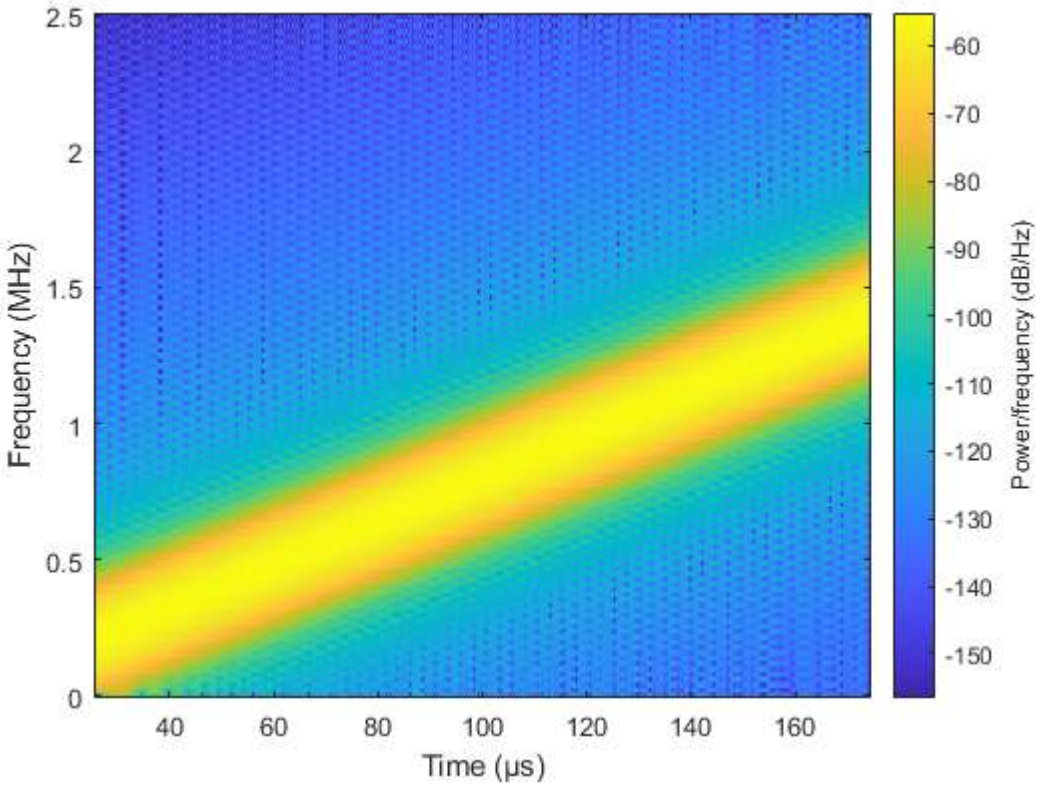
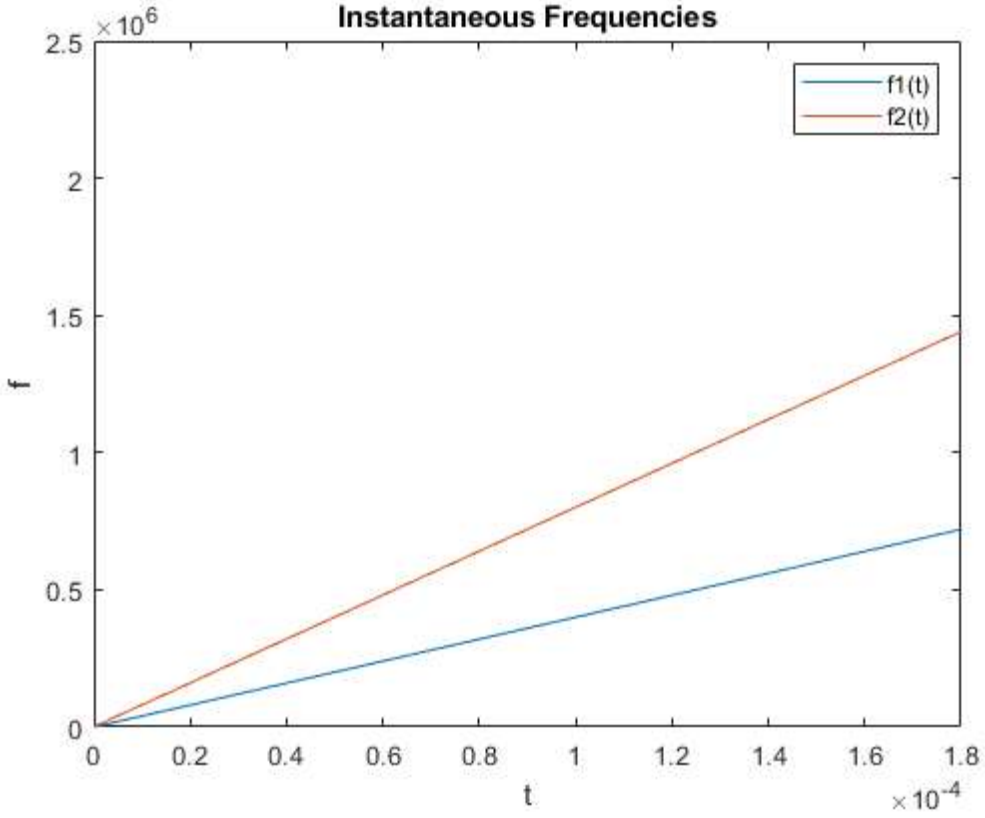
Contents

