# AMATH 582 Homework 2

Shannon Dow

February, 2020

**Abstract**

In this assignment, time-frequency analysis will be used to analyze two pieces of music and produce spectrograms. Due to the limitations of the Fourier transform, the Gabor transform will be used to retrieve both frequency and time information simultaneously. Unfortunately, to get more resolution in either time or frequency, resolution must be sacrificed in the other. The Gabor transform uses a small, centered filter over the signal to get frequency data at a localized time. The following outlines how experimenting with different filters, filter widths, and sampling coarseness affects the spectrograms. Finally, a score will be produced for the song Mary had a Little Lamb, with a chosen Gabor filter, width, and coarseness.

## 1 Introduction and Overview

### 1.1 Part 1

In part 1, we are analyzing 9 seconds of a piece of music. The goal is to produce a spectrogram of the music, which provides both frequency and time information about the piece. To achieve this, we will be making use of the Gabor Transform or a Short-Time Fourier Transform method. This will involve sliding a localized time filter through the signal and retrieving frequency data.

We want to explore this method by changing the type of filter, filter widths, as well as coarseness of the discretization. All of these will effect the spectrogram, hopefully in predictable ways. I will implement three different types of filters: Gaussian, Mexican Hat, and Shannon.

### 1.2 Part 2

In the second part, we will be performing time frequency analysis on two different versions of Mary Had a Little Lamb. One is played by the piano and the other, a recorder. Using the Gabor transforms, we will compare the analysis of these two instruments. We expect the difference in the analysis will revolve around the timbre, which is the overtones created by the instrument. Additionally, the frequency of the notes played by the instruments will likely be different.

## 2 Theoretical Background

### 2.1 Time Frequency Analysis Using Gabor Transforms

This assignment requires analysis of both time and frequency. While Fourier transforms give good frequency data, they do not give information about when in time these frequencies occur. In the previous assignment, we were only looking for one frequency signature, which could be located using averaging and maximizing. This highlights the limitations of the Fourier transform, as it is really only powerful if we want to focus on one stationary frequency. Since we will be analyzing music in hopes of producing a score, there are changes in both time and frequency. To get time and frequency resolution, we will be making use of the Gabor Transform or a Short-Time Fourier Transform method.

Gabor made a small change to the Fourier kernel:

$$g_{\tau,\omega}(\tau) = e^{i\omega\tau} g(\tau - t)$$

The idea is to create a local time filter (g), centered around $\tau$ and slide the filter across the entire signal. This pulls out frequency content at all of the times. The transform involves an integral across all the values of $\tau$. In practice, we will be using the discrete Gabor transform where $\tau = nt_0$. and $\nu = m\omega_0$.

$$g_{m,n} = e^{i2\pi m\omega_0 t} g(t - nt_0)$$

When you have both time and frequency data, you can produce a spectrogram of the data.

## 2.2 Limitations of Gabor

To obtain both time and frequency information simultaneously, the Gabor transform does sacrifice some resolution in both these variables. When filtering, any signal data that is outside of the filter is lost. Additionally, the size or width (a) of the filter is key in deciding how much resolution is lost in either time or frequency. With a large filter, you pick up more of the frequency data, so you will get sharper resolution in frequency, but you will not know the exact time of these frequencies. Conversely, a very narrow filter will give excellent resolution in time, but will lose the sharpness in frequency. This demonstrates Heisenberg uncertainty principle. You cannot have exact information about both time and frequency simultaneously. Depending on the application and goals of the analysis, a different size filter can be used.

## 2.3 Types of Gabor Filters

### 2.3.1 The Gaussian Gabor Window:

$$g(t) = e^{-a(t-b)^2}$$

The parameters a vary the width and the center of the Gaussian window respectively.

### 2.3.2 The Mexican Hat Gabor Window:

$$mh(t) = (1 - a(t-b)^2)e^{-\frac{1}{2}a(t-b)^2}$$

The parameters, similar to the Gaussian vary the width(a) and the center(b).

Both the Gaussian and Mexican Hat windows are smooth, so sometimes a window with corners can pick up different results, depending on the signal.

