Homework 1

Kaifa Lu

Solution to Problem 1:

Step 1: The input to the plant is alpha stable noise with $\alpha = 1.8$. Based on the characteristic function $\emptyset(t) = \exp\left(-|t|^{1.8}\right)$, we can first generate 10,000 samples of the alpha stable noise using *filter_x = levy_stable.rvs(alpha = 1.8, beta = 0, size = 10000)* as the input $x(n)$ to the plant. Also, we plotted the autocorrelation and power-spectrum density (PSD) to confirm the input to the plant.

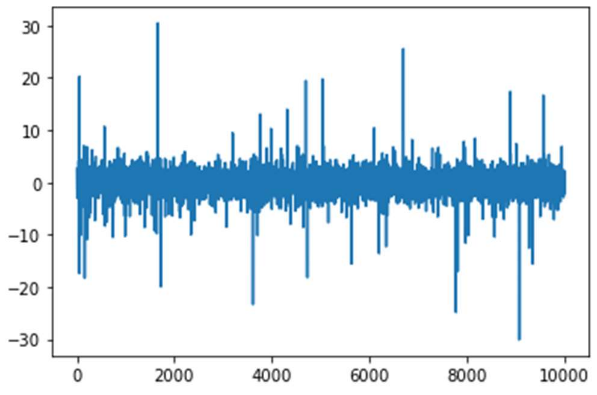

Figure 1 Plot of 10000 samples from the alpha stable noise

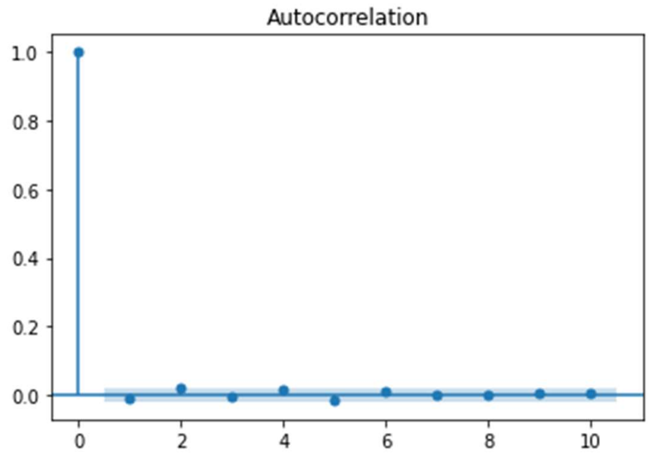

Figure 2 Autocorrelation of the input to the plant

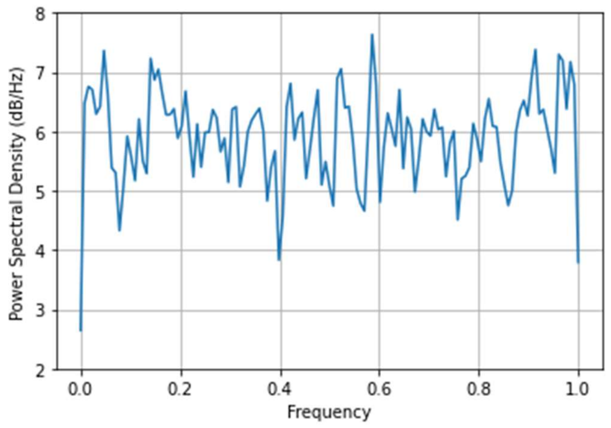

Figure 3 Power Spectrum Density (PSD) of the input to the plant

Step 2: Simplify transfer function to establish the relationship between the input $x(n)$ and output $y(n)$:

$$H(z) = \frac{1 - z^{-10}}{1 - z^{-1}} = \sum_{i=0}^{9} z^{-i}$$
$$y(n) = x(n) \sum_{i=0}^{9} z^{-i}$$
$$y(n) = \sum_{i=0}^{9} x(n-i)$$

Therefore, the plant is a FIR system.

Step 3: The noisy output of the plant is obtained through adding the Gaussian noise of power $N = 0.1$ to the output of transfer function as follows:

$$y(n) = \sum_{i=0}^{9} x(n-i) + Gaussian\ Noise$$

Also, we plotted the Power Spectrum Density of the output with or without Gaussian Noise to the plant.

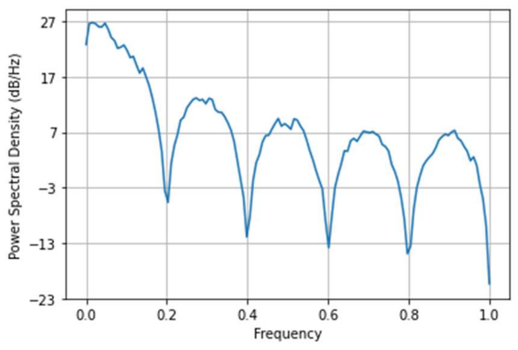

Figure 4 Power Spectrum Density of the output without Gaussian Noise to the plant

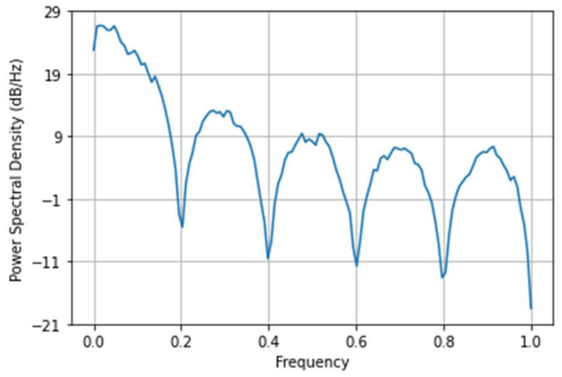

Figure 5 Power Spectrum Density of the output with Gaussian Noise to the plant

Step 4: Once we have obtained the input and output of the plant, we designed a Wiener filter to identify the unknown plant transfer function under different scenarios. For each scenario, (1) we first needed to estimate the autocorrelation function R of the time lagged input $x(n)$ and cross-correlation vector P between the time lagged input $x(n)$ and the desired output $x(n)$. (2) Then the transfer function can be characterized by the optimized weight $w(n) = R^{-1}P$. (3) Finally, we computed the weighted error power $WSNR = 10\log\left(\frac{W^{*T}W^{*}}{(W^{*}-W(n))^{T}(W^{*}-W(n))}\right)$ to compare the accuracy of the system identification. Where $W^{*} = [1,1,1,1,1,1,1,1,1,1]^{T}$

i. <u>Vary filter of order from 5, 15 to 30 and window of size from 100, 500 to 1000</u>

