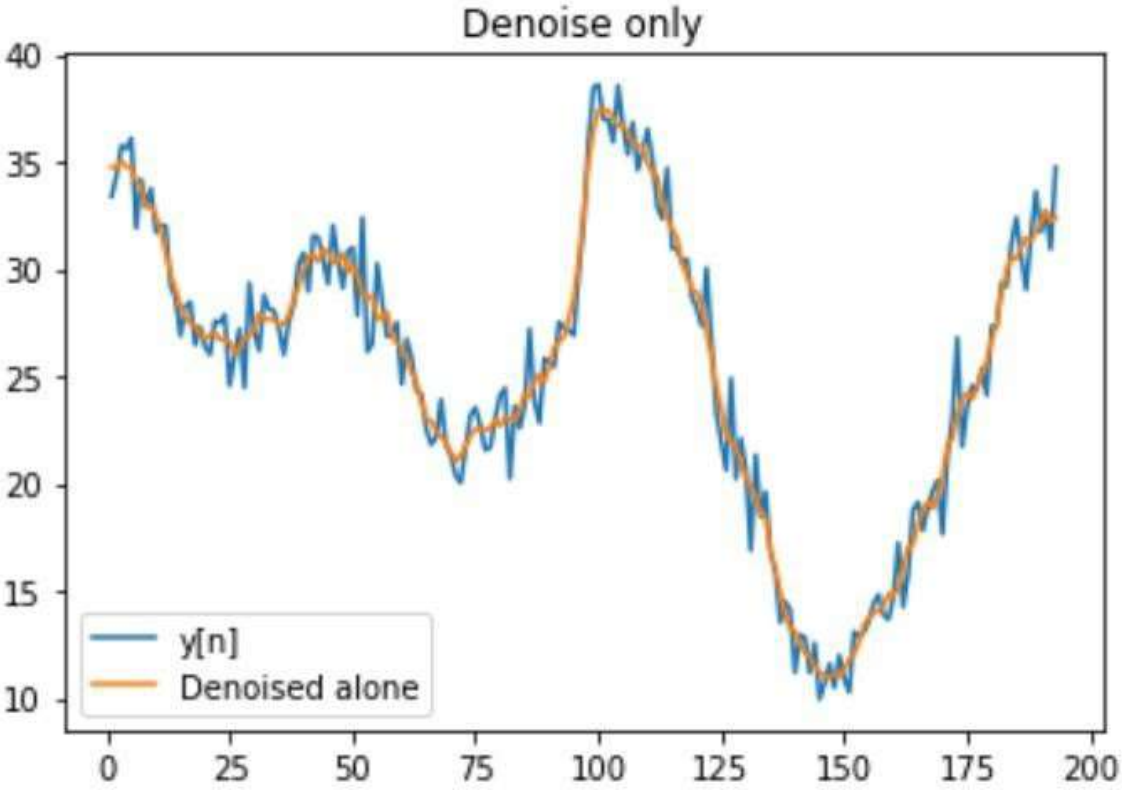


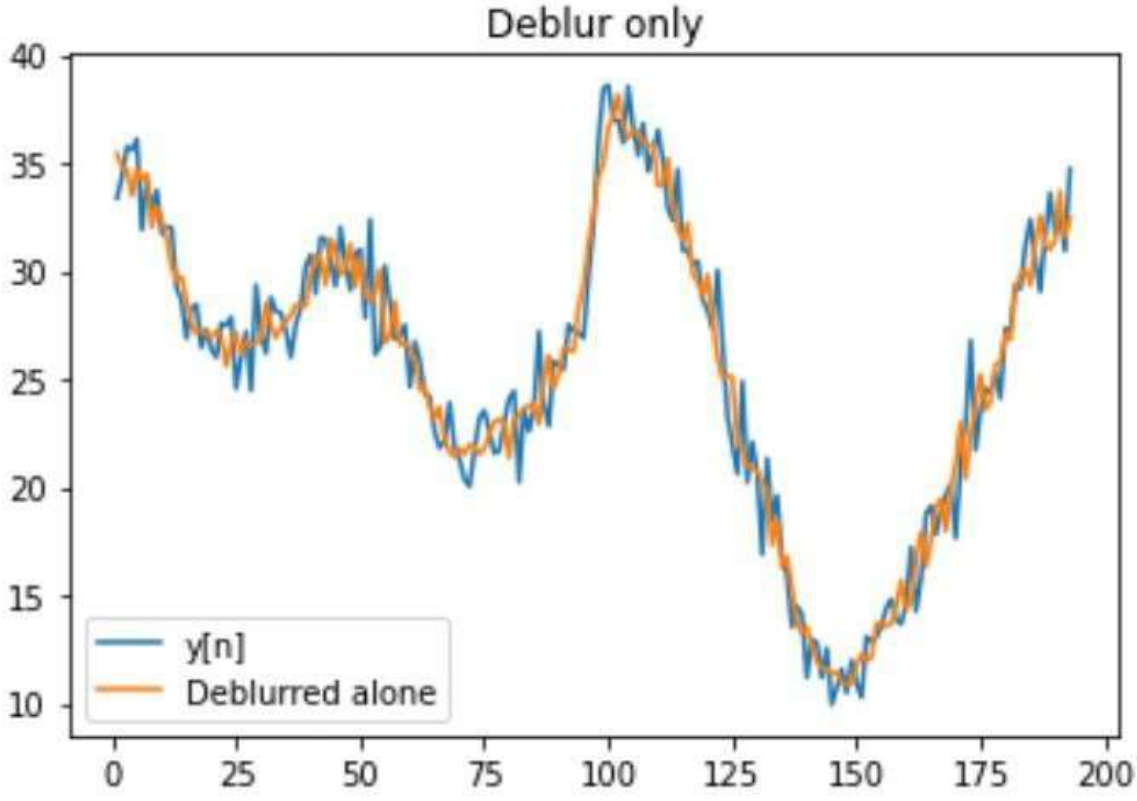


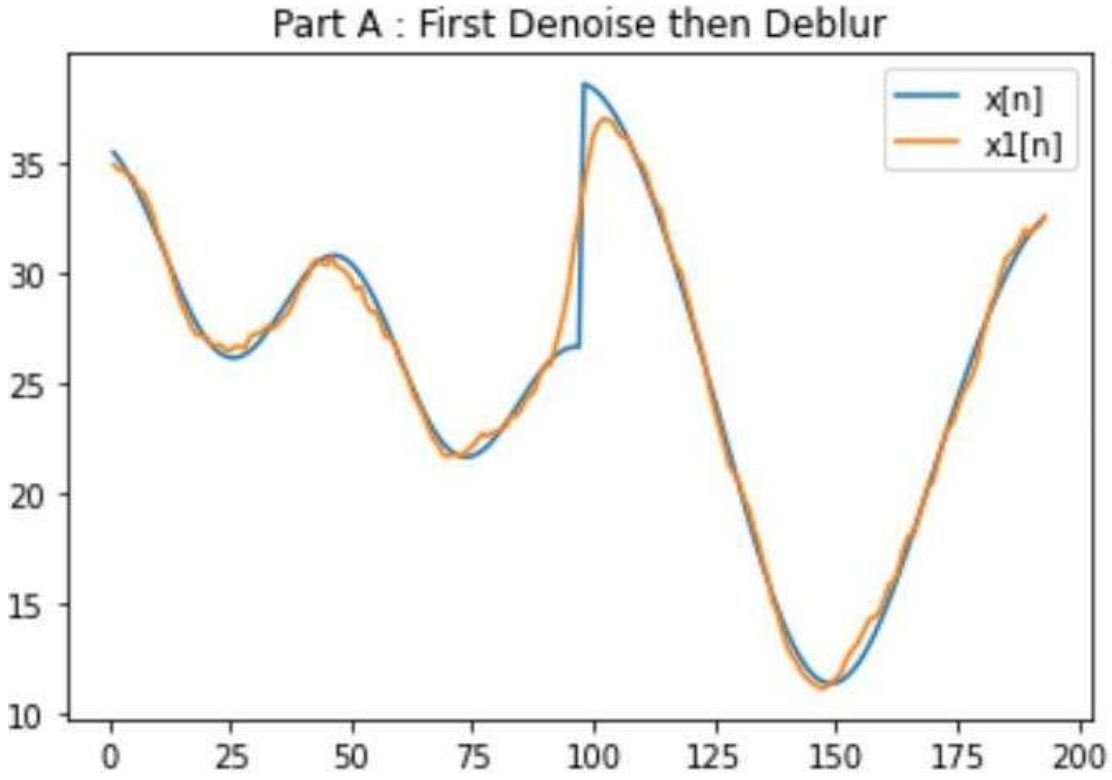

Y : DTFT of y[n]
5000
4000
3000
2000
1000
0
−1000
0
200
400
600
800
1000

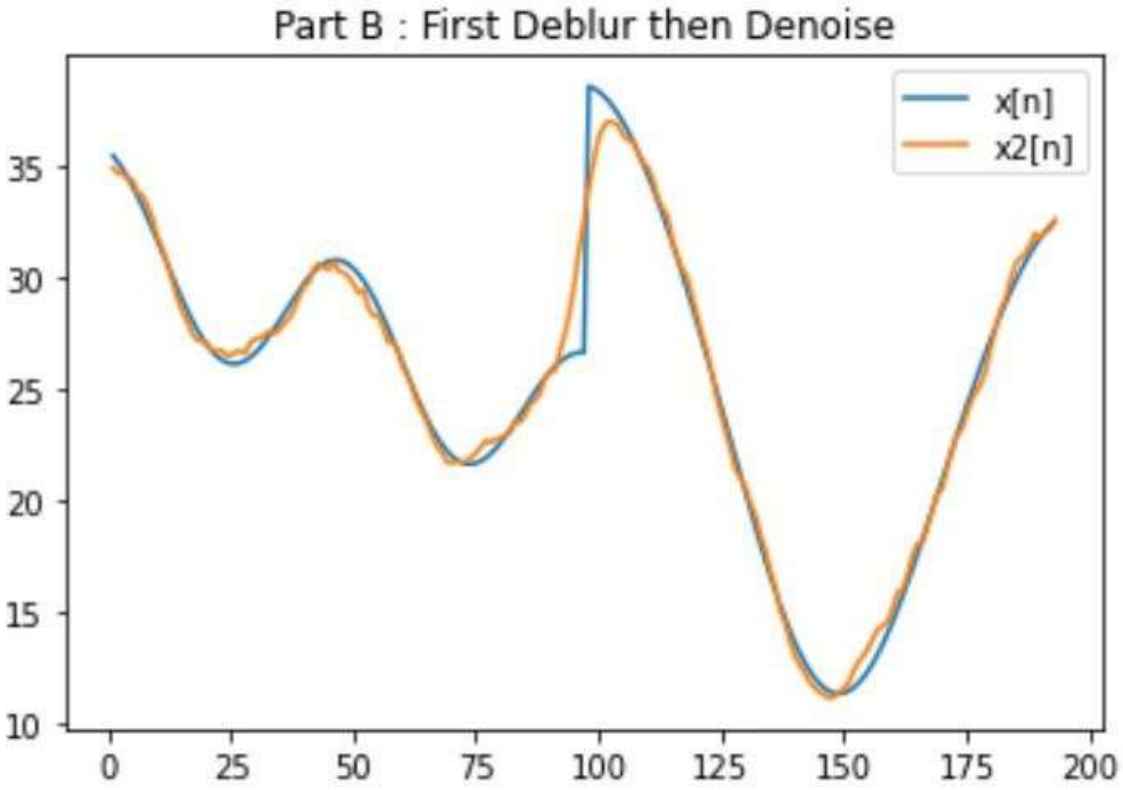

Inverse DTFT of H (H is DTFT of h[n])
0.35
0.30
0.25
0.20
0.15
0.10
0.05
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Inverse DTFT of H (H is DTFT of h[n])
0.35
0.30
0.25
0.20
0.15
0.10
0.05
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Denoise only
40
35
30
25
20
15
10
y[n]
Denoised alone
0
25
50
75
100
125
150
175
200


Deblur only
y[n]
Deblurred alone
40
35
30
25
20
15
10
0
25
50
75
100
125
150
175
200


Part A : First Denoise then Deblur
x[n]
x1[n]
35
30
25
20
15
10
0
25
50
75
100
125
150
175
200



