Yunus Emre Tüysüz-2444032
Ata Oğuz Tanrıkulu 2443927

12.5.2023

# EE430: Term Project Part 1

## 1. Introduction:

In this project, we will delve into the Short-Time Fourier Transform (STFT) concept, gaining a comprehensive understanding of how it operates. We will leverage this knowledge to create our spectrogram function, which will be applied to a variety of signals. These signals will encompass both synthetically generated sinusoidal signals and real-world audio captured via a microphone or sourced from existing recordings. Our primary objective is to conduct an in-depth analysis of these signals through time-domain and spectrogram visualizations, while also exploring the impact of different parameters. A MATLAB App was implemented to achieve these functionalities, which will be explained in detail in this report. The outcomes of these investigations, along with accompanying plots, will be presented in this report.

## 2. MATLAB Application

In this part, functionalities of our MATLAB app and its graphical interface is explained.

### 2.1. Data Acquisition

In this part, input data is taken from a user as a sound signal. We use two modes to get the data: Sound data from a microphone, Sound data from a file. In the app, two pages was constructed for each of these functionalities.

#### 2.1.1. Sound data from a microphone

In this part, sound data is acquired by recording the user sound. User can adjust the sampling rate of the recording. They can start recording by pushing record button and stop it by pushing stop button. They can play the sound afterwards if they want to.

To visualize its time domain signal and STFT, they can press spectrogram button. They can also adjust window type, shift and length. Image of the interface was shown in the Figure-1 and 2.

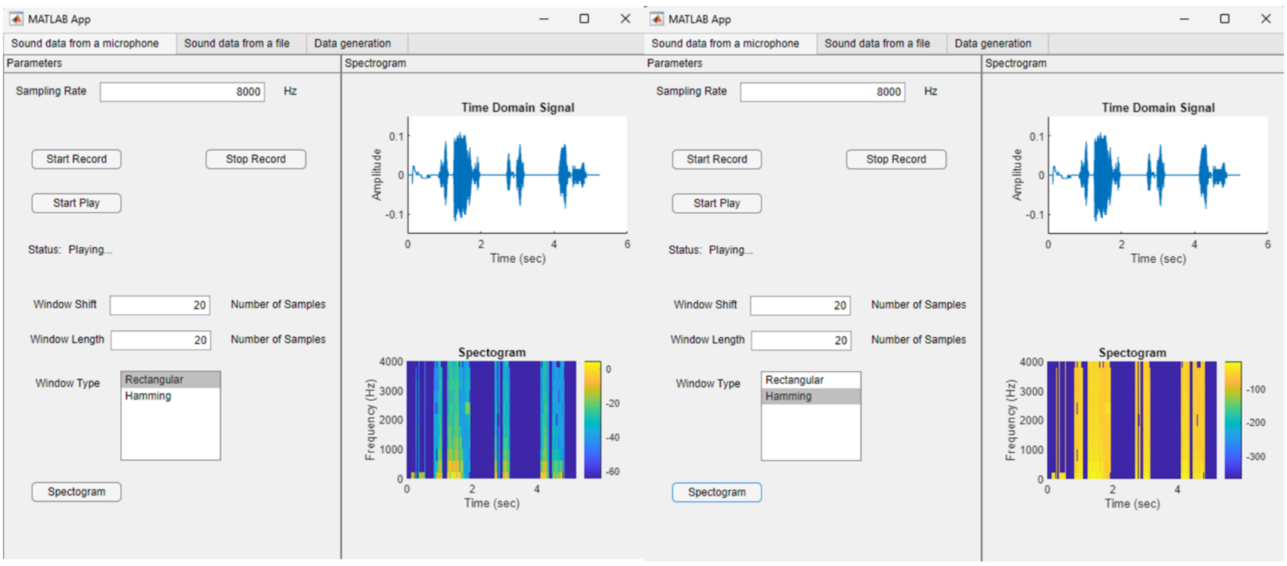


*Figure 1 First example that shows functionality of Data acquisition using microphone*

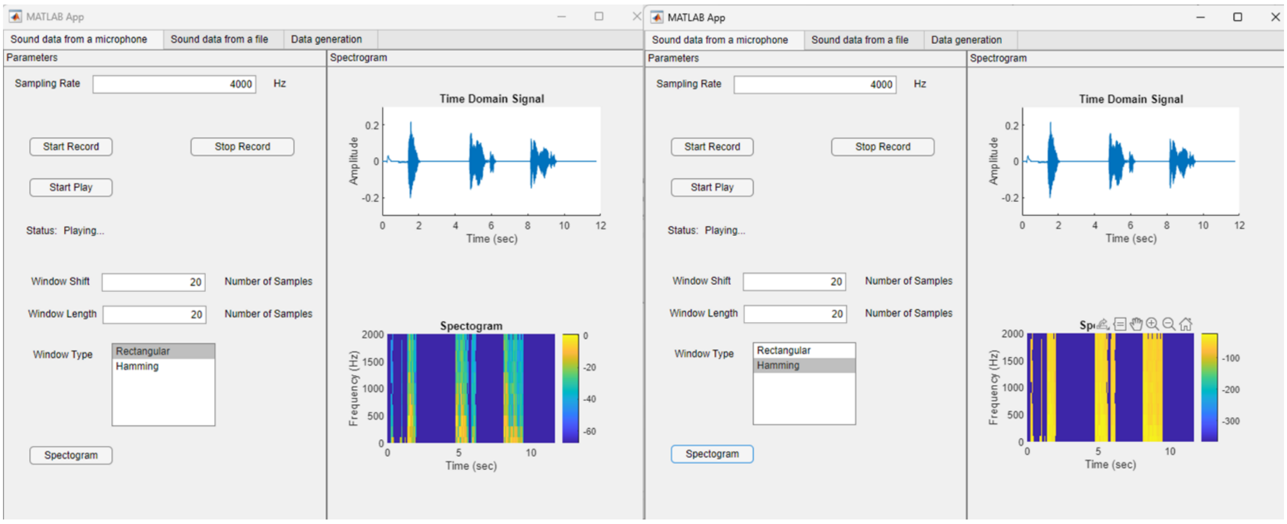


Figure 2 Second example that shows functionality of Data acquisition using microphone

## 2.1.2. Sound data from a file

Users also upload a ".wav" or ".mp3" as a sound signal and plot its spectrogram and time domain signal. Like the previous part, they can adjust window type, shift, length of the spectrogram and listen the sound signal they were uploaded. Interface is shown in Figure-3.

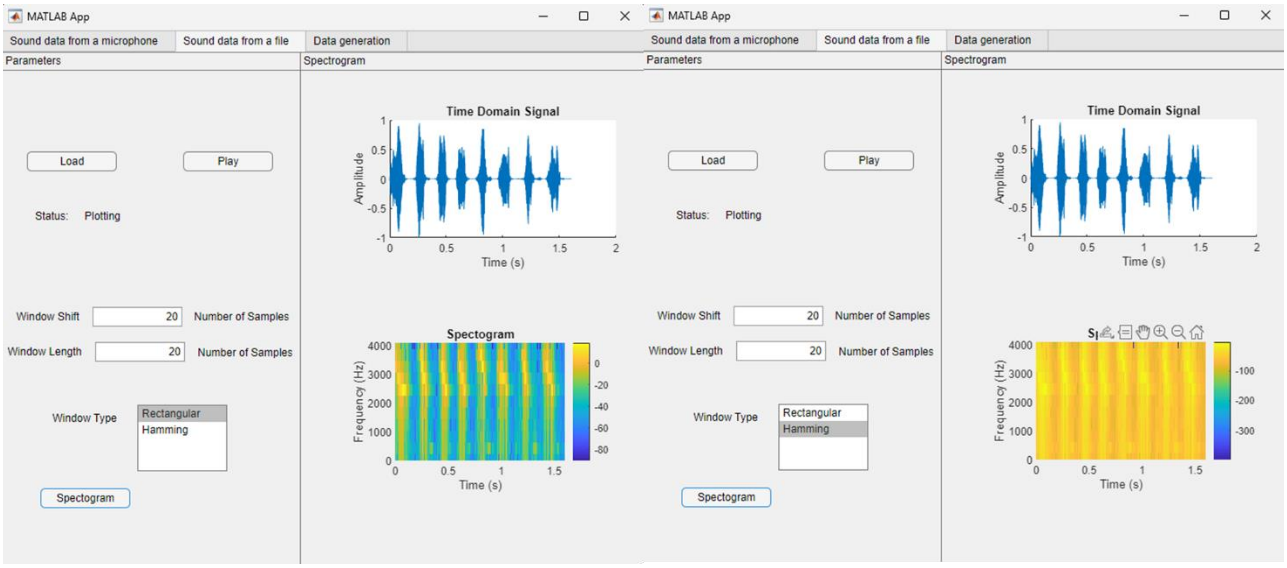


Figure 3 Example that shows functionality of Data acquisition using sound data from a file

## 2.2. Data Generation

In this part, desired data can be created by a user. We have three different data generation modes: Sinusoidal signal, windowed sinusoidal and signal involving multiple components. For each mode user can adjust the parameters such as amplitude, frequency, phase and duration. Also, user can see time-domain and frequency-domain plots.

### 2.2.1. Sinusoidal signal

In this part, sinusoidal signal can be generated by specifying the parameters such as amplitude frequency phase, duration and sampling rate. The examples can be seen in Figure 4.

