Time-Frequency Analysis and Enhancement in Speech Processing

Final Project Report, EEE 505 Spring 2022

Amanda Hirte, Elizabeth Jones, Srikar Reddy

Abstract

This project focuses on furthering the research on speech enhancement techniques by exploring the relationship between different Time-Frequency Representations (TFRs) of a particular audio signal and the resulting noise suppression achieved. More specifically, the team focused on the implementation of two TFR methods: the Short Time Fourier Transform (STFT), and the adaptive window STFT (ASTFT). For noise suppression and audio data enhancement, Spectral Subtraction was used to provide an estimation of the original signal for both data from STFT and the ASTFT. The resulting estimation was then compared to the original speech data and error events were counted when the difference was greater than a small specified range, and recorded across a range of signal-to-noise ratio (SNR) values to provide an estimation of the bit error rate (BER). The results showed successful implementation of the TFR methods and the enhancement method in reducing noise in speech files.

1. Introduction

Noise suppression in the field of communication systems is a heavily researched topic, seeing as noise, while problematic, is inevitable when transmitting a signal. The team looked at STFT and ASTFT as the implemented TFR methods. The team created a database of multiple types of audio files, but for the purposes of this report, they will focus on only one of them, a repeated C note file created by the team. To gather speech data, the team analyzed audio files by converting the data into an array of information to be processed. For improved success in the implementation of the spectral subtraction method, 0.*25ms* of a signal with a known noise equal to zero was embedded into the beginning of the audio data to serve as a speech pause. After completing both TFRs, the spectral subtraction implementation utilized the spectrum data to generate a noise power estimation. This estimation was then removed from the noisy signal to generate an estimated noiseless signal based on the embedded speech pause [2]. Analysis was then done to compare the original noiseless signal, the estimated signal produced by spectral subtraction, and the noisy signal in both the time and frequency domains. In leveraging spectral subtraction as a noise suppression technique to enhance speech, the team was able to draw conclusions on the effectiveness of the system.

Conclusions drawn in this project by comparing the original audio file with the post enhancement files will allow the team to provide recommendations on the most advantageous combination of a TFR with the spectral subtraction for the repeated C note file. The conclusions about the repeated C note file will be broadened to encompass the other types of audio signals studied in this project despite not being a focus of the report. These conclusions are important because they will provide insight into the effectiveness of spectral subtraction as a noise reduction method and will provide a guide on how to ensure successful noise reduction when processing an audio file.

This report is organized as follows. The three team members will each provide a subsection in Section *2* detailing their contributions to the project and how they achieved their assigned tasks and objectives. Section *3* will then cover results and discussion. Section 3.1 will provide a detailed description of how the audio files were read into MATLAB and how the team edited the audio files to allow for efficiency and quality analysis. Section 3.2 of this report will detail the implementation of the two chosen TFRs. Section 3.3 describes the spectral subtraction process and how it can be applied to each TFR by providing equations and the specific methodology the team followed to implement it. Section 3.4 will provide information on the analysis data the team gathered, how they gathered it, and figures representing that data to provide an in depth discussion on the findings of the project. Lastly, in the concluding Section 5, the results will be summarized and detailed deductions will be made from the data gathered in the efficiency analysis. The paper "Single channel speech enhancement technique - An implementation on Matlab" by N. Danthi and A. R. Aswatha as well as "Speech Enhancement using Spectral

Subtraction-type Algorithms: A Comparison and Simulation Study" by Navneet Updhyay and Abhijit Karmakar were the two main references the team used to complete this project [3][4].

2. Task Management

2.1 Amanda

Amanda generated a database of audio signals, including a repeated C note, male spoken word and a female spoken word. The audio signals were developed on the Apple software Garageband, using a small keyboard for the repeated C and a high quality microphone for the clean speech spoken word files. She completed the code to create noisy signals at the receiver by using the MATLAB function awgn(). She found that the use of awgn() was most suitable for this project rather than wgn() because the awgn() function uses an input SNR. She proposed that the team could use this SNR input feature to implement an efficiency analysis technique, in which the team would sweep over a range of SNRs and analyze the BER of the sent and recovered signals at the respective SNRs to compare the performance of the two TFRs, STFT and ASTFT, as the noise decreases.

After the completion of the status report, Amanda worked on the implementation of the chosen speech enhancement method, spectral subtraction. She completed research and found papers that showed helpful step-by-step methodology that she was able to follow successfully in her MATLAB code. Her spectral subtraction code was made compatible with the MATLAB coded STFT, and she found a way to graph the noisy signal after spectral subtraction to visually observe the success of the enhancement. Upon trying to implement the spectral subtraction method, she had the idea to add an initial silence to each audio file read into MATLAB to increase the efficiency and success of the spectral subtraction. She spoke with the team about her idea, and they agreed that *0.25ms* of initial empty speech/silence in the audio file would not be detectable to the human ear but would be suitable for the success of the spectral subtraction. Amanda then implemented the code that actually completed this new task.

In the final paper, Amanda wrote the portions on introducing the project as well as expanding on the motivation for completing it. She wrote her section about her tasks as well as the segment on the process and results of spectral subtraction. Once the paper was done she assisted in editing as well as formatting the references with the other team members.

2.2 Elizabeth

Elizabeth completed the STFT implementation for the noiseless signal x and the noisy signal y as one of the TFR methods leveraged in this project. The STFT method was selected as one of the TFRs implemented because it is the only linear TFR that preserves both time and frequency shifts which allows the team to easily switch between the time and frequency domain when doing enhancement and efficiency analysis. She conducted research to find the best method for recovering the magnitude of a signal from the resulting time-frequency output of both STFT and ASTFT. These recovered signals were used post spectral subtraction to provide error analysis which she completed by comparison of the spectral subtraction estimation of the noiseless signal x to the true value.

When completing the STFT implementation, Elizabeth compared the inbuilt STFT function in MATLAB to the time-frequency STFT code provided by Antonia Papandreou-Suppappola in the EEE 505 course. In comparing the resulting outputs of the two approaches, she ultimately chose to implement the STFT with the tfr method provided as it resulted in a more appropriately sized resulting STFT whereas the STFT inbuilt function resulted in information loss when the estimations were recovered. Thus, when comparing the data vectors for the signals and the estimations, the sizes were not compatible. When implementing this function, Elizabeth also had to do research and do a case study on which window would provide the best results. After testing a variation of windows including Bartlett and Blackman, a hamming window with the length of the spectral pause added to the signal was selected. The window length was determined by the speech pauses in order to make the noise estimation discussed in Section 3 easiest. In addition, she needed to identify the best range for the error cases for the spectral subtraction estimations using both ASTFT data and the STFT data. This was done by comparing the average changes to the overall error count and modifying them to smaller values to get as close to a pure match as possible.

For the report, Elizabeth wrote about her tasking, the results of speech data generation, and the discussion section. She also compiled all of the coding algorithms developed by the team and ensured compatibility across variables, ensured proper formatting of plots in the code, and aided in developing the proposed paper structure.

2.3 Srikar

Srikar had the responsibility of generating the code to find and plot the spectrogram of the clean and noisy audio signals, to implement the ASTFT algorithm in the overall study, and to determine the BER of the estimated results taken from the spectral subtraction implementation.

Using the original clean signal data and the noisy signal data, the spectrogram , which is the square of the absolute value of the Fourier transform, was found. The spectrogram provided important information on the magnitude of the noise added to the original clean signal and provided a reference point for later analysis [1]. Further the spectrogram serves as a comparison dataset to the STFT and ASTFT TFRs that would be implemented in this project. In addition, he did an adequate literature review and gathered information on the ideal performances of the STFT and ASTFT TFR methods. All the necessary plots were documented and displayed for further comparison.

