**Homework 8** –Biomedical Digital Signal Processing

The submission has two .m files:

1. **Wiener\_RVD1.m:** function definitions and code for the Wiener filter,
2. **Rithika\_Wiener.m:** Implementation of the filter for the questions

This Wiener\_RVD1.m file uses the Wiener filter to remove white noise in the SigNoise and SigLessNoise data through two methods: Decision Directed and Spectral Subtraction. The Wiener filter is very efficient at removing linearly added noise that is uncorrelated with the clean signal.

Decision Directed method uses Signal to Noise Ratio (SNR) as a means of eliminating noise. It rejects low SNR components and aims to preserve higher SNR components. Spectral Subtraction uses the estimation of Power Spectral Density (PSD) of the noisy signal, and removes it from the PSD of the true signal.

The program works in the following stages: (1) Segmentation and Windowing, (2) Fast Fourier Transform of Segments, (3) Estimation of Noise PSD, (4) Detection of Speech and Noise, (5) Removal of Noise, and (6) Alignment and Reconstruction of Segmented Signal

The user-defined inputs to the function are:

1. *signal*: input signal i.e. SigNoisy or SigLessNoisy
2. *fs*: 48828.125 Hz
3. *method:* the method desired for noise removal, input:
   1. ‘Decision’ for Decision Directed, or
   2. ‘Spectral’ for Spectral Subtraction
4. *init:* the length of the initial segment in seconds
5. *win\_type:* Type of window used, input
   1. ‘Ham’ for Hamming,
   2. ‘Rec’ for Rectangular,
   3. ‘Black’ for Blackman,
   4. ‘Kai’ for Kaiser,
   5. ‘Tri’ for Triangular, or
   6. ‘Hann’ for Hann
6. *start\_only:* method to compute noise PSD, input
   1. 1 for using the initial segment only, or
   2. 0 for using the entire signal
7. *rho*: Weight for past and present information for PSD calculation in Spectral Subtraction method,
8. *alpha*: Weight for past and present information for SNR calculation in Decision Directed method,
9. *win\_len*: Length of the window
10. *shift\_p:* Percentage shift of the window

The noise in the two noisy signals is ‘additive white noise’. This is evident from Fig 2, where we can observe that it affects every frequency component in the noisy signals. The noise is at a higher magnitude for SigNoisy data than it is in SigNoiseLess.

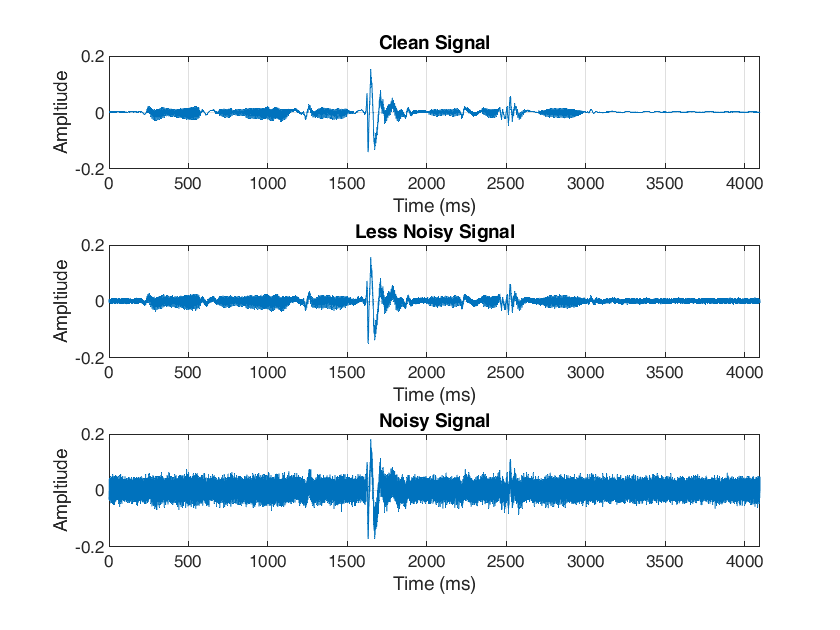


Fig 1: Time domain plots of the noisy, less noisy, and clean signals for comparison

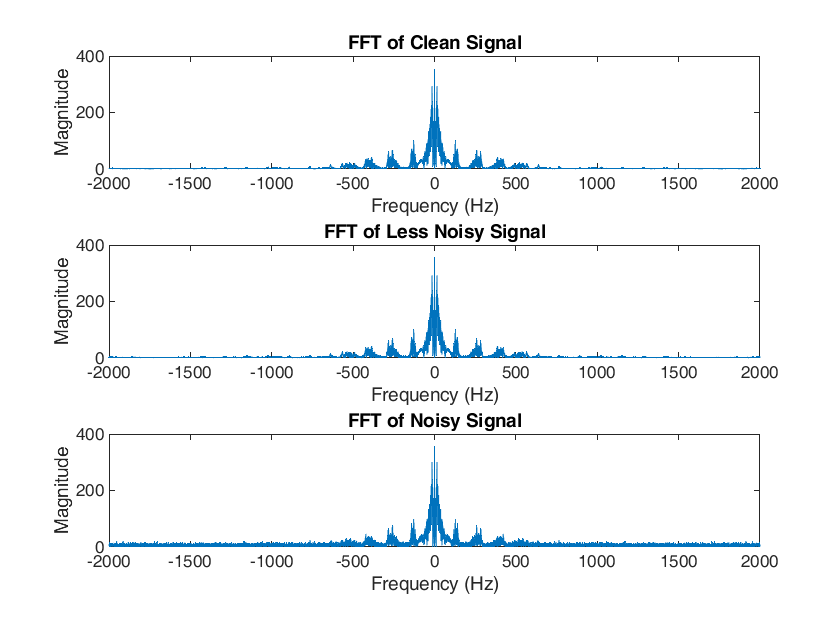


Fig 2: (Zoomed in) Frequency domain plots of the signals

PART I.

