EE430 Project

Part-1

Kudret Esmer 2304608

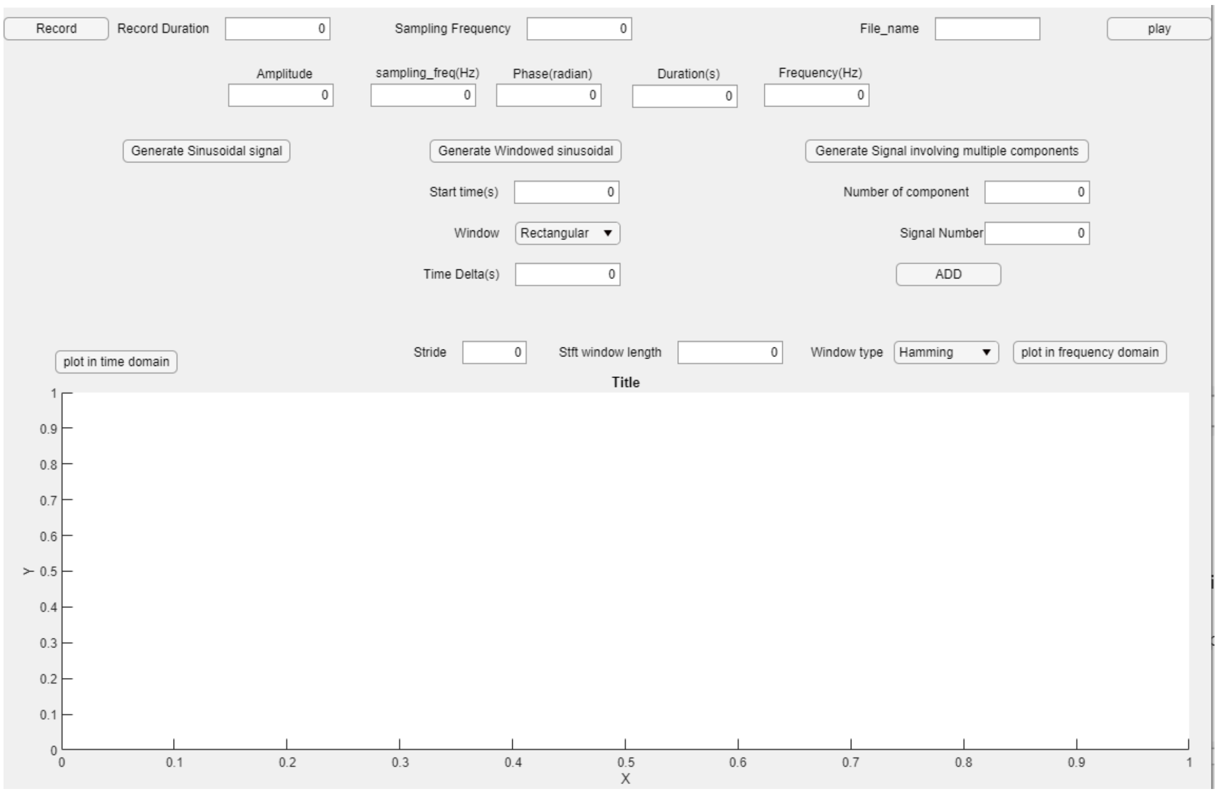
Hakan Emre Gedik 2304657

Contents

Introduction

Since they can present dense amount of knowledge about a finite time domain signal with a single image to the trained eye, spectrogram plots are widely used in signal processing. Applications range from audio processing to vibration analysis for spectrograms provide valuable insights especially in time domain signals which information is hidden in certain parts of the signal rather than all along it. These plots are based on STFT, which performs frequency domain analysis on windowed time domain data. Images outputted by the spectrogram analysis provide the time in the x axes, whereas in the y axis each pixel color codes the strength of the frequency component. The signal is clipped around a center point with a predetermined window length and DFT operation is applied on the clipped signal. In this way, the frequency behavior of each time interval is extracted from the signal. In this project, we implemented the spectrogram function using STFT on our own. We developed a MATLAB GUI to obtain plots for various sound signals that are either recorded, taken from an audio file, or generated artificially. The GUI designed allows hyper-parameters of spectrograms such as stride, window length, and window type. For the artificially generated audio data, the GUI can provide a sinusoidal, a windowed sinusoidal, or a superposition of sinusoidals. For each component, parameters such as amplitude, frequency, start time, and window type, are expected from the user. This project helped our eyes to be "trained" in reading spectrograms, since in the developing process, we observed the spectrogram plots of various signals with different choices of hyper-parameters.

User Manual

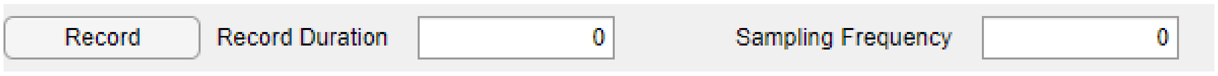

Record
play
Generate Sinusoidal signal
Generate Windowed sinusoidal
Generate Signal involving multiple components
ADD
plot in time domain
plot in frequency domain
Title
Y
X

Record Duration 0
Sampling Frequency 0
File_name
Amplitude 0
sampling_freq(Hz) 0
Phase(radian) 0
Duration(s) 0
Frequency(Hz) 0
Start time(s) 0
Window Rectangular
Time Delta(s) 0
Number of component 0
Signal Number 0
Stride 0
Stft window length 0
Window type Hamming

Figure 1 Designed GUI

Data acquisition

Sound data from the microphone

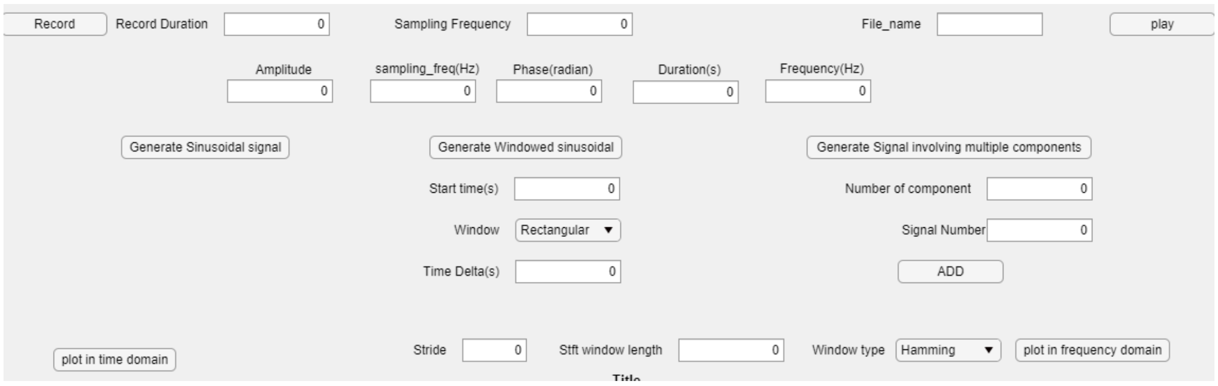


Record Duration 0

Sampling Frequency 0

Figure 2 Recording Section

In order to record the voice from the microphone the duration of the record (s) and the sampling frequency (Hz) should be specified. These specifications should be written given fields in the figure 2. Then, record button can be used to record the sound data. After pushing the button, the program will start recording the voice at given sampling frequency. Afterwards, the recorded voice can be played by pushing play button as indicated in the figure 3. Furthermore, recording can be plotted both in the time domain and frequency domain. Detail about plots will be given later.


