Time-Frequency Analysis of Audio Signals

Christina Smith

AMATH 482 - Winter 2020

Abstract

In this two-part exercise, we perform time-frequency analysis on three sets of audio data. In the first, we analyze Handel's 'Messiah' by creating spectrograms and exploring how the output changes in relation to precision, accuracy, and resolution in the time in frequency domains as we vary the window width and time-step parameters of a Gábor filter.

In the second part, we evaluate two audio samples of 'Mary Had a Little Lamb'; one played on a piano, one on a recorder. Using a Gábor filter we analyze the data to reproduce the musical score and compare the overtones generated by each instrument.

I. Introduction

This project focuses on analyzing data from signals which vary in frequency and strength over time. The first part is an exercise in exploring how different parameters of the Gábor filter affect the accuracy of our data in regards to frequency and time. The Gábor filter contains a scalar parameter for window width, and a vector for the time increments at which we filter the data. We see that there are trade offs between the resolution we can achieve in regards to time and frequency as we vary the parameter values.

In part two of this project, we evaluate two audio clips of the same score played on two different instruments (piano, recorder). By processing this data using the Gábor method we identify the central frequencies of the strongest signals at points in time and can use this information to reproduce the musical score. By performing Fourier transforms on both audio signals, we can view the different ranges of frequencies and overtones which contribute to differences in sound between the two instruments.

II. Theoretical Background

In our previous work, we successfully used a Gaussian filter and fast Fourier transform (FFT) to filter a signal with a frequency that was constant over time.

$$\text{Gaussian Filter:} \quad f(k) = \mathrm{e}\hat{}(-a(k-k_c)^2) \tag{1}$$

Where a determines the width of the filter, k represents a given point in our frequency domain, and k_c represents the central frequency of the data.

In this exercise, the frequency of our signal data changes over time so we will need to modify our approach to analyzing this data. The solution is to to consider smaller increments of time within our data using a Gábor filter.

$$\text{Gábor Filter:} \quad g(\mathbf{t}-\boldsymbol{\tau}) = \mathrm{e}\hat{}(-a(\mathbf{t}-\boldsymbol{\tau}_j)^2) \tag{2}$$

Where a determines the width of the filter, $\mathbf{t}$ contains the time information corresponding to each of our signal data points, and $\boldsymbol{\tau}$ contains the values of the time domain at which we center the filter as we step through the data. The result is that our filter changes as we traverse the sample, filtering less data from the time point at which it is centered and more data from points in time further from this center.

Using this filter, we can extract frequency information for time intervals of our data. The filter width affects the resolution we can achieve in both the time and frequency domains. A smaller a value creates a wider window because as a approaches 0, the value of g approaches 1. Inversely, as a increases, the value of g decreases.

A wider window will capture more frequency data, making the frequency estimate more accurate, however we will be less certain at which point in time this frequency occurs. A narrower window gives us greater certainty in regards to time however we lose frequency data as wavelengths wider than the window will be filtered out.

We can see this effect well by plotting spectrograms of our data. Figure 1 shows how the frequency resolution decreases as the filter window width decreases, as well as how an FFT leaves no resolution in the time domain.

Another factor that will influence the output of our filtering process is the amount of overlap in our filtering windows. When we have a and $\boldsymbol{\tau}$ values corresponding to smaller window and a greater translations, our spectrograms lose detail.

