C0E 472: DIGITAL SIGNAL PROCESSING

Lab 2:

AM and FM Sinusoidal Signals

INDEX NUMBER: 3050420

NAME: HARDY HAWA TUNTEIYA

COURSE: BSc. Biomedical Engineering

Year 4

1. INTRODUCTION

This lab focused on synthesizing some AM and FM signals. Later on, a spectrogram was used in order to verify that they have the correct frequency content.

2. AMPLITUDE MODULATION (AM) SYNTHESIS - BEAT NOTES

a. An M-file called beat.m was created to implement the signal:

$$x(t) = A\cos(2\pi(f_c - f_\Delta)t) + B\cos(2\pi(f_c + f_\Delta)t)$$

The purpose of the beat.m is to add two signals with slightly different frequencies. Below is its code:

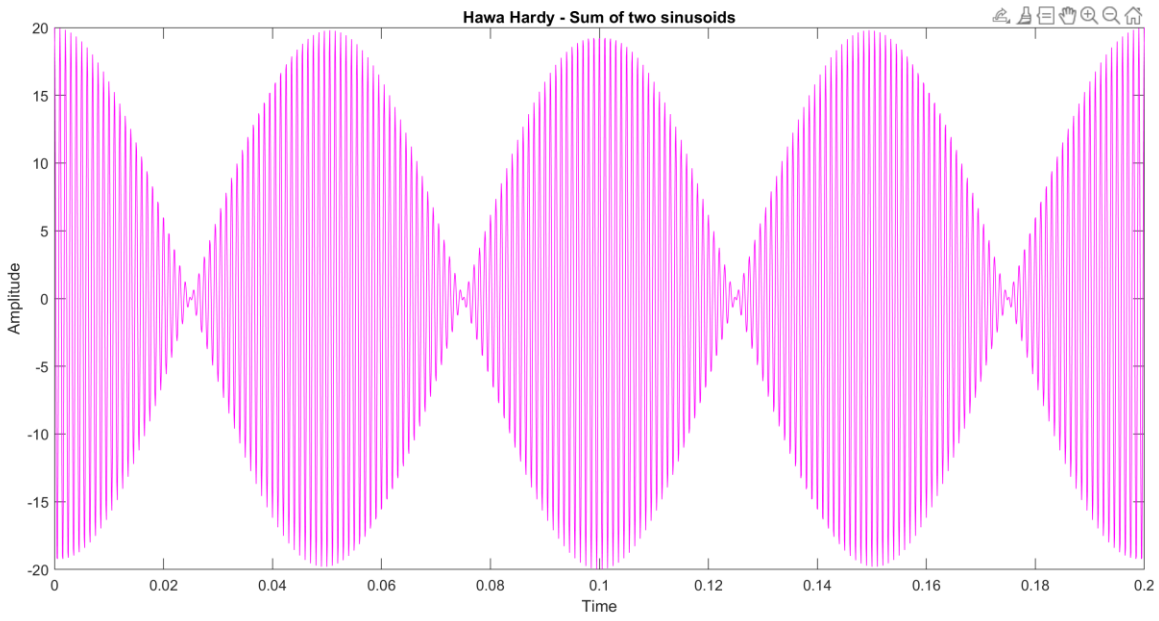
```
function [xx, tt] = beat(A, B, fc, delf, fsamp, dur)
%BEAT compute samples of the sum of two cosine waves
% usage:
% [xx, tt] = beat(A, B, fc, delf, fsamp, dur)
%
% A = amplitude of lower frequency cosine
% B = amplitude of higher frequency cosine
% fc = center frequency
% delf = frequency difference
% fsamp = sampling rate
% dur = total time duration in seconds
% xx = output vector of samples
%--Second Output:
% tt = time vector corresponding to xx

    tt = 0 : 1/fsamp : dur;
    xx = A*cos(2*pi*tt*(fc - delf))+ B*cos(2*pi*tt*(fc + delf));
end
```

b. The code below was written to test the beat.m file generated above)

```
%Compute samples of sum of the two cosine waves.
%A = Amplitude of the lower frequency cosine
%B = Amplitude of higher frequency cosine
%fc = center frequency
%delf = frequency difference
```

```
%fsamp = sampling frequency
%dur = total time duration in seconds
%xx = outputs vector of samples
% == second output
%tt == time vector of samples


A = 10;
B = 10;
fc = 1000;
delf = 10;
fsamp = 11025;
dur = 1;
[xx, tt] = beat(A, B, fc, delf, fsamp, dur);
soundsc(xx, fsamp);
dur = 0.2;
[xx, tt] = beat(A, B, fc, delf, fsamp, dur);
plot(tt, xx, 'm');
xlabel('Time');
ylabel('Amplitude');
```

