# HW 4

## 

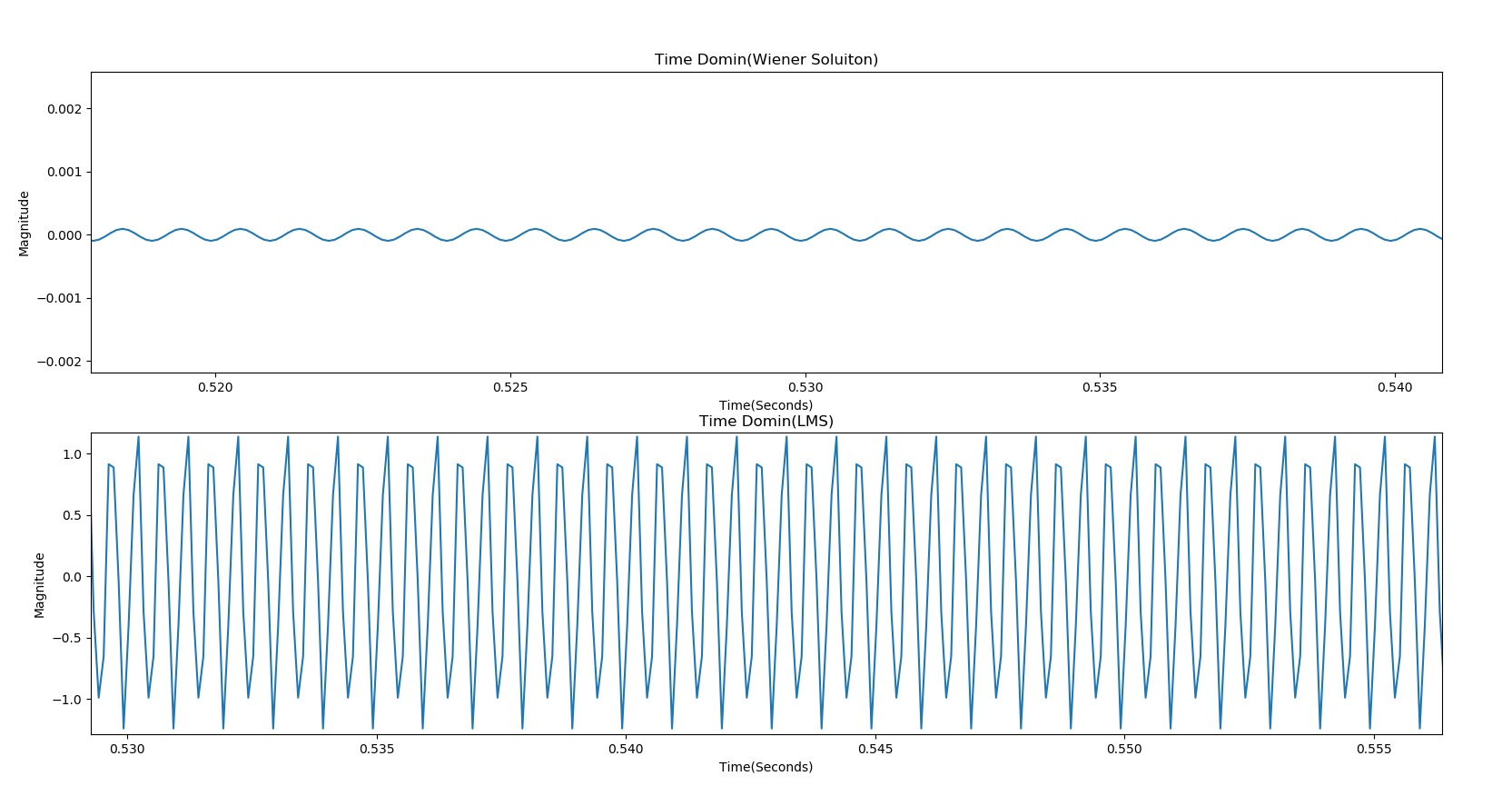


Figure 1 the Output Signal in Time domin between Wiener Solution and LMS

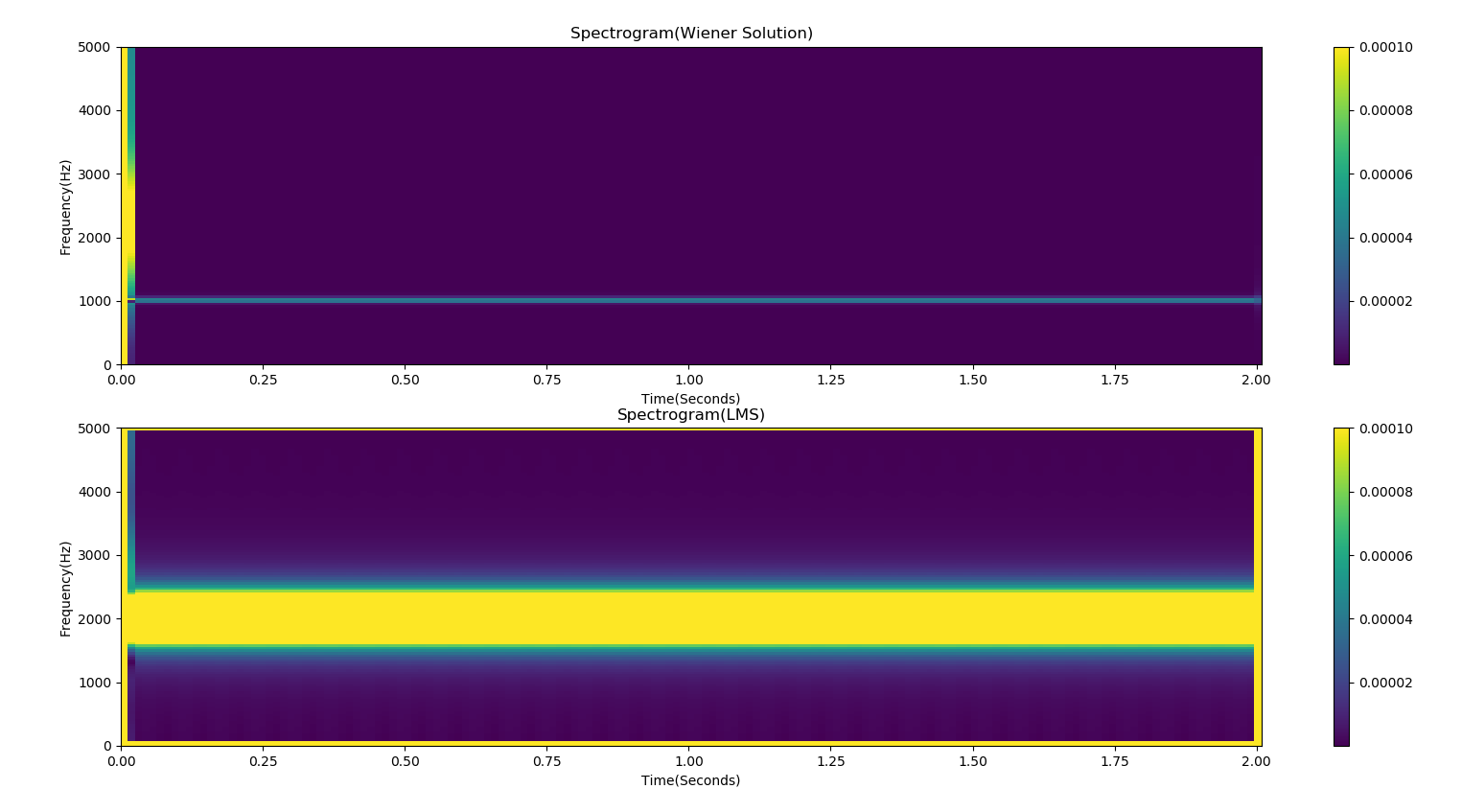


Figure 2 Spectrogram (Output Signal) between Wiener Solution and LMS

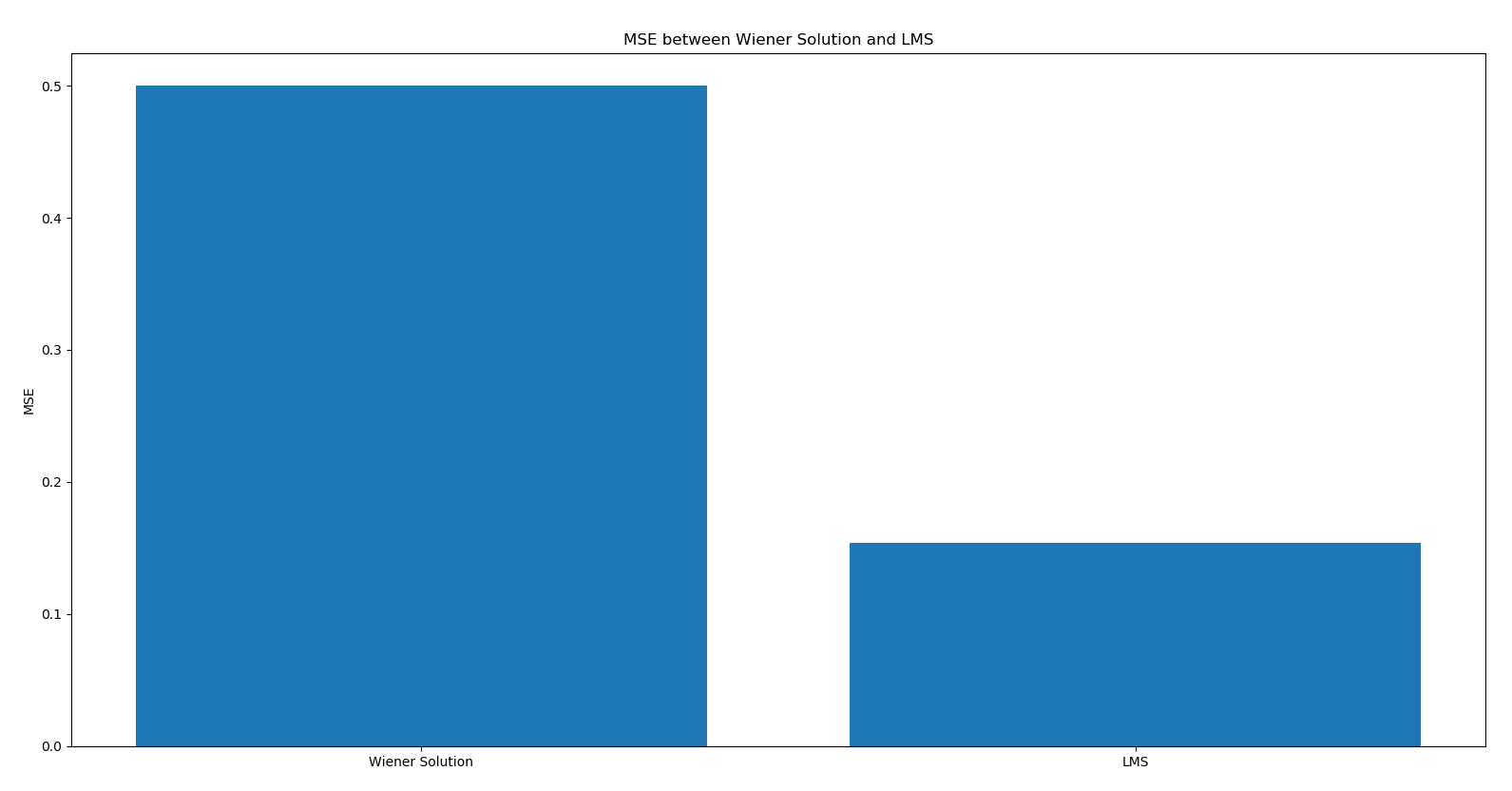


Figure 3 MSE between Wiener Solution and LMS

Implement details:

For input signal x, I created a 2 seconds data with single sinewave at 1KHz, sampled at 10KHz.

For desired response d, I created a 2 seconds data with single sinewave at 2KHz, sampled at 10KHz.

For Wiener filter, I set order = 10, window = 20000.

For LMS, I set order = 10, step size = 0.2.

From Figure 1 we can see that magnitude of Wiener Solution is so small while the magnitude of LMS is close to 1. From Figure 2 we can see that the output frequency of Wiener Solution is still 1KHz while the output frequency of LMS is kept at 2KHz. Finally, from the Figure 3 we can see that the MSE of Wiener Solution is larger than the MSE of LMS. So from those figures above, we can conclude that Wiener Filter has worse performance. The reason is that LMS is adaptive filter which can adjust parameters according to an optimization algorithm but Wiener filter has only calculated once without any iteration.