1) **Scenario 1**: filter of order = 5 and window size = 100 samples

$$W(5) = [0.05,1.08,1.09,0.98,0.62]^{T}$$

$$W^{*}\_sized = [1,1,1,1,1]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^{*}}{(W^{*} - W(n))^T(W^{*} - W(n))}\right) = 6.7197$$

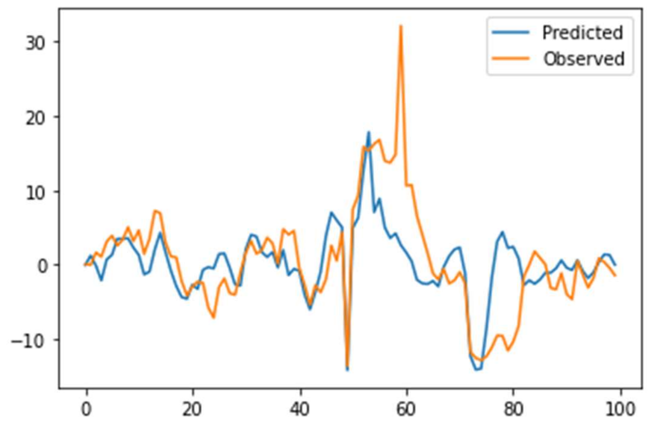

Figure 6 Comparison between the observed and predicted values

2) **Scenario 2**: filter of order = 15 and window size = 100 samples

$$W(15) = \begin{bmatrix} 0.001,0.997,0.999,0.996,0.996,1.000,1.001,1.003,1.001,0.997, \\ 1.000,0.002,0.001,0.001,-0.001 \end{bmatrix}^T$$

$$W^{*}\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^{*}}{(W^{*} - W(n))^T(W^{*} - W(n))}\right) = 6.9942$$

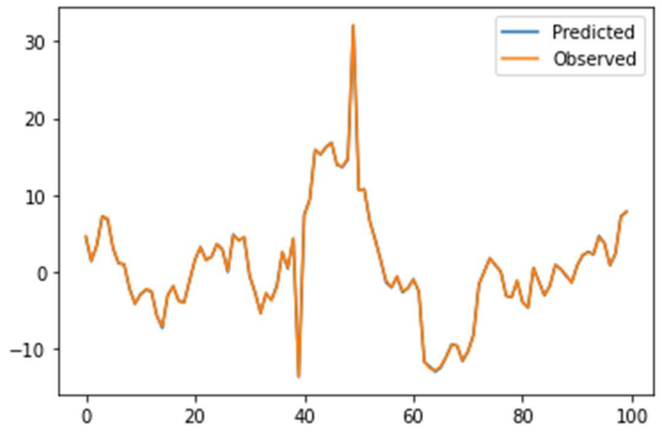

Figure 7 Comparison between the observed and predicted values

3) **Scenario 3**: filter of order = 30 and window size = 100 samples

$$W(30) = \begin{bmatrix} 0.000,0.997,1.000,0.997,0.998,1.000,1.002,1.004,1.002,0.996,1.001 \\ 0.003,0.001,0.002,0.001,0.005,0.003,-0.003,0.002,-0.000,-0.003 \\ 0.001,0.004,-0.000,0.001,0.007,-0.001,-0.005,0.000,0.002 \end{bmatrix}^T$$

$$W^*\_sized = [1,1,1,1,1,1,1,1,1,1,0,\cdots,0]^T$$

$$WSNR = 10\log 10\left(\frac{W^{*T}W^*}{(W^*-W(n))^T(W^*-W(n))}\right) = 6.9852$$

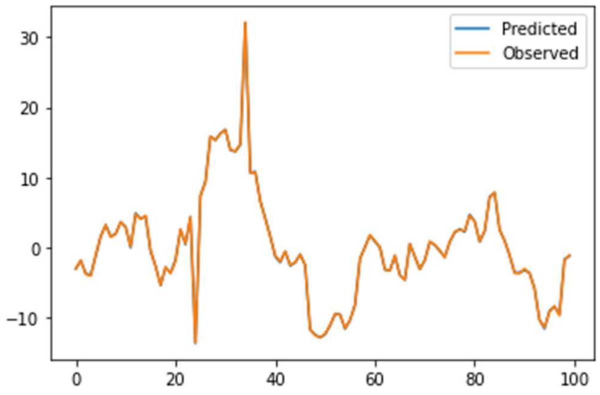

Figure 8 Comparison between the observed and predicted values

4) **Scenario 4**: filter of order = 15 and window size = 500 samples

$$W(15) = \begin{bmatrix} -0.000,0.998,0.999,0.998,0.997,0.999,1.000,0.999,1.001,0.998, \\ 1.000,0.002,0.002,0.000,-0.002 \end{bmatrix}^T$$

$$W^*\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log 10\left(\frac{W^{*T}W^*}{(W^*-W(n))^T(W^*-W(n))}\right) = 6.9890$$

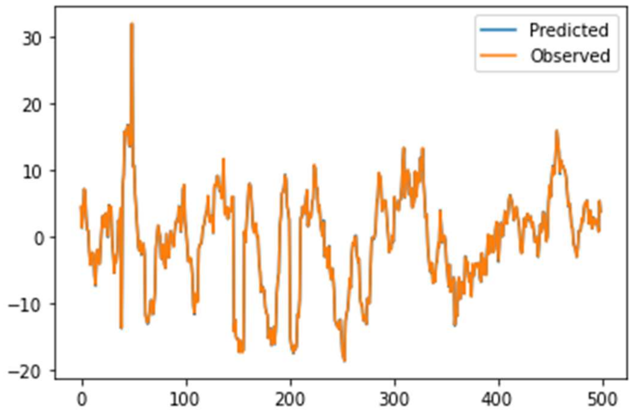

Figure 9 Comparison between the observed and predicted values

5) **Scenario 5**: filter of order = 15 and window size = 1000 samples

$$W(15) = \begin{bmatrix} 0.001,1.000,1.000,0.999,1.000,0.999,1.000,0.997,1.000,1.000, \\ 0.998,0.002,0.001,0.000,-0.002 \end{bmatrix}^T$$

$$W^{*}\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^{*}}{(W^{*}-W(n))^T(W^{*}-W(n))}\right) = 7.0014$$

