AMATH 482 Homework 2
Reproduce the Music Score

Weien Guo

February 10, 2021

Abstract

This report discusses Gabor transform and its application in audio analysis. We will explore the influence of the window size parameter, a, in the process of reproducing the music score of a given audio file. Overtones and frequencies produced by other instruments could be ignored by using filters in the frequency space. Such filters are able to isolate different instruments and provide cleaner and more readable spectrograms for us.

1 Introduction and Overview

Given two audio files, **GNR.m4a** (*Sweet Child O'Mine* by Guns N' Roses, 14-second clip), and **Floyd.m4a** (*Comfortably Numb* by Pink Floyd, 60-second clip), we want to reproduce the music score for the guitar part in both, and the bass part in the latter. The files are converted into MATLAB in the form of vector, representing the amplitude.

For *Sweet Child O'Mine*, only guitar is played in this 14-second clip. Its music score can be retrieved by applying Gabor transform. In contrast, *Comfortably Numb* has three instruments playing at the same time, guitar, drums, and bass. We will first use a Shannon filter to isolate the frequencies produced by the bass, and then use Gabor filtering to get its music score/identify the notes, and perform a similar procedure for analyzing the guitar solo part.

Since all notes have distinct frequencies, plotting the frequencies in spectrograms can help us to directly identify the notes and the time notes are played. We will use this visualization tool extensively in this report.

2 Theoretical Background

From Assignment 1, we have seen the power of Fourier transform in analyzing the frequency of a signal. It formula and the details about the MATLAB built-in function `fftshift()` can be found in the previous report. We have also found that we lose information about the time domain as we perform the Fourier transformation on a signal, and Gabor transform can aid us in retaining the time information.

2.1 The Gabor Transform

The idea of *Gabor transform* is to apply Fourier transform on the filtered signal as sliding the filter window across time. It is also called the short-time Fourier transform (STFT). The formula for the Gabor transform is

$$\tilde{f}_g(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}dt$$

where $g(t)$ is the filter function, $f(t)$ is the signal function, and τ is the center of the filter. By traversing τ in time, we get $\tilde{f}_g(\tau, k)$, that is the frequency information near τ. The outcome of the Gabor transform depends heavily on the choice of the filter function $g(t)$. Usually, a *Gaussian* filter is used,

$$F(k) = e^{-a(t-\tau_i)^2}.$$

The parameter a is the width of the filter window. Larger a would give a smaller window and less time information is captured. A smaller a gives a greater window, and less frequency information is included. We would need to find a good window size for the Gabor transform to work well. And based on the Heisenberg uncertainty principle, we cannot have all the time and frequency information at the same time.

Based on our problem setup, τ is not continuous. The *Discrete Gabor Transform* should be used here. The formula is

$$\tilde{f}_g(m,n) = \int_{-\infty}^{\infty} f(t)g(t - nt_0)e^{2\pi i m\omega_0 t}dt$$

where m and n are integers and ω_0 and t_0 are positive constants.

2.2 Wavelets

The filter function in Gabor transform is called wavelets. There are infinitely many of them. Some assumptions for this function would be real, symmetric, and with unit L_2-norm. We refer to those as the *mother wavelet*,

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$$

where $a > 0$ is the *scaling* parameter and b is the *translation* parameter. Some commonly used wavelets include the Haar wavelet and the Mexican Hat wavelet.

The Shannon filter is another filter we will use in this report. It is a step function, taking in values 0 or 1. This gives Shannon filter the ability to cut off unwanted frequencies.

2.3 Spectrograms

Knowing that Gabor transform uses a sliding window, we would need to combine the result from each window for presenting our analysis. The best way is to use a spectrogram. It is a plot that aligns the all Fourier transforms with the x-axis being the center of the filter, and y-axis being frequencies. With all the information presented in a spectrogram, we can easily decide whether to alter the parameters for better results or not.