* 1. *Decision Directed vs Spectral Subtraction*Fig 3, 4 and 5 compare the performance of the Decision Directed and Spectral Subtraction methods. Fig 3 and Fig 4 depict the filtering of the Noisy Signal and Fig 5 depicts the filtering of the Less Noisy Signal.

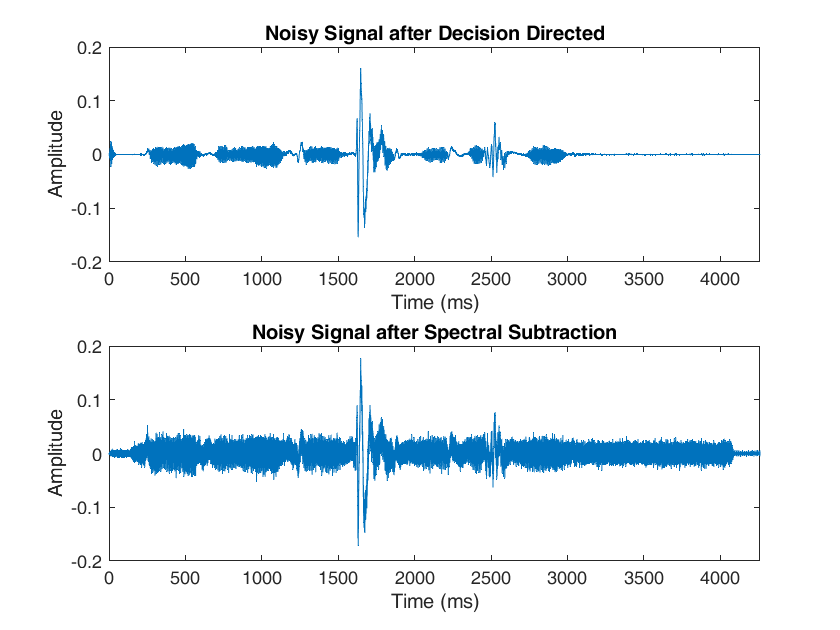
**

Fig 3: Alpha = 0.99 (Decision Directed), Rho = 0.25 (Spectral Subtraction)  
on the Noisy Signal

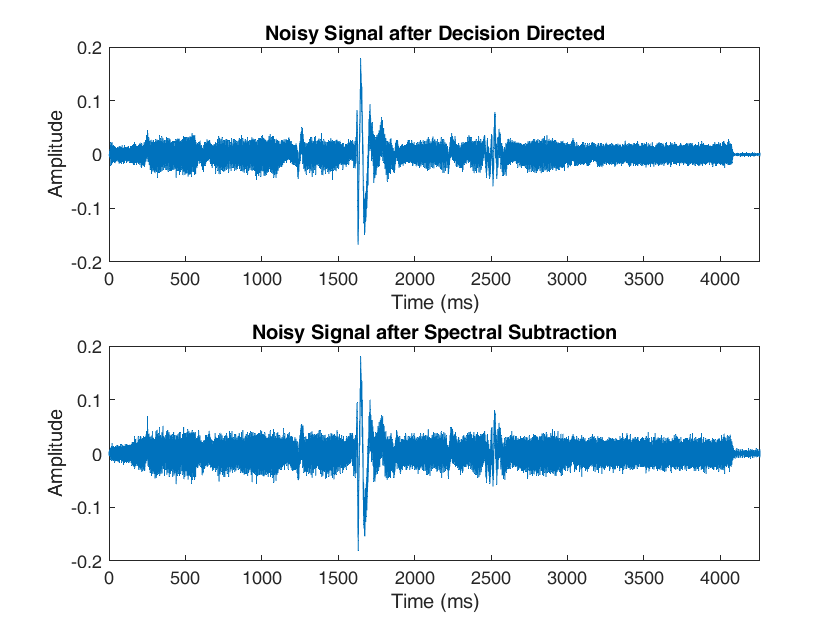
**

Fig 4: Alpha = 0.5 (Decision Directed), Rho = 0.5 (Spectral Subtraction)  
on the Noisy Signal

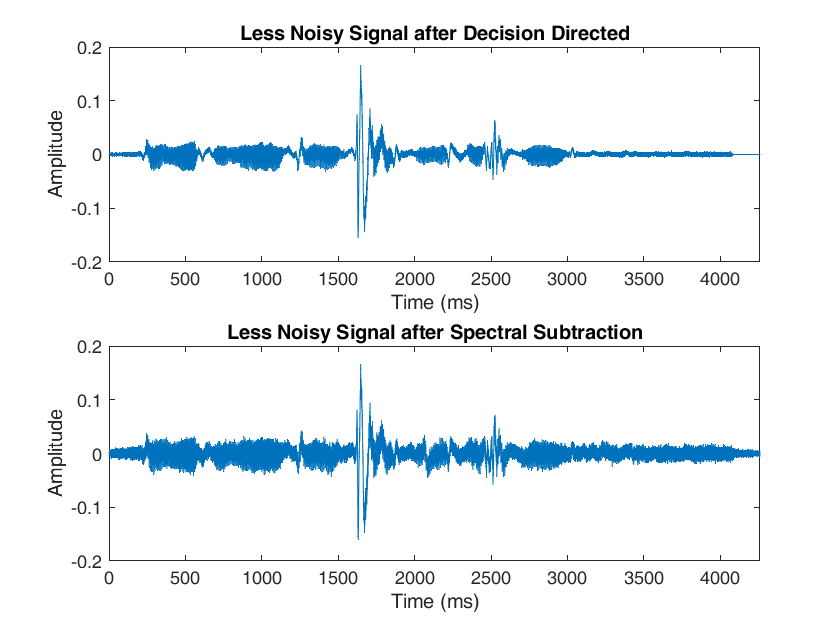
**

Fig 5: Alpha = 0.25 (Decision Directed), Rho = 0.99 (Spectral Subtraction)   
on the Less Noisy signal