## 

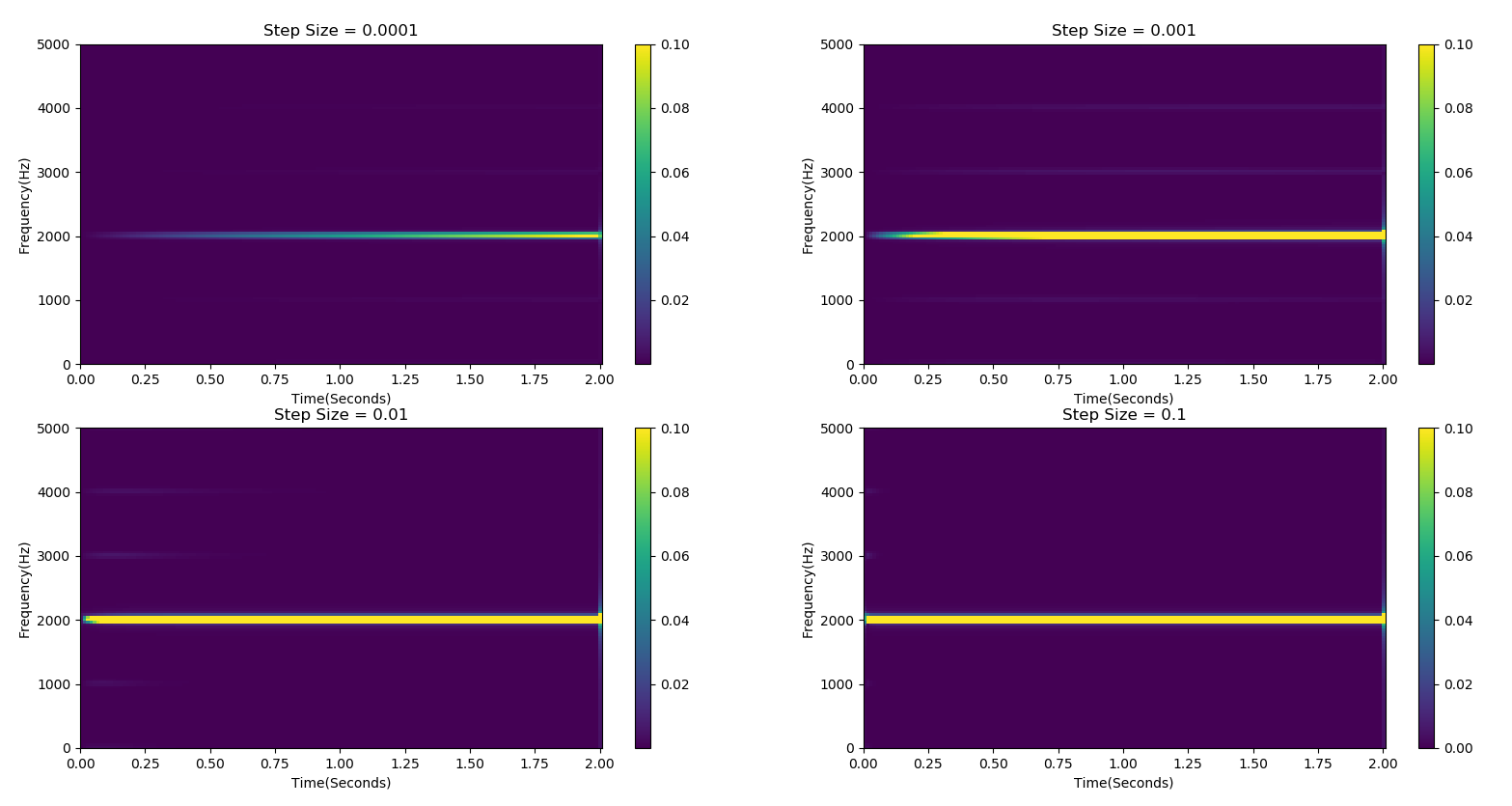


Figure 4 the Effect of Step Size

For testing different step size, I set kernel size = 1, order = 10, and choose step size = 0.0001, 0.001, 0.01, 0.1.

Form Figure 4 we can see that the higher the step size, the faster the convergence rate. All four tests above have converged to 2KHz.

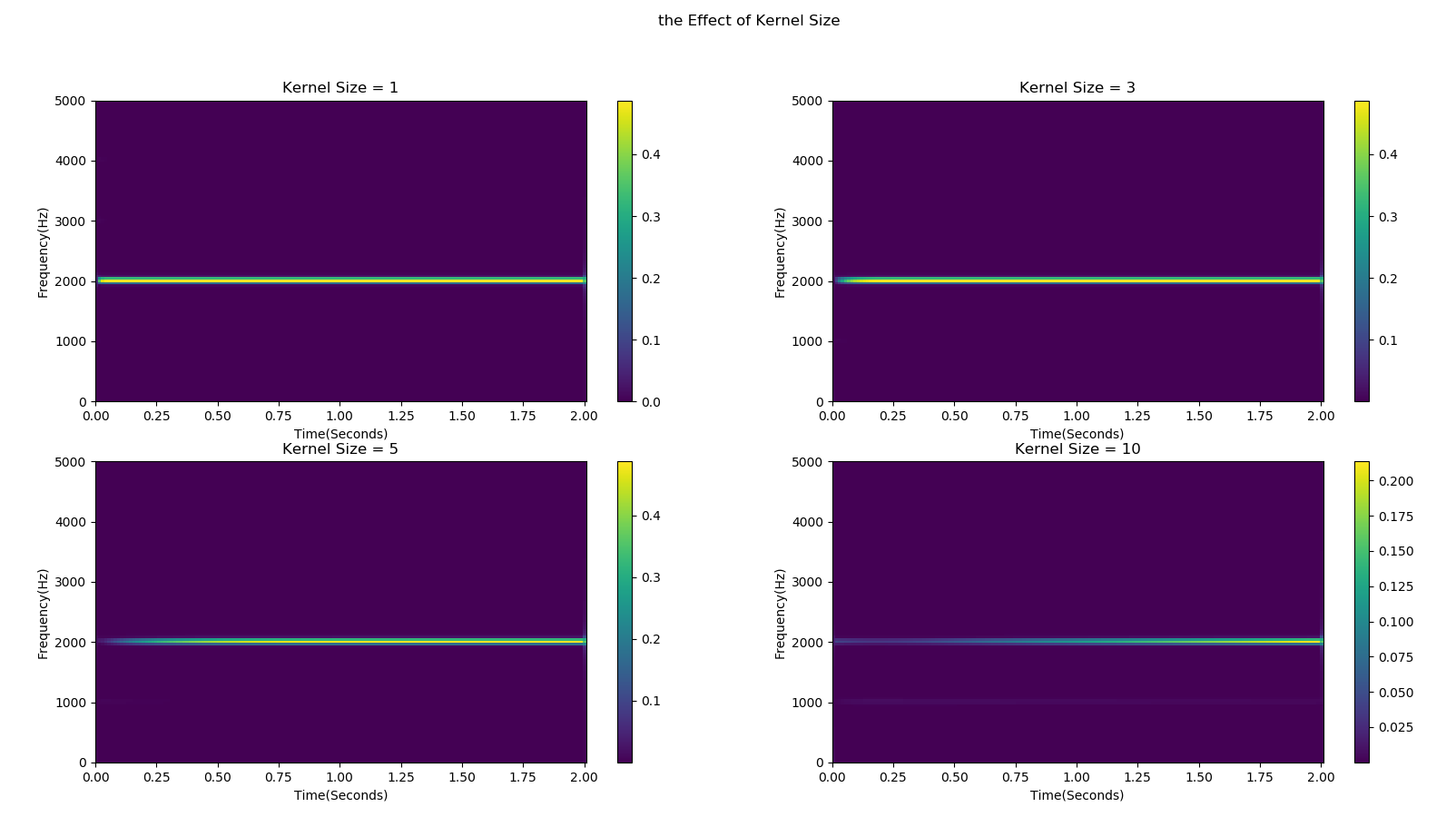


Figure 5 the Effect of Kernel Size

For testing different kernel size, I set step size = 0.1, order = 10 and choose kernel size = 1, 3, 5, 10.

Form Figure 5 we can see that the larger the kernel size, the lower the convergence rate. All four tests above have converged to 2KHz.