OUTPUT



Description of the Waveform

The envelope, which is the outer shape of the waveform, appears as a slowly varying sinusoidal pattern, is referred to as the "envelope."

Within the envelope, there is a rapidly oscillating high-frequency signal. The amplitude of the envelope is 20 when measured on the graph. This corresponds to how theoretically; the amplitude of an envelope is determined by the sum of the amplitudes of the original sinusoids.

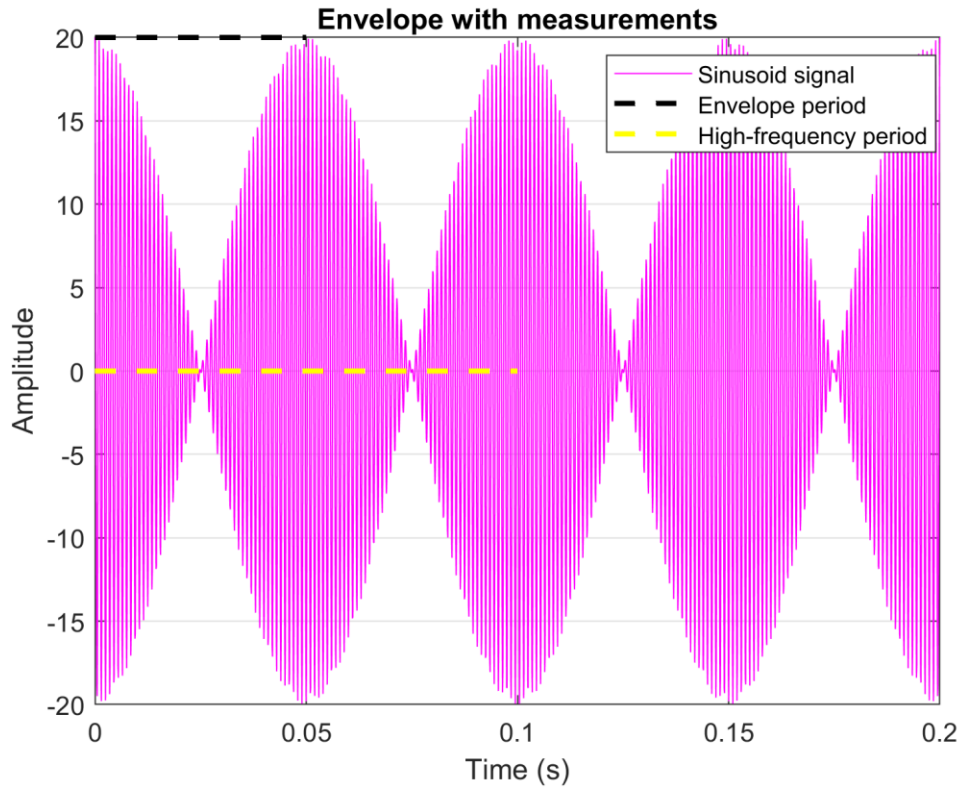
The frequency of the envelope corresponds to the difference between the two original sinusoidal frequencies. Which is delf, 10. The period of the envelope is the distance between two consecutive peaks:

$$Envelope_{period} = 0.05$$

The sound produced sounds bell-like.

```
% MATLAB code for plotting with period measurements
figure;
plot(tt, xx, 'm');
title('Envelope with measurements');
xlabel('Time (s)');
ylabel('Amplitude');
hold on;


% Plot envelope period
envelope_period = 1 / (2 * delf);
plot([0 envelope_period], [20 20], 'k--', 'LineWidth', 2);


% Plot high-frequency period
high_freq_period = 1 / delf;
plot([0 high_freq_period], [0 0], 'y--', 'LineWidth', 2);


legend('Sinusoid signal', 'Envelope period', 'High-frequency period');
grid on;
```