Using the same clean and noisy audio signals, the ASTFT was implemented in MATLAB. The ASTFT chosen in this project was dependent on the Constant Q Transform (CQT). The CQT calculates and implements a varying window length based on the signal, and applying this to the STFT results in varying time-frequency resolution. This is beneficial because the time-frequency resolution is a tradeoff relationship, meaning that a signal cannot have perfect resolution in both time and frequency. A window is often chosen such that the resolution for both reaches some sort of compromise. An ASTFT with a changing window size allows one to receive better time resolution at higher frequencies and a high frequency resolution at low frequencies rather than compromise either. Srikar used MATLAB to apply the constant Q transform to the noisey and generate a time-frequency plot. The resulting noise suppressed data, which was acquired by applying spectral subtraction to the spectral information recorded by both STFT and ASTFT, was compared to the original noiseless data and a bit rate was calculated. This BER was then calculated over different SNRs, ranging from *1dB* to a *20dB*. Further, the BER vs the SNR for both the cases were plotted on a single semi-logarithmic plot.

Sikar wrote the abstract, Section 3.2, the conclusion in the report, and aided in the final editing.

3. Results and Discussion

3.1 Speech Data Generation

In order to generate speech data for analysis, the team first developed a database of clean audio files. For the purposes of this study, clean files were determined to have no additive noise. In order to verify that there was no noise, the audio file was added to MATLAB, the data was extracted, and then the signal was plotted to initially verify that the baseline results were within the expected range for the three audio files previously mentioned. The audio files were all leveraged when verifying the functionality of the TFR and spectral subtraction algorithms were implemented, but for the purposes of the remainder of the results and discussion, the focus will be on the repeated C file. The reason for choosing this particular audio file as a basis for this paper is that the C note has a known center frequency of 261.63 Hz. Meaning that the audio file does not have an excessive amount of variation in frequency which makes it easier for the human eye to observe the effectiveness of the methods applied to it.

Following the generation of the audio files, the inbuilt matlab function audioread() was used to read the audio file and convert it into a vector of audio data that could be used in the analysis. In order to verify that the audioread function was collecting the signal information correctly, the team plotted the audio data in the time domain, the frequency domain via the Fourier transform, and lastly, the power spectrogram of the audio file.

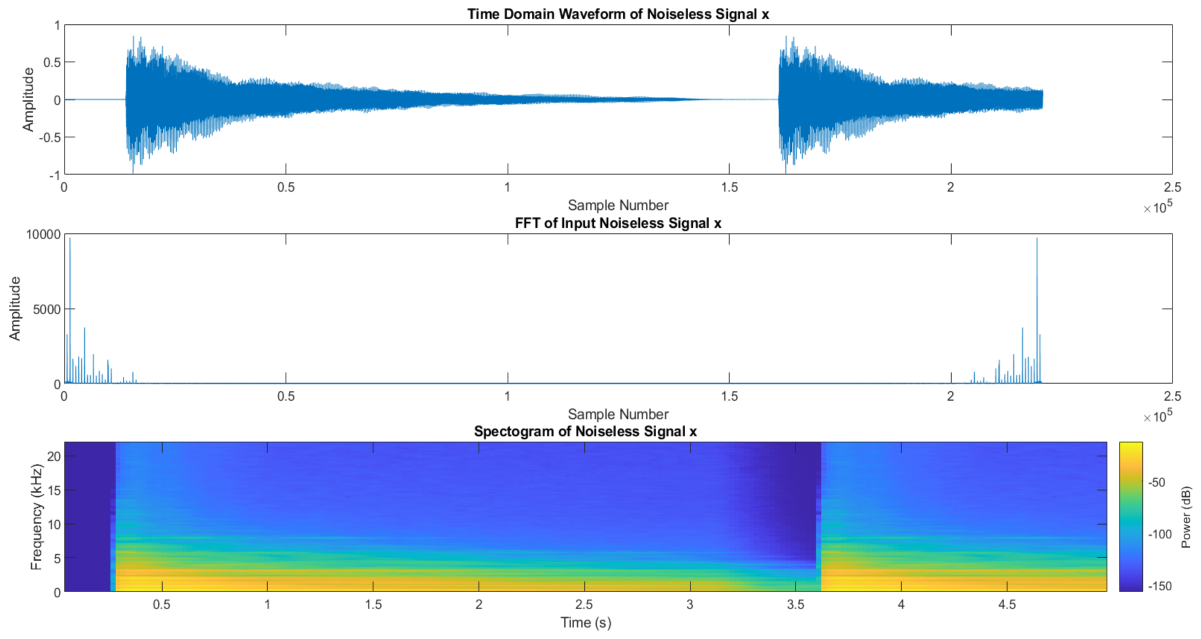


Figure (1): Noiseless Signal x Time Domain, Fourier Transform, and Spectrogram

As seen in the time domain waveform plot of Figure(1), there are two pulses of signal information, which is when the C notes were played in the original audio file. Additionally as seen in Figure(1) the spectrogram has the highest power concentrations at frequency values around *0.2kHz* which verifies that the audio file has successfully been converted into a data array ‘x’. As seen in the time domain waveform plot, there are two pulses of signal information when the C notes were played in the original audio file. Following the successful generation of the audio data, noise was introduced into the system using the agwn() function in MATLAB which introduced additive white gaussian noise into the system [3], meaning that the noisy signal, denoted y, would be equal to the original signal, x, added with the noise. However, in order to compare the estimated noise discussed in Section 3, the known noise value must be calculated. The results were plotted to provide the baseline reference for the signal with noise added, referenced as “noisy signal.” The results of these plots verify that there is noise added to the overall signal while maintaining the integrity of the repeated C file.

