Humeyra Bodur (2442606)
Berfin Kökmen (2443463)

07.12.2023

EE430 Term Project

Part 1

Introduction

In this report, we delve into the analysis of speech signals using a MATLAB-based graphical user interface (GUI). The system captures and processes audio data from a microphone or existing files and offers the unique capability of generating time-domain signals through mathematical expressions. The user-friendly GUI facilitates parameter adjustments during data acquisition, generation, and spectrogram analysis, emphasizing the importance of spectrogram insights into temporal and frequency characteristics. By carefully tuning parameters like window length, shift, and type, we aim to enhance the interpretability of spectrograms.

Data Acquisition

The data acquisition module is for capturing audio inputs in real-time from a microphone and from existing sound files. It allows users to record and process sound data, forming a flexible foundation for subsequent signal analysis within the project.

Figure 1 illustrates the main page, showcasing the three buttons: "Record Audio," "Upload Audio File" and "Generate Signal." Each button serves as a portal, guiding users to distinct tabs within the MATLAB app.

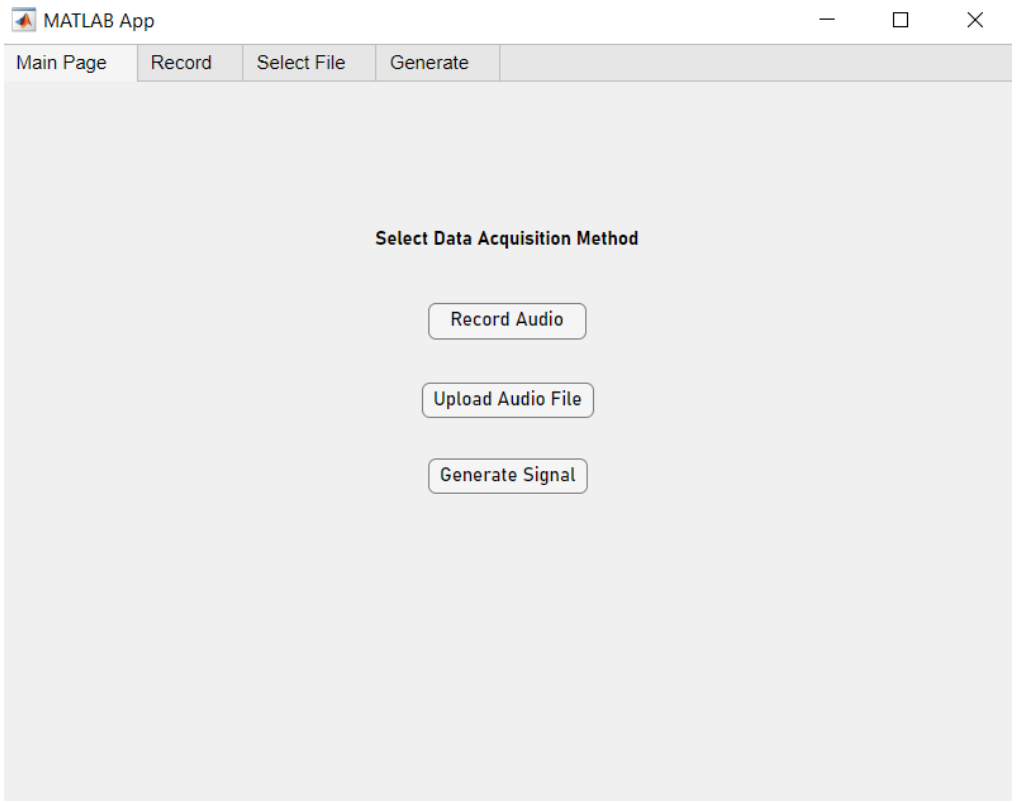


Figure 1. Main Page

Sound Data from a Microphone:

This data acquisition module is designed to capture sound data from a microphone connected to the computer. The '*captureAudio'* function facilitates this process, allowing the system to record the user's voice or audio playback from an external device, such as a mobile phone. One of the notable features of this function is its flexibility in arranging various sampling rates. The user interface allows for the adjustment of the analog-to-digital conversion sampling rate. Additionally, users can specify the duration for which the audio is recorded, offering further customization. The captured audio data is then saved to a file named 'LastSaved.wav' in the current working directory, ensuring a persistent record of the acquired sound data.

Upon pressing the 'Record Audio' button on the Main Page, the MATLAB App navigates users to the Record Audio Tab, as depicted in Figure 2. To record audio in real-time, users must input numerical values for sampling rate and recording time. After pressing the record button, the *'captureAudio'* function is executed, indicated by a change in the button color from green to red (see Figure 3), signifying that the computer is actively recording audio. Once the recording is complete (the button turns green again), users can play the recorded audio by clicking the play button which calls the '*openAudio*' function. When recording process is completed, users can plot the spectrogram of the recorded audio. To achieve this, specific inputs, such as window length, window shift (hop size), and window type, should be provided in the lower-left part of the screen. Users input numeric values for window length and window shift, and a dropdown menu allows the selection of window types ('rectangular,' 'tukey,' or 'hamming'). Upon pressing the 'Plot Spectrogram' button, the sound signal is first decimated in order to not deal with big data sizes. Then the *'spectrogram_plotter_sound_GUI'* function executes, plotting the spectrogram graph of the recorded audio in the white space located in the lower-right part of the screen (see Figure 4).

Figure 2. Record Audio Tab

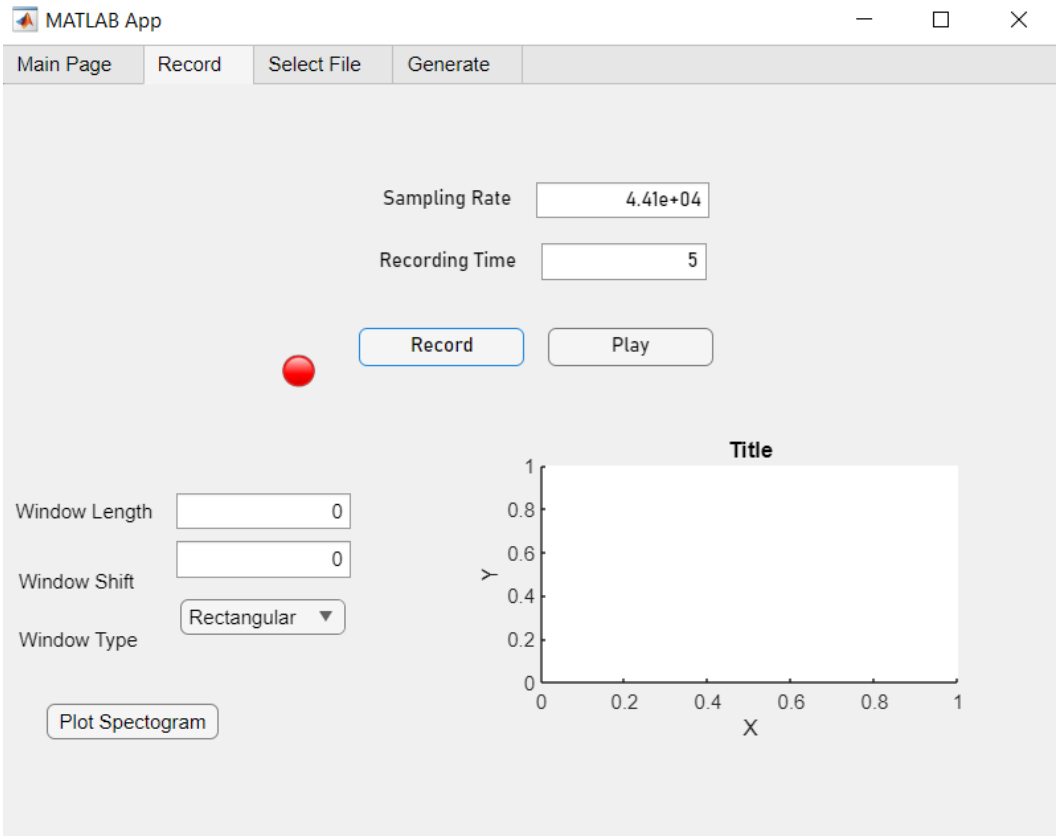


Figure 3. During recording.

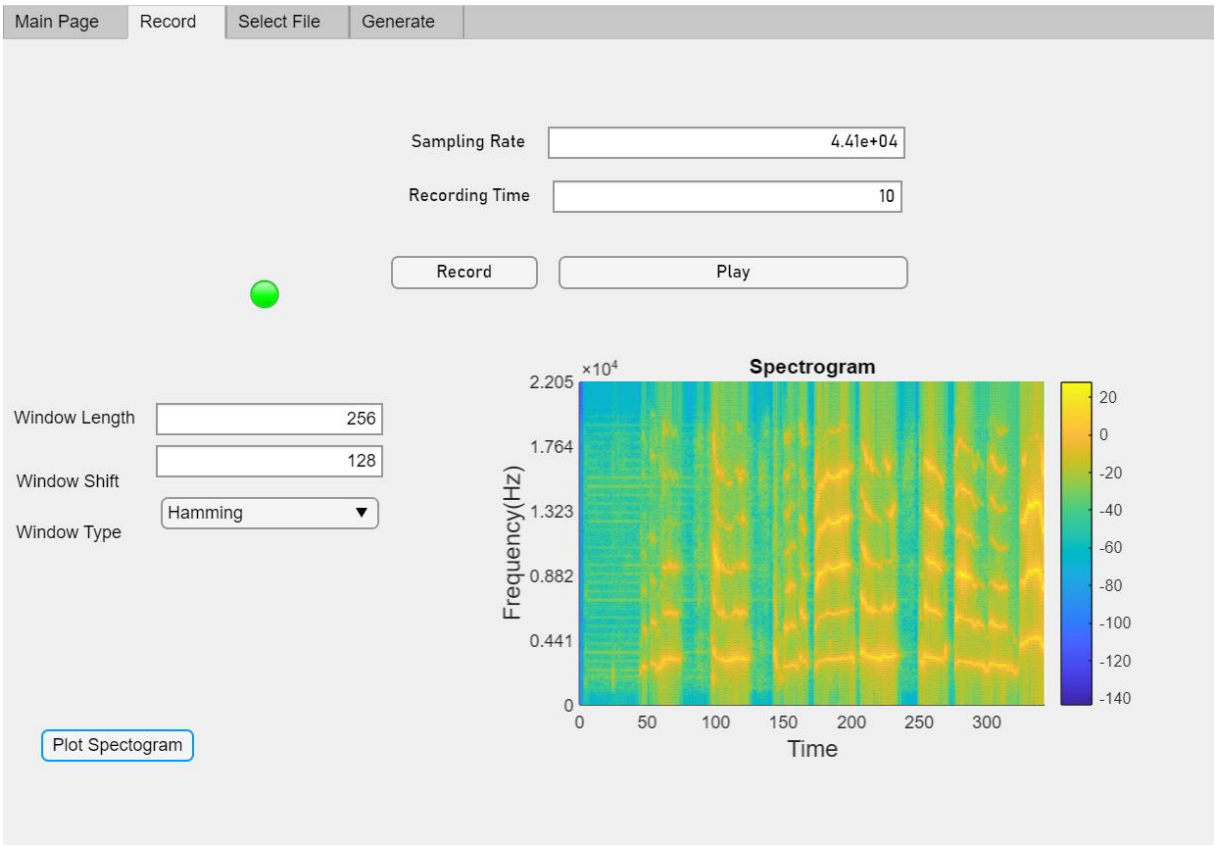


Figure 4. Plotting Spectrogram of recorded audio

Sound data from a file:

To handle an existing file in '.mp3' or '.wav' format, the Select File Tab can be utilized (see Figure 5). After clicking the 'Upload a File' button, the GUI allows users to choose the specific file they wish to process (see Figure 6). Additionally, users can play the uploaded file by clicking the play button, which invokes the *'openAudio'* function. To generate a spectrogram for the uploaded audio file, the process aligns with that of the Record tab. Users need to input values for window length, window shift (hop size), and window type. Upon clicking the 'Plot Spectrogram' button, , the sound signal is first decimated in order to not deal with big data sizes. Then, the *'spectrogram_plotter_sound_GUI'* function will be triggered. Figure 7 shows the spectrogram plot of a sound data of uploaded file.

Figure 5. Select File Tab.

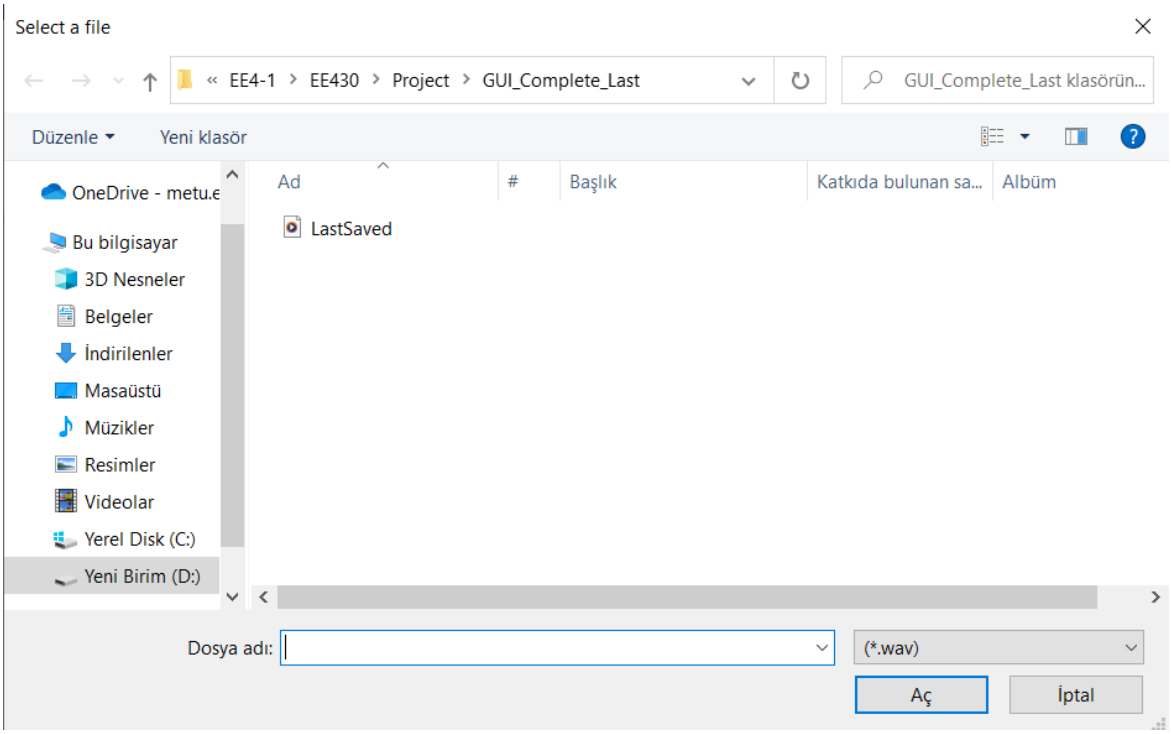


Figure 6. Uploading '.wav' or '.mp3' audio files.

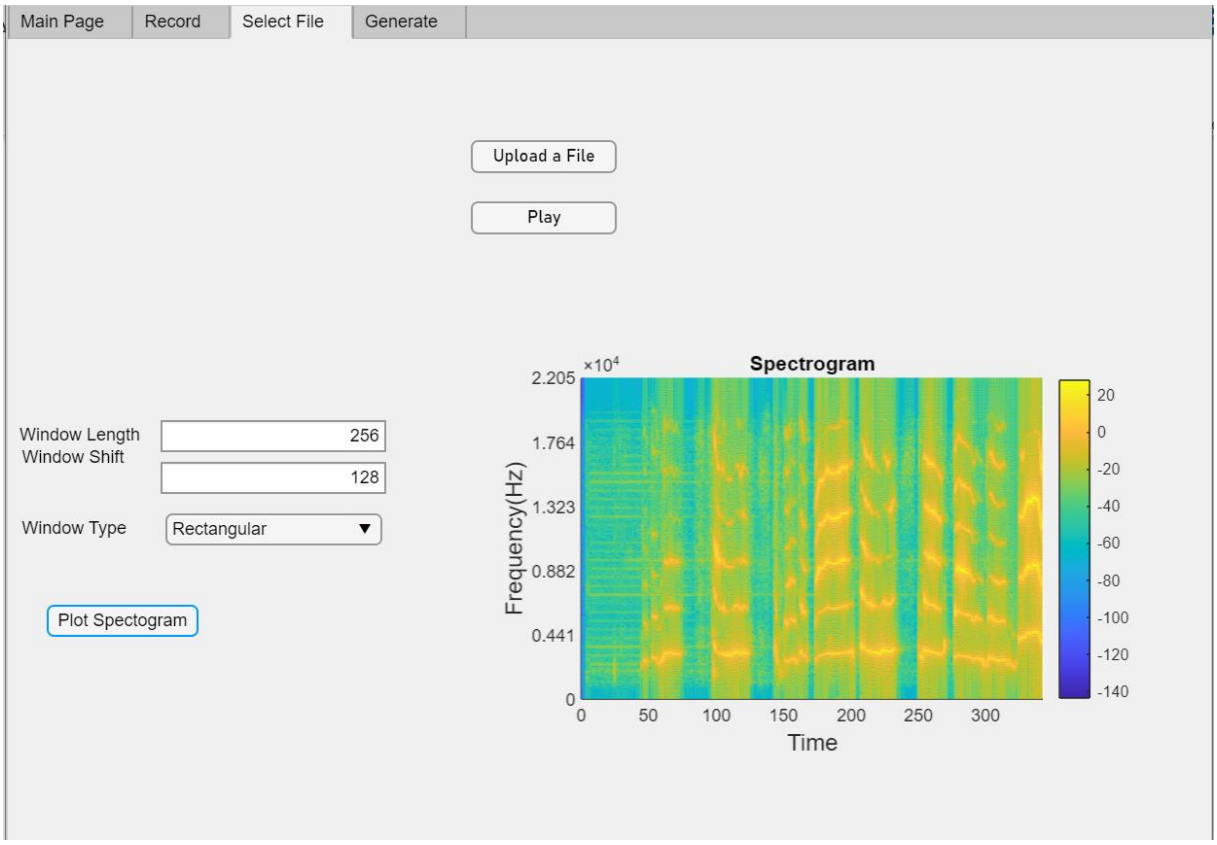


Figure 7. Plotting the spectrogram of already existing audio file.

Data Generation

If the user clicks the "Generate Signal" button on the Main Page, as depicted in Figure 1, the application will navigate the user to the Generate Tab, shown in Figure 8. In the data generation section, the User Interface provides three options for generating data to plot a spectrogram—sinusoidal, windowed sinusoidal, and signals involving multiple components, as specified in the Project Description Document.

In the app, the user needs to select one of these options by checking the corresponding checkbox and then input the required values listed under the chosen option. For generating a sinusoidal signal, users must input amplitude, frequency, phase, and duration. The execution of the *'generate_sinusoidal'* function will generate the sinusoidal signal with the provided input values.

For a windowed signal, the required inputs, including amplitude, frequency, phase, window type, start time, and length, are displayed in Figure 8. To generate the windowed signal, the *'generate_windowed_sinusoidal'* function is executed.