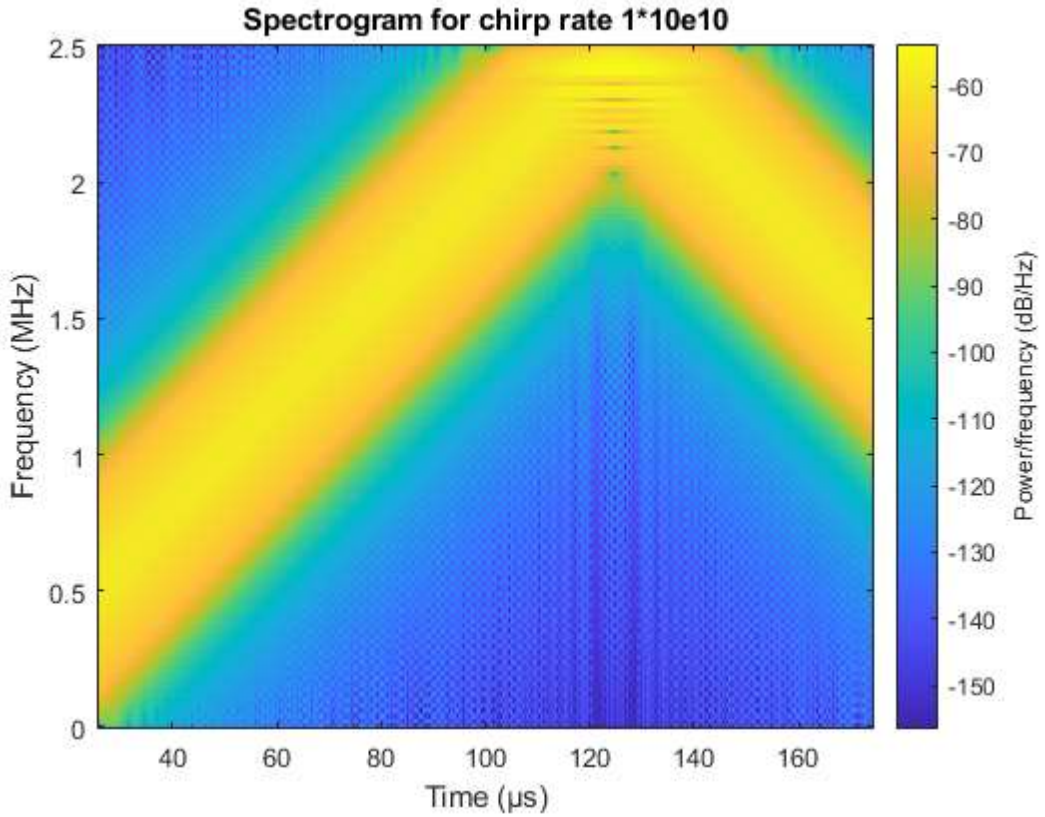

```
clc;
clear;
close all;
```

Frequency Modulated Signals

```
% Q1
fs = 5 * 10^6;
chirp_rate = 4 * 10^9;
chirp_duration = 200 * 10 ^ -6;
t = linspace(0, chirp_duration, 1000);
x = cos(2 * pi * chirp_rate .* t.^2);

figure();
spectrogram(x, triang(256), 255, 256, fs, 'yaxis');

% Q2
f1 = chirp_rate * t;
f2 = (1/(2 * pi)) * diff(2 * pi * chirp_rate * t.^2)./diff(t);

figure();
plot(t, f1, t(1:end-1), f2);
xlabel('t');
xlim([0 180 * 10 ^ -6]);
ylim([0 2.5 * 10 ^ 6]);
ylabel('f');
legend('f1(t)', 'f2(t)');
title('Instantaneous Frequencies')

% f2 corresponds closer to the slope of the ridge in the spectrogram

% Q3
chirp_rate = 1 * 10^10;
x2 = cos(2 * pi * chirp_rate * t.^2);
f1 = chirp_rate * t;
f2 = (1/(2 * pi)) * diff(2 * pi * chirp_rate * t.^2)./diff(t);
% f2 corresponds closer to the slope of the ridge in the spectrogram

figure();
spectrogram(x2, triang(256), 255, 256, fs, 'yaxis');
title('Spectrogram for chirp rate 1*10e10')

% increasing the chirp rate dramatically increases the slope of the ridge
% on the spectrogram. The Initial instantaneous frequency equation still
% matches the slope of the ridge on the spectrogram.
```


Frequency (MHz)
Time (μs)
Power/frequency (dB/Hz)