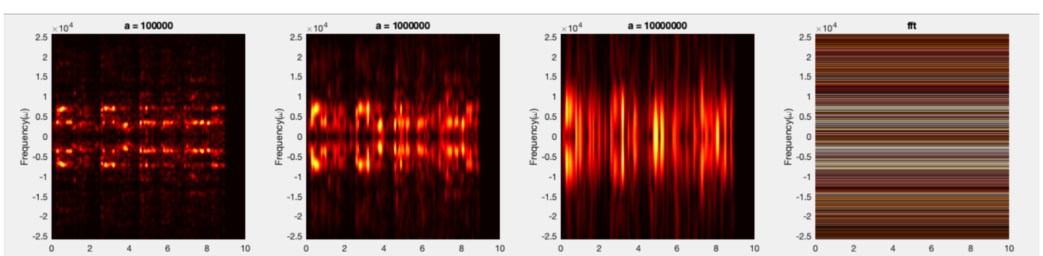

Figure 1: Effects of Changing Width of Gábor Window on 'Messiah' Spectrogram

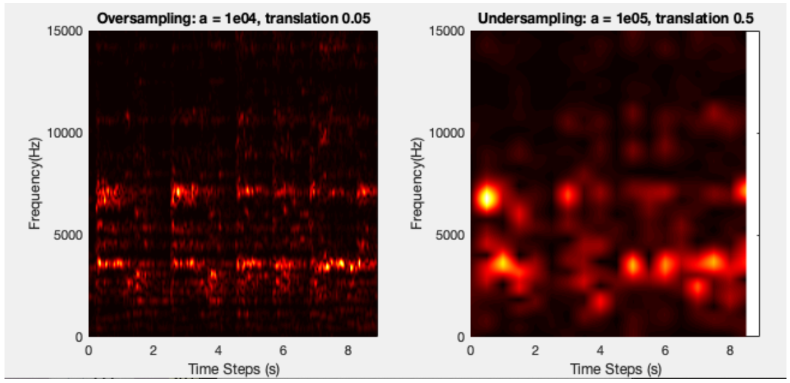

Figure 2: Oversampled and Undersampled Data from Handel's 'Messiah'

In cases like this, there is less overlap, or maybe even gaps, between our filter windows. There are less areas of high signal shown in the spectrogram and the signal values become less precise in time and frequency. This is called undersampling.

Oversampling occurs when there is too much overlap between filter windows. In this case same data is processed by many different filter windows. We end up with more areas of high signal data that are more precise, but less accurate. Figure 3 shows examples of spectrograms that have been created from data that has been undersampled and oversampled.

Having very small filter windows can be problematic if we want to analyze the main tones in an audio clip. In music, the true notes are the lowest frequency so they are the ones first lost if we make the filter window width too narrow. Other tones appear in the audio data called overtones. Overtones are generated in multiples of the frequency of the original note. Figure 3 shows an example of this from music played on a piano an a recorder respectively. You can see there are periodic peaks at higher frequencies which lose amplitude as they get farther from the frequency of the original note. If our Gábor is narrower than a wavelength of the lowest frequency then we will lose the signal data for the true note played.

III. Algorithm Implementation and Development

When we load the .wav file in Matlab, we get a vector containing signal data and a scalar value of the sampling frequency. Dividing the number of samples by the frequency gives us the length of our audio clip, our time domain. We create a vector with values in our time domain corresponding to each element in our sample data vector. We also create a vector for our frequency domain values. Note that the length of this vector must match the length of the data vector. Our base formula will depend on whether our data contains an even or odd number of elements (n). (See Appx. B, lines 8 - 16)

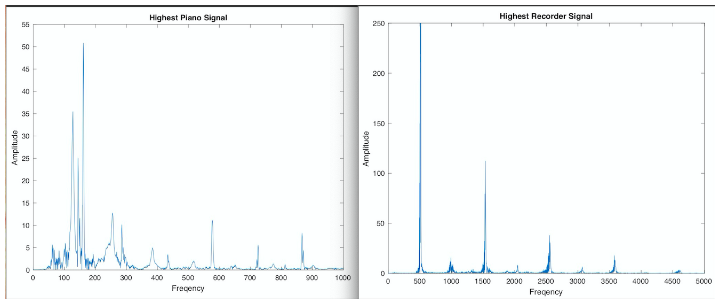

Figure 3: Unfiltered Signal Frequencies of a Piano and a Recorder

$$\text{Formula for even number of elements:} \quad k = [0 : (n/2 - 1) - n/2 : -1] \tag{3}$$

$$\text{Formula for odd number of elements:} \quad k = [0 : (n-1)/2 - (n-1)/2 : -1] \tag{4}$$

We can now begin to filter and analyze the data. As noted in Section II of this report, it is important to choose parameters which balance the time and frequency output resolution. This is is a process of evaluating the data we have and a bit of trial and error.

After determining suitable values for the parameters and filtering the data with the Gábor method, we perform an FFT. Since our data has been filtered by time, the FFT gives us an estimate of the frequencies over the time period included in the Gábor window. In order to get frequency data for our entire audio clip, we repeat this Gábor filtering on different time segments of the data. This is where it is necessary to consider the size of our Gábor window, as well as what increments of time we are using to traverse our data. Choices with too much or too little overlap can result in under or oversampling as discussed in Section II. (See Appx. B, lines 179 - 189)

We can repeat this process in a loop, iterating for every time step we want to take through the data. At each time step, we get a vector of signal data with indices that correspond to the values in our frequency vector. We create a matrix from these individual vectors.

