

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.

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

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 consider smaller increments of time within our data using a Gábor filter.

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

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.

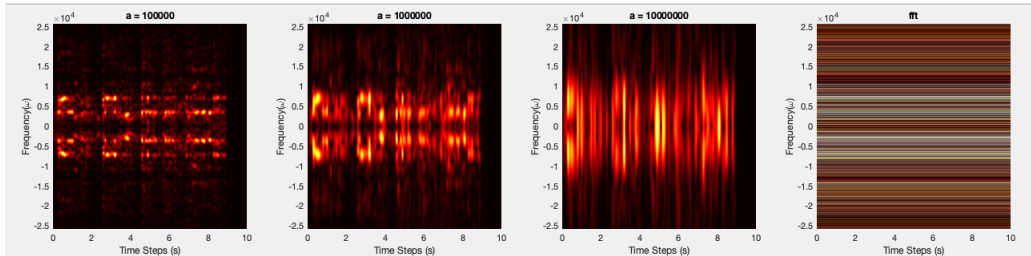


Figure 1: Effects of Changing Width of Gábor Window on 'Messiah' Spectrogram

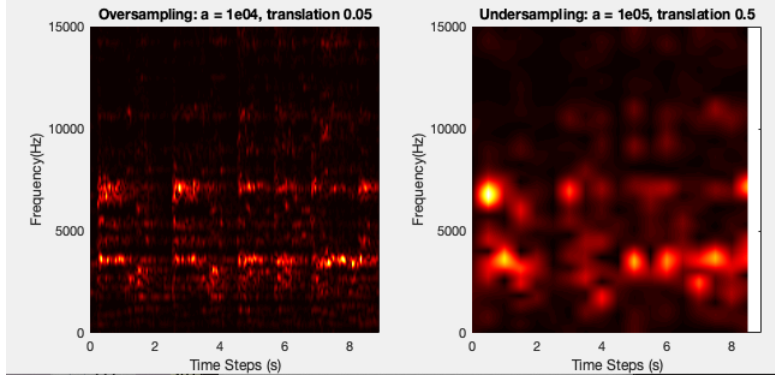


Figure 2: Oversampled and Undersampled Data from Handel's 'Messiah'

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 and 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)

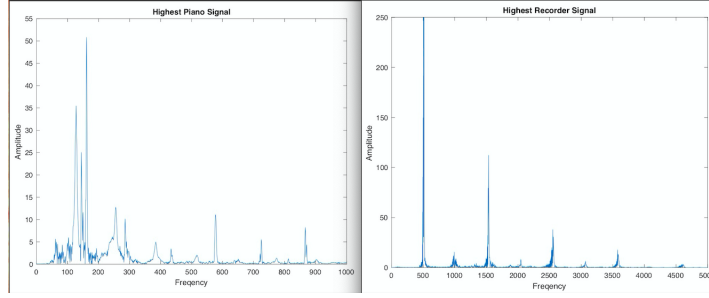


Figure 3: Unfiltered Signal Frequencies of a Piano and a Recorder

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

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

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 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 the 2π to 1 to represent the units Hz, which are cycles per second.

IV. Computational Results & Supplementary Plots

Figure 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.

Piano



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

Recorder



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.

