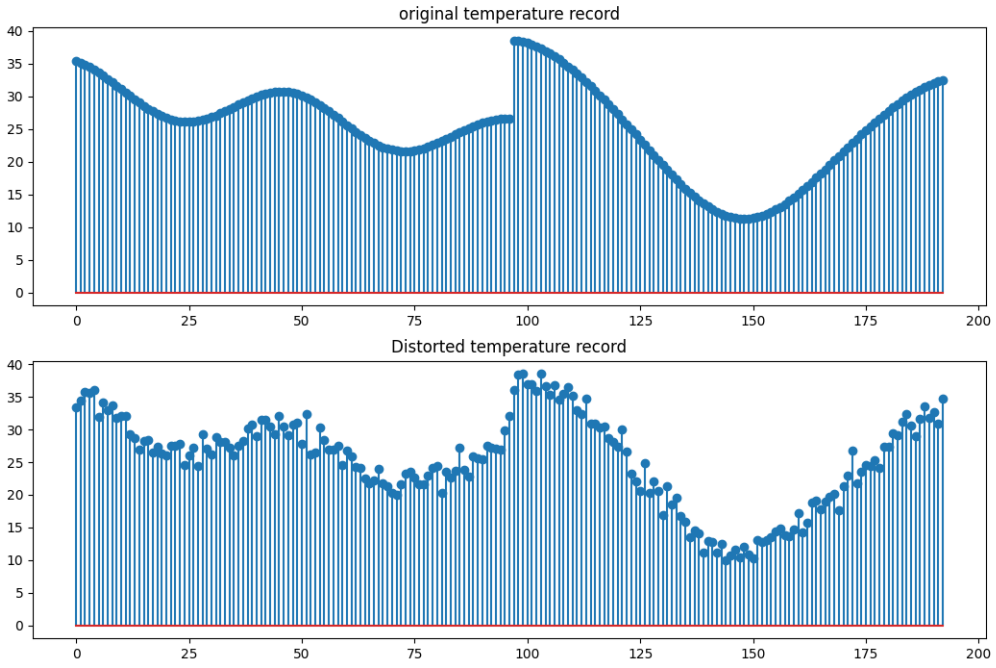
SNS Programming Assignment EEL2010

Ayush Anand(B20AI005), Aman Thakur(B20AI003)

July 25, 2021

1 The Problem Statement

In this problem we have been given a signal y[n] which arises due to the effects of blurring and noise on the original signal x[n].We also know the impulse response used for blurring.



Our task is to recover the signal x[n] from y[n]. It is not possible to obtain x[n] exactly hence we need a metric to describe how close our output is from the original signal x[n]. We choose that metric to be MSE(Mean squared Error).

2 About Metrics

2.1 MSE

The MSE metric is defined as

$$\sum \frac{(y_{true} - y_{pred})^2}{N}$$

. The lower the MSE value the better the prediction is.

2.2 delta metric

The problem with using only MSE to evaluate the performance of our system is that it is more sensitive to improvements in noise but not very sensitive to improvements in the deblurring. This arises due to the fact that there is only one prominent discontinuity at about t=96 in the plot of x[n] which means that improvement in deblurring part will have major effect on only a very small part.We must devise a certain metric to understand the impact of our deblurring strategy. We choose this metric as