OUTPUT



2.1 MORE ON SPECTROGRAMS

SPECTROGRAM OF A BEAT SIGNAL

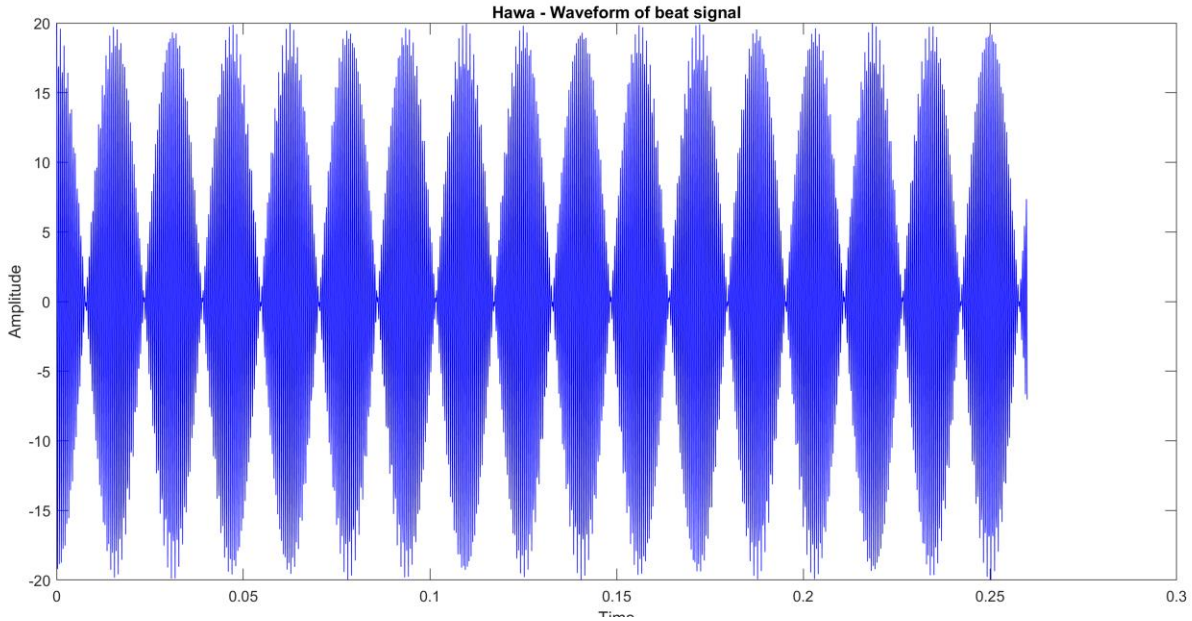
A beat signal was created with given parameters using the beat.m function and plotted to visualize the signal's waveform. In order to notice the trade-off between using either time resolution or frequency resolution in spectrograms; two spectrograms were created with different window lengths.