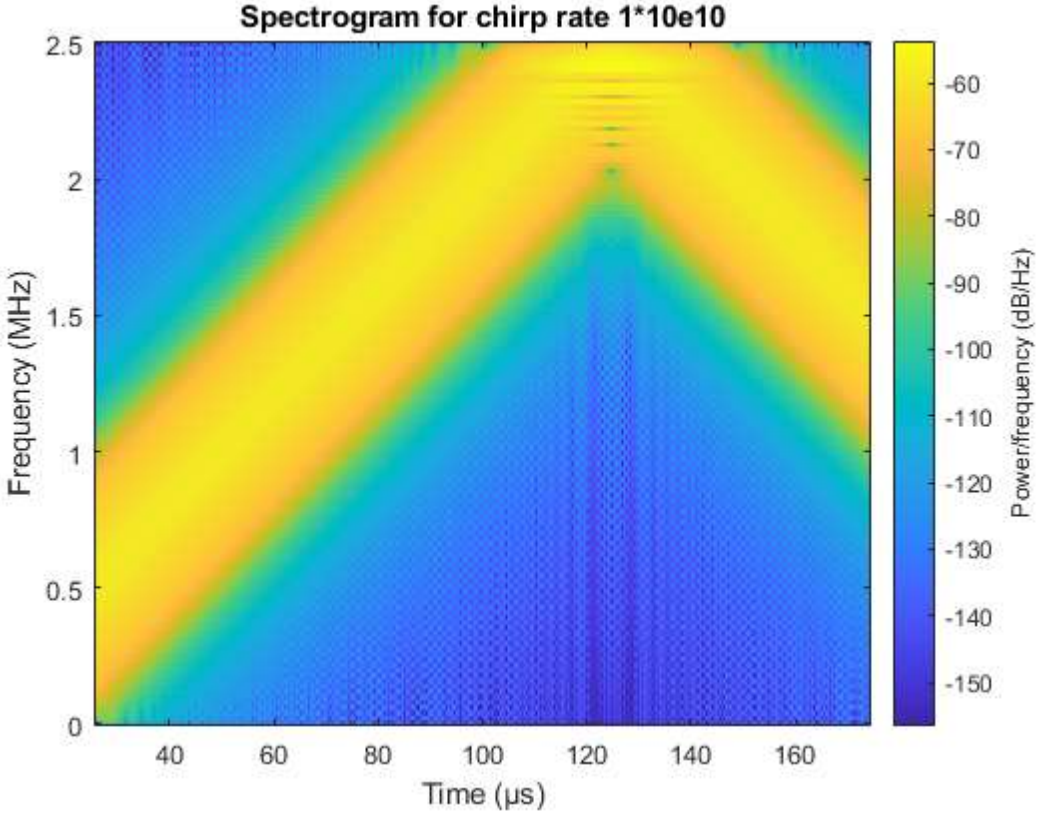

Instantaneous Frequencies
f1(t)
f2(t)
f
t

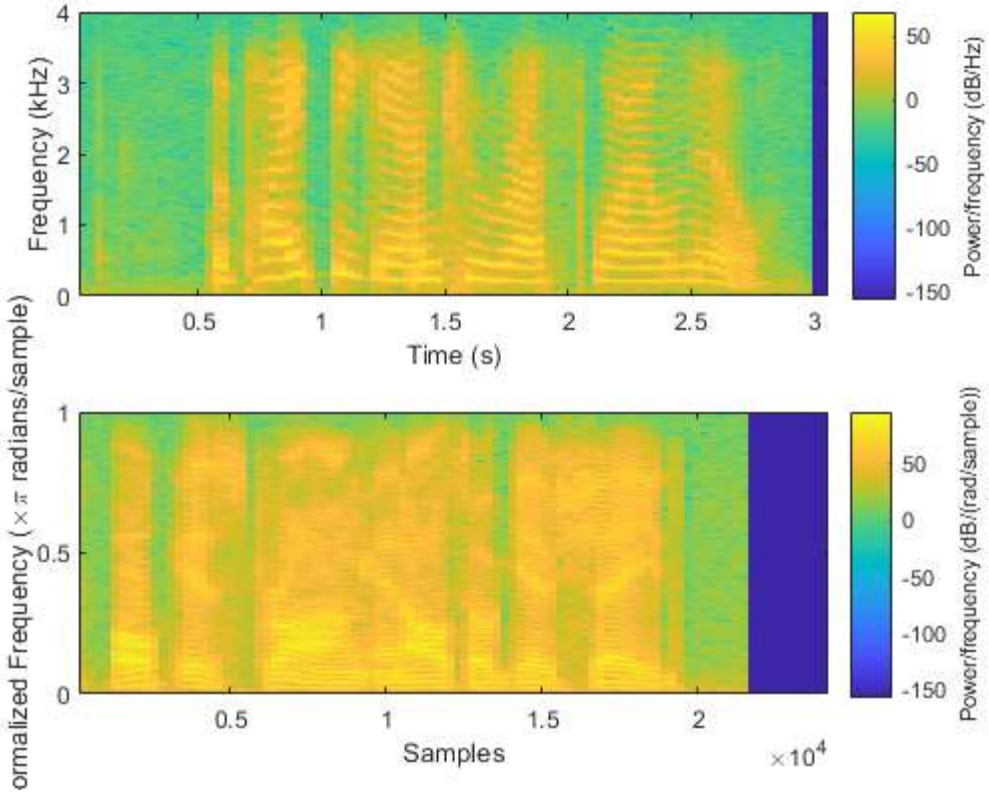


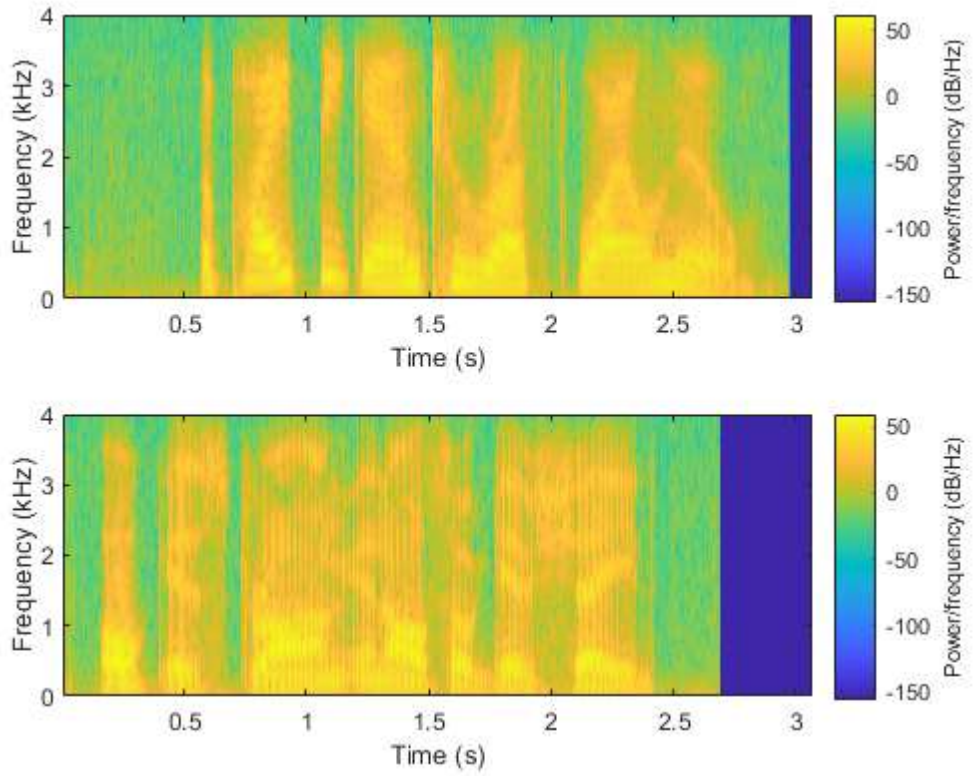
Narrowband and Wideband Spectrograms

```
load('s1.mat');
load('s5.mat');

% Q4
fs = 8000;
%soundsc(s1, fs);
n1 = 512;
n2 = 4096;
n3 = 64;

figure();
subplot(2, 1, 1)
spectrogram(s1, triang(512), (512)/2, 512, fs, 'yaxis');
%figure(1);
subplot(2, 1, 2)
spectrogram(s5, triang(512), (512)/2, 512, 'yaxis');
%subplot(2, 1, 2)
%spectrogram(s1, triang(n1), n1-1, n1, fs, 'yaxis');

%figure(n2);
%subplot(2, 1, 1)
%spectrogram(s1, triang(n2), n2-1, n2, fs, 'yaxis');
%subplot(2, 1, 2)
%spectrogram(s1, triang(n2), n2-1, n2, fs, 'yaxis');

%figure(n3);
%subplot(2, 1, 1)
%spectrogram(s1, triang(n3), n3-1, n3, fs, 'yaxis');
%subplot(2, 1, 2)
%spectrogram(s5, triang(n3), n3-1, n3, fs, 'yaxis');
```

```
% The above comments were results of playing around with different values
% for window size / fft size / jump size. Values were selected as powers of
% two that conform to R <= L <= N.
% 512 seems to be the clearest and still has distinct markings even when jumping
% half of the window.

% Q5

figure();
subplot(2, 1, 1)
spectrogram(s1, triang(64), 63, 4096, fs, 'yaxis');
subplot(2, 1, 2)
spectrogram(s5, triang(64), 63, 4096, fs, 'yaxis');
% high temporal resolution means lower window size, and low frequency
% resolution means larger fft size
```


Spectrogram for chirp rate 1*10e10
Frequency (MHz)
Time (μs)
Power/frequency (dB/Hz)