Record
Generate Sinusoidal signal
Generate Windowed sinusoidal
Generate Signal involving multiple components
ADD
play
plot in time domain
plot in frequency domain
Title

Record Duration 0

Sampling Frequency 0

File_name

Amplitude 0

sampling_freq(Hz) 0

Phase(radian) 0

Duration(s) 0

Frequency(Hz) 0

Start time(s) 0

Window Rectangular

Time Delta(s) 0

Number of component 0

Signal Number 0

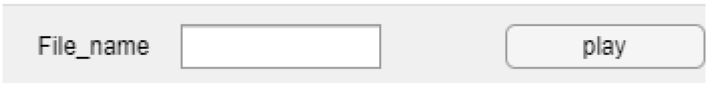
Stride 0

Stft window length 0

Window type Hamming

Figure 3 The Designed GUI

Sound data from a file

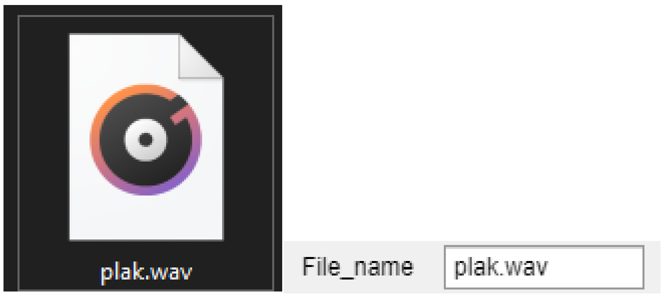


File_name

Figure 4 File name and play button

In order to process data from the existing sound files, write the full name of the existing data. Moreover, existing data should be located at the same location with the program itself. The location to provide name showed in the figure 4.

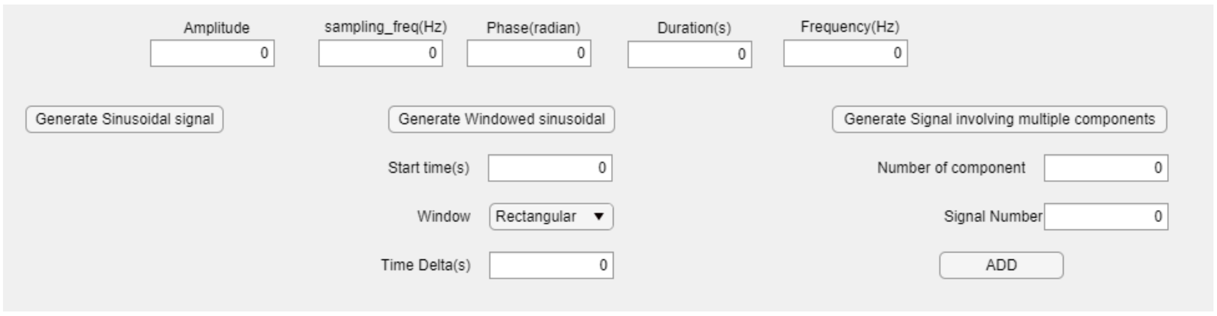
For example, to load given data in figure 5 which is located in the project file, plak.wav should be written to the File_name section which is also indicated in the figure 5 b.



File_name: plak.wav

Figure 5 .wav File Example

Data generation



Amplitude	sampling_freq(Hz)	Phase(radian)	Duration(s)	Frequency(Hz)
0	0	0	0	0

Generate Sinusoidal signal
Generate Windowed sinusoidal
Generate Signal involving multiple components
ADD

Start time(s): 0

Window: Rectangular

Time Delta(s): 0

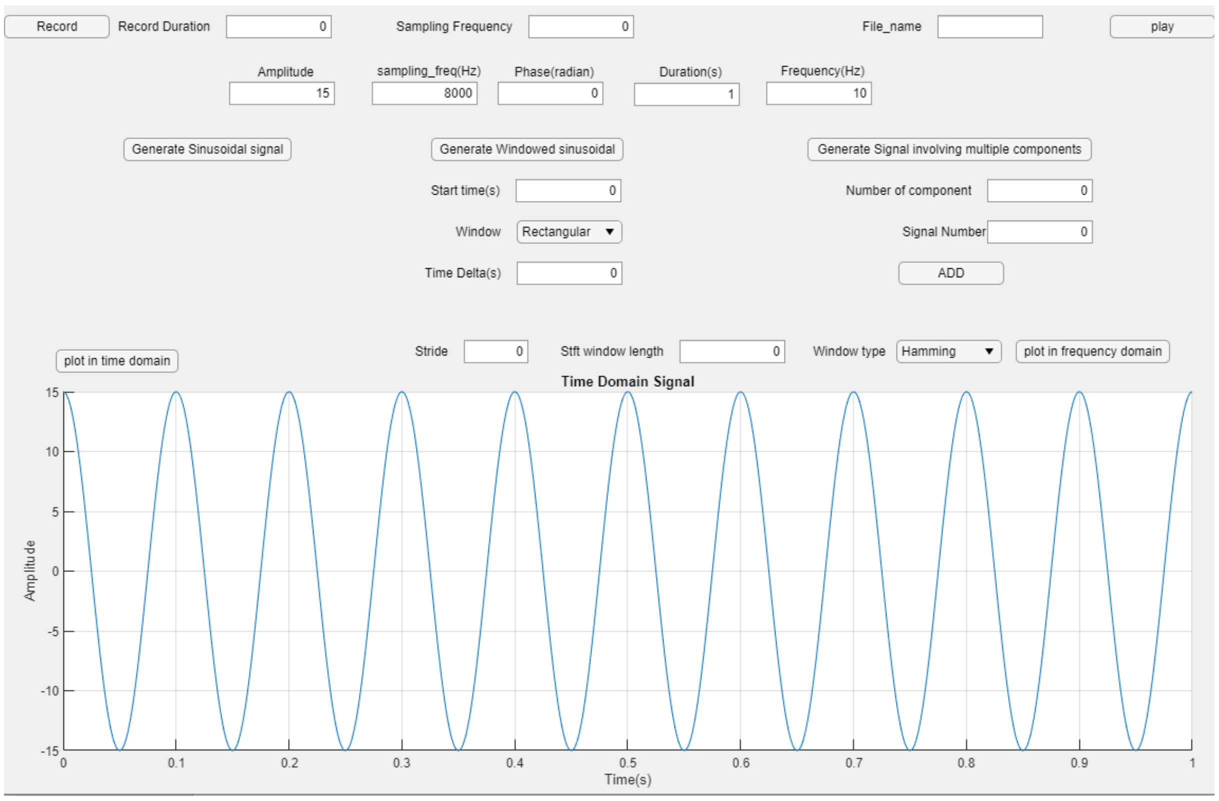
Number of component: 0

Signal Number: 0

Figure 6 Data Acquisition Part of the GUI

Sinusoidal Signal

To generate a sinusoidal signal amplitude, sampling frequency, phase, duration, and frequency of the signal should be written related fields in the figure 6. After specifying these parameters generate sinusoidal signal button can be pushed to generate the signal.