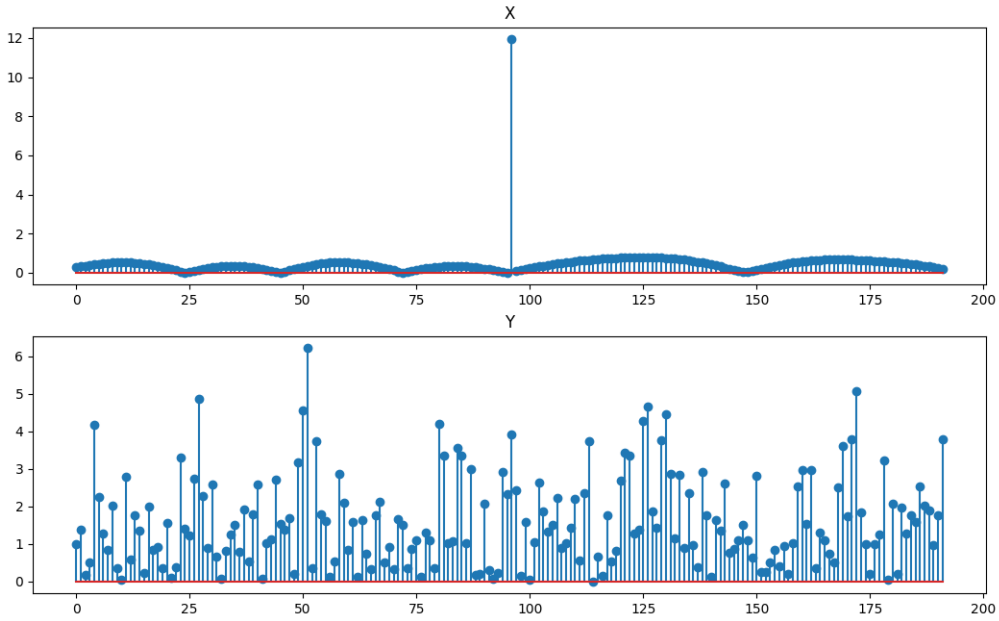
$$(y[97] - y[96])$$

. If this metric is higher and MSE is decent then we know that deblurring system works well.

2.3 Plotting

While metrics are good we feel that it is also important to judge the output based on its graph. For this purpose we plot the discrete time signals with a stem plot. The choice of stem plot is not arbitrary, Line plots interpolate the distance between two discrete points adjacent to each other thereby making it tough to judge the amount of blur involved in the signal.Scatter plots on the other hand do handle this well but it becomes tough to keep track

of time coordinate in scatter plots. Therefore we felt stem plot to be the most appropriate choice. For visualising the Blur in a signal we can plot the consecutive differences of the signal.The point where a sharp spike occurs is an edge.



3 Modelling Blur and Noise

Let I_c represent the clear signal and I_d represent the distorted signal. We assume that the clear signal is first deblurred and then some additive noise has been added to it. This implies

$$I_c = I_d * h + N$$

Where h is the deblurring kernel and N is the additive Noise.

4 Implementing Convolution

Convolution in 1D,2D, or any dimension is essentially reversing the kernel with which we are convoluting and then taking a "sliding window" sum.What that means is we loop through the signal and place the n=0 point of the given kernel at the position where the loop currently is, And then take a weighted sum.
Check the utils.py file for the implementation of convolution.

4.1 Padding and Edge effects in convolution

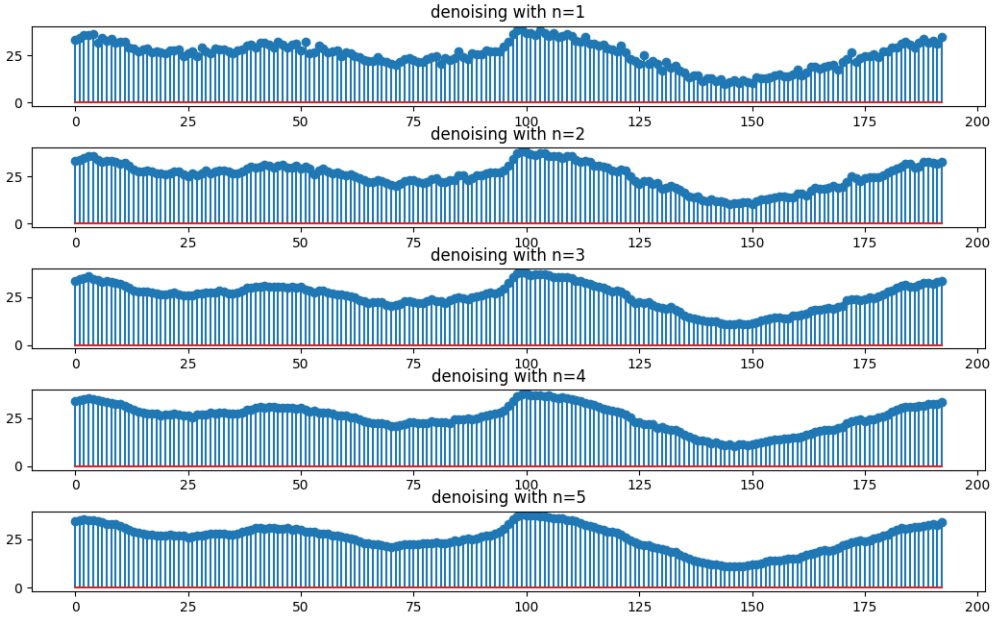
While the convolution works well in the interiors of the signal certain problems arise near the two edges of the signal. The "sliding window" lies outside the signal and hence an ambiguity arises as to what should be taken as the value of signal outside the edges. This value is known as "padding", it is sometimes kept 0 however the problem with keeping the padding as 0 is the fact that the weighted sum gets dragged down to 0 and does not really resemble the convolution very well in context of denoising. Hence we chose to keep the padding same as the value of signal at the point corresponding to n=0 of the kernel.