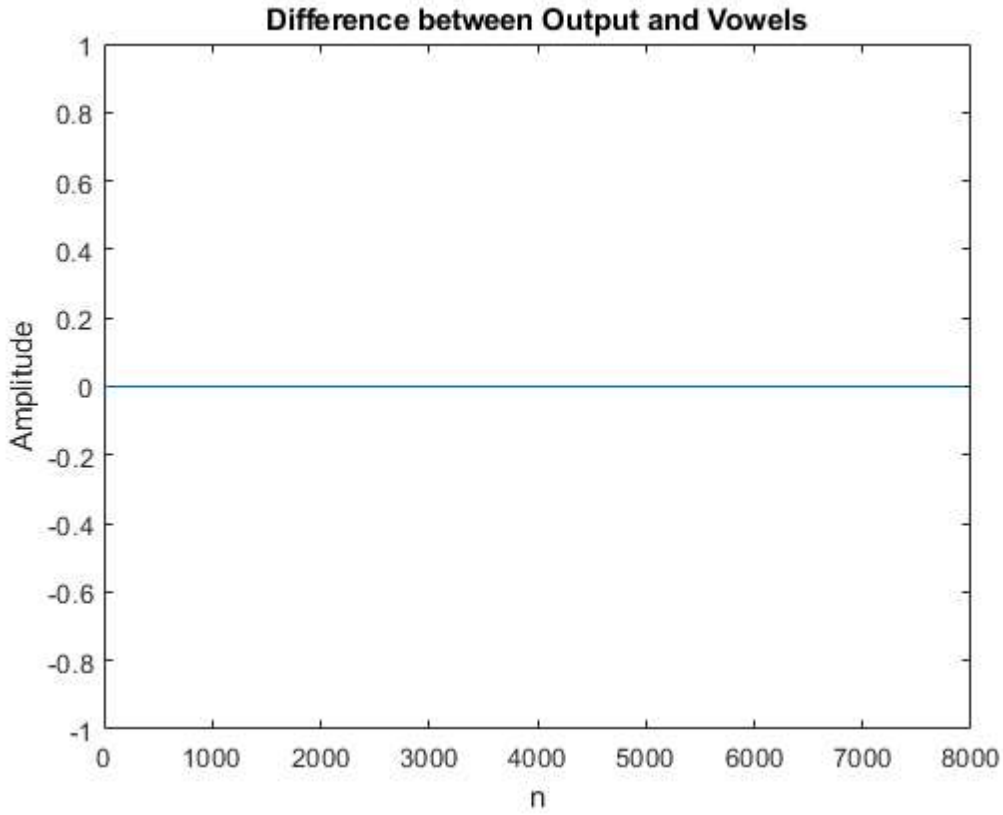

Frequency (kHz)
Time (s)
Power/frequency (dB/Hz)
ormalized Frequency (×π radians/sample)
Samples
×10^4
Power/frequency (dB/(rad/sample))