### 2.3.3 The Shannon Step Function Window:

This just creates a box of desired height and width

$$sf(t) = \begin{cases} h & t - a \leq t \leq t + a \\ 0 & otherwise \end{cases}$$

Where a is again the width parameter and h is the height

# 3 Algorithm Implementation and Development

## 3.1 Part 1

For this section, we are performing time-frequency analysis on nine seconds of the song: Handel's Messiah. We first want to discretize both the time and frequency domains. Since V is our signal and Fs is the sampling rate, we get that `t = (1:length(v))/Fs`. `k = (1/L)*[0:(n/2) -n/2:-1]` where L is the number of t values and we choose these points since we have an odd number of points.

In this assignment, I will explore three different Gabor windows: Gaussian, Mexican Hat, and Shannon. In each, I will vary the width of the window and the coarseness of the sampling. This will involve changing the n term from above, how many time steps you use in the discretization. We will then see how these things affect the spectrograms.

### 3.2 Creating a Sliding Window

Using two `for` loops, one to multiply the signal by the filter at each discretization, and one to test and plot different filter widths, we can create multiple spectrograms of the data. First, a list is created to store the transformed and shifted signal values: `vgt_spec`. The variable `tslide` is what discretizes the time interval, to decide how many times you multiply by the filter window. Within the loop for each value of `tslide`, the signal is multiplied by a filter centered around each step of `tslide`. Then, the frequency data in that window is determined by taking the Fourier transform (`fft`). The frequency data is shifted to match the nodes and stored in the list. The spectrograms are then plotted using `pcolor` so that we can analyze frequency and time data.

**Algorithm 1:** Sliding Gabor Window

**for** Each discretization of tslide **do**
  create centered window
  multiply filter by signal
  Fourier transpose the result
  Add the absolute value of shifted version to a storage vector
**end for**

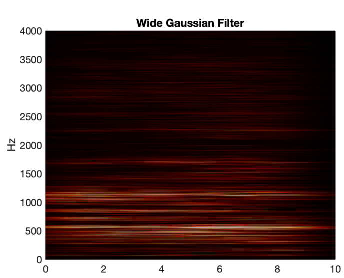
To experiment with coarseness, I changed the step within `tslide`, to change the sampling rate. It is important to note that when adjusting the width of both the Gaussian and Mexican Hat filters, a large value of the width parameter results in a narrow filter and vice versa.

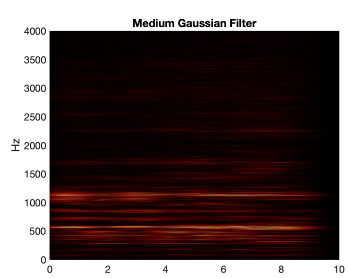
### 3.3 Part 2

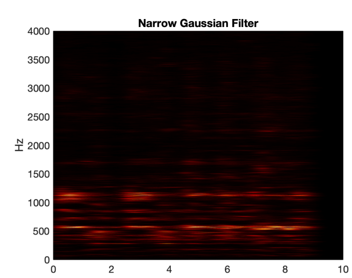
In this section, we are analyzing the tune Mary had a Little Lamb played by both a piano and a recorder. For both songs, they are being read by MATLAB using `audioread()` producing a vector of the song signal data and the sampling rate. While similar, the discretization is different for this since we have an even number of points: `k = (1/L)*[0:(n/2)-1 -(n/2):-1]` To produce a spectrogram of both songs, we will be using the Gabor transform, as in part one. I will be using the gaussian Gabor filter window with a medium length to retrieve both time and frequency data with resolution.

## 4 Computational Results

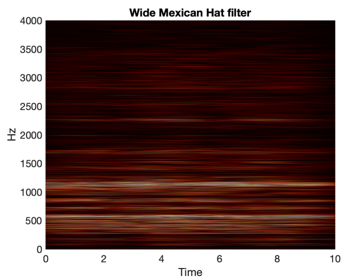
GAUSSIAN WINDOW: As you make the filter wider, you pick up more frequency data. This makes it blur horizontally. The smaller the filter, the more time resolution, but frequency is less clear.

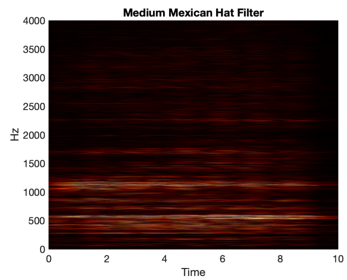


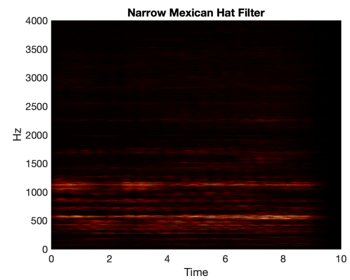



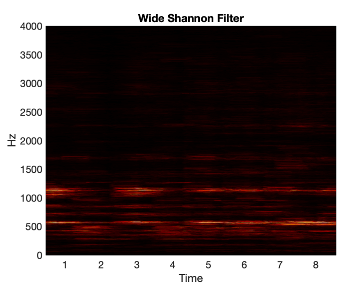
MEXICAN HAT WINDOW: Using similar widths for the Mexican Hat window, the spectrograms are also similar. However, the Gaussian seems more clear.