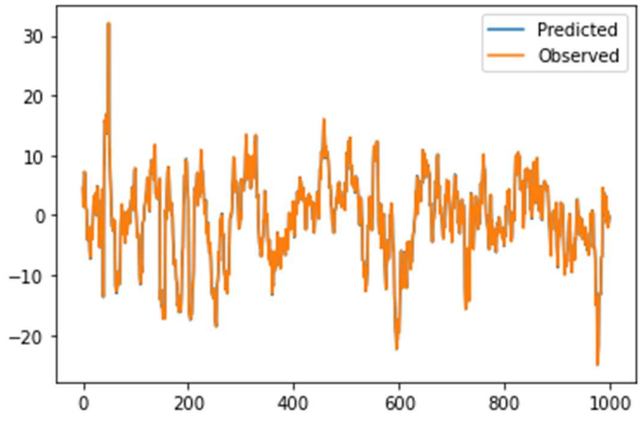

Figure 10 Comparison between the observed and predicted values

ii. Vary different windows of the input (10000 samples)

1) **Scenario 1**: randomly select the starting point of window (e.g., start from the 1000th sample), filter of order = 15 and window size = 500 samples

$$W(15) = \begin{bmatrix} -0.001,0.999,0.996,0.998,1.007,1.005,0.999,0.998,0.998,1.001, \\ 1.001,0.000,-0.002,-0.003,0.001 \end{bmatrix}^T$$

$$W^{*}\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^{*}}{(W^{*}-W(n))^T(W^{*}-W(n))}\right) = 6.9818$$

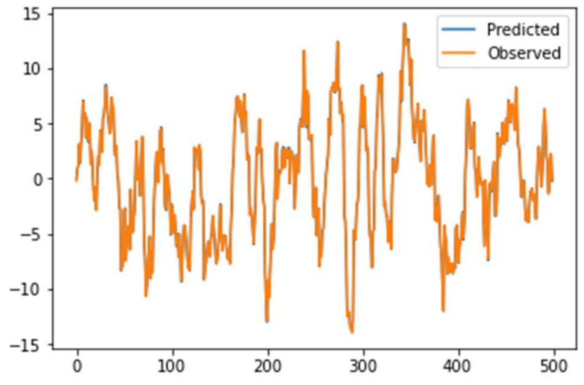

Figure 11 Comparison between the observed and predicted values

iii. Increase the noise N = 0.3, 1.5

1) **Scenario 1**: Noise N = 0.3, filter of order = 15 and window size = 500 samples

$$W(15) = \begin{bmatrix} -0.001,1.000,0.994,0.992,1.004,1.008,1.003,1.003,0.992,0.996, \\ 1.007,-0.005,-0.004,0.006,0.005 \end{bmatrix}^T$$

$$W^{*}\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^{*}}{(W^{*}-W(n))^T(W^{*}-W(n))}\right) = 6.9554$$

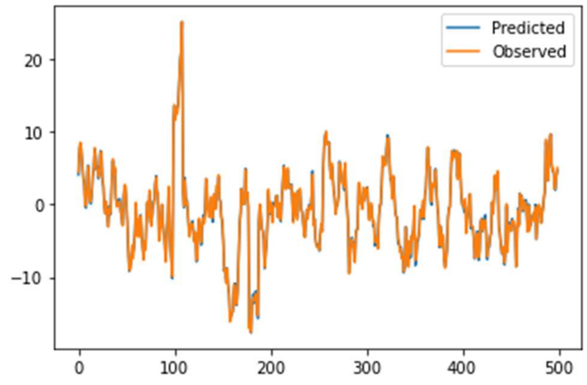

Figure 12 Comparison between the observed and predicted values

2) **Scenario 2**: Noise N = 1.5, filter of order = 15 and window size = 500 samples

$$W(15) = \begin{bmatrix} -0.007,0.975,1.010,1.054,1.003,1.012,1.069,1.000,0.990,0.993, \\ 0.985,0.039,0.002,-0.003,0.027 \end{bmatrix}^T$$

$$W^{*}\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^{*}}{(W^{*}-W(n))^T(W^{*}-W(n))}\right) = 6.9994$$

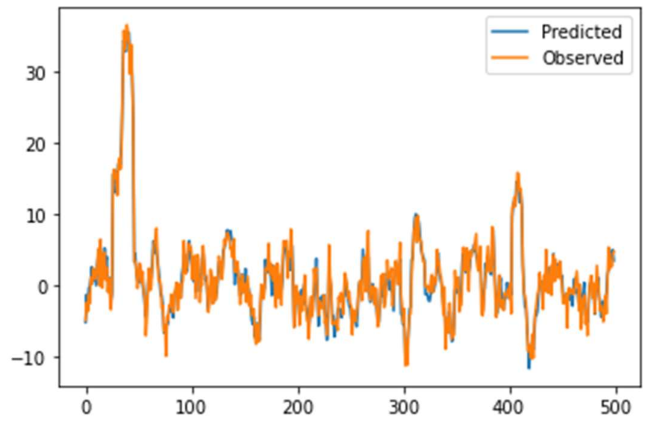

Figure 13 Comparison between the observed and predicted values

Step 5: A follow-up question is that we used the Least Mean Squares (LMS) Adaptive Filter to repeat the above process and identify the unknown plant transfer function under different scenarios. For each scenario, (1) we first initialized the original weight vector to calculate the error $e(n) = d(n) - y(n)$ between the desired and predicted output of the plant. (2) Then we used the gradient descent algorithm to search the optimal weight vector until the minimum. (3) Finally, we computed the weighted error power $WSNR = 10\log\left(\frac{W^{*T}W^*}{(W^*-W(n))^T(W^*-W(n))}\right)$ to compare the accuracy of the system identification.

i. <u>Vary filter of order from 5, 15 to 30 and window of size from 100, 500 to 1000</u>

1) **Scenario 1**: filter of order = 5 and window size = 100 samples

Learning rate = 0.003

$$W(5) = [-0.11,0.82,1.05,1.00,0.94]^T$$

$$W^*\_sized = [1,1,1,1,1]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^*}{(W^*-W(n))^T(W^*-W(n))}\right) = 5.9352$$

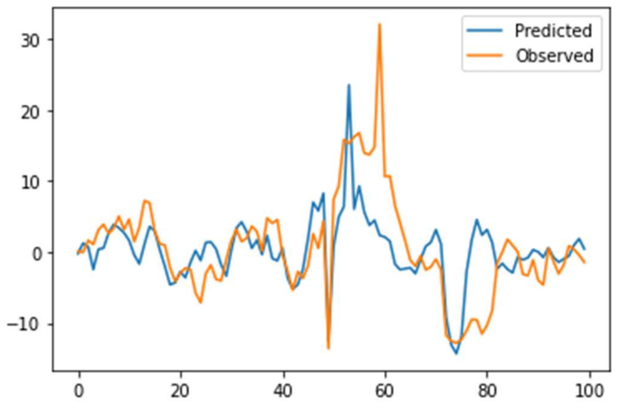

Figure 14 Comparison between the observed and predicted values

2) **Scenario 2**: filter of order = 15 and window size = 100 samples

Learning rate = 0.002

$$W(15) = \begin{bmatrix} -0.03,0.97,1.12,1.16,1.03,1.09,1.02,1.00,1.05, \\ 0.91,1.16,0.14,0.01,0.03,-0.06 \end{bmatrix}^T$$

$$W^{*}\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^{*}}{(W^{*}-W(n))^T(W^{*}-W(n))}\right) = 6.0242$$