When we have filtered data for all time steps, we can create a spectrogram. Using Matlab's pcolor() function, we pass in the vectors containing our time step values and our frequency values, as well as our matrix of signal data in the frequency domain. The result is a plot with axes corresponding to time and frequency and colors corresponding to the signal strength at a given frequency at a given point in time. From this we can analyze the data to determine which frequencies occur at which time. (See Appx. B, lines 190 - 197)

It is important to note that when we are making this spectrogram, we want to compare our frequencies to the frequencies of musical notes. We need to change our scaling factor for k. Originally it had been scaled by 2π in order to correspond with the FFT, however now we change change the 2π to 1 to represent the units Hz, which are cycles per second.

IV.Computational Results & Supplementary Plots

Figures 4 shows the piano score for 'Mary Had a Little Lamb'.
Figure 5 shows the recorder score for 'Mary Had a Little Lamb'.
Figure 6 shows the spectrograms for both audio files.

V.Summary and Conclusions

Through this exercise we have used the Gábor method to evaluate signals that fluctuate in time. The results have shown that the width of the filter and the increments in time at which we filter the data will affect the output of this process.

In part 1, we see that larger coefficient in the filter exponent results in a narrower filter window, which will produce greater resolution in the time domain, but less in the frequency domain. A smaller coefficient produces a larger window with an inverse effect on the domain resolutions.

A smaller filter and a greater time step, may result in undersampling which outputs less precise signal data. A larger filter and smaller time steps may result in oversampling which results in more precise data with less accuracy.

In Part 2, we compare the overtones of a piano and a recorder. The recorder has less overtones and in a signal/frequency plot, they are more clearly differentiated than the piano overtones. The piano has more overtones and so the signal/frequency plot shows periodic groups of overtones. By this we can deduce that the piano has a 'richer' timbre, encompassing more frequencies than the recorder.

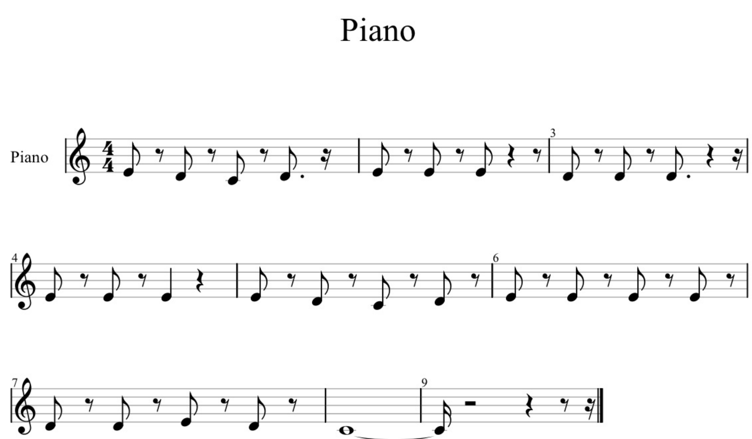

Figure 4: 'Mary Had a Little Lamb' Score from Piano Audio

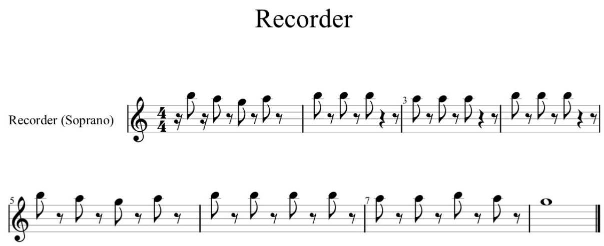

Figure 5: 'Mary Had a Little Lamb' Score from Recorder Audio

Appendix A: Matlab Functions

audioread(): This function takes in an audio file and returns a vector of the signal data as well as a scalar value of the sampling frequency.

pcolor(): This function takes in two vectors and a matrix and plots a spectrogram with axes representing the vector values and colors representing the data values at the given coordinates.

