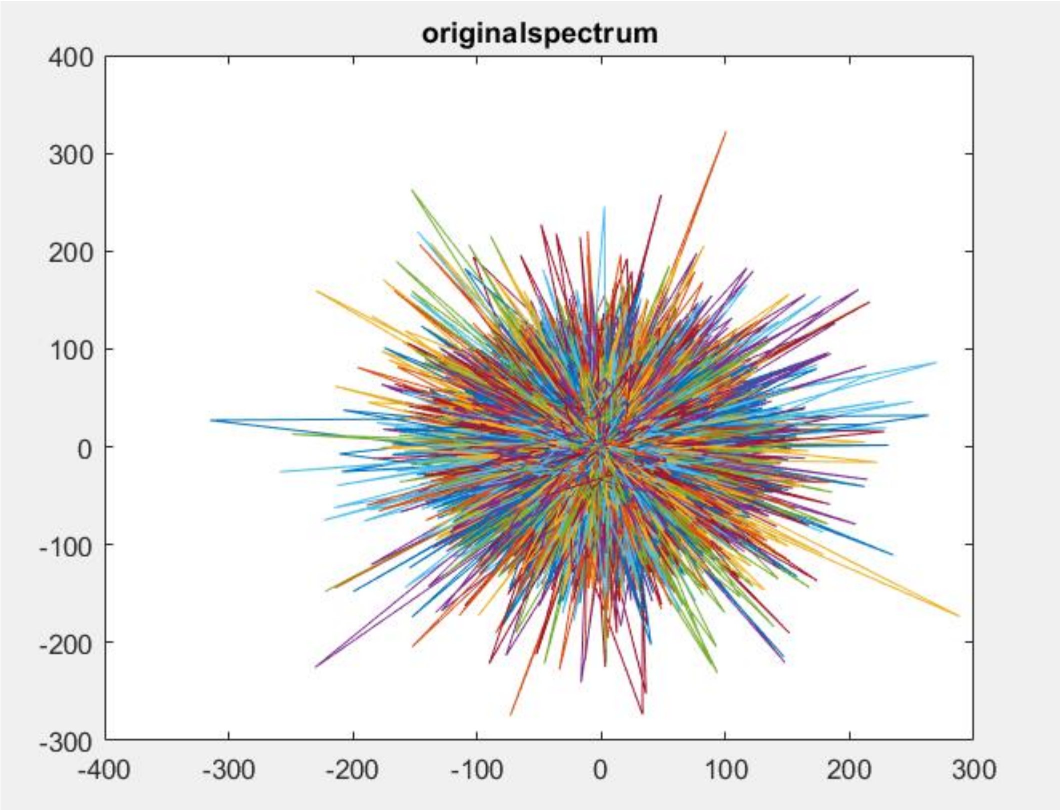
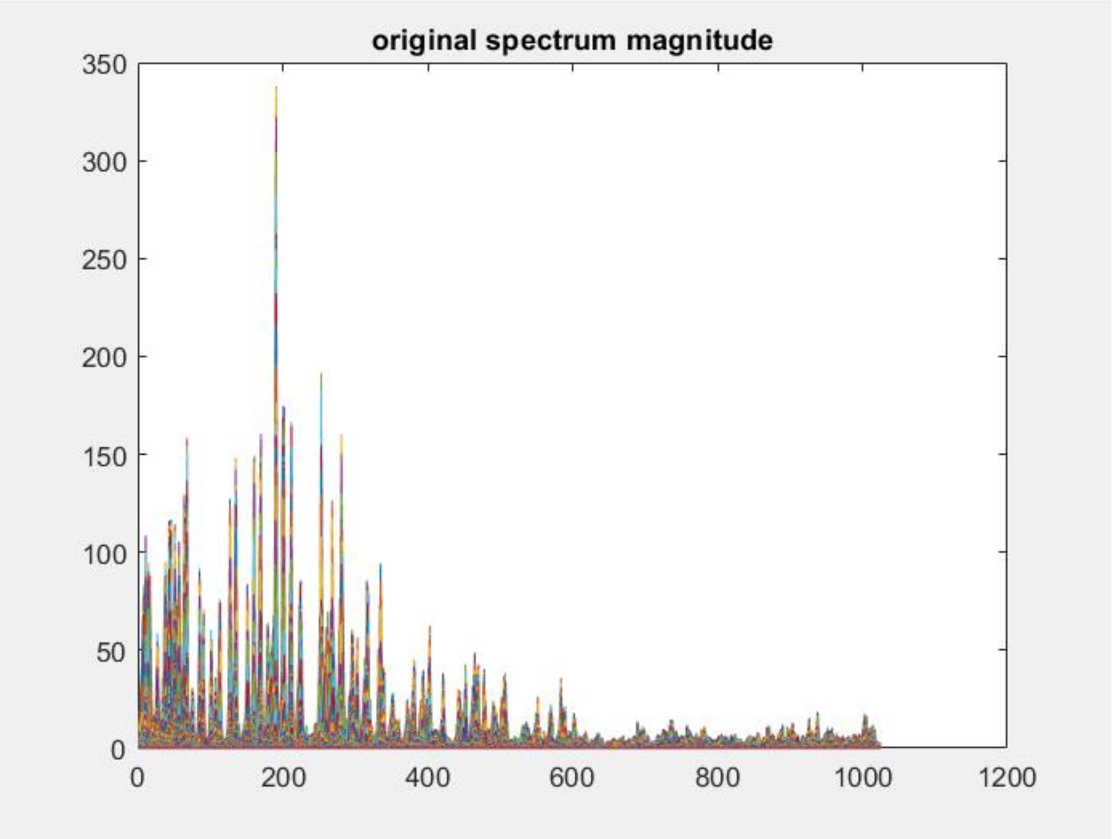
HALİT ERDOĞAN 040160260

PROJECTION (POLYUSHKA)

```
%first, the audio is read.
[s,fs] = audioread('polyushka.wav');
% Audio is resampled at 16000/fs times the original sample
rate which is obviously equal to 16000
s = resample(s,16000,fs);

%by using stft function, the spectrum of B is extracted
and plotted
% 2048 is fft size, 256 is hopsize between adjacent
frames, 0 is padding
% and hann(2048) is window
spectrum = stft(s', 2048, 256, 0, hann(2048));
figure; plot(spectrum); title ('originalspectrum');
```

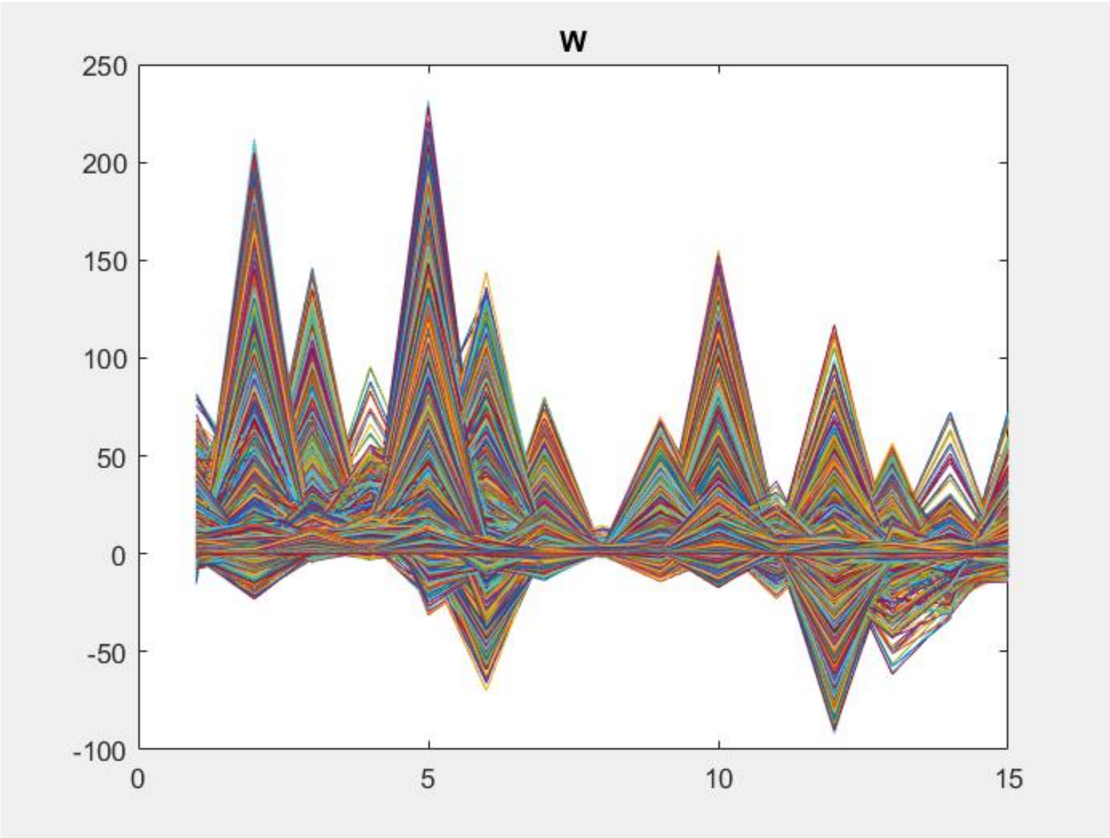



```