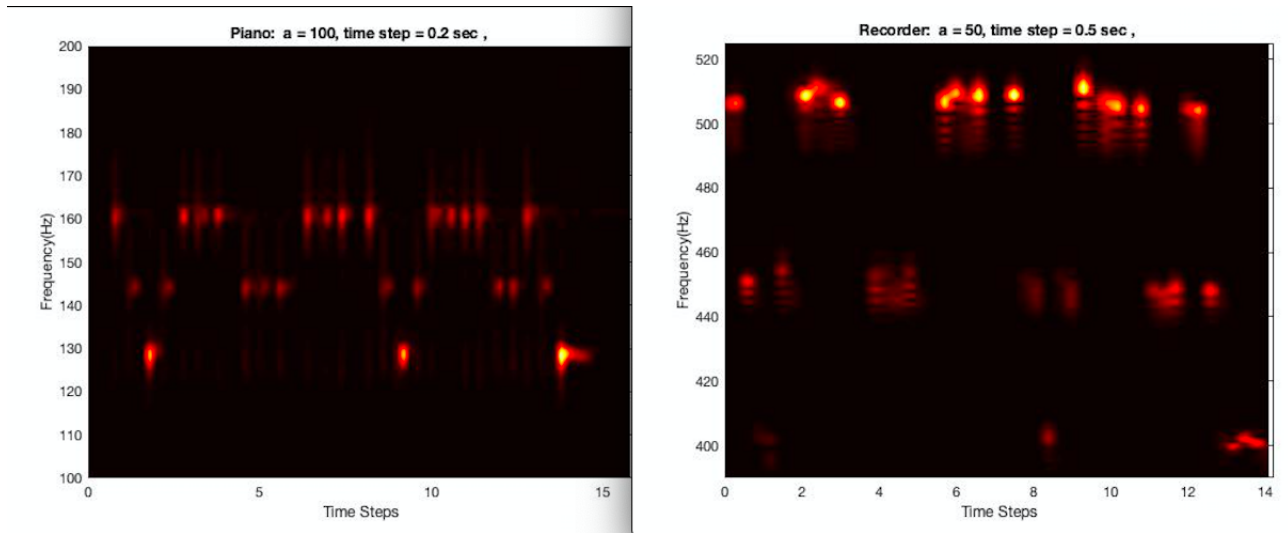


Figure 6: 'Mary Had a Little Lamb' Spectrograms of Piano and Recorder Audio

Appendix B: Matlab Code

```
1
2 %% HW 02 Part 1
3
4 % Construct signal and plot signal and FFT
5
6 close all; clear all; clc;
7
8 load handel
9 % y = audio signal
10 S = y'; % transpose y vector
11 n=length(S); % number of samples
12 L=n/Fs; % time domain
13 t2=linspace(0,L,n+1);
14 t=t2(1:n);
15 k=(2*pi/2*L)*[0:(n-1)/2 -(n-1)/2:-1]; % frequency vector
16 ks=fftshift(k);
17
18 % figure(1)
19 plot((1:length(y))/Fs,S)
20 xlabel('Time [sec]')
21 ylabel('Amplitude')
22 title('Signal as a function of time')
23 % p8 = audioplayer(S,Fs);
24 % playblocking(p8);
25
26 St=fft(S);
27
28 % figure(2)
29 subplot(2,1,1)
30 % plot the function against time
31 plot(t,S,'k','Linewidth',2)
32 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
33
34 subplot(2,1,2)
35 % plot FFT against (normalized) frequency
36 plot(ks,abs(fftshift(St))/max(abs(St)),'r','Linewidth',2);
37 title("Signal v Frequency",'FontSize',12)
38 set(gca,'FontSize',11)
39 xlabel('frequency (\omega)'), ylabel('FFT(S)')
40
41 % Construct Gabor window and add to time domain plot
42 tau = 4;
43 a = 100; % width of filter
44 g = exp(-a*(t-tau).^2); % the filter
45 subplot(2,1,1)
46 plot(t,S,'k',t,g,'m','Linewidth',2)
47 title("Signal v Time & Gabor Window: a = 100, tau = 4",'FontSize',12)
48 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
49
```

```

50 %% Apply filter and take fft
51
52 a = 100; % width of filter
53 g = exp(-a*(t-tau).^2); % the filter
54 Sg = g.*S;
55 Sgt = fft(Sg);
56
57 % figure (3)
58 subplot(3,1,1)
59 % subplot(3,2,2)
60 plot(t,S,'k','Linewidth',2)
61 axis([2 6 -1 1])
62 title("Signal v Time & Gabor Window: a = 100",'FontSize',12)
63 hold on
64 plot(t,g,'m','Linewidth',2) % shows the filter on the S(t) plot
65 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
66
67 subplot(3,1,2)
68 % subplot(3,2,4)
69 plot(t,Sg,'k','Linewidth',2)
70 axis([2 6 -1 1])
71 title("Gabor Filtered Signal v Time",'FontSize',12)
72 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('Sg(t)')
73
74
75 subplot(3,1,3)
76 % subplot(3,2,6)
77 plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2)
78 title("FFT of Gabor Filtered Signal v Frequency",'FontSize',12)
79 set(gca,'FontSize',11)
80 xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
81
82
83 a = 1; % width of filter
84 g = exp(-a*(t-tau).^2); % Gaussian, but can try Shannon or Mexican hat
85 Sg = g.*S;
86 Sgt = fft(Sg);
87
88 figure(4)
89 subplot(3,1,1)
90 % subplot(3,2,1)
91 plot(t,S,'k','Linewidth',2)
92 axis([2 6 -1 1])
93 title("Signal v Time & Gabor Window: a = 1",'FontSize',12)
94 hold on
95 plot(t,g,'m','Linewidth',2)
96 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
97
98 % less accuracy in the time domain
99 subplot(3,1,2)
100 % subplot(3,2,3)