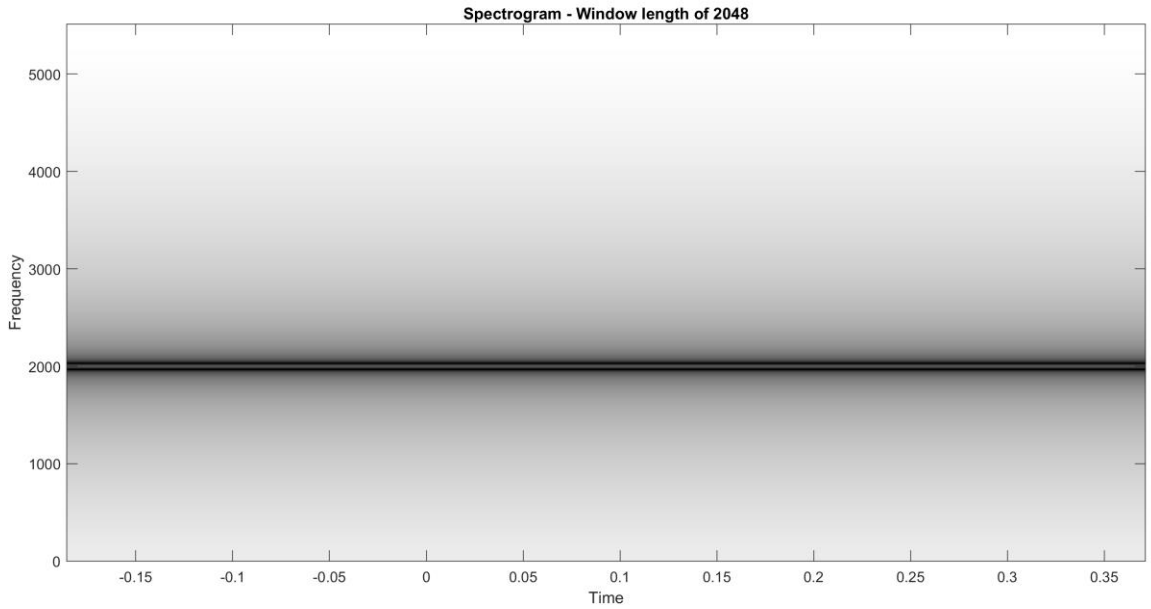
First a window length of 2048 to see how well frequencies could be correctly identified and then 16 to observe the rapid changes in frequency with respect to time.

```
% Beat signal parameters
A = 10;
B = 10;
fc = 2000;
delf = 32;
fsamp = 11025;
dur = 0.26;
```

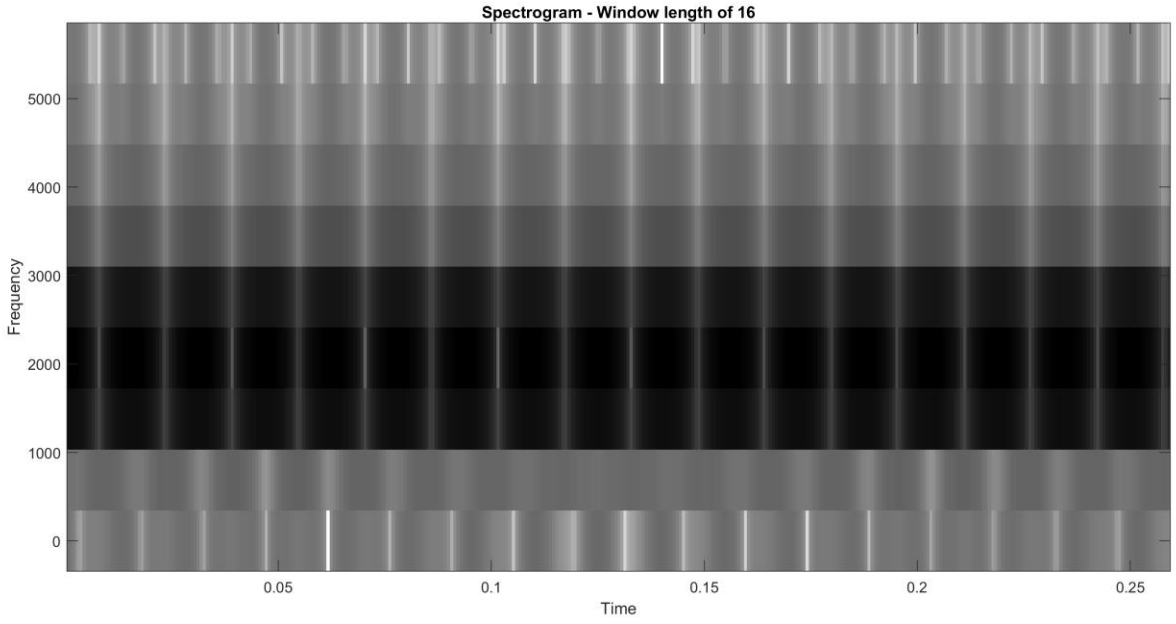
```
% Generating beat signal
[xx, tt] = beat(A, B, fc, delf, fsamp, dur);


% Plot the beat signal
figure;
plot(tt, xx, 'b');
xlabel('Time');
ylabel('Amplitude');
title('Hawa - Waveform of beat signal', 'FontWeight', 'Bold');


% Spectrogram with window length of 2048
figure;
specgram(xx, 2048, fsamp);
colormap(1-gray(256));
title('Spectrogram - Window length of 2048');


% Spectrogram with window length of 16
figure;
specgram(xx, 16, fsamp);
colormap(1-gray(256));
title('Spectrogram - Window length of 16');
```

OUTPUT





A spectrogram with a long window length, like 2048, shows detailed frequency information over a longer time, which helps identify the two distinct frequencies (f_c - delf and f_c + delf). This means it's great at showing precise frequencies but not so good at tracking quick changes.