Record Duration 0
Sampling Frequency 0
File_name
Amplitude 15
sampling_freq(Hz) 8000
Phase(radian) 0
Duration(s) 1
Frequency(Hz) 10
Start time(s) 0
Window Rectangular
Time Delta(s) 0
Number of component 0
Signal Number 0
Stride 0
Stft window length 0
Window type Hamming

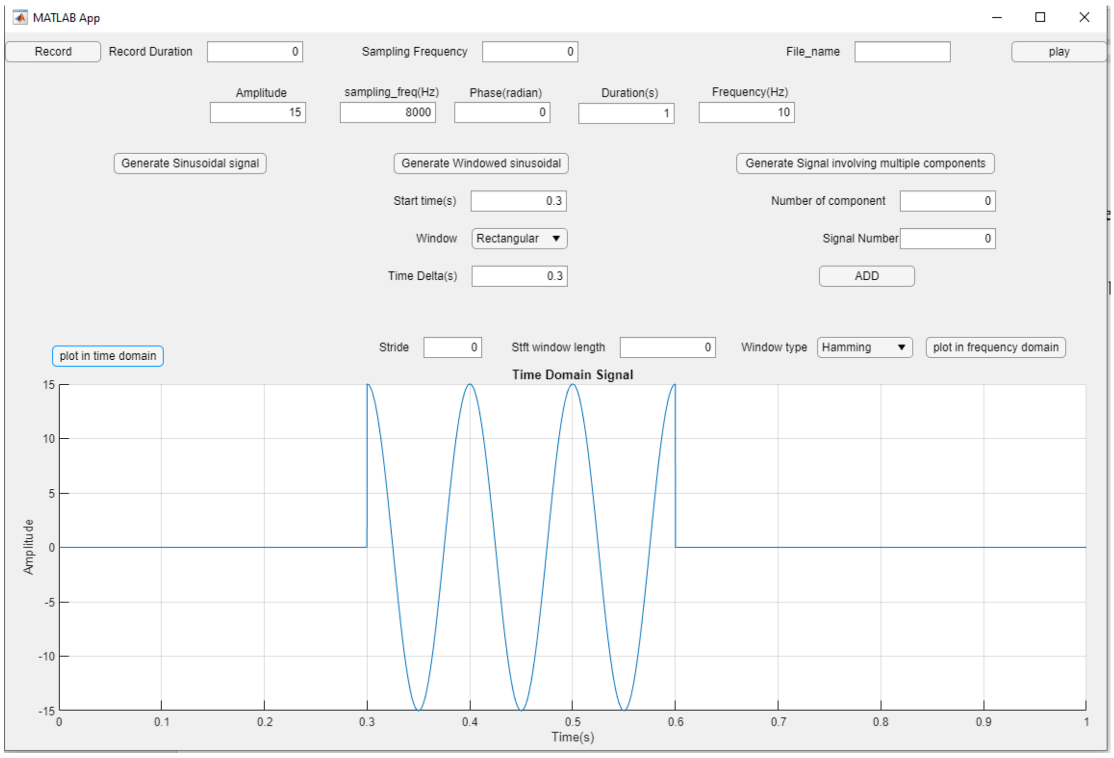
Figure 7 Example Sinusoidal Signal

Windowed Sinusoidal

Windowed sinusoidal can be generated using same fields for amplitude, sampling frequency, phase, duration, and frequency of the signal. In addition to these parameters starting time of the window, window shape and time delta should be provided. After pushing Generate Windowed sinusoidal program will generate a sinusoidal and a window that has the length equal to Time Delta then will take their dot product after shifting them by specified Start time.

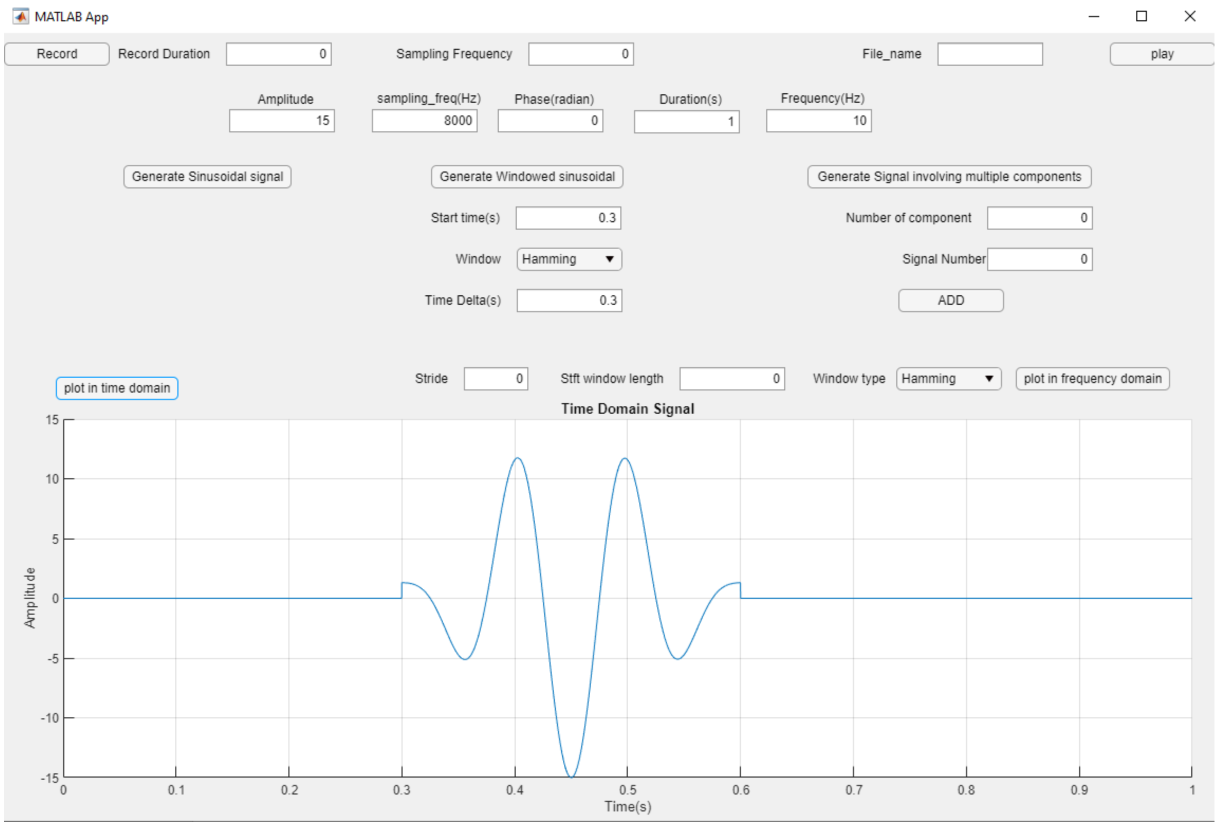
One can see 2 examples below.