5 Denoising

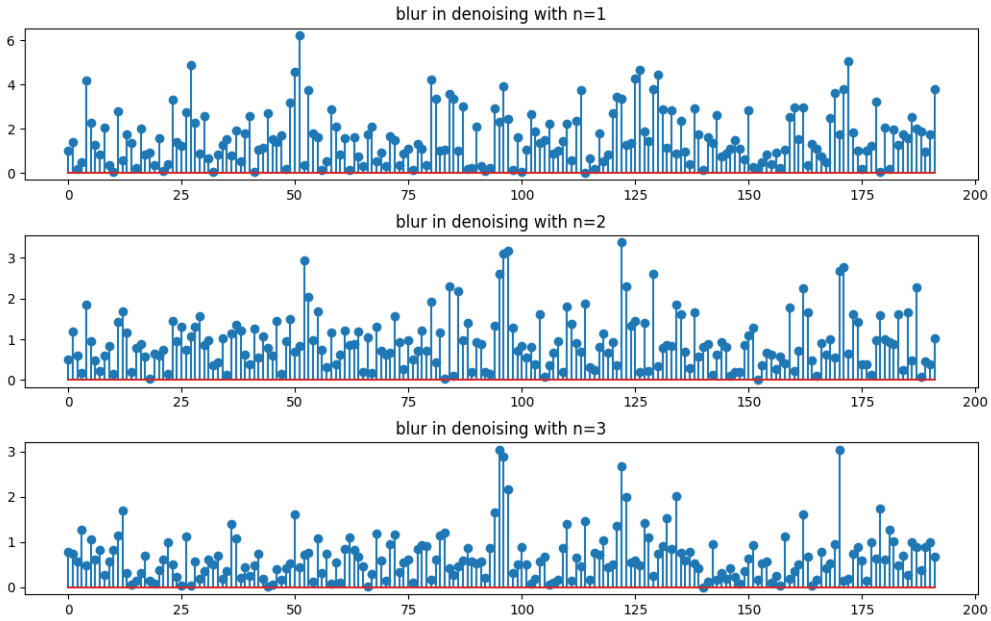
Noise is usually additive and generally follows a mean-0 distribution. One way to reduce noise is to take averages of some adjacent time instants. This is equivalent to taking the convolution of the signal with a kernel of the form

$$\begin{bmatrix} \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix}$$

.The Results of denoising the signal are shown in the figure.


denoising with n=1
denoising with n=2
denoising with n=3
denoising with n=4
denoising with n=5
0
25
0
25
50
75
100
125
150
175
200

To judge the blur here is difference graph(Note: The y-axis scale of n=1).The decrease in peak signifies more blurring.



As we can see on increasing the kernel size there appears to be a decrease in the noise. However there is a tradeoff between Noise and Blur. Increasing the kernel size also increases the amount of blur in the produced signal. This is because averaging on the boundary of discontinuities makes the discontinuity look more continuous. We feel the ideal kernel size for deblurring taking into account this trade-off is 5.

6 Deblurring

Going back to our model described in section 3 we apply Fourier transform on both the sides yielding

$$\mathcal{F}(I_d) = \mathcal{F}(I_c)\mathcal{F}(h) + \mathcal{F}(N)$$

. Solving this equation in terms of $\mathcal{F}(I_c)$ we get

$$\mathcal{F}(I_c) = \frac{\mathcal{F}(I_d) - \mathcal{F}(N)}{\mathcal{F}(h)}$$

If we neglect the noise for a moment and then take the inverse Fourier transform of the above expression we can recover the clean signal.In practice there will still be some inaccuracies and moreover noise cannot be completely eradicated.