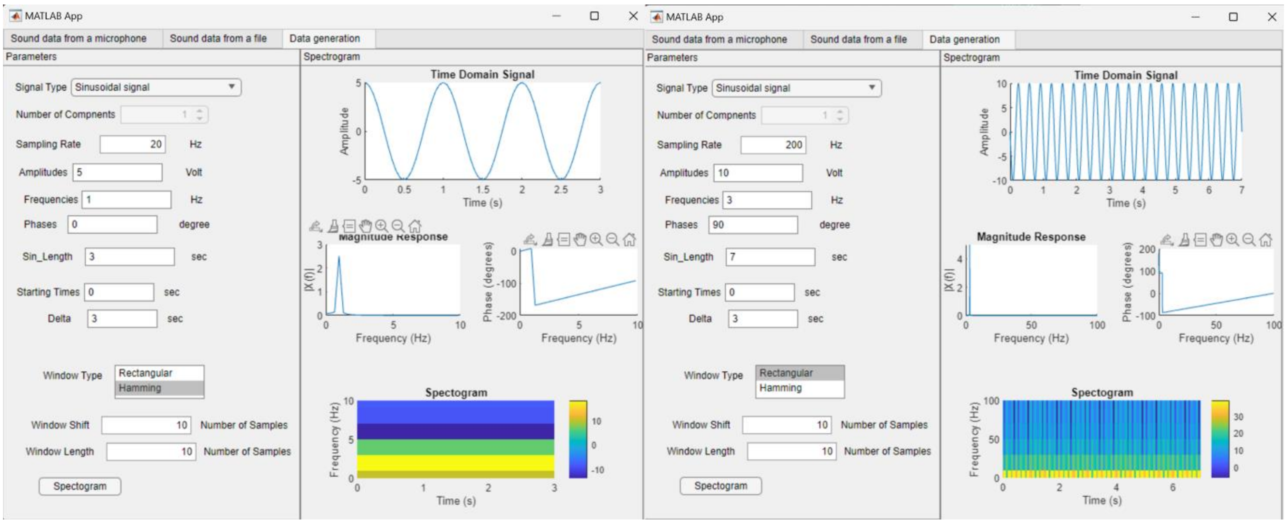


*Figure 4 Generating 2 different sinusoidal by setting different parameters.*

### 2.2.2. Windowed sinusoidal

In this part, windowed sinusoidal can be generated by selecting window type and setting the parameters such as amplitude frequency phase, duration and sampling rate. The examples can be seen in Figure 5.

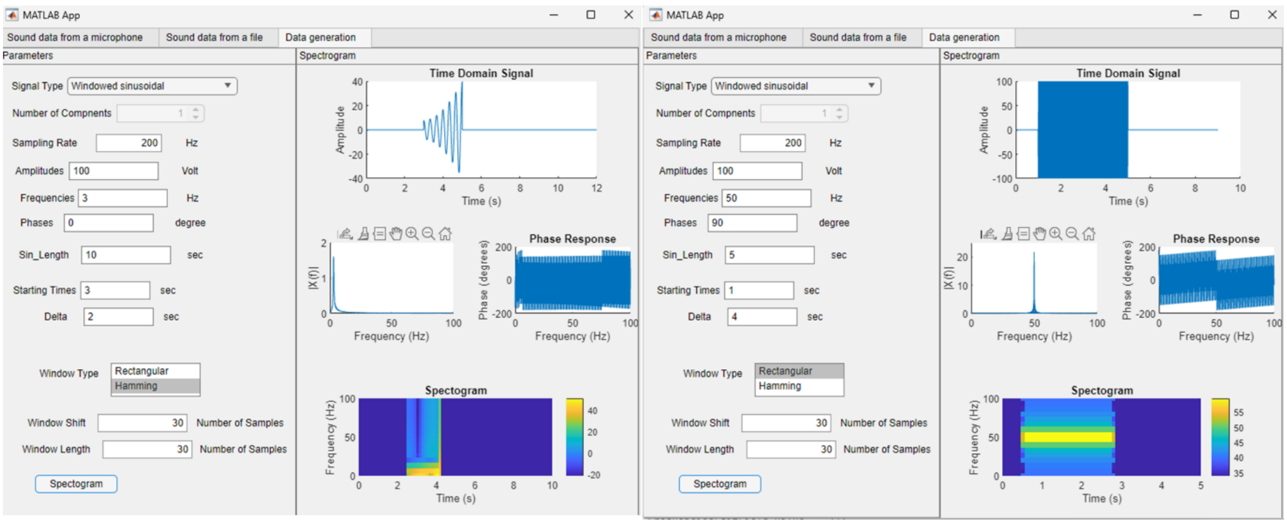


*Figure 5 Generating 2 different windowed sinusoidal using different parameters.*

### 2.2.3. Signal involving multiple components

In this part, signal involving multiple components generated by setting the parameters such as number of components, amplitude frequency phase, duration for each component and sampling rate. The examples can be seen in Figure 6 and 7 respectively.

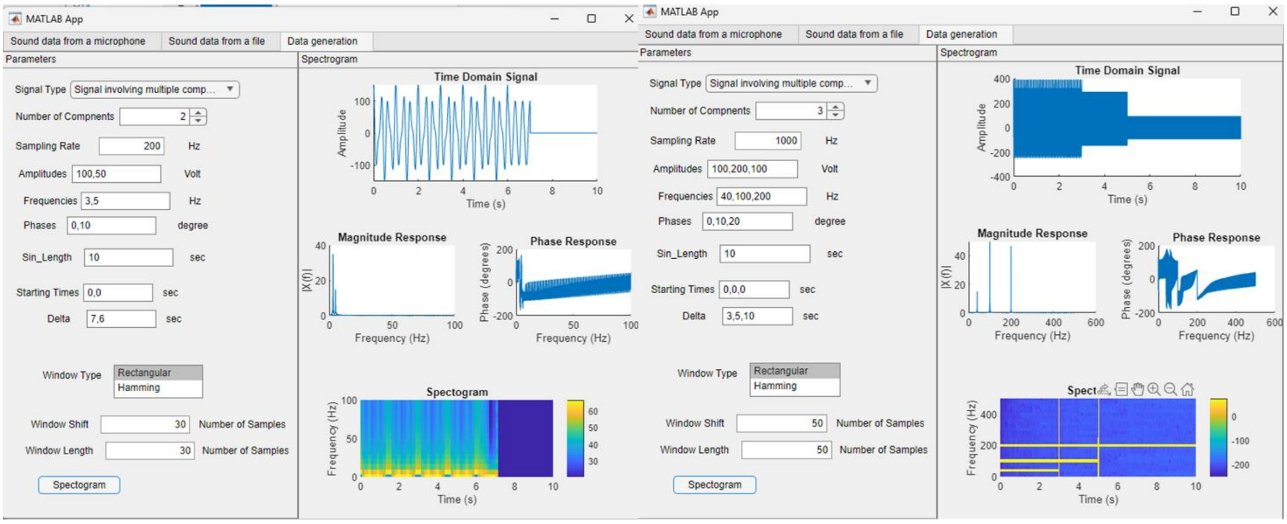


*Figure 6 Generating 2 different signal that contains multiple sinusoidal components.*

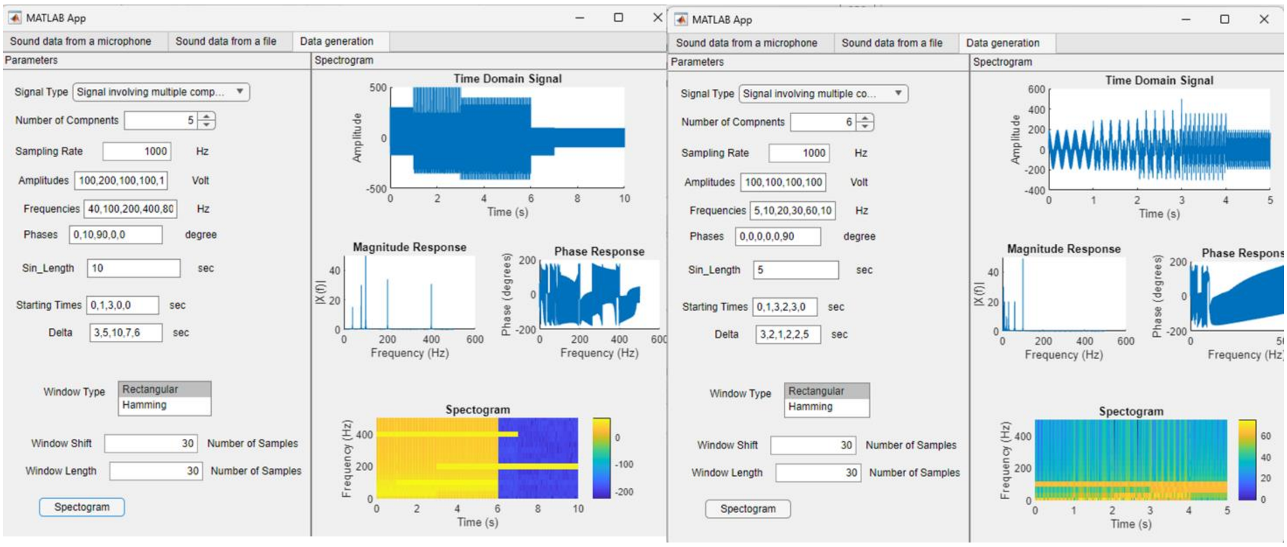


*Figure 7 More example that shows functionality of the generation of signal involving multiple components*

## 2.3. Spectrogram

In this part, we are required to develop MATLAB code for creating a spectrogram of a discrete-time signal. The spectrogram, a visual representation of frequency variations over time, is plotted using the magnitude of the short-time Fourier transform (STFT). This involves computing the STFT for the signal, followed by its discrete Fourier transform (DFT), which highlights the frequency content at specific time intervals. Key aspects of this process include selecting the window length and shift, as well as the type of window function, each influencing the analysis outcome. Our task extends to comparing our spectrogram with MATLAB's built-in function, deepening our understanding of digital signal processing techniques. For each figure the spectrogram has already shown. More examples will be provided, and the spectrogram code can be seen in Appendix.

# 3. Answer to the Why question

Setting the window shift equal to or less than the window length (also known as the frame length) has several implications:

1. **Overlap:** When the window shift is equal to the window length, there is no overlap between consecutive frames. This can result in a spectrogram with high temporal resolution but may introduce artifacts in the analysis, making it less suitable for certain applications. Overlapping frames, achieved by setting the window shift to a value less than the window length, help mitigate these artifacts.
2. **Resolution:** A smaller window shift allows for finer temporal resolution because it samples the signal more frequently. This is particularly important when analysing signals with rapidly changing characteristics, such as musical notes or speech phonemes.
3. **Smoothing:** Overlapping frames can be used to smooth the spectrogram and reduce noise or small-scale variations in the signal. By averaging information from adjacent frames, you can obtain a more stable representation of the signal's spectral content over time.
4. **Temporal Precision:** In some applications, such as speech analysis or musical instrument recognition, it's essential to capture the fine details of the signal's temporal dynamics. A window shift smaller than the window length helps in achieving this precision.