```



```

101 plot(t,Sg,'k','Linewidth',2)
102 axis([2 6 -1 1])
103 title("Gabor Filtered Signal v Time",'FontSize',12)
104 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('Sg(t)')
105
106 % more detail in the frequency domain
107 subplot(3,1,3)
108 % subplot(3,2,5)
109 plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
110 title("FFT of Gabor Filtered Signal v Frequency",'FontSize',12)
111 set(gca,'FontSize',11)
112 xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
113
114
115 %% Change window size - narrower window
116 % larger a, less accuracy in frequency, more accuracy in time
117 a = 1000;
118 g = exp(-a*(t-tau).^2);
119 Sg = g.*S;
120 Sgt = fft(Sg);
121
122 figure(5)
123 subplot(3,1,1)
124 plot(t,S,'k','Linewidth',2)
125 axis([2 6 -1 1])
126 title("Signal v Time & Gabor Window: a = 1,000",'FontSize',12)
127 hold on
128 plot(t,g,'m','Linewidth',2)
129 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
130
131 % less accuracy in the time domain
132 subplot(3,1,2)
133 plot(t,Sg,'k','Linewidth',2)
134 axis([2 6 -1 1])
135 title("Gabor Filtered Signal v Time",'FontSize',12)
136 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('Sg(t)')
137
138 % more detail in the frequency domain
139 subplot(3,1,3)
140 plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
141 title("FFT of Gabor Filtered Signal v Frequency",'FontSize',12)
142 set(gca,'FontSize',11)
143 xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
144
145 %% Animation of shifting window
146
147 a = 1;
148 tstep=0:0.1:100;
149
150 % figure(5)
151 for j=1:length(tstep) % [1:1001]

```

```

152     g=exp(-a*(t-tstep(j)).^2);
153     Sg=g.*S;
154     Sgt=fft(Sg);
155
156     subplot(3,1,1)
157     plot(t,S,'k','Linewidth',2)
158     hold on
159     plot(t,g,'m','Linewidth',2)
160     title("Signal v Time & Gabor Window: a = 1",'FontSize',12)
161     hold off
162     set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
163
164     subplot(3,1,2)
165     plot(t,Sg,'k','Linewidth',2)
166     title("Gabor Filtered Signal v Time",'FontSize',12)
167     set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('Sg(t)')
168
169     subplot(3,1,3)
170     plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
171     title("FFT of Gabor Filtered Signal v Frequency",'FontSize',12)
172     set(gca,'FontSize',11)
173     xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
174     drawnow
175     pause(0.1)
176 end
177
178 %% Calculate Gabor transform and plot spectrogram
179 a = 1e03;
180 tstep = 0:0.04:L;
181
182 Sgt_spec = zeros(length(tstep),n);
183
184 for j=1:length(tstep) % [1:101]
185     g=exp(-a*(t-tstep(j)).^2);
186     Sg=g.*S; % apply new filter
187     Sgt=fft(Sg); % transform to frequency domain
188     Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
189 end
190
191 figure(6)
192 pcolor(tstep,ks,Sgt_spec'),
193 shading interp
194 title('Unfiltered: a = 1e03, inc = 0.0446','FontSize',16)
195 set(gca,'FontSize',11)
196 axis([0 L 3e03 8e03])
197 colormap(hot)
198
199 %% Oversampling and Undersampling
200
201 a = 1e04;
202 tstep = 0:0.05:L;