%the magnitude and the phase of audio is calculated and
%magnitude is plotted
music = abs(spectrum);
figure; plot(music); title ('original spectrum
magnitude');
sphase = spectrum ./ (abs(spectrum) + eps); %eps is added
in case there is 0 in spectrum
```

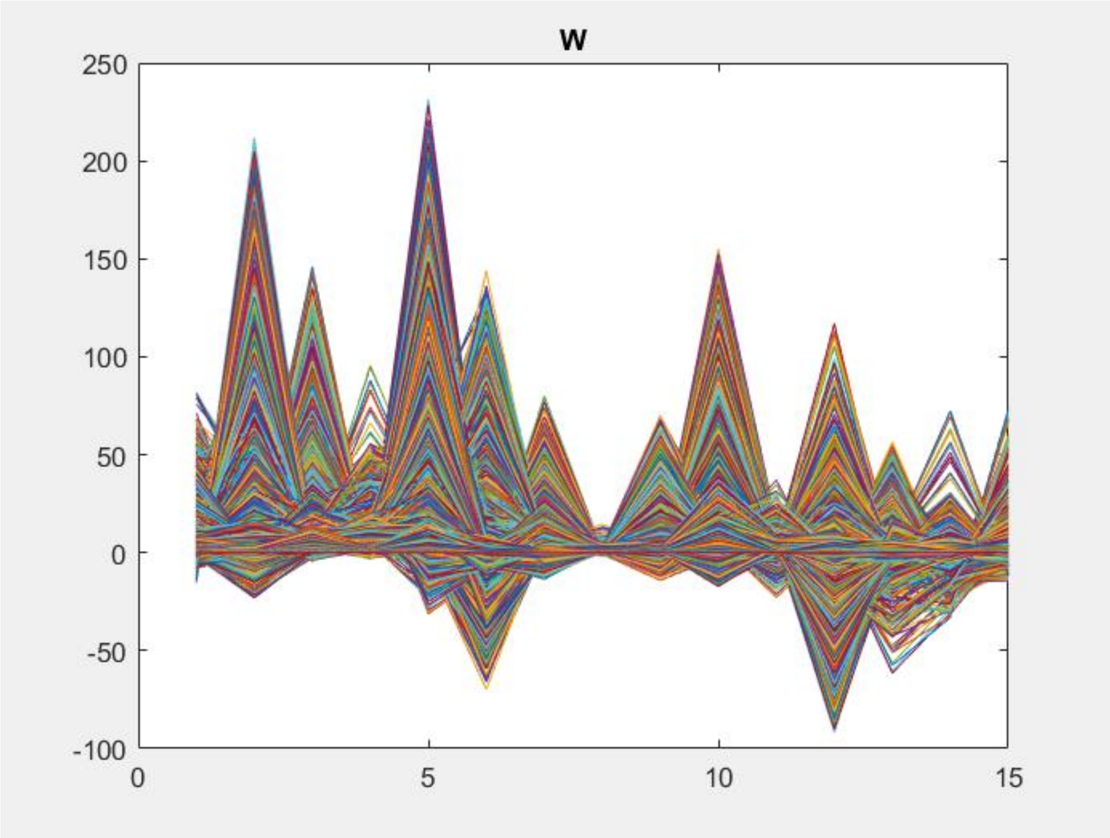



```
%read all the .wav files in the following directory
notesfolder = 'notes15/';
listname = dir([notesfolder '*.wav']);


%define empty array for notes
notes = [];

%create a loop that will save the spectrums of each note /
.wav file to the
%notes variable as different columns
for k = 1:length(listname)
    [n,fn] = audioread([notesfolder listname(k).name]);
    n = n(:,1);
    n = resample(n,16000,fn);
    spectrum_n = stft(n', 2048, 256, 0, hann(2048));
    %find the central frame
    middle = ceil(size(spectrum_n,2)/2);
    note = abs(spectrum_n(:,middle));
    %clean up everything more than 40 db below the peak
    note(find(note < max(note(:))/100)) = 0;
    %normalise the note to unit length
    note = note / norm(note);
    % assign the calculated note to the empty array
    notes = [notes, note];
```

```
end

%since the audio is composed of these notes, (notes * w =
music) where w is
%the ith row of w is the transcription of the ith note. We
calculate and plot w.