MATLAB has its built-in function `spectrogram()` to plot spectrograms. But in this report, we will make spectrograms without it. MATLAB uses $e^{i2\pi ft}$, which scales out the 2π in periods of oscillation, and we have been using e^{ikt}. This difference gives the resulting frequency different units. MATLAB has it in Hertz, and we have it as angular frequency. Becasue we are performing time-analysis on music, and the frequency of notes are usually measured in Hertz, we would need to re-scale the k vector by $\frac{1}{L}$ instead of $\frac{2\pi}{L}$ to get Hertz.

3 Algorithm Implementation and Development

3.1 Setup

1. Load the music file (.m4a) into MATLAB.

2. Define time domain, which is the length is the record time in seconds.
3. Define the frequency domain, scale it using $\frac{1}{L}$, and apply `fftshift()` to swap the domain.

3.2 Gabor Transform and Spectrogram

1. Define the window width parameter a, and time-step vector τ. We can do a few tests to get good a and dt. Figure 1 shows the results with different a and dt for the first 4 second from *Sweet Child O'Mine*. The colored parts in the third plot are more like dots compared to the other three. So for this song, we use $a = 300$ and $dt = 0.1$. Similar tests were performed on *Comfortably Numb*. When analyzing the bass part, we set $a = 100$ and $dt = 0.2$, and $a = 1000$ and $dt = 0.1$ for the guitar part.

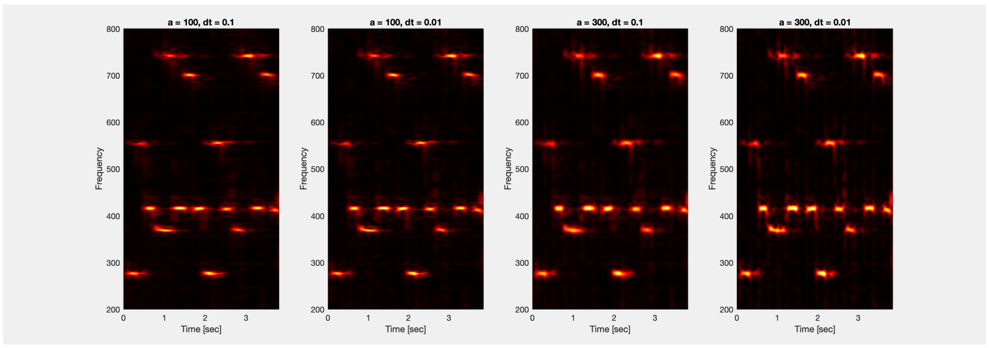

Figure 1: Window size and time-step test for GNR.m4a

2. "Slide" the filtering window over time using a for loop. In the loop, we first define the Gaussian filter, and then multiply this filter with the music signal. Fast Fourier transform is applied on this filtered signal. We take the absolute value of it and swap the halves. The result is saved in a matrix which has `length(`τ`)` as number of rows.
3. A spectrogram in color is generated based on the matrix for visualization. X-axis is time, and y-axis is frequency. We can adjust `Ylim` to focus on different range of frequency.
4. As we are in the process of identifying note, we can plot horizontal lines, whose values equal certain notes, on spectrograms to help us know which notes are in the clip.

Both clips are also broken down into smaller portions for faster run time.

3.3 Using the Shannon Filter

Because different instruments appear in Floyd.m4a, and instruments have certain range, we can apply the Shannon filter to isolate frequencies produced by certain instrument based on its range. Bass only produces frequency below 200 Hertz in this clip, so we can remove higher frequencies and only focus on lower ones. When reproducing the guitar notes, we create another filter that filters out the low frequencies. This process is performed on the original signal, before the Gabor transform.

1. Perform a fast Fourier transform on the original music signal.
2. Create a Shannon filter (step-function) that has value 1 on indices for lower frequency and value 0 for higher frequency.