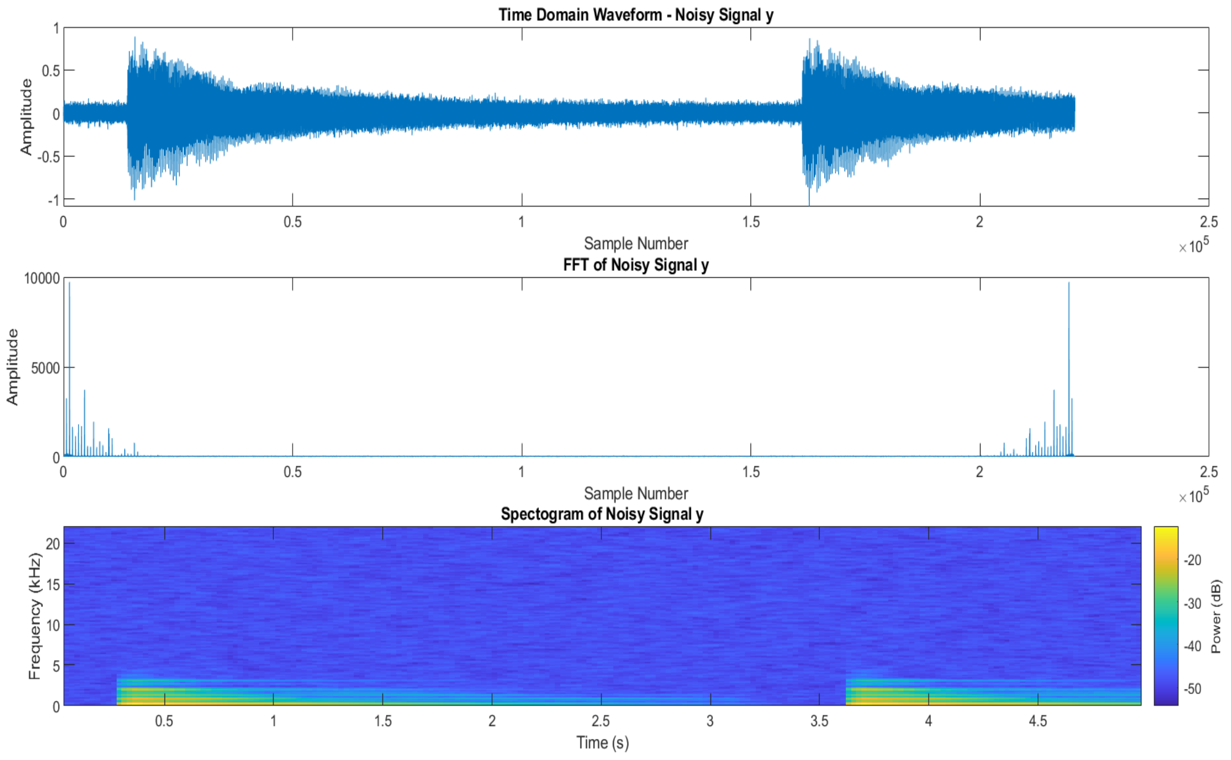


Figure (2): Noiseless Signal x Time Domain, Fourier Transform, and Spectrogram

3.2 TFR Implementation

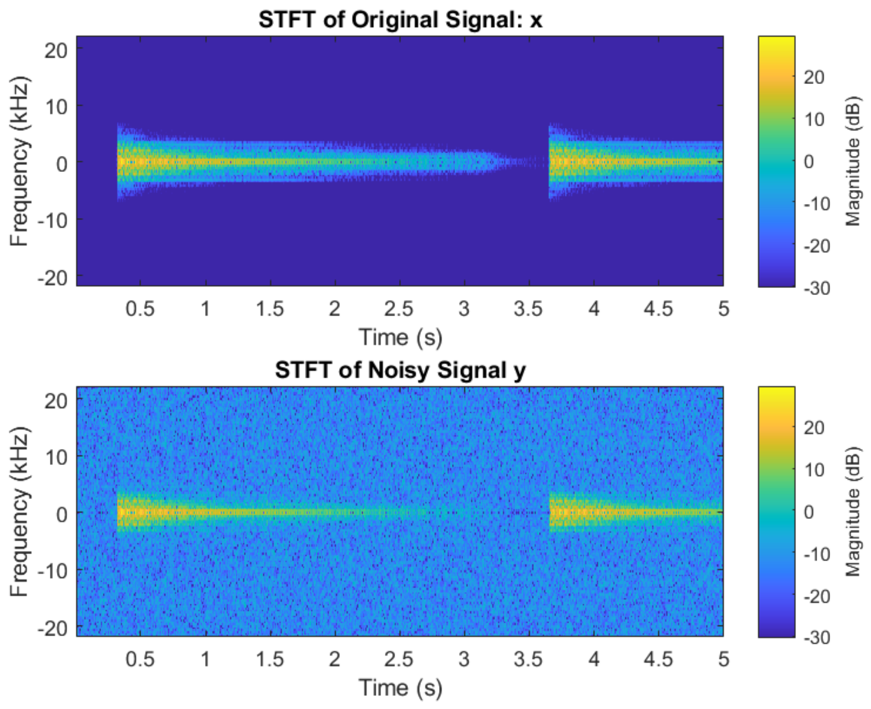
3.2.1 Short Time Fourier Transform (STFT):

The STFT is a more efficient method of TFR than a spectrogram. The basic working principle of STFT is as follows, at its core a set of localized Fourier Transforms (FTs) which are obtained by sliding a fixed window over time. While performing this TFR it is necessary to assume that the signal is stationary, and the duration of the window is fixed. Further, it is also important to note that as the window size is reduced the number of localized FTs increases which increases the efficiency of the STFT. The STFT follows the following equation (1). Where, $x(\tau)$ is the input signal and $h(\tau - t)$is the time shifting window. Both these quantities are convolved with each other, to get the information to frequency domain, by multiplying by $e^{-j2\pi f\tau}$.

$$S_x(t, f; h) = \int_{\tau} [x(\tau) * h(\tau - t)] e^{-j2\pi f\tau} d\tau \qquad (1)$$

In the project, the team uses MATLAB to perform a STFT on the generated clean and noisy signal samples. The team used the command tfrstft(), which is taken from the time frequency toolbox provided by the professor. For this command to execute there is a need to define the window type and the window size. The team chose the hamming window because of its narrow main lobe width with an added advantage of lower side lobe attenuation. The size of the window was determined by finding the size of the silence in the original signal. With these design choices the team achieved an optimal TFR; the resulting plots are all shown in Figure(3).

In Figure (3) there are two plots, the first plot is the TFR of the STFT of the clean audio signal that the team generated. And the second plot is the TFR of the STFT of the noisy signal that the team generated by adding additive white gaussian noise. This representation provides us with the frequency information, ranging from $-20KHz\ to\ 20KHz$ and the time information ranging from $0s\ to\ 5s$. The team also observe the noise power ranging from *-30dB to 30dB.*



Figure(3): STFT of the clean signal and the noisy signal.

3.2.2 Adaptive Window Short Time Fourier Transform (ASTFT):

The ASTFT is another TFR, which is an improvement to the well-known STFT. The major drawback of STFT is that its temporal resolution and spectral resolution follows the uncertainty principle, i.e. if the window size is large, STFT provides us with a higher spectral resolution and lower temporal resolution and with a smaller window size the temporal resolution would be high and the spectral resolution would be low [5].

In this project the team implemented the ASTFT in MATLAB by following the block diagram in Figure(4) [5]. According to the block diagram, the power spectral density of the desired signal is taken and normalized. Then, the mean and variance are calculated by using the equations demonstrated in Figure(4). If the calculated variance is greater than a user-defined threshold, proceed with the ASTFT; if not, then proceed with STFT. This threshold is basically the distinction between a narrow band and a wide band signal. Even though the signal that the team uses is in a narrow band, the team still calculates the ASTFT. This is to show that the described approach to implementing ASTFT is successful.