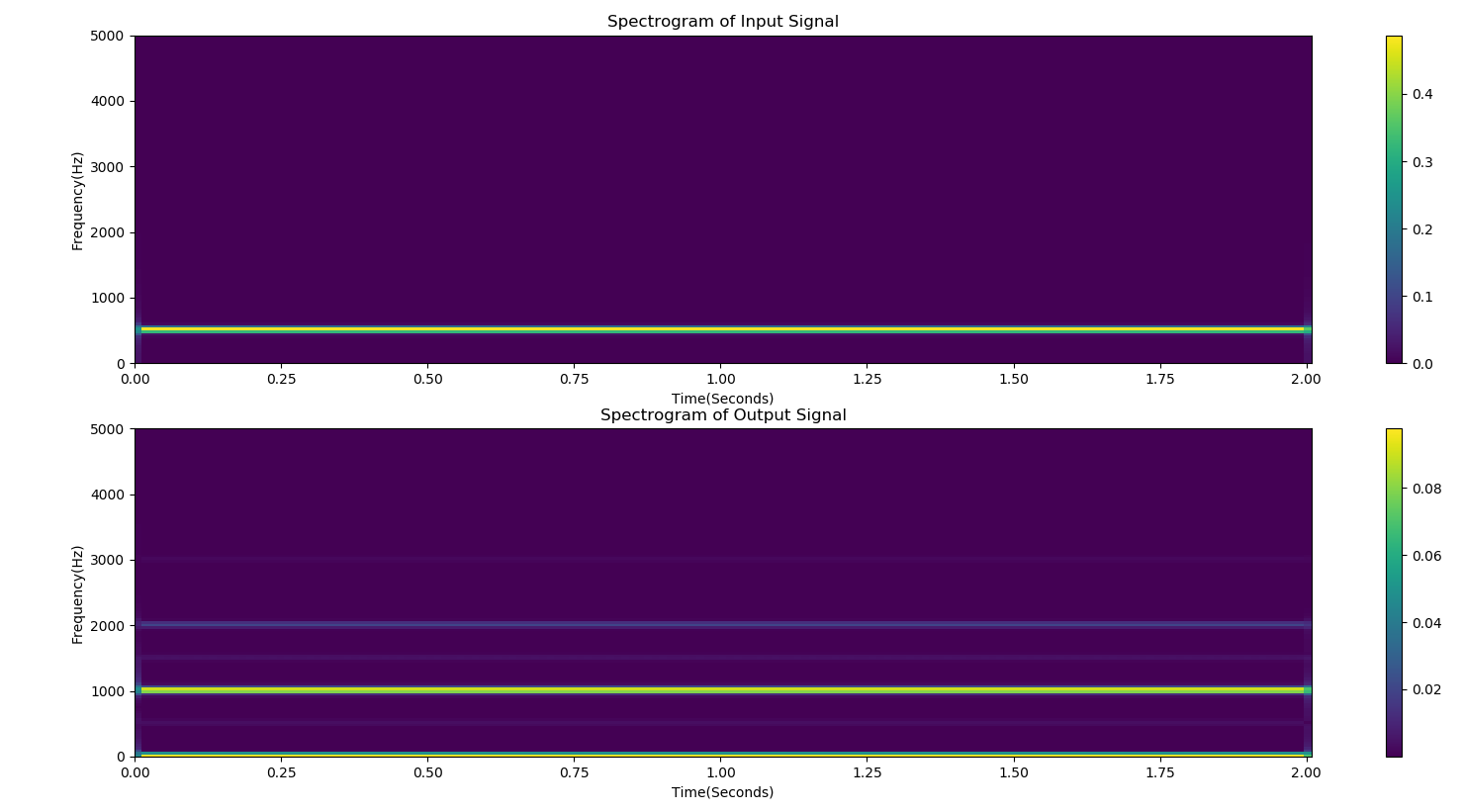


Figure 6 Test by Input Frequency = 500Hz

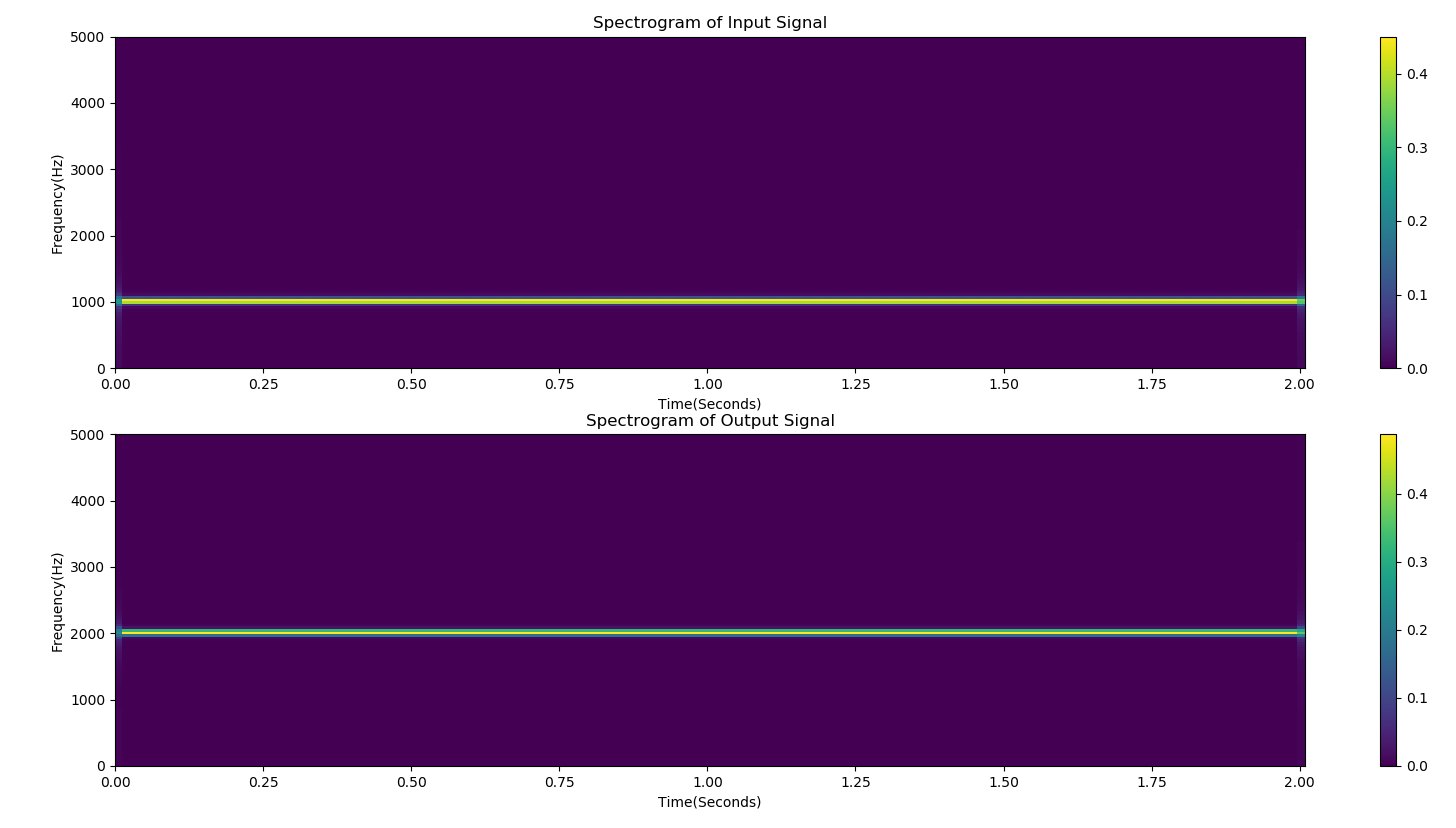


Figure 7 Test by Input Frequency = 1000Hz

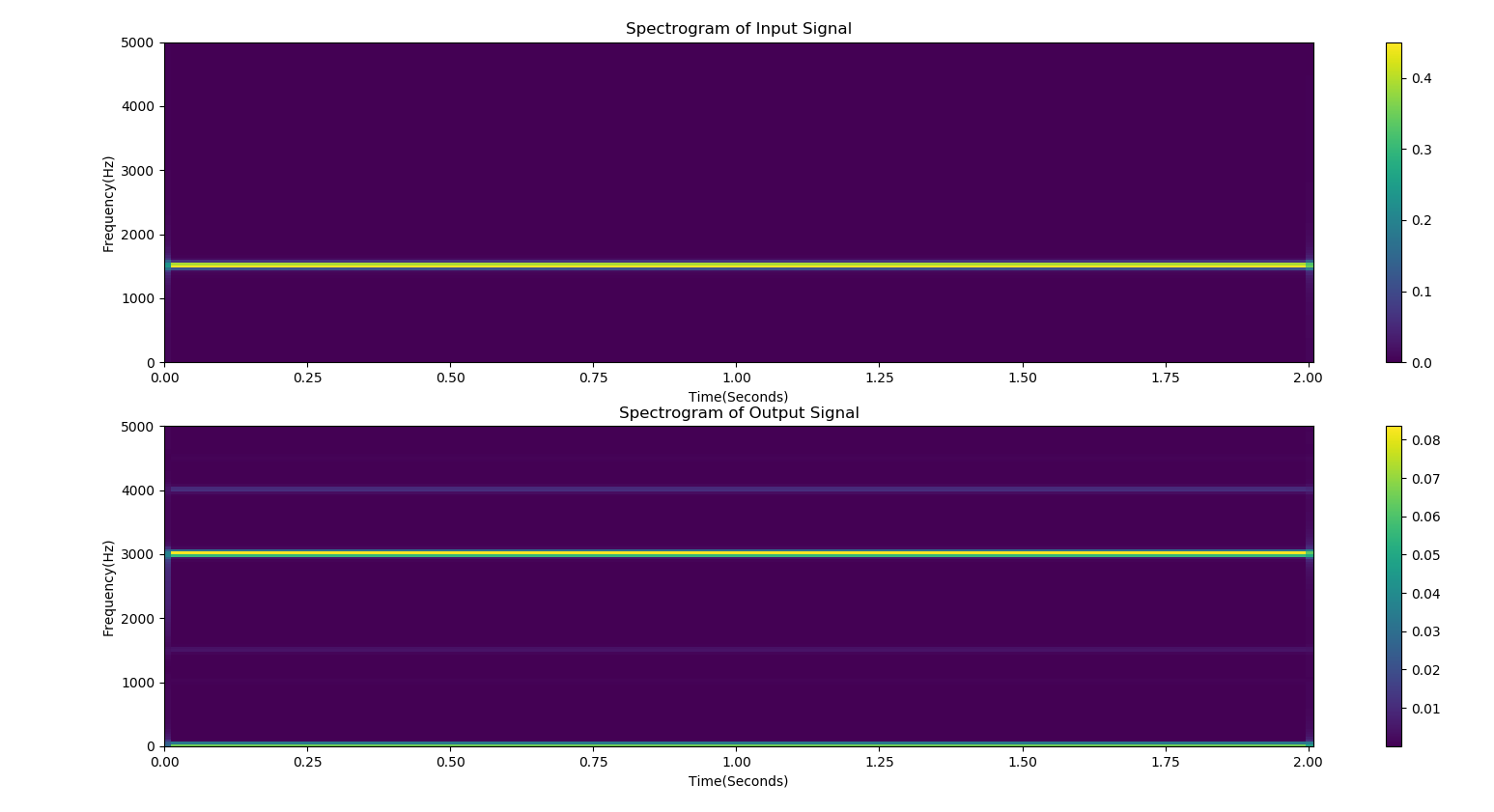