```

```

203
204 Sgt_spec = zeros(length(tstep),n);
205
206 for j=1:length(tstep) % [1:101]
207     g=exp(-a*(t-tstep(j)).^2);
208     Sg=g.*S; % apply new filter
209     Sgt=fft(Sg); % transform to frequency domain
210     Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
211 end
212
213 figure(7)
214 subplot(1,2,1)
215 pcolor(tstep,ks,Sgt_spec. '),
216 shading interp
217 title('Oversampling: a = 1e04, translation 0.05','FontSize',16)
218 set(gca,'FontSize',11)
219 axis([0 L 0 1.5e04])
220 colormap(hot)
221
222
223 a = 1e05;
224 tstep = 0:.5:L;
225
226 Sgt_spec = zeros(length(tstep),n);
227
228 for j=1:length(tstep) % [1:101]
229     g=exp(-a*(t-tstep(j)).^2);
230     Sg=g.*S; % apply new filter
231     Sgt=fft(Sg); % transform to frequency domain
232     Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
233 end
234
235 % figure(6)
236 subplot(1,2,2)
237 pcolor(tstep,ks,Sgt_spec. '),
238 shading interp
239 title('Undersampling: a = 1e05, translation 0.5','FontSize',16)
240 set(gca,'FontSize',11)
241 axis([0 L 0 1.5e04])
242 colormap(hot)
243
244 %% Filter the signal data in the frequency domain using a Gaussian
    filter
245 %
246 Sgt_spec_filtered = zeros(length(tstep),n);
247 tau = .5e-05;
248 it = length(tstep);
249
250 for j=1:it % [1:201]
251     test = Sgt_spec(j,:);
252     % get the max signal and index

```

```

253     [test_M, test_I] = max(test);
254     % get frequency associated with max signal
255     % use ks because we built Sgt_spec with fftshift
256     test_center_freq = abs(ks(test_I));
257     test_filter = exp(-tau*(ks-test_center_freq).^2);
258     test_f = test_filter.*test; % Apply the filter to the signal in
        frequency space
259     Sgt_spec_filtered(j,:) = fftshift(test_f); % We don't want to scale
        it
260
261
262     figure(7)
263     subplot(2,1,1)
264     plot(abs(ks), test, 'r', 'Linewidth', 2);
265     title("Unfiltered Signal v Frequency at timestep = " + num2str(j), '
        Fontsize', 12)
266     set(gca, 'Fontsize', 11), xlabel('Frequency'), ylabel('Signal')
267     drawnow
268
269     subplot(2,1,2)
270     plot(abs(ks), test_f, 'r', 'Linewidth', 2);
271     title("Filtered Signal v Frequency at timestep = " + num2str(j), '
        Fontsize', 12)
272     set(gca, 'Fontsize', 11), xlabel('Frequency'), ylabel('Signal')
273     drawnow
274     pause(2)
275 end
276
277 figure(8)
278 pcolor(tstep, ks, Sgt_spec_filtered. '),
279 shading interp
280 title('Unfiltered: a = 1e06, translation = 8.9249e-02', 'Fontsize', 16)
281 set(gca, 'Fontsize', 11)
282 colormap(hot)
283
284 %% Spectrograms for varying translation sizes
285 figure(9)
286
287 a = 1e5;
288 tstep_vec = [1 0.5 0.1 0.05];
289 for jj = 1:length(tstep_vec)
290
291     tstep = 0:tstep_vec(jj):10;
292     Sgt_spec = zeros(length(tstep), n);
293
294     for j=1:length(tstep)
295         g=exp(-a*(t-tstep(j)).^2);
296         Sg=g.*S;
297         Sgt=fft(Sg);
298         Sgt_spec(j,:) = fftshift(abs(Sgt));
299     end