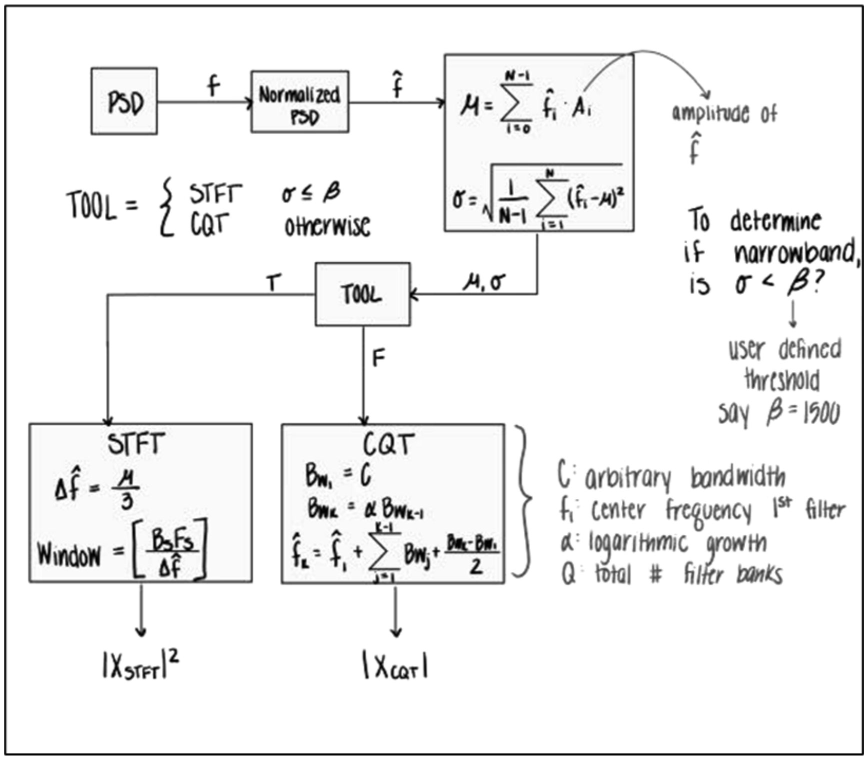
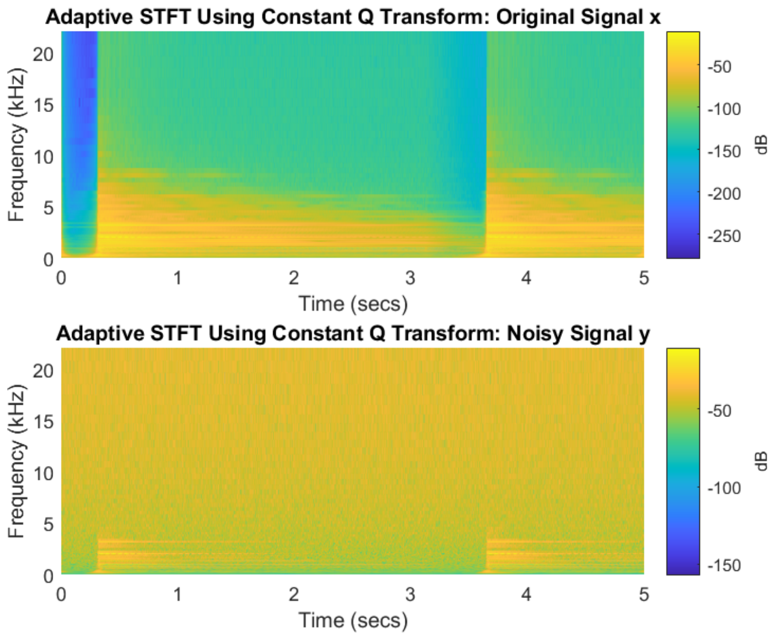

PSD
f
Normalized PSD
$\hat{f}$
$\mu = \sum_{i=0}^{N-1} \hat{f}_i \cdot A_i$
$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\hat{f}_i - \mu)^2}$
amplitude of $\hat{f}$
TOOL = { STFT $\sigma \le \beta$; CQT otherwise
To determine if narrowband, is $\sigma < \beta$?
user defined threshold say $\beta = 1500$
T
TOOL
μ, σ
F
STFT
$\Delta\hat{f} = \frac{\mu}{3}$
Window = $\left[\frac{B_s F_s}{\Delta\hat{f}}\right]$
CQT
$Bw_1 = C$
$Bw_k = \alpha Bw_{k-1}$
$\hat{f}_k = \hat{f}_1 + \sum_{j=1}^{k-1} Bw_j + \frac{Bw_k - Bw_1}{2}$
C: arbitrary bandwidth
f_1: center frequency 1st filter
α: logarithmic growth
Q: total # filter banks
$|X_{STFT}|^2$
$|X_{CQT}|$

Figure (4): ASTFT Block Diagram

The varying window size is achieved by using the constant Q transform (CQT) command in MATLAB, which is cqt(). The CQT in MATLAB has a filter bank with window bins following the geometric progression, which constitutes the varying windows. Once the ASTFT is applied to both the noisy and the original signal, the team gets the time frequency plots that are shown in Figure(5).

In Figure(5) there are two subplots, the first plot is the TFR of the ASTFT of the clean audio signal that the team generated. The second plot is the TFR of the ASTFT of the noisy signal that the team generated by adding additive white gaussian noise. This representation provides us with the frequency information, ranging from $0KHz\ to\ 20KHz$ and the time information ranging from $0s\ to\ 5s$. The noise power can be observed ranging from *–250dB to -50dB*. In comparison to the STFT, there is more observed noise spectral information in the ASTFT TFR.



Figure(5): ASTFT of the clean signal and the noisy signal.

3.3. Enhancement Technique: The Spectral Subtraction Process

Consider the noisy signal created in MATLAB

$$y[n] = x[n] + d[n] \tag{2}$$

where $x[n]$ is the sampled audio signal read into MATLAB with the added *0.25m*s of initial empty speech and $d[n]$ is an additive white gaussian noise based on a particular SNR. As described in section 2, the STFT of the received signal $y[n]$ is calculated. Taking the STFT of the entirety of equation (2), gives the following:

$$Y(w,k) \;=\; X(w,k) \;+\; D(w,k) \tag{3}$$

where w is the frequency and k is the frame number. Given that the STFT calculated does not split the time domain into frames, k can be disregarded completely. At the receiver, $Y(w)$ is known, and the goal is to estimate the input signal $X(w)$ by estimating the additive noise $\widehat{D}(w)$. The power spectrum of $y[n]$ is uncorrelated with the additive white gaussian noise, so equation (3) can be rewritten as

$$|Y(w)|^2 \;=\; |X(w)|^2 \;+ \left|\widehat{D}(w)\right|^2 \tag{4}$$

The estimation of the noise signal $\left|\widehat{D}(w)\right|^2$ is estimated by observing the sections of $Y(w)$ that are empty speech because the STFT of empty speech is equal to zero. Meaning that at times of empty speech,

$$|Y(w)|^2 \;=\; \left|\widehat{D}(w)\right|^2 \tag{5}$$

The first M samples of $Y(w)$ are known to be empty speech given that the initial M samples of silence were added to the input file [4]. In the case where the location of the empty speech samples are not known, a speech detection framework would need to be created in order to find the indices and properly estimate $\widehat{D}(w)$ [2]. The estimation of the STFT of the noise can be calculated by averaging the frames of the M empty speech frames.

$$\left|\widehat{D}(w)\right|^2 = \frac{1}{M}\sum_{j=1}^{M} |Y(w,j)|^2 \tag{6}$$

Implementing this equation on MATLAB resulted in a 1xM noise estimation matrix. This 1xM matrix was repeated as many times as needed to be the same size as $Y(w)$. Using the calculated and sized $\widehat{D}(w)$ matrix, a spectral subtraction filter can be calculated as such

$$H^{2}(w) = (1 - \frac{\left|\widehat{D}(w)\right|^{2}}{|Y(w)|^{2}}) \qquad (7)$$

$$H(w) = max\{0, (1 - \frac{\left|\widehat{D}(w)\right|^{2}}{|Y(w)|^{2}})^{1/2}\} \qquad (8)$$

The last step is for the receiver to make an estimate of the input signal by using the spectral subtraction filter. In a truer sense, the phase of the input signal is needed to make a proper reconstruction. However, the human ear is not privy to the short-term phase of a signal, so using the phase of the received noise signal and then applying the filter should work just the same. Using this information, the following equation was used to obtain an estimate of $\widehat{X}(w)$:

$$\left|\widehat{X}(w)\right|^{2} = H(w) * Y(w) \qquad (9)$$

The receiver uses $\widehat{X}(w)$ as it's received input signal [4]. This process for spectral subtraction was implemented for both the STFT and the ASTFT, wherein equation (3) was obtained by taking the ASTFT of both sides of equation (2). The rest of the spectral subtraction process can then be applied in the same manner to the ASTFT as the regular STFT.