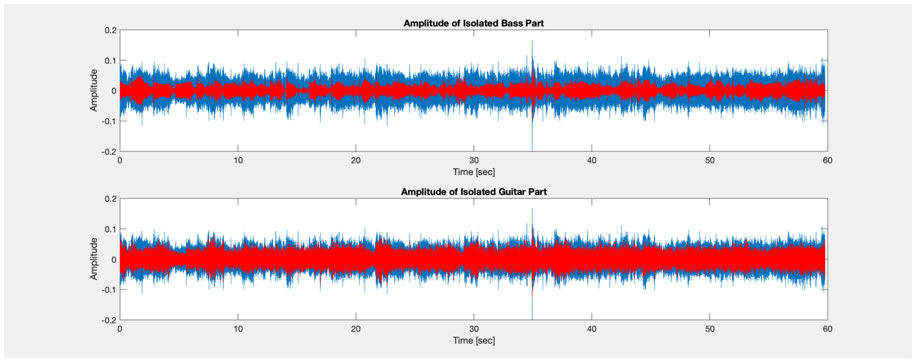

Figure 2: Isolated instrument parts for Comfortably Numb

3. Multiply the Shannon filter with the FFT result, and get the filtered signal back by inverse FFT. Figure 2 shows signal after using the Shannon filter. The upper plot is the isolated bass signal on top of the original one for *Comfortably Numb*. When playing the signal, we cannot hear the guitar and drum any more, which means the isolation is successful. The bottom plot shows the signal without the bass (low) frequency. The remaining record still has guitar and drums because their music range overlap.

4. We can then do the Gabor transform, as described in the previous section, on the filtered signal for reproducing the music scale.

4 Computational Results

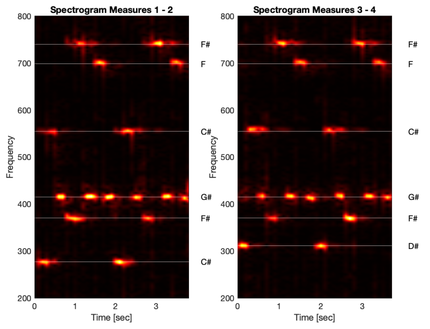

Figure 3: Spectrograms of the first 4 measures in Sweet Child O'Mine

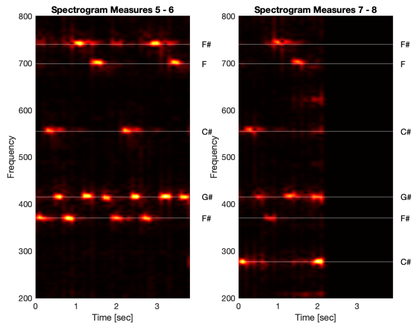

Figure 4: Spectrograms of Sweet Child O'Mine starting from measure 5

Figure 3 and 4 are the spectrograms from results of Gabor transform on GNR.m4a, *Sweet Child O'Mine*. Each plot consists with context from two measures. The guitar here plays notes with same length and follows a pattern, which is an eight-note measure repeats twice. Each repeated part is different from the others only by the first note. For example, in the first two measures, there are sixteen notes, and the first eight notes are the same with the second eight notes. Measure three and measure four are also identical. This repeated part only has the first note different from the first repeated part. The pattern continues. The clip stops around the start of measure eight, but we can see that measure seven is the same as measure one.

The notes are:

$$C^\# - C^\# - G^\# - F^\# - F^\# - G^\# - F - G^\# \quad | \quad C^\# - C^\# - G^\# - F^\# - F^\# - G^\# - F - G^\#$$
$$D^\# - C^\# - G^\# - F^\# - F^\# - G^\# - F - G^\# \quad | \quad D^\# - C^\# - G^\# - F^\# - F^\# - G^\# - F - G^\#$$
$$F^\# - C^\# - G^\# - F^\# - F^\# - G^\# - F - G^\# \quad | \quad F^\# - C^\# - G^\# - F^\# - F^\# - G^\# - F - G^\#$$
$$C^\# - C^\# - G^\# - F^\# - F^\# - G^\# - F - G^\# \quad | \quad C^\# -$$