The Decision Directed method generally outperforms spectral subtraction and is capable of eliminating more noise. This is because the Decision Directed method relies on the estimation of one variable, i.e. SNR[k], for noise removal. The Spectral Subtraction method, however, requires the estimation of two variables PYY[k] and PNN[k] for noise removal, which is room for more error.

Spectral subtraction can also generate significant amount of distortion of its own when Rho = 1.

* 1. *Computing Noise from the Initial Segment vs Updating Noise with all segments*

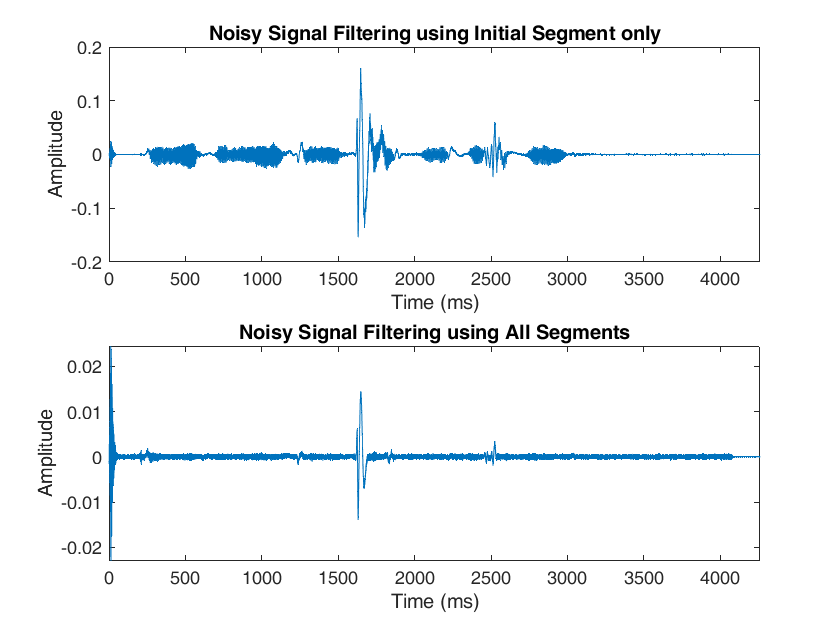


Fig 6: Filtering the Noisy signal by computing noise PSD from just the initial segment vs using all segments

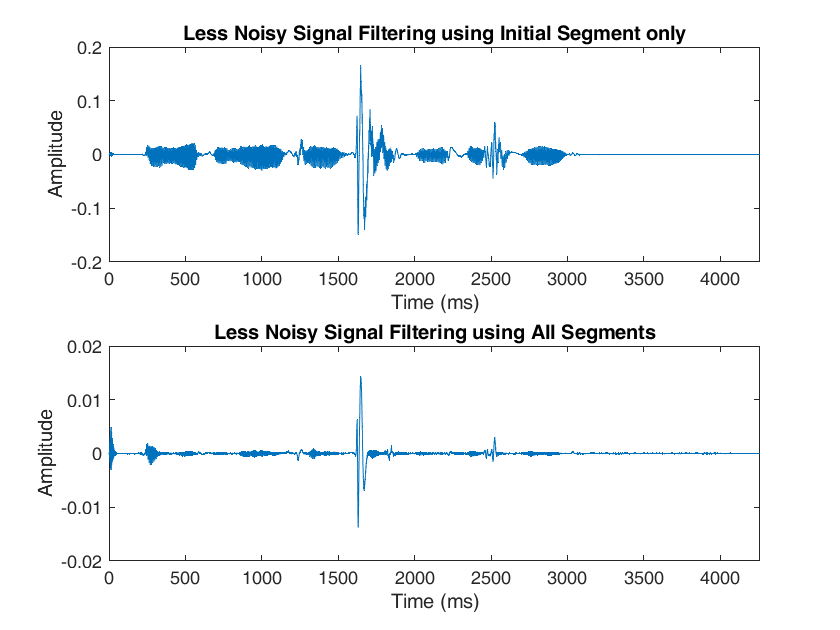


Fig 7: Filtering the Less Noisy signal by computing PSD from just the initial segment vs using all segments

When opting to use the entire signal to characterize the noise, the program continuously updates noise characteristics through all segments. When opting to use just the initial segment, noise characteristics are computed using just the initial segment and considered an accurate representation of the noise in all segments, including ones where there is speech.

We can see in Fig 6 and Fig 7 that using all segments to compute the noise PSD does very little to preserve the signature of the speech signal. In contrast, using just the initial segment does well with noise suppression over the duration of the signal. This may seem counterintuitive, but this is because the noise characteristics are unchanging, i.e. stationary. It is better to use a segment where there is a complete absence of speech signal (and thus just noise) to estimate noise characteristics and extrapolate those characteristics for the entire signal.

Using all segments to update noise characteristics is a better approach when dealing with noise that changes its characteristics with time.

PART II.

1. *Different Windows*

The function defined in **Wiener\_RVD1.m** implements 6 windows. However, since the overlap-add condition has been studied in class with a greater focus on Hamming and Rectangular windows, I have restricted later parts of this analysis to those two windows.