As shown in Figure(6) below, the graph of the STFT of the initially received noisy $Y(w)$ and the estimated $\widehat{X}(w)$ show a clear reduction in the noise after applying the spectral subtraction filter.

Because the spectral subtraction filter is calculated based on the estimation of the noise, it is clear that the success of extracting a non-erroneous $\widehat{X}(w)$ from the initially received $Y(w)$ is fully dependent on the accuracy of that noise estimation. In this case, it is easy to detect the frames of $Y(w)$ that contain empty speech and estimate the noise based on those frames, as the location of those frames is known. This is not always the case. An inaccurate noise estimation will result in a slightly skewed $\widehat{X}(w)$ as not all of the noise will be fully subtracted.

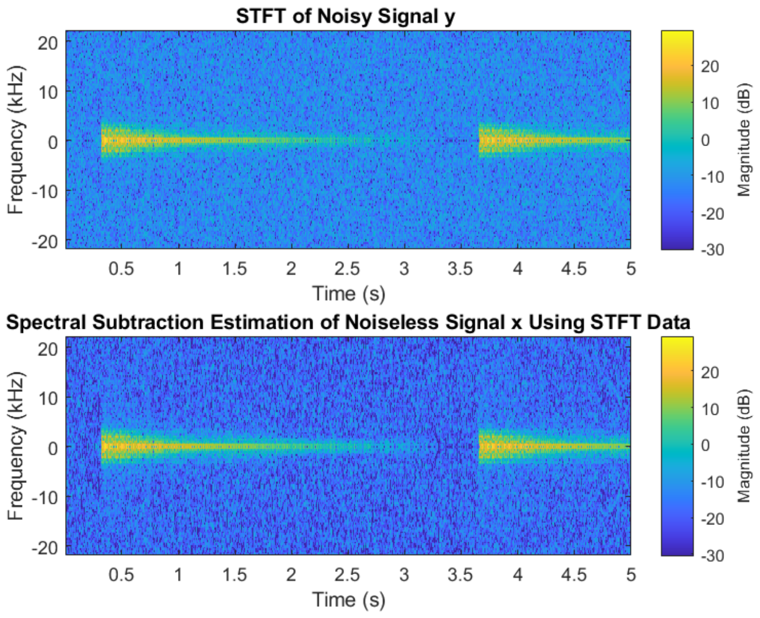


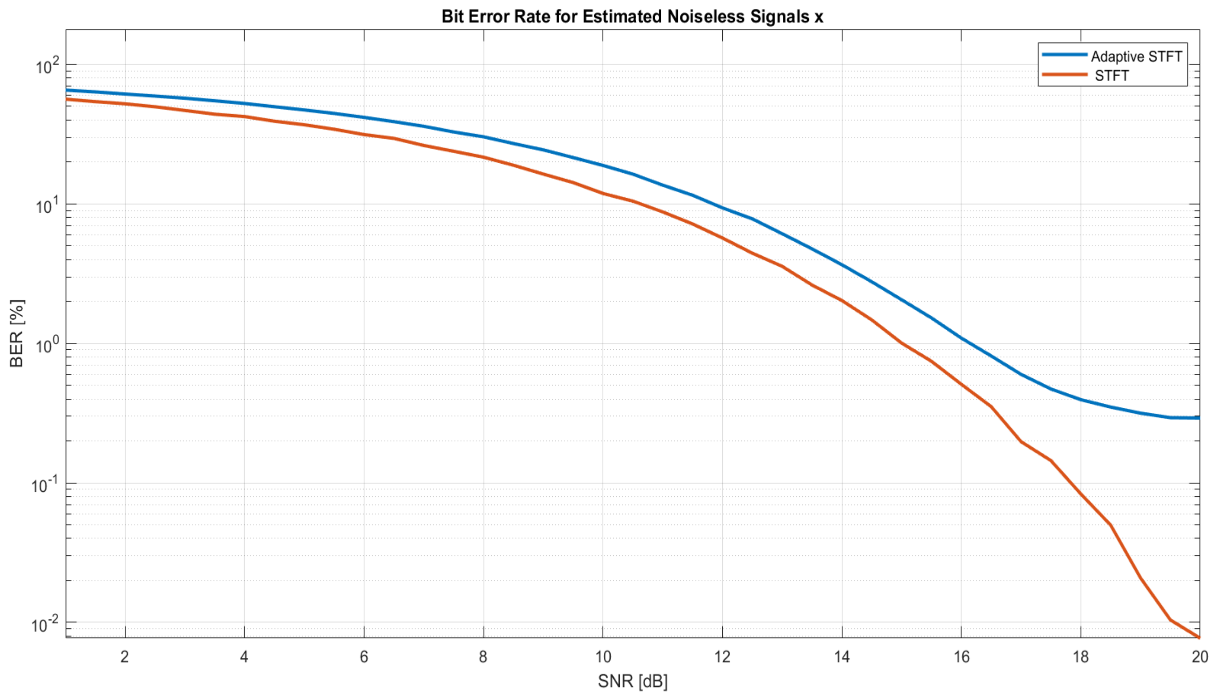
Figure (6): Comparison of the Time Frequency of Noisy Signal y with Estimation of Noiseless Signal x

3.4 Discussion

3.4.1 Efficiency

In order to determine the efficiency of the system generated in this study, the team compared the known noiseless signal data, x, to the estimations provided from the spectral subtraction method across both the STFT TFR and the ASTFT TFR. As discussed in Section 3, the spectral subtraction enhancement provided recovered signal estimations for x. By comparing the estimation to the known value, the team was able to calculate and record the error in this estimation process. In standard communication systems, if known information is sent, it is done so in bits or a modulation scheme, however, in this audio speech data format, the array x is composed of a variety of values that are not consistent across the audio files investigated. Usually, an error case will occur when a value within the x array does not match that indexed value in the estimated array. If implemented in the current system, this would provide all errors and not provide any information on efficiency. Therefore, an error case was defined as the absolute difference between the known and the estimated value being greater than a specific range. Initially, the team looked at the average value of the differences, but this still yielded a very large error. This could be due to the SNR ratio of the system across the single iteration or could be due to the noise estimation found in spectral subtraction being too different from the true added noise. However, these changes were within the acceptable ranges of the model and therefore the error case was also ranged for both the STFT and the ASTFT spectral subtraction estimation results. While this error count does provide information on the effectiveness of the system, a more robust method for evaluating the overall performance was considered. To do so, the team took one iteration of the code and iterated through varying SNR values. The SNR values considered varied from 1dB to 20dB in increments of 0.5dB to gather information on the performance at both low and high SNR ratios. This manually adjusts the SNR magnitude of the white gaussian noise being added to the noiseless signal x. In order to capture this efficiency data, the team calculated the BER as follows:

$$BER = \frac{sum(error\ events)}{L} * 100;\ where\ L = length(x_{estimate}) \tag{10}$$



Figure(7): BER versus SNR plot for Spectral Subtraction Estimations using ASTFT and STFT Data