w = pinv(notes) * music; %we use pinv() instead of inv()
since notes is not square matrix
figure; plot(w); title ('W');
```




```
%all negative values in w is converted to 0 since there
cannot be
%negative magnitude for a frequency
[i1,i2] = size(w);
for i=1:i1*i2
   if w(i)<=0
       w(i)=0;
   end
end

%the audio spectrum is reconstructed and plotted using the
original formula (notes * w)
reconstructed = notes * w;
figure; plot(reconstructed); title ('reconstructed
spectrum');
```

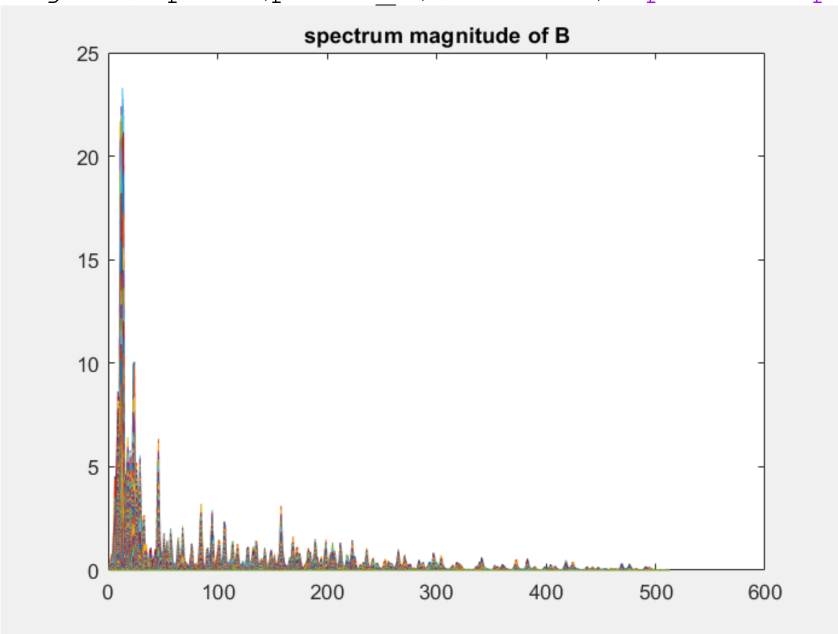


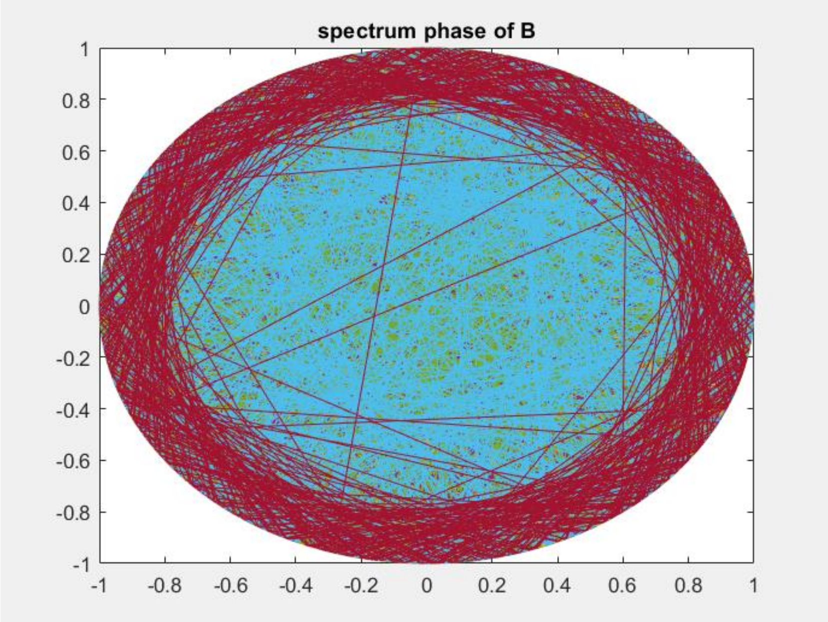

```
%the reconstructed spectrum is then converted to
reconstructed signal with