In the figure one rectangular window is used to generate windowed sinusoidal. However, in the figure 9 Hamming window is used.



Record Duration 0
Sampling Frequency 0
File_name
Amplitude 15
sampling_freq(Hz) 8000
Phase(radian) 0
Duration(s) 1
Frequency(Hz) 10
Start time(s) 0.3
Window Rectangular
Time Delta(s) 0.3
Number of component 0
Signal Number 0
Stride 0
Stft window length 0
Window type Hamming

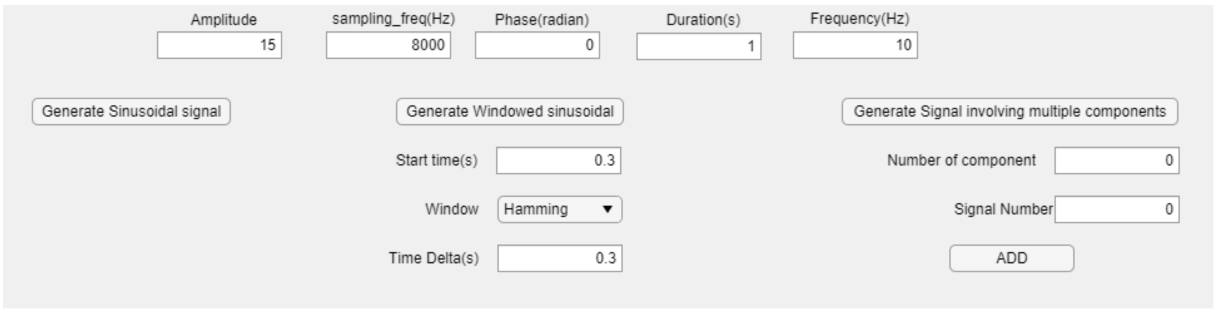
Figure 8 Rectangular Windowed Sinusoidal



Record Duration 0
Sampling Frequency 0
File_name
Amplitude 15
sampling_freq(Hz) 8000
Phase(radian) 0
Duration(s) 1
Frequency(Hz) 10
Start time(s) 0.3
Window Hamming
Time Delta(s) 0.3
Number of component 0
Signal Number 0
Stride 0
Stft window length 0
Window type Hamming

Figure 9 Hamming Windowed Sinusoidal

Signal involving multiple components



Amplitude 15
sampling_freq(Hz) 8000
Phase(radian) 0
Duration(s) 1
Frequency(Hz) 10

Generate Sinusoidal signal
Generate Windowed sinusoidal
Generate Signal involving multiple components

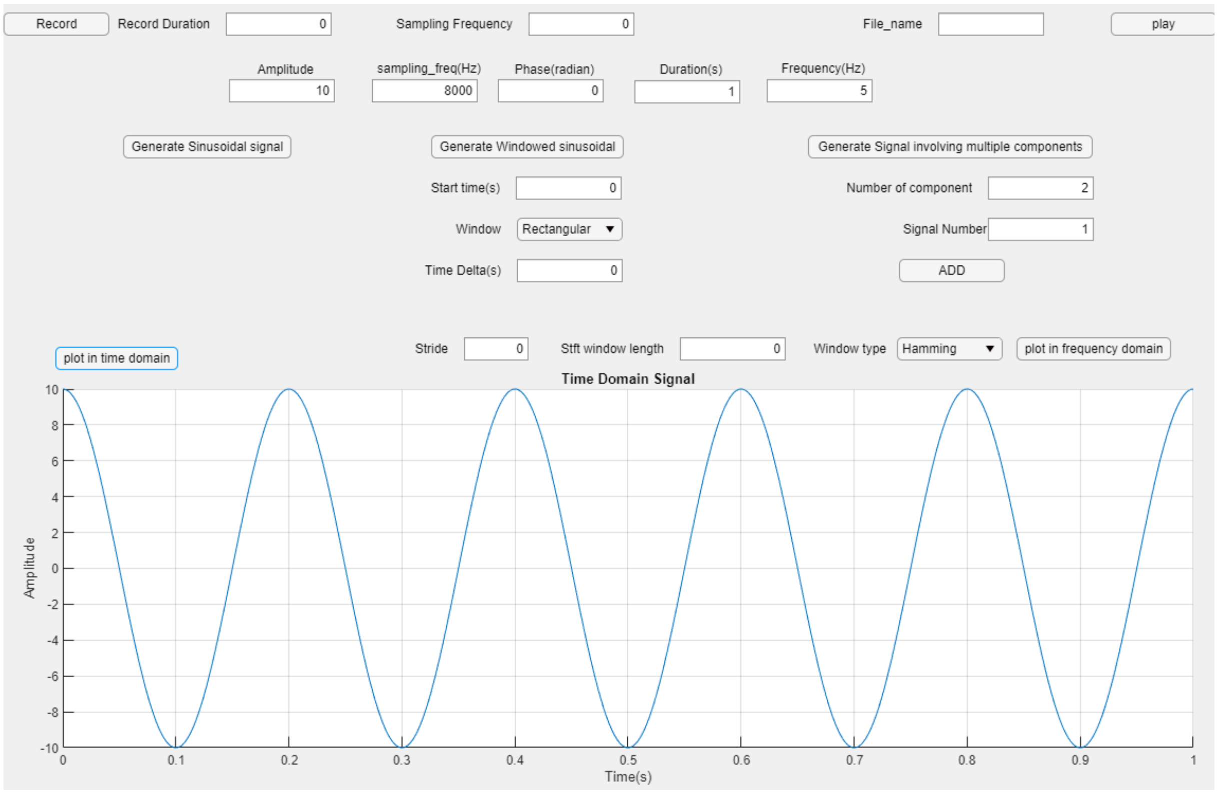
Start time(s) 0.3
Window Hamming
Time Delta(s) 0.3

Number of component 0
Signal Number 0
ADD

Figure 10 Multiple signal generation GUI

In order to generate signals involving multiple components total number of the component should be specified via Number of component section. Afterwards by choosing signal number and specifying amplitude, sampling frequency, phase, duration, and the frequency of the n'th signal can be generated by pushing add button. For example, first component can be generated by writing 1 to Signal Number section and pushing ADD button. Second one can be added using the same procedure except that Signal Number should be 2.

For example, the sinusoidal is specified and added to the multiple component signal then plotted in the figure 11. Afterwards, the second component is indicated, added to our multiple component signal and plotted. Total number of the component can be changed by the user.