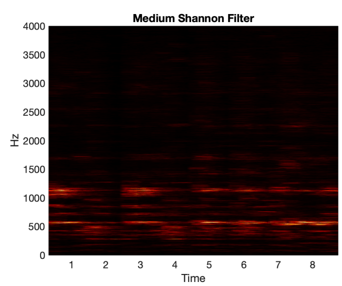


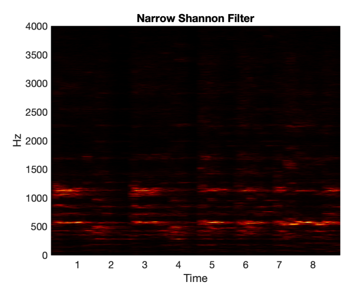



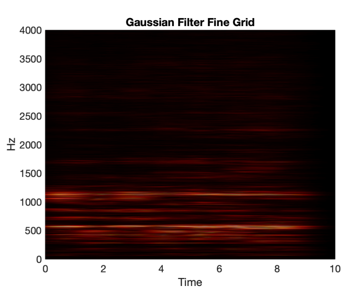

SHANNON STEP WINDOW: The Shannon filter has sharper edges. It picks up changes in frequency.

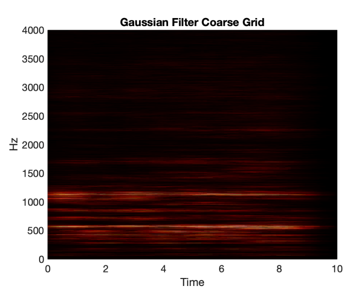


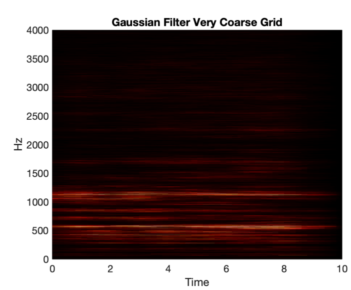




The following three spectrograms represent changes to the coarseness, or number of times the gabor window is moved along and evaluated. These show various changes to the sampling rate. The coarser the grid, larger value of the step within `tslide`. The spectrogram on the left shows a very fine grid with over-sampling and on the left a large grid with under-sampling.



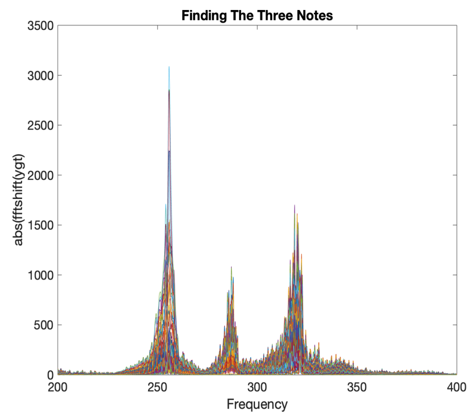



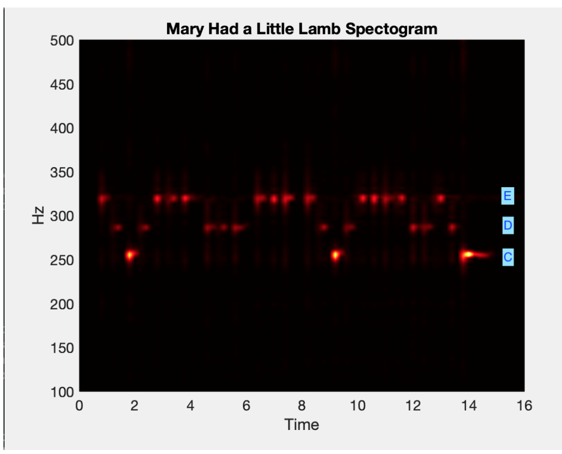
## 4.1 Part 2: Mary Had a Little Lamb

### 4.1.1 Piano

Plotting the song data in the Fourier domain shows that there are three main frequencies that produce the three notes from the spectrogram. Using estimation and the diagram, we can determine that the three notes are approximately:

- E: 329.63 Hz
- D: 293.66 Hz
- C: 261.63 Hz

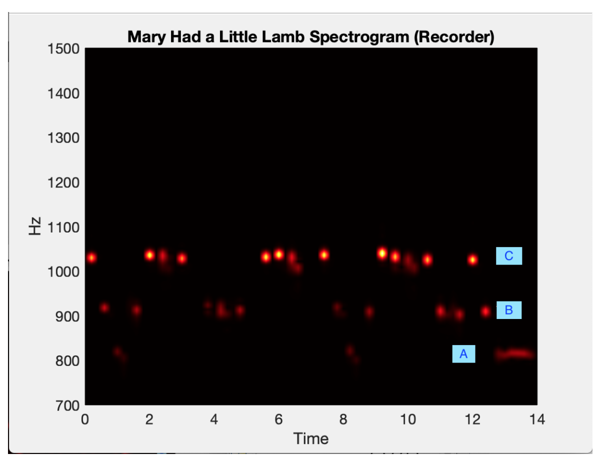



This is the spectrogram for the piano. Using techniques from part one, I used a Gaussian Gabor window. Then, I chose a width for the window and timestep for the sampling to produce a clear spectrogram. With a width parameter of 100 and sampling rate of 0.2, you can clearly see the different notes being played at different times. While there are some visible overtones, with the cropped range, they do not distract from the results shown.

This is the spectrogram for the recorder, playing the same song. The recorder plays at a higher frequency. With a width parameter of 100 and sampling rate of 0.2 again, I was able to produce an adequate spectrogram. The recorder has fewer visible overtones, likely due to the difference in how sound is produced compared to a piano. Matching the frequencies to notes, the three notes being played are:

- C: 1046.5 Hz
- B: 987.77 Hz
- A: 880.00 Hz



# 5 Summary and Conclusions

## 5.1 Part 1

After experimenting with different filters with different widths, I was able to find a medium sized filter that was a compromise between time and frequency resolution. As anticipated, a wide filter blurs the data horizontally. This gives us good frequency data, but sacrifices clear time information. With music, this means that we are more clear on which note is being played, but not necessarily when. The narrower the filter, the timing the notes are played becomes more clear, however, since we compromise some frequency resolution, we lose clarity on the note being played. In all, the Shannon, or step function filter gave the best information as to what was going on. Since the notes change, it goes directly from one frequency to the next. The step function is not continuous and therefore has edges. It is easier to see the breaks in between the notes being played. As far as the sampling rate goes, there is not a large difference between a very fine sampling rate and relatives larger one. Since this requires almost double the computations, or runs through the loop, and does not add much information, it probably unnecessary. However, the very coarse grid does loose some resolution in both time and frequency. These can be adjusted based on need and computational power.

## 5.2 Part 2

After experimenting with different parameters for the Gabor filters for both the piano and recorder data, I was able to produce spectrograms for both pieces. In my results, there were three notes in each piece, however the recorder played notes at a much higher frequency, which was an expected result due to the instrument. In the case for the piano, although difficult to see in a small picture and partially cut out of the range, there were more overtones. This was shown by some color at the top of the spectrogram. Since the piano creates sound differently than a recorder, it is likely due to the timbre of the piano. The piano spectrogram is also more consistent in time. This is probably because the note played is less dependent on user. Every time you hit the key, it plays the note, where the recorder player could put in different amount of breath at each note.

# Appendix A MATLAB Functions

- `fft()` This is the fast Fourier transform of the data. It transforms signal data into frequency data.
- `fftshift()` This shifts the domain of the Fourier transform so that it is more readable.
- `pcolor` This is used to make a spectrogram. It takes in both the time and frequency data to produce picture of the relationship between the data.
- `[y,Fs] = audioread()` This takes in an audio file and produces a vector y of the sampled data as well as the sampling rate Fs.