To generate a signal with multiple components, users must first enter the number of components they want to involve in the signal. Subsequently, for each component (denoted as 'm' times), users should input amplitude, frequency, phase. After each input they submit the values and then enter the next ones. The duration remains the same for all signals, i.e., the last duration entered is taken into account. The *'generate_summed_sinusoidals*' function will then sum these signals with different components.

Once the signal is generated using one of these three options, the process of plotting a spectrogram remains the same as in the Record Tab and Select File Tab. The necessary inputs are displayed in the lower-left part of the Generate Tab. Clicking the 'Plot Spectrogram' button triggers the *'plotter_for_generated_signals_GUI'* function, displaying the spectrogram in the left white space, while the other white space shows the generated signal magnitude over time for two periods. Figure 9,10,11 shows the plots for signals of different options of generation.

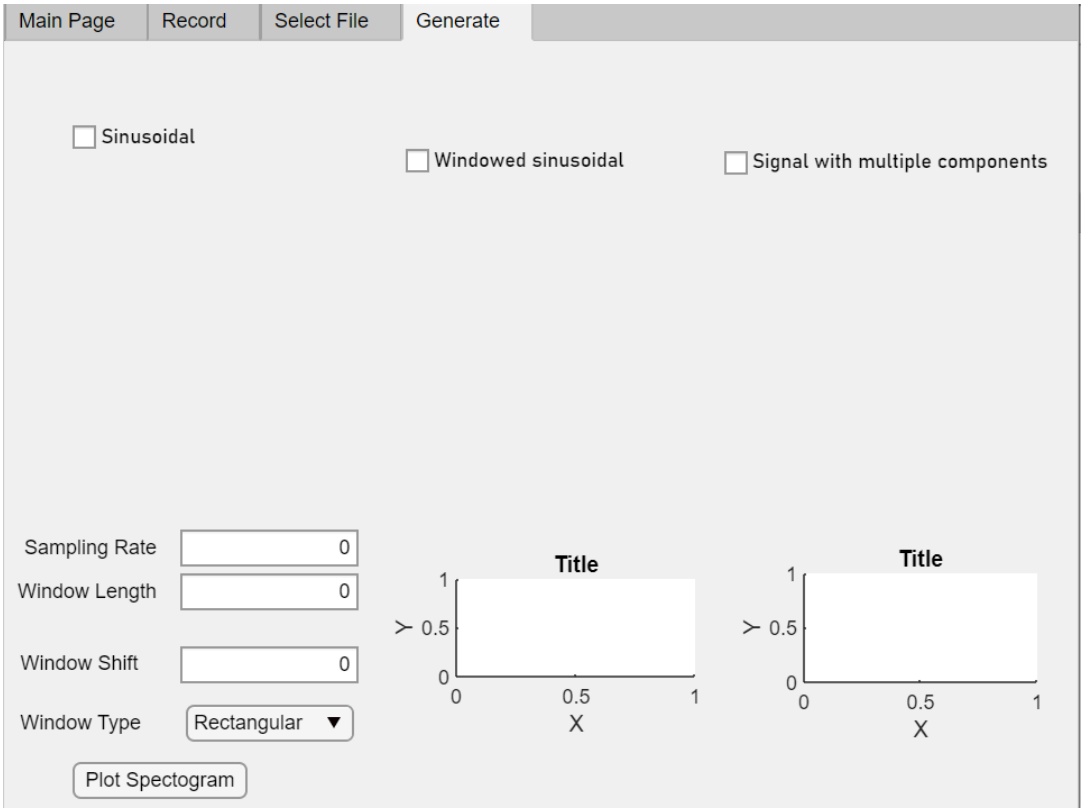


Figure 8. Generate Signal Tab.

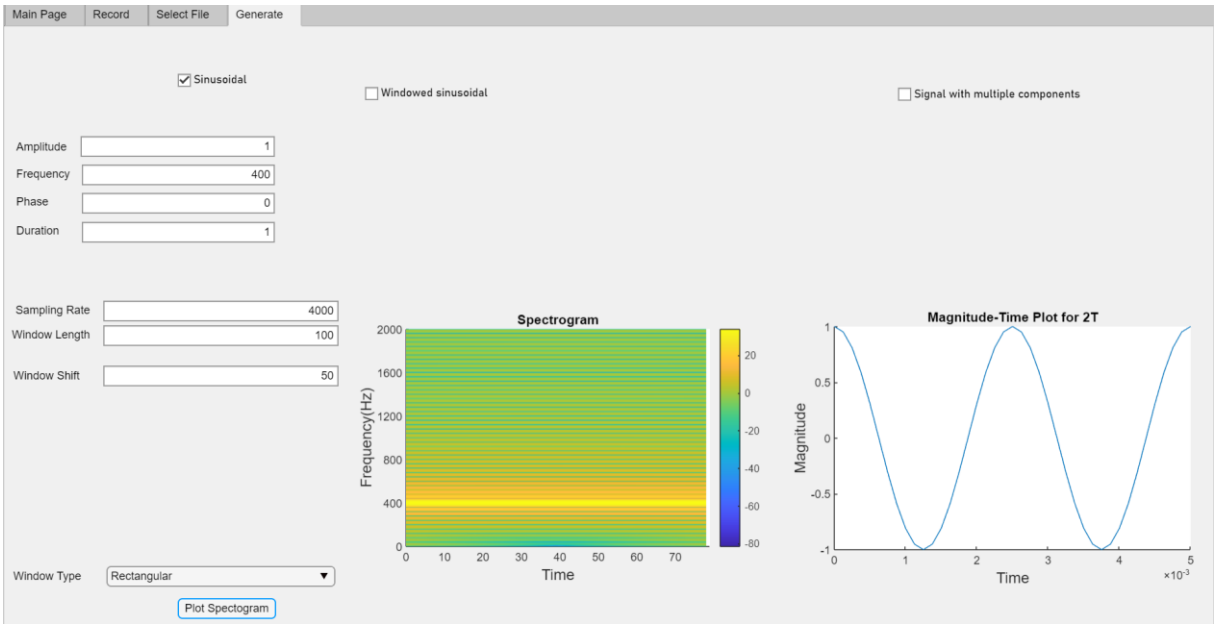


Figure 9. Plots for Generated Sinusoidal.

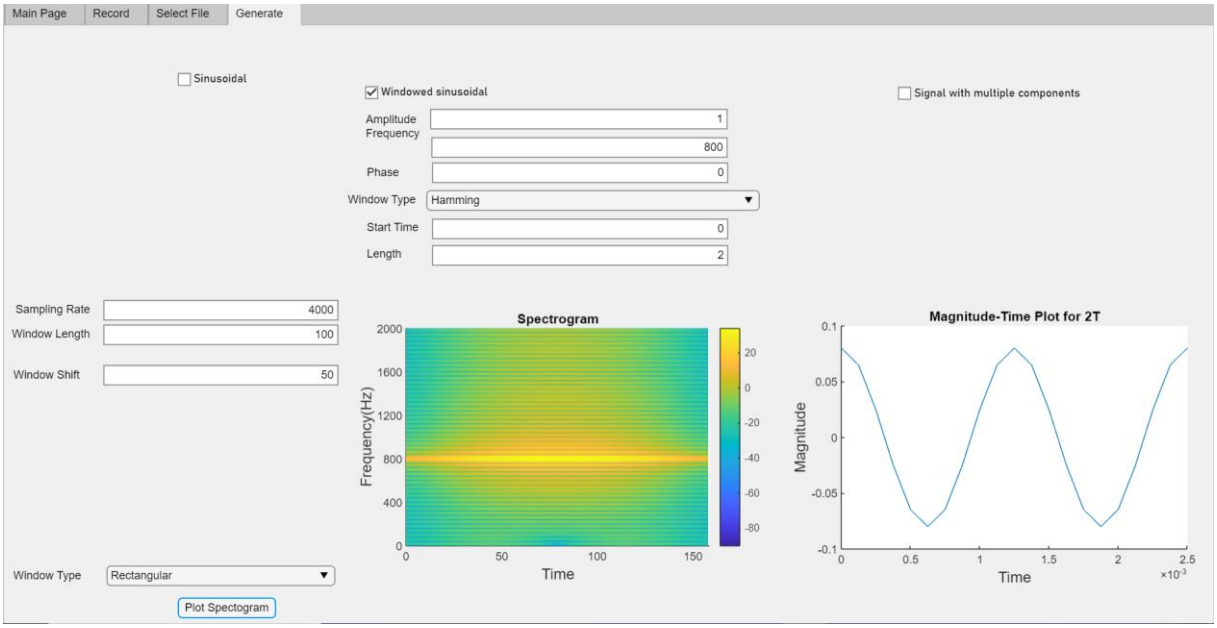


Figure 10. Plots for Generated Windowed Sinusoidal.

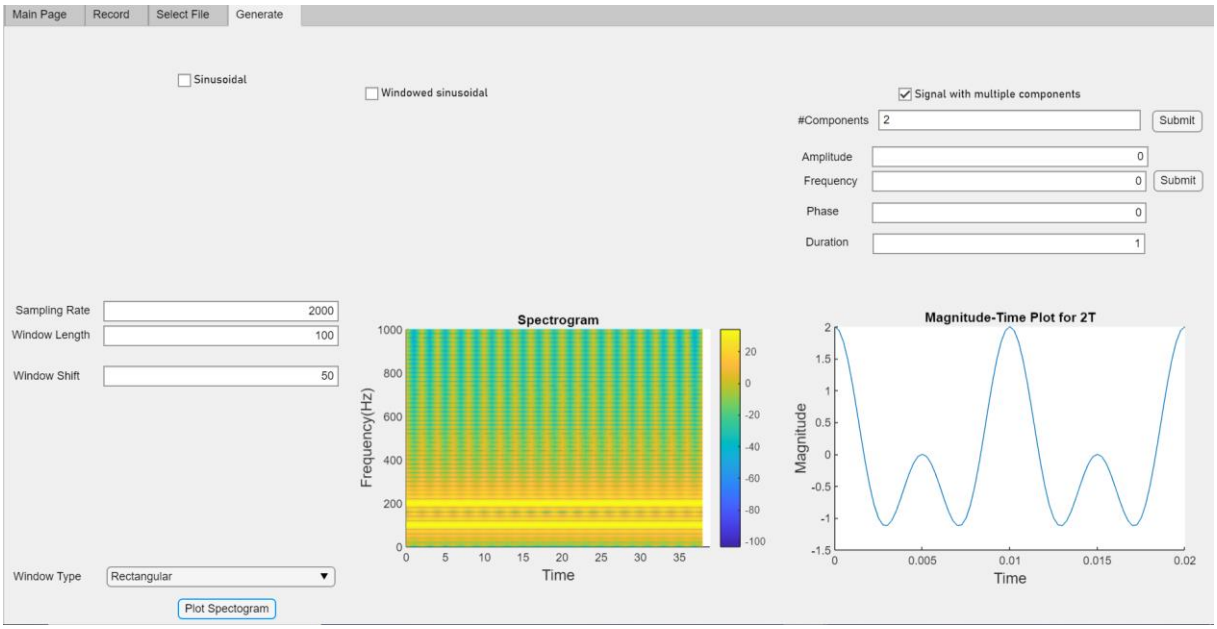


Figure 11. Plots for Generated Signal with Multiple Components.

Spectrogram

A spectrogram is basically a visual representation of a signal's frequency content over time, which is obtained through the Short Time Fourier Transform (STFT). STFT is a time-varying extension of the Discrete Fourier Transform (DFT). In STFT the signal is divided into short overlapping windows, and for each window, the DFT is computed. This enables the analysis of the signal's frequency content as it changes over time. A spectrogram consists of 3 axes: time on the x-axis, frequency on the y-axis, and intensity on the so called “color” axis. As can be seen in the previous section of the report, spectrograms play a vital role in signal processing. They help us observe dynamic changes in frequency components in time-varying signals such as sound signal and sinusoidal signals of different types.

In our implementation of it, this is done through the function “*spectogram_plotter_generated_signals_GUI”* or *“spectogram_plotter_sound_GUI”*. It takes the axes, signal, window length, window type and window size, and sampling frequency as input. It has no return value; the plotting is done inside the function. The function first does some basic check such as input shape, window length constraints to avoid errors. Then the window function is assigned according to the user input. The STFT is taking in a for loop. First windowing takes place where we take a signal segment of the signal and apply the chosen window function. The Fourier Transform of the chosen signal section is then computed and its magnitude is then assigned to the STFT matrix. After this is completed the next signal segment is chosen based on the window shift. The process is repeated until the end of the signal. Lastly the surface plot is plotted.

The significance of parameters such as window type and overlap cannot be understated in the context of STFT. The choice of window function also plays a crucial role in shaping the characteristics of the spectral analysis. For instance, the Hamming window applied in this

implementation helps us handle spectral leakage and enhances the accuracy of frequency representation. The window shift should be set to at most equal to the window length in order to be able to capture all parts of the signal. This constraint enables the overlap between consecutive windows and ensures a smoother transition in the time-frequency representation and helps us avoid potential distortions in the analysis. Taking these parameters into account before generating a spectrogram increases the reliability and interpretability of that spectrogram. Further analysis of these parameters will be presented in the 2nd question.

Question 1

The appropriate sampling rate for a sound signal in the human audible range is typically set to at least 40 kHz, adhering to the Nyquist-Shannon sampling theorem. This ensures that frequencies up to 20 kHz, the upper limit of human hearing, can be accurately represented in the digital signal. The sampling rate of the recording would depend on the specifications of the recording device or software. Sampling rate for my recorded audio was 44.1 kHz. For this question, the spoken word was 'Duration'. The letters are roughly located in order between 0.6 seconds to 1.4 seconds. During speaking the letter 'A', the graph made a peak. The used parameters are shown in the Figure 14. For window length is 256 and window shift is 128.

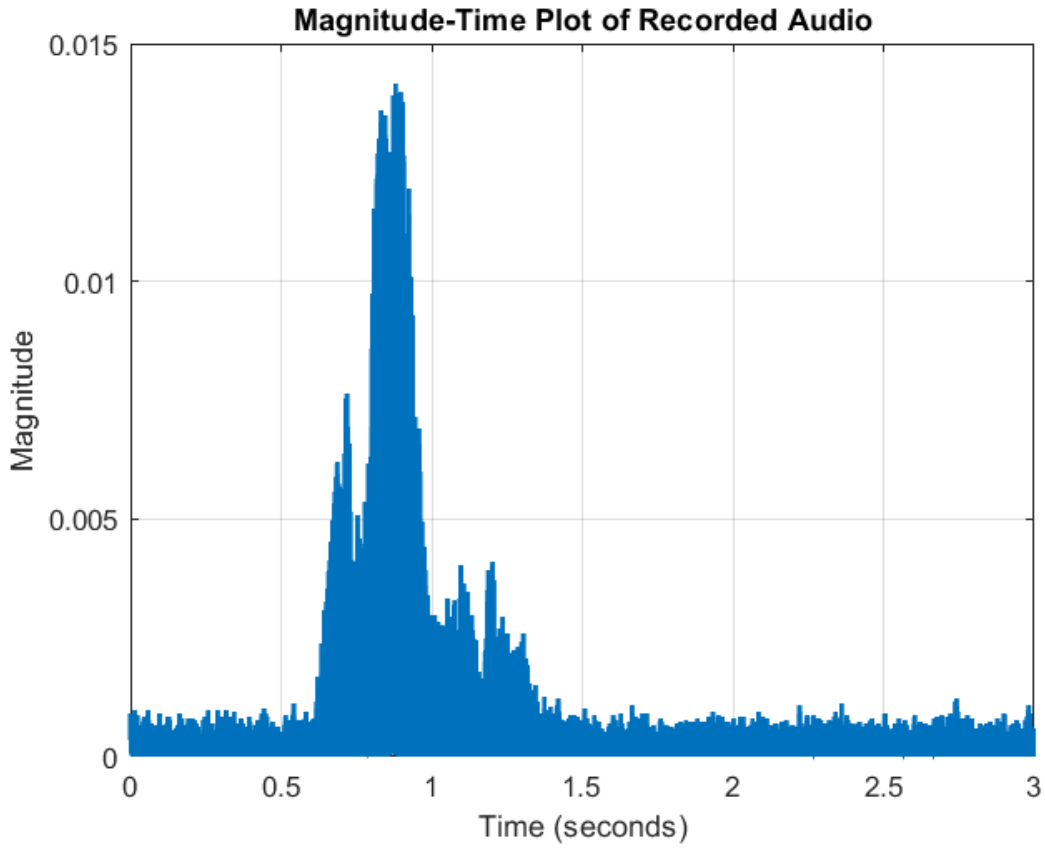


Figure 12. Magnitude-Time Plot of Recorded a Spoken Word Audio.

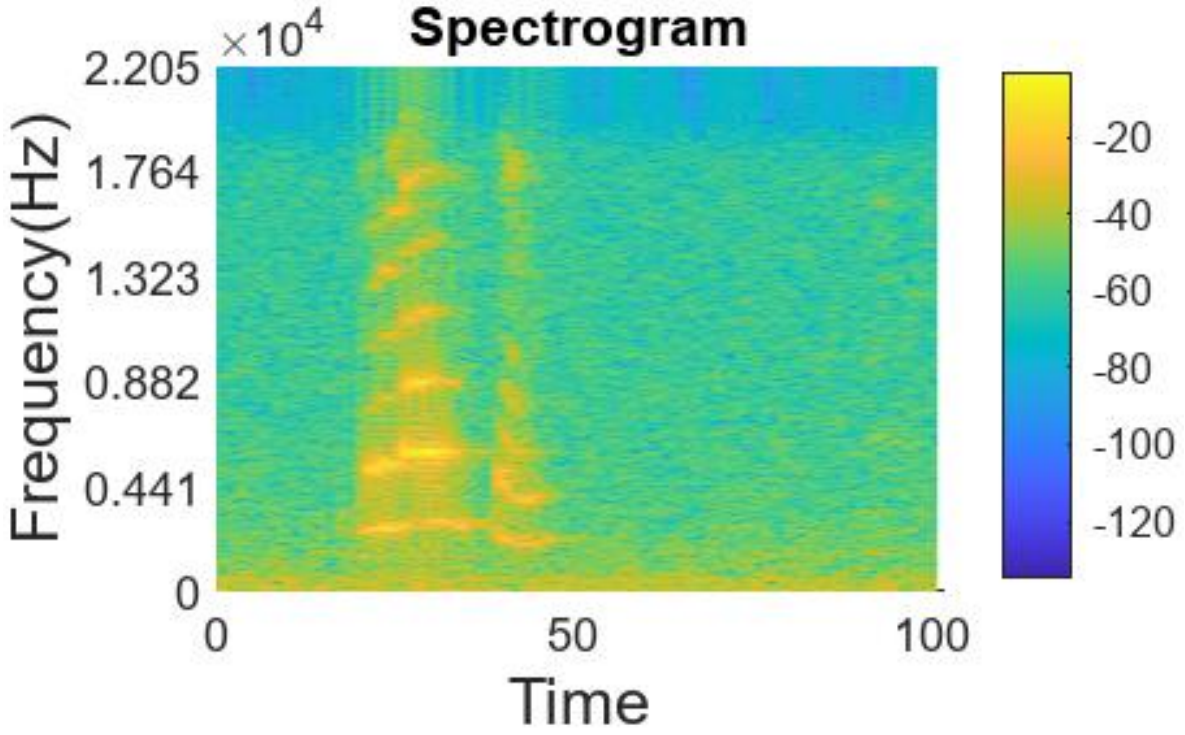


Figure 13. Spectrogram Plot of Recorded a Spoken Word Audio.