%stft function, with the same phase, fft size, hopsize,
padding and window
rsignal =
stft(reconstructed.*sphase,2048,256,0,hann(2048));

%the player variable of the audio is defined using
audioplayer function
player = audioplayer(rsignal, fs);

%the audio is played
play(player);
```

LINEAR TRANSFORMATION (PIANO TO GUITAR)

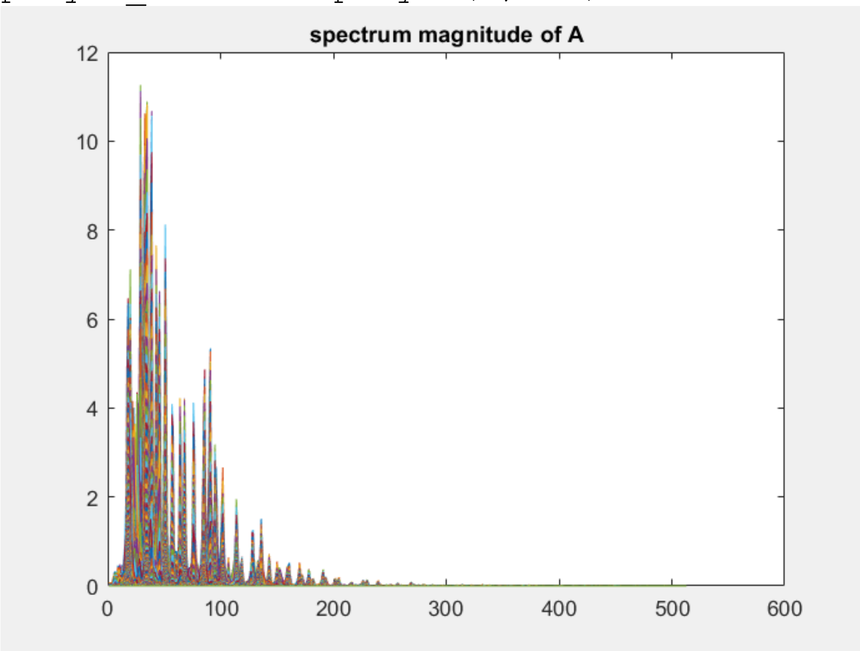
```
%first, the guitar version of silent night is read. (audio
B)
[B,fsB] = audioread('silentnight_guitar.aif'); % B is the
samples and fsB is the sample rate
B = resample(B,16000,fsB); % B is resampled at 16000/fsB
times the original sample rate which is obviously equal to
16000