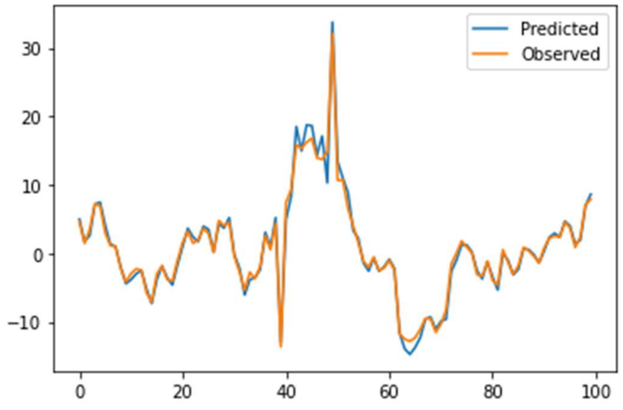

Figure 15 Comparison between the observed and predicted values

3) **Scenario 3**: filter of order = 30 and window size = 100 samples

$$W(30) = \begin{bmatrix} -0.14,0.80,0.90,0.93,0.79,0.92,0.86,0.88,0.90,0.81,1.05,0.12,0.06, \\ 0.08,0.04,0.00,-0.00,-0.01,-0.09,-0.15,-0.13,-0.11,-0.05 \\ -0.04,-0.04,0.04,0.02,-0.01,0.01,-0.03 \end{bmatrix}^T$$

$$W^{*}\_sized = [1,1,1,1,1,1,1,1,1,1,0,\cdots,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^{*}}{(W^{*}-W(n))^T(W^{*}-W(n))}\right) = 5.7121$$

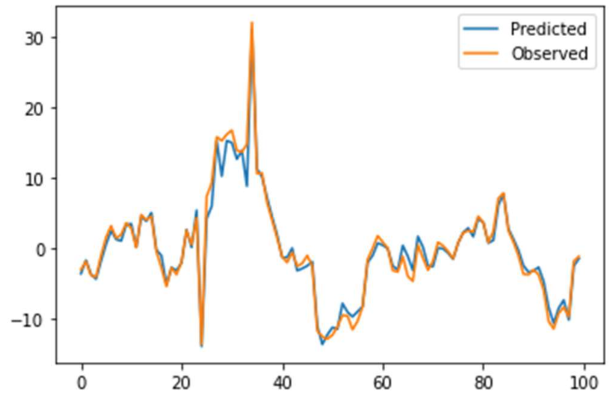

Figure 16 Comparison between the observed and predicted values

4) **Scenario 4**: filter of order = 15 and window size = 500 samples

Learning rate = 0.004

$$W(15) = \begin{bmatrix} 0.003,0.993,1.008,1.008,1.007,0.998,0.991,1.005,0.994, \\ 0.999,0.996,-0.007,0.001,-0.000,0.004 \end{bmatrix}^T$$

$$W^*\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^*}{(W^*-W(n))^T(W^*-W(n))}\right) = 7.0207$$

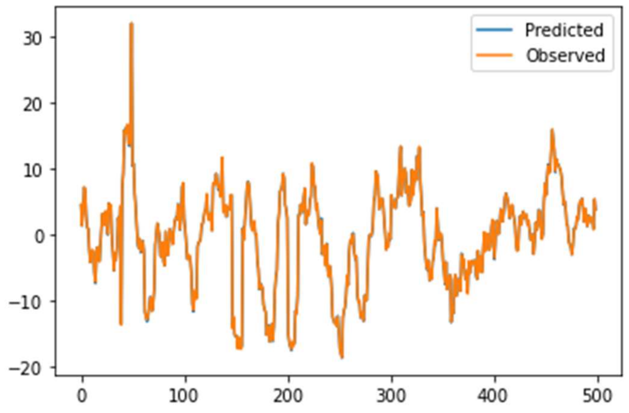

Figure 17 Comparison between the observed and predicted values

5) **Scenario 5**: filter of order = 15 and window size = 1000 samples

Learning rate = 0.016

$$W(15) = \begin{bmatrix} 0.094,0.886,0.993,1.191,0.988,1.027,0.991,0.718,0.970, \\ 0.927,0.916,-0.048,0.051,0.195,-0.148 \end{bmatrix}^T$$

$$W^*\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^*}{(W^*-W(n))^T(W^*-W(n))}\right) = 7.3057$$

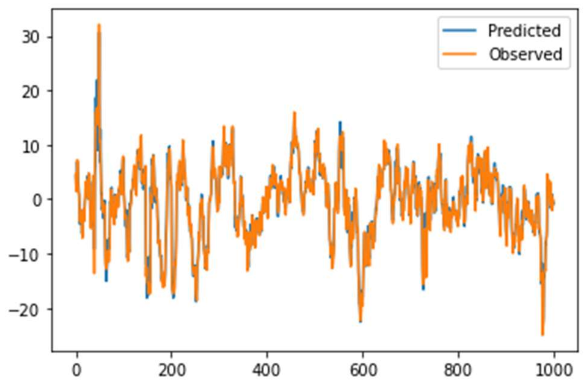

Figure 18 Comparison between the observed and predicted values

ii. Vary different windows of the input (10000 samples)

1) **Scenario 1**: randomly select the starting point of window (e.g., start from the 1000th samples), filter of order = 15 and window size = 500 samples

Learning rate = 0.01

$$W(15)=\begin{bmatrix}-0.012,0.993,0.999,1.003,1.005,1.004,0.992,0.986,0.988,\\ 1.005,1.000,-0.004,-0.004,-0.010,0.014\end{bmatrix}^T$$

$$W^{*}\_sized=[1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR=10\log10\left(\frac{W^{*T}W^{*}}{(W^{*}-W(n))^T(W^{*}-W(n))}\right)=6.9353$$