6.1 Getting the fourier transform of signal and Blur impulse response

First we describe the algorithm for obtaining the fourier transform. By the definition of FT we have

$$FT = \sum X(jw)e^{-njw}$$

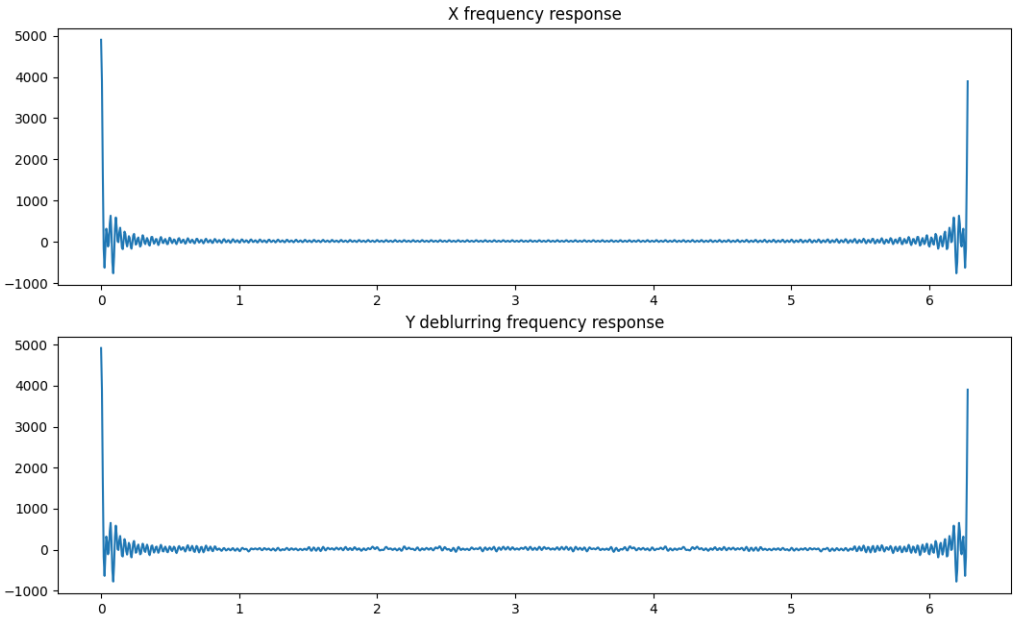
. We can implement this by using a simple FOR loop and adding terms one by one. We however evaluate this function for the special case of the $\begin{bmatrix}\frac{1}{16} & \frac{4}{16} & \frac{6}{16} & \frac{4}{16} & \frac{1}{16}\end{bmatrix}$ kernel. It evaluates to

$$\frac{2cos(2w) + 8cos(w) + 6}{16}$$

which when simplified becomes

$$cos(\frac{w}{2})^4$$

. This function has been implemented under BlurFourier function for performance improvements. For the signal we just evaluate its fourier transform with a simple for loop by passing the signal Y to Fourier function.



6.2 Getting the inverse fourier transform

The inverse fourier transform of X(jw) is given by

$$x[n] = \frac{1}{2\pi} \int X(jw) e^{jwn} dw$$

.It is difficult to programmatically calculate the exact integral without the use of external libraries.We can however get a good approximation by converting this integral to a Riemann Sum and then evaluating it with a simple for

loop.Dividing the interval between 0 to 2π in N parts we get the width of each strip as $dw = \frac{2\pi}{N}$ and the ith strip will have height $X(jw)e^{jwn}$ where $w = \frac{2\pi i}{N}$. Now the sum becomes:-

$$\frac{1}{N}\sum X(j\frac{2\pi i}{N})e^{j\frac{2\pi i}{N}n}$$

We implement the inverse fourier transform in the Inv function contained in utils.py file.

6.3 Singularity of FT of blur response

As described in section 6.2 we can see that at $w = \pi$ the FT of blur response equates to 0 and at values of w adjacent π the value of FT is very close to 0. This leads to problems in computing the Fourier Transform of the ratio described in the deblurring model(section 1.5).As we will be computing a Riemann sum we will only need to wory about some specific angular frequences and not actually calculate the expression at $w = \pi$. However any practical programming language does not deal well with extremely large values and therefore arises a need to somehow prevent these large Values from arising.One way is to approximate the answer by thresholding the denominator. i.e.

$$\mathcal{F}(h) = T$$

when

$$\mathcal{F}(h) < T$$

. The idea behind this approximation is that by choosing a small enough appropriate T we "almost" simulate a zero in the denominator and hence obtaining a good approximation. We however instead of $\mathcal{F}(h) = T$ chose to implement $\mathcal{F}(h) = \mathcal{F}(h) + T$ in order to preserve the relative relation between the points within threshold.