```

```

300
301     subplot(2,2,jj)
302     pcolor(tstep,ks,Sgt_spec. '),
303     shading interp
304     title(['a = 1e5, tstep = ',num2str(tstep_vec(jj))], 'FontSize',12)
305     set(gca, 'FontSize',11)
306     colormap(hot)
307 end
308
309 % plot of the FFT, we know it has no time info
310 figure(10)
311 Sgt_spec = repmat(fftshift(abs(St)),length(tstep),1);
312 pcolor(tstep,ks,Sgt_spec. '),
313 shading interp
314 title('fft', 'FontSize',11)
315 set(gca, 'FontSize',11)
316 colormap(hot)
317
318 %% Spectrograms for varying window sizes
319 figure(11)
320
321 a_vec = [1e5 1e6 1e7];
322 for jj = 1:length(a_vec)
323     a = a_vec(jj);
324     tstep=0:0.1:10;
325     Sgt_spec = zeros(length(tstep),n);
326     for j=1:length(tstep)
327         g=exp(-a*(t-tstep(j)).^2);
328         Sg=g.*S;
329         Sgt=fft(Sg);
330         Sgt_spec(j,:) = fftshift(abs(Sgt));
331     end
332
333     subplot(2,2,jj)
334     pcolor(tstep,ks,Sgt_spec. '),
335     shading interp
336     title(['a = ',num2str(a)], 'FontSize',12)
337     set(gca, 'FontSize',11)
338     colormap(hot)
339 end
340
341 Sgt_spec = repmat(fftshift(abs(St)),length(tstep),1);
342 subplot(2,2,4)
343 pcolor(tstep,ks,Sgt_spec. '),
344 shading interp
345 title('fft', 'FontSize',12)
346 set(gca, 'FontSize',11)
347 colormap(hot)
348
349 %% Rescale our Gabor transform spectrograms to have the same units for
    frequency

```

```

350 figure(19)
351
352 % k is in terms of 2pi/sec
353 % Hz is in terms of period/sec
354 % both are frequencies, so change the 2pi in k, to 1
355  $k = (1 * \pi / L) * [0 : (n-1)/2 \quad -(n-1)/2 : -1];$  % frequency vector
356  $ks = \text{fftshift}(k);$ 
357  $tstep = 0:L/100:L;$ 
358
359 % Spectrograms for varying a
360  $a\_vec = [5 \quad 1 \quad 0.2];$ 
361 for  $jj = 1:\text{length}(a\_vec)$ 
362      $a = a\_vec(jj);$ 
363      $tstep = 0:0.1:10;$ 
364      $Sgt\_spec = \text{zeros}(\text{length}(tstep), n);$ 
365     for  $j = 1:\text{length}(tstep)$ 
366          $g = \exp(-a * (t - tstep(j)).^2);$ 
367          $Sg = g * S;$ 
368          $Sgt = \text{fft}(Sg);$ 
369          $Sgt\_spec(j, :) = \text{fftshift}(\text{abs}(Sgt));$ 
370     end
371
372     subplot(2,2,jj)
373     pcolor( $tstep, ks, Sgt\_spec.$ '),
374     shading interp
375     title([ $'a = '$ , num2str(a)], 'FontSize', 12)
376     set(gca, 'Ylim', [-10 10], 'FontSize', 11)
377     colormap(hot)
378 end
379
380
381  $Sgt\_spec = \text{repmat}(\text{fftshift}(\text{abs}(St)), \text{length}(tstep), 1);$ 
382 subplot(2,2,4)
383 pcolor( $tstep, ks, Sgt\_spec.$ '),
384 shading interp
385 title('fft', 'FontSize', 12)
386 set(gca, 'Ylim', [-10 10], 'FontSize', 11)
387 colormap(hot)
388
389 %% HW 02 Part 2 Recorder
390
391 close all; clear all; clc;
392
393
394  $[y, Fs] = \text{audioread}('music2.wav');$ 
395  $S = y';$  % transpose the vector
396  $T = 1/Fs;$  % period (second per sample)
397  $L = \text{length}(S)/Fs;$  % record time in seconds
398
399
400 figure(1)