# Appendix B MATLAB Code

## B.1 Github Link

https://github.com/shannondow/AMATH-582-Homeworks-2020

```
close all; clear all; clc
%% HOMEWORK 2:
%This code is divided into 5 sections:
%Gaussian Gabor Window
%Mexican Hat Window
%Shannon Step Window
%Piano Spectrogram
```

```matlab
%Recorder Spectrogram
%% Gaussian Gabor Window
%Plots the portion of Handel's Messiah to analyze (9 seconds)
load handel
v = y'/2;
figure(1)
subplot(2,1,1)
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

%Plays recording
%p8 = audioplayer(v,Fs);
%playblocking(p8);

%Frequency domain

n = length(v);
t = (1:n)/Fs;
L = t(end);
k = (1/L)*[0:(n-1)/2 -(n-1)/2:-1]; ks = fftshift(k);
vt = fft(v);
subplot(2,1,2)
plot((ks),abs(fftshift(vt))/max(abs(vt)));
xlabel('frequency (\omega)'), ylabel('FFT(V)')
title('abs(fftshift(vt))/max(abs(vt))');

%%Gaussian Gabor Window
figure(2)
width=[10 1 0.2];
for j=1:3
    g1=exp(-width(j)*(t-4).^2);
    subplot(3,1,j)
    plot(t,v,'k'), hold on
    plot(t,g1,'k','Linewidth',[2])
    set(gca,'Fontsize',[14])
    ylabel('V(t), g(t)')

end
xlabel('time(t)')

% Gabor Transform
figure(3)

g=exp(-2*(t-4).^2);
vg=g.*v; vgt=fft(vg);

subplot(3,1,1), plot(t,v,'k'), hold on
plot(t,g,'k','Linewidth',[2])
set(gca,'Fontsize',[14])
ylabel('v(t), g(t)'), xlabel('time (t)')

subplot(3,1,2), plot(t,vg,'k')
```

```matlab
set(gca,'Fontsize',[14])
ylabel('v(t)g(t)'), xlabel('time (t)')

subplot(3,1,3), plot(ks,abs(fftshift(vgt))/max(abs(vgt)),'k')
set(gca,'Fontsize',[14])
ylabel('FFT(vg)'), xlabel('frequency (\omega)')

% SLIDING GABOR WINDOW

vgt_spec=[];
tslide=0:1:10;
a = [0.5,2,10];
for i = 1:length(a)
for j=1:length(tslide)
    g=exp(-a(i)*(t-tslide(j)).^2);
    vg=g.*v;
    vgt=fft(vg);
    vgt_spec=[vgt_spec;
    abs(fftshift(vgt))];
end
figure(3+i)
pcolor(tslide,ks,vgt_spec.'),
shading interp
colormap(hot)
vgt_spec=[];
xlabel('Time');
ylabel('Hz')
ylim([0,4000]);
end

%% Mexican Hat Filter:
close all; clear all; clc
%Plots the portion of Handel's Messiah to analyze (9 seconds)
load handel
v = y'/2;
n = length(v);
t = (1:n)/Fs;
L = t(end);
k = (1/L)*[0:(n/2) -n/2:-1]; ks = fftshift(k);

%Mexican Hat Window
width=[10 1 0.2];
figure(1)
for j=1:3
    g2=(1-width(j)*(t-4).^2).*exp(-0.5.*width(j)*(t-4).^2);
    subplot(3,1,j)
    plot(t,v,'k'), hold on
    plot(t,g2,'k','Linewidth',[2])
    set(gca,'Fontsize',[14])
    ylabel('V(t), g(t)')
end
xlabel('time (t)')
```

```
% SLIDING GABOR WINDOW

vgt_spec=[];
tslide=0:0.1:10;
a = [0.3,2,5,15];
for i = 1:length(a)
for j=1:length(tslide)
    g =(1-a(i)*(t-tslide(j)).^2).*exp(-0.5.*a(i)*(t-tslide(j)).^2);
    vg=g.*v;
    vgt=fft(vg);
    vgt_spec=[vgt_spec;
    abs(fftshift(vgt))];
end
%Create Spectrograms
figure(3+i)
pcolor(tslide,ks,vgt_spec.'),
shading interp
colormap(hot)
xlabel('Time');
ylabel('Hz')
ylim([0,4000]);
vgt_spec=[];
end

%% Shannon Step Filter:
close all; clear all; clc
%Plots the portion of Handel's Messiah to analyze (9 seconds)
load handel
v = y'/2;
n = length(v);
t = (1:n)/Fs;
L = t(end);