Comparison:

- Sum Square Difference (SSD) of x1[n] with respect to x[n] : 144.34195453467495
- Sum Square Difference (SSD) of x2[n] with respect to x[n] : 143.41910330051917

Conclusions :

After calculations standard square difference of both X1[n] and X2[n] w.r.t X[n], we find that correlation between X1[n] and X[n] is slightly lower.
Hence, we conclude that X2[n] is a better estimate of original signal for given distorted signal.Therefore we can say that the original signal was first blurred and then noise was added to it.

Explanation :

Blurring, by default, removes high- frequency information. Thus, the high frequencies of *H* must go towards zero. The reason for this is that the high-frequency information of *X* is lost when it is blurred -- thus, the high-frequency components of *Y* must go towards zero. For that to happen, the high-frequency components of *H* must also go towards zero. As a result of the high-frequency components of *H* being almost zero, we see that the high-frequency components of *Y* is amplified significantly (as we almost divide by zero) when we deconvolve with the FT. This is not a problem in the noise-free case. In the noisy case, however, this is a problem. The reason for this is that noise is, by definition, high-frequency information. So when we try to deconvolve *Y*, the noise is amplified to an almost infinite extent. Therefore we first try to remove the noise of the signal before we try to deblur that. This is the reason why we get better result in case1.

The End

Thank You!