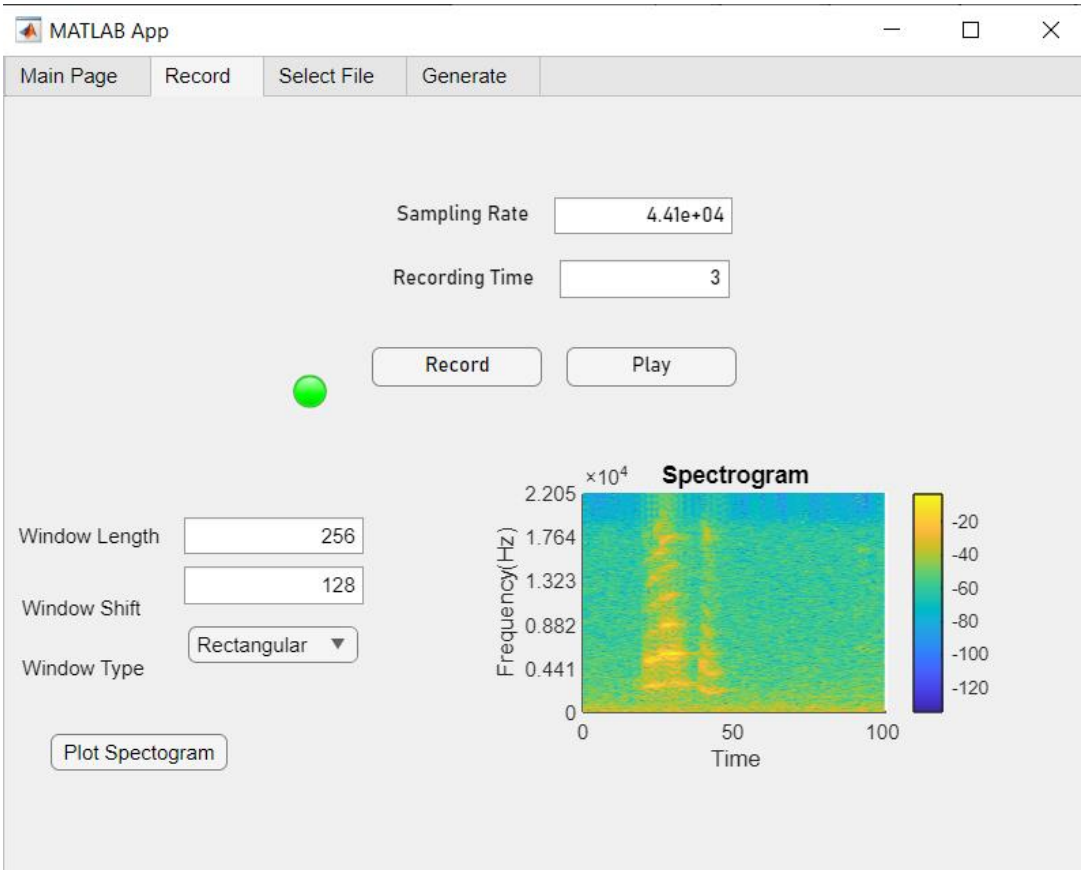


Figure 14. The parameters used for recording a word.

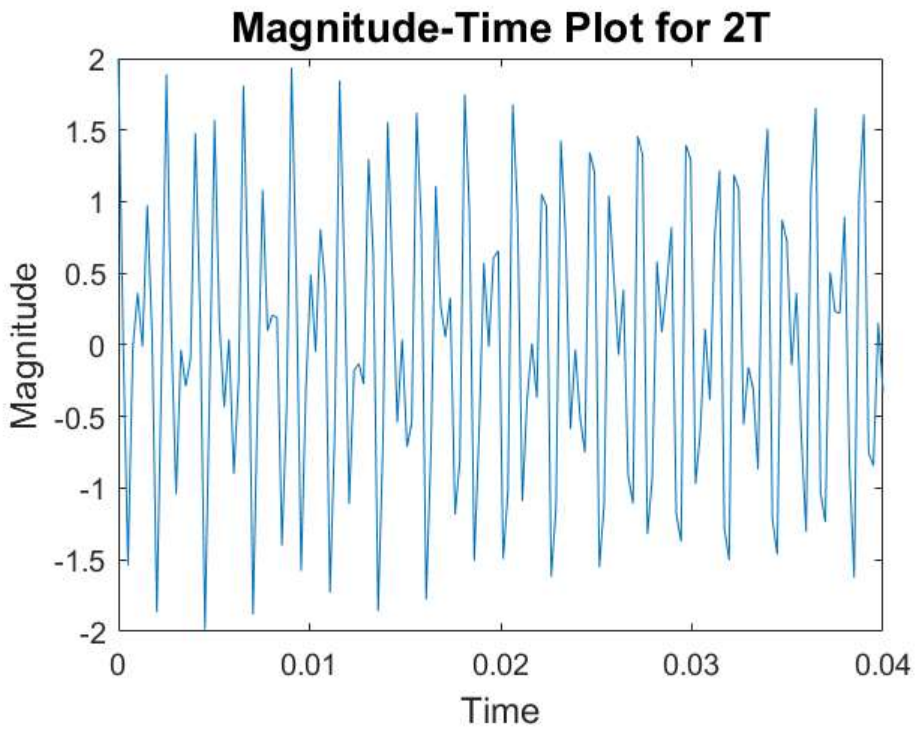
Q2)

i.

```
sampling_frequency_div2 = 2000;
amplitude_m = [1 1];
phase_m = [0 0];
total_duration = 150*10^(-3)
```

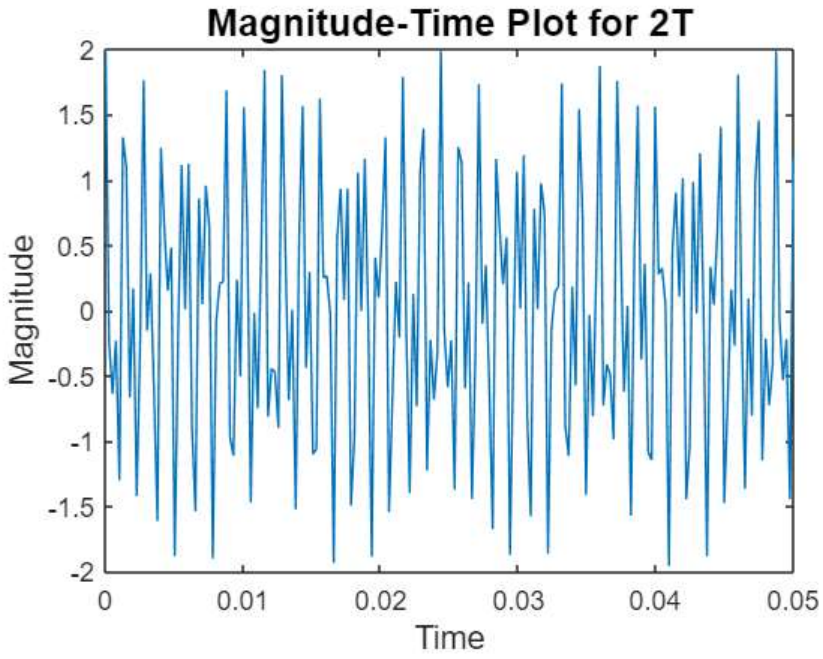
```
total_duration = 0.1500
```

```
% First we generate function handles of summed sinusoidals using generator_summed_sinusoids

% Function handle 1: 770 Hz & 1209 Hz, 40ms;
frequency_m_1 = [770 1209];
handle_1 = generate_summed_sinusoidals(2,amplitude_m,frequency_m_1,phase_m);
duration_1 = 40*10^(-3);

t = linspace(0,duration_1,2*sampling_frequency_div2*duration_1);

sampled_1 = handle_1(t);

figure
plot(t,sampled_1);
xlabel('Time','Fontsize',12);
ylabel('Magnitude','Fontsize',12);
title('Magnitude-Time Plot for 2T','Fontsize',14);
xlim([0 duration_1]);
```

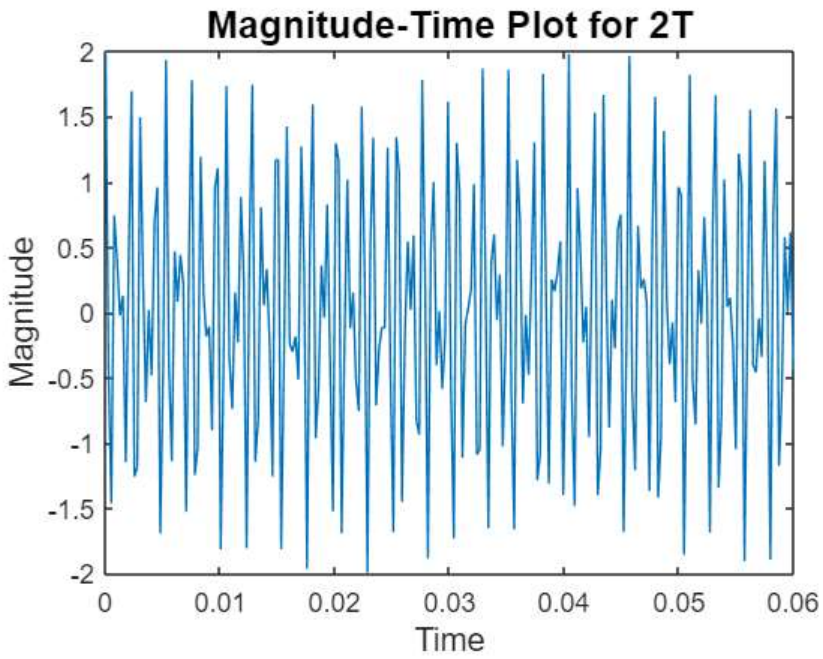



```
% Function handle 2: 697 Hz & 1477 Hz, 50ms;
frequency_m_2 = [697 1477];
handle_2 = generate_summed_sinusoidals(2,amplitude_m,frequency_m_2,phase_m);
duration_2 = 50*10^(-3);

t = linspace(0,duration_2,2*sampling_frequency_div2*duration_2);

sampled_2 = handle_2(t);

figure
plot(t,sampled_2);
xlabel('Time','Fontsize',12);
ylabel('Magnitude','Fontsize',12);
title('Magnitude-Time Plot for 2T','Fontsize',14);
```

```
xlim([0 duration_2]);
```




```
% Function handle 3: 941 Hz & 1336 Hz, 60ms;

frequency_m_3 = [941 1336];
handle_3 = generate_summed_sinusoidals(2,amplitude_m,frequency_m_3,phase_m);
duration_3 = 60*10^(-3);

t = linspace(0,duration_3,2*sampling_frequency_div2*duration_3);

sampled_3 = handle_3(t);

figure
plot(t,sampled_3);
xlabel('Time','Fontsize',12);
ylabel('Magnitude','Fontsize',12);
title('Magnitude-Time Plot for 2T','Fontsize',14);
xlim([0 duration_3]);
```




```
% Then we sum these handles according to their corresponding intervals

t = linspace(0,total_duration,2*sampling_frequency_div2*total_duration)
```

```
t = 1×600
         0    0.0003    0.0005    0.0008    0.0010    0.0013    0.0015    0.0018    0.0020    0.0023    0.0025    0.0028    0.0030
```

```
length(t)
```

```
ans = 600
```

```
summed_signal = zeros(1,length(t));
size_of_signal = length(t);

end_of_signal_1 = size_of_signal*duration_1/total_duration
```

```
end_of_signal_1 = 160
```

```
end_of_signal_2 = size_of_signal*duration_2/total_duration + end_of_signal_1
```

```
end_of_signal_2 = 360
```

```
summed_signal(1:end_of_signal_1)
```

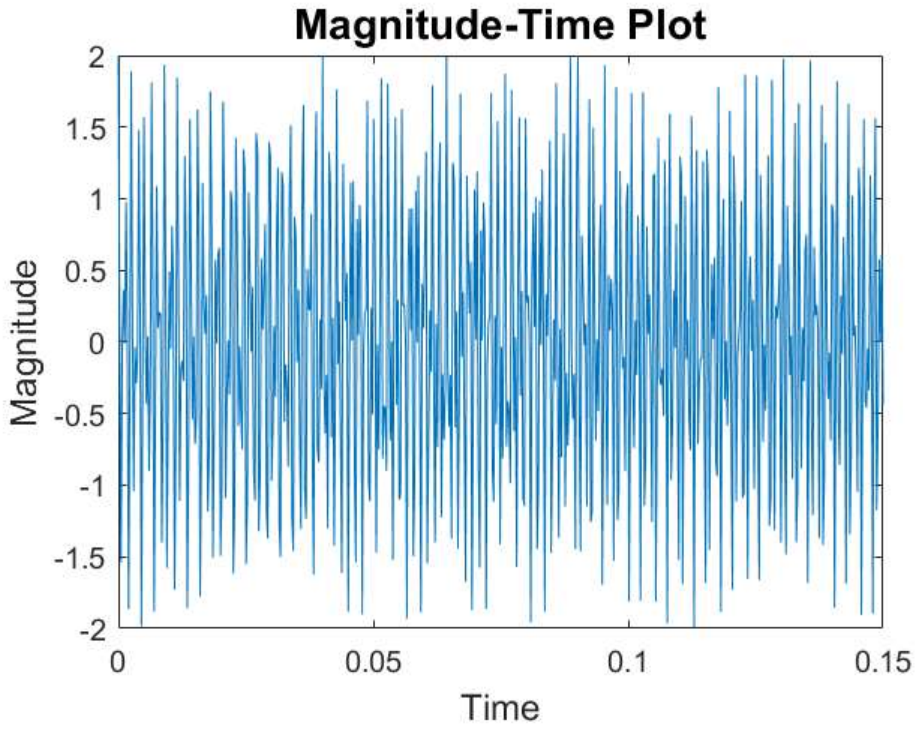
```
ans = 1×160
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
```

```
sampled_1
```

```
sampled_1 = 1×160
    2.0000    0.0126   -1.5374   -0.0204    0.3638   -0.0110    0.9772    0.0711   -1.8649   -0.1208    1.8891    0.1153   -1.0401
```

```
summed_signal(1:end_of_signal_1) = sampled_1;
summed_signal(end_of_signal_1+1: end_of_signal_2) = sampled_2;
summed_signal(end_of_signal_2 +1:size_of_signal) = sampled_3;
```

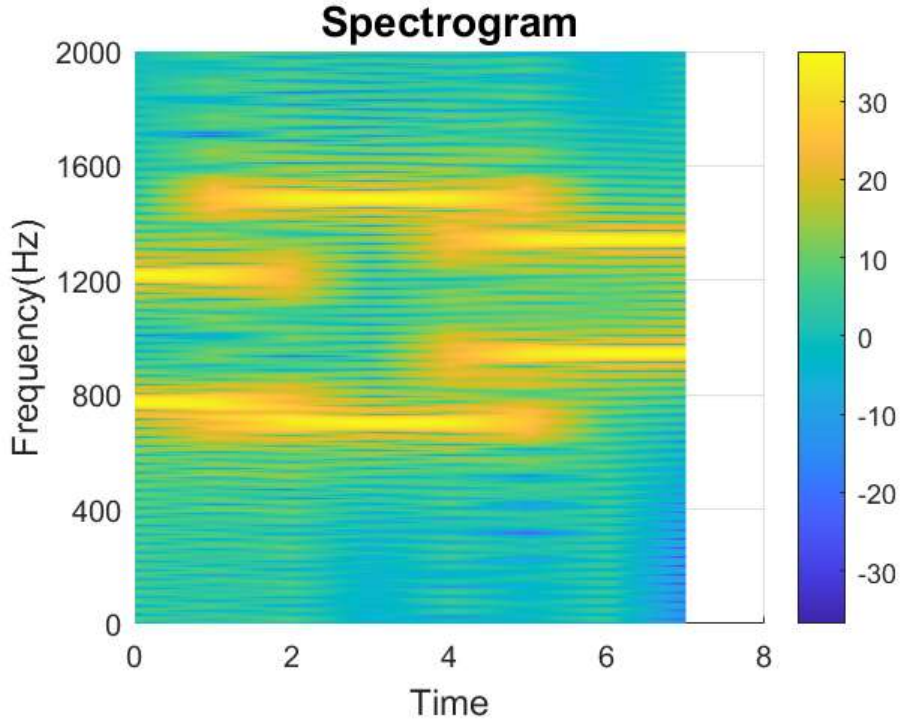
```
figure
plot(t,summed_signal);
xlabel('Time','Fontsize',12);
ylabel('Magnitude','Fontsize',12);
title('Magnitude-Time Plot','Fontsize',14);
```



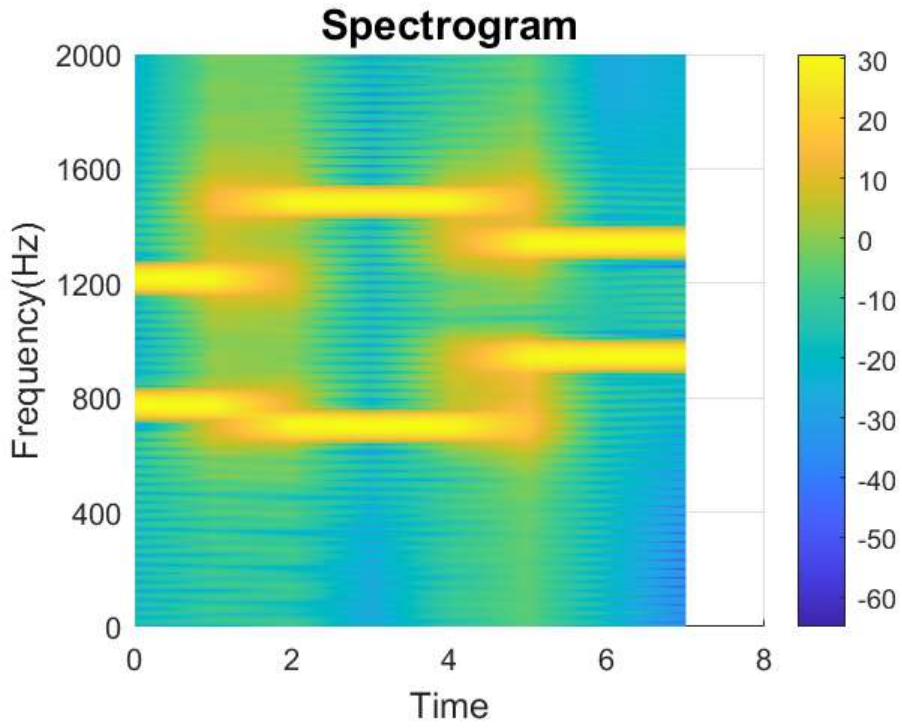

```
% xlim([0.035 0.055]);
```

Window Type Comparison:

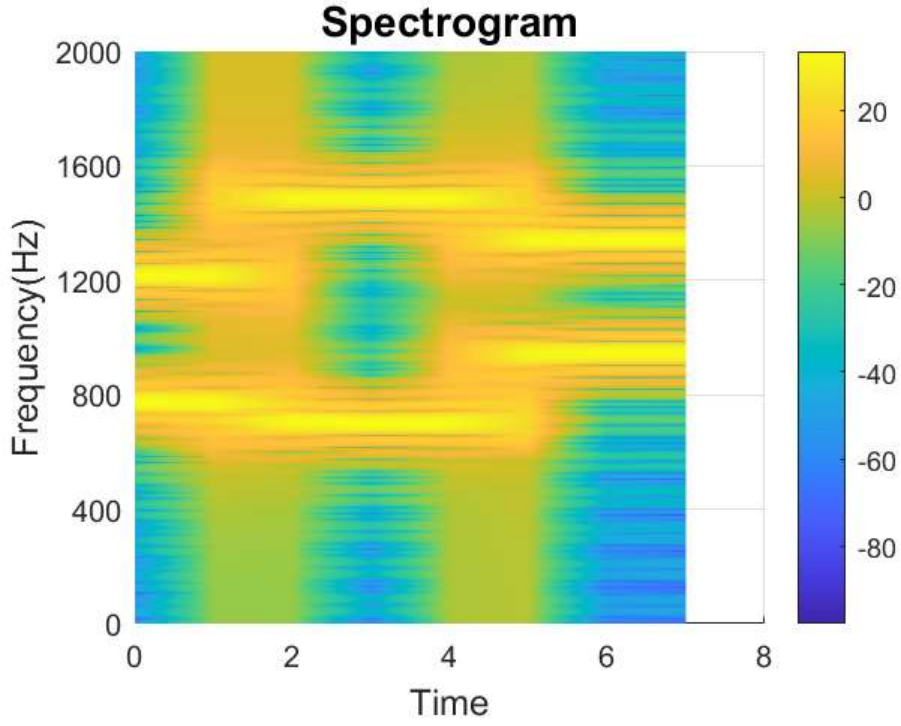
```
window_length = 128;
window_shift = 64;

figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'rectwin',sampling_frequency_div2)
```




```
figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'hamming',sampling_frequency_div2)
```



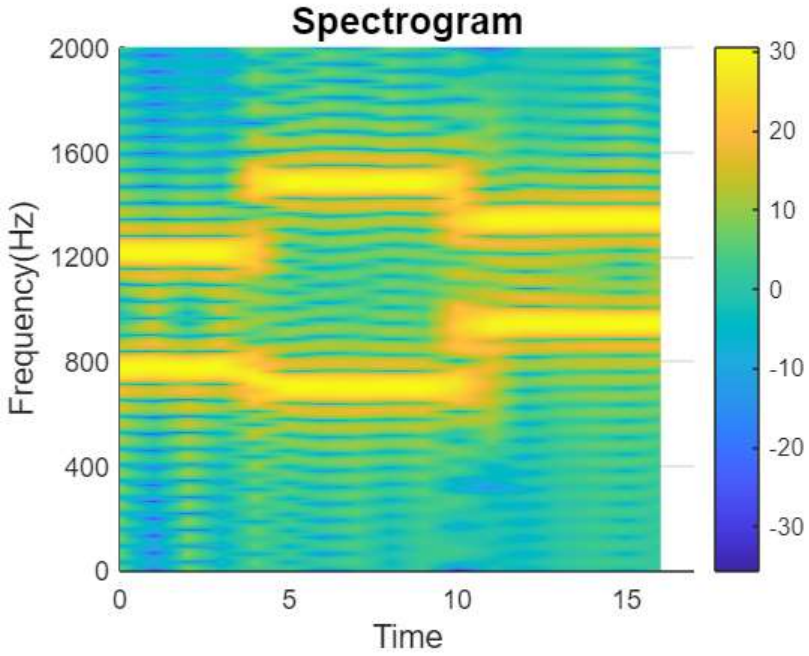

```
figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'tukeywin',sampling_frequency_div2)
```



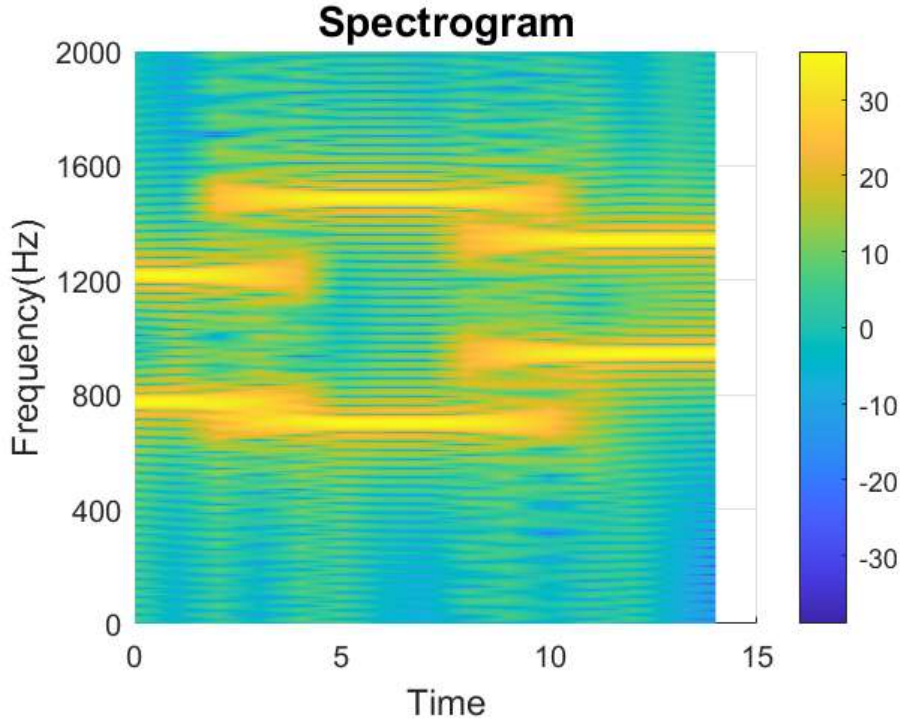
Comments: The main lobe becomes of the window in Hamming is wider compared to the Rectangular window, which leads to a decrease in time resolution. However since the side lobes become smaller as we go to the edges, which increases the frequency resolution. This is also the case in Tukey window where the side lobes get smaller. However, this improvement in frequency resolutioncauses a wider main lobe in the time domain, which can reduce time resolution as can bee seen in the image. The highest frequency resolution is observed for Hamming window whereas the highest time resolution is observed for Rectangular.

Window shift constant, window length changes:

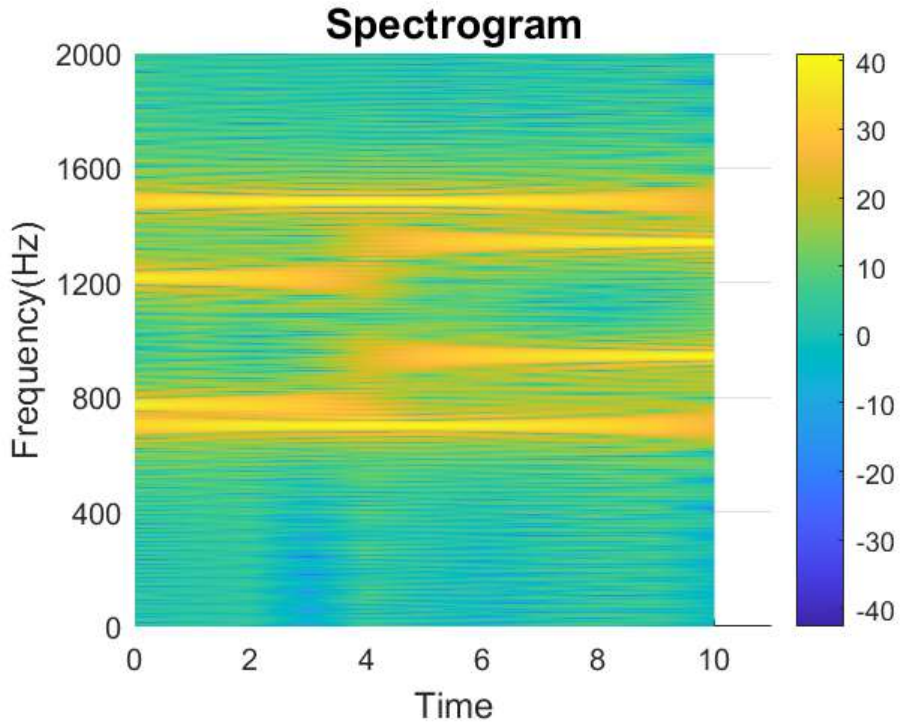
```
window_length = 64;
window_shift = 32;

figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'rectwin',sampling_frequency_div2)
```



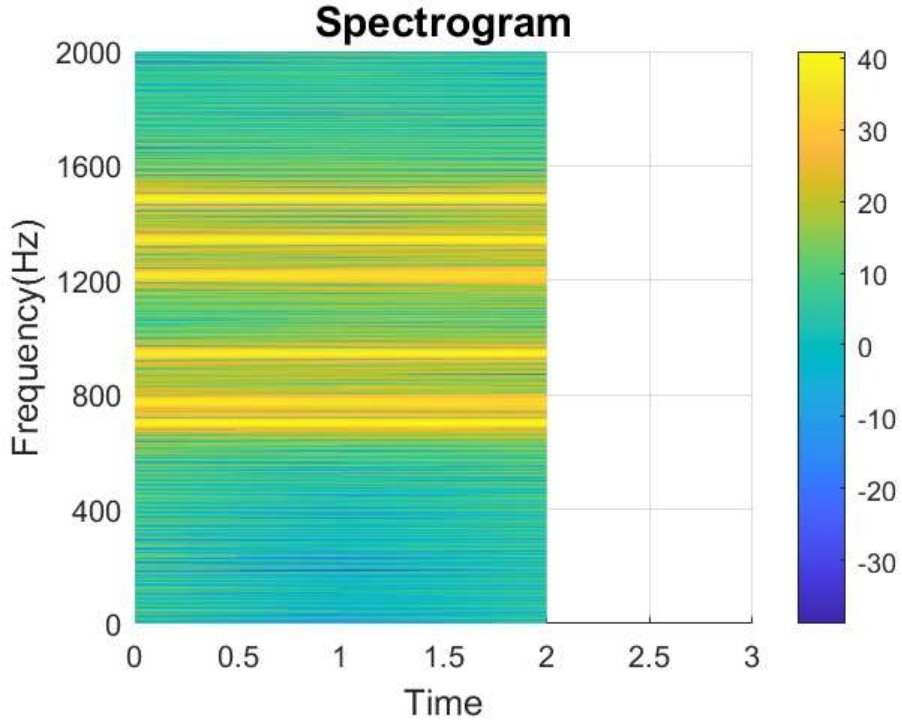

```
window_length = 128;
window_shift = 32;

figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'rectwin',sampling_frequency_div2)
```




```
window_length = 256;
window_shift = 32;

figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'rectwin',sampling_frequency_div2)
```



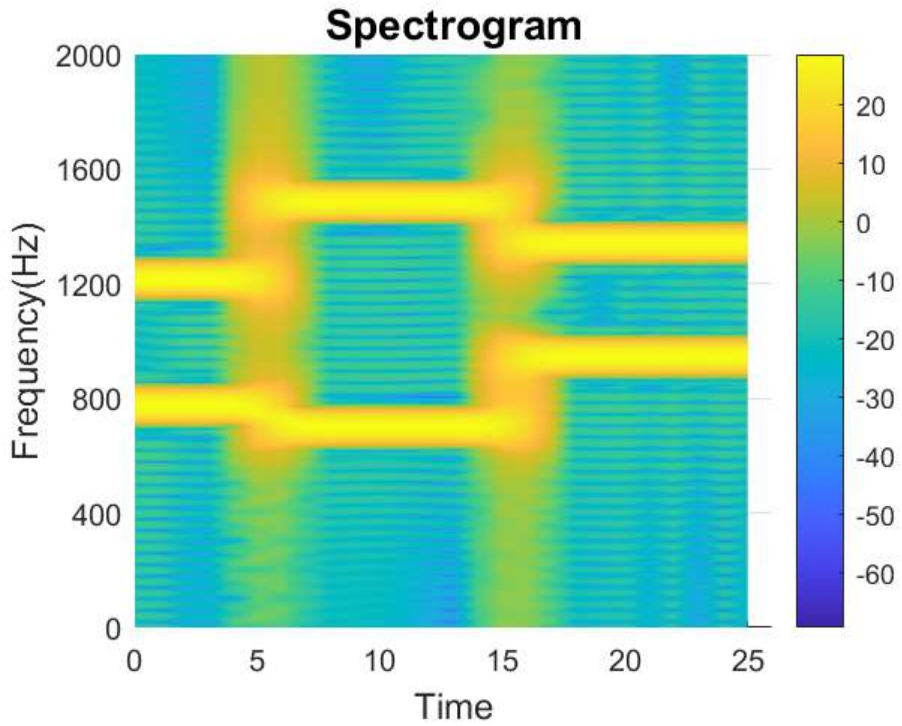

```
window_length = 512;
window_shift = 32;

figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'rectwin',sampling_frequency_div2)
```



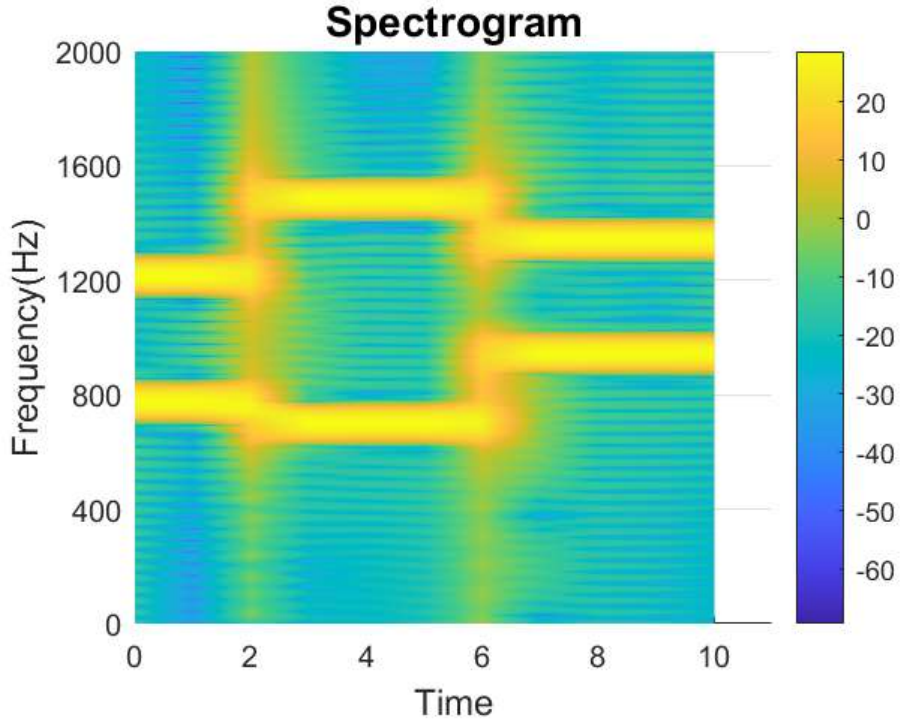
Comments: As window length increases while window shift is sonstant, we observe that the frequency resolution increases while the time resolution decreases. For the last case where window length is 1048, the frequency changing time points are not visible at all. A shorter window will capture a smaller segment of the signal, which results in a better time localization of the frequency content.

Window length constant, window shift changes:

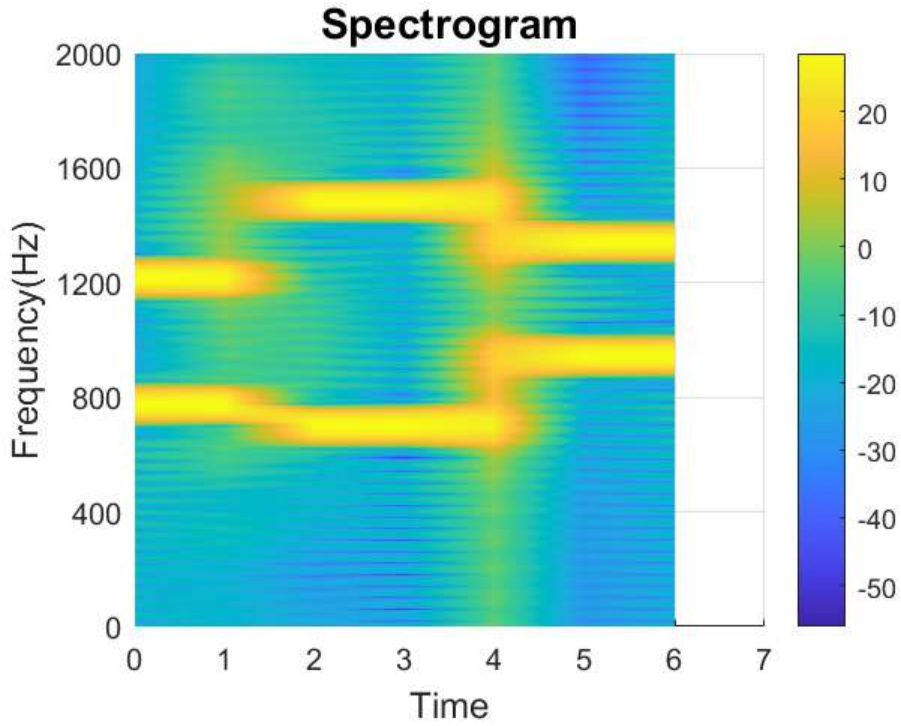
```
window_length = 100;
window_shift = 20;

figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'hamming',sampling_frequency_div2)
```



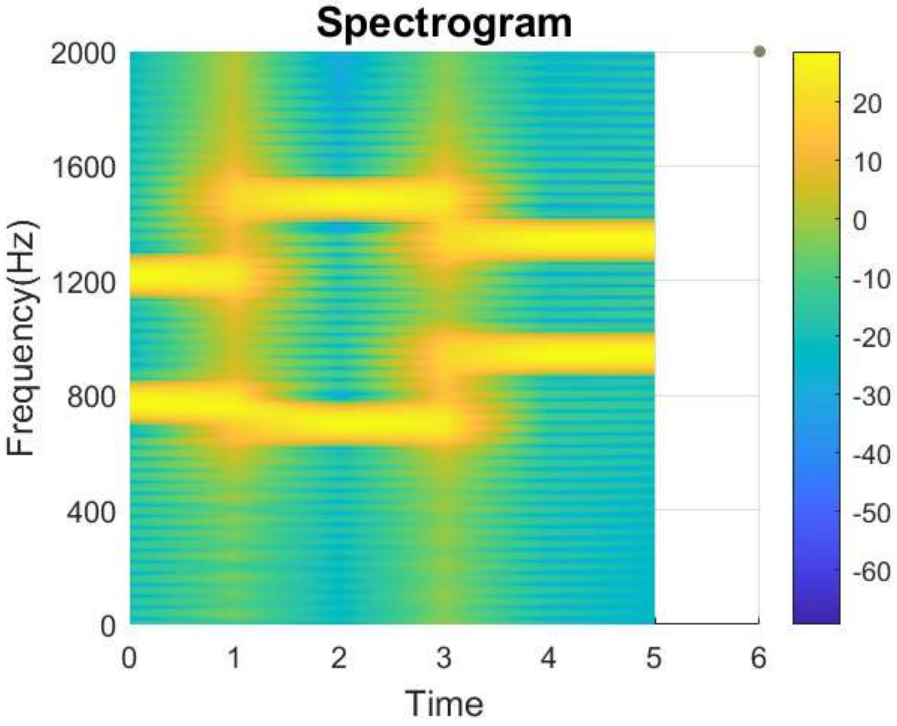

```
window_length = 100;
window_shift = 50;

figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'hamming',sampling_frequency_div2)
```




```
window_length = 100;
window_shift = 75;

figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'hamming',sampling_frequency_div2)
```




```
window_length = 100;
window_shift = 100;

figure
spectogram_plotter_generated_signals_vol2(summed_signal,window_length,window_shift,'hamming',sampling_frequency_div2)
```



Comments: Opposite to the previous case, as we increase window shift while keeping the window length constant, we observe that the time resolution increases but the frequency resolution decreases. This is due to the fact that as the larger windows overlap more and thus the frequency content is lost more.

ii.