6.4 Implementing deblurring

Now we combine the previous three parts to implement the deblurring i.e

1. Obtain fourier transform of blurring kernel and distorted signal
2. Divide the two(with thresholding)

3. Sample the function obtained in step 2 based on the number of partitions done in Riemann sum

4. Evaluate the inverse fourier transform

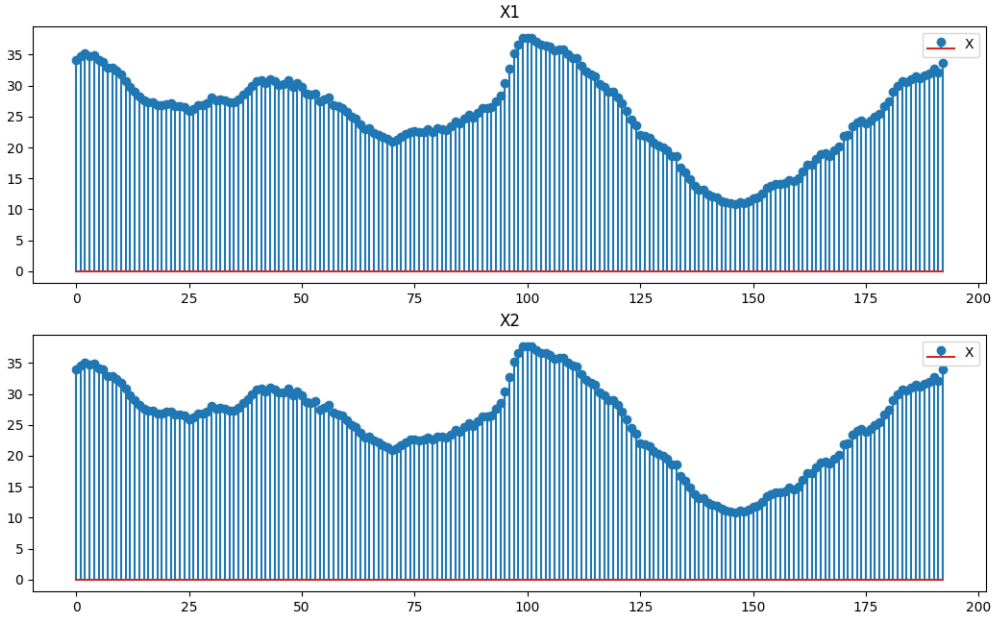
There are two important questions which affect the quality of our produced output :

- What should be the value of T in Sec 6.3?
- How many partitions of Riemann sum do we need to take?

To answer these Questions one approach would be to take a subset of the data and then by Trial and Error find the appropriate T and number of partitions. We got the appropriate value of T as 0.7 and the number of partitions as 193(Which is the same as the number of samples in the signal).
Do refer to the appendix at the bottom of this document for more details on these two questions.