Record
Record Duration 0
Sampling Frequency 0
File_name
play

Amplitude 10
sampling_freq(Hz) 8000
Phase(radian) 0
Duration(s) 1
Frequency(Hz) 5

Generate Sinusoidal signal
Generate Windowed sinusoidal
Generate Signal involving multiple components

Start time(s) 0
Window Rectangular
Time Delta(s) 0

Number of component 2
Signal Number 1
ADD

plot in time domain
Stride 0
Stft window length 0
Window type Hamming
plot in frequency domain

Figure 11 Signal After Adding the First Component of the Signal

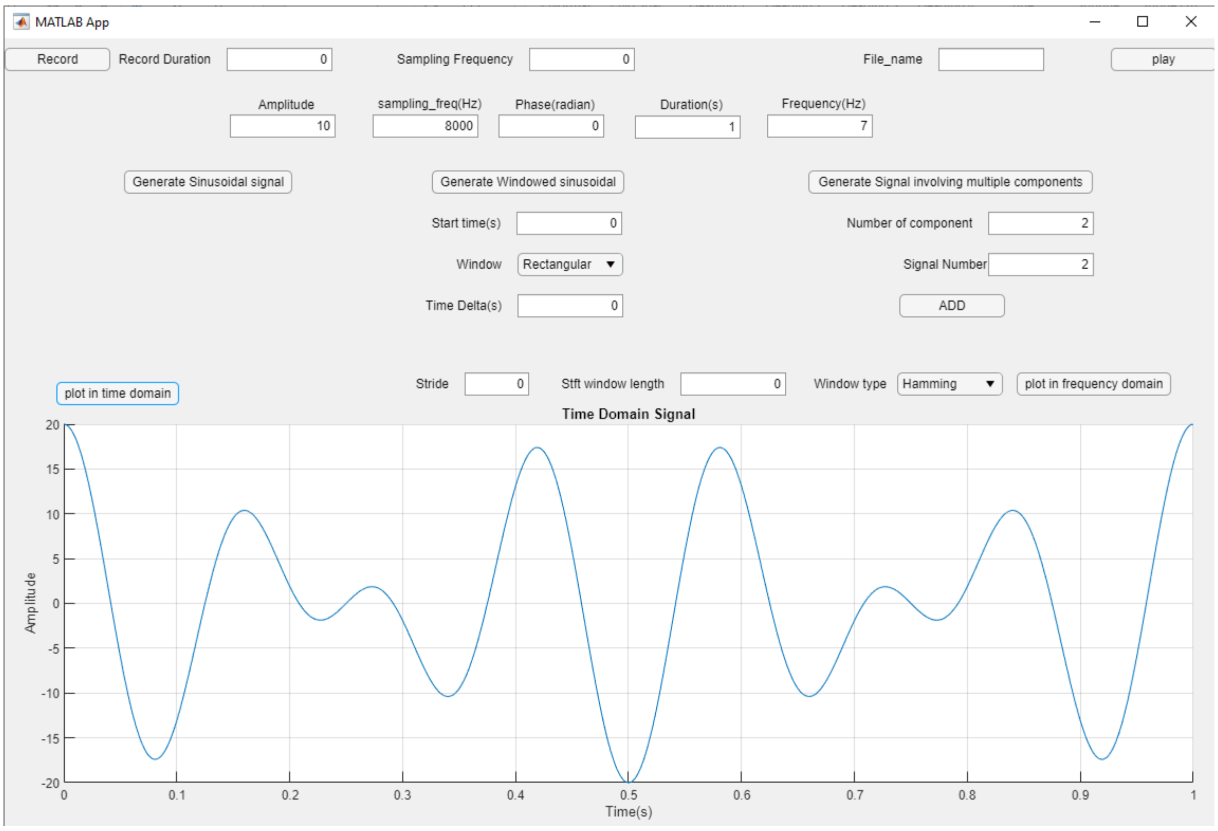


Figure 12 signal after adding second component of the signal

Plotting options

Time domain

After getting sound signal or sinusoidal signal either from the computer or user, these signals can be plotted in the time domain by using the plot in time domain button.

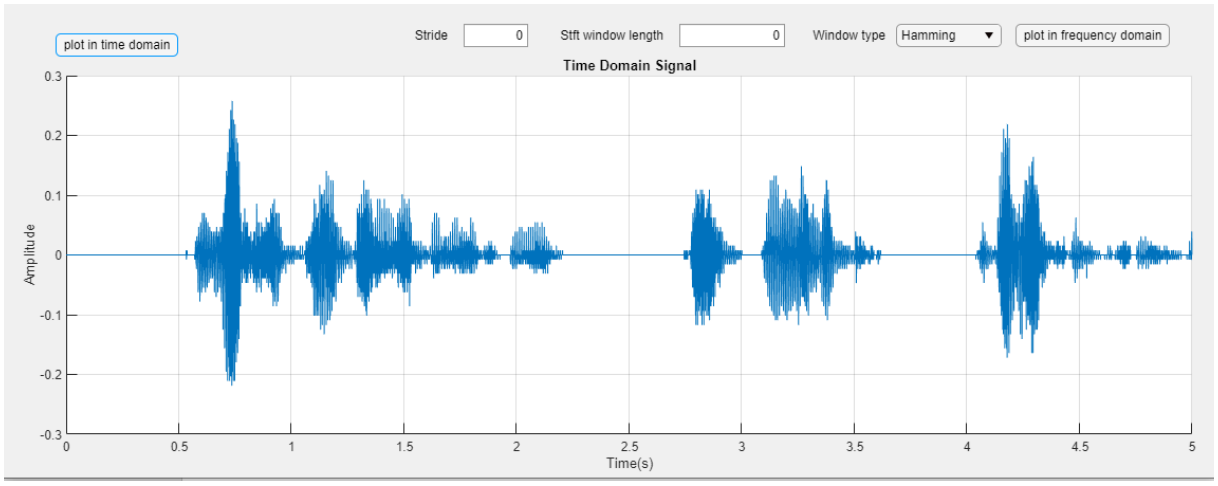


Figure 13 Time domain plot of example sound record

Frequency domain

All signals can also be plotted in the frequency domain using STFT. Stride window length and the window type should be specified by the user. After setting these specifications spectrogram can be plotted by pushing the plot in frequency domain button.

Stride is the number of samples between two consecutive STFT window. STFT window length is also in terms of the number of samples. Window type is indicating the STFT window, it is either Rectangular or Hamming.

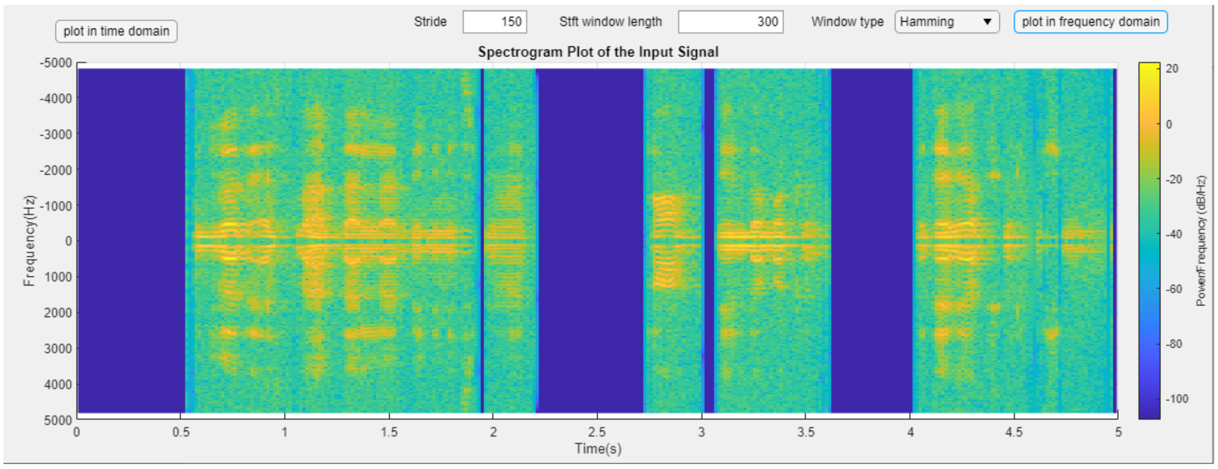


Figure 14 Spectrogram of the signal that is plotted in time domain in the figure 13