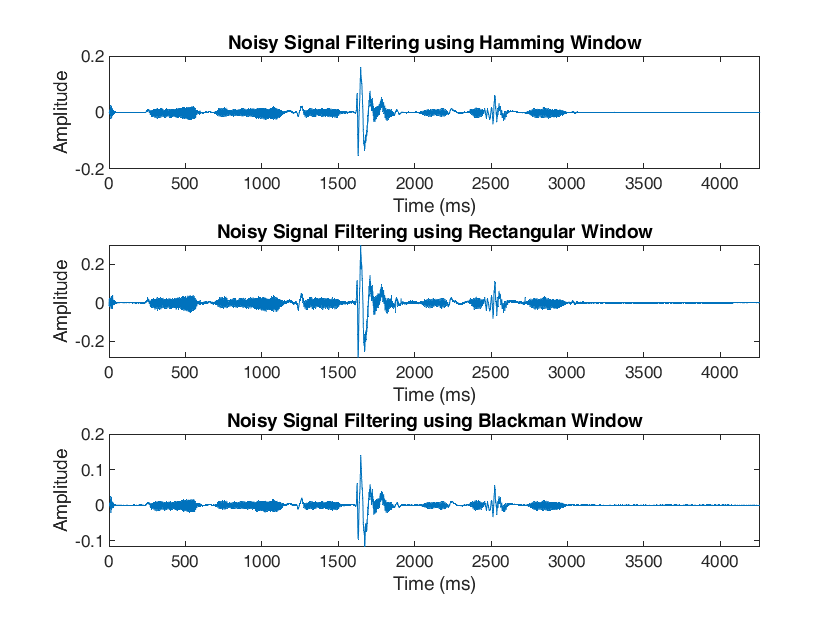


Fig 8: Noisy Signal filtering using Hamming, Rectangular, and Blackman windows for comparison.   
All windows had a length of 0.025 seconds and a shift of 50%.

Under the specified parameters, Hamming window performs the best. Rectangular window does an acceptable job; the speech signal is louder but the window introduces some distortion which is visible in the silent segment after the speech signal ends The Blackman window introduces some attenuation of the speech signal along with distortions. Since the window lengths and the shift percentages are the same, these differences arise from the widths and heights of the various lobes for each window.

*b. Changing window length*

I used 3 window lengths for this part. To emphasize certain results, I used one ideal case and two ‘extreme’ cases i.e., very long window and very short window. The ideal case had a window length of 0.025 seconds. The short window was 0.005 seconds and the long window was 0.15 seconds.

Note: The ‘ideal’ case is simply an approximate window length that can provide good frequency and time resolution simultaneously. This is probably not the optimal value, although it is good enough to filter out almost all of the noise, and it is safe to suggest it is close to the optimal value.

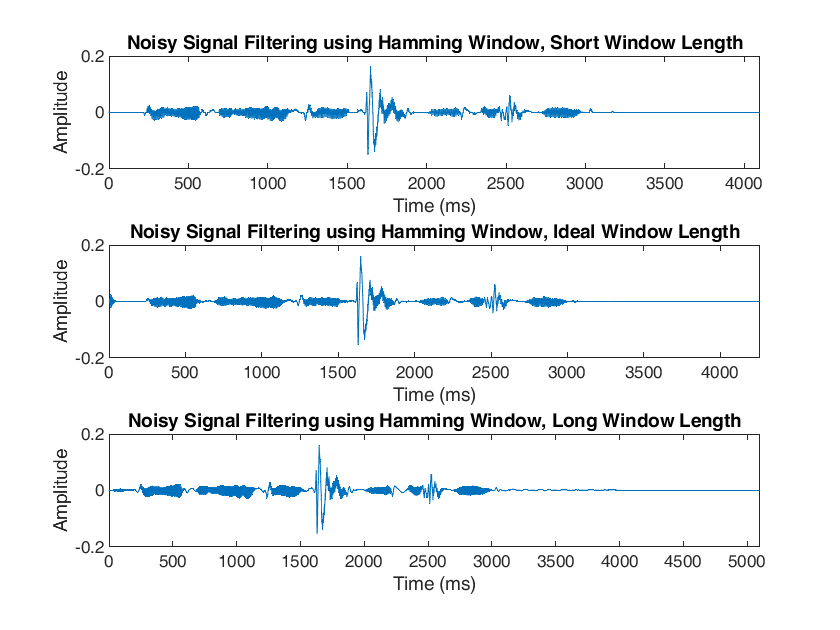


Fig 9. Noisy Signal Filtering with the Hamming window and increasing window lengths

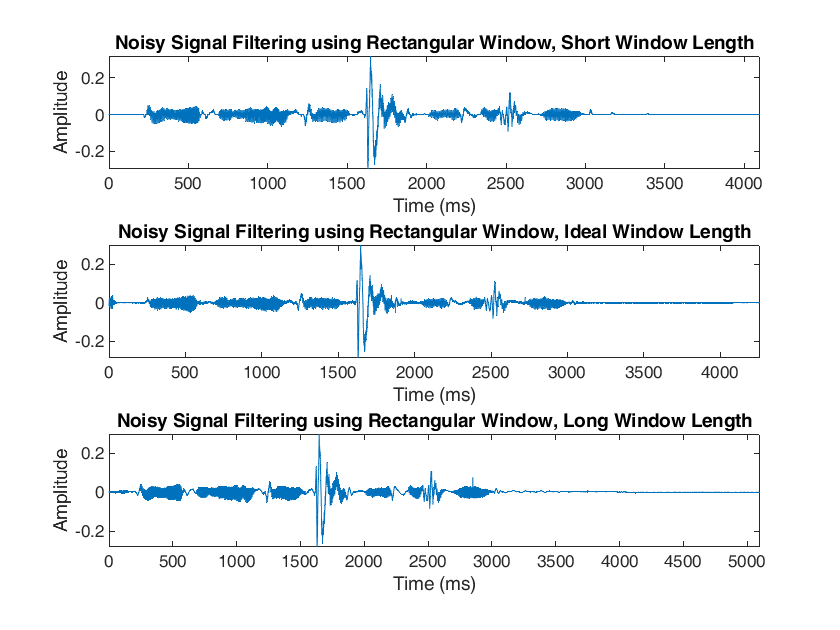


Fig 10. Noisy Signal Filtering with the Rectangular window and increasing window lengths