However, it's worth noting that a smaller window shift also increases computational requirements because more frames need to be processed, which can be a trade-off in terms of processing speed and memory usage. Therefore, the choice of window shift should be made based on the specific requirements of your analysis and the trade-offs between temporal resolution, computational complexity, and the characteristics of the signal you are analysing.

# 4. Questions

In this part, questions of the project were answered and required plots and figures were demonstrated.

## 4.1. Question 1

In this part, an audio signal was recorded with the “Sound data from a microphone” option and its time domain signal and spectrogram was investigated. Word “Report” was recorded by us. Time domain signal, spectrogram and, sampling rate, window options could be seen in Figure 8 at the app interface.

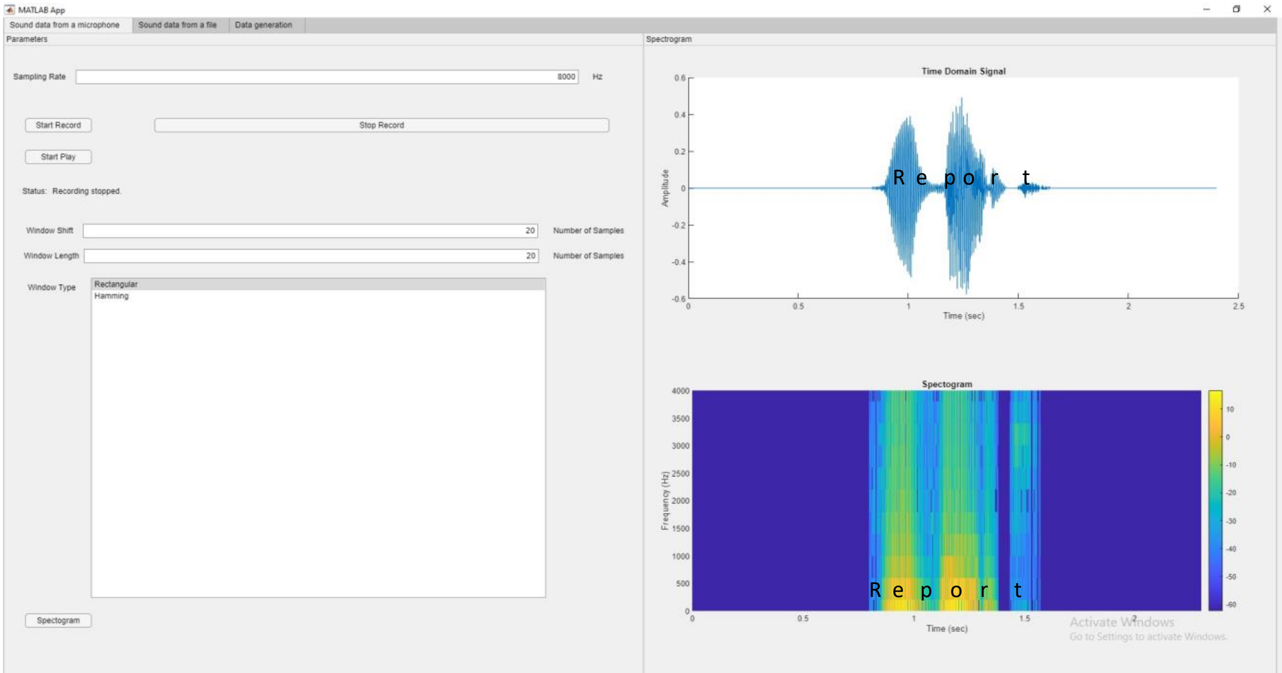


*Figure 8 A word ''Report'' recorded using microphone.*

In this figure, each letter of the "Report" was shown in time domain and spectrogram.

As you can see, sampling rate is chosen as 8 kHz. Usually, human speech has bandlimited to 20 Hz – 20 kHz. So, to fully record a human voice without aliasing, minimally 40 kHz sampling rate should be used. However, most of the signal power in human speech is limited to 3-4 kHz. So, practically using 8 kHz as sampling rate is mostly efficient to record human audio signals.

Also, spectrogram of the same signal was plotted for hamming window type (other parameters were kept same). Result is in Figure 9.

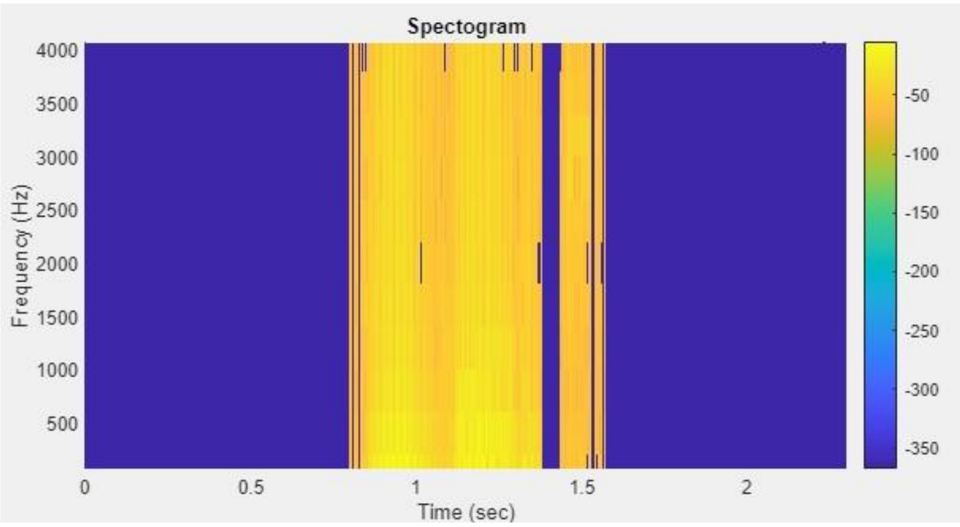


*Figure 9 Spectrogram of same signal using hamming windows type.*

## 4.2. Question 2

In this part, Dual-tone multi-frequency (DTMF) is investigated.

### 4.2.1. i.

In this part, DTMF signal of sequence [4,3,0] was generated. Duration of the keys were 40, 50 and 60 ms respectively. Sampling rate is set as 4 kHz. Time domain, frequency domain and STFT of the signal were shown in below Figure 10 and 11 with different window options.