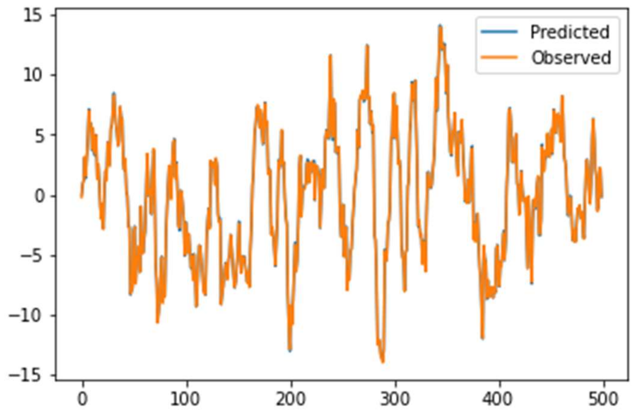

Figure 19 Comparison between the observed and predicted values

iii. Increase the noise N = 0.3, 1.5

1) **Scenario 1**: Noise N = 0.3, filter of order = 15 and window size = 500 samples

Learning rate = 0.01

$$W(15) = \begin{bmatrix} 0.043,1.014,0.991,0.993,0.998,0.995,0.982,1.003,1.012, \\ 1.000,1.051,-0.004,0.034,-0.002,0.026 \end{bmatrix}^T$$

$$W^{*}\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^{*}}{(W^{*}-W(n))^T(W^{*}-W(n))}\right) = 6.9386$$

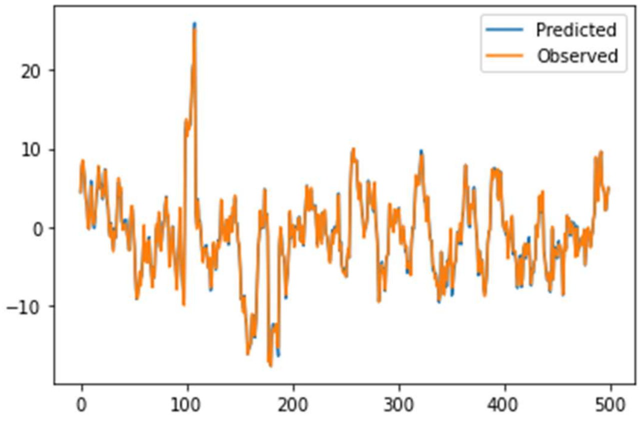

Figure 20 Comparison between the observed and predicted values

2) **Scenario 2**: Noise N = 1.5, filter of order = 15 and window size = 500 samples

Learning rate = 0.01

$$W(15) = \begin{bmatrix} 0.016,0.857,0.905,1.144,0.881,0.961,1.021,1.123,1.190, \\ 0.983,0.816,-0.101,0.050,-0.093,-0.247 \end{bmatrix}^T$$

$$W^{*}\_sized = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0]^T$$

$$WSNR = 10\log10\left(\frac{W^{*T}W^{*}}{(W^{*}-W(n))^T(W^{*}-W(n))}\right) = 7.3681$$

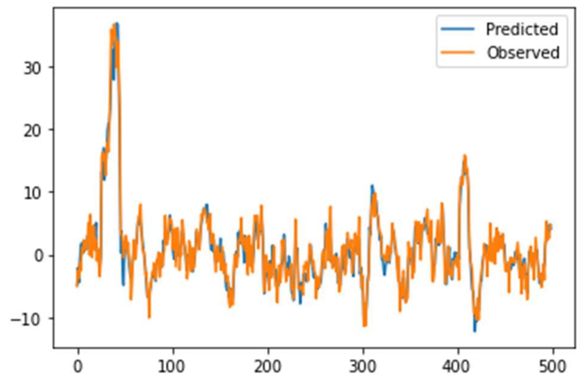

Figure 21 Comparison between the observed and predicted values

Major Findings to Problem 1:

(1) Effect of filter of orders on the performances of Wiener filter and LMS:

It is easily found from the transfer function that the output of the unknown plant at the time (n) is strongly associated with the inputs at the time $(n-9,\cdots,n)$. Therefore, when the filter of order (equaling to 5) is less than 10, the two filters both present relatively large errors between the filter output and the desired values. However, when the filter of order increases to 15 and 30 (both more than 10), the two filters both show very small errors between the filter output and the desired values. It is worth noting that increasing the filter of order from 15 to 30 cannot further improve the filter performances.

(2) Effect of window of sizes on the performances of Wiener filter and LMS:

Increasing the window of sizes can further improve the performances of Wiener filter and LMS adaptive filter. This means that more samples are fed into the two filters, which is more favorable for making the optimized weights output by the two filters much closer to the optimal weight vector behind the transfer function of the unknown plant.

(3) Effect of different windows of the input on the performances of Wiener filter and LMS:

Since the unknown plant is a FIR system, varying windows of the input to the Wiener filter and LMS adaptive filter hardly produced any significant impacts on the performances of the two filter predictors. Specifically, the optimized weights output by the two filters approximately equal the optimal weights behind the transfer function and the WSNR values almost remain unchanged although the window of the input is different.

(4) Effect of increasing the noise N on the performances of Wiener filter and LMS:

Increasing the noise N from 0.1 to 0.3 only produces very limited adverse effects on the performances of the two filters. This means that when the noise N rises from 0.1 to 0.3, the optimized weight vectors output by the two filters still approximately equal to the optimal weight vector. However, when the noise N increases to 1.5, there seems to be some significant deviations between the optimized weight vectors of the two filters and the optimal weight vector behind the transfer function of the unknown plant.

(5) Comparison between the Wiener filter and LMS adaptive filter:

In general, the Wiener filter and LMS adaptive filter both present excellent performances in predicting the output of the unknown plant even though some unexpected noises are added into the plant. However, to some extent, the Wiener filter outperforms the LMS adaptive filter in this time series, particularly when the samples fed into the two filters are limited, as well as when adding more noises into the time series. Furthermore, the prediction performances of the LMS adaptive filter are greatly affected by the learning rate. This means that only when the learning rate is suitably selected can the LMS adaptive filter bring a good prediction performance.

Solution to Problem 2

Step 1: Load and read the time series file *speech.wav* using *wavfile.read('speech.wav')* and plot the data, as well as wide-band sound spectrogram, separately as shown in Figure 1 and Figure 2.