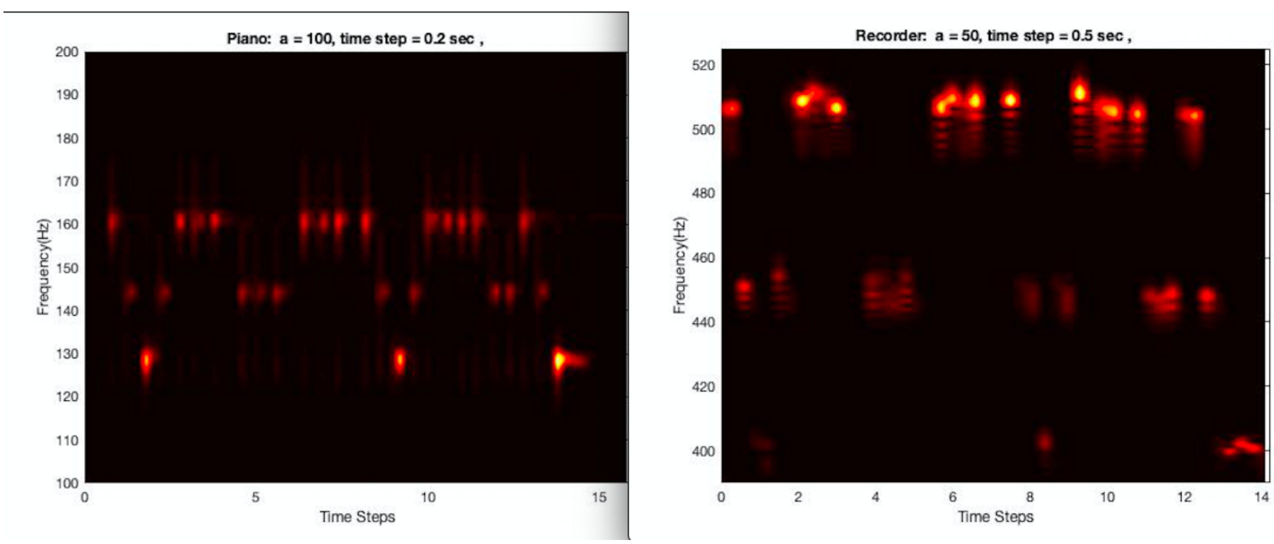

Figure 6: 'Mary Had a Little Lamb' Spectrograms of Piano and Recorder Audio

Appendix B: Matlab Code

```

%% HW 02 Part 1

% Construct signal and plot signal and FFT

close all; clear all; clc;

load handel
% y = audio signal
S = y'; % transpose y vector
n=length(S);    % number of samples
L=n/Fs;  % time domain
t2=linspace(0,L,n+1);
t=t2(1:n);
k=(2*pi/2*L)*[0:(n-1)/2 -(n-1)/2:-1]; % frequency vector
ks=fftshift(k);

% figure(1)
plot((1:length(y))/Fs,S)
xlabel('Time [sec]')
ylabel('Amplitude')
title('Signal as a function of time')
% p8 = audioplayer(S,Fs);
% playblocking(p8);

St=fft(S);

% figure(2)
subplot(2,1,1)
% plot the function against time
plot(t,S,'k','Linewidth',2)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

subplot(2,1,2)
% plot FFT against (normalized) frequency
plot(ks,abs(fftshift(St))/max(abs(St)),'r','Linewidth',2);
title("Signal v Frequency",'Fontsize',12)
set(gca,'Fontsize',11)
xlabel('frequency (\omega)'), ylabel('FFT(S)')

% Construct Gabor window and add to time domain plot
tau = 4;
a = 100; % width of filter
g = exp(-a*(t-tau).^2);  % the filter
subplot(2,1,1)
plot(t,S,'k',t,g,'m','Linewidth',2)
title("Signal v Time & Gabor Window: a = 100, tau = 4",'Fontsize',12)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