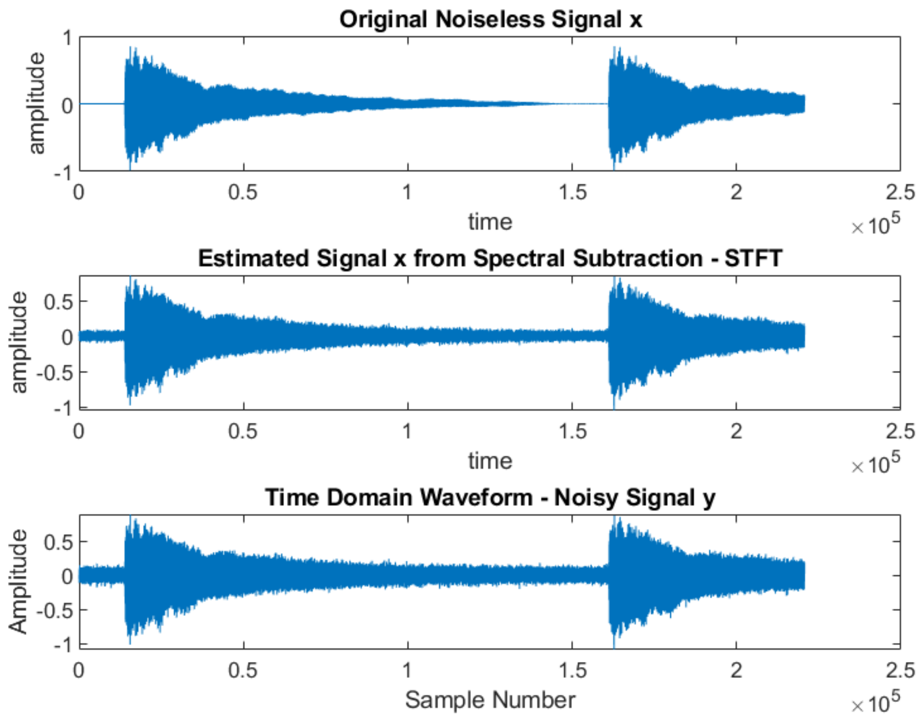
Using equation (10) to calculate the BER, the team was able to generate a BER versus SNR curve shown in Figure(7) that follows the traditional expectation with high errors at low SNR values and smaller errors as SNR increases. This can be seen in Figure(7) for both spectral subtraction estimations of the noiseless signal x. Ultimately, the combination of establishing an error event case for both the STFT and ASTFT spectral subtraction results allowed the team to quantify the effectiveness of the TFRs combined with the spectral subtraction enhancement.

3.4.2 TFR Comparison

Both the STFT and the ASTFT provide an estimation for the frequency domain speech data. However, as expected from the block diagram shown in Figure(4) of Section 3.2.2, one will often perform better than the other, even if just marginally. The goal of utilizing both the STFT approach and the ASTFT approach is to verify results with similar performance in instances where the signal is narrowband. Therefore, as seen in the successful estimation of the frequency domain information across both TFRs, they are able to provide information on the signal. Furthermore, as seen in Figure(7), the variation in the BER between the two methods is not significant. In fact, it is predicted that if the BER was calculated across more than one realization that the BER values would indeed converge for both ASTFT and STFT given that the signal in question is narrowband [5].

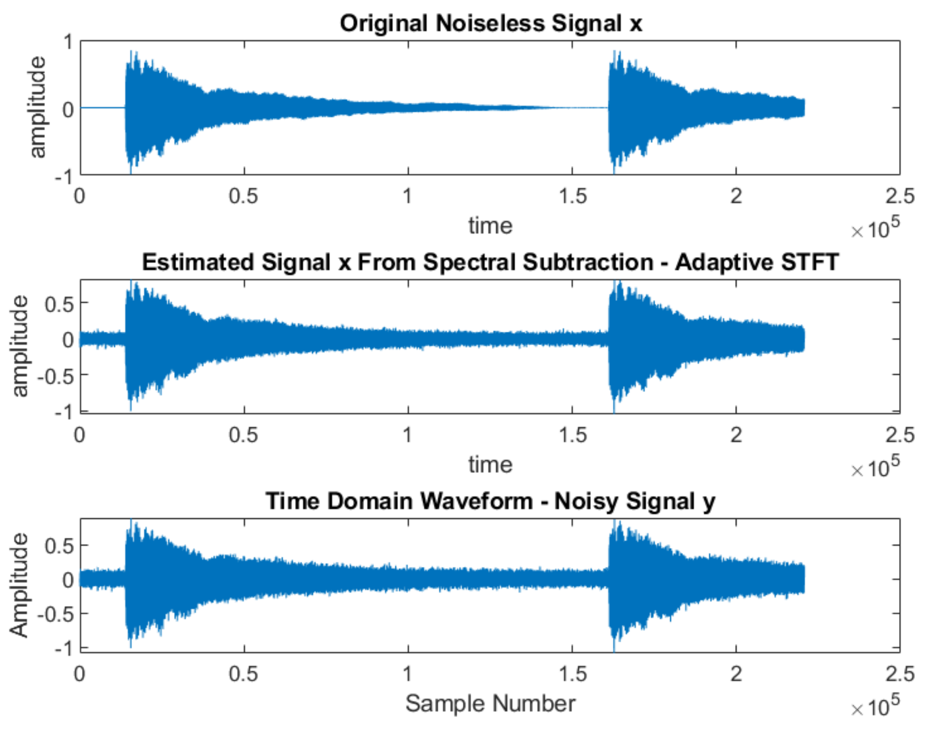
3.4.3 Effectiveness of Spectral Subtraction

As discussed in Section 3.3, the estimation of the noise signal $\left|\widehat{D}(w)\right|^2$ is estimated by observing the sections of Y(w) that are empty speech because the STFT of empty speech is equal to zero. During the implementation of this algorithm, the estimation of the noise proved to be a bit difficult. In the signals used, the areas of speech pause were not of sufficient length to provide a noise estimate that would not result in many errors during efficiency analysis. Therefore, the team chose to introduce *0.25ms* of time into the beginning of the audio file. In doing so, a longer speech pause was simulated which allowed for a more accurate estimation of the noise signal. The spectral subtraction method was then applied to both TFR methods and the discussion of results is as follows.



Figure(8): Comparison of the Noiseless Signal x to the Estimated Signal x from Spectral Subtraction using STFT Data and the Noisy Signal y

The results in the frequency domain are discussed in Section 3 and now the discussion will focus on the time domain results shown in Figure(8) for the spectral subtraction estimation of the noiseless signal x using the STFT data and in Figure(9) for the spectral subtraction estimation of the noiseless signal x using the ASTFT data. The top plot is the original noiseless signal x data plotted as time cross magnitude while the lower plot is the noisy signal y. As seen in the center plots of both Figure(8) above and Figure(9) below, the estimated signal does not entirely match that of the noiseless signal but it does show noise reduction when compared to the noisy signal. This demonstrates that the methodology of spectral subtraction implemented as an enhancement method was effective in generating an estimate for the noise power and then removing to great an expected signal data vector for x.



Figure(9): Comparison of the Noiseless Signal x to the Estimated Signal x from Spectral Subtraction using ASTFT Data and the Noisy Signal y

As discussed in Section 3.4.1, the results of this data array were then compared to the true data across SNR values to quantify the error. Future studies could focus on the improvement of this estimation and some of the suggestions generated during the execution of the study include adjusting the length of speech pauses, increasing the overall length of the signal data, and altering the type of noise added. The team explored adjusting the type of noise to solely Gaussian in this study, however the spectral estimations provided from the STFT and the ASTFT results worsened so the team chose to move forward with the agwn(). Additionally, the team chose not to increase the overall length of the signal data because it would increase the overall computational complexity of the program which was already quite high. In using advanced algorithms for both the STFT and ASTFT implementations and iterating across SNR values, the computation and execution time for the code were quite high.