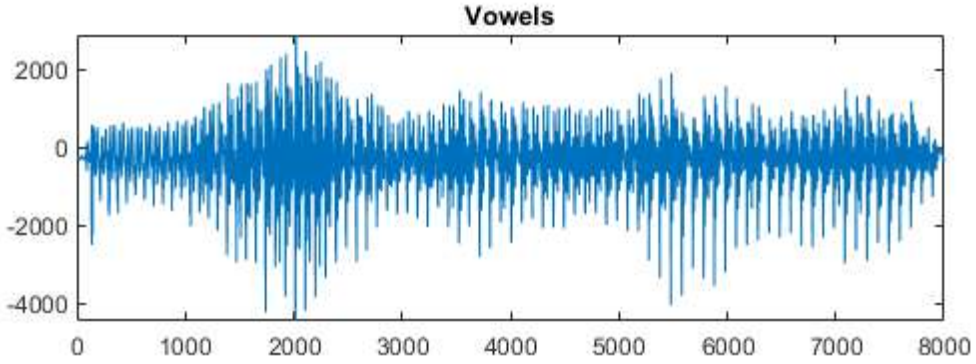


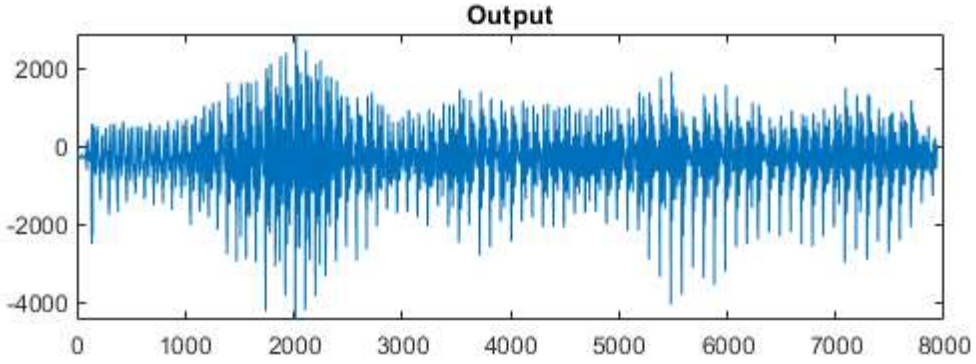
Modified Short-Time Fourier Transforms

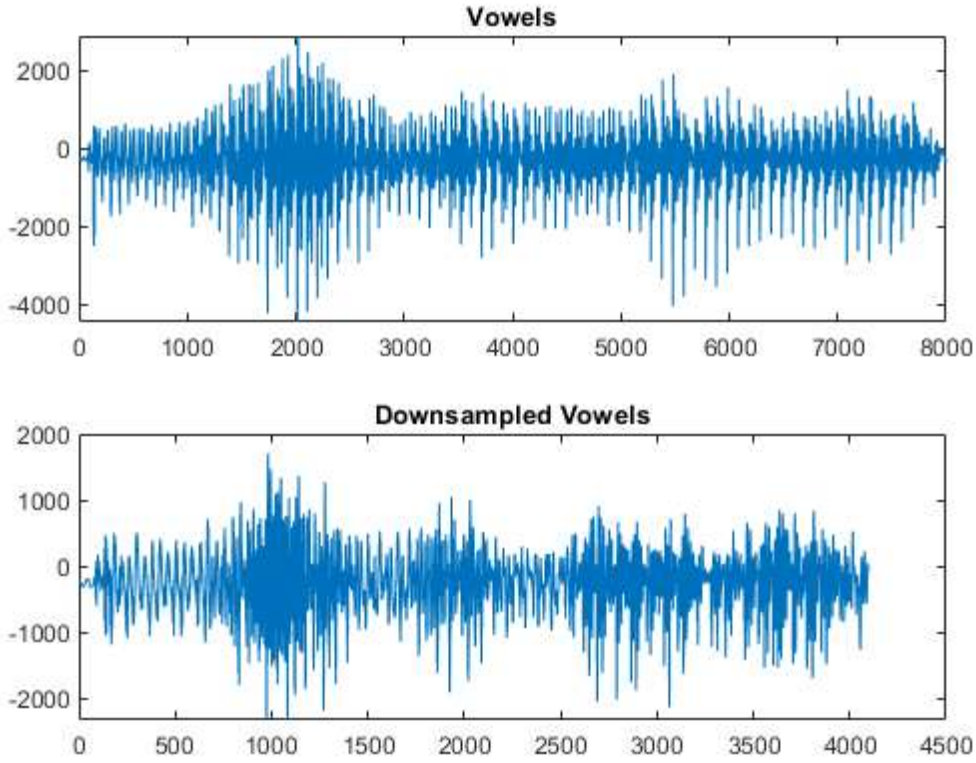
```
% Q6

load('vowels.mat');
fs = 5e6;

v = padarray(vowels, 2^nextpow2(length(vowels)) - length(vowels), 0, 'post');
y = spectrogram(vowels, rectwin(256), 128, 1024, fs, 'yaxis');
output = istft(y, 1024);

%soundsc(vowels, fs);
%pause(3);
%soundsc(output, fs);

% No audbile difference between the two.

figure();
plot(output - vowels(1:length(output)));
title('Difference between Output and Vowels');
xlabel('n');
ylabel('Amplitude');
xlim([0 length(vowels)]);
ylim([-1 1])

figure();
subplot(2, 1, 1)
plot(vowels);
title('Vowels');
subplot(2, 1, 2)
plot(output);
title('Output');
```

```
% Q7
yDown = y(:, 1:2:end);
fastOutput = istft(yDown, 1024);

%soundsc(vowels, fs);
%pause(3);
%soundsc(fastOutput, fs);

% after removing every other element, the sound is sped up.

figure();
subplot(2, 1, 1);
plot(vowels);
title('Vowels')
subplot(2, 1, 2);
plot(fastOutput);
title('Downsampled Vowels');

function [out] = istft(signal, n)
    signal = [signal; flipud(signal)];
    inv = ifft(signal, n, 'symmetric');
    out = zeros(n, size(inv, 2));
    index = 1;

    for i = 1:size(inv, 2)
        out(index:index+255, i) = real(inv(1:256, i)); % cycle through columns
        index = index + 128; % increment to next 128 values
    end

    out = sum(out, 2); % add up all columns (columns are staggered with values)
    out(129:length(out) - 128) = out (129:length(out) - 128)/2;
end
```


Difference between Output and Vowels
Amplitude