k = (1/L)*[0:((n-1)/2) -(n-1)/2:-1]; ks = fftshift(k);

width=1000;
zt_spec=[];
j = width;
tslide = [];
while j< n-width
Zs=zeros(1,n);
mask=ones(1,2*width);
%tstepmult=0;
%tstep=round((tstepmult+1)*tslide(j));
Zs(j+1-width:j+width)= mask;
Zsf = Zs.*v;
Zsft = fft(Zsf);
zt_spec=[zt_spec;
    abs(fftshift(Zsft))];
j = j+800;
tslide = [tslide;j];
end
tslide = tslide./Fs;
pcolor(tslide,ks,zt_spec.'),
shading interp
```

```
colormap(hot)
xlabel('Time');
ylabel('Hz')
ylim([0,4000]);

%% Part 2: Piano

close all, clear all, clc;
 tr_piano=16;  % record time in seconds
 y= audioread('music1.wav'); Fs=length(y)/tr_piano;
 figure(1)
 subplot(2,1,1); plot((1:length(y))/Fs,y);
 xlabel('Time [sec]'); ylabel('Amplitude');
 title('Mary had a little lamb (piano)');  drawnow
 p8 = audioplayer(y,Fs); playblocking(p8);

%Discretize time and frequency domain:
n = length(y);
t = (1:n)/Fs;
L = t(end);
k = (1/L)*[0:(n/2)-1 -(n/2):-1]; ks = fftshift(k);
yt = fft(y);
y = y';
%Plot in the frequency domain:
%subplot(2,1,2); plot((ks),abs(fftshift(yt))/max(abs(yt)));
%xlabel('frequency (\omega)'), ylabel('FFT(y)')
%title('abs(fftshift(yt))/max(abs(yt))');
% Gabor Transform
figure(2)

g=exp(-100*(t-4).^2);
yg=g.*y; ygt=fft(yg);

subplot(3,1,1), plot(t,y,'k'), hold on
plot(t,g,'k','Linewidth',[2])
set(gca,'Fontsize',[14])
ylabel('v(t), g(t)'), xlabel('time (t)')

subplot(3,1,2), plot(t,yg,'k')
set(gca,'Fontsize',[14])
ylabel('v(t)g(t)'), xlabel('time (t)')

figure(3);
subplot(2,1,1); plot(ks,abs(fftshift(ygt))/max(abs(ygt)),'k');
hold on;

%Create a Gaussian gabor filter:
ygt_spec=[];
tslide=0:0.2:16;
for j=1:length(tslide)
    g=exp(-(100)*(t-tslide(j)).^2);
    yg=g.*y;
    ygt=fft(yg);
    ygt_spec=[ygt_spec;
```

```
    abs(fftshift(ygt))];

end

%Make Piano Spectrogram
figure(4)
pcolor(tslide,ks,ygt_spec.')
shading interp
colormap(hot)
set(gca,'Ylim',[100,500],'fontsize',[14]);
vgt_spec=[];
xlabel('Time');
ylabel('Hz');
title('Mary Had a Little Lamb Spectrogram (Piano)');

%% Part 2: Recorder
tr_rec=14;  % record time in seconds
 y2=audioread('music2.wav'); Fs=length(y2)/tr_rec;
 subplot(2,1,1); plot((1:length(y2))/Fs,y2);
 xlabel('Time [sec]'); ylabel('Amplitude');
 title('Mary had a little lamb (recorder)');
 %p8 = audioplayer(y,Fs); playblocking(p8);

%Discretize time and frequency domain:
n = length(y2);
t2 = (1:n)/Fs;
L = t2(end);
k = (1/L)*[0:(n/2)-1 -(n/2):-1]; ks = fftshift(k);
y2t = fft(y2);
y2 = y2';

%Plot in the frequency domain:
subplot(2,1,2); plot((ks),abs(fftshift(y2t))/max(abs(y2t)));
xlabel('frequency (\omega)'), ylabel('FFT(y)')
title('abs(fftshift(yt))/max(abs(yt))');


y2gt_spec=[];
tslide=0:0.2:14;
for j=1:length(tslide)
    g=exp(-100*(t2-tslide(j)).^2);
    y2g=g.*y2;
    y2gt=fft(y2g);
    y2gt_spec=[y2gt_spec;
    abs(fftshift(y2gt))];
end
%Create Recorder Spectrogram
figure(4)
 pcolor(tslide,ks,y2gt_spec.'),
shading interp
set(gca,'Ylim',[700,1500],'fontsize',[14]);
colormap(hot)
vgt_spec=[];
ylabel('Hz');
```

```
xlabel('Time');
title('Mary Had a Little Lamb Spectrogram (Recorder)');
```