The Nyquist Theorem for sampling states that:

In order to avoid aliasing, the sampling frequency Fs must be at least larger or equal to the twice largest frequency component in a signal, i.e., Fs>=fmax*2. Hence if our frequency is Fs the largest frequency we can have in our signal in order to avoid aliasing and to certainly reconstruct the signal is Fs/2. When the pairs are added up their common period becomes the smallest common multiple of these two values. This can be shown as Tnew = T1*T2/GCD(T1,T2), where GCD is the operator for the greatest common divider. Since T1 and T2 values for this example are not integer values, we can do same operation by finding the GCD of the frequencies. Then Tnew = 1/GCD(f1,f2) and the new common frequency will be Fcommon = 1/Tnew = GCD(f1,f2). Hence the number of samples we must take from the frequency domain will simply be N = Fs/(2*Fnew). If it is not an integer we should choose the next largest integer.

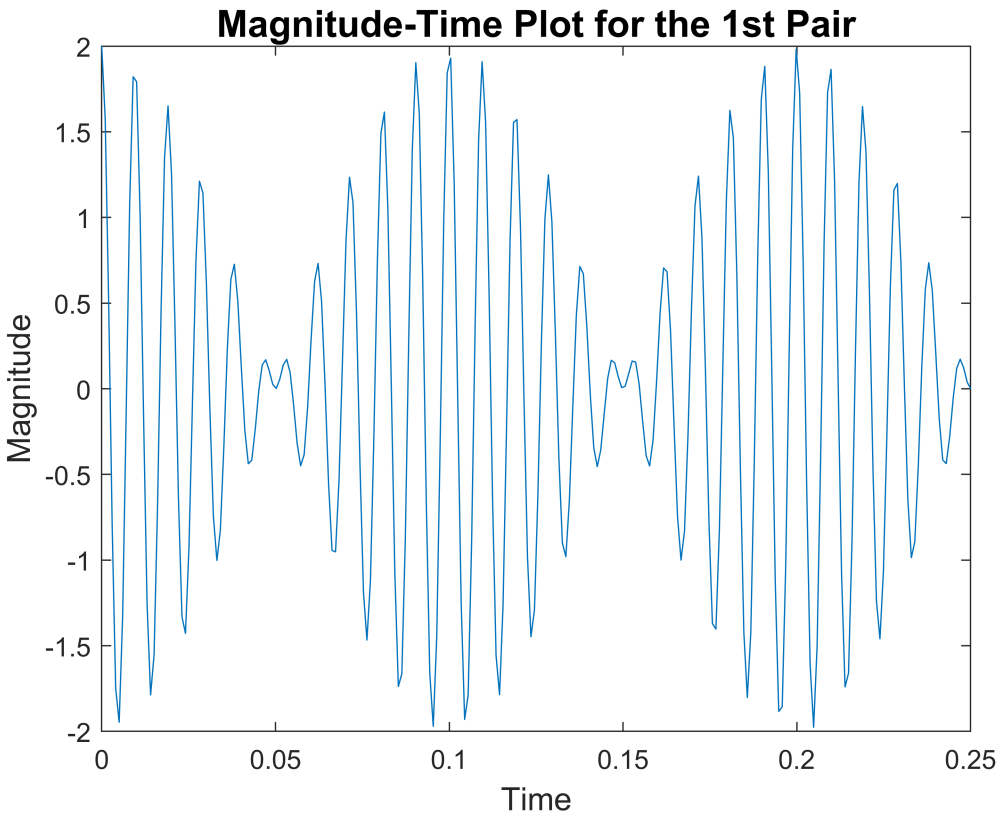
```
sampling_frequency_div_2 = 500;
amplitude_m = [1 1];
phase_m = [0 0];
total_duration = 0.25;

% First we generate function handles of summed sinusoidals using
% generator_summed_sinusoids
```

Pair 1

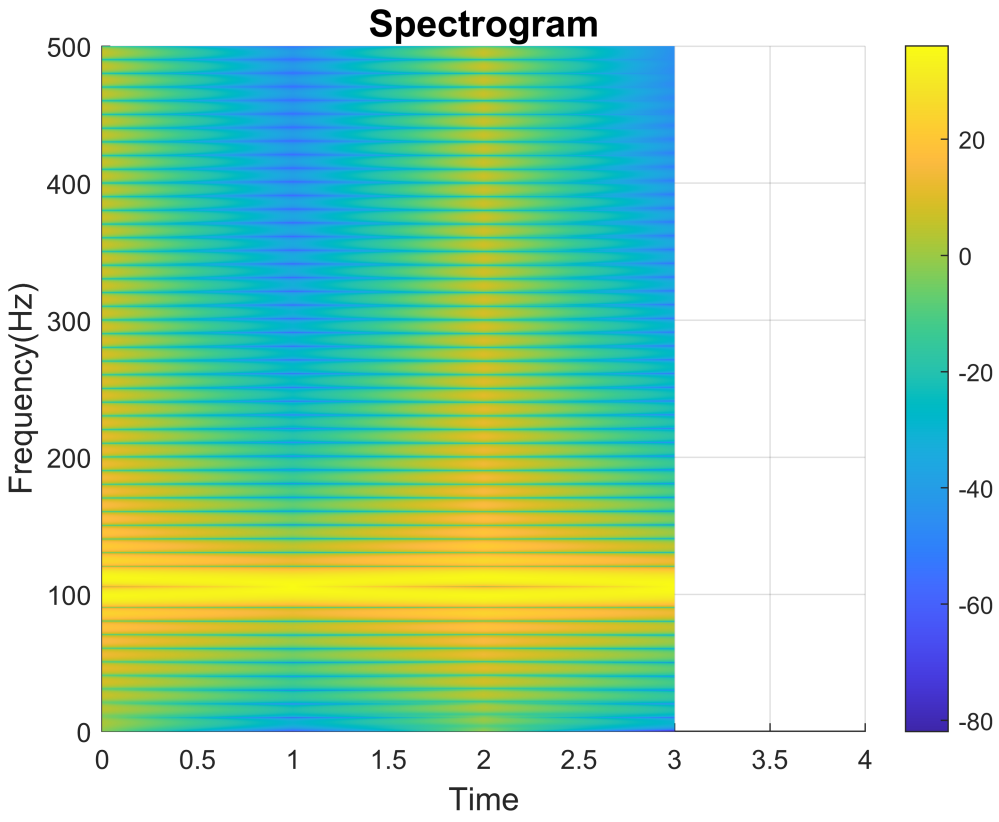
```
% Function handle 1: 100 Hz & 110 Hz;
frequency_m_1 = [100 110];
handle_1 = generate_summed_sinusoidals(2,amplitude_m,frequency_m_1,phase_m);

t = linspace(0,total_duration,2*sampling_frequency_div_2*total_duration)
```

```
t = 1×250
         0    0.0010    0.0020    0.0030    0.0040    0.0050    0.0060    0.0070 ···
```

```
pair_1 = handle_1(t);

figure
plot(t,pair_1);
xlabel('Time','Fontsize',12);
ylabel('Magnitude','Fontsize',12);
title('Magnitude-Time Plot for the 1st Pair','Fontsize',14);
xlim([0 total_duration]);
```




```
window_length = 100;
window_shift = 50;

figure
spectogram_plotter_generated_signals_vol2(pair_1,window_length,window_shift,'rect',sampling_fre
```




```
fcommon = gcd(frequency_m_1(1),frequency_m_1(2));
Tcommon = 1/fcommon;
num_of_samples_needed = sampling_frequency_div_2/fcommon;
disp(['The common period is:', num2str(Tcommon)]);
```

```
The common period is:0.1
```

```
disp(['The number of samples needed is:', num2str(num_of_samples_needed)]);
```

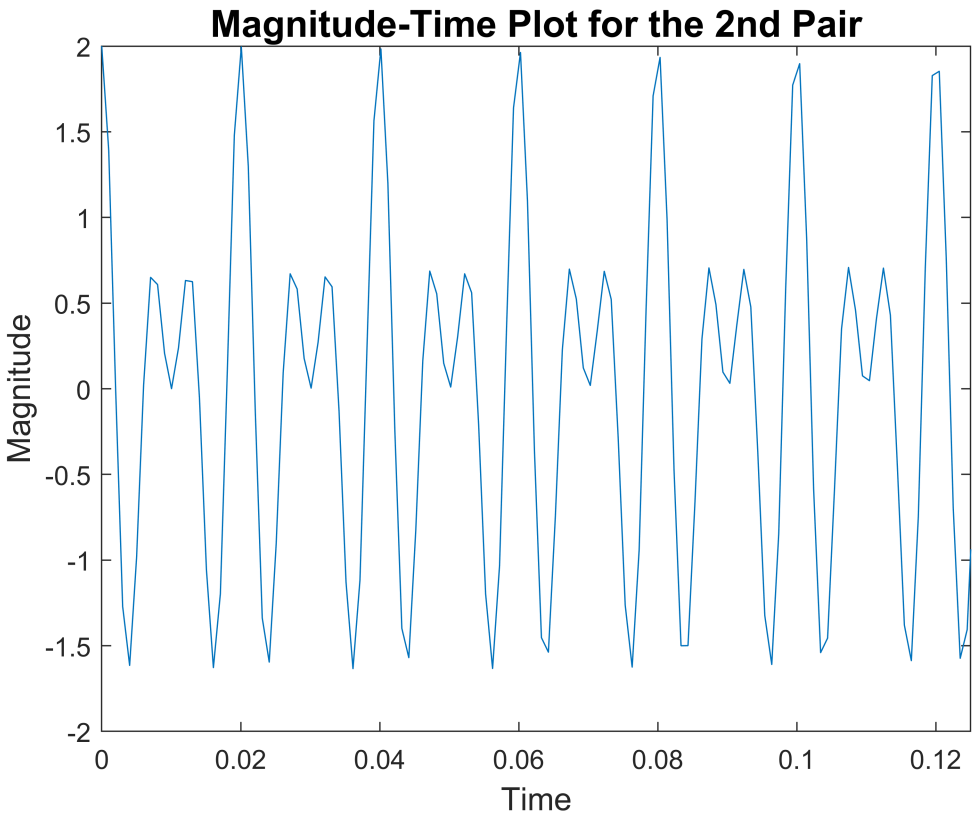
```
The number of samples needed is:50
```

We can also observe from the time domain signal how the new period is now 0.1 s.

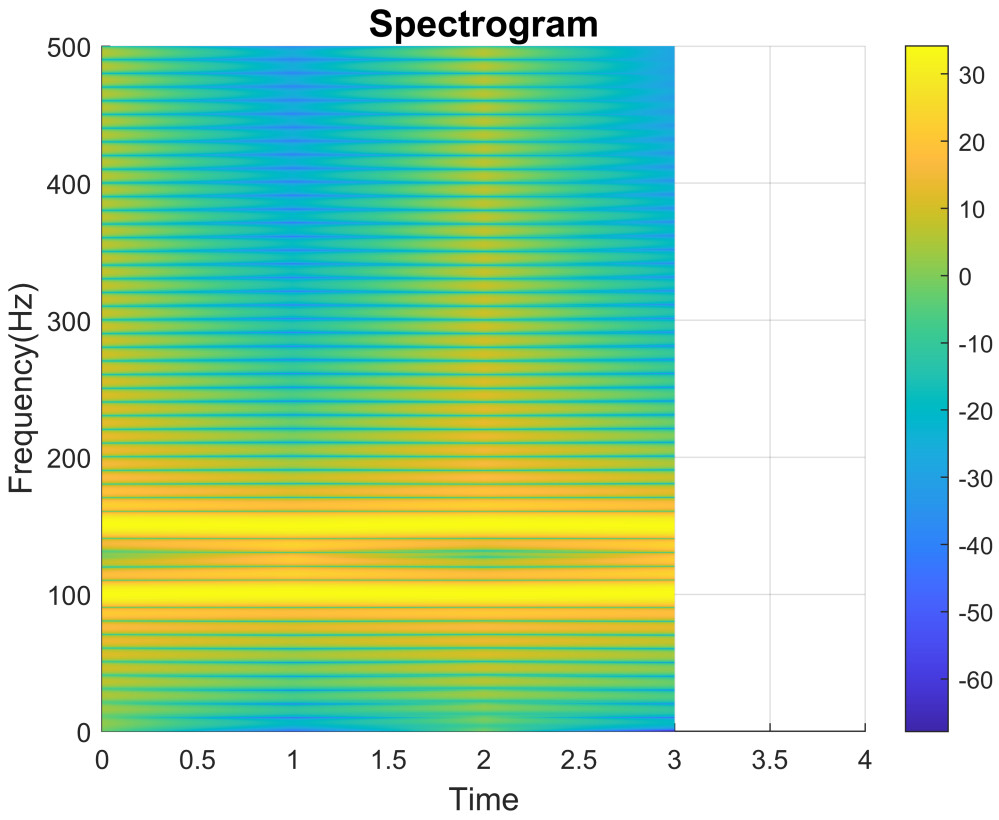
Pair 2

```
% Function handle 2: 100 Hz & 150 Hz;
frequency_m_2 = [100 150];
handle_2 = generate_summed_sinusoidals(2,amplitude_m,frequency_m_2,phase_m);

t = linspace(0,total_duration,2*sampling_frequency_div_2*total_duration);

pair_2 = handle_2(t);

figure
```

```
plot(t,pair_2);
xlabel('Time','Fontsize',12);
ylabel('Magnitude','Fontsize',12);
title('Magnitude-Time Plot for the 2nd Pair','Fontsize',14);
xlim([0 total_duration/2]);
```




```
figure
spectogram_plotter_generated_signals_vol2(pair_2,window_length,window_shift,...
```




```
    'rectwin',sampling_frequency_div_2)

fcommon = gcd(frequency_m_2(1),frequency_m_2(2));
Tcommon = 1/fcommon;
num_of_samples_needed = sampling_frequency_div_2/fcommon;
disp(['The common period is:', num2str(Tcommon)]);
```

```
The common period is:0.02
```

```
disp(['The number of samples needed is:', num2str(num_of_samples_needed)]);
```

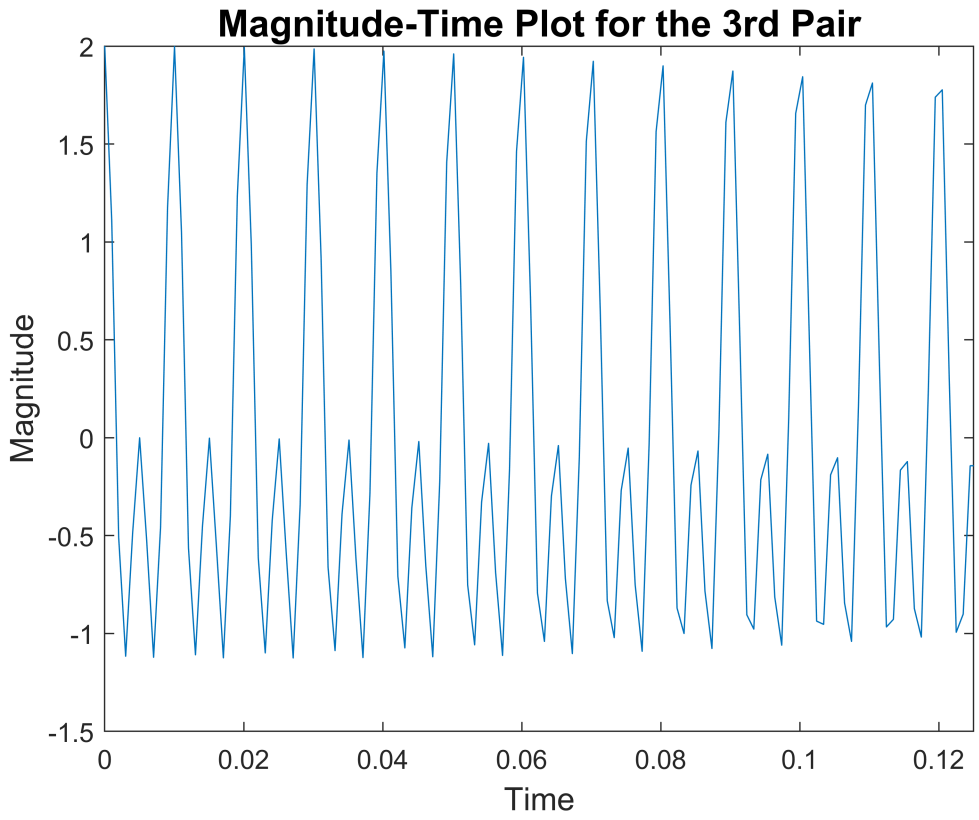
```
The number of samples needed is:10
```

Again, we can observe from the time plot that the new period is 0.02, just as we calculated using the formula.