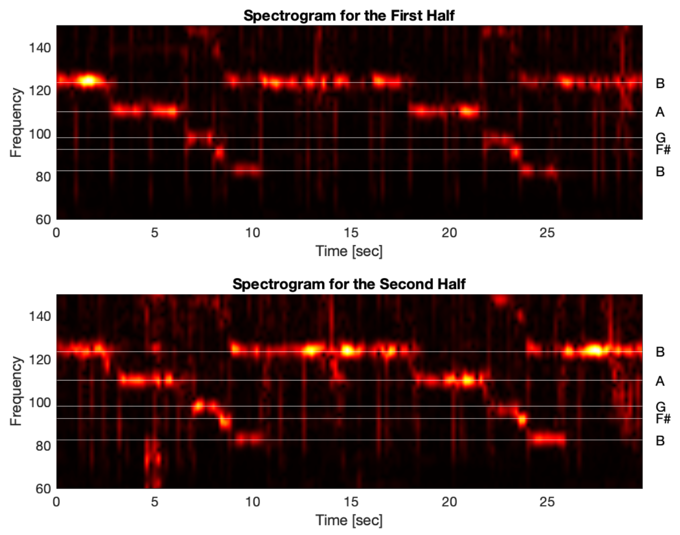

Figure 5: Spectrogram for the bass part in Comfortably Numb

Figure 5 is the result we get for the bass in *Comfortably Numb*. Because this clip has 60 seconds, we divided it into two 30-second portions for faster computation and less memory usage. The Shannon filter is also used before the application of Gabor transform, which also saves some run time when plotting. Window size and time step has been tested for the best result. From the plots, we can see that the first half and the second half are very similar. But we do realize that there are some notes not captured by the spectrogram, and the length of each note cannot be determined with confidence. From Figure 5, we can also see that the order of notes $B - A - G - F^\# - B$ repeats for four times.

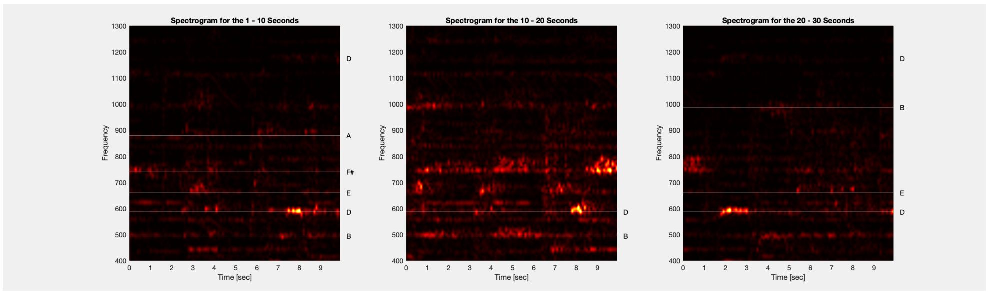

Figure 6: Spectrograms for 1 - 30 seconds guitar solo in Comfortably Numb

The spectrograms in Figure 6 and 7 are for reproducing the notes in *Comfortably Numb* guitar solo, but we can see that notes are very hard to identify because the presence of drum frequencies.