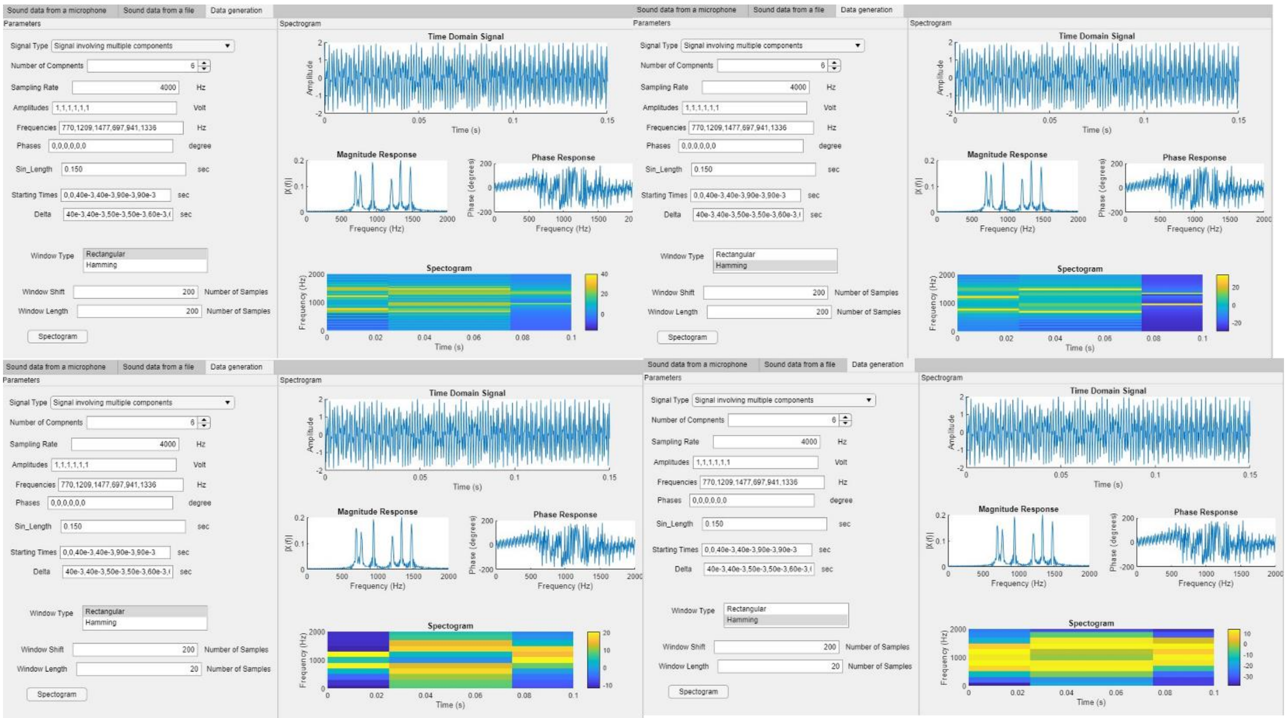

*Figure 10: Example of Time, Frequency Domain and STFT of a DTMF signal*

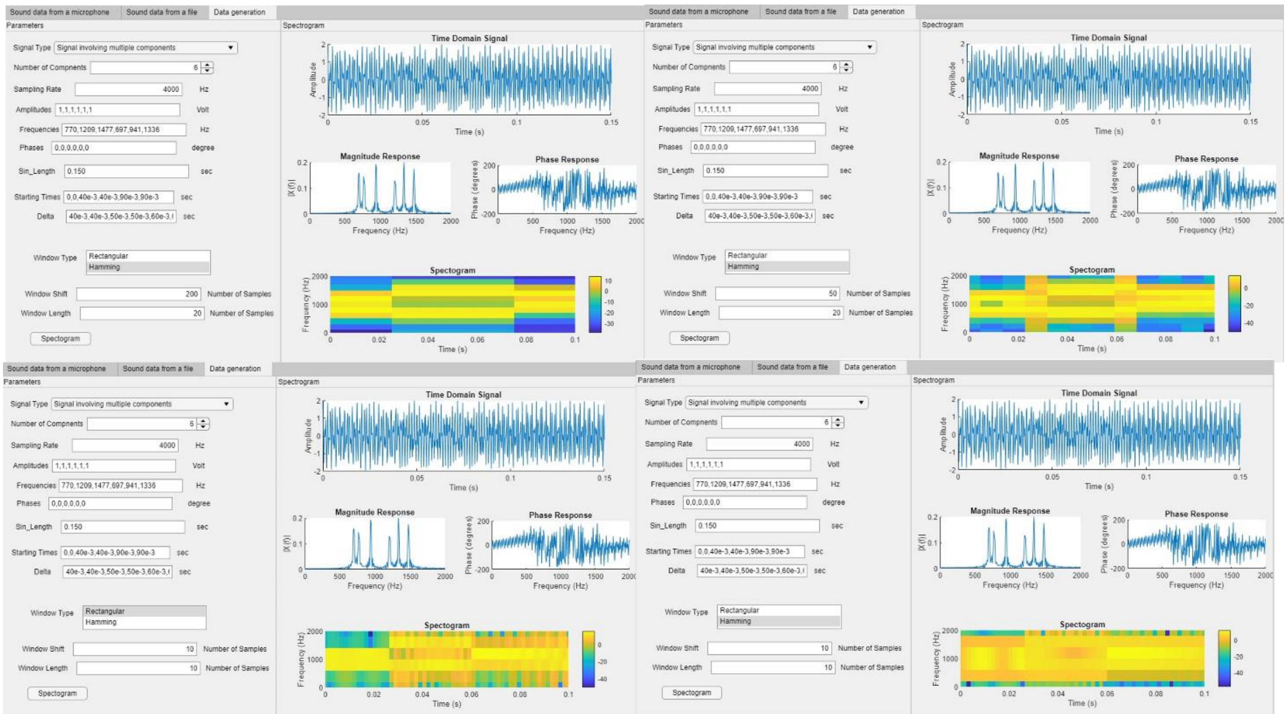

*Figure 11: Example of Time, Frequency Domain and STFT of a DTMF signal*

- **Time-Frequency Resolution Trade-off:** When you vary the window length and window shift, you'll notice a trade-off between time and frequency resolution. Shorter windows (and smaller shifts) provide better time resolution but poorer frequency resolution, making them suitable for capturing transient events in the signal. Longer windows (and larger shifts) offer better frequency resolution but may smear out rapid changes in the signal over time.
- **Effects of Different Window Types:** Different window types (e.g., rectangular and Hamming) affect the spectral characteristics of the signal. Hamming windows, for example, reduce spectral leakage but have a wider main lobe compared to rectangular windows. The choice of window type can impact the sharpness of spectral peaks and the suppression of sidelobes in the spectrogram.

### 4.2.2. ii.

In this part, 3 signals were generated at 1 kHz sampling rate, which involve 2 sinusoid waves. To sample the Discrete-Time Fourier Transform (DTFT) exactly at the frequencies f1 and f2, we need to ensure that the sampling frequency (fs) is an integer multiple of the difference between these frequencies. Mathematically, this can be expressed as:

fs = k * |f2 - f1|, where k is an integer where fs = 1000 Hz

For the three frequency pairs:

For (100 Hz, 110 Hz): fs = k * |110 Hz - 100 Hz| = 10 Hz. The minimum k that satisfies this equation is k = 100.

For (100 Hz, 150 Hz): fs = k * |150 Hz - 100 Hz| = 50 Hz. The minimum k that satisfies this equation is k = 20.

For (100 Hz, 200 Hz): fs = k * |200 Hz - 100 Hz| = 100 Hz. The minimum k that satisfies this equation is k = 10.

Therefore, the minimum number of frequency points needed to sample the DTFT exactly at the frequencies f1 and f2 for each pair is 100, 20, and 10, respectively. These values ensure that the sampling frequency is an integer multiple of the frequency difference, allowing for accurate representation of the sinusoidal components in the signal.

According to these values, time domain, frequency domain and STFT of the signals were plotted. Sampling rate adjusted as 1000 Hz and duration of the signal set as 1 sec, so that there 1000 sample points. For STFT calculation, in order to see exact frequencies, window shift should be set as the |f1 – f2|. Plots are in Figure 12.

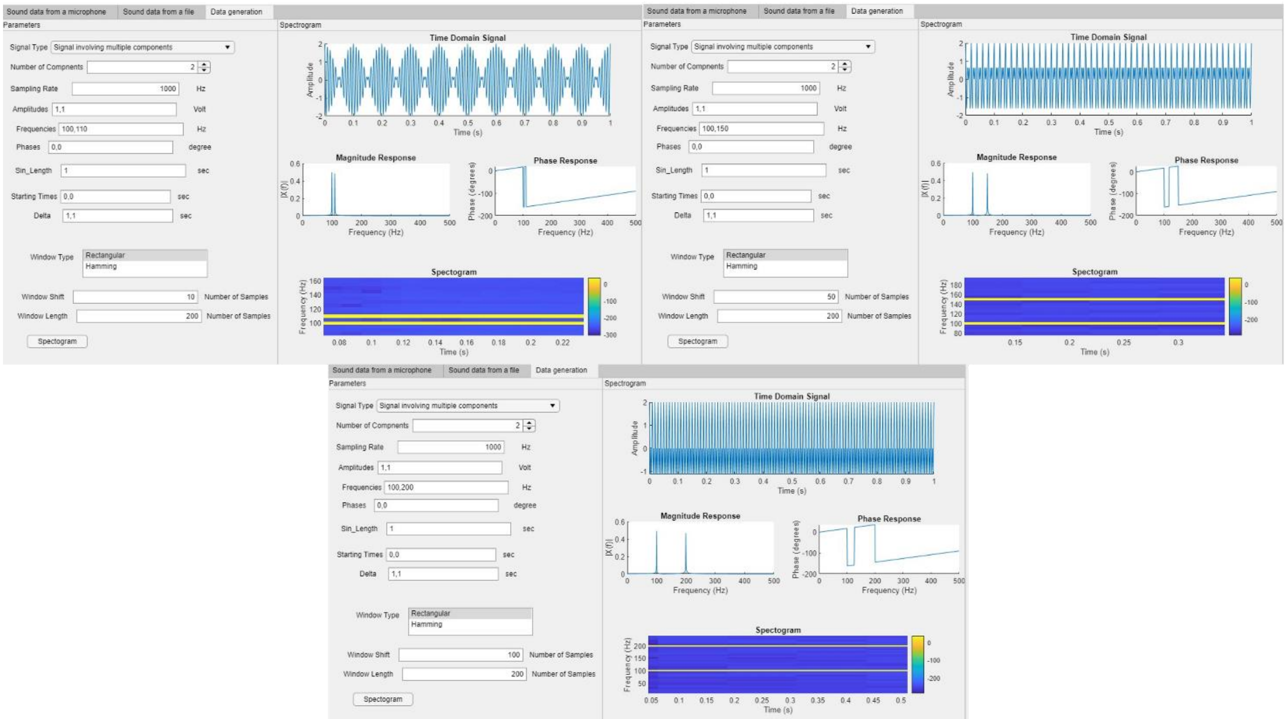


*Figure 12: Example of Time, Frequency Domain and STFT of Two Super Positioned Sine Waves*

### 4.2.3. iii.

The differences between diagonal frequencies:

- 1209Hz - 697Hz = 512Hz
- 1336Hz – 770 Hz = 566Hz
- 1477Hz - 852Hz = 625Hz
- 1633H – 941Hz = 692Hz

The differences between diagonal frequencies are not constant, which means the frequencies are not separated linearly.

Regarding the challenges in identification via Short-Time Fourier Transform (STFT), non-linear separation can pose a problem. The STFT operates on segments of the signal and is used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time. If the sampling frequency (fs) is not an integer multiple of the difference between these frequencies (as mentioned in part ii), there could be issues in accurately identifying the frequencies due to aliasing or spectral leakage.

### 4.2.4. iv.

To avoid the non-linearity problem mentioned in the earlier part, window shifts could be chosen in a way that it is the smallest frequency difference between the possible frequency pairs so that, number of samples of DTFT can cover the worst-case scenario. In this way, all the possible frequency pairs could be spotted on the spectrogram of the signal and DTMF signal could be decoded.

Other methods used or that can be used for decoding DTMF signals include:

- **Goertzel Algorithm:** This is a digital signal processing technique that provides a means for efficient evaluation of individual terms of the Discrete Fourier Transform (DFT), and is particularly well-suited for DTMF decoding due to its computational efficiency for detecting specific frequencies.
- **Zero Crossing Rate (ZCR):** This method can be used to estimate the frequency by counting the number of times the signal crosses the zero axis. It's computationally simple but less accurate than frequency-based methods.
- **Digital Filtering:** Implementing band-pass filters that isolate the DTMF frequency bands can be another approach. Each filter corresponds to one DTMF frequency, and the output of the filters can be used to detect the presence of the tones.

# 5. Conclusion

In summary, our project involved the development of an application designed to process sound data obtained from both microphones and file sources. Moreover, the application is equipped to generate three types of signals, including sinusoidal and windowed sinusoids and signal involving multiple components. It allows users to control essential signal parameters such as sampling rate, amplitude, frequency, phase, and the duration of the signal. Additionally, we developed a spectrogram function with adjustable window size, window shift, and window type. Through experimentation with these parameters, we were able to observe and understand the impact of each on the resulting spectrogram, thereby gaining valuable insights into signal processing.