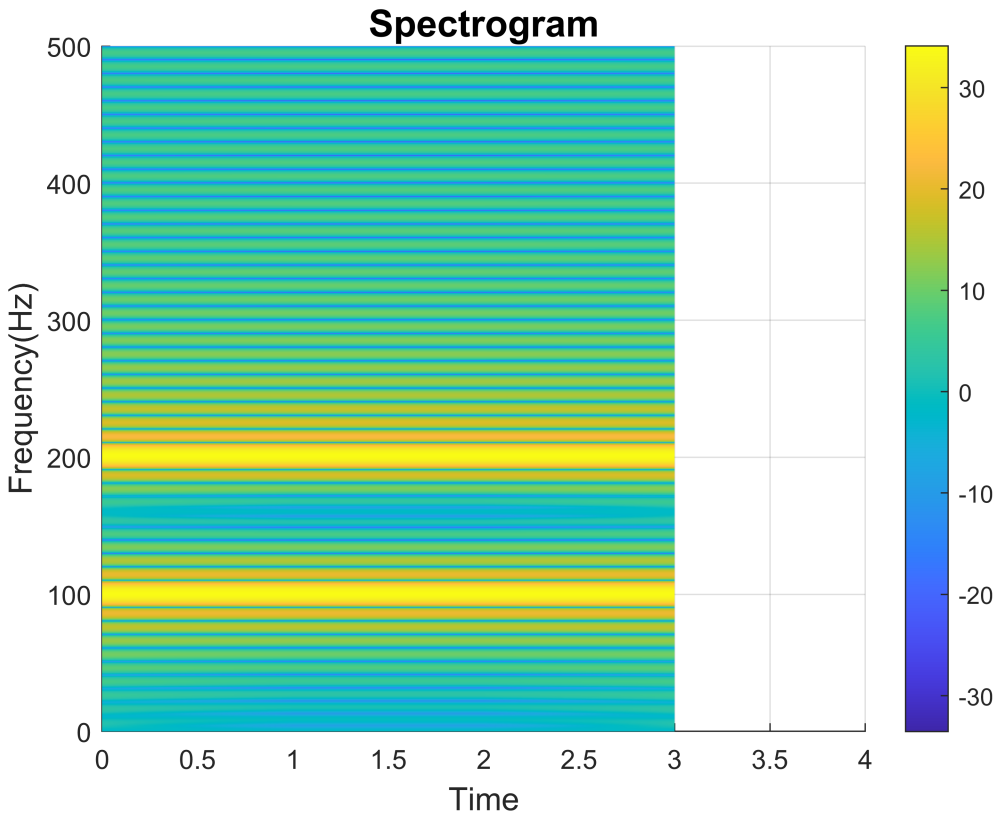
Pair 3

```
% Function handle 2: 100 Hz & 150 Hz;
frequency_m_3 = [100 200];
handle_3 = generate_summed_sinusoidals(2,amplitude_m,frequency_m_3,phase_m);

t = linspace(0,total_duration,2*sampling_frequency_div_2*total_duration);

pair_3 = handle_3(t);

figure
```

```
plot(t,pair_3);
xlabel('Time','Fontsize',12);
ylabel('Magnitude','Fontsize',12);
title('Magnitude-Time Plot for the 3rd Pair','Fontsize',14);
xlim([0 total_duration/2]);
```




```
figure
spectogram_plotter_generated_signals_vol2(pair_3,window_length,window_shift,...
```




```
    'rectwin',sampling_frequency_div_2)

fcommon = gcd(frequency_m_3(1),frequency_m_3(2));
Tcommon = 1/fcommon;
num_of_samples_needed = sampling_frequency_div_2/fcommon;
disp(['The common period is:', num2str(Tcommon)]);
```

```
The common period is:0.01
```

```
disp(['The number of samples needed is:', num2str(num_of_samples_needed)]);
```

```
The number of samples needed is:5
```

Again, we can observe from the time plot that the new period is 0.01, just as we calculated using the formula.

Since one of the frequencies is always 100Hz we can use this to analyze better. We see that as the greatest common divider between 100Hz and f2 increases, the common period is lower and he common frequency becomes higher. Thus we'll need less samples to analyze the signal.

iii.

We look at the colum frequencies and row frequencies to find the relation in each. It turns out that the ratios between both the column an row frequencies are not linear. For example, the ratio between 697 Hz and 770 Hz is different from the ratio between 770 Hz and 852 Hz. The distance between consecutive signals is also not the same. So there is also not a linearity with an offset.

The varying separation distances between frequencies makes it harder to choose a fixed window size for the STFT. Thus in some cases it becomes hard to effectively capture all DTMF frequencies. For example, if the window size is too small, we might miss lower-frequency components, and if it's too large, it may introduce spectral leakage and affect our identification accuracy. This effect can be seen more clearly in part 2. For instance the value of window length 50 we have chosen for this example succesfully captures the sepeartion between 2nd and 3rd pair while it does not do so well for the 1st pair.

iv.

We apply the STFT to the signal in question. The STFT allows us to observe how the frequency content of the signal changes over time. Of course the window size, shift and other parameters should be chosen accordingly. Different parameters may be tried to find the optimal values.

Then we analyze the resulting spectrogram produced by the STFT. Since DTMF signals have distinct frequency pairs associated with each button press, that change over time we first look for the time instances in which these changes appear. After identifying these we determine the frequency pairs in each time interval.

All that is left to do is to use the table in order to figure it which pairs correspond to which value.

Other alternatives might be:

- Anlayzing in time domain first to identify different singals, and after identifying those intervals taking the Fourier Transform.
- Goertzel Algoirthm is also one of the commonly used methods for decoding DTMF signals.
- Machine Learnig techniques are also a good alternative that might be useful.

Conclusion

In this report, a MATLAB-based graphical user interface (GUI) designed for the analysis of different types of signals was introduced. More specifically, this GUI displays the signal's frequency content over time, obtained through the Short Time Fourier Transform (STFT) enables to analyze it via visuals.

The system enables capturing and processing of audio data from a microphone or existing files. Furthermore, it includes a feature for generating different types of time-domain signals determined by user input. Parameter adjustments during data acquisition, generation, and spectrogram analysis, and tuning of parameters for the spectrogram plot such as window length, shift, and type are also available to the user. The GUI which is made up of different tabs is introduced and explained using the screenshots of each menu, and examples of the usages of these tabs. The functions used in each of the parts are also mentioned with a short explanation of their purpose. The spectrogram of the generated signal can be plotted as a surface plot for each data acquisition type if the user chooses to.

Throughout the report, the importance of parameters like window type, window length, and overlap in STFT analysis has been analyzed. In addition, Question 2 provided even more details on the effects of choosing the right window. Optimization of these parameters enhances the accuracy and reliability of the spectrogram representation, providing valuable insights into temporal and frequency characteristics. A general observation was that there always was a tradeoff between time and frequency resolutions.

The questions at the end of the report serve as important examples of the usage of STFT and provide even more insight on the analysis of frequency in time. Question one does this by showing the example of human speech analysis and how to determine the appropriate sampling rate. Whereas Question 2 did this by introducing the example of the telecommunication method DTMF.