B = B(:,1); % B is made equal to its first column
```

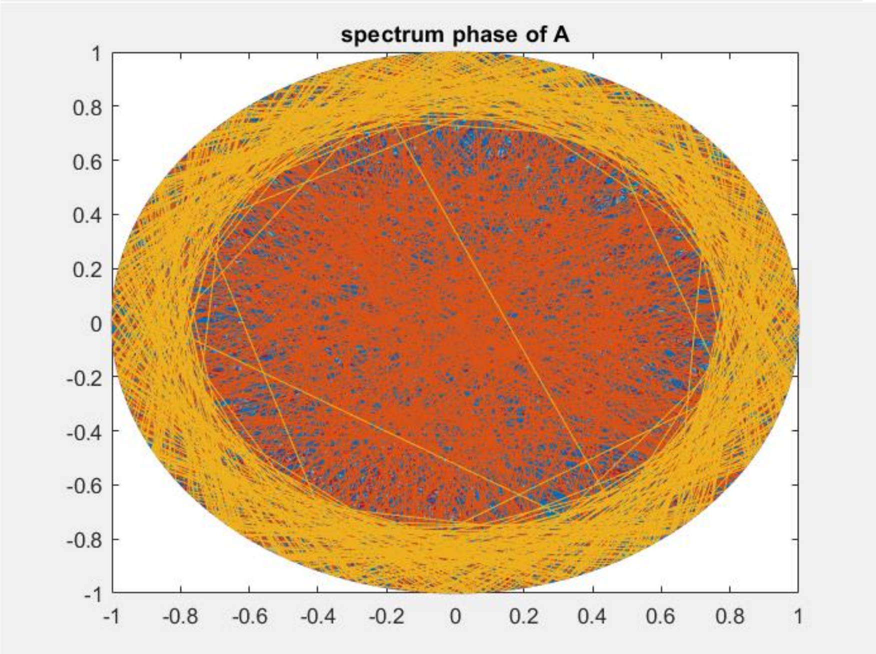
```
%by using stft function, the spectrum of B is extracted
% 1024 is fft size, 256 is hopsize between adjacent
frames, 0 is padding
% and hann(1024 is window
spectrum_B = stft(B', 1024, 256, 0, hann(1024));

%the magnitude and the phase of B is calculated and
magnitude is plotted
music_B = abs(spectrum_B);
figure; plot(music_B); title('spectrum magnitude of B');
phase_B = spectrum_B ./ (abs(spectrum_B) + eps); %eps is
added in case there is 0 in spectrum_b
figure; plot(phase_B); title('spectrum phase of B');
```



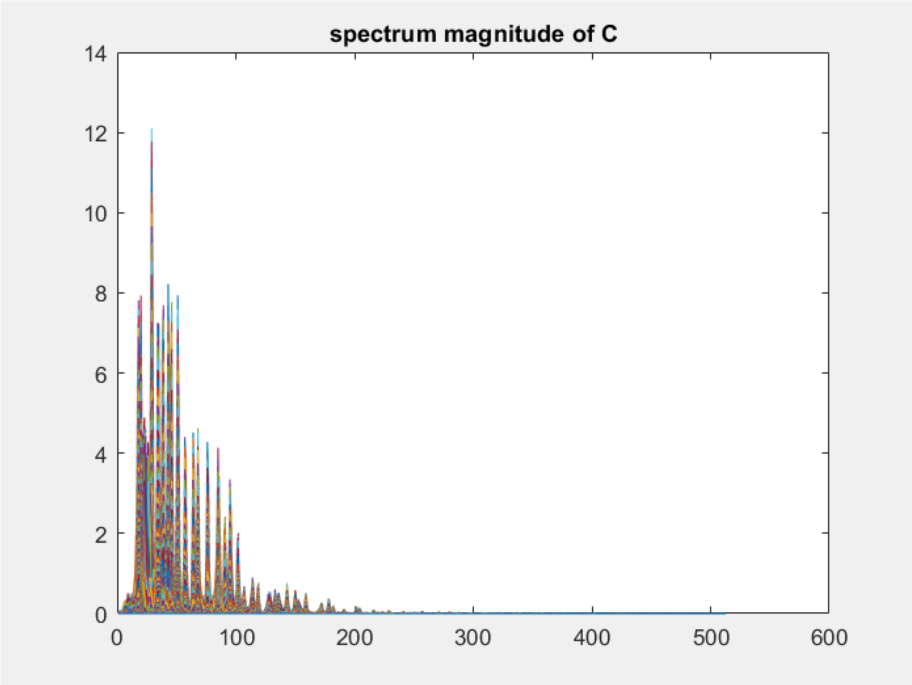