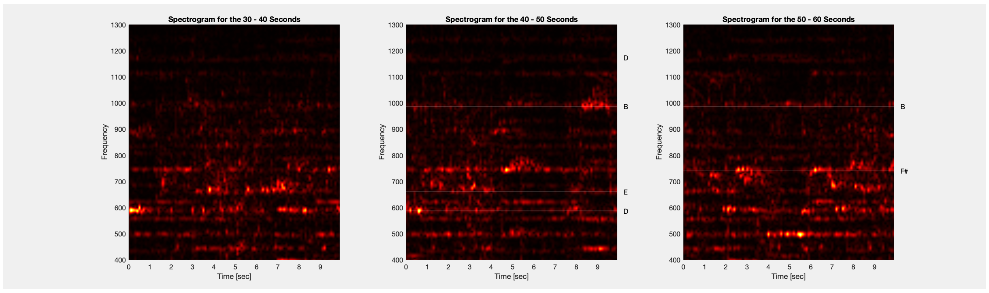

Figure 7: Spectrograms for 30 - 60 seconds guitar solo in Comfortably Numb

In the first half, it is still possible to see some guitar notes, which do not appear for a long time, have a clear emphasis and do not fall in the long red lines, but it becomes very difficult in the second half. The notes are (in order of appearance in the first plot): $F^{\#}$ (739.99 Hertz), D (587.33 Hertz), E (659.26 Hertz), A (880 Hertz), B (493.88 Hertz) and D (1174.7 Hertz).

5 Summary and Conclusions

Started with original music clips, we are able to reproduce the music scale by using Gabor transform. When dealing with music that has multiple instruments, we can apply a Shannon filter in the frequency space to isolate instruments. Spectrograms are especially helpful in visualizing the results from Gabor transform, and we can see when and what notes are played based on those spectrograms. The guitar part in *Sweet Child O'Mine* by Guns N' Roses has repeated measures, and the bass in *Comfortably Numb* by Pink Floyd also repeats, but with minor changes. The guitar solo in *Comfortably Numb* is the hardest to identify. We are only able to see some notes, and could not find the melody. But we look forward to studying more powerful methods for solving this problem.

Appendix A MATLAB Functions

- `[y, fs] = audioread(X)` reads the audio file `X`, and returns the data in Y and the sample rate FS in Hertz.
- `audioplayer(Y, Fs)` creates an audioplayer object for signal Y using sample rate Fs. `playblocking(OBJ)` plays the audio audioplayer sample.
- `y = fftshift(X)` returns shifted `x` with zero-frequency component at the center of spectrum.
- `y = fft(X)` returns the discrete fast Fourier transform on vector `X`.
- `y = ifft(X)` returns the inverse discrete fast Fourier transform on vector `X`.
- `pcolor(X,Y,C)` makes a pseudocolor plot of matrix C with axes X and Y. The values of the elements in C determines the color in each cell of the plot.
- `ones(M,N)` creates an M-by-N matrix of ones. Similarly, `zeros(M,N)` creates an M-by-N matrix of zeros.