```

```
%% Apply filter and take fft

a = 100; % width of filter
g = exp(-a*(t-tau).^2);  % the filter
Sg = g.*S;
Sgt = fft(Sg);

% figure(3)
subplot(3,1,1)
% subplot(3,2,2)
plot(t,S,'k','Linewidth',2)
axis([2 6 -1 1])
title("Signal v Time & Gabor Window: a = 100",'Fontsize',12)
hold on
plot(t,g,'m','Linewidth',2)  % shows the filter on the S(t) plot
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

subplot(3,1,2)
% subplot(3,2,4)
plot(t,Sg,'k','Linewidth',2)
axis([2 6 -1 1])
title("Gabor Filtered Signal v Time",'Fontsize',12)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('Sg(t)')


subplot(3,1,3)
% subplot(3,2,6)
plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2)
title("FFT of Gabor Filtered Signal v Frequency",'Fontsize',12)
set(gca,'Fontsize',11)
xlabel('frequency (\omega)'), ylabel('FFT(Sg)')


a = 1; % width of filter
g = exp(-a*(t-tau).^2); % Gaussian, but can try Shannon or Mexican hat
Sg = g.*S;
Sgt = fft(Sg);

figure(4)
subplot(3,1,1)
% subplot(3,2,1)
plot(t,S,'k','Linewidth',2)
axis([2 6 -1 1])
title("Signal v Time & Gabor Window: a = 1",'Fontsize',12)
hold on
plot(t,g,'m','Linewidth',2)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

% less accuracy in the time domain
subplot(3,1,2)
% subplot(3,2,3)
```

```
plot(t,Sg,'k','Linewidth',2)
axis([2 6 -1 1])
title("Gabor Filtered Signal v Time",'Fontsize',12)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('Sg(t)')

% more detail in the frequency domain
subplot(3,1,3)
% subplot(3,2,5)
plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
title("FFT of Gabor Filtered Signal v Frequency",'Fontsize',12)
set(gca,'Fontsize',11)
xlabel('frequency (\omega)'), ylabel('FFT(Sg)')


%% Change window size - narrower window
% larger a, less accuracy in frequency, more accuracy in time
a = 1000;
g = exp(-a*(t-tau).^2);
Sg = g.*S;
Sgt = fft(Sg);

figure(5)
subplot(3,1,1)
plot(t,S,'k','Linewidth',2)
axis([2 6 -1 1])
title("Signal v Time & Gabor Window: a = 1,000",'Fontsize',12)
hold on
plot(t,g,'m','Linewidth',2)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

% less accuracy in the time domain
subplot(3,1,2)
plot(t,Sg,'k','Linewidth',2)
axis([2 6 -1 1])
title("Gabor Filtered Signal v Time",'Fontsize',12)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('Sg(t)')

% more detail in the frequency domain
subplot(3,1,3)
plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
title("FFT of Gabor Filtered Signal v Frequency",'Fontsize',12)
set(gca,'Fontsize',11)
xlabel('frequency (\omega)'), ylabel('FFT(Sg)')

%% Animation of shifting window

a = 1;
tstep=0:0.1:100;

% figure(5)
for j=1:length(tstep) % [1:1001]
```

```
    g=exp(-a*(t-tstep(j)).^2);
    Sg=g.*S;
    Sgt=fft(Sg);

    subplot(3,1,1)
    plot(t,S,'k','Linewidth',2)
    hold on
    plot(t,g,'m','Linewidth',2)
    title("Signal v Time & Gabor Window: a = 1",'Fontsize',12)
    hold off
    set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

    subplot(3,1,2)
    plot(t,Sg,'k','Linewidth',2)
    title("Gabor Filtered Signal v Time",'Fontsize',12)
    set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('Sg(t)')

    subplot(3,1,3)
    plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
    title("FFT of Gabor Filtered Signal v Frequency",'Fontsize',12)
    set(gca,'Fontsize',11)
    xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
    drawnow
    pause(0.1)
end

%% Calculate Gabor transform and plot spectrogram
a = 1e03;
tstep = 0:0.04:L;

Sgt_spec = zeros(length(tstep),n);

for j=1:length(tstep) % [1:101]
    g=exp(-a*(t-tstep(j)).^2);
    Sg=g.*S; % apply new filter
    Sgt=fft(Sg); % transform to frequency domain
    Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
end

figure(6)
pcolor(tstep,ks,Sgt_spec.'),
shading interp
title('Unfiltered: a = 1e03, inc = 0.0446','Fontsize',16)
set(gca,'Fontsize',11)