Figure 8 Test by Input Frequency = 1500Hz

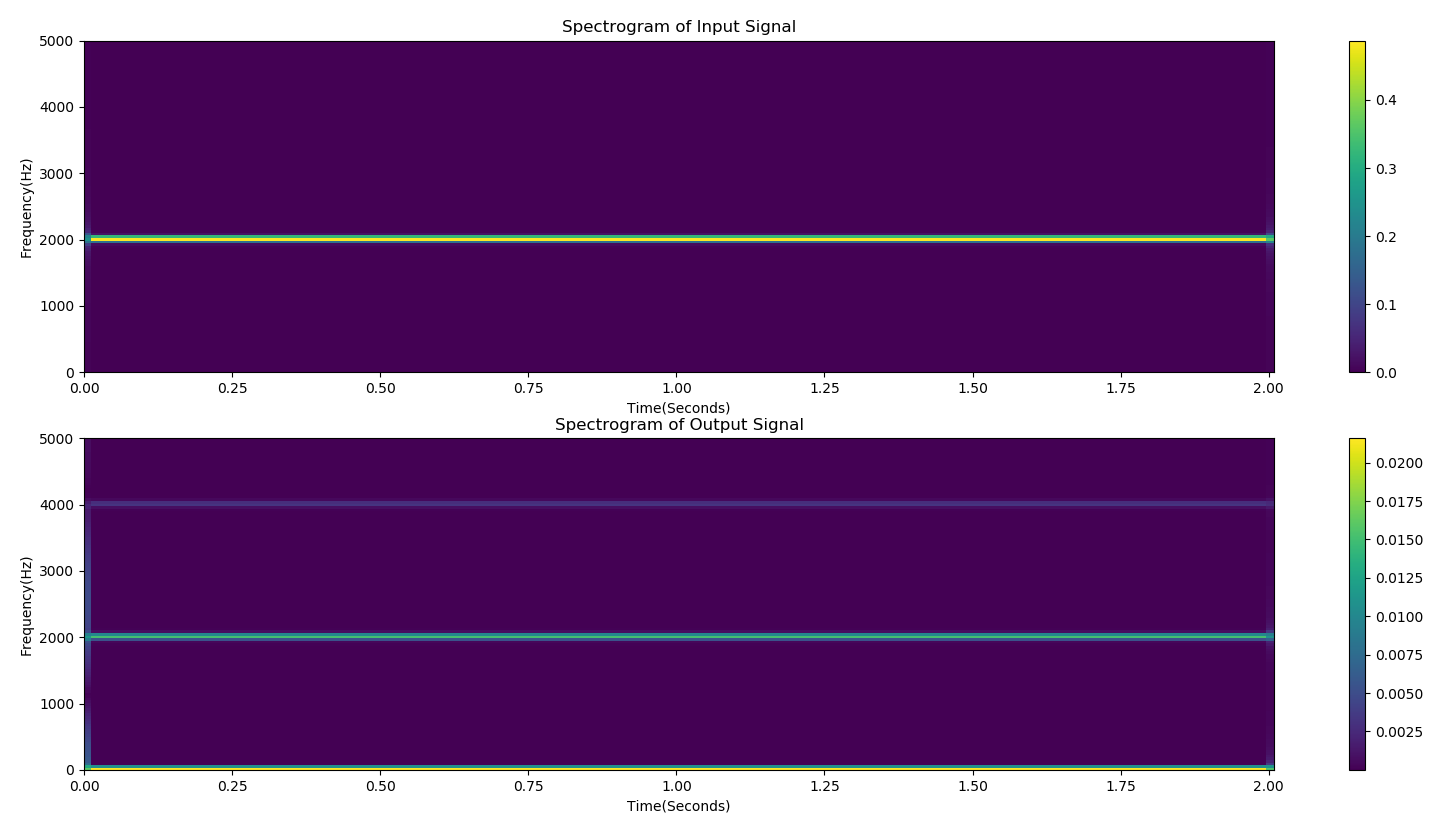


Figure 9 Test by Input Frequency = 2000Hz

Firstly, I trained the model with step size = 0.1, kernel size = 1.