Tests and Sample Plots

In this part, various tests are performed, and sample spectrogram plots are taken with real data recorded by us, and artificially generated data. The recorded audio data is the word "examination". Each syllable can seem to be marked on the spectrogram plot. The artificially generated data is a simple sine wave with a predetermined frequency, for which in the plot two bands symmetric around zero is observed.

Part 1-

Part a –

The word "examination" is recorded with help of the GUI. In figure 15, the time domain plot can be observed. The words can be marked on the time domain plot roughly by using the fact that people put stress on the starting of each syllable. Hence, the rises in the strength of the audio signal corresponds to the starts of the syllables.

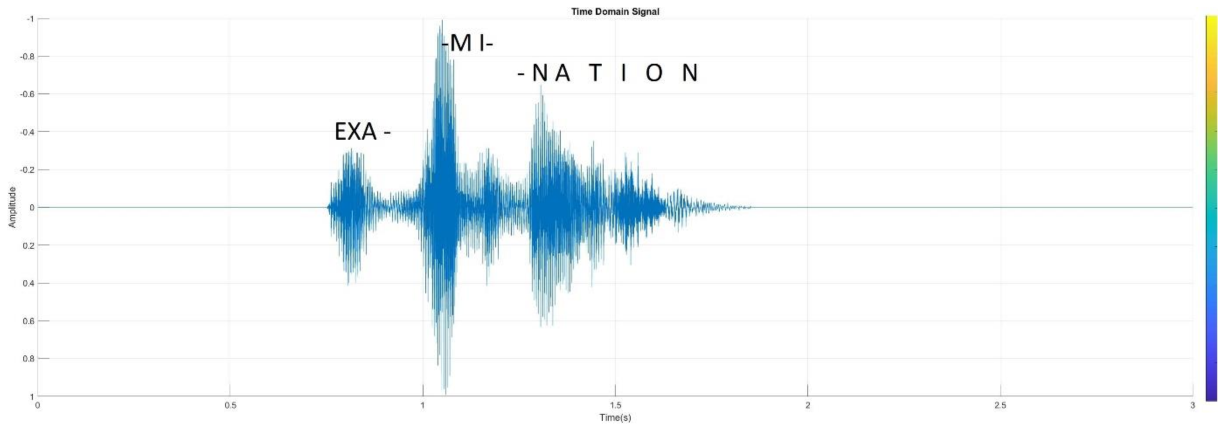


Figure 15 Time Domain Plot of the Recording

Part b –

In this part, spectrogram plots are provided, and the letters in the time axis is roughly indicated. People tend to change the pitch and strength of their voice in the beginning of each syllable. Sudden changes in the amplitude of the voice causes a square wave like frequency domain behaviour. Bands are observed in the beginning of the syllables at the dominant frequency and its harmonics.

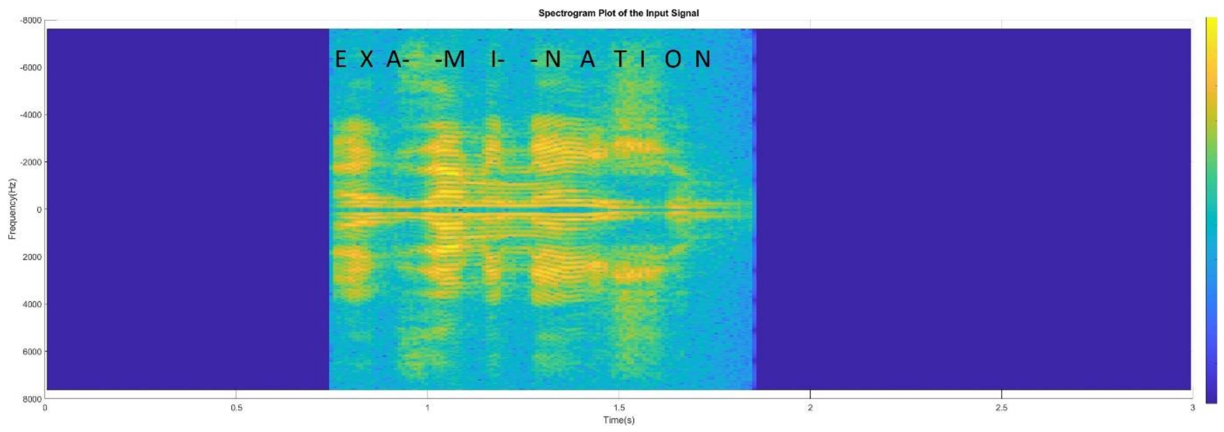


Figure 16 Spectrogram Plot of the Recording, Hamming Window, Window Length:304, Stride: 152

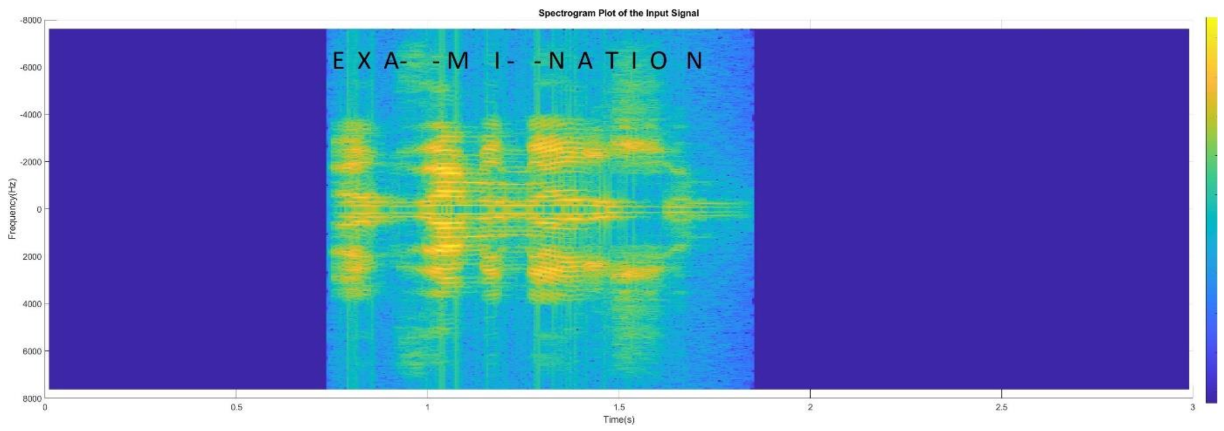


Figure 17 Spectrogram Plot of the Recording, Rectangle Window, Window Length:400, Stride: 90

Observing the plots in figure 16 and 17, hamming window seems to clarify the bands and offer some noise elimination. For the spectrogram in figure 16, the window length and the stride are decided using [1]. For the plot in figure 17, we swept values of window length and overlap to determine the maximum noise elimination.