axis([0 L 3e03 8e03])
colormap(hot)

%% Oversampling and Undersampling

a = 1e04;
tstep = 0:0.05:L;
```

```
Sgt_spec = zeros(length(tstep),n);

for j=1:length(tstep) % [1:101]
    g=exp(-a*(t-tstep(j)).^2);
    Sg=g.*S; % apply new filter
    Sgt=fft(Sg); % transform to frequency domain
    Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
end

figure(7)
subplot(1,2,1)
pcolor(tstep,ks,Sgt_spec.'),
shading interp
title('Oversampling: a = 1e04, translation 0.05','Fontsize',16)
set(gca,'Fontsize',11)
axis([0 L 0 1.5e04])
colormap(hot)


a = 1e05;
tstep = 0:.5:L;

Sgt_spec = zeros(length(tstep),n);

for j=1:length(tstep) % [1:101]
    g=exp(-a*(t-tstep(j)).^2);
    Sg=g.*S; % apply new filter
    Sgt=fft(Sg); % transform to frequency domain
    Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
end

% figure(6)
subplot(1,2,2)
pcolor(tstep,ks,Sgt_spec.'),
shading interp
title('Undersampling: a = 1e05, translation 0.5','Fontsize',16)
set(gca,'Fontsize',11)
axis([0 L 0 1.5e04])
colormap(hot)

%% Filter the signal data in the frequency domain using a Gaussian filter
%
Sgt_spec_filtered = zeros(length(tstep),n);
tau = .5e-05;
it = length(tstep);

for j=1:it % [1:201]
    test = Sgt_spec(j,:);
    % get the max signal and index
```

```
    [test_M, test_I] = max(test);
    % get frequency associated with max signal
    % use ks because we built Sgt_spect with fftshift
    test_center_freq = abs(ks(test_I));
    test_filter = exp(-tau*(ks-test_center_freq).^2);
    test_f = test_filter.*test; % Apply the filter to the signal in
        frequency space
    Sgt_spec_filtered(j,:) = fftshift(test_f); % We don't want to scale
        it


    figure(7)
    subplot(2,1,1)
    plot(abs(ks),test,'r','Linewidth',2);
    title("Unfiltered Signal v Frequency at timestep = " + num2str(j),'
        Fontsize',12)
    set(gca,'Fontsize',11), xlabel('Frequency'), ylabel('Signal')
    drawnow

    subplot(2,1,2)
    plot(abs(ks),test_f,'r','Linewidth',2);
    title("Filtered Signal v Frequency at timestep = " + num2str(j),'
        Fontsize',12)
    set(gca,'Fontsize',11), xlabel('Frequency'), ylabel('Signal')
    drawnow
    pause(2)
end

figure(8)
pcolor(tstep,ks,Sgt_spec_filtered.'),
shading interp
title('Unfiltered: a = 1e06, translation = 8.9249e-02','Fontsize',16)
set(gca,'Fontsize',11)
colormap(hot)

%% Spectrograms for varying translation sizes
figure(9)

a = 1e5;
tstep_vec = [1 0.5 0.1 0.05];
for jj = 1:length(tstep_vec)

    tstep = 0:tstep_vec(jj):10;
    Sgt_spec = zeros(length(tstep),n);

    for j=1:length(tstep)
        g=exp(-a*(t-tstep(j)).^2);
        Sg=g.*S;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end
```

```
    subplot(2,2,jj)
    pcolor(tstep,ks,Sgt_spec.'),
    shading interp
    title(['a = 1e5,  tstep = ',num2str(tstep_vec(jj))],'Fontsize',12)
    set(gca,'Fontsize',11)
    colormap(hot)
end

% plot of the FFT, we know it has no time info
figure(10)
Sgt_spec = repmat(fftshift(abs(St)),length(tstep),1);
pcolor(tstep,ks,Sgt_spec.'),
shading interp
title('fft','Fontsize',11)
set(gca,'Fontsize',11)
colormap(hot)

%% Spectrograms for varying window sizes
figure(11)

a_vec = [1e5 1e6 1e7];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tstep=0:0.1:10;
    Sgt_spec = zeros(length(tstep),n);
    for j=1:length(tstep)
        g=exp(-a*(t-tstep(j)).^2);
        Sg=g.*S;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    subplot(2,2,jj)
    pcolor(tstep,ks,Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',12)
    set(gca,'Fontsize',11)
    colormap(hot)
end

Sgt_spec = repmat(fftshift(abs(St)),length(tstep),1);
subplot(2,2,4)
pcolor(tstep,ks,Sgt_spec.'),
shading interp
title('fft','Fontsize',12)
set(gca,'Fontsize',11)
colormap(hot)

%% Rescale our Gabor transform spectrograms to have the same units for
    frequency
```

```
figure(19)