We can see that as the length of the window increases, the windowing operation seems to negatively impact the filtering performance. This is true; utilizing larger time segments compromises on time resolution of the signal. This is why we can observe ‘shrinking’ of the speech signal in the third plot of Fig 9 and Fig 10. This occurred with the exaggerated window length of 0.15 seconds. The output also increased by a whole second. Further extending the window length to 0.2 seconds demonstrated a near-complete loss of time resolution.

The shorter window appears to do a better job than the ideal window and the long window. The shorter window required significantly more computing time when compared to the other two windows. But the issue doesn’t simply lie in computing time alone. The larger problem with using short windows can be demonstrated in the frequency domain, by taking the FFT of the signal from the shorter window and comparing it to the FFT of the signal from the larger window.

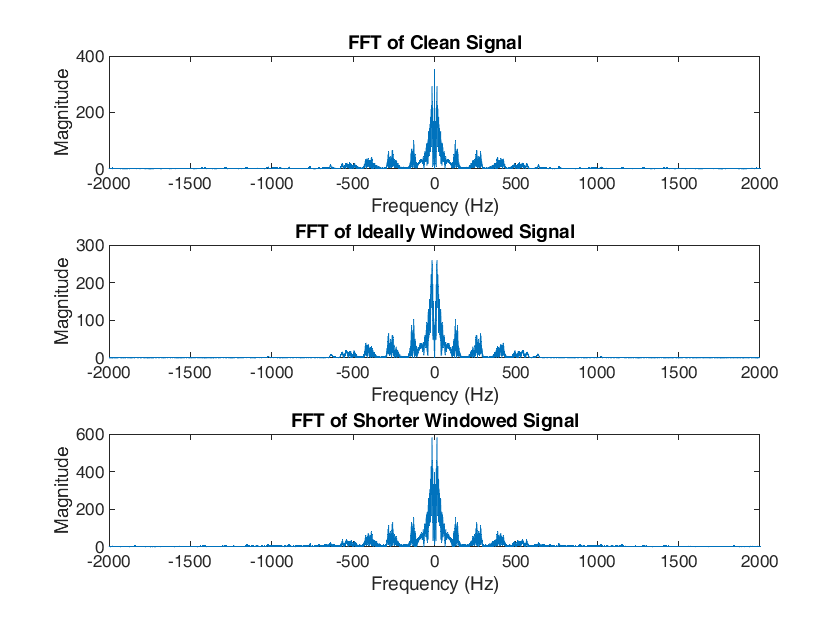


Fig 11. FFTs of the clean signal for comparison to the ideal window and the short window

The ‘ideally’ windowed signal is very similar to the FFT of the clean signal. The time-domain representation shows that this was achieved without any observable time-related distortions. The shorter windowed signal, however, comes with frequency distortion. This is because using extremely short lengths of windows broadens the main lobes in the frequency domain, causing smearing. Thus, shortening the window length beyond a reasonable length does not improve filter performance.

*a. Changing Overlap*

The shift percentage is the amount of distance the window shifts when compared to the total window length. Therefore, a lower shift percentage indicates more overlap while a higher shift percentage indicates less overlap. The higher overlap condition used a shift percentage of 20%, the moderate overlap used a shift of 50% and the lower overlap condition used a shift of 80%. All conditions used the window length of 0.025 seconds.

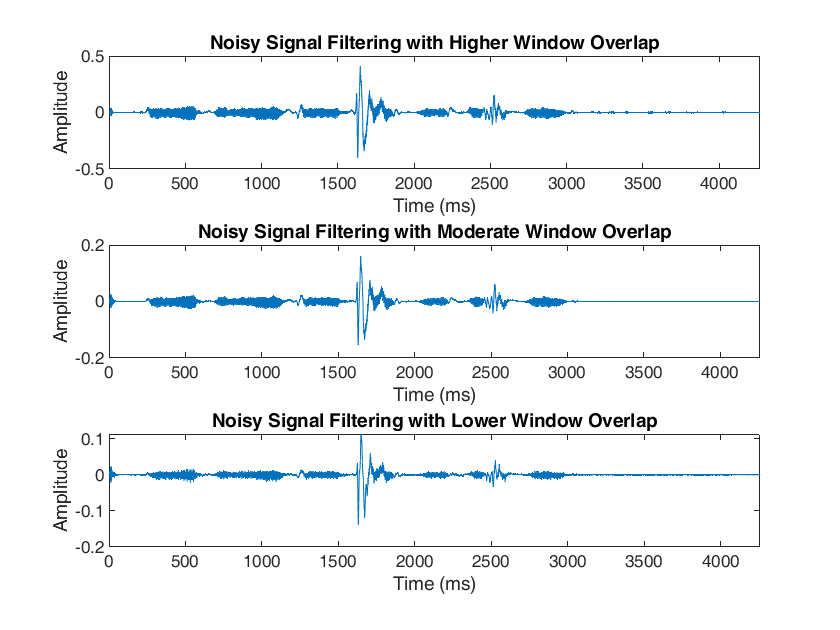


Fig 12: Noisy signal filtering with reducing window overlap (higher shift percentages)