Part 2-

A constant frequency sinusoid is analysed using the GUI. The frequency is set to 6000Hz. As expected, bands formed in -6000Hz and 6000Hz in the spectrogram plots.

Part a –

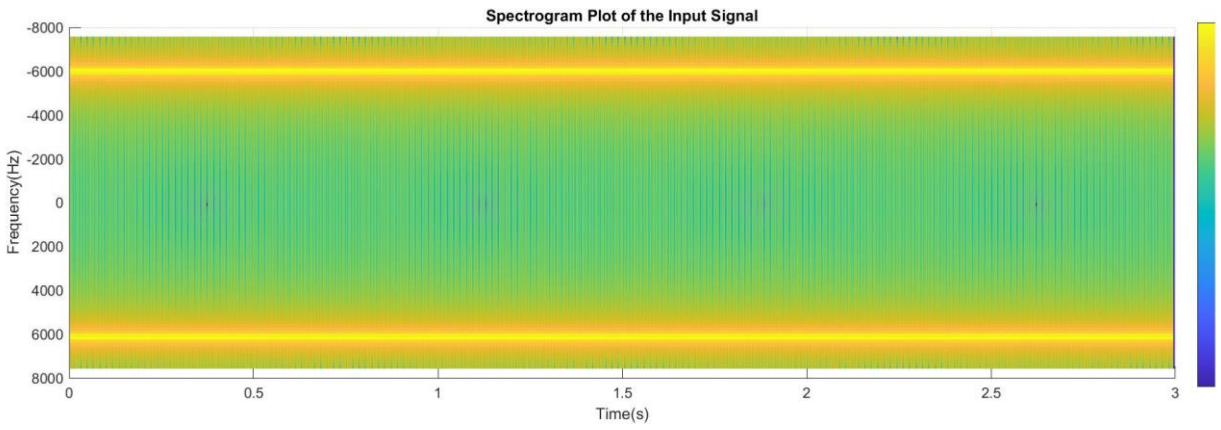


Figure 18 Spectrogram Plot of the Sine Wave, Rectangle Window, Window Length:104, Stride: 52

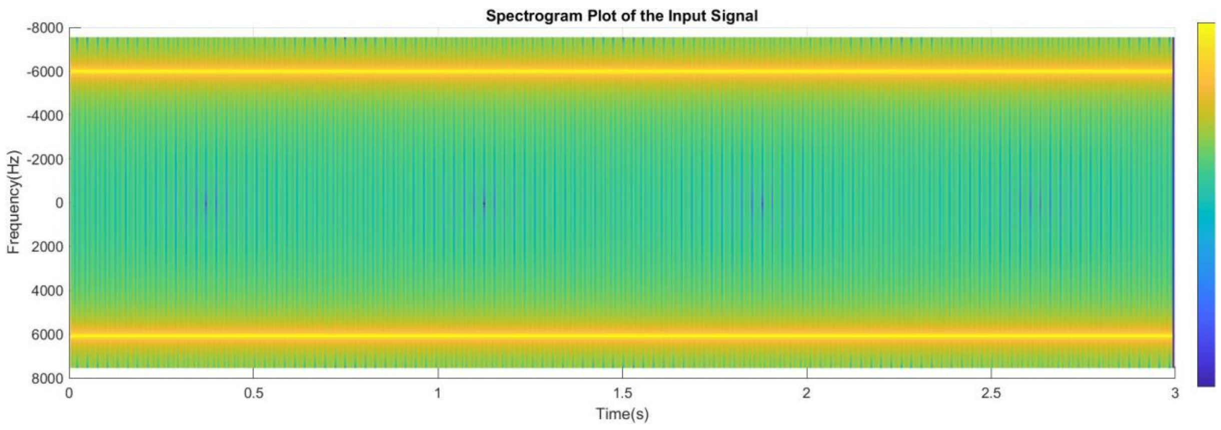


Figure 19 Spectrogram Plot of the Sine Wave, Rectangle Window, Window Length:164, Stride: 82

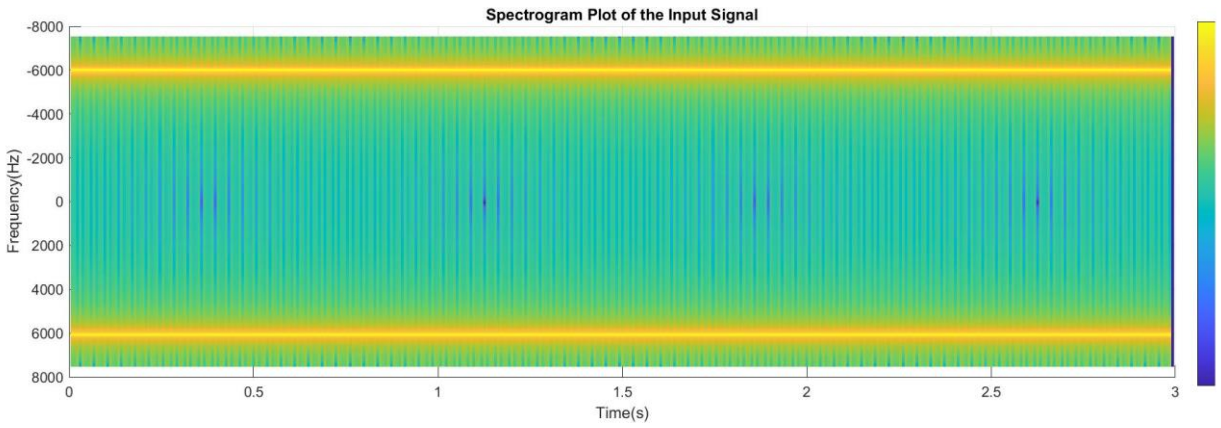


Figure 20 Spectrogram Plot of the Sine Wave, Rectangle Window, Window Length:224, Stride: 112

As the window length increases, then resolution of the plots is increased, if figures 18, 19, and 20 are observed. Furthermore, the thickness of the bands is reduced since increasing the window length corresponds to giving more information about the sinusoid being sinusoid to the frequency extracting operation STFT, so to speak.

Part b –

Taking the sampling rate of the sinusoid (15000Hz), frequency of the sinusoid (6000) into account, the window time lengths(s) are set to integer multiples of the period of the sinusoid. We took window lengths number of samples to be 50, 100, and 500, and for each one of them, stride is chosen so that windows do not overlap.