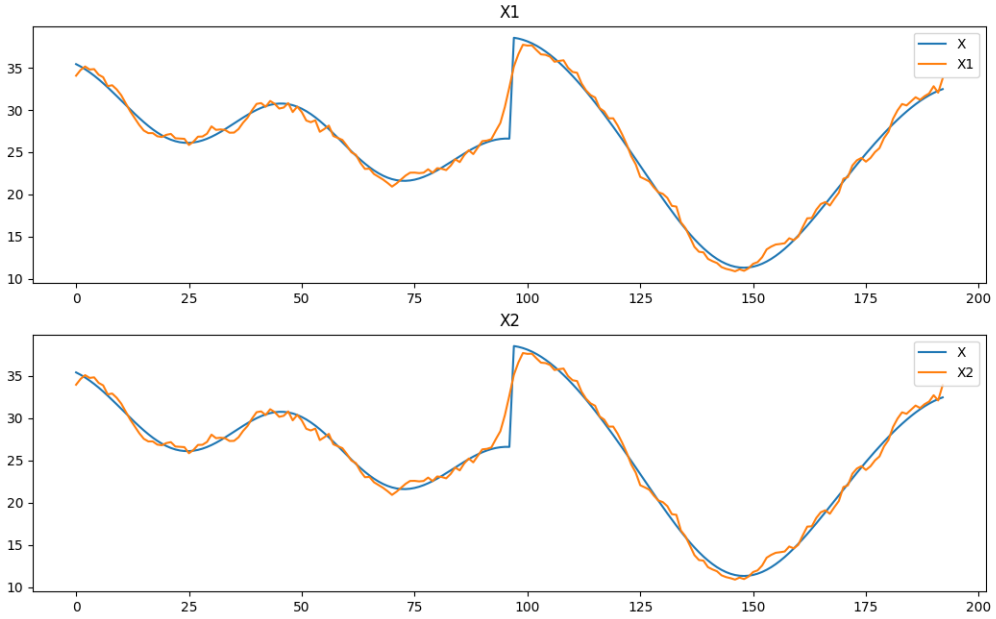
7 Final Result

Now we finally evaluate the two signals X1 and X2 as asked in the Q. X1 is produced after denoising then deblurring while X2 is produced after deblurring then denoising



The MSE scores of the produced outputs are:-

Model	MSE	Delta Metric
X1(denoise-Deblur)	0.74818	2.4614
X2(deBlur-deNoise)	0.75184	2.4620
Y(original distorted signal)	2.0611	3.9061



Both X1 and X2 are pretty close to each other with X1 giving a slightly better performance.

7.1 Theoretical explanation of differences between X1 and X2

Let us return back to

$$\mathcal{F}(I_c) = \frac{\mathcal{F}(I_d) - \mathcal{F}(N)}{\mathcal{F}(h)}$$

. Now in this equation ideally if $\mathcal{F}(N)$ were to be 0 we would have good results but usually it is not the case. In the case of X1(deNoising then deblurring) the $\mathcal{F}(N)$ reduces due to denoising being done first, And then when performing the deblurring given a decent threshold value T the small remaining noise would not be amplified so much. In case of X2 we first deBlur it, At this stage it has pretty significant noise however DeNoising after this removes those high frequency components(Noise) hence the similar results. However the Tradeoff here is the fact that deNoising also introduces some

Blurring into the output but since there is only one sharp edge it does not have that much of a pronounced effect.

8 Team Member Contributions

Both of us worked jointly to implement this report. At various instances we helped each other debug the code and we both worked on all of the parts of this assignment together.We spent time discussing things like the effects of convolution at the edges and other subtleties which arose while implementing.The report writing work is primarily done by me(Ayush Anand) but not without the consultation of my team mate.We discussed the issues and while I would write the code he helped me review it and remove the bugs.

9 Appendix

Let us return to the two very important questions which we posed in section 6.4.

9.1 Threshold value T

First we analyze the effect of threshold value T on our produced signal.The main purpose of T is to limit the denominator so that

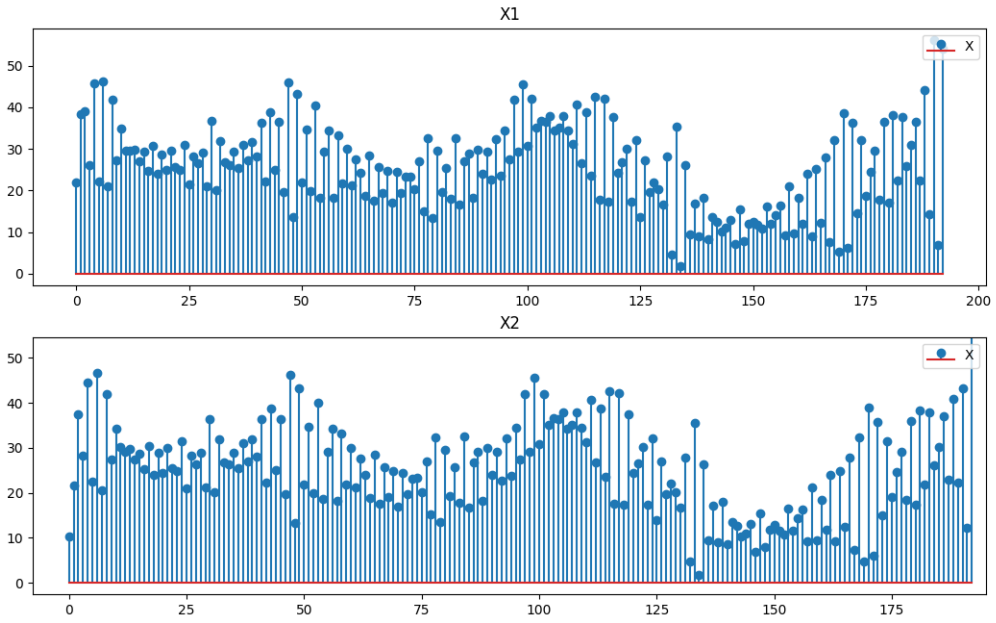
- Programming languages do not get to deal with extremely large values(overflow problems could occur for example).
- The noise does not get very amplified.