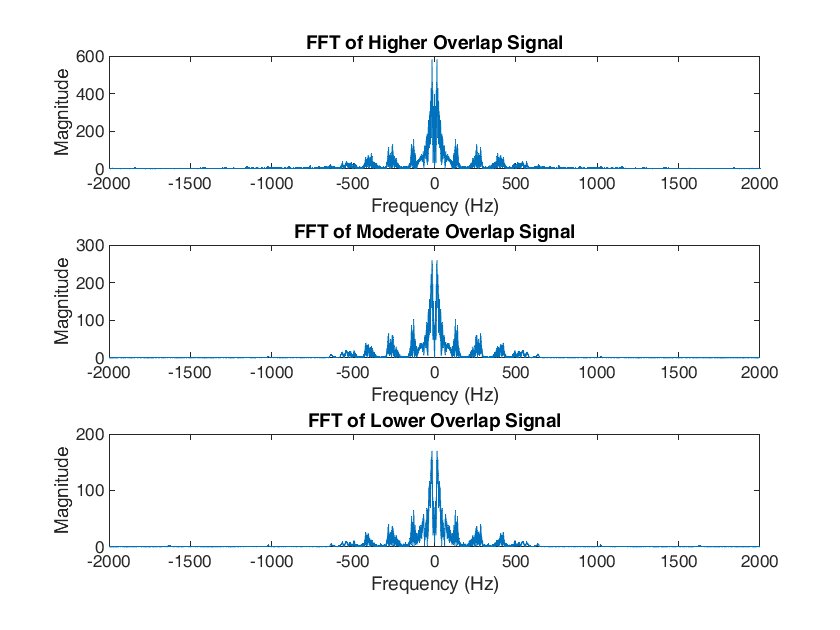


Fig 13: FFTs of Noisy signals with reducing window overlap (higher shift percentages)

High overlap (low shift) increases computation considerably. More samples are shared by adjacent windows and therefore the reconstructed signal has smooth transitions between segments. Low overlap (high shift) decreases computation and shares fewer samples between adjacent windows. This makes individual segments more distinct and can capture transient changes better, at the cost of reduced smoothness between segments. For the signal provided to us, a shift percentage of 50% seems ideal.

The condition for window overlap of any signal must satisfy COLA (Condition for Overlap Add). The sum of all windowed segments must be equal to 1. This is for there to be a perfect reconstruction of the original signal from the window frames.

Thus, changing both parameters (window length and overlap percentage) provides a lot of control with regard to filtering the signals so long as the conditions for overlap are kept in mind and the window lengths are not exaggerated compared to the stationary components of the signal.

PART III.

1. *Changing Alpha*

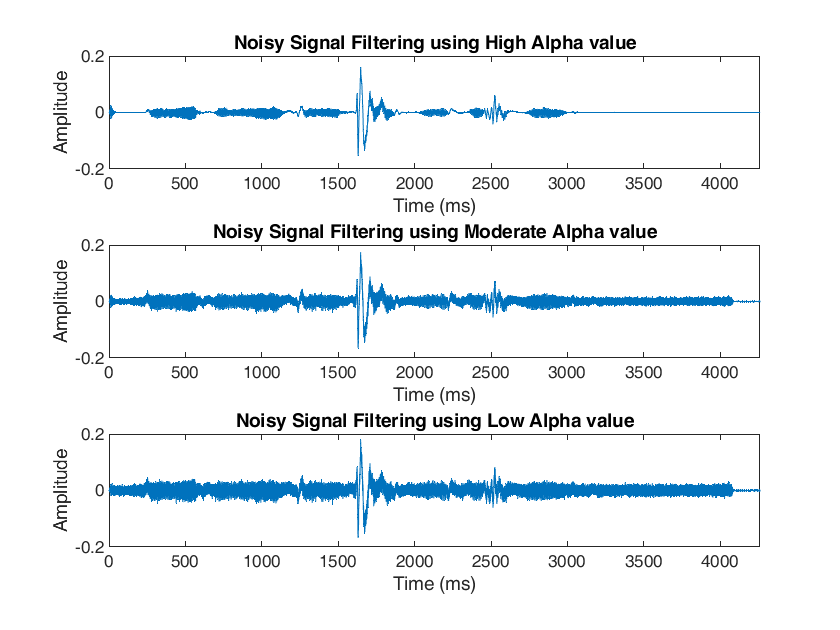


Fig 14: Noisy Signal Filtering through decreasing Alpha Values  
(Alpha = 0.99, Alpha = 0.75, Alpha = 0.25)

The alpha value weighs the importance of past and present information in the estimation of true SNR. The past information, through theory, is more accurately computed but is not representative of current segment’s SNR. The present SNR is temporally accurate but is a non-ideal estimation of SNR (it uses noisy signal Y[k,m] rather than true signal X[k,m] while in its computation).

High alpha values increase the emphasis on accuratly computed past information and decrease the emphasis on inaccurately computed present information. These settings work better for our signal.

The performance of the filter through Decision Directed method on the Noisy signal improves with higher alpha values. The closer alpha is to zero, the lesser smoothening there is.

1. *Changing rho*

PSD for the noise and for the noisy signal for Spectral Subtraction can be computed using Welch’s or periodogram methods. For the noise, however, it is complicated to use Welch’s method as it would require making subsegments of the initial noise segment and averaging them. It would also have considerable bias. Periodiogram has better frequency resolution but exhibits high variance. What could be done instead is a weighted average of the past and present signal information. Averaging segments over time will preserve frequency resolution and decrease bias. We can also control the weight of the past and present information.