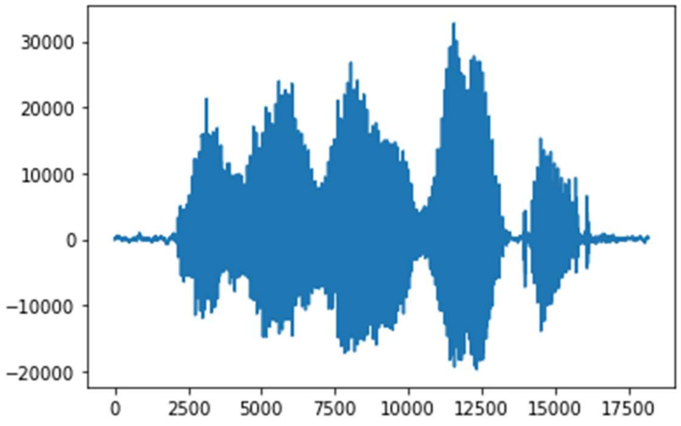

Figure 1 Plot of the utterance data

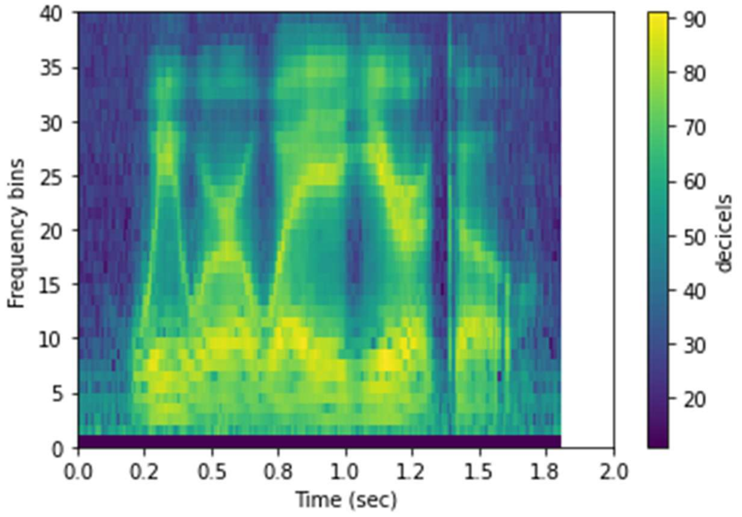

Figure 2 Wide-band sound spectrogram for the utterance

Step 2: Import the utterance data into the Wiener filter for denoising under different window sizes and filter lengths to compare the quality of Wiener predictors in this time series. It is worth noting that since the speech is not stationary, the starting time of one window is very important in determining the weight vector of the Wiener filter. This means that the optimal weight vector may vary greatly when the starting time of one window is different. A good approach to addressing this issue is to attempt different starting time of one window and further compare the performance of the Weiner filter to output the optimal weight vector.

i. <u>Vary filter of order from 6 to 15 and window of size from 100, 200 to 500</u>

1) **Scenario 1**: filter of order = 6 and window size = 100 samples

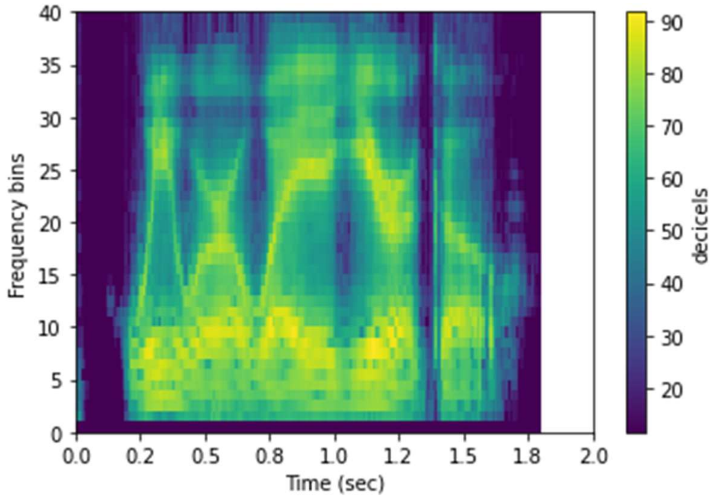

Figure 3 Wide-band sound spectrogram for filtering the utterance

2) **Scenario 2**: filter of order = 15 and window size = 100 samples

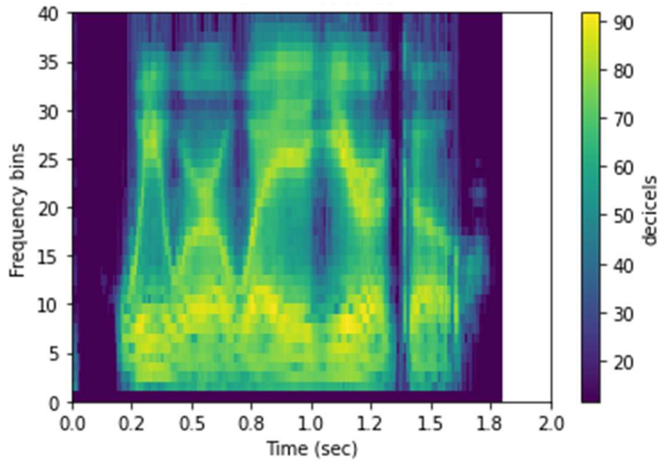

Figure 4 Wide-band sound spectrogram for filtering the utterance

3) **Scenario 3**: filter of order = 15 and window size = 200 samples

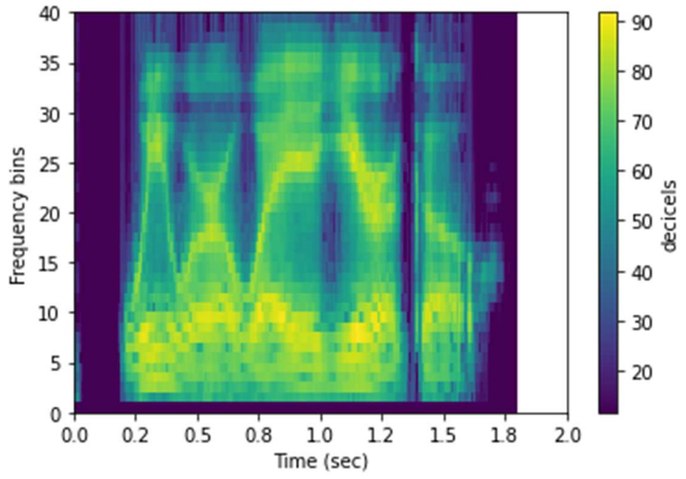

Figure 5 Wide-band sound spectrogram for filtering the utterance

4) **Scenario 4**: filter of order = 15 and window size = 500 samples

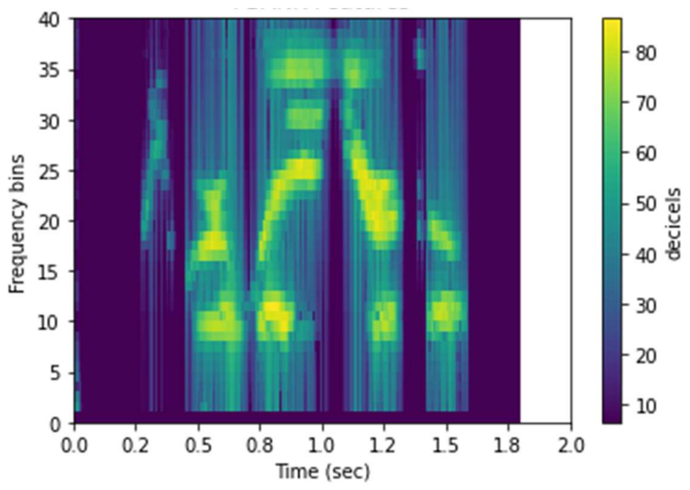

Figure 6 Wide-band sound spectrogram for filtering the utterance

ii. Vary the starting time of window to see how filter parameters vary over time

1) **Scenario 1**: start tine of window from the 1st sample, filter of order = 15 and window size = 500 samples