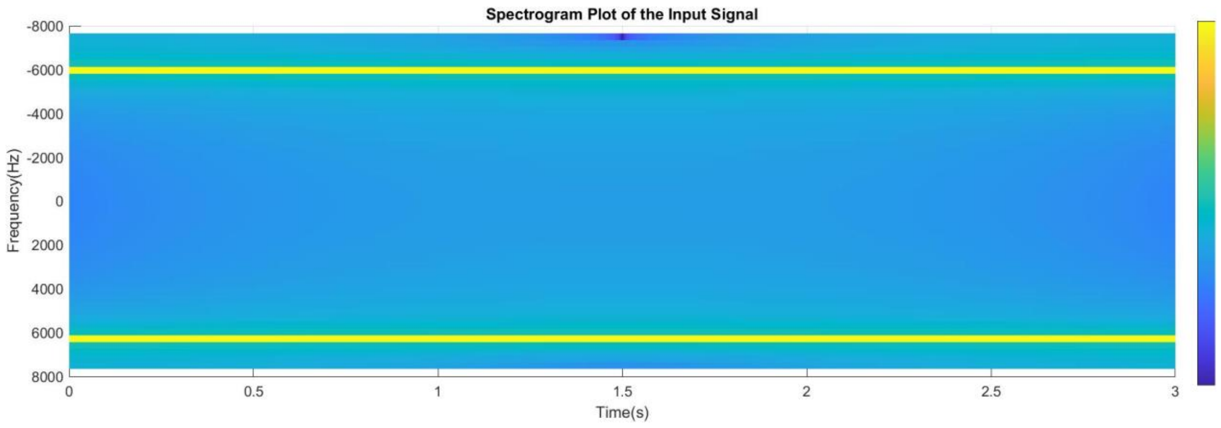


Figure 21 Spectrogram Plot of the Sine Wave, Rectangle Window, Window Length:50, Stride: 50

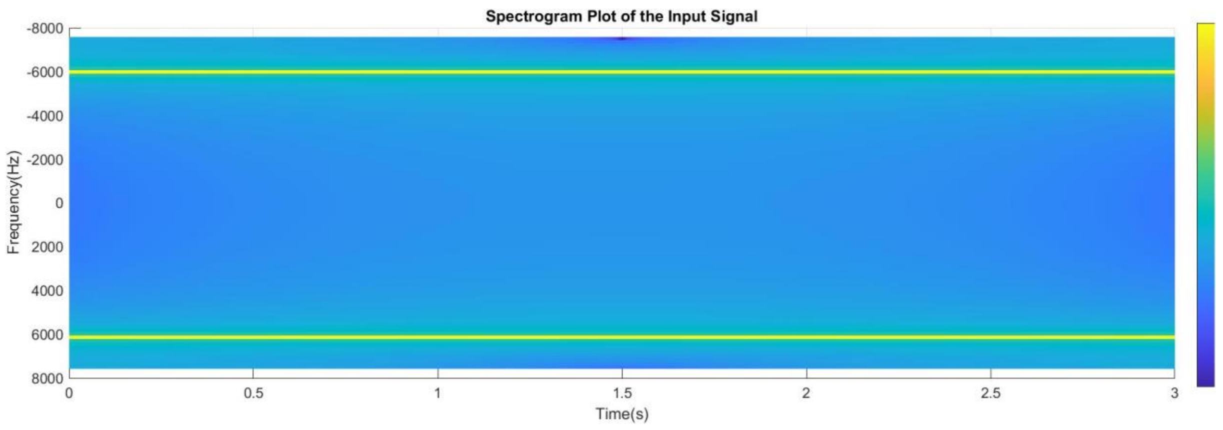


Figure 22 Spectrogram Plot of the Sine Wave, Rectangle Window, Window Length:100, Stride: 100

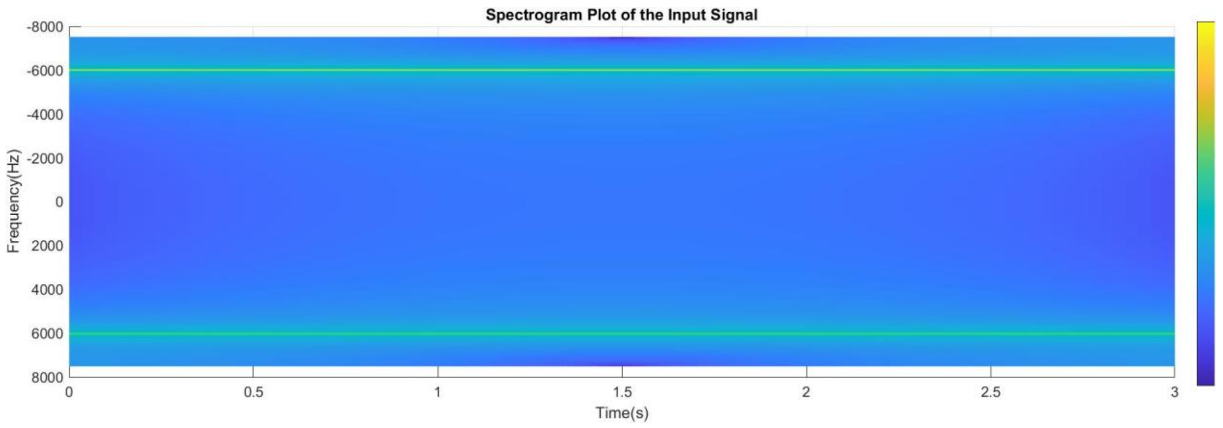


Figure 23 Spectrogram Plot of the Sine Wave, Rectangle Window, Window Length:500, Stride:500

Part c –

Comparing the spectrogram plots obtained in part-a and part-b, choosing the window length almost obliterates the noise. The reason for this is that for the case the window length is chosen to be integer multiple of the sinusoid, enclosed signal contains whole number of full periods of the sinusoid. Hence, ideally, the DFT result is only nonzero at the frequencies of the sinusoid. Other case is equivalent to the time shift applied on the sinusoid to be inputted the DFT operation, and the window do not contain whole number of periods inside. The portion of the signal at the end would act as an imperfect partial sinusoid since it is not completed to its full period. This is the reason for the noisy behaviour between the bands in the plots given in part-a.

Conclusion

Spectrograms are valuable plots and find their applications in various domains of signal processing. They provide localized spectrum characteristic of time domain signal, and the time vs. frequency colour coded plot offers regional insights that might not be obtained by treating the signal as a whole. In this project, we obtained spectrogram plots of audio signals that are either generated artificially or recorded. It is seen that by observing the spectrogram plots, we can roughly point out the letters of a word contained in a speech signal. By also observing the windowed sinusoidals and signals that are superposition of sinusoidals, we see that spectrogram plots extracts the time dependent frequency characteristic of the dissected time domain signal. Forming of bands provides the knowledge about the existence of a certain frequency, and the strength of it in a certain region. Furthermore, spectrogram plots can be treated as frequency related meaningful images about the signals. They can be used to train CNNs since signal dependent patterns exists and they extracted from these images.

References

[1] : Vazhenina, D., &amp; Markov, K. (2020). End-to-end noisy speech recognition using Fourier and Hilbert spectrum features. Electronics, 9(7), 1157. https://doi.org/10.3390/electronics9071157