Then I tested the generalization of the trained model. From the Figure 6 to Figure 9, we can see that when test by input frequency = 1000Hz, the model has the best output result, which the output is exactly 2000Hz. This is because we trained the model by 1000Hz input signal. Also, the farther test input frequency is from 1000 Hz, the worse the output result. We can see that 2000Hz input signal has the worst output compared with other input signals.

## 

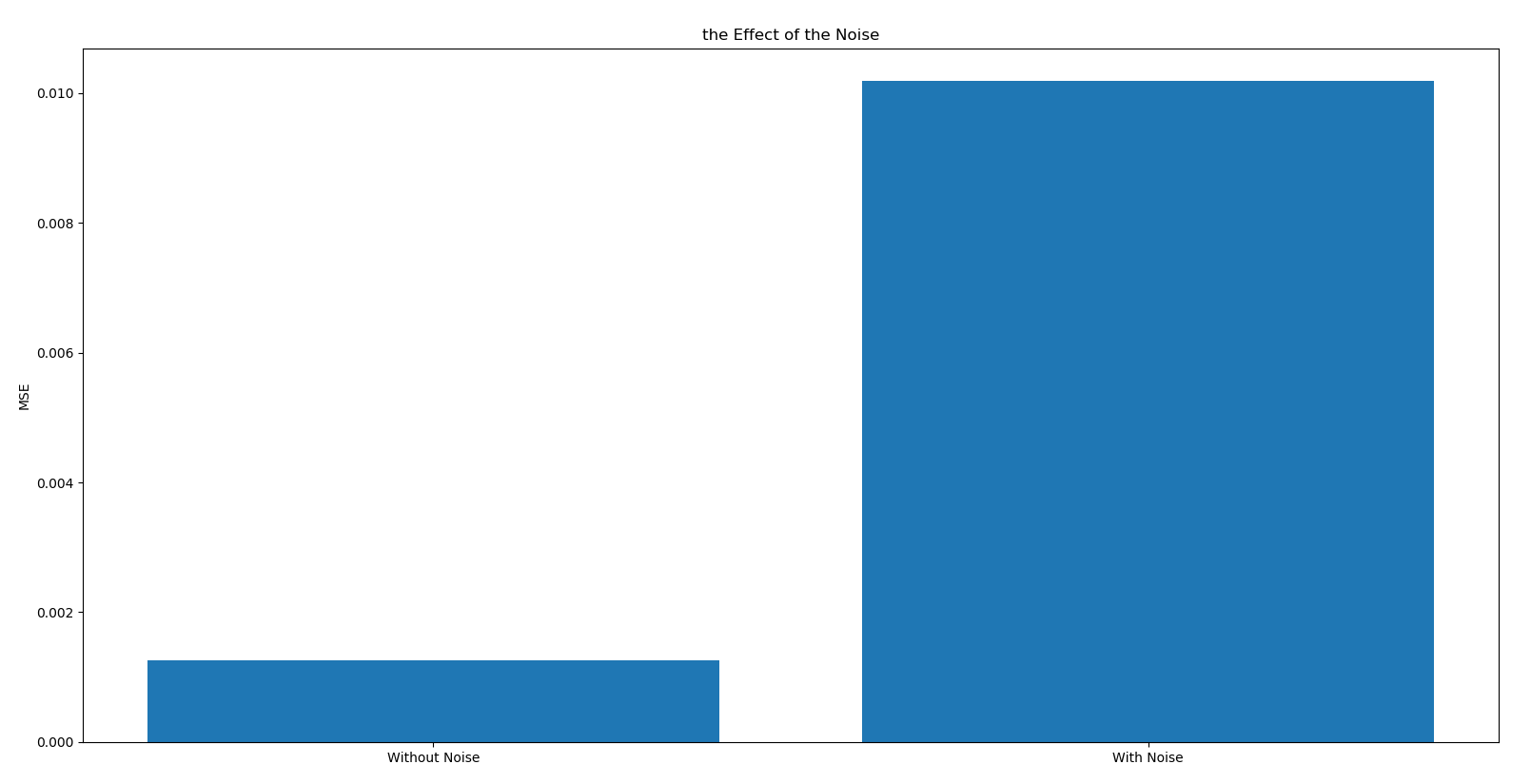


Figure 10 the Effect of the Noise

To study the effect of the noise, I add the noise to the desired response and use KLMS algorithm with step size = 0.1, kernel size = 1, order = 10.

From Figure 10 we can see that the MSE is larger when we add the noise to the desired response. So the noise increases the model’s error.