Appendix B MATLAB Code

```
% Clean workspace
clear all; close all; clc

%%
[y, Fs] = audioread('GNR.m4a');
tr_gnr = length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Sweet Child O Mine');
p8 = audioplayer(y,Fs);
playblocking(p8);

%% explore window size and time step
close all; clear all; clc;

[y,Fs] = audioread('GNR.m4a');

y = y(1:185000);
S = y';
tr_gnr = length(y)/Fs;

L = tr_gnr;
n = length(y);
t2 = linspace(0, L, n+1);
t = t2(1:n);
k = (1 / L) * [0:n/2-1 -n/2:-1];
ks = fftshift(k);

a = 100;
tau = 0:0.1:L;

% a = 100;
% tau = 0:0.01:L;

% a = 300;
% tau = 0:0.1:L;

% a = 300;
% tau = 0:0.01:L;

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

subplot(1,4,1)
% subplot(1,4,2)
% subplot(1,4,3)
% subplot(1,4,4)

pcolor(tau, ks, Sgt_spec.');
shading interp;
colormap(hot);
set(gca,'Ylim', [200 800]);
xlabel('Time␣[sec]');
ylabel('Frequency');
title('a␣=␣100,␣dt␣=␣0.1');
% title('a = 100, dt = 0.01');
% title('a = 300, dt = 0.1');
% title('a = 300, dt = 0.01');

%% part 1 : guitar in GNR

close all; clear all; clc;

% repeat 1 and 2
% repeat 3 and 4
[y,Fs] = audioread('GNR.m4a');

y = y(1:185000);
% y = y(185001 : 365000);
% y = y(365001 : 550000);
% y = y(550001 : end);

S = y';
tr_gnr = length(y)/Fs;

L = tr_gnr;
n = length(y);
t2 = linspace(0, L, n+1);
t = t2(1:n);
k = (1 / L) * [0:n/2-1 -n/2:-1];
ks = fftshift(k);

% Spectrogram
a = 300;
tau = 0:0.1:L;
Sgt_spec = [];
for j=1:length(tau)
    g = exp(-a * (t-tau(j)).^2);
    Sg = g .* S;
    Sgt = fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end

subplot(1,2,1)
% subplot(1,2,2)
```

```
pcolor(tau, ks, Sgt_spec.');
shading interp;
colormap(hot);
set(gca,'Ylim', [200 800]);
xlabel('Time [sec]');
ylabel('Frequency');
title('Spectrogram Measures 1 - 2');
% title('Spectrogram Measures 3 - 4');
% title('Spectrogram Measures 5 - 6');
% title('Spectrogram Measures 7 - 8');

yline(554.37, 'w');
text(4.1, 554.37, sprintf('C#'),'FontSize',10);
yline(415.3, 'w');
text(4.1, 415.3, sprintf('G#'),'FontSize',10);
yline(370, 'w');
text(4.1, 370, sprintf('F#'),'FontSize',10);
yline(740, 'w');
text(4.1, 740, sprintf('F#'),'FontSize',10);
yline(698.5, 'w');
text(4.1, 698.5, sprintf('F'),'FontSize',10);
yline(277.18, 'w');
text(4.1, 277.18, sprintf('C#'),'FontSize',10);

% yline(311.13, 'w');
% text(4.1, 311.13, sprintf('D#'),'FontSize',10);

saveas(gcf, 'Sweet1.png')
% saveas(gcf, 'Sweet2.png')

%% part 2 : bass in Floyd
close all; clear all; clc;

%% Shannon filter

[y, Fs] = audioread('Floyd.m4a');

S = [y', 0, 0, 0];
tr_floyd = length(S)/Fs;

L = tr_floyd;
n = length(S);
t2 = linspace(0, L, n+1);
t = t2(1:n);
k = (1 / L) * [0:n/2-1 -n/2:-1];
ks = fftshift(k);

Sf = fft(S);
[Ny, Nx] = size(S);
wy = 10000;
filter = ones(size(S));
```

```
filter(wy+1 : Nx-wy+1) = zeros(1, Nx-2*wy+1);

Sfiltered = Sf .* filter;
Sg_f = ifft(Sfiltered);

tr_gnr = length(y)/Fs;
plot((1:length(y))/Fs, y);
xlabel('Time [sec]');
ylabel('Amplitude');
hold on
plot((1:length(S))/Fs,Sg_f, 'r');
title('Amplitude of Isolated Bass Part');

saveas(gcf, 'Floyd Comparison.png')

% play the isolated bass
% the melody played by the guitar disappears

% p8 = audioplayer(Sg_f,Fs);
% playblocking(p8);