% k is in terms of 2pi/sec
% Hz is in terms of period/sec
% both are frequencies, so change the 2pi in k, to 1
k=(1*pi/L)*[0:(n-1)/2 -(n-1)/2:-1]; % frequency vector
ks=fftshift(k);
tstep=0:L/100:L;

% Spectrograms for varying a
a_vec = [5 1 0.2];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tstep=0:0.1:10;
    Sgt_spec = zeros(length(tstep),n);
    for j=1:length(tstep)
        g=exp(-a*(t-tstep(j)).^2);
        Sg=g.*S;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    subplot(2,2,jj)
    pcolor(tstep,ks,Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',12)
    set(gca,'Ylim',[-10 10],'Fontsize',11)
    colormap(hot)
end


Sgt_spec = repmat(fftshift(abs(St)),length(tstep),1);
subplot(2,2,4)
pcolor(tstep,ks,Sgt_spec.'),
shading interp
title('fft','Fontsize',12)
set(gca,'Ylim',[-10 10],'Fontsize',11)
colormap(hot)

%% HW 02 Part 2 Recorder

 close all; clear all; clc;


 [y,Fs] = audioread('music2.wav');
 S = y'; % transpose the vector
 T = 1/Fs; % period (second per sample)
 L =length(S)/Fs;  % record time in seconds


figure(1)
```

```
 plot((1:length(S))/Fs,S);
 xlabel('Time [sec]'); ylabel('Amplitude');
 title('Mary had a little lamb (recorder)');
 % p8 = audioplayer(y,Fs); playblocking(p8);

n=length(S); % data points
t2=linspace(0,L,n+1); %
t=t2(1:n); % our time vector matching the number of signal data points

k= (2*pi/(2*L))*[0:(n/2-1) -n/2:-1];
ks=fftshift(k);

%% Fourier transform of the function
% Signal in frequency domain
S_fft=fft(S);

% figure(2)
% plot the function against time
subplot(2,1,1)
plot(t,S,'k','Linewidth',2)
axis([0 L 0 1.1])
set(gca,'Fontsize',11'), xlabel('Time (t)'), ylabel('S(t)')


subplot(2,1,2)
% plot (percentage of?)function against frequency
plot(ks,abs(fftshift(S_fft))/max(abs(S_fft)),'r','Linewidth',2);
title("Signal v Frequency")
axis([-5e3 5e3 0 1.1])
set(gca,'Fontsize',11'), xlabel('frequency (\omega)'), ylabel('FFT(S)')

% % Construct Gabor window and add to time domain plot

tau = 4;
a = 10;
g = exp(-a*(t-tau).^2);

% subplot(2,1,1)
plot(t,S,'k',t,g,'m','Linewidth',2)
title("Recorder: Signal v Time")
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

%% Apply filter and take fft

Sg = g.*S;
Sg_fft = fft(Sg);

% figure(3)
subplot(3,1,1)
plot(t,S,'k','Linewidth',2)
title("Signal v Time & Gabor Window: a = 10, tau = 4",'Fontsize',12)
```

```
hold on
plot(t,g,'m','Linewidth',2)  % shows the filter on the S(t) plot
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

% filtered signal as a function of time
% signal data ~[3, 6],, otherwise flat
subplot(3,1,2)
plot(t,Sg,'k','Linewidth',2)
title("Filtered Signal v Time",'Fontsize',12)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('Sg(t)')

% FFT(S) % as a function of frequency
% way less peaks and dips!
subplot(3,1,3)
plot(ks,abs(fftshift(Sg_fft))/max(abs(Sg_fft)),'r','Linewidth',2);
title("FFT of Filtered Signal v Frequency",'Fontsize',12)
axis([-5e3 5e3 0 1.1])
set(gca,'Fontsize',11)
xlabel('frequency (\omega)'), ylabel('FFT(Sg)')

%% Change window size (filter width) - wider window

tau = 4;
a = 1;
g = exp(-a*(t-tau).^2);
Sg = g.*S;
Sgt = fft(Sg);

% shows how filter widens on the S(t) plot
% figure(4)
subplot(3,1,1)
plot(t,S,'k','Linewidth',2)
title("Signal v Time & Gabor Window: a = 1, tau = 4",'Fontsize',12)
hold on
plot(t,g,'m','Linewidth',2)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

% less accuracy in the time domain
subplot(3,1,2)
plot(t,Sg,'k','Linewidth',2)
title("Filtered Signal v Time",'Fontsize',12)
set(gca,'Fontsize',12), xlabel('Time (t)'), ylabel('Sg(t)')

% more detail in the frequency domain
subplot(3,1,3)
plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
title("FFT of Filtered Signal v Frequency",'Fontsize',12)
axis([-5e3 5e3 0 1.1])
set(gca,'Fontsize',11)
xlabel('frequency (\omega)'), ylabel('FFT(Sg)')