A spectrogram with a window length of 16 captures rapid frequency changes more effectively but frequency resolution is reduced, resulting in more blurring.

At around 1000Hz to 3000Hz, for both window lengths, the spectrogram gets darker in color, showing the sound's intensity. The darkest color appears at 2000Hz. This change in color shows the intensity of the sound at different frequencies.

3. FREQUENCY MODULATION (FM) SYNTHESIS – CHIRP SIGNAL

A function written during the Warm-up, called mychirp, was made to synthesize a chirp signals.

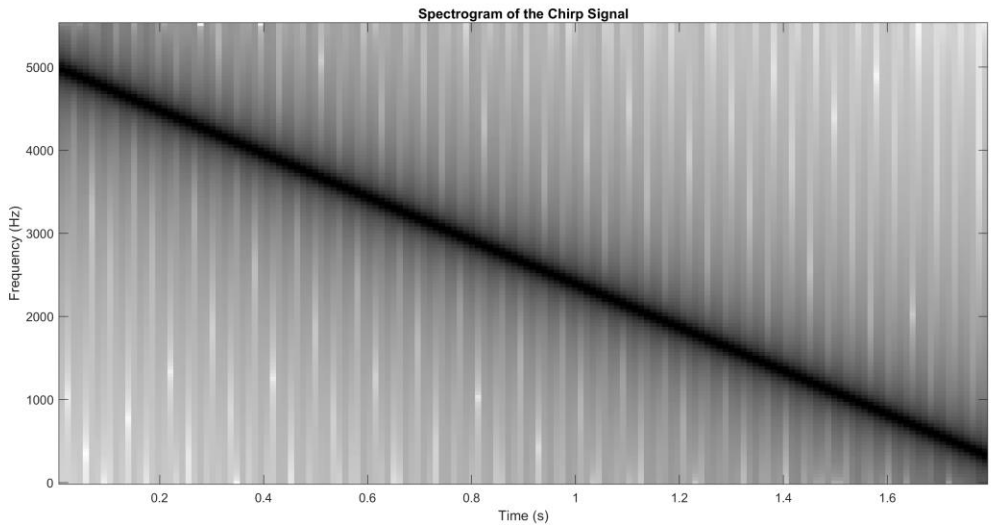
```
function [xx, tt] = mychirp( f1, f2, dur, fsamp )
%MYCHIRP generate a linear-FM chirp signal
%
% usage: xx = mychirp( f1, f2, dur, fsamp )
%
% f1 = starting frequency
% f2 = ending frequency
% dur = total time duration
% fsamp = sampling frequency (OPTIONAL: default is 11025)
%
% xx = (vector of) samples of the chirp signal
% tt = vector of time instants for t=0 to t=dur
%
    if( nargin < 4 ) %-- Allow optional input argument
    fsamp = 11025;
    end


dt= 1/fsamp;
dur = 1.8;
tt = 0 : dt : dur ;
f_step = (f2 -f1)/dur;
psi = pi*f_step*(tt.^2);
xx= cos(2*pi*f1*tt + psi);
soundsc(xx, fsamp)
end
```

3.1 SPECTROGRAM OF A CHRIP

Below is the code for the generated chip signal

```
% Parameters for the chirp signal
f1 = 5000;     % Starting frequency (Hz)
f2 = 300;      % Ending frequency (Hz)
dur = 3;       % Total duration (seconds)
fsamp = 11025; % Sampling frequency (Hz)

% Generate the chirp signal using the mychirp function
[xx, tt] = my_chirp(f1, f2, dur, fsamp);

% Play the chirp signal
soundsc(xx, fsamp);

% Plot the chirp signal
figure;
plot(tt, xx);
xlabel('Time (s)');
ylabel('Amplitude');
title('Waveform of the Chirp Signal');

% Create and plot the spectrogram
figure;
specgram(xx, 256, fsamp); % Using a window length of 256
colormap(1-gray(256));
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Spectrogram of the Chirp Signal');

% Listen to the signal
soundsc(xx, fsamp);
```

OUTPUT



The linear frequency indicated by the dark straight line, starts at 5000Hz and ends at 300Hz. The period is 3 seconds. This spectrogram verifies the instantaneous frequencies were correct with their continuous decrease in frequency with time.