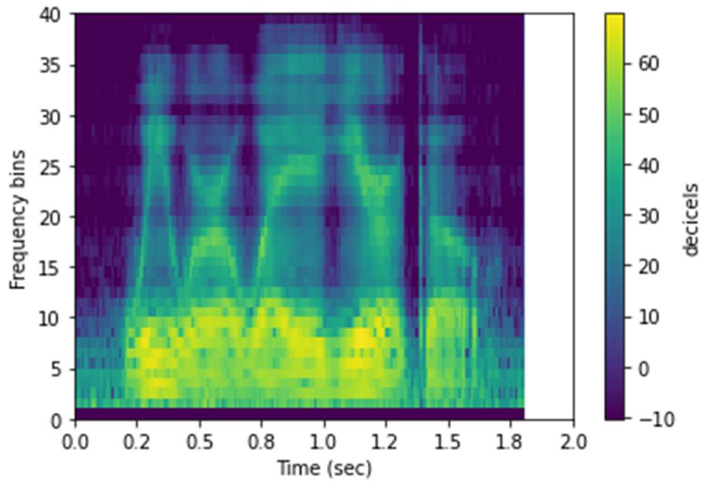

Figure 7 Wide-band sound spectrogram for filtering the utterance

2) **Scenario 2**: start tine of window from the 1000th sample, filter of order = 15 and window size = 500 samples

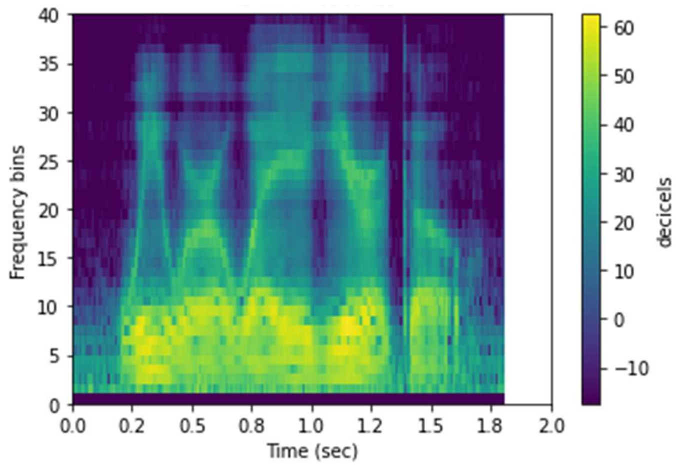

Figure 8 Wide-band sound spectrogram for filtering the utterance

3) **Scenario 3**: start tine of window from the 2000th sample, filter of order = 15 and window size = 500 samples

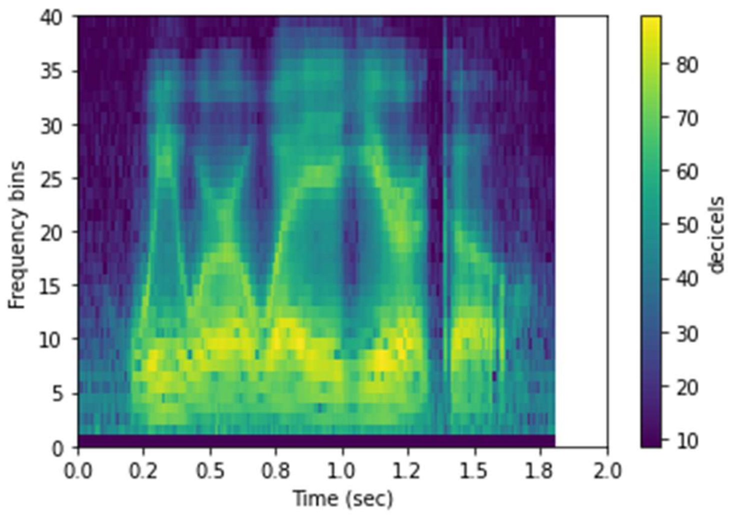

Figure 9 Wide-band sound spectrogram for filtering the utterance

Step 3: Import the utterance data into the LMS adaptive filter for denoising under different window sizes and filter lengths to repeat the above process and compare the quality of LMS predictors in this time series.

i. <u>Vary filter of order from 6 to 15 and window of size from 100, 200 to 500</u>