4. Conclusions

The team successfully collected speech information in the form of an array of data, and implemented two successful iterations of enhancement using TFR data. The results of which fell within an acceptable degree of accuracy and efficiency for the purposes of this study. The two previously discussed TFRs: STFT and ASTFT, were successfully implemented across 40 SNR values for one iteration. The results of these simulations yielded time-frequency plots that the team was able to use to find results as discussed in Section 3.4. Furthermore, spectral subtraction was successfully implemented utilizing the spectral information collected from both TFRs. The output estimations of the noiseless signal that were estimated were compared to the known noiseless speech data, x. The efficiency plot accurately recorded the BER information for both the TFRs. However, the process to complete the error analysis across the two TFRs and two implementations of spectral subtraction was highly computationally inefficient both in the processing power needed and the time duration of the implementation. Thus, only one realization for the SNR loop was considered. Future studies may benefit from improving the efficiency of the algorithms presented in this project and then implementing across additional realizations to determine if the BER for the two TFR methods would converge.

The audio samples generated were determined to be narrow band. Taking into account the fact that the computational complexity of ASTFT is higher than STFT and the observation that for a narrowband signal, both the TFRs will yield similar results, the team recommends the use of STFT instead of ASTFT as the spectral information feeder to the spectral subtraction technique for speech enhancement. However, if the signal were to be in wide band, the team expects ASTFT to perform with high efficiency and within an accepted accuracy range. Additionally, both methods would be expected to have high spectral and temporal resolution. Further, this project can be improved by coming up with an adaptive spectral subtraction technique for a non stationary audio signal.

While the previous studies have discussed STFT, ASTFT and noise reduction, presenting information for the general approach, they fell short when implementing these concepts in a larger scope. The value that this study presents is the comparison of the results obtained when implementing spectral subtraction with multiple resulting data outputs from TFRs and the corresponding accuracy quantitatively.

5. Acknowledgements

[1] Antonia Papandreou-Suppappola.

6. References

[1] B. Boashah, “Chapter 6 - advanced implementation and realization of TFDs,” Time-Frequency Signal Analysis and Processing (Second Edition), pp. 331-385, 2016

[2] M. Vyas, “A gaussian mixture model based speech recognition system using MATLAB,” Signal Image Processing: An International Journal (SIPIJ), vol. 4, no. 4, 2013

[3] N. Danthi andA.R. Aswatha, “Single channel speech enhancement technique - An implementation on Matlab,” International Conference on Innovative Mechanisms for Industry Applications, pp. 369-374, 2017

[4] N. Updahyay and A. Karmakar, “Speech enhancement using spectral subtraction-type algorithms: a comparison and simulation study,” Procedia Computer Science, vol. 54, pp. 574-584, 2015

[5] S. Nisar and O.U. Khan, Tariq M. “An efficient adaptive window size selection method for improving spectrogram visualization,” Journal of Computational Intelligence and Neuroscience, 2016

Appendix A: MATLAB Code

```
505 Project
Read in Audio File
clear all;
[x,Fs] = audioread('Repeated C.aif');
t = 0:1/Fs:1-1/Fs;

% reshape to ensure single array
[m,n] = size(x);
if n~=1
    x = reshape(x,[m*n,1]);
end

%Shortened x for debugging purposes
LX = length(x);
a = ceil(0.2*LX);
x = x(1:a);

Add 0.25 ms Initial Silence
M = (0.25*Fs) %0.25 ms # samples
add_IS = zeros(M,1);

x = vertcat(add_IS,x);

Plot the Noiseless and Noisy Audio Files
Without Noise
figure();

% time domain waveform
subplot(3,1,1, 'align')
plot(x)
title('Time Domain Waveform of Noiseless Signal x')
xlabel('Sample Number'); ylabel('Amplitude')

% FFT of audio file
xf = fft(x);
subplot(3,1,2, 'align')
plot(abs(xf))
title('FFT of Input Noiseless Signal x')
xlabel('Sample Number'); ylabel('Amplitude')
% Spectrogram
subplot(3,1,3, 'align')
pspectrum(x,Fs,'spectrogram','TimeResolution',0.1)
title('Spectogram of Noiseless Signal x')
```

```
% save value into an array
x_spec = pspectrum(x,Fs,'spectrogram','TimeResolution',0.1);
```

Generate Noise Across SNR Values

```
SNR = 0:0.5:20; % signal strength in dBW
for ss = 1: length(SNR)
   sigw = 1./SNR;
   N = length(x);

   y = awgn(x,SNR(ss),'measured');
   Noise = y - x;
```

With Noise

```
figure();
% time domain waveform
subplot(3,1,1, 'align')
plot(y)
title('Time Domain Waveform - Noisy Signal y');
xlabel('Sample Number'); ylabel('Amplitude');

% FFT of noisy signal
xf_n = fft(y);

subplot(3,1,2, 'align')
plot(abs(xf_n))
title('FFT of Noisy Signal y')
xlabel('Sample Number'); ylabel('Amplitude')

% spectrogram of noisy signal
subplot(3,1,3, 'align')
pspectrum(y,Fs,'spectrogram','TimeResolution',0.1)
title('Spectogram of Noisy Signal y')
% save value into array
y_spec = pspectrum(y,Fs,'spectrogram','TimeResolution',0.1);
```

STFT

```
% Parameters

wind = hamming(M);
n = 1:length(x);
NN = 128;

Ts = 1/Fs;
Td = Ts*length(x)-1;
tt = linspace(0,Td,N);
```

```
ff = linspace(0,Fs,NN);

[Sx,tx,fx] = tfrstft(x,n,NN,wind);
[Sy,ty,fy] = tfrstft(y,n,NN,wind);
[SNoise,tnoise,ynoise] = tfrstft(Noise,n,NN,wind);

figure();

% without noise
subplot(2,1,1)
stft(x,Fs)
title('STFT of Original Signal: x')

% with noise
subplot(2,1,2)
stft(y,Fs)
title('STFT of Noisy Signal y')
```

Adaptive STFT Implementation

notes on outcome of cqt function

```
   ccfs = x_cqt
   cvf = bandpass center frequencies
   g = Gabor frames
   fshifts = frequency shifts in DFT bins
   fintervals = frequency intervals corresponding the rows of cfs
   The kth element of fshifts is the frequency shift in DFT bins between the
   ((k-1) mod N) and (k mod N) element of fintervals with k = 0,1,2,...,N-1
   where N is the number of frequency shifts. Because MATLAB® indexes from 1,
   fshifts(1) contains the frequency shift between fintervals{end} and
   fintervals{1}, fshifts(2) contains the frequency shift between fintervals{1}
   and fintervals{2}, and so on.
   bw = returns the bandwidth, bw, in DFT bins of the frequency intervals, fintervals
```