MATLAB CODES

```
function captureAudio(samplingRate, recordingDuration)

    outputFileName = 'LastSaved.wav';

    % Check if the input sampling rate is valid
    if samplingRate <= 0
        error('Sampling rate must be greater than 0.');
    end

    % Check if the recording duration is valid
    if recordingDuration <= 0
        error('Recording duration must be greater than 0.');
    end

    % Create an audio recorder object
    recorder = audiorecorder(samplingRate, 16, 1);

    % Record audio for the specified duration
    disp('Start recording...');
    recordblocking(recorder, recordingDuration);
    disp('Recording complete.');

    % Get the recorded audio data
    audioData = getaudiodata(recorder);

    audiowrite(outputFileName, audioData, samplingRate);
end


function [selectedChannel,samplingRate]=openAudio(file) % wav or mp3 file chosen
by the user.

    [audioData,samplingRate] = audioread(file);

    % Check the number of channels
    numChannels = size(audioData, 2);

    if numChannels > 1
        % If there are multiple channels, choose one channel (e.g., the first
channel)
        selectedChannel = audioData(:, 1);
        disp('Selected only the first channel.');
    else
        % If there is only one channel, use the original data
        selectedChannel = audioData;
        disp('The file has only one channel.');
    end

    disp('Playing audio...');
    player = audioplayer(audioData, samplingRate);
    playblocking(player);
    disp('Playback complete.');

end
function signal = generate_sinusoidal(amplitude, frequency, phase)

    signal = @(t) amplitude * cos(2 * pi * frequency * t + phase*pi/180);
```

```
end
function signal = generate_summed_sinusoidals(M, Am, f, p_vector)

    % Initialize the signal
    y = @(t) Am(1) * cos(2 * pi * f(1) * t + p_vector(1)*pi/180) ;

    % Generate the signal using a for loop
    for i = 2:M
        x = @(t) Am(i) * cos(2 * pi * f(i) * t + p_vector(i));
        y = @(t) y(t) + x(t);
    end;

    signal = y;

end


function signal = generate_windowed_sinusoidal(amplitude, frequency, phase,
window_type, start_time, interval_length)

    sinusoidal_part = @(t) amplitude * cos(2 * pi * frequency * t + phase*pi/180);

    a = start_time;
    b = interval_length + start_time;
    alpha = 5;


    if window_type == 'rectwin'
        window_function = @(t) (t >= a & t <= b);

    elseif window_type == 'hamming'
        window_function = @(t) (0.54 - 0.46 * cos(2 * pi * (t - a) / (b - a))) .*
((t >= a) & (t <= b));

    elseif window_type == 'tukeywin'
        window_function = @(t) ((t >= a) & (t <= b)) .* ...
        (0.5 * (1 + cos(pi * ((t - a) / (alpha * (b - a)) - 1))) .* (t >= a) & (t
< a + alpha * (b - a)) + ...
        (t >= a + alpha * (b - a)) & (t <= b));
    end


   signal= @(t) sinusoidal_part(t) .* window_function(t); %This is for limiting
the interval,  it'll be detremined by start_time and interval length

end


function
plotter_for_generated_signals_GUI(axes1,axes2,signal_handle,f,winLength,winShift,w
indow_type,duration,sampling_frequency)


        sampling_frequency_div_2 = sampling_frequency/2;
        t = linspace(0,duration,2*sampling_frequency_div_2*duration);

        y = signal_handle(t);
```

```
        spectogram_plotter_generated_signals_GUI(axes1,y, winLength,
winShift,window_type,sampling_frequency_div_2);

        t = linspace(0,duration,2*sampling_frequency*duration);
        y = signal_handle(t);
        plot(axes2,t,y);
        xlabel(axes2,'Time','Fontsize',12);
        ylabel(axes2,'Magnitude','Fontsize',12);
        title(axes2,'Magnitude-Time Plot for 2T','Fontsize',14);
        xlim(axes2,[0 2/f]);

end


function
spectogram_plotter_generated_signals_GUI(axes,sound_signal,window_length,window_sh
ift,window_type,sampling_frequency)

    % Transpose is taken if the signal is a column vector , instead of a
    % row vector

    if(size(sound_signal,1) > 5)
        sound_signal = sound_signal';
    end

    win = 0;    % initial blank window

    %************************************************
    % Controlling if we satisfy the length constraints:

    if( length(sound_signal) < window_length )

    disp(' Error: The signal is too short for this window length. Submit a new
signal or choose a new window length.');

    end

    if( window_shift > window_length )
    disp(' Error: Enter a valid window length that is longer than the window
shift.');
    end

    %************************************************


    % Assigning the window type:
    if( strcmp(window_type,'rectwin')) %#ok<*STCMP>
        win=ones(1,window_length);
    elseif(strcmp(window_type,'hamming'))
        win=hamming(window_length)';
    elseif(strcmp(window_type,'tukeywin'))
        win=tukeywin(window_length)';
    else
        win=ones(1,window_length);
    end

    %************************************************

    begin_point = 1;    % Initial point for thewindow
```

```
    vertical_line_no=1; % Counter for the vertical lines

    %Calculating the STFT

    while( begin_point + window_length -1 <= length(sound_signal) )

        STFT(vertical_line_no,:) = abs(fft(sound_signal( begin_point :
begin_point+window_length-1 ).*win,window_length*100));
        begin_point = begin_point + window_shift;
        vertical_line_no = vertical_line_no+1 ;

    end


    time_values = 0 : size(STFT,1)-1; %Time values to be plotted
    frequency_values = linspace(0,sampling_frequency,window_length*50); %The plot
will look different depending on the sampling frequency of the sound


    STFT_in_DB=(20*log10(STFT))'; % Coverting the magnitudes to decibels for
better analysis

    %************************************************

    % Plotting the STFT
    % Axes is added in order to plot it in a determined plot in App
    % Designer

    surf(axes,time_values,frequency_values,STFT_in_DB(1:window_length*50,:));
    shading(axes,"interp");
    view(axes,2); %The view should be from above since our plot is 2D in view
    xlabel(axes,'Time','Fontsize',12);
    ylabel(axes,'Frequency(Hz)','Fontsize',12);
    title(axes,'Spectrogram','Fontsize',14);
    colorbar(axes);
    ylim(axes,[0 sampling_frequency]);
    xlim(axes,[0 size(STFT,1)]);
    yticks(axes,linspace(0, sampling_frequency, 6));

end
```

```
classdef SpectogramAppVersion4_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        TabGroup                        matlab.ui.container.TabGroup
        MainPageTab                     matlab.ui.container.Tab
        SelectDataAcquisitionMethodLabel  matlab.ui.control.Label
        GenerateSignalButton            matlab.ui.control.Button
        UploadAudioFileButton           matlab.ui.control.Button
        RecordAudioButton               matlab.ui.control.Button
        RecordTab                       matlab.ui.container.Tab
        Lamp                            matlab.ui.control.Lamp
        WindowTypeDropDown_2            matlab.ui.control.DropDown
        WindowTypeDropDown_2Label       matlab.ui.control.Label
        WindowShiftEditField            matlab.ui.control.NumericEditField
        WindowShiftEditFieldLabel       matlab.ui.control.Label
        WindowLengthEditField           matlab.ui.control.NumericEditField
        WindowLengthEditFieldLabel      matlab.ui.control.Label
        PlotSpectogramButton            matlab.ui.control.Button
        PlayButton                      matlab.ui.control.Button
        RecordButton                    matlab.ui.control.Button
        RecordingTimeEditField          matlab.ui.control.NumericEditField
        RecordingTimeEditFieldLabel     matlab.ui.control.Label
        SamplingRateEditField           matlab.ui.control.NumericEditField
        SamplingRateEditFieldLabel      matlab.ui.control.Label
        UIAxes                          matlab.ui.control.UIAxes
        SelectFileTab                   matlab.ui.container.Tab
        WindowTypeDropDown_3            matlab.ui.control.DropDown
        WindowTypeDropDown_3Label       matlab.ui.control.Label
        WindowShiftEditField_2          matlab.ui.control.NumericEditField
        WindowShiftEditField_2Label     matlab.ui.control.Label
        WindowLengthEditField_2         matlab.ui.control.NumericEditField
        WindowLengthEditField_2Label    matlab.ui.control.Label
        PlotSpectogramButton_2          matlab.ui.control.Button
        PlayButton_2                    matlab.ui.control.Button
        UploadaFileButton               matlab.ui.control.Button
        UIAxes_2                        matlab.ui.control.UIAxes
        GenerateTab                     matlab.ui.container.Tab
        SamplingRateEditField_2         matlab.ui.control.NumericEditField
        SamplingRateEditField_2Label    matlab.ui.control.Label
        DurationEditField_2             matlab.ui.control.NumericEditField
        DurationLabel_2                 matlab.ui.control.Label
        DurationEditField               matlab.ui.control.NumericEditField
        DurationLabel                   matlab.ui.control.Label
        ComponentsLabel                 matlab.ui.control.Label
        WindowTypeLabel                 matlab.ui.control.Label
        PhaseLabel_3                    matlab.ui.control.Label
        PhaseEditField_3                matlab.ui.control.NumericEditField
        LengthLabel                     matlab.ui.control.Label
        LengthEditField                 matlab.ui.control.NumericEditField
        StartTimeLabel                  matlab.ui.control.Label
        FrequencyLabel_3                matlab.ui.control.Label
        AmplitudeLabel_3                matlab.ui.control.Label
        StartTimeEditField              matlab.ui.control.NumericEditField
        FrequencyEditField_3            matlab.ui.control.NumericEditField
        AmplitudeEditField_3            matlab.ui.control.NumericEditField
        PhaseLabel_2                    matlab.ui.control.Label
        FrequencyLabel_2                matlab.ui.control.Label
```

```
        AmplitudeLabel_2              matlab.ui.control.Label
        PhaseEditField_2              matlab.ui.control.NumericEditField
        FrequencyEditField_2          matlab.ui.control.NumericEditField
        AmplitudeEditField_2          matlab.ui.control.NumericEditField
        WindowTypeDropDown_4          matlab.ui.control.DropDown
        WindowTypeDropDown_4Label     matlab.ui.control.Label
        WindowShiftEditField_3        matlab.ui.control.NumericEditField
        WindowShiftEditField_3Label   matlab.ui.control.Label
        WindowLengthEditField_3       matlab.ui.control.NumericEditField
        WindowLengthEditField_3Label  matlab.ui.control.Label
        SubmitButton_2                matlab.ui.control.Button
        PhaseLabel                    matlab.ui.control.Label
        FrequencyLabel                matlab.ui.control.Label
        AmplitudeLabel                matlab.ui.control.Label
        PlotSpectogramButton_3        matlab.ui.control.Button
        PhaseEditField                matlab.ui.control.NumericEditField
        SubmitButton                  matlab.ui.control.Button
        FrequencyEditField            matlab.ui.control.NumericEditField
        AmplitudeEditField            matlab.ui.control.NumericEditField
        ComponetsEditField            matlab.ui.control.EditField
        WindowTypeDropDown            matlab.ui.control.DropDown
        SignalwithmultiplecomponentsCheckBox  matlab.ui.control.CheckBox
        WindowedsinusoidalCheckBox    matlab.ui.control.CheckBox
        SinusoidalCheckBox            matlab.ui.control.CheckBox
        UIAxes2_2                     matlab.ui.control.UIAxes
        UIAxes2                       matlab.ui.control.UIAxes
        ContextMenu                   matlab.ui.container.ContextMenu
        Menu                          matlab.ui.container.Menu
        Menu_2                        matlab.ui.container.Menu
        Menu2_2                       matlab.ui.container.Menu
        Menu_3                        matlab.ui.container.Menu
        Menu2                         matlab.ui.container.Menu
    end


    properties (Access = private)
        CurrentFilePath
        Switch
%         Amplitude1
%         Frequency1
%         Phase1
%         Amplitude2
%         Frequency2
%         Phase2
%         StartTime1
%         Lenght1
        MyDropdown
    end

    properties (Access = public)
        N
        FrequencyMatrix
        AmplitudeMatrix
        PhaseMatrix
    end
```

```
% Callbacks that handle component events
methods (Access = private)

    % Button pushed function: RecordAudioButton
    function RecordAudioButtonPushed(app, event)
        app.TabGroup.SelectedTab = app.RecordTab;
    end

    % Button pushed function: UploadAudioFileButton
    function UploadAudioFileButtonPushed(app, event)
        app.TabGroup.SelectedTab = app.SelectFileTab;
    end

    % Button pushed function: GenerateSignalButton
    function GenerateSignalButtonPushed(app, event)

        % Hiding the components at the beginning. We will show them
        % only if the corresponding function type is chosen.



        app.TabGroup.SelectedTab = app.GenerateTab;

        % For summed sinusoidals:

        app.AmplitudeEditField.Visible = 'off';
        app.AmplitudeLabel.Visible = 'off';
        app.FrequencyLabel.Visible = 'off';
        app.FrequencyEditField.Visible = 'off';
        app.PhaseLabel.Visible = 'off';
        app.PhaseEditField.Visible = 'off';
        app.SubmitButton.Visible = 'off';
        app.ComponentsLabel.Visible = 'off';
        app.ComponetsEditField.Visible = 'off';
        app.SubmitButton_2.Visible = 'off';
        app.DurationEditField_2.Visible = 'off';
        app.DurationLabel_2.Visible = 'off';


        % For sinusoidals:

        app.AmplitudeEditField_2.Visible = 'off';
        app.AmplitudeLabel_2.Visible = 'off';
        app.FrequencyLabel_2.Visible = 'off';
        app.FrequencyEditField_2.Visible = 'off';
        app.PhaseLabel_2.Visible = 'off';
        app.PhaseEditField_2.Visible = 'off';
        app.DurationEditField.Visible = 'off';
        app.DurationLabel.Visible = 'off';

        % For windowed sinusoidals:

        app.AmplitudeEditField_3.Visible = 'off';
        app.AmplitudeLabel_3.Visible = 'off';
        app.FrequencyLabel_3.Visible = 'off';
        app.FrequencyEditField_3.Visible = 'off';
        app.PhaseLabel_3.Visible = 'off';
        app.PhaseEditField_3.Visible = 'off';
```

```
        app.StartTimeLabel.Visible = 'off';
        app.StartTimeEditField.Visible = 'off';
        app.LengthEditField.Visible = 'off';
        app.LengthLabel.Visible = 'off';
        app.WindowTypeDropDown.Visible = 'off';
        app.WindowTypeLabel.Visible = 'off';

    end

    % Button pushed function: UploadaFileButton
    function UploadaFileButtonPushed(app, event)
        % Button pushed function: UploadFileButton
        % Open a file dialog for the user to select a file
        [fileName, filePath] = uigetfile({'*.wav'; '*.mp3'}, 'Select a file');

        % Check if the user clicked 'Cancel'
        if isequal(fileName, 0)
            disp('User canceled file selection.');
        else
            % Construct the full file path
            fullFilePath = fullfile(filePath, fileName);
            app.CurrentFilePath=fullFilePath;

        end
    end

    % Button pushed function: RecordButton
    function RecordButtonPushed(app, event)
        samplingRate=app.SamplingRateEditField.Value;
        recordingDuration=app.RecordingTimeEditField.Value;
        app.Lamp.Color = 'r';
        captureAudio(samplingRate,recordingDuration);
        app.Lamp.Color = 'g';
    end

    % Button pushed function: PlayButton
    function PlayButtonPushed(app, event)
        openAudio('LastSaved.wav');
    end

    % Button pushed function: PlayButton_2
    function PlayButton_2Pushed(app, event)

            openAudio(app.CurrentFilePath);

    end

    % Callback function
    function SinusoidalSwitch_2ValueChanged(app, event)
        %switchState = app.Switch.Value;
        %app.AmplitudeEditField.Enable = switchState;
    end

    % Callback function
    function SinusoidalCheckBoxValueChanged(app, event)



    end
```

```
% Callback function
function WindowedsinusoidalCheckBoxValueChanged(app, event)

end

% Callback function
function SignalwithmultiplecomponentsCheckBoxValueChanged(app, event)
    app.AmplitudeEditField_3.Enable = 'off';
    value = app.SignalwithmultiplecomponentsCheckBox.Value;
    app.AmplitudeEditField_3.Enable = value;

end

% Callback function
function ButtonPushed(app, event)
    app.AmplitudeEditField_3.Enable = 'on';
end

% Callback function
function AmplitudeEditField_3ValueChanged(app, event)
     app.AmplitudeEditField_3.Enable="off"


end

% Size changed function: GenerateTab
function GenerateTabSizeChanged(app, event)
    position = app.GenerateTab.Position;

end

% Value changed function: WindowedsinusoidalCheckBox
function WindowedsinusoidalCheckBoxValueChanged2(app, event)

    if(app.WindowedsinusoidalCheckBox.Value == 1)
    % Turning the visibility of these boxes on
        app.AmplitudeEditField_3.Visible = 'on';
        app.AmplitudeLabel_3.Visible = 'on';
        app.FrequencyLabel_3.Visible = 'on';
        app.FrequencyEditField_3.Visible = 'on';
        app.PhaseLabel_3.Visible = 'on';
        app.PhaseEditField_3.Visible = 'on';
        app.StartTimeLabel.Visible = 'on';
        app.StartTimeEditField.Visible = 'on';
        app.LengthEditField.Visible = 'on';
        app.LengthLabel.Visible = 'on';
        app.WindowTypeDropDown.Visible = 'on';
        app.WindowTypeLabel.Visible = 'on';
    else
        app.AmplitudeEditField_3.Visible = 'off';
        app.AmplitudeLabel_3.Visible = 'off';
        app.FrequencyLabel_3.Visible = 'off';
        app.FrequencyEditField_3.Visible = 'off';
        app.PhaseLabel_3.Visible = 'off';
        app.PhaseEditField_3.Visible = 'off';
        app.StartTimeLabel.Visible = 'off';
        app.StartTimeEditField.Visible = 'off';
        app.LengthEditField.Visible = 'off';
```

```
            app.LengthLabel.Visible = 'off';
            app.WindowTypeDropDown.Visible = 'off';
            app.WindowTypeLabel.Visible = 'off';

        end
    end

    % Value changed function: SinusoidalCheckBox
    function SinusoidalCheckBoxValueChanged2(app, event)

        if(app.SinusoidalCheckBox.Value == 1)
            app.AmplitudeEditField_2.Visible = 'on';
            app.AmplitudeLabel_2.Visible = 'on';
            app.FrequencyLabel_2.Visible = 'on';
            app.FrequencyEditField_2.Visible = 'on';
            app.PhaseLabel_2.Visible = 'on';
            app.PhaseEditField_2.Visible = 'on';
            app.DurationEditField.Visible = 'on'
            app.DurationLabel.Visible = 'on';
        else
            app.AmplitudeEditField_2.Visible = 'off';
            app.AmplitudeLabel_2.Visible = 'off';
            app.FrequencyLabel_2.Visible = 'off';
            app.FrequencyEditField_2.Visible = 'off';
            app.PhaseLabel_2.Visible = 'off';
            app.PhaseEditField_2.Visible = 'off';
            app.DurationEditField.Visible = 'off';
            app.DurationLabel.Visible = 'off';
        end


    end

    % Button pushed function: PlotSpectogramButton
    function PlotSpectogramButtonPushed(app, event)

    [audioData,samplingRate] = audioread("LastSaved.wav");
    decimated_audio = decimate(audioData, 10);

    selectedWindow = app.WindowTypeDropDown.Value;

    switch selectedWindow
            case 'Rectangular'
                WindowType = 'rectwin';
            case 'Tukey'
                WindowType = 'tukeywin';
            case 'Hamming'
                WindowType = 'hamming';
            otherwise
                WindowType = 'rectwin';
        end


spectogram_plotter_sound_GUI(app.UIAxes,decimated_audio,app.WindowLengthEditField.
Value,app.WindowShiftEditField.Value,WindowType,samplingRate/2) % Takes ~10
seconds for a 10 second signal

    end
```

```
        % Button pushed function: PlotSpectogramButton_2
        function PlotSpectogramButton_2Pushed(app, event)
        [audioData,samplingRate] = audioread(app.CurrentFilePath);
        decimated_audio = decimate(audioData, 10);

        selectedWindow = app.WindowTypeDropDown_3.Value;

        switch selectedWindow
                case 'Rectangular'
                    WindowType = 'rectwin';
                case 'Tukey'
                    WindowType = 'tukeywin';
                case 'Hamming'
                    WindowType = 'hamming';
                otherwise
                    WindowType = 'rectwin';
            end


spectogram_plotter_sound_GUI(app.UIAxes_2,decimated_audio,app.WindowLengthEditFiel
d_2.Value,app.WindowShiftEditField_2.Value,WindowType,samplingRate/2) % Takes ~10
seconds for a 10 second signal

        end

        % Value changed function: SignalwithmultiplecomponentsCheckBox
        function SignalwithmultiplecomponentsCheckBoxValueChanged2(app, event)
                if app.SignalwithmultiplecomponentsCheckBox.Value==1
                    app.AmplitudeEditField.Visible = 'on';
                    app.AmplitudeLabel.Visible = 'on';
                    app.FrequencyLabel.Visible = 'on';
                    app.FrequencyEditField.Visible = 'on';
                    app.PhaseLabel.Visible = 'on';
                    app.PhaseEditField.Visible = 'on';
                    app.SubmitButton.Visible = 'on';
                    app.ComponentsLabel.Visible = 'on';
                    app.ComponetsEditField.Visible = 'on';
                    app.SubmitButton_2.Visible = 'on';
                    app.DurationEditField_2.Visible = 'on';
                    app.DurationLabel_2.Visible = 'on';
                else
                    app.AmplitudeEditField.Visible = 'off';
                    app.AmplitudeLabel.Visible = 'off';
                    app.FrequencyLabel.Visible = 'off';
                    app.FrequencyEditField.Visible = 'off';
                    app.PhaseLabel.Visible = 'off';
                    app.PhaseEditField.Visible = 'off';
                    app.SubmitButton.Visible = 'off';
                    app.ComponentsLabel.Visible = 'off';
                    app.ComponetsEditField.Visible = 'off';
                    app.SubmitButton_2.Visible = 'off';
                    app.DurationEditField_2.Visible = 'off';
                    app.DurationLabel_2.Visible = 'off';
                end
        end

        % Button pushed function: SubmitButton
        function SubmitButtonPushed(app, event)
            index = size(app.FrequencyMatrix,1)-app.N+1;
```

```
            if(index<size(app.FrequencyMatrix,1)+1)
                index
                app.FrequencyMatrix(index) = app.FrequencyEditField.Value;
                app.AmplitudeMatrix(index) = app.AmplitudeEditField.Value;
                app.PhaseMatrix(index) = app.PhaseEditField.Value;
                app.N = app.N-1;
                app.FrequencyEditField.Value = 0;
                app.AmplitudeEditField.Value = 0;
                app.PhaseEditField.Value = 0;
                app.FrequencyMatrix
            end

        end

        % Button pushed function: SubmitButton_2
        function SubmitButton_2Pushed(app, event)
            app.N = str2num(app.ComponetsEditField.Value);
            app.AmplitudeMatrix = zeros(app.N,1);
            app.FrequencyMatrix = zeros(app.N,1);
            app.PhaseMatrix = zeros(app.N,1);
        end

        % Button pushed function: PlotSpectogramButton_3
        function PlotSpectogramButton_3Pushed(app, event)

        selectedWindow = app.WindowTypeDropDown_4.Value;

        switch selectedWindow
                case 'Rectangular'
                    WindowType = 'rectwin';
                case 'Tukey'
                    WindowType = 'tukeywin';
                case 'Hamming'
                    WindowType = 'hamming';
                otherwise
                    WindowType = 'rectwin';
        end

        if(app.SignalwithmultiplecomponentsCheckBox.Value == 1)
            frequency = app.FrequencyMatrix;
            function_handle =
generate_summed_sinusoidals(size(app.FrequencyMatrix,1),app.AmplitudeMatrix,app.Fr
equencyMatrix,app.PhaseMatrix);
            duration = app.DurationEditField_2.Value;
        end
        if(app.SinusoidalCheckBox.Value == 1)
            amplitude = app.AmplitudeEditField_2.Value;
            phase = app.PhaseEditField_2.Value;
            frequency = app.FrequencyEditField_2.Value;
            function_handle = generate_sinusoidal(amplitude,frequency,phase);
            duration = app.DurationEditField.Value;
        end

        if(app.WindowedsinusoidalCheckBox.Value == 1)
            amplitude = app.AmplitudeEditField_3.Value;
            phase = app.PhaseEditField_3.Value;
            frequency = app.FrequencyEditField_3.Value;
            start_time = app.StartTimeEditField.Value;
            interval_length = app.LengthEditField.Value;
```

```
            duration = interval_length;

            selectedWindowGenerator = app.WindowTypeDropDown.Value;
            switch selectedWindowGenerator
                case 'Rectangular'
                    window_type__for_generator = 'rectwin';
                case 'Tukey'
                    WindowType = 'tukeywin';
                case 'Hamming'
                    window_type__for_generator = 'hamming';
                otherwise
                    window_type__for_generator = 'rectwin';
             end

            function_handle =
generate_windowed_sinusoidal(amplitude,frequency,phase,window_type__for_generator,
start_time,interval_length);
         end

        frequency = min(frequency); %we'll need this for plotting 2 periods of the
signals


plotter_for_generated_signals_GUI(app.UIAxes2,app.UIAxes2_2,function_handle,freque
ncy,app.WindowLengthEditField_3.Value,app.WindowShiftEditField_3.Value,WindowType,
duration,app.SamplingRateEditField_2.Value);

        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';

            % Create TabGroup
            app.TabGroup = uitabgroup(app.UIFigure);
            app.TabGroup.Position = [1 0 640 481];

            % Create MainPageTab
            app.MainPageTab = uitab(app.TabGroup);
            app.MainPageTab.Title = 'Main Page';

            % Create RecordAudioButton
            app.RecordAudioButton = uibutton(app.MainPageTab, 'push');
            app.RecordAudioButton.ButtonPushedFcn = createCallbackFcn(app,
@RecordAudioButtonPushed, true);
            app.RecordAudioButton.FontName = 'Bahnschrift';
            app.RecordAudioButton.Position = [270 294 100 23];
            app.RecordAudioButton.Text = 'Record Audio';

            % Create UploadAudioFileButton
            app.UploadAudioFileButton = uibutton(app.MainPageTab, 'push');
```

```
            app.UploadAudioFileButton.ButtonPushedFcn = createCallbackFcn(app,
@UploadAudioFileButtonPushed, true);
            app.UploadAudioFileButton.FontName = 'Bahnschrift';
            app.UploadAudioFileButton.Position = [266 244 109 23];
            app.UploadAudioFileButton.Text = 'Upload Audio File';

            % Create GenerateSignalButton
            app.GenerateSignalButton = uibutton(app.MainPageTab, 'push');
            app.GenerateSignalButton.ButtonPushedFcn = createCallbackFcn(app,
@GenerateSignalButtonPushed, true);
            app.GenerateSignalButton.FontName = 'Bahnschrift';
            app.GenerateSignalButton.Position = [270 196 101 23];
            app.GenerateSignalButton.Text = 'Generate Signal';

            % Create SelectDataAcquisitionMethodLabel
            app.SelectDataAcquisitionMethodLabel = uilabel(app.MainPageTab);
            app.SelectDataAcquisitionMethodLabel.HorizontalAlignment = 'center';
            app.SelectDataAcquisitionMethodLabel.FontName = 'Bahnschrift';
            app.SelectDataAcquisitionMethodLabel.FontWeight = 'bold';
            app.SelectDataAcquisitionMethodLabel.Position = [234 346 172 22];
            app.SelectDataAcquisitionMethodLabel.Text = 'Select Data Acquisition
Method';

            % Create RecordTab
            app.RecordTab = uitab(app.TabGroup);
            app.RecordTab.Title = 'Record';

            % Create UIAxes
            app.UIAxes = uiaxes(app.RecordTab);
            title(app.UIAxes, 'Title')
            xlabel(app.UIAxes, 'X')
            ylabel(app.UIAxes, 'Y')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Position = [288 60 300 185];

            % Create SamplingRateEditFieldLabel
            app.SamplingRateEditFieldLabel = uilabel(app.RecordTab);
            app.SamplingRateEditFieldLabel.HorizontalAlignment = 'right';
            app.SamplingRateEditFieldLabel.FontName = 'Bahnschrift';
            app.SamplingRateEditFieldLabel.Position = [226 376 83 22];
            app.SamplingRateEditFieldLabel.Text = 'Sampling Rate';

            % Create SamplingRateEditField
            app.SamplingRateEditField = uieditfield(app.RecordTab, 'numeric');
            app.SamplingRateEditField.FontName = 'Bahnschrift';
            app.SamplingRateEditField.Position = [324 376 105 22];

            % Create RecordingTimeEditFieldLabel
            app.RecordingTimeEditFieldLabel = uilabel(app.RecordTab);
            app.RecordingTimeEditFieldLabel.HorizontalAlignment = 'right';
            app.RecordingTimeEditFieldLabel.FontName = 'Bahnschrift';
            app.RecordingTimeEditFieldLabel.Position = [224 339 88 22];
            app.RecordingTimeEditFieldLabel.Text = 'Recording Time';

            % Create RecordingTimeEditField
            app.RecordingTimeEditField = uieditfield(app.RecordTab, 'numeric');
            app.RecordingTimeEditField.FontName = 'Bahnschrift';
            app.RecordingTimeEditField.Position = [327 339 100 22];
```

```
            % Create RecordButton
            app.RecordButton = uibutton(app.RecordTab, 'push');
            app.RecordButton.ButtonPushedFcn = createCallbackFcn(app,
@RecordButtonPushed, true);
            app.RecordButton.FontName = 'Bahnschrift';
            app.RecordButton.Position = [217 287 100 23];
            app.RecordButton.Text = 'Record';

            % Create PlayButton
            app.PlayButton = uibutton(app.RecordTab, 'push');
            app.PlayButton.ButtonPushedFcn = createCallbackFcn(app,
@PlayButtonPushed, true);
            app.PlayButton.FontName = 'Bahnschrift';
            app.PlayButton.Position = [331 287 100 23];
            app.PlayButton.Text = 'Play';

            % Create PlotSpectogramButton
            app.PlotSpectogramButton = uibutton(app.RecordTab, 'push');
            app.PlotSpectogramButton.ButtonPushedFcn = createCallbackFcn(app,
@PlotSpectogramButtonPushed, true);
            app.PlotSpectogramButton.Position = [28 61 104 22];
            app.PlotSpectogramButton.Text = 'Plot Spectogram';

            % Create WindowLengthEditFieldLabel
            app.WindowLengthEditFieldLabel = uilabel(app.RecordTab);
            app.WindowLengthEditFieldLabel.HorizontalAlignment = 'right';
            app.WindowLengthEditFieldLabel.Position = [4 188 88 22];
            app.WindowLengthEditFieldLabel.Text = 'Window Length';

            % Create WindowLengthEditField
            app.WindowLengthEditField = uieditfield(app.RecordTab, 'numeric');
            app.WindowLengthEditField.Position = [106 188 106 22];

            % Create WindowShiftEditFieldLabel
            app.WindowShiftEditFieldLabel = uilabel(app.RecordTab);
            app.WindowShiftEditFieldLabel.HorizontalAlignment = 'right';
            app.WindowShiftEditFieldLabel.Position = [5 145 76 22];
            app.WindowShiftEditFieldLabel.Text = 'Window Shift';

            % Create WindowShiftEditField
            app.WindowShiftEditField = uieditfield(app.RecordTab, 'numeric');
            app.WindowShiftEditField.Position = [106 159 106 22];

            % Create WindowTypeDropDown_2Label
            app.WindowTypeDropDown_2Label = uilabel(app.RecordTab);
            app.WindowTypeDropDown_2Label.HorizontalAlignment = 'right';
            app.WindowTypeDropDown_2Label.Position = [6 110 77 22];
            app.WindowTypeDropDown_2Label.Text = 'Window Type';

            % Create WindowTypeDropDown_2
            app.WindowTypeDropDown_2 = uidropdown(app.RecordTab);
            app.WindowTypeDropDown_2.Items = {'Rectangular', 'Tukey', 'Hamming'};
            app.WindowTypeDropDown_2.Position = [109 124 100 22];
            app.WindowTypeDropDown_2.Value = 'Rectangular';

            % Create Lamp
            app.Lamp = uilamp(app.RecordTab);
            app.Lamp.Position = [170 274 20 20];
```

```
            % Create SelectFileTab
            app.SelectFileTab = uitab(app.TabGroup);
            app.SelectFileTab.Title = 'Select File';

            % Create UIAxes_2
            app.UIAxes_2 = uiaxes(app.SelectFileTab);
            title(app.UIAxes_2, 'Title')
            xlabel(app.UIAxes_2, 'X')
            ylabel(app.UIAxes_2, 'Y')
            zlabel(app.UIAxes_2, 'Z')
            app.UIAxes_2.Position = [288 60 300 185];

            % Create UploadaFileButton
            app.UploadaFileButton = uibutton(app.SelectFileTab, 'push');
            app.UploadaFileButton.ButtonPushedFcn = createCallbackFcn(app,
@UploadaFileButtonPushed, true);
            app.UploadaFileButton.FontName = 'Bahnschrift';
            app.UploadaFileButton.Position = [270 365 100 23];
            app.UploadaFileButton.Text = 'Upload a File';

            % Create PlayButton_2
            app.PlayButton_2 = uibutton(app.SelectFileTab, 'push');
            app.PlayButton_2.ButtonPushedFcn = createCallbackFcn(app,
@PlayButton_2Pushed, true);
            app.PlayButton_2.FontName = 'Bahnschrift';
            app.PlayButton_2.Position = [270 323 100 23];
            app.PlayButton_2.Text = 'Play';

            % Create PlotSpectogramButton_2
            app.PlotSpectogramButton_2 = uibutton(app.SelectFileTab, 'push');
            app.PlotSpectogramButton_2.ButtonPushedFcn = createCallbackFcn(app,
@PlotSpectogramButton_2Pushed, true);
            app.PlotSpectogramButton_2.Position = [28 47 104 22];
            app.PlotSpectogramButton_2.Text = 'Plot Spectogram';

            % Create WindowLengthEditField_2Label
            app.WindowLengthEditField_2Label = uilabel(app.SelectFileTab);
            app.WindowLengthEditField_2Label.HorizontalAlignment = 'right';
            app.WindowLengthEditField_2Label.Position = [4 174 88 22];
            app.WindowLengthEditField_2Label.Text = 'Window Length';

            % Create WindowLengthEditField_2
            app.WindowLengthEditField_2 = uieditfield(app.SelectFileTab,
'numeric');
            app.WindowLengthEditField_2.Position = [106 174 106 22];

            % Create WindowShiftEditField_2Label
            app.WindowShiftEditField_2Label = uilabel(app.SelectFileTab);
            app.WindowShiftEditField_2Label.HorizontalAlignment = 'right';
            app.WindowShiftEditField_2Label.Position = [5 159 76 22];
            app.WindowShiftEditField_2Label.Text = 'Window Shift';

            % Create WindowShiftEditField_2
            app.WindowShiftEditField_2 = uieditfield(app.SelectFileTab,
'numeric');
            app.WindowShiftEditField_2.Position = [106 145 106 22];

            % Create WindowTypeDropDown_3Label
            app.WindowTypeDropDown_3Label = uilabel(app.SelectFileTab);
```

```
            app.WindowTypeDropDown_3Label.HorizontalAlignment = 'right';
            app.WindowTypeDropDown_3Label.Position = [6 110 77 22];
            app.WindowTypeDropDown_3Label.Text = 'Window Type';

            % Create WindowTypeDropDown_3
            app.WindowTypeDropDown_3 = uidropdown(app.SelectFileTab);
            app.WindowTypeDropDown_3.Items = {'Rectangular', 'Tukey', 'Hamming'};
            app.WindowTypeDropDown_3.Position = [109 110 100 22];
            app.WindowTypeDropDown_3.Value = 'Rectangular';

            % Create GenerateTab
            app.GenerateTab = uitab(app.TabGroup);
            app.GenerateTab.SizeChangedFcn = createCallbackFcn(app,
@GenerateTabSizeChanged, true);
            app.GenerateTab.Title = 'Generate';

            % Create UIAxes2
            app.UIAxes2 = uiaxes(app.GenerateTab);
            title(app.UIAxes2, 'Title')
            xlabel(app.UIAxes2, 'X')
            ylabel(app.UIAxes2, 'Y')
            zlabel(app.UIAxes2, 'Z')
            app.UIAxes2.Position = [230 44 187 113];

            % Create UIAxes2_2
            app.UIAxes2_2 = uiaxes(app.GenerateTab);
            title(app.UIAxes2_2, 'Title')
            xlabel(app.UIAxes2_2, 'X')
            ylabel(app.UIAxes2_2, 'Y')
            zlabel(app.UIAxes2_2, 'Z')
            app.UIAxes2_2.Position = [437 41 183 119];

            % Create SinusoidalCheckBox
            app.SinusoidalCheckBox = uicheckbox(app.GenerateTab);
            app.SinusoidalCheckBox.ValueChangedFcn = createCallbackFcn(app,
@SinusoidalCheckBoxValueChanged2, true);
            app.SinusoidalCheckBox.Text = 'Sinusoidal';
            app.SinusoidalCheckBox.FontName = 'Bahnschrift';
            app.SinusoidalCheckBox.Position = [42 389 78 22];

            % Create WindowedsinusoidalCheckBox
            app.WindowedsinusoidalCheckBox = uicheckbox(app.GenerateTab);
            app.WindowedsinusoidalCheckBox.ValueChangedFcn =
createCallbackFcn(app, @WindowedsinusoidalCheckBoxValueChanged2, true);
            app.WindowedsinusoidalCheckBox.Text = 'Windowed sinusoidal';
            app.WindowedsinusoidalCheckBox.FontName = 'Bahnschrift';
            app.WindowedsinusoidalCheckBox.Position = [240 375 135 22];

            % Create SignalwithmultiplecomponentsCheckBox
            app.SignalwithmultiplecomponentsCheckBox =
uicheckbox(app.GenerateTab);
            app.SignalwithmultiplecomponentsCheckBox.ValueChangedFcn =
createCallbackFcn(app, @SignalwithmultiplecomponentsCheckBoxValueChanged2, true);
            app.SignalwithmultiplecomponentsCheckBox.Text = 'Signal with multiple
components';
            app.SignalwithmultiplecomponentsCheckBox.FontName = 'Bahnschrift';
            app.SignalwithmultiplecomponentsCheckBox.Position = [429 374 199 22];

            % Create WindowTypeDropDown
```

```
            app.WindowTypeDropDown = uidropdown(app.GenerateTab);
            app.WindowTypeDropDown.Items = {'Rectangular', 'Hamming', 'Tukey'};
            app.WindowTypeDropDown.Position = [305 261 101 22];
            app.WindowTypeDropDown.Value = 'Rectangular';

            % Create ComponetsEditField
            app.ComponetsEditField = uieditfield(app.GenerateTab, 'text');
            app.ComponetsEditField.Position = [531 347 31 22];

            % Create AmplitudeEditField
            app.AmplitudeEditField = uieditfield(app.GenerateTab, 'numeric');
            app.AmplitudeEditField.Position = [524 308 46 22];

            % Create FrequencyEditField
            app.FrequencyEditField = uieditfield(app.GenerateTab, 'numeric');
            app.FrequencyEditField.Position = [524 282 44 22];

            % Create SubmitButton
            app.SubmitButton = uibutton(app.GenerateTab, 'push');
            app.SubmitButton.ButtonPushedFcn = createCallbackFcn(app,
@SubmitButtonPushed, true);
            app.SubmitButton.Position = [575 282 53 23];
            app.SubmitButton.Text = 'Submit';

            % Create PhaseEditField
            app.PhaseEditField = uieditfield(app.GenerateTab, 'numeric');
            app.PhaseEditField.Position = [524 249 44 22];

            % Create PlotSpectogramButton_3
            app.PlotSpectogramButton_3 = uibutton(app.GenerateTab, 'push');
            app.PlotSpectogramButton_3.ButtonPushedFcn = createCallbackFcn(app,
@PlotSpectogramButton_3Pushed, true);
            app.PlotSpectogramButton_3.Position = [42 7 104 22];
            app.PlotSpectogramButton_3.Text = 'Plot Spectogram';

            % Create AmplitudeLabel
            app.AmplitudeLabel = uilabel(app.GenerateTab);
            app.AmplitudeLabel.Position = [450 307 59 22];
            app.AmplitudeLabel.Text = 'Amplitude';

            % Create FrequencyLabel
            app.FrequencyLabel = uilabel(app.GenerateTab);
            app.FrequencyLabel.Position = [452 282 62 22];
            app.FrequencyLabel.Text = 'Frequency';

            % Create PhaseLabel
            app.PhaseLabel = uilabel(app.GenerateTab);
            app.PhaseLabel.Position = [455 250 40 22];
            app.PhaseLabel.Text = 'Phase';

            % Create SubmitButton_2
            app.SubmitButton_2 = uibutton(app.GenerateTab, 'push');
            app.SubmitButton_2.ButtonPushedFcn = createCallbackFcn(app,
@SubmitButton_2Pushed, true);
            app.SubmitButton_2.Position = [574 345 53 23];
            app.SubmitButton_2.Text = 'Submit';

            % Create WindowLengthEditField_3Label
            app.WindowLengthEditField_3Label = uilabel(app.GenerateTab);
```

```
app.WindowLengthEditField_3Label.HorizontalAlignment = 'right';
app.WindowLengthEditField_3Label.Position = [4 119 88 22];
app.WindowLengthEditField_3Label.Text = 'Window Length';

% Create WindowLengthEditField_3
app.WindowLengthEditField_3 = uieditfield(app.GenerateTab, 'numeric');
app.WindowLengthEditField_3.Position = [106 119 106 22];

% Create WindowShiftEditField_3Label
app.WindowShiftEditField_3Label = uilabel(app.GenerateTab);
app.WindowShiftEditField_3Label.HorizontalAlignment = 'right';
app.WindowShiftEditField_3Label.Position = [5 76 76 22];
app.WindowShiftEditField_3Label.Text = 'Window Shift';

% Create WindowShiftEditField_3
app.WindowShiftEditField_3 = uieditfield(app.GenerateTab, 'numeric');
app.WindowShiftEditField_3.Position = [106 76 106 22];

% Create WindowTypeDropDown_4Label
app.WindowTypeDropDown_4Label = uilabel(app.GenerateTab);
app.WindowTypeDropDown_4Label.HorizontalAlignment = 'right';
app.WindowTypeDropDown_4Label.Position = [6 41 77 22];
app.WindowTypeDropDown_4Label.Text = 'Window Type';

% Create WindowTypeDropDown_4
app.WindowTypeDropDown_4 = uidropdown(app.GenerateTab);
app.WindowTypeDropDown_4.Items = {'Rectangular', 'Tukey', 'Hamming'};
app.WindowTypeDropDown_4.Position = [109 41 100 22];
app.WindowTypeDropDown_4.Value = 'Rectangular';

% Create AmplitudeEditField_2
app.AmplitudeEditField_2 = uieditfield(app.GenerateTab, 'numeric');
app.AmplitudeEditField_2.Position = [82 319 63 22];

% Create FrequencyEditField_2
app.FrequencyEditField_2 = uieditfield(app.GenerateTab, 'numeric');
app.FrequencyEditField_2.Position = [83 289 62 22];

% Create PhaseEditField_2
app.PhaseEditField_2 = uieditfield(app.GenerateTab, 'numeric');
app.PhaseEditField_2.Position = [83 259 62 22];

% Create AmplitudeLabel_2
app.AmplitudeLabel_2 = uilabel(app.GenerateTab);
app.AmplitudeLabel_2.Position = [14 318 59 22];
app.AmplitudeLabel_2.Text = 'Amplitude';

% Create FrequencyLabel_2
app.FrequencyLabel_2 = uilabel(app.GenerateTab);
app.FrequencyLabel_2.Position = [14 289 62 22];
app.FrequencyLabel_2.Text = 'Frequency';

% Create PhaseLabel_2
app.PhaseLabel_2 = uilabel(app.GenerateTab);
app.PhaseLabel_2.Position = [14 260 40 22];
app.PhaseLabel_2.Text = 'Phase';

% Create AmplitudeEditField_3
app.AmplitudeEditField_3 = uieditfield(app.GenerateTab, 'numeric');
```

```
app.AmplitudeEditField_3.Position = [309 347 63 22];

% Create FrequencyEditField_3
app.FrequencyEditField_3 = uieditfield(app.GenerateTab, 'numeric');
app.FrequencyEditField_3.Position = [310 317 62 22];

% Create StartTimeEditField
app.StartTimeEditField = uieditfield(app.GenerateTab, 'numeric');
app.StartTimeEditField.Position = [311 231 62 22];

% Create AmplitudeLabel_3
app.AmplitudeLabel_3 = uilabel(app.GenerateTab);
app.AmplitudeLabel_3.Position = [241 346 59 22];
app.AmplitudeLabel_3.Text = 'Amplitude';

% Create FrequencyLabel_3
app.FrequencyLabel_3 = uilabel(app.GenerateTab);
app.FrequencyLabel_3.Position = [241 331 62 22];
app.FrequencyLabel_3.Text = 'Frequency';

% Create StartTimeLabel
app.StartTimeLabel = uilabel(app.GenerateTab);
app.StartTimeLabel.Position = [242 232 60 22];
app.StartTimeLabel.Text = 'Start Time';

% Create LengthEditField
app.LengthEditField = uieditfield(app.GenerateTab, 'numeric');
app.LengthEditField.Position = [311 203 62 22];

% Create LengthLabel
app.LengthLabel = uilabel(app.GenerateTab);
app.LengthLabel.Position = [242 204 42 22];
app.LengthLabel.Text = 'Length';

% Create PhaseEditField_3
app.PhaseEditField_3 = uieditfield(app.GenerateTab, 'numeric');
app.PhaseEditField_3.Position = [311 290 62 22];

% Create PhaseLabel_3
app.PhaseLabel_3 = uilabel(app.GenerateTab);
app.PhaseLabel_3.Position = [242 290 40 22];
app.PhaseLabel_3.Text = 'Phase';

% Create WindowTypeLabel
app.WindowTypeLabel = uilabel(app.GenerateTab);
app.WindowTypeLabel.Position = [222 261 77 22];
app.WindowTypeLabel.Text = 'Window Type';

% Create ComponentsLabel
app.ComponentsLabel = uilabel(app.GenerateTab);
app.ComponentsLabel.Position = [446 346 80 22];
app.ComponentsLabel.Text = '#Components';

% Create DurationLabel
app.DurationLabel = uilabel(app.GenerateTab);
app.DurationLabel.Position = [14 229 51 22];
app.DurationLabel.Text = 'Duration';

% Create DurationEditField
```

```
            app.DurationEditField = uieditfield(app.GenerateTab, 'numeric');
            app.DurationEditField.Position = [83 228 62 22];

            % Create DurationLabel_2
            app.DurationLabel_2 = uilabel(app.GenerateTab);
            app.DurationLabel_2.Position = [454 217 51 22];
            app.DurationLabel_2.Text = 'Duration';

            % Create DurationEditField_2
            app.DurationEditField_2 = uieditfield(app.GenerateTab, 'numeric');
            app.DurationEditField_2.Position = [525 216 42 22];

            % Create SamplingRateEditField_2Label
            app.SamplingRateEditField_2Label = uilabel(app.GenerateTab);
            app.SamplingRateEditField_2Label.HorizontalAlignment = 'right';
            app.SamplingRateEditField_2Label.Position = [8 145 84 22];
            app.SamplingRateEditField_2Label.Text = 'Sampling Rate';

            % Create SamplingRateEditField_2
            app.SamplingRateEditField_2 = uieditfield(app.GenerateTab, 'numeric');
            app.SamplingRateEditField_2.Position = [106 145 106 22];

            % Create ContextMenu
            app.ContextMenu = uicontextmenu(app.UIFigure);

            % Create Menu
            app.Menu = uimenu(app.ContextMenu);
            app.Menu.Text = 'Menu';

            % Create Menu_2
            app.Menu_2 = uimenu(app.Menu);
            app.Menu_2.Text = 'Menu';

            % Create Menu2_2
            app.Menu2_2 = uimenu(app.Menu);
            app.Menu2_2.Text = 'Menu2';

            % Create Menu_3
            app.Menu_3 = uimenu(app.Menu2_2);
            app.Menu_3.Text = 'Menu';

            % Create Menu2
            app.Menu2 = uimenu(app.ContextMenu);
            app.Menu2.Text = 'Menu2';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = SpectogramAppVersion4_exported

            % Create UIFigure and components
            createComponents(app)
```

```
            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