# 6. Appendix

```
    function [S, T, F] = spec(~, signal, winSize, winShift, winType, samplingRate, signalDuration)
        noverlap=winSize-winShift;
        totalSamples = floor((signalDuration) * samplingRate + 1);

        % Create the window based on the specified type
        if strcmp(winType, "Rectangular")
            windowVec = transpose(rectwin(winSize));
        end
        if strcmp(winType, "Hamming")
            windowVec = transpose(hamming(winSize));
        end

        numWindows = floor((length(signal) - noverlap) / winShift);

        freqVectorLength = ceil((winSize + 1) / 2);

        T = 0:0.1:(totalSamples-1)/samplingRate;
        F = ((0:freqVectorLength-1)/winSize)*samplingRate;
        stftMatrix = zeros(numWindows, freqVectorLength);
        segmentStart = 1;
        for k = 1:numWindows
            segment = signal(segmentStart:segmentStart + winSize - 1);
            segmentFFT = fft(segment .* windowVec);
            stftMatrix(k, :) = segmentFFT(1:freqVectorLength);
            segmentStart = segmentStart + winShift;
        end

        S = transpose(stftMatrix);
    end
end
```

```
classdef dsp < matlab.apps.AppBase


    % Properties that correspond to app components
    properties (Access = public)
```

```
        UIFigure                          matlab.ui.Figure
        TabGroup                          matlab.ui.container.TabGroup
        SounddatafromamicrophoneTab       matlab.ui.container.Tab
        GridLayout                        matlab.ui.container.GridLayout
        ParametersPanel                   matlab.ui.container.Panel
        NumberofSamplesLabel              matlab.ui.control.Label
        WindowShiftEditField              matlab.ui.control.NumericEditField
        WindowShiftEditFieldLabel         matlab.ui.control.Label
        WindowTypeListBox_2               matlab.ui.control.ListBox
        WindowTypeListBox_2Label          matlab.ui.control.Label
        NumberofSamplesLabel_2            matlab.ui.control.Label
        WindowLengthEditField             matlab.ui.control.NumericEditField
        WindowLengthEditFieldLabel        matlab.ui.control.Label
        SpectogramButton                  matlab.ui.control.Button
        StartPlayButton                   matlab.ui.control.Button
        StopRecordButton                  matlab.ui.control.Button
        StartRecordButton                 matlab.ui.control.Button
        Label                             matlab.ui.control.Label
        StatusLabel                       matlab.ui.control.Label
        HzLabel                           matlab.ui.control.Label
        SamplingRateEditField             matlab.ui.control.NumericEditField
        SamplingRateEditFieldLabel        matlab.ui.control.Label
        SpectrogramPanel                  matlab.ui.container.Panel
        UIAxes4                           matlab.ui.control.UIAxes
        UIAxes                            matlab.ui.control.UIAxes
        SounddatafromafileTab             matlab.ui.container.Tab
        SpectrogramPanel_2                matlab.ui.container.Panel
        UIAxes5                           matlab.ui.control.UIAxes
        UIAxes2                           matlab.ui.control.UIAxes
        ParametersPanel_2                 matlab.ui.container.Panel
        WindowLengthEditField_2           matlab.ui.control.NumericEditField
        WindowLengthEditField_2Label      matlab.ui.control.Label
        WindowShiftEditField_2            matlab.ui.control.NumericEditField
        WindowShiftEditField_2Label       matlab.ui.control.Label
        WindowTypeListBox_3               matlab.ui.control.ListBox
        WindowTypeListBox_3Label          matlab.ui.control.Label
        NumberofSamplesLabel_4            matlab.ui.control.Label
        NumberofSamplesLabel_3            matlab.ui.control.Label
        SpectogramButton_2                matlab.ui.control.Button
        Label_3                           matlab.ui.control.Label
        StatusLabel_2                     matlab.ui.control.Label
        PlayButton                        matlab.ui.control.Button
        LoadButton                        matlab.ui.control.Button
        DatagenerationTab                 matlab.ui.container.Tab
        SpectrogramPanel_3                matlab.ui.container.Panel
        UIAxes8                           matlab.ui.control.UIAxes
        UIAxes7                           matlab.ui.control.UIAxes
        UIAxes6                           matlab.ui.control.UIAxes
        UIAxes3                           matlab.ui.control.UIAxes
        ParametersPanel_3                 matlab.ui.container.Panel
        DeltaEditField                    matlab.ui.control.EditField
        DeltaEditFieldLabel               matlab.ui.control.Label
        secLabel_3                        matlab.ui.control.Label
        NumberofSamplesLabel_6            matlab.ui.control.Label
```

```
        NumberofSamplesLabel_5          matlab.ui.control.Label
        WindowLengthEditField_3         matlab.ui.control.NumericEditField
        WindowLengthEditField_3Label    matlab.ui.control.Label
        WindowShiftEditField_3          matlab.ui.control.NumericEditField
        WindowShiftEditField_3Label     matlab.ui.control.Label
        SpectogramButton_3              matlab.ui.control.Button
        WindowTypeListBox               matlab.ui.control.ListBox
        WindowTypeListBoxLabel          matlab.ui.control.Label
        HzLabel_3                       matlab.ui.control.Label
        SamplingRateEditField_2         matlab.ui.control.NumericEditField
        SamplingRateEditField_2Label    matlab.ui.control.Label
        NumberofCompnentsSpinner        matlab.ui.control.Spinner
        NumberofCompnentsSpinnerLabel   matlab.ui.control.Label
        AmplitudesEditField             matlab.ui.control.EditField
        AmplitudesEditFieldLabel        matlab.ui.control.Label
        FrequenciesEditField            matlab.ui.control.EditField
        FrequenciesEditFieldLabel       matlab.ui.control.Label
        Sin_LengthEditField             matlab.ui.control.EditField
        Sin_LengthEditFieldLabel        matlab.ui.control.Label
        StartingTimesEditField          matlab.ui.control.EditField
        StartingTimesEditFieldLabel     matlab.ui.control.Label
        PhasesEditField                 matlab.ui.control.EditField
        PhasesEditFieldLabel            matlab.ui.control.Label
        secLabel_2                      matlab.ui.control.Label
        secLabel                        matlab.ui.control.Label
        degreeLabel                     matlab.ui.control.Label
        HzLabel_2                       matlab.ui.control.Label
        VoltLabel                       matlab.ui.control.Label
        SignalTypeDropDown              matlab.ui.control.DropDown
        SignalTypeDropDownLabel         matlab.ui.control.Label
    end



    properties (Access = private)
        Recorder; % Description
        loadedAudioFile; % Description
    end

    methods (Access = private)

        function [S, T, F] = spec(~, signal, winSize, winShift, winType,
samplingRate, signalDuration)
            noverlap=winSize-winShift;
            totalSamples = floor((signalDuration) * samplingRate + 1);

            % Create the window based on the specified type
            if strcmp(winType, "Rectangular")
                windowVec = transpose(rectwin(winSize));
            end
            if strcmp(winType, "Hamming")
                windowVec = transpose(hamming(winSize));
            end
```

```
            numWindows = floor((length(signal) - noverlap) / winShift);

            freqVectorLength = ceil((winSize + 1) / 2);

            T = 0:0.1:(totalSamples-1)/samplingRate;
            F = ((0:freqVectorLength-1)/winSize)*samplingRate;
            stftMatrix = zeros(numWindows, freqVectorLength);
            segmentStart = 1;
            for k = 1:numWindows
                segment = signal(segmentStart:segmentStart + winSize - 1);
                segmentFFT = fft(segment .* windowVec);
                stftMatrix(k, :) = segmentFFT(1:freqVectorLength);
                segmentStart = segmentStart + winShift;
            end

            S = transpose(stftMatrix);
        end
    end



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


        % Callback function
        function StartRecordingButtonValueChanged(app, event)

        end


        % Callback function
        function RecordButtonValueChanged(app, event)

        end


        % Button pushed function: StartRecordButton
        function StartRecordButtonPushed(app, event)
            samplingRate = app.SamplingRateEditField.Value;
            app.Recorder = audiorecorder(8000,8,1);
            record(app.Recorder);
            app.Label.Text = 'Recording...';
        end


        % Button pushed function: StopRecordButton
        function StopRecordButtonPushed(app, event)
            stop(app.Recorder);
            app.Label.Text = 'Recording stopped.';
        end


        % Button pushed function: StartPlayButton
```

```
        function StartPlayButtonPushed(app, event)
            recordedData = getaudiodata(app.Recorder);
            if ~isempty(recordedData)
                app.Label.Text = 'Playing...';
                player = audioplayer(recordedData, app.Recorder.SampleRate);
                playblocking(player); % Play the audio
            else
                app.Label.Text = 'No recorded data available.';
            end
        end


        % Button pushed function: LoadButton
        function LoadButtonPushed(app, event)
            [filename, pathname] = uigetfile({'*.wav';'*.mp3';}, 'Select an
Audio File');
            if ischar(filename)
                % An audio file was selected
                app.loadedAudioFile = fullfile(pathname, filename);
                app.Label_3.Text = sprintf('Loaded: %s', filename);
            else
                % No file was selected or the user canceled the operation
                app.StatusLabel.Text = 'No audio file selected.';
            end
        end


        % Button pushed function: PlayButton
        function PlayButtonPushed(app, event)
            [audioData, sampleRate] = audioread(app.loadedAudioFile);
            player = audioplayer(audioData, sampleRate);
            app.Label_3.Text = 'Playing...';
            playblocking(player); % Play the audio and block further actions
until playback is finished

        end


        % Value changed function: SignalTypeDropDown
        function SignalTypeDropDownValueChanged(app, event)
            value = app.SignalTypeDropDown.Value;
            switch value
                case 'Sinusoidal signal'
                    app.NumberofCompnentsSpinner.Enable = 'off';
                    app.NumberofCompnentsSpinner.Value = 1;


                case 'Windowed sinusoidal'
                    app.NumberofCompnentsSpinner.Enable = 'off';
                    app.NumberofCompnentsSpinner.Value = 1;


                case 'Signal involving multiple components'
                    app.NumberofCompnentsSpinner.Enable = 'on';
```

```
            end
        end


        % Button pushed function: SpectogramButton
        function SpectogramButtonPushed(app, event)
            clear app.UIAxes4;
            clear app.UIAxes;
            recordedData = getaudiodata(app.Recorder);
            samplingRate = app.SamplingRateEditField.Value;
            n = length(recordedData)-1;
            max_time = n / samplingRate;
            t = 0:(1/samplingRate):max_time;
            plot(app.UIAxes,t,recordedData);
            app.Label_3.Text = 'Plotting';

            windowSize = app.WindowLengthEditField.Value;
            windowShift = app.WindowShiftEditField.Value;
            windowType = app.WindowTypeListBox_2.Value;

[S,T,F]=spec(app,recordedData,windowSize,windowShift,windowType,samplingRate,m
ax_time);


            axes(app.UIAxes4);
            imagesc(app.UIAxes4,T, F, (20*log10(abs(S))))
            axis(app.UIAxes4,'xy')

            xlim(app.UIAxes4,[min(T),max(T)]);
            ylim(app.UIAxes4,[min(F),max(F)]);
            colorbar(app.UIAxes4);

        end


        % Button pushed function: SpectogramButton_2
        function SpectogramButton_2Pushed(app, event)
            clear app.UIAxes2;
            clear app.UIAxes5;
            [recordedData, sampleRate] = audioread(app.loadedAudioFile);
            n = length(recordedData)-1;
            max_time = n / sampleRate;
            t = 0:(1/sampleRate):max_time;
            plot(app.UIAxes2,t,recordedData);
            app.Label_3.Text = 'Plotting';

            windowSize = app.WindowLengthEditField_2.Value;
            windowShift = app.WindowShiftEditField_2.Value;
            windowType = app.WindowTypeListBox_3.Value;
```