without noise

```
total_BW = Fs;

L_x = length(x);

% Take fft and find norm and RMS values
x_FFT = fft(x)/L_x;
df_x = Fs/L_x;
f_norm_x = abs(x_FFT).^2/df_x;
f_RMS_x = sqrt(df_x * sum(abs(f_norm_x)));

% store information on mean and std_dev for use later
```

```
mean_x = sum(f_norm_x*f_RMS_x);
temp_x = (f_norm_x-mean_x).^2;
std_dev_x = sqrt((1/(L_x-1))*sum(temp_x));

% execute constant Q transform function
[cfs_x,f_x,g_x,fshifts_x,fintervals_x,bw_x] = cqt(x);
```
with noise
```
total_BW = Fs;

L_y = length(y);

% Take fft and find norm and RMS values
y_FFT = fft(y)/L_y;
df_y = Fs/L_y;
f_norm_y = abs(y_FFT).^2/df_y;
f_RMS_y = sqrt(df_y * sum(abs(f_norm_y)));

% store information on mean and std_dev for use later
mean_y = sum(f_norm_x*f_RMS_x);
temp_y = (f_norm_x-mean_y).^2;
std_dev_y = sqrt((1/(L_y-1))*sum(temp_y));

% execute constant Q transform function
[cfs_y,f_y,g_y,fshifts_y,fintervals_y,bw_y] = cqt(y);
```
Plot results
```
figure();

subplot(2,1,1)
cqt(x,'SamplingFrequency',Fs)
title('Adaptive STFT Using Constant Q Transform: Original Signal x')

subplot(2,1,2)
cqt(y,'SamplingFrequency',Fs)
title('Adaptive STFT Using Constant Q Transform: Noisy Signal y')
```
Spectral Subtraction on TFR 1: STFT

Spectral Subtraction

Empty Speech is Initial 0.25 ms silence (11025 samples in x)
```
% STFT: Y(w) = S(w) + D(w)
Y_abs = (abs(Sy)).^2;
S_abs = (abs(Sx)).^2;

% First M columns of Sy are silence
```

```
YSP = [];
for col_index = 1:M
    YSP_temp = Sy(:,col_index);
    YSP = [YSP YSP_temp];
end

Dhat_abs = mean(abs(YSP)).^2;
[a b] = size(Y_abs);
repetitions = ceil(b/M);
Dhat_abs = repmat(Dhat_abs,a,repetitions);
[g h] = size(Dhat_abs);
rid_columns = h-b
rid_elements = rid_columns*g
Dhat_abs = Dhat_abs(1:((g*h)-rid_elements));
Dhat_abs = reshape(Dhat_abs,a,b);
% Real D
D_abs = Y_abs - S_abs;
D_abs2 = (abs(SNoise)).^2;

H2w = 1-(Dhat_abs)./Y_abs;
Shat_abs = H2w.*Y_abs;
b_temp = max(0,H2w);
Hw = sqrt(b_temp);

Sxhat = Hw.*Sy;
```

Recover Signal and Plot in Time Domain

```
xhat_STFT = tfristft(Sxhat,tx,wind);

figure();
subplot(3,1,1)
plot(x)
title('Original Noiseless Signal x')
xlabel('time'); ylabel('amplitude');
subplot(3,1,2)
plot(xhat_STFT)
title('Estimated Signal x from Spectral Subtraction - STFT')
xlabel('time'); ylabel('amplitude');

subplot(3,1,3)
plot(y)
title('Time Domain Waveform - Noisy Signal y');
xlabel('Sample Number'); ylabel('Amplitude');
```

Plot in Frequency Domain

```
% without noise
figure();
subplot(2,1,1)
stft(y,Fs)
title('STFT of Noisy Signal y ')
% with noise
subplot(2,1,2)
stft(xhat_STFT,Fs)
% title('Estimated STFT of Original Signal')
title('Spectral Subtraction Estimation of Noiseless Signal x Using STFT Data')
```

Spectral Subtraction on TFR 2: Adaptive STFT
Empty Speech is Initial 0.25 ms silence (11025 samples in x)

```
% STFT: Y(w) = S(w) + D(w)
Y_abs_ASTFT = (abs(cfs_y)).^2;
S_abs_ASTFT = (abs(cfs_x)).^2;

% First M columns of Sy are silence

YSP2 = [];
for col_index2 = 1:M
   YSP_temp2 = Sy(:,col_index2);
   YSP2 = [YSP2 YSP_temp2];
end

Dhat2_abs = mean(abs(YSP2)).^2;
[a2, b2] = size(Y_abs_ASTFT);
repetitions2 = ceil(b2/M);
Dhat2_abs = repmat(Dhat2_abs,a2,repetitions2);
[g2, h2] = size(Dhat2_abs);
rid_columns2 = h2-b2
rid_elements2 = rid_columns2*g2
Dhat2_abs = Dhat2_abs(1:((g2*h2)-rid_elements2));
Dhat2_abs = reshape(Dhat2_abs,a2,b2);
% Real D
D_abs_ASTFT = Y_abs_ASTFT - S_abs_ASTFT;
D_abs2_ASTFT = (abs(SNoise)).^2;

H2w_ASTFT = 1-(Dhat2_abs)./Y_abs_ASTFT;
Shat_abs_ASTFT = H2w_ASTFT.*Y_abs_ASTFT;
b_temp2 = max(0,H2w_ASTFT);
Hw2 = sqrt(b_temp2);

Sxhat2 = Hw2.*cfs_y;
```

Recover Signal and Plot in Time Domain

```
wind2=220687;
xhat_ASTFT = icqt(Sxhat2,g_x,fshifts_x);

figure();
subplot(3,1,1)
plot(x)
title('Original Noiseless Signal x')
xlabel('time'); ylabel('amplitude');

subplot(3,1,2)
plot(xhat_ASTFT)
title('Estimated Signal x From Spectral Subtraction - Adaptive STFT')
xlabel('time'); ylabel('amplitude');

subplot(3,1,3)
plot(y)
title('Time Domain Waveform - Noisy Signal y');
xlabel('Sample Number'); ylabel('Amplitude');
```

Plot in Frequency Domain

```
figure();
% without noise
subplot(2,1,1)
stft(y,Fs)
title('STFT of Noisy Signal y')

% with noise
subplot(2,1,2)
stft(xhat_ASTFT,Fs)
%title('Estimated STFT of Original Signal Using Spectral Subtraction on Adaptive STFT')
title('Spectral Subtraction Estimation of Noiseless Signal x Using Adaptive STFT Data')
```

Efficiency Analysis - STFT

```
    errors = 0;
    L = length(xhat_STFT);

    % x_actual = x(1:L); % disregard size decrease from STFT
    errors=0;
    range = 0.05;
    diff_STFT = abs(x - xhat_STFT);
    %range = (max(diff_STFT)-min(diff_STFT))/length(diff_STFT);
    for i = 1 : L
        if diff_STFT(i) > range
            errors = errors+1;
        end
    end

    percent(ss) = (errors/L)*100;

end


semilogy(SNR,percent, 'linewidth',2);
grid on
xlabel('SNR (dB)');
ylabel('BER');

title('BER Vs SNR graph For STFT')
Efficiency Analysis - Adaptive STFT
    errors = 0;
    L = length(xhat_ASTFT);

    % x_actual = x(1:L); % disregard size decrease from STFT
    errors=0;
    range = 0.05;
    diff_STFT = abs(x - xhat_ASTFT);
    %range = (max(diff_STFT)-min(diff_STFT))/length(diff_STFT);
    for i = 1 : L
        if diff_STFT(i) > range
            errors = errors+1;
        end
    end

    percentA(ss) = (errors/L)*100;

end

semilogy(SNR,percentA, 'linewidth',2);
grid on
hold on
semilogy(SNR,percent(1:39), 'linewidth',2);
xlabel('SNR (dB)');
ylabel('BER');
xlim([1 20])
ylim([0 179])
title('BER Vs SNR graph For STFT')
```