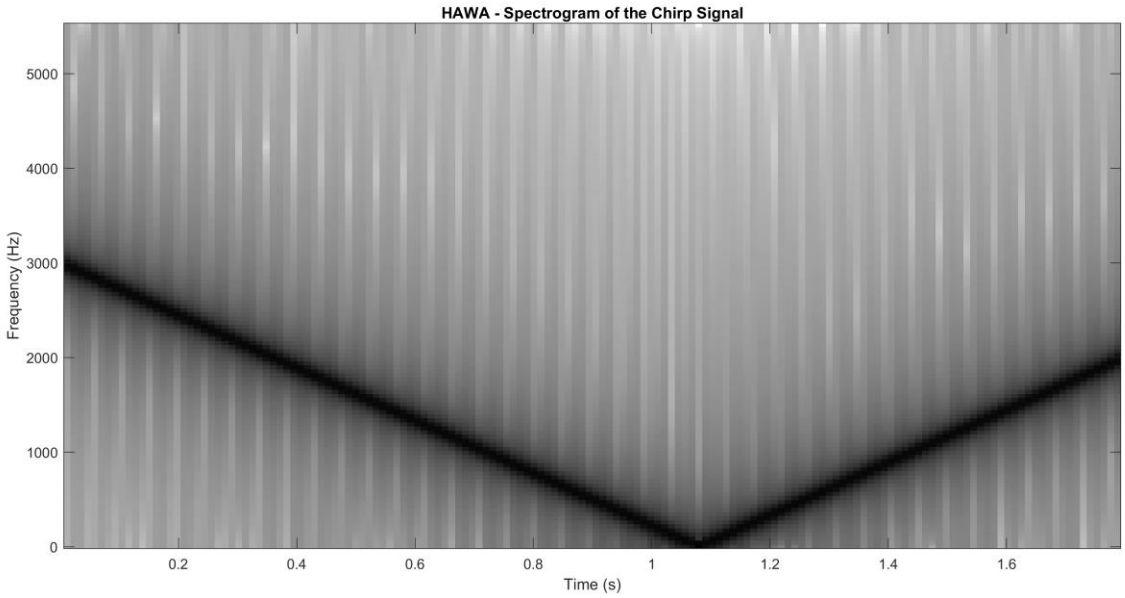
3.2 THE CHIRP PUZZLE

A second chirp signal was created for this report. The instantaneous frequency was configured to start at 3000 Hz and end at -2000 Hz (negative frequency). The signal was listened to, to assess whether it chirped down, up, or both. Subsequently, a spectrogram of the second chirp signal was created.

MATLAB CODE

```
% Parameters for the chirp signal
f1 = 3000;     % Starting frequency (Hz)
f2 = -2000;    % Ending frequency (Hz)
dur = 3;       % Total duration (seconds)
fsamp = 11025; % Sampling frequency (Hz)

% Generate the chirp signal using the mychirp function
[xx, tt] = my_chirp(f1, f2, dur, fsamp);

% Play the chirp signal
soundsc(xx, fsamp);

% Plot the chirp signal
figure;
plot(tt, xx);
xlabel('Time (s)');
ylabel('Amplitude');
title('Waveform of the Chirp Signal');

% Create and plot the spectrogram
figure;
specgram(xx, 256, fsamp); % Using a window length of 256
colormap(1-gray(256));
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Spectrogram of the Chirp Signal');
```

OUTPUT



After listening to the second chirp signal, I noticed that the frequency went both up and down. The spectrogram visually showed this change over time, illustrating the shift from positive to negative frequencies. This happens because the spectrum theory includes both types of frequency components. This changing frequency over time means that the frequency parts in the spectrum were moving, leading to the sounds and visuals observed in the spectrogram.