Yunus Emre Tüysüz-2444032 12.5.2023
Ata Oğuz Tanrıkulu 2443927

```
[S,T,F]=spec(app,recordedData,windowSize,windowShift,windowType,sampleRate,max
_time);

            axes(app.UIAxes5);
            imagesc(app.UIAxes5,T, F, 20*log10(abs(S)))
            axis(app.UIAxes5,'xy')
            xlim(app.UIAxes5,[min(T),max(T)]);
            ylim(app.UIAxes5,[min(F),max(F)]);

            colorbar(app.UIAxes5);
        end


        % Button pushed function: SpectogramButton_3
        function SpectogramButton_3Pushed(app, event)
            clear app.UIAxes3;
            clear app.UIAxes6;

            sampleRate = app.SamplingRateEditField_2.Value;
            switch app.SignalTypeDropDown.Value
                case 'Sinusoidal signal'

                    amplitude = str2double(app.AmplitudesEditField.Value);
                    freq = str2double(app.FrequenciesEditField.Value);
                    phase = str2double(app.PhasesEditField.Value)/180*pi;
                    len = str2double(app.Sin_LengthEditField.Value);

                    t = 0:(1/sampleRate):len;
                    signal = amplitude * cos(2*pi*freq*t+phase);
                case 'Windowed sinusoidal'
                     amplitude = str2double(app.AmplitudesEditField.Value);
                     freq = str2double(app.FrequenciesEditField.Value);
                     phase = str2double(app.PhasesEditField.Value)/180*pi;
                     len = str2double(app.Sin_LengthEditField.Value);
                     delt = str2double(app.DeltaEditField.Value);

                     switch app.WindowTypeListBox.Value
                         case 'Rectangular'
                             Window = rectwin(len*sampleRate+1);
                         case 'Hamming'
                             Window = hamming(len*sampleRate+1);
                     end

                     t0 = str2double(app.StartingTimesEditField.Value);

                     t = 0:(1/sampleRate):(len+delt); % Adjust the time vector
to match the duration of the signal
                     signal = zeros(1,length(t));

                     for i = 1:length(t) % Start the loop from 1
                         if t(i) >= t0 && t(i) <= t0+delt
                             windowIndex = round((t(i)-t0)*sampleRate) + 1; %
Compute the index for the window
```

```
                            signal(i) = Window(windowIndex) * amplitude *
cos(2 * pi * freq * (t(i)-t0) + phase);
                        end
                    end
                case 'Signal involving multiple components'
                    number_of_signals = app.NumberofCompnentsSpinner.Value;
                    amplitudes =
str2double(strsplit(app.AmplitudesEditField.Value,','));
                    freqs =
str2double(strsplit(app.FrequenciesEditField.Value,','));
                    phases =
str2double(strsplit(app.PhasesEditField.Value,',')) /180 * pi;
                    len = str2double(app.Sin_LengthEditField.Value);
                    start_times =
str2double(strsplit(app.StartingTimesEditField.Value,','));
                    deltas =
str2double(strsplit(app.DeltaEditField.Value,','));
                    t = 0:(1/sampleRate):len;
                    signal = zeros(1,length(t));
                    for i = 1:number_of_signals
                        for j = 1:length(t)
                            if t(j) >= start_times(i) && t(j) <=
start_times(i) + deltas(i)
                                signal(j) = signal(j) + amplitudes(i) *
cos(2*pi*freqs(i) * t(j) + phases(i));
                            end
                        end

                    end
            end

            plot(app.UIAxes3,t,signal);

            N = length(signal);
            signalFFT = fft(signal);

            f = sampleRate*(0:(N/2))/N;

            magnitudeSpectrum = abs(signalFFT/N);
            magnitudeSpectrum = magnitudeSpectrum(1:N/2+1);

            phaseSpectrum = angle(signalFFT);
            phaseSpectrum = phaseSpectrum(1:N/2+1);

            plot(app.UIAxes7,f, magnitudeSpectrum)


            plot(app.UIAxes8,f, phaseSpectrum * 180/pi) % Converting to
degrees


            windowSize = app.WindowLengthEditField_3.Value;
            windowShift = app.WindowShiftEditField_3.Value;
            windowType = app.WindowTypeListBox.Value;
```

```
[S,T,F]=spec(app,signal,windowSize,windowShift,windowType,sampleRate,len);

            axes(app.UIAxes6);
            imagesc(app.UIAxes6,T, F, 20*log10(abs(S)))
            axis(app.UIAxes6,'xy')
            colorbar(app.UIAxes6);
            xlim(app.UIAxes6,[min(T),max(T)]);
            ylim(app.UIAxes6,[min(F),max(F)]);
        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 = [100 100 718 596];
            app.UIFigure.Name = 'MATLAB App';


            % Create TabGroup
            app.TabGroup = uitabgroup(app.UIFigure);
            app.TabGroup.Position = [1 1 752 596];


            % Create SounddatafromamicrophoneTab
            app.SounddatafromamicrophoneTab = uitab(app.TabGroup);
            app.SounddatafromamicrophoneTab.Title = 'Sound data from a
microphone';


            % Create GridLayout
            app.GridLayout = uigridlayout(app.SounddatafromamicrophoneTab);
            app.GridLayout.ColumnWidth = {'2.96x', '1.93x', '1x'};
            app.GridLayout.RowHeight = {73.33, 423.33, '1x'};
            app.GridLayout.ColumnSpacing = 0;
            app.GridLayout.RowSpacing = 0;
            app.GridLayout.Padding = [0 0 0 0];


            % Create SpectrogramPanel
            app.SpectrogramPanel = uipanel(app.GridLayout);
            app.SpectrogramPanel.Title = 'Spectrogram';
            app.SpectrogramPanel.Layout.Row = [1 3];
            app.SpectrogramPanel.Layout.Column = [2 3];
```

```
            % Create UIAxes
            app.UIAxes = uiaxes(app.SpectrogramPanel);
            title(app.UIAxes, 'Time Domain Signal')
            xlabel(app.UIAxes, 'Time (sec)')
            ylabel(app.UIAxes, 'Amplitude')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Position = [24 336 300 185];


            % Create UIAxes4
            app.UIAxes4 = uiaxes(app.SpectrogramPanel);
            title(app.UIAxes4, 'Spectogram')
            xlabel(app.UIAxes4, 'Time (sec)')
            ylabel(app.UIAxes4, 'Frequency (Hz)')
            zlabel(app.UIAxes4, 'Z')
            app.UIAxes4.Position = [24 63 300 185];


            % Create ParametersPanel
            app.ParametersPanel = uipanel(app.GridLayout);
            app.ParametersPanel.Title = 'Parameters';
            app.ParametersPanel.Layout.Row = [1 3];
            app.ParametersPanel.Layout.Column = 1;


            % Create SamplingRateEditFieldLabel
            app.SamplingRateEditFieldLabel = uilabel(app.ParametersPanel);
            app.SamplingRateEditFieldLabel.HorizontalAlignment = 'right';
            app.SamplingRateEditFieldLabel.Position = [9 492 84 22];
            app.SamplingRateEditFieldLabel.Text = 'Sampling Rate';


            % Create SamplingRateEditField
            app.SamplingRateEditField = uieditfield(app.ParametersPanel,
'numeric');
            app.SamplingRateEditField.Position = [108 492 172 22];
            app.SamplingRateEditField.Value = 8000;


            % Create HzLabel
            app.HzLabel = uilabel(app.ParametersPanel);
            app.HzLabel.Position = [301 492 25 22];
            app.HzLabel.Text = 'Hz';


            % Create StatusLabel
            app.StatusLabel = uilabel(app.ParametersPanel);
            app.StatusLabel.Position = [28 315 46 22];
            app.StatusLabel.Text = 'Status: ';


            % Create Label
            app.Label = uilabel(app.ParametersPanel);
            app.Label.Position = [73 315 279 22];
```

```
            app.Label.Text = ' ';


            % Create StartRecordButton
            app.StartRecordButton = uibutton(app.ParametersPanel, 'push');
            app.StartRecordButton.ButtonPushedFcn = createCallbackFcn(app,
@StartRecordButtonPushed, true);
            app.StartRecordButton.Position = [32 417 100 22];
            app.StartRecordButton.Text = 'Start Record';


            % Create StopRecordButton
            app.StopRecordButton = uibutton(app.ParametersPanel, 'push');
            app.StopRecordButton.ButtonPushedFcn = createCallbackFcn(app,
@StopRecordButtonPushed, true);
            app.StopRecordButton.Position = [226 417 100 22];
            app.StopRecordButton.Text = 'Stop Record';


            % Create StartPlayButton
            app.StartPlayButton = uibutton(app.ParametersPanel, 'push');
            app.StartPlayButton.ButtonPushedFcn = createCallbackFcn(app,
@StartPlayButtonPushed, true);
            app.StartPlayButton.Position = [32 368 100 22];
            app.StartPlayButton.Text = 'Start Play';


            % Create SpectogramButton
            app.SpectogramButton = uibutton(app.ParametersPanel, 'push');
            app.SpectogramButton.ButtonPushedFcn = createCallbackFcn(app,
@SpectogramButtonPushed, true);
            app.SpectogramButton.Position = [32 72 100 22];
            app.SpectogramButton.Text = 'Spectogram';


            % Create WindowLengthEditFieldLabel
            app.WindowLengthEditFieldLabel = uilabel(app.ParametersPanel);