```

```
%% Change window size - narrower window

tau = 4;
a = 100;
g = exp(-a*(t-tau).^2);
Sg = g.*S;
Sgt = fft(Sg);

% % shows how filter widens on the S(t) plot
% figure(5)
subplot(3,1,1)
plot(t,S,'k','Linewidth',2)
title("Signal v Time & Gabor Window: a = 100, tau = 4",'Fontsize',12)
hold on
plot(t,g,'m','Linewidth',2)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

% less accuracy in the time domain
subplot(3,1,2)
plot(t,Sg,'k','Linewidth',2)
title("Filtered Signal v Time",'Fontsize',12)
set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('Sg(t)')

% more detail in the frequency domain
subplot(3,1,3)
plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
title("FFT of Filtered Signal v Frequency",'Fontsize',12)
axis([-5e3 5e3 0 1.1])
set(gca,'Fontsize',11)
xlabel('frequency (\omega)'), ylabel('FFT(Sg)')

%% Animation of shifting window

% figure(6)
a = 1;
tstep=0:0.1:L;

for j=1:length(tstep)
    g=exp(-a*(t-tstep(j)).^2);
    Sg=g.*S;
    Sgt=fft(Sg);

    subplot(3,1,1)
    plot(t,S,'k','Linewidth',2)
    title("Signal v Time & Gabor Window: a = 1, tau = 1",'Fontsize',12)
    hold on
    plot(t,g,'m','Linewidth',2)
    hold off
    set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('S(t)')

    subplot(3,1,2)
```

```
    plot(t,Sg,'k','Linewidth',2)
    title("Signal v Time & Gabor Window: a = 1, tau = 1",'Fontsize',12)
    set(gca,'Fontsize',11), xlabel('Time (t)'), ylabel('Sg(t)')

    subplot(3,1,3)
    plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
    title("FFT of Gabor Filtered Signal v Frequency",'Fontsize',12)
    axis([-5e3 5e3 0 1.1])
    set(gca,'Fontsize',11)
    xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
    drawnow
    pause(0.1)
end

%% Calculate Gabor transform and plot spectrogram

k_Hz= (1/(2*L))*[0:(n/2-1) -n/2:-1];
ks_Hz=fftshift(k_Hz);

a = 50;
tstep=0:0.3:L;
inc = tstep(2)-tstep(1);

Sgt_spec = zeros(length(tstep),n);

for j=1:length(tstep)
    g=exp(-a*(t-tstep(j)).^2);
    Sg=g.*S; % apply new filter
    Sgt=fft(Sg); % transform to frequency domain
    Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
end

% figure(7)
pcolor(tstep,ks_Hz,Sgt_spec.'),
shading interp
title('Mary Had A Little Lamb:  a = 50, time step = 0.5 sec , ','Fontsize',12)
set(gca,'Fontsize',11)
axis([0 L 390 525])
xlabel('Time Steps'), ylabel('Frequency(Hz)')
colormap(hot)

%% View one note and overtones

% max signal in the time domain
[max_S, idx] = max(S);
max_time = t(idx);

% find tstep(j) closest to max signal
above = find(tstep > max_time);
nearest_above = above(1);
```

```
below = find(tstep < max_time);
nearest_below = below(length(below));

% get sample with max signal data
max_sample = Sgt_spec(nearest_below,:);

% full frequency plot
figure(8)
plot(ks_Hz, max_sample);
xlabel('Freqency'); ylabel('Amplitude');
title('Highest Recorder Signal');
axis([0 5e3 0 250])

% zoom in on overtones
figure(9)
plot(ks_Hz, max_sample);
xlabel('Freqency'); ylabel('Amplitude');
title('Recorder Overtones');
axis([0 5e3 0 50])
```