```
function
spectogram_plotter_generated_signals_vol2(generated_signal,window_length,window_sh
ift,window_type,sampling_frequency)

    % Transpose is taken if the signal is a column vector , instead of a
    % row vector

    if(size(generated_signal,1) > 5)
        generated_signal = generated_signal';
    end

    win = 0;    % initial blank window

    %************************************************
    % Controlling if we satisfy the length constraints:

    if( length(generated_signal) < window_length )

    disp(' Error: The signal is too short for this window length. Submit a new
signal or choose a new window length.');

    end

    if( window_shift > window_length )
    disp(' Error: Enter a valid window length that is longer than the window
shift.');
    end

    %************************************************


    % Assigning the window type:
    if( strcmp(window_type,'rectwin')) %#ok<*STCMP>
        win=ones(1,window_length);
    elseif(strcmp(window_type,'hamming'))
        win=hamming(window_length)';
    elseif(strcmp(window_type,'tukeywin'))
        win=tukeywin(window_length)';
    else
        win=ones(1,window_length);
    end

    %************************************************

    begin_point = 1;    % Initial point for thewindow
    vertical_line_no=1; % Counter for the vertical lines

    %Calculating the STFT

    while( begin_point + window_length -1 <= length(generated_signal) )

        STFT(vertical_line_no,:) = abs(fft(generated_signal( begin_point :
begin_point+window_length-1 ).*win,window_length*100));
        begin_point = begin_point + window_shift;
        vertical_line_no = vertical_line_no+1 ;

    end
```

```
    time_values = 0 : size(STFT,1)-1; %Time values to be plotted
    frequency_values = linspace(0,sampling_frequency,window_length*50); %The
default fs = 4000


    STFT_in_DB=(20*log10(STFT))'; % Coverting the magnitudes to decibels for
better analysis

    %************************************************

    % Plotting the STFT

    surf(axes,time_values,frequency_values,STFT_in_DB(1:window_length*50,:));
    shading("interp");
    view(2); %The view should be from above since our plot is 2D in view
    xlabel('Time','Fontsize',12);
    ylabel('Frequency(Hz)','Fontsize',12);
    title('Spectrogram','Fontsize',14);
    colorbar;
    ylim([0 sampling_frequency]);
    xlim([0 size(STFT,1)]);
    yticks(linspace(0, sampling_frequency, 6));

end
```