The second point here is particularly important. Choosing T around 0.3-0.9 range will **not amplify** the noise so much and hence we get very little difference between x1 and x2. However if we choose T to be around say 0.01 then X1 will significantly outperform X2.This is because the noise has been significantly reduced in the first step of denoising in X1, And then while deblurring tho the little remaining noise gets very amplified due to the small denominator its still less than in case of x2.
What happens in X2 however is that when we deblur the signal in first step the noise is pretty significant and it gets amplified by a very large factor.The second step i.e denoising will not be very effective now because the mean of

the noise has been amplified too much so for e.g if the mean originally was 0.05 it will now be somewhere near 5 in the signal produced by first step, Denoising hinges on the fact that the noise has an approximately mean-0 distribution. The MSE scores for T=0.01 are:-

Model	MSE	Delta Metric
X1(denoise-Deblur)	57.1647	14.268
X2(deBlur-deNoise)	66.8775	14.595
Y(original distorted signal)	2.0611	3.9061

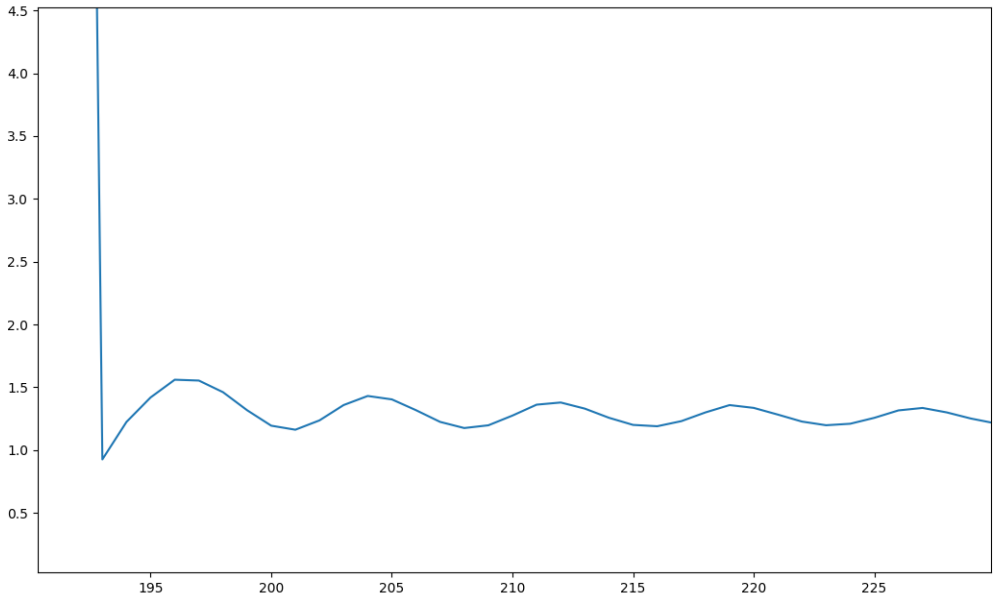
Note that the delta metric is higher which signifies that the deblurring is working a little better for T=0.01 but because the noise is significant enough to bring down the MSE it is not very desirable to use T=0.01. The stem plots for T=0.01 are:-



9.2 Number of riemann partitions

Now we analyze the effect which the number of riemann partitions we choose have on the produced signal.To do that we first plot a graph between MSE

value vs the number of partitions.



Note: The graph has been zoomed in.
This graph is what motivated the choice for choosing 193 partitions. Taking more partitions makes the MSE also to approach a limit but it is suboptimal this is probably due to the presence of some noise in the signal.