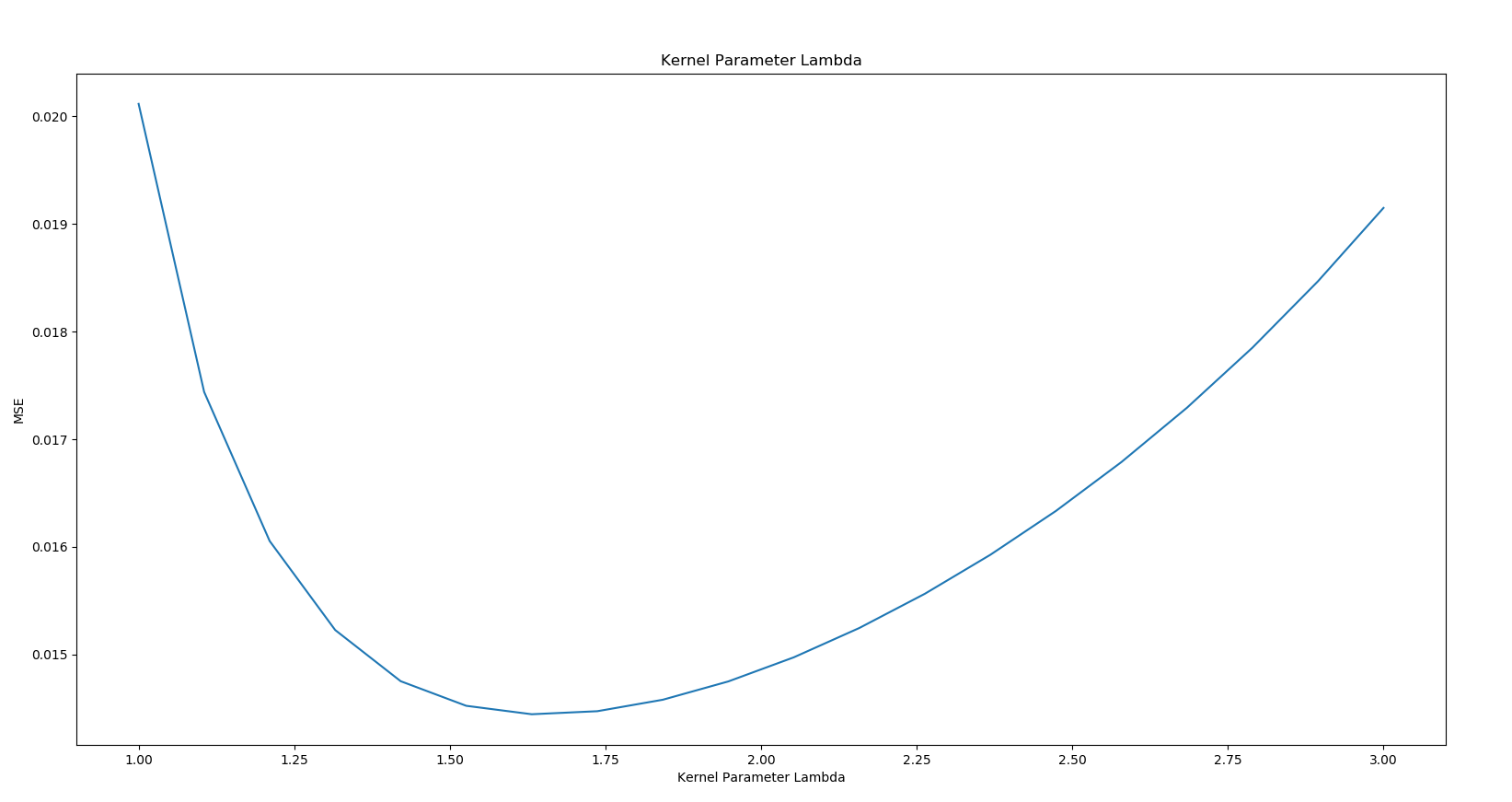


Figure 11 the Effect of Kernel Size

To determine the kernel size in MCC for optimal results, I set step size = 0.2, kernel parameter = 2.

From Figure 11 we can see that when kernel size = 1.6, the MCC-KLMS has the lowest MSE, which means the optimal result.