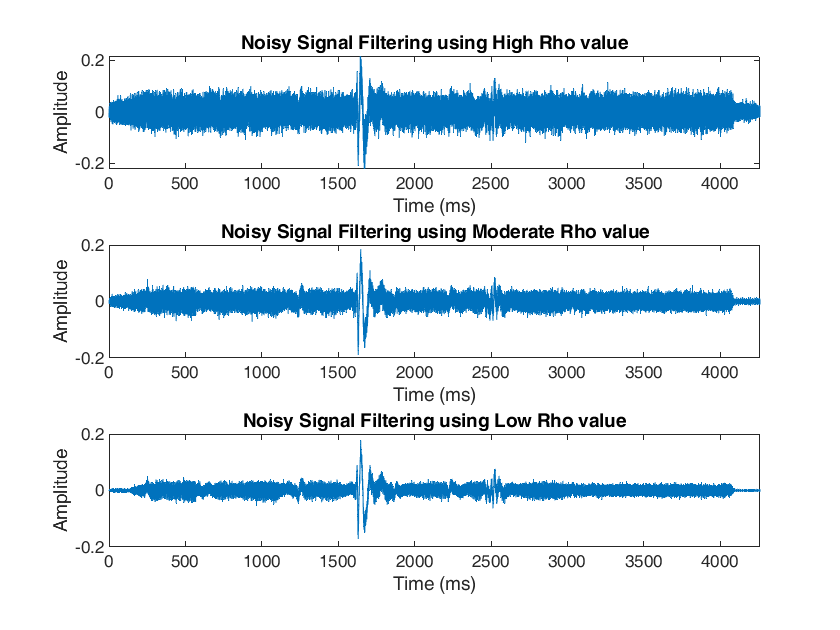


Fig 15: Noisy Signal Filtering through decreasing Rho values  
(Rho = 1, Rho = 0.75, Rho = 0.25)

Rho represents a weight on the importance of past and present information for Spectral Subtraction. A higher rho uses more of the present information for the computation of PSD. No smoothening is done when Rho is equal to 1. Rho of 0.5 indicates an average of past and present information being used. A low rho uses a considerable amount of smoothening by using past PSD information.

PART IV.

1. *SNR*

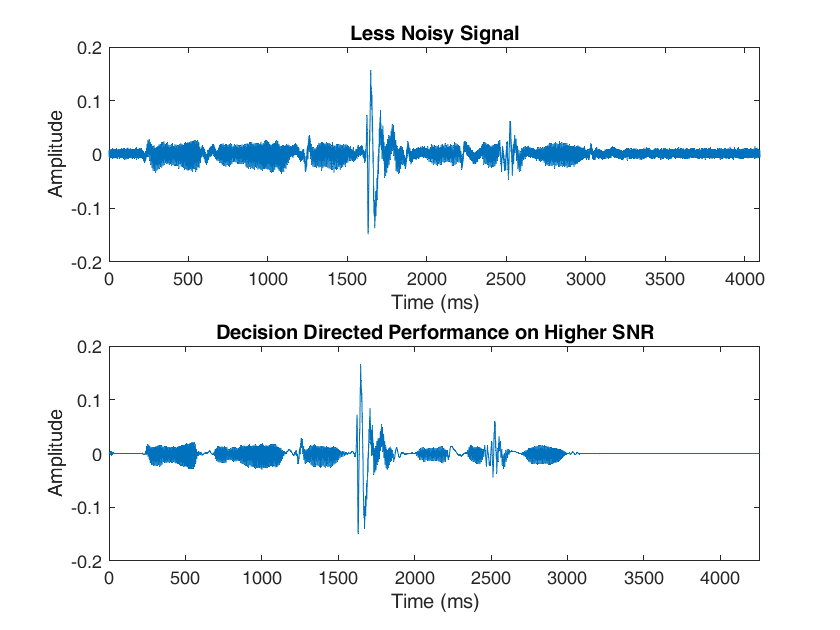


Fig 16: Decision Directed Performance on low SNR

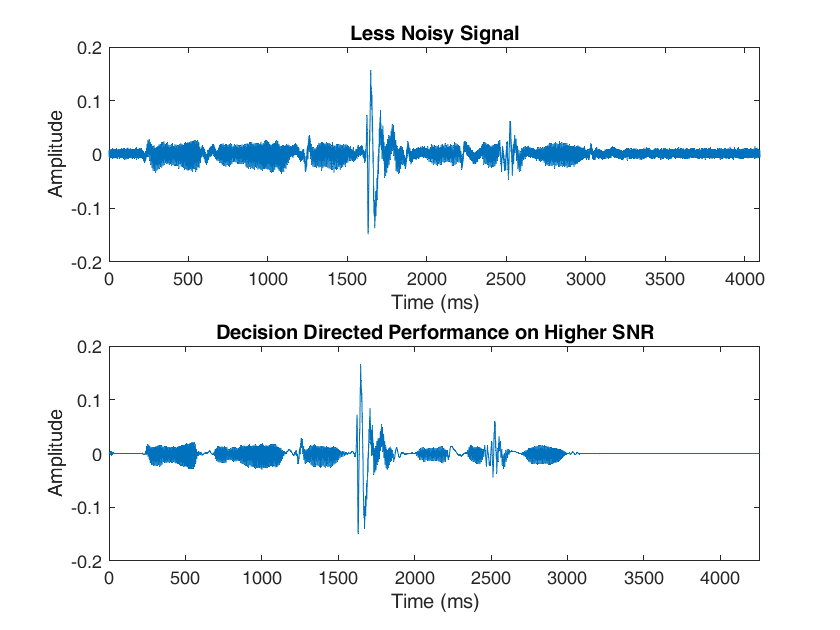


Fig 17: Decision Directed Performance on high SNR

The Wiener filter performs better with low SNR signals. When the SNR is very high, Wiener coefficients approach 1. 0. When the Wiener coefficients approaches 1, the filter characteristics approach those of an all-pass unity filter. This is because when the signal components are considerably greater than the noise components, the Wiener filter needs to retain the information. When the noise components are significantly greater than the signal components, Wiener filter gets rid of it, thereby performing better.