```

```

401     plot((1:length(S))/Fs,S);
402     xlabel('Time [sec]'); ylabel('Amplitude');
403     title('Mary had a little lamb (recorder)');
404     % p8 = audioplayer(y,Fs); playblocking(p8);
405
406     n=length(S); % data points
407     t2=linspace(0,L,n+1); %
408     t=t2(1:n); % our time vector matching the number of signal data points
409
410     k= (2*pi/(2*L))*[0:(n/2-1) -n/2:-1];
411     ks=fftshift(k);
412
413     %% Fourier transform of the function
414     % Signal in frequency domain
415     S_fft=fft(S);
416
417     % figure(2)
418     % plot the function against time
419     subplot(2,1,1)
420     plot(t,S,'k','Linewidth',2)
421     axis([0 L 0 1.1])
422     set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
423
424
425     subplot(2,1,2)
426     % plot (percentage of?)function against frequency
427     plot(ks,abs(fftshift(S_fft))/max(abs(S_fft)),'r','Linewidth',2);
428     title("Signal v Frequency")
429     axis([-5e3 5e3 0 1.1])
430     set(gca,'FontSize',11), xlabel('frequency (\omega)'), ylabel('FFT(S)')
431
432     %% Construct Gabor window and add to time domain plot
433
434     tau = 4;
435     a = 10;
436     g = exp(-a*(t-tau).^2);
437
438     % subplot(2,1,1)
439     plot(t,S,'k',t,g,'m','Linewidth',2)
440     title("Recorder: Signal v Time")
441     set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
442
443     %% Apply filter and take fft
444
445     Sg = g.*S;
446     Sg_fft = fft(Sg);
447
448     % figure(3)
449     subplot(3,1,1)
450     plot(t,S,'k','Linewidth',2)
451     title("Signal v Time & Gabor Window: a = 10, tau = 4",'FontSize',12)

```

```

452 hold on
453 plot(t,g,'m','Linewidth',2) % shows the filter on the S(t) plot
454 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
455
456 % filtered signal as a function of time
457 % signal data ~[3, 6],, otherwise flat
458 subplot(3,1,2)
459 plot(t,Sg,'k','Linewidth',2)
460 title("Filtered Signal v Time",'FontSize',12)
461 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('Sg(t)')
462
463 % FFT(S) % as a function of frequency
464 % way less peaks and dips!
465 subplot(3,1,3)
466 plot(ks,abs(fftshift(Sg_fft))/max(abs(Sg_fft)), 'r','Linewidth',2);
467 title("FFT of Filtered Signal v Frequency",'FontSize',12)
468 axis([-5e3 5e3 0 1.1])
469 set(gca,'FontSize',11)
470 xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
471
472 %% Change window size (filter width) – wider window
473
474 tau = 4;
475 a = 1;
476 g = exp(-a*(t-tau).^2);
477 Sg = g.*S;
478 Sgt = fft(Sg);
479
480 % shows how filter widens on the S(t) plot
481 % figure(4)
482 subplot(3,1,1)
483 plot(t,S,'k','Linewidth',2)
484 title("Signal v Time & Gabor Window: a = 1, tau = 4",'FontSize',12)
485 hold on
486 plot(t,g,'m','Linewidth',2)
487 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
488
489 % less accuracy in the time domain
490 subplot(3,1,2)
491 plot(t,Sg,'k','Linewidth',2)
492 title("Filtered Signal v Time",'FontSize',12)
493 set(gca,'FontSize',12), xlabel('Time (t)'), ylabel('Sg(t)')
494
495 % more detail in the frequency domain
496 subplot(3,1,3)
497 plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)), 'r','Linewidth',2);
498 title("FFT of Filtered Signal v Frequency",'FontSize',12)
499 axis([-5e3 5e3 0 1.1])
500 set(gca,'FontSize',11)
501 xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
502