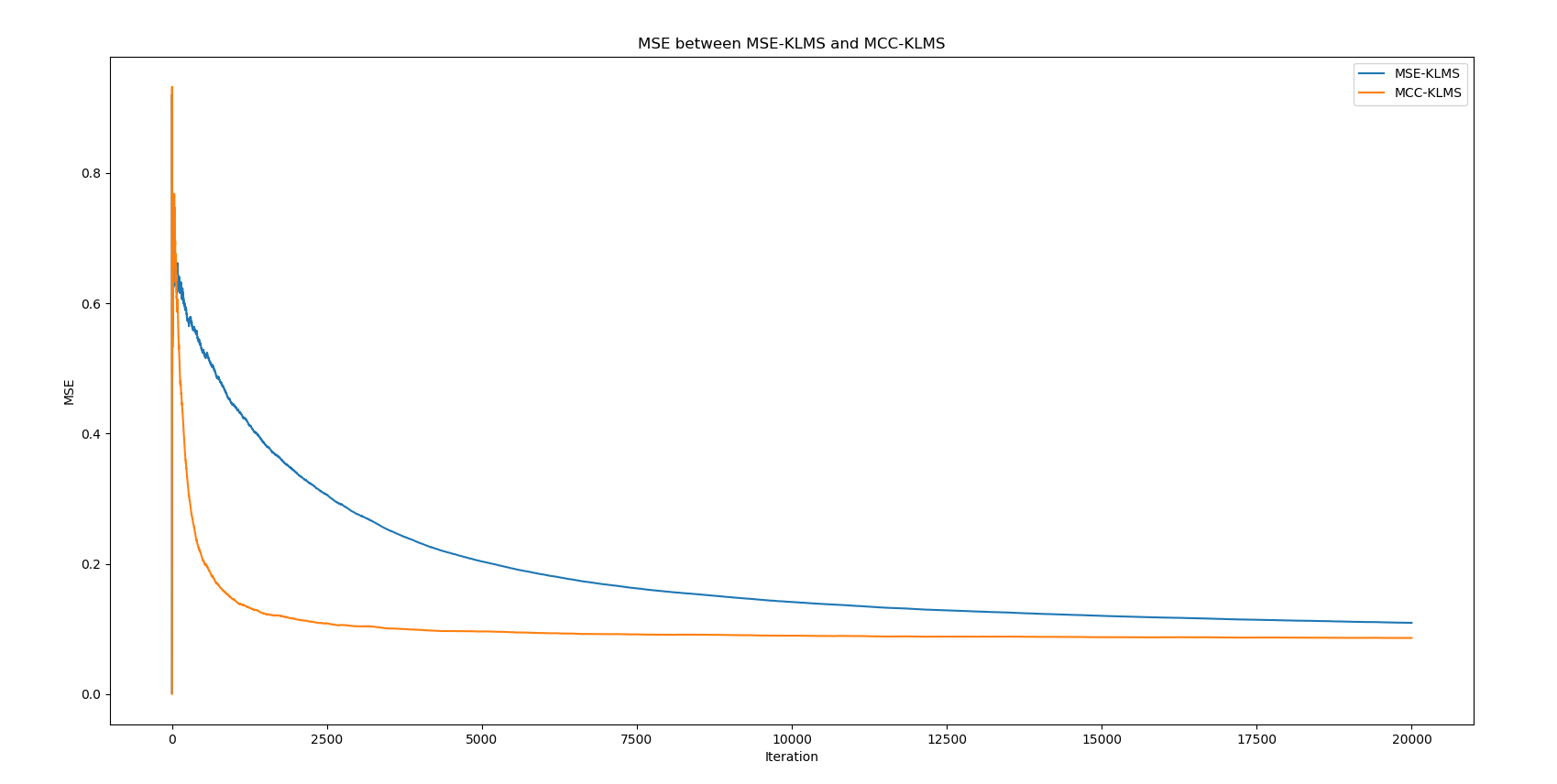


Figure 12 MSE between MSE-KLMS and MCC-KLMS

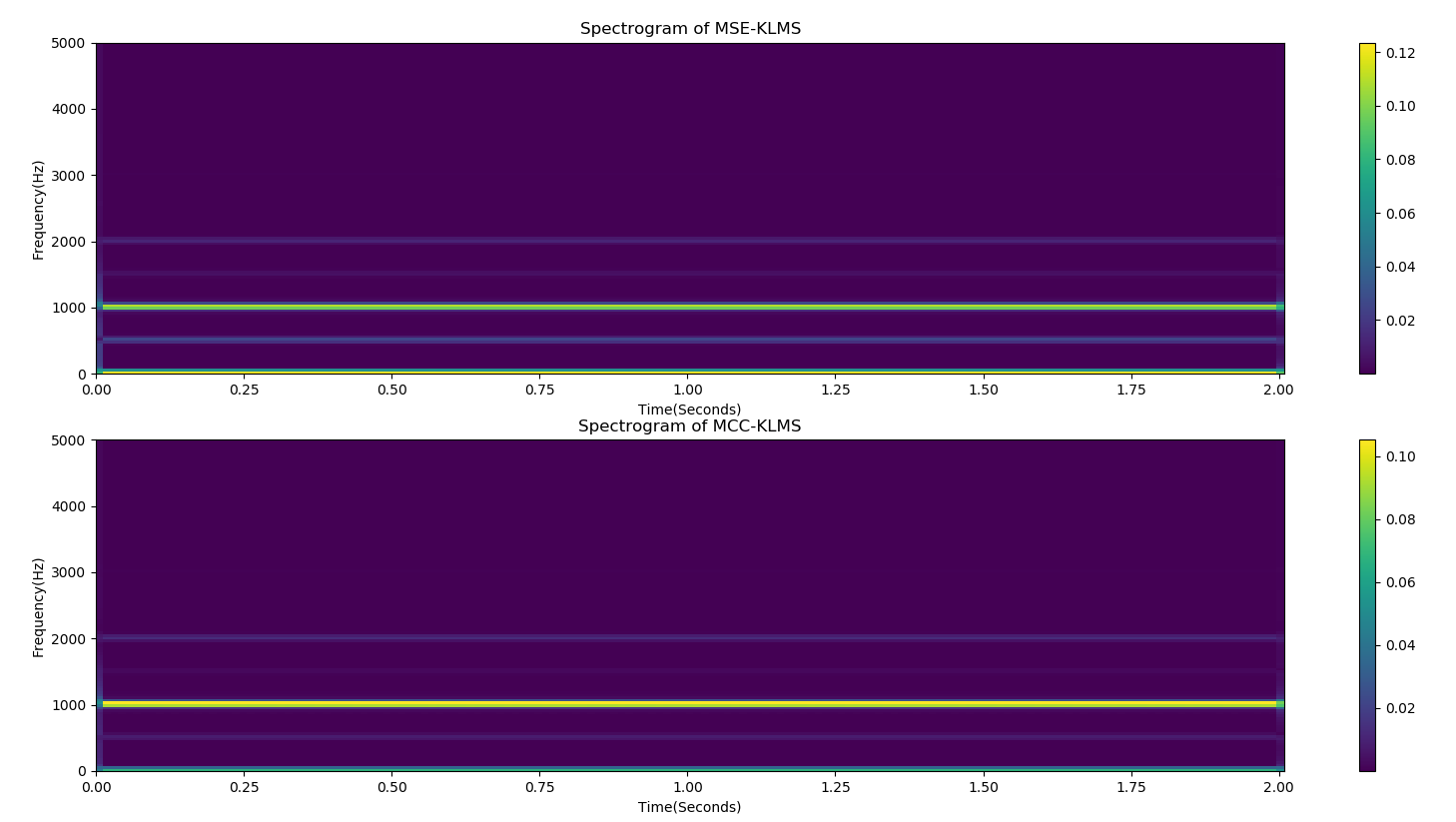


Figure 13 Spectrogram of MSE-KLMS and MCC-KLMS (Frequency of Test Input Signal = 500Hz)

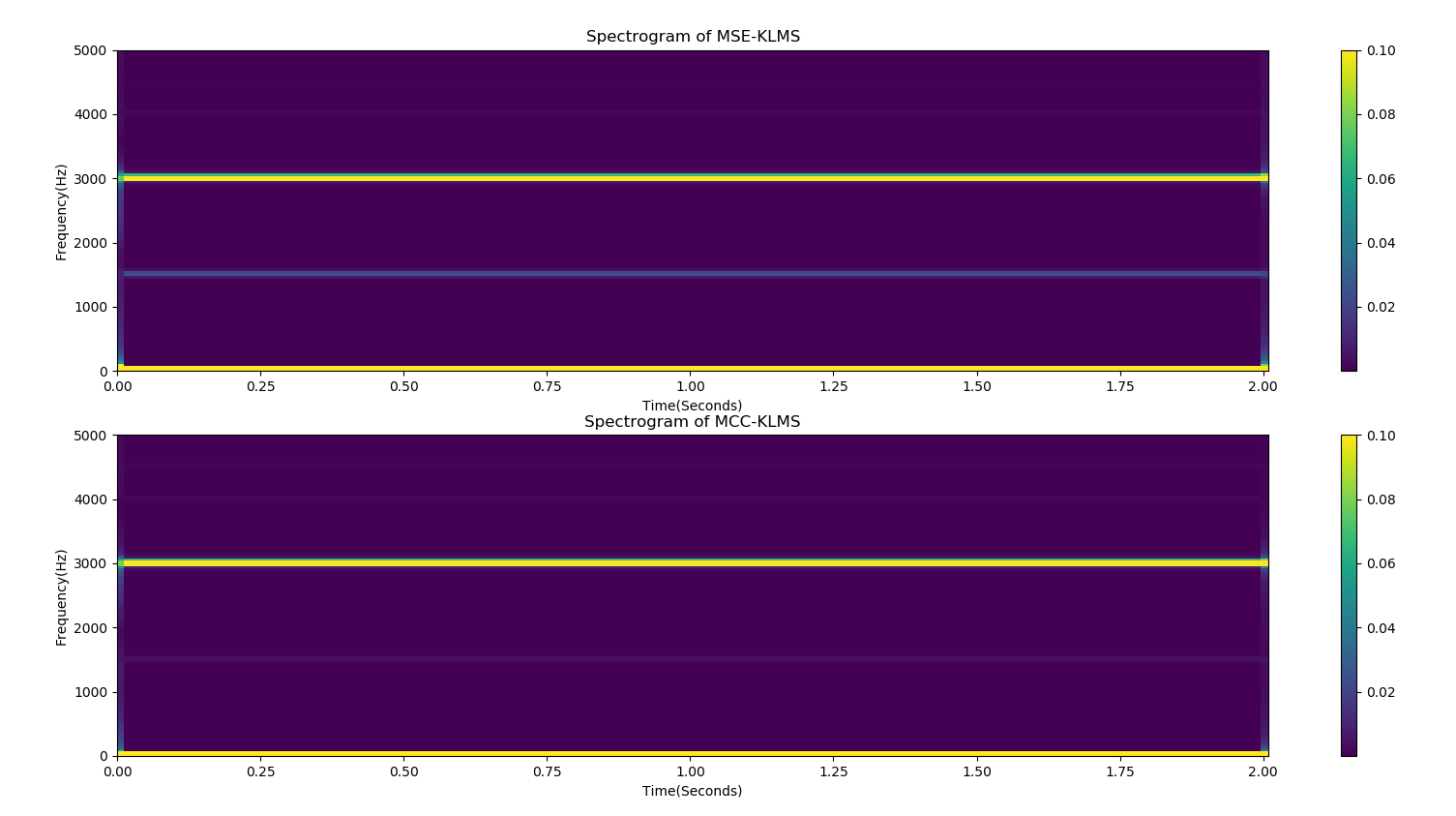


Figure 14 Spectrogram of MSE-KLMS and MCC-KLMS (Frequency of Test Input Signal = 1500Hz)

To compare the performance between MSE-KLMS and MCC-KLMS, I set step size = 0.2, kernel size = 1.6, and for MCC-KLMS I set kernel parameter = 2.

From Figure 12 we can see that the convergence rate of MCC-KLMS is faster and the MSE of MCC-KLMS is lower than that of MSE-KLMS. From Figure 13 and Figure 14 we can see that when the frequency of test input signal = 500Hz and 1500Hz, the output of MCC-KLMS is better than the output of MSE-KLMS. So the performance of MCC-KLMS is better than MSE-KLMS. We can also conclude that the generalization of the MCC-KLMS model is better than MSE-KLMS model when we add a Gaussian mixture noise to the desired response.