%% repeat 1 and 2
leng = length(Sg_f);

y = Sg_f(1:leng/2);
% y = Sg_f(leng/2+1:end);

S = y;
tr_floyd = length(y)/Fs;

L = tr_floyd;
n = length(y);
t2 = linspace(0, L, n+1);
t = t2(1:n);
k = (1 / L) * [0:n/2-1 -n/2:-1];
ks = fftshift(k);

% Spectrogram
a = 100;
tau = 0:0.2:L;
Sgt_spec = [];
for j=1:length(tau)
    g = exp(-a * (t-tau(j)).^2);
    Sg = g .* S;
    Sgt = fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end

subplot(2, 1, 1)
% subplot(2, 1, 2)
```

```
pcolor(tau, ks, Sgt_spec.');
shading interp;
colormap(hot);
set(gca,'Ylim', [60 150]);
xlabel('Time [sec]');
ylabel('Frequency');
title('Spectrogram for the First Half');
% title('Spectrogram for the Second Half');

yline(123.5,'w');
text(30.5,123.5,sprintf('B'));
yline(110,'w');
text(30.5,110,sprintf('A'));
yline(98,'w');
text(30.5,98,sprintf('G'));
yline(92.5,'w');
text(30.5,92.5,sprintf('F#'));
yline(82.4,'w');
text(30.5,82.4,sprintf('B'));

saveas(gcf, 'floyd.png')

%% part 3 : guitar in Floyd
close all; clear all; clc;

%% Shannon filter

[y, Fs] = audioread('Floyd.m4a');

S = y(1:end-1)';
tr_floyd = length(S)/Fs;

L = tr_floyd;
n = length(S);
t2 = linspace(0, L, n+1);
t = t2(1:n);
k = (1 / L) * [0:n/2-1 -n/2:-1];
ks = fftshift(k);

Sf = fft(S);
[Ny,Nx] = size(S);
wy = 10000;
filter = zeros(size(S));
filter(wy+1 : Nx-wy+1) = ones(1, Nx-2*wy+1);
wy = 100000;
filter(wy+1 : Nx-wy+1) = zeros(1, Nx-2*wy+1);

Sfiltered = Sf .* filter;
Sg_f = ifft(Sfiltered);

tr_gnr = length(y)/Fs;
```

```
plot((1:length(y))/Fs,y);
xlabel('Time [sec]');
ylabel('Amplitude');
hold on
plot((1:length(S))/Fs,((Sg_f)), 'r');
title('Amplitude of Isolated Guitar Part');

saveas(gcf, 'Floyd Guitar Comparison.png')

%%
% the sound of bass disappears
p8 = audioplayer(Sg_f,Fs);
playblocking(p8);

%% repeat 1, 2, and 3
%% repeat 4, 5, and 6
leng = length(Sg_f);

y = Sg_f(1:leng/6);
% y = Sg_f(leng/6+1:leng/3);
% y = Sg_f(leng/3+1:leng/2);
% y = Sg_f(leng/2+1:4*leng/6);
% y = Sg_f(4*leng/6+1:5*leng/6);
% y = Sg_f(5*leng/6+1:end);

S = y;
tr_floyd = length(y)/Fs;

L = tr_floyd;
n = length(y);
t2 = linspace(0, L, n+1);
t = t2(1:n);
k = (1 / L) * [0:n/2-1 -n/2:-1];
ks = fftshift(k);

% Spectrogram
a = 1000;
tau = 0:0.1:L;
Sgt_spec = [];
for j=1:length(tau)
    g = exp(-a * (t-tau(j)).^2);
    Sg = g .* S;
    Sgt = fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end

subplot(1, 3, 1)
% subplot(1, 3, 2)
% subplot(1, 3, 3)

pcolor(tau, ks, Sgt_spec.');
```

```
shading interp;
colormap(hot);
set(gca,'Ylim', [400 1300]);
xlabel('Time␣[sec]');
ylabel('Frequency');
title('Spectrogram␣for␣the␣1␣-␣10␣Seconds');
% title('Spectrogram for the 10 - 20 Seconds');
% title('Spectrogram for the 20 - 30 Seconds');
% title('Spectrogram for the 30 - 40 Seconds');
% title('Spectrogram for the 40 - 50 Seconds');
% title('Spectrogram for the 50 - 60 Seconds');

yline(740,'w');
text(10.2,740,sprintf('F#'));
yline(660,'w');
text(10.2,660,sprintf('E'));
yline(587,'w');
text(10.2,587,sprintf('D'));
yline(494,'w');
text(10.2,494,sprintf('B'));
yline(880,'w');
text(10.2,880,sprintf('A'));
% yline(1174.7,'w');
text(10.2,1174.7,sprintf('D'));

saveas(gcf, 'guitar1.png')
% saveas(gcf, 'guitar2.png')
```