            app.WindowLengthEditFieldLabel.HorizontalAlignment = 'right';
            app.WindowLengthEditFieldLabel.Position = [25 215 88 22];
            app.WindowLengthEditFieldLabel.Text = 'Window Length';


            % Create WindowLengthEditField
            app.WindowLengthEditField = uieditfield(app.ParametersPanel,
'numeric');
            app.WindowLengthEditField.Position = [120 215 100 22];


            % Create NumberofSamplesLabel_2
            app.NumberofSamplesLabel_2 = uilabel(app.ParametersPanel);
            app.NumberofSamplesLabel_2.Position = [242 215 112 22];
            app.NumberofSamplesLabel_2.Text = 'Number of Samples';
```

```
            % Create WindowTypeListBox_2Label
            app.WindowTypeListBox_2Label = uilabel(app.ParametersPanel);
            app.WindowTypeListBox_2Label.HorizontalAlignment = 'right';
            app.WindowTypeListBox_2Label.Position = [31 166 77 22];
            app.WindowTypeListBox_2Label.Text = 'Window Type';

            % Create WindowTypeListBox_2
            app.WindowTypeListBox_2 = uilistbox(app.ParametersPanel);
            app.WindowTypeListBox_2.Items = {'Rectangular', 'Hamming'};
            app.WindowTypeListBox_2.Position = [131 118 100 74];
            app.WindowTypeListBox_2.Value = 'Rectangular';

            % Create WindowShiftEditFieldLabel
            app.WindowShiftEditFieldLabel = uilabel(app.ParametersPanel);
            app.WindowShiftEditFieldLabel.HorizontalAlignment = 'right';
            app.WindowShiftEditFieldLabel.Position = [28 254 76 22];
            app.WindowShiftEditFieldLabel.Text = 'Window Shift';

            % Create WindowShiftEditField
            app.WindowShiftEditField = uieditfield(app.ParametersPanel,
'numeric');
            app.WindowShiftEditField.Position = [119 254 100 22];

            % Create NumberofSamplesLabel
            app.NumberofSamplesLabel = uilabel(app.ParametersPanel);
            app.NumberofSamplesLabel.Position = [242 254 112 22];
            app.NumberofSamplesLabel.Text = 'Number of Samples';

            % Create SounddatafromafileTab
            app.SounddatafromafileTab = uitab(app.TabGroup);
            app.SounddatafromafileTab.Title = 'Sound data from a file';

            % Create ParametersPanel_2
            app.ParametersPanel_2 = uipanel(app.SounddatafromafileTab);
            app.ParametersPanel_2.Title = 'Parameters';
            app.ParametersPanel_2.Position = [1 0 333 572];

            % Create LoadButton
            app.LoadButton = uibutton(app.ParametersPanel_2, 'push');
            app.LoadButton.ButtonPushedFcn = createCallbackFcn(app,
@LoadButtonPushed, true);
            app.LoadButton.Position = [28 439 100 22];
            app.LoadButton.Text = 'Load';

            % Create PlayButton
            app.PlayButton = uibutton(app.ParametersPanel_2, 'push');
```

```
            app.PlayButton.ButtonPushedFcn = createCallbackFcn(app,
@PlayButtonPushed, true);
            app.PlayButton.Position = [202 439 100 22];
            app.PlayButton.Text = 'Play';

            % Create StatusLabel_2
            app.StatusLabel_2 = uilabel(app.ParametersPanel_2);
            app.StatusLabel_2.Position = [37 378 43 22];
            app.StatusLabel_2.Text = 'Status:';

            % Create Label_3
            app.Label_3 = uilabel(app.ParametersPanel_2);
            app.Label_3.Position = [92 378 260 22];
            app.Label_3.Text = ' ';

            % Create SpectogramButton_2
            app.SpectogramButton_2 = uibutton(app.ParametersPanel_2, 'push');
            app.SpectogramButton_2.ButtonPushedFcn = createCallbackFcn(app,
@SpectogramButton_2Pushed, true);
            app.SpectogramButton_2.Position = [43 64 100 22];
            app.SpectogramButton_2.Text = 'Spectogram';

            % Create NumberofSamplesLabel_3
            app.NumberofSamplesLabel_3 = uilabel(app.ParametersPanel_2);
            app.NumberofSamplesLabel_3.Position = [218 226 112 22];
            app.NumberofSamplesLabel_3.Text = 'Number of Samples';

            % Create NumberofSamplesLabel_4
            app.NumberofSamplesLabel_4 = uilabel(app.ParametersPanel_2);
            app.NumberofSamplesLabel_4.Position = [214 266 112 22];
            app.NumberofSamplesLabel_4.Text = 'Number of Samples';

            % Create WindowTypeListBox_3Label
            app.WindowTypeListBox_3Label = uilabel(app.ParametersPanel_2);
            app.WindowTypeListBox_3Label.HorizontalAlignment = 'right';
            app.WindowTypeListBox_3Label.Position = [51 153 77 22];
            app.WindowTypeListBox_3Label.Text = 'Window Type';

            % Create WindowTypeListBox_3
            app.WindowTypeListBox_3 = uilistbox(app.ParametersPanel_2);
            app.WindowTypeListBox_3.Items = {'Rectangular', 'Hamming'};
            app.WindowTypeListBox_3.Position = [151 105 100 74];
            app.WindowTypeListBox_3.Value = 'Rectangular';

            % Create WindowShiftEditField_2Label
            app.WindowShiftEditField_2Label = uilabel(app.ParametersPanel_2);
```

```
            app.WindowShiftEditField_2Label.HorizontalAlignment = 'right';
            app.WindowShiftEditField_2Label.Position = [10 266 76 22];
            app.WindowShiftEditField_2Label.Text = 'Window Shift';

            % Create WindowShiftEditField_2
            app.WindowShiftEditField_2 = uieditfield(app.ParametersPanel_2,
'numeric');
            app.WindowShiftEditField_2.Position = [101 266 100 22];

            % Create WindowLengthEditField_2Label
            app.WindowLengthEditField_2Label = uilabel(app.ParametersPanel_2);
            app.WindowLengthEditField_2Label.HorizontalAlignment = 'right';
            app.WindowLengthEditField_2Label.Position = [1 227 88 22];
            app.WindowLengthEditField_2Label.Text = 'Window Length';

            % Create WindowLengthEditField_2
            app.WindowLengthEditField_2 = uieditfield(app.ParametersPanel_2,
'numeric');
            app.WindowLengthEditField_2.Position = [104 227 100 22];

            % Create SpectrogramPanel_2
            app.SpectrogramPanel_2 = uipanel(app.SounddatafromafileTab);
            app.SpectrogramPanel_2.Title = 'Spectrogram';
            app.SpectrogramPanel_2.Position = [333 0 418 572];

            % Create UIAxes2
            app.UIAxes2 = uiaxes(app.SpectrogramPanel_2);
            title(app.UIAxes2, 'Time Domain Signal')
            xlabel(app.UIAxes2, 'Time (s)')
            ylabel(app.UIAxes2, 'Amplitude')
            zlabel(app.UIAxes2, 'Z')
            app.UIAxes2.Position = [57 330 300 185];

            % Create UIAxes5
            app.UIAxes5 = uiaxes(app.SpectrogramPanel_2);
            title(app.UIAxes5, 'Spectogram')
            xlabel(app.UIAxes5, 'Time (s)')
            ylabel(app.UIAxes5, 'Frequency (Hz)')
            zlabel(app.UIAxes5, 'Z')
            app.UIAxes5.Position = [57 82 300 185];

            % Create DatagenerationTab
            app.DatagenerationTab = uitab(app.TabGroup);
            app.DatagenerationTab.Title = 'Data generation';

            % Create ParametersPanel_3
```

```
            app.ParametersPanel_3 = uipanel(app.DatagenerationTab);
            app.ParametersPanel_3.Title = 'Parameters';
            app.ParametersPanel_3.Position = [0 1 352 571];


            % Create SignalTypeDropDownLabel
            app.SignalTypeDropDownLabel = uilabel(app.ParametersPanel_3);
            app.SignalTypeDropDownLabel.HorizontalAlignment = 'right';
            app.SignalTypeDropDownLabel.Position = [10 513 68 22];
            app.SignalTypeDropDownLabel.Text = 'Signal Type';


            % Create SignalTypeDropDown
            app.SignalTypeDropDown = uidropdown(app.ParametersPanel_3);
            app.SignalTypeDropDown.Items = {'Sinusoidal signal', 'Windowed
sinusoidal', 'Signal involving multiple components'};
            app.SignalTypeDropDown.ValueChangedFcn = createCallbackFcn(app,
@SignalTypeDropDownValueChanged, true);
            app.SignalTypeDropDown.Position = [83 513 198 22];
            app.SignalTypeDropDown.Value = 'Sinusoidal signal';


            % Create VoltLabel
            app.VoltLabel = uilabel(app.ParametersPanel_3);
            app.VoltLabel.Position = [213 410 26 22];
            app.VoltLabel.Text = 'Volt';


            % Create HzLabel_2
            app.HzLabel_2 = uilabel(app.ParametersPanel_3);
            app.HzLabel_2.Position = [219 377 25 22];
            app.HzLabel_2.Text = 'Hz';


            % Create degreeLabel
            app.degreeLabel = uilabel(app.ParametersPanel_3);
            app.degreeLabel.Position = [205 347 43 22];
            app.degreeLabel.Text = 'degree';


            % Create secLabel
            app.secLabel = uilabel(app.ParametersPanel_3);
            app.secLabel.Position = [187 265 25 22];
            app.secLabel.Text = 'sec';


            % Create secLabel_2
            app.secLabel_2 = uilabel(app.ParametersPanel_3);
            app.secLabel_2.Position = [220 308 25 22];
            app.secLabel_2.Text = 'sec';


            % Create PhasesEditFieldLabel
            app.PhasesEditFieldLabel = uilabel(app.ParametersPanel_3);
```

```