```



```

503 %% Change window size – narrower window
504
505 tau = 4;
506 a = 100;
507 g = exp(-a*(t-tau).^2);
508 Sg = g.*S;
509 Sgt = fft(Sg);
510
511 %% shows how filter widens on the S(t) plot
512 % figure (5)
513 subplot(3,1,1)
514 plot(t,S,'k','Linewidth',2)
515 title("Signal v Time & Gabor Window: a = 100, tau = 4",'FontSize',12)
516 hold on
517 plot(t,g,'m','Linewidth',2)
518 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
519
520 % less accuracy in the time domain
521 subplot(3,1,2)
522 plot(t,Sg,'k','Linewidth',2)
523 title("Filtered Signal v Time",'FontSize',12)
524 set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('Sg(t)')
525
526 % more detail in the frequency domain
527 subplot(3,1,3)
528 plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
529 title("FFT of Filtered Signal v Frequency",'FontSize',12)
530 axis([-5e3 5e3 0 1.1])
531 set(gca,'FontSize',11)
532 xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
533
534 %% Animation of shifting window
535
536 % figure (6)
537 a = 1;
538 tstep=0:0.1:L;
539
540 for j=1:length(tstep)
541     g=exp(-a*(t-tstep(j)).^2);
542     Sg=g.*S;
543     Sgt=fft(Sg);
544
545     subplot(3,1,1)
546     plot(t,S,'k','Linewidth',2)
547     title("Signal v Time & Gabor Window: a = 1, tau = 1",'FontSize',12)
548     hold on
549     plot(t,g,'m','Linewidth',2)
550     hold off
551     set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('S(t)')
552
553     subplot(3,1,2)

```

```

554     plot(t,Sg,'k','Linewidth',2)
555     title("Signal v Time & Gabor Window: a = 1, tau = 1",'FontSize',12)
556     set(gca,'FontSize',11), xlabel('Time (t)'), ylabel('Sg(t)')
557
558     subplot(3,1,3)
559     plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
560     title("FFT of Gabor Filtered Signal v Frequency",'FontSize',12)
561     axis([-5e3 5e3 0 1.1])
562     set(gca,'FontSize',11)
563     xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
564     drawnow
565     pause(0.1)
566 end
567
568 %% Calculate Gabor transform and plot spectrogram
569
570 k_Hz= (1/(2*L))*[0:(n/2-1) -n/2:-1];
571 ks_Hz=fftshift(k_Hz);
572
573 a = 50;
574 tstep=0:0.3:L;
575 inc = tstep(2)-tstep(1);
576
577 Sgt_spec = zeros(length(tstep),n);
578
579 for j=1:length(tstep)
580     g=exp(-a*(t-tstep(j)).^2);
581     Sg=g.*S; % apply new filter
582     Sgt=fft(Sg); % transform to frequency domain
583     Sgt_spec(j,:) = fftshift(abs(Sgt)); % We don't want to scale it
584 end
585
586 % figure (7)
587 pcolor(tstep,ks_Hz,Sgt_spec. '),
588 shading interp
589 title('Mary Had A Little Lamb: a = 50, time step = 0.5 sec , ', '
        Fontsize',12)
590 set(gca,'FontSize',11)
591 axis([0 L 390 525])
592 xlabel('Time Steps'), ylabel('Frequency(Hz)')
593 colormap(hot)
594
595 %% View one note and overtones
596
597 % max signal in the time domain
598 [max_S, idx] = max(S);
599 max_time = t(idx);
600
601 % find tstep(j) closest to max signal
602 above = find(tstep > max_time);
603 nearest_above = above(1);

```

```

604 below = find(tstep < max_time);
605 nearest_below = below(length(below));
606
607 % get sample with max signal data
608 max_sample = Sgt_spec(nearest_below,:);
609
610 % full frequency plot
611 figure(8)
612 plot(ks_Hz, max_sample);
613 xlabel('Frequency'); ylabel('Amplitude');
614 title('Highest Recorder Signal');
615 axis([0 5e3 0 250])
616
617 % zoom in on overtones
618 figure(9)
619 plot(ks_Hz, max_sample);
620 xlabel('Frequency'); ylabel('Amplitude');
621 title('Recorder Overtones');
622 axis([0 5e3 0 50])

```