1) **Scenario 1**: filter of order = 6 and window size = 100 samples

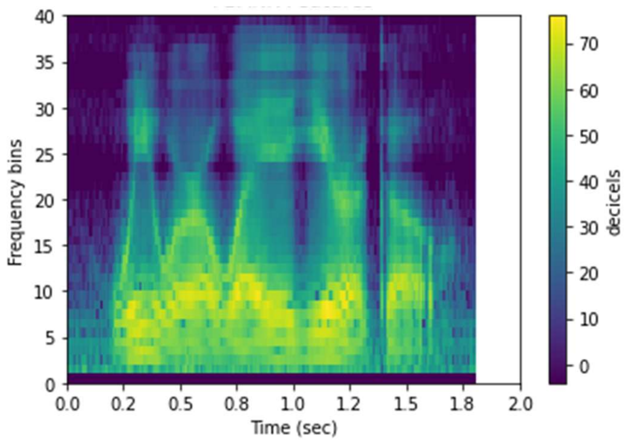

Figure 10 Wide-band sound spectrogram for filtering the utterance

2) **Scenario 2**: filter of order = 15 and window size = 100 samples

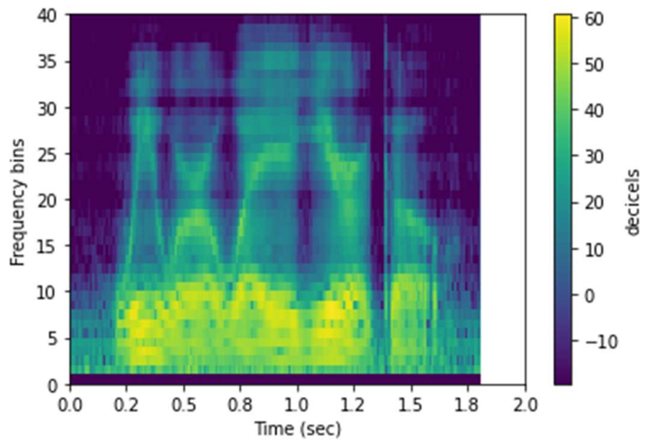

Figure 11 Wide-band sound spectrogram for filtering the utterance

3) **Scenario 3**: filter of order = 15 and window size = 200 samples

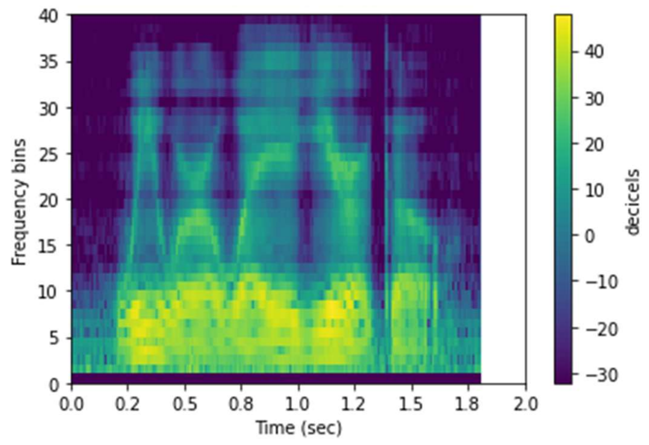

Figure 12 Wide-band sound spectrogram for filtering the utterance

4) **Scenario 4**: filter of order = 15 and window size = 500 samples

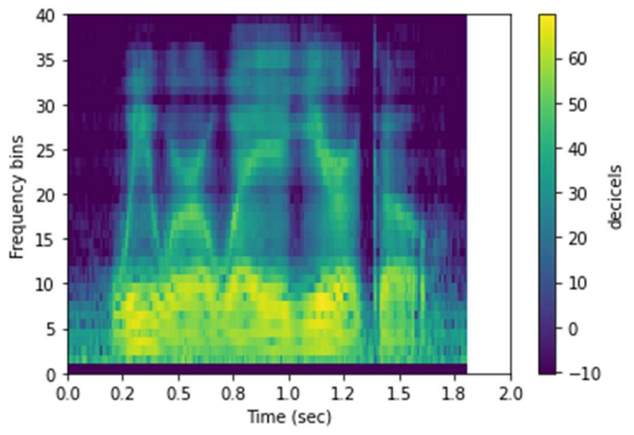

Figure 13 Wide-band sound spectrogram for filtering the utterance

ii. Vary the starting time of window to see how filter parameters vary over time

1) **Scenario 1**: start tine of window from the 1000th sample, filter of order = 15 and window size = 500 samples

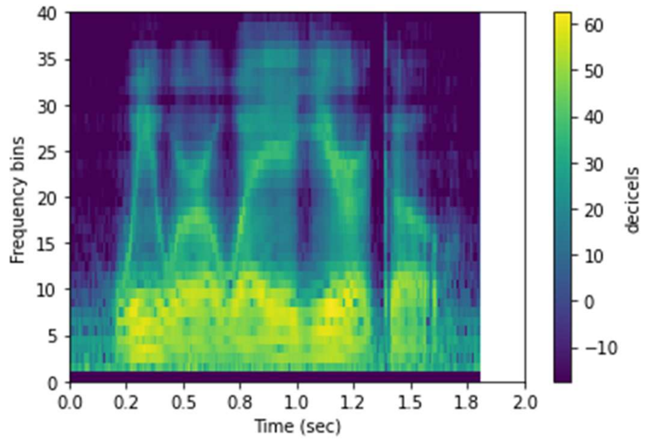

Figure 14 Wide-band sound spectrogram for filtering the utterance

2) **Scenario 2**: start tine of window from the 2000th sample, filter of order = 15 and window size = 500 samples

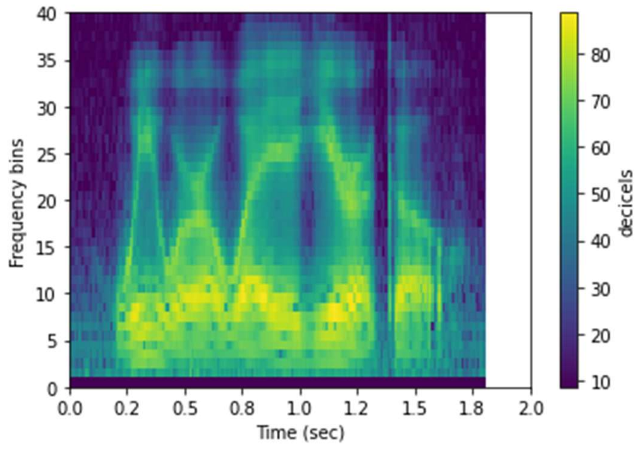

Figure 15 Wide-band sound spectrogram for filtering the utterance

Major Findings to Problem 2:

(1) Effect of filter of orders and window of sizes on the performances of Wiener filter and LMS adaptive filter:

Increasing the filter of orders from 6 to 15 is more favorable for denoising in this time series, particularly for the LMS adaptive filter. In addition, an increase in the window of sizes brings more significant impacts on the performances of the Wiener filter than the LMS adaptive filter. Furthermore, the Wiener filter generally outperforms the LMS adaptive filter in the prediction performances of this time series.

(2) Effects of starting time of windows on the performances of Wiener filter and LMS adaptive filter:

When the starting time of windows varies, the prediction performances of the two filters both exhibit significant variations, mainly attributed to the fact that the speech is non-stationary. Specifically, the two filters both demonstrate better prediction performances, less noises in this time series after filtering, when the window start from the 1000th sample than from the 2000th sample. This mainly results from a higher time-series similarity among the specific pieces of speech when we select the starting time of windows. Under this circumstance, the two filters usually bring a high quality of filter predictors in this time series.

(3) Compare the prediction error with major changes in the spectrogram:

Higher prediction errors of the Wiener filter and LMS adaptive filter are often observed where the major changes in the spectrogram appear. This is because the major changes in the spectrogram usually present significant nonlinearity between adjacent time series, which is difficult for the two filters to capture the nonlinear relationships. To address this issue, we should be careful to select the windows of the time series and mitigate the adverse impacts of the nonlinearity on improving prediction performances of the Wiener filter and LMS adaptive filter. Additionally, when designing the LMS adaptive filter, the selection of the learning rate is vital in promoting the convergence of the filter, and otherwise, the LMS adaptive filter easily gets stuck in poor quality of filter predictors.