            app.PhasesEditFieldLabel.HorizontalAlignment = 'right';
            app.PhasesEditFieldLabel.Position = [20 347 46 22];
            app.PhasesEditFieldLabel.Text = 'Phases';

            % Create PhasesEditField
            app.PhasesEditField = uieditfield(app.ParametersPanel_3, 'text');
            app.PhasesEditField.Position = [79 347 100 22];
            app.PhasesEditField.Value = '0';

            % Create StartingTimesEditFieldLabel
            app.StartingTimesEditFieldLabel = uilabel(app.ParametersPanel_3);
            app.StartingTimesEditFieldLabel.HorizontalAlignment = 'right';
            app.StartingTimesEditFieldLabel.Position = [12 265 82 22];
            app.StartingTimesEditFieldLabel.Text = 'Starting Times';

            % Create StartingTimesEditField
            app.StartingTimesEditField = uieditfield(app.ParametersPanel_3,
'text');
            app.StartingTimesEditField.Position = [99 265 76 22];
            app.StartingTimesEditField.Value = '0';

            % Create Sin_LengthEditFieldLabel
            app.Sin_LengthEditFieldLabel = uilabel(app.ParametersPanel_3);
            app.Sin_LengthEditFieldLabel.HorizontalAlignment = 'right';
            app.Sin_LengthEditFieldLabel.Position = [19 308 66 22];
            app.Sin_LengthEditFieldLabel.Text = 'Sin_Length';

            % Create Sin_LengthEditField
            app.Sin_LengthEditField = uieditfield(app.ParametersPanel_3,
'text');
            app.Sin_LengthEditField.Position = [100 308 100 22];
            app.Sin_LengthEditField.Value = '3';

            % Create FrequenciesEditFieldLabel
            app.FrequenciesEditFieldLabel = uilabel(app.ParametersPanel_3);
            app.FrequenciesEditFieldLabel.HorizontalAlignment = 'right';
            app.FrequenciesEditFieldLabel.Position = [20 377 72 22];
            app.FrequenciesEditFieldLabel.Text = 'Frequencies';

            % Create FrequenciesEditField
            app.FrequenciesEditField = uieditfield(app.ParametersPanel_3,
'text');
            app.FrequenciesEditField.Position = [96 377 100 22];
            app.FrequenciesEditField.Value = '1';

            % Create AmplitudesEditFieldLabel
```

```
            app.AmplitudesEditFieldLabel = uilabel(app.ParametersPanel_3);
            app.AmplitudesEditFieldLabel.HorizontalAlignment = 'right';
            app.AmplitudesEditFieldLabel.Position = [14 410 65 22];
            app.AmplitudesEditFieldLabel.Text = 'Amplitudes';

            % Create AmplitudesEditField
            app.AmplitudesEditField = uieditfield(app.ParametersPanel_3,
'text');
            app.AmplitudesEditField.Position = [85 410 100 22];
            app.AmplitudesEditField.Value = '1';

            % Create NumberofCompnentsSpinnerLabel
            app.NumberofCompnentsSpinnerLabel =
uilabel(app.ParametersPanel_3);
            app.NumberofCompnentsSpinnerLabel.HorizontalAlignment = 'right';
            app.NumberofCompnentsSpinnerLabel.Position = [11 480 126 22];
            app.NumberofCompnentsSpinnerLabel.Text = 'Number of Compnents';

            % Create NumberofCompnentsSpinner
            app.NumberofCompnentsSpinner = uispinner(app.ParametersPanel_3);
            app.NumberofCompnentsSpinner.Enable = 'off';
            app.NumberofCompnentsSpinner.Position = [143 480 100 22];
            app.NumberofCompnentsSpinner.Value = 1;

            % Create SamplingRateEditField_2Label
            app.SamplingRateEditField_2Label = uilabel(app.ParametersPanel_3);
            app.SamplingRateEditField_2Label.HorizontalAlignment = 'right';
            app.SamplingRateEditField_2Label.Position = [11 444 84 22];
            app.SamplingRateEditField_2Label.Text = 'Sampling Rate';

            % Create SamplingRateEditField_2
            app.SamplingRateEditField_2 = uieditfield(app.ParametersPanel_3,
'numeric');
            app.SamplingRateEditField_2.Position = [118 444 71 22];
            app.SamplingRateEditField_2.Value = 50;

            % Create HzLabel_3
            app.HzLabel_3 = uilabel(app.ParametersPanel_3);
            app.HzLabel_3.Position = [218 444 25 22];
            app.HzLabel_3.Text = 'Hz';

            % Create WindowTypeListBoxLabel
            app.WindowTypeListBoxLabel = uilabel(app.ParametersPanel_3);
            app.WindowTypeListBoxLabel.HorizontalAlignment = 'right';
            app.WindowTypeListBoxLabel.Position = [44 164 77 22];
            app.WindowTypeListBoxLabel.Text = 'Window Type';
```

```
            % Create WindowTypeListBox
            app.WindowTypeListBox = uilistbox(app.ParametersPanel_3);
            app.WindowTypeListBox.Items = {'Rectangular', 'Hamming'};
            app.WindowTypeListBox.Position = [136 147 100 41];
            app.WindowTypeListBox.Value = 'Rectangular';

            % Create SpectogramButton_3
            app.SpectogramButton_3 = uibutton(app.ParametersPanel_3, 'push');
            app.SpectogramButton_3.ButtonPushedFcn = createCallbackFcn(app,
@SpectogramButton_3Pushed, true);
            app.SpectogramButton_3.Position = [44 30 100 22];
            app.SpectogramButton_3.Text = 'Spectogram';

            % Create WindowShiftEditField_3Label
            app.WindowShiftEditField_3Label = uilabel(app.ParametersPanel_3);
            app.WindowShiftEditField_3Label.HorizontalAlignment = 'right';
            app.WindowShiftEditField_3Label.Position = [29 104 76 22];
            app.WindowShiftEditField_3Label.Text = 'Window Shift';

            % Create WindowShiftEditField_3
            app.WindowShiftEditField_3 = uieditfield(app.ParametersPanel_3,
'numeric');
            app.WindowShiftEditField_3.Position = [120 104 100 22];
            app.WindowShiftEditField_3.Value = 10;

            % Create WindowLengthEditField_3Label
            app.WindowLengthEditField_3Label = uilabel(app.ParametersPanel_3);
            app.WindowLengthEditField_3Label.HorizontalAlignment = 'right';
            app.WindowLengthEditField_3Label.Position = [23 72 88 22];
            app.WindowLengthEditField_3Label.Text = 'Window Length';

            % Create WindowLengthEditField_3
            app.WindowLengthEditField_3 = uieditfield(app.ParametersPanel_3,
'numeric');
            app.WindowLengthEditField_3.Position = [126 72 100 22];
            app.WindowLengthEditField_3.Value = 10;

            % Create NumberofSamplesLabel_5
            app.NumberofSamplesLabel_5 = uilabel(app.ParametersPanel_3);
            app.NumberofSamplesLabel_5.Position = [231 104 112 22];
            app.NumberofSamplesLabel_5.Text = 'Number of Samples';

            % Create NumberofSamplesLabel_6
            app.NumberofSamplesLabel_6 = uilabel(app.ParametersPanel_3);
            app.NumberofSamplesLabel_6.Position = [235 72 112 22];
            app.NumberofSamplesLabel_6.Text = 'Number of Samples';
```

```
            % Create secLabel_3
            app.secLabel_3 = uilabel(app.ParametersPanel_3);
            app.secLabel_3.Position = [191 233 25 22];
            app.secLabel_3.Text = 'sec';

            % Create DeltaEditFieldLabel
            app.DeltaEditFieldLabel = uilabel(app.ParametersPanel_3);
            app.DeltaEditFieldLabel.HorizontalAlignment = 'right';
            app.DeltaEditFieldLabel.Position = [50 233 33 22];
            app.DeltaEditFieldLabel.Text = 'Delta';

            % Create DeltaEditField
            app.DeltaEditField = uieditfield(app.ParametersPanel_3, 'text');
            app.DeltaEditField.Position = [103 233 76 22];
            app.DeltaEditField.Value = '3';

            % Create SpectrogramPanel_3
            app.SpectrogramPanel_3 = uipanel(app.DatagenerationTab);
            app.SpectrogramPanel_3.Title = 'Spectrogram';
            app.SpectrogramPanel_3.Position = [352 0 399 572];

            % Create UIAxes3
            app.UIAxes3 = uiaxes(app.SpectrogramPanel_3);
            title(app.UIAxes3, 'Time Domain Signal')
            xlabel(app.UIAxes3, 'Time (s)')
            ylabel(app.UIAxes3, 'Amplitude')
            zlabel(app.UIAxes3, 'Z')
            app.UIAxes3.Position = [45 378 310 172];

            % Create UIAxes6
            app.UIAxes6 = uiaxes(app.SpectrogramPanel_3);
            title(app.UIAxes6, 'Spectogram')
            xlabel(app.UIAxes6, 'Time (s)')
            ylabel(app.UIAxes6, 'Frequency (Hz)')
            zlabel(app.UIAxes6, 'Z')
            app.UIAxes6.Position = [33 17 333 148];

            % Create UIAxes7
            app.UIAxes7 = uiaxes(app.SpectrogramPanel_3);
            title(app.UIAxes7, 'Magnitude Response')
            xlabel(app.UIAxes7, 'Frequency (Hz)')
            ylabel(app.UIAxes7, '|X(f)|')
            zlabel(app.UIAxes7, 'Z')
            app.UIAxes7.Position = [2 214 191 140];

            % Create UIAxes8
            app.UIAxes8 = uiaxes(app.SpectrogramPanel_3);
```

```
            title(app.UIAxes8, 'Phase Response')
            xlabel(app.UIAxes8, 'Frequency (Hz)')
            ylabel(app.UIAxes8, 'Phase (degrees)')
            zlabel(app.UIAxes8, 'Z')
            app.UIAxes8.Position = [210 214 189 135];


            % 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 = dsp


            % 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
```