It should be noted that while it technically performs better for lower SNR signals, this is simply because there is significant amount of noise to be removed. It is not unusual for inputs of high SNR signals to come out of a Wiener filter sounding better than inputs of low SNR signals. This depends a lot on the filter’s parameters.

*b. Filtering Clean Signal*

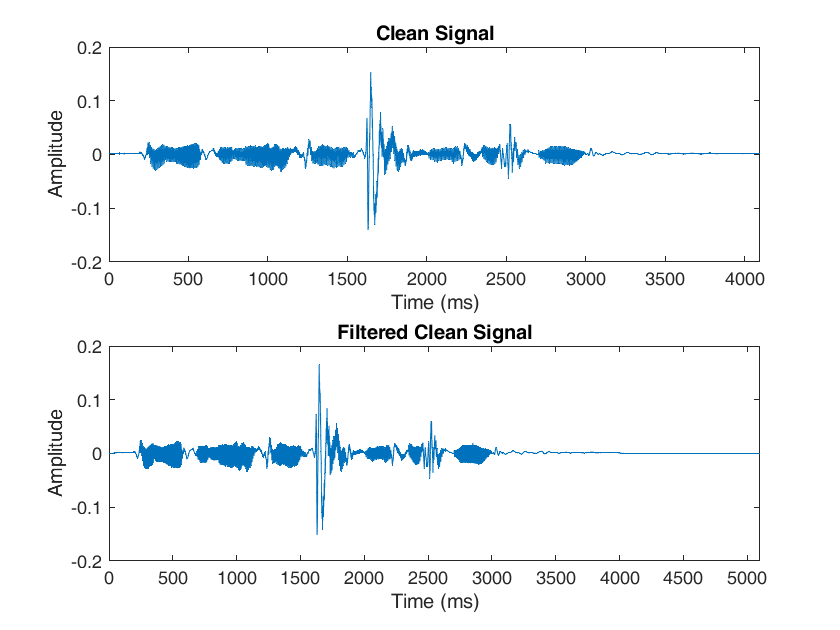


Fig 18: Clean Signal vs Filtered Clean Signal plot

We can see that passing the clean signal through the filter does not output the same signal. Some distortions are observed, including a ‘shrinking’ of the signal. Based on the properties of the filter, other distortions may be introduced. This is because the filter isn’t perfect. Some components of the speech signal may be detected as noise, causing distortions even with a good window length, sufficient zero padding and COLA-satisfying shifting.

**BEST DENOISED SIGNAL:**

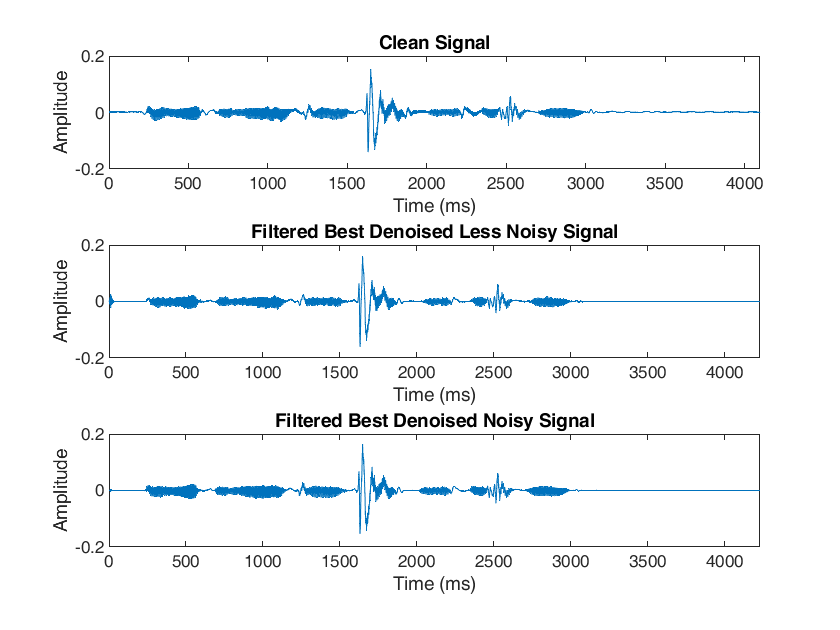
Using the following inferences:

1. Decision Directed is a better method to denoise white noise for this particular signal using a Wiener filter.
2. Using the initial noise segment to compute noise characteristics was better than using the entire segment.
3. A higher Alpha is better for Decision Directed method.
4. The Hamming window seemed to perform best.

I was able to implement the Wiener filter to get rid of most of the noise in the two signals.

The parameters were:

1. method = ‘Decision’
2. init = 0.25,
3. win\_type = ‘Ham’,
4. start\_only = 1,
5. alpha = 0.99,
6. win\_len = 0.020,
7. shift\_p = 0.5

  
Fig 19: The best-denoised signals – Less Noisy and Noisy – compared to the Clean Signal