n
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1000
2000
3000
4000
5000
6000
7000
8000


Vowels
2000
0
-2000
-4000
0
1000
2000
3000
4000
5000
6000
7000
8000


Output
2000
0
-2000
-4000
0
1000
2000
3000
4000
5000
6000
7000
8000



What happened in this lab?

For the first few parts of this lab, we created spectrograms for a chirp signal. The spectrogram for a chirp shows the linear rate of change in frequency over time. Next, two models for calculating the instantaneous frequency of the chirp were tested. In order to test which model was better, one can compare the slopes of the spectrograms with the slopes of the modeled instantaneous frequencies. Because the spectrogram is a range of frequencies over time, the instantaneous frequency should resemble a similar slope. Next, two signals, s1 and s5, were loaded in. Spectrograms were made of those signals as well. The goal was to create a spectrogram that would provide accurate enough information to understand what is happening in the signal. After some experimentation, it was found that a window size of 512, a jump of half a window, and an FFT size of 512, was sufficient enough to generate a clear spectrogram. Finally, the last part of the lab was to implement the inverse short time fourier transform. The STFT can be used to change the signal but preserve as much of the original frequency ranges as possible. As show above, the signal was decimated- every other sample was removed, but when using the ISTFT on it, it ended up sounding the same (but faster, as the signal was half as long).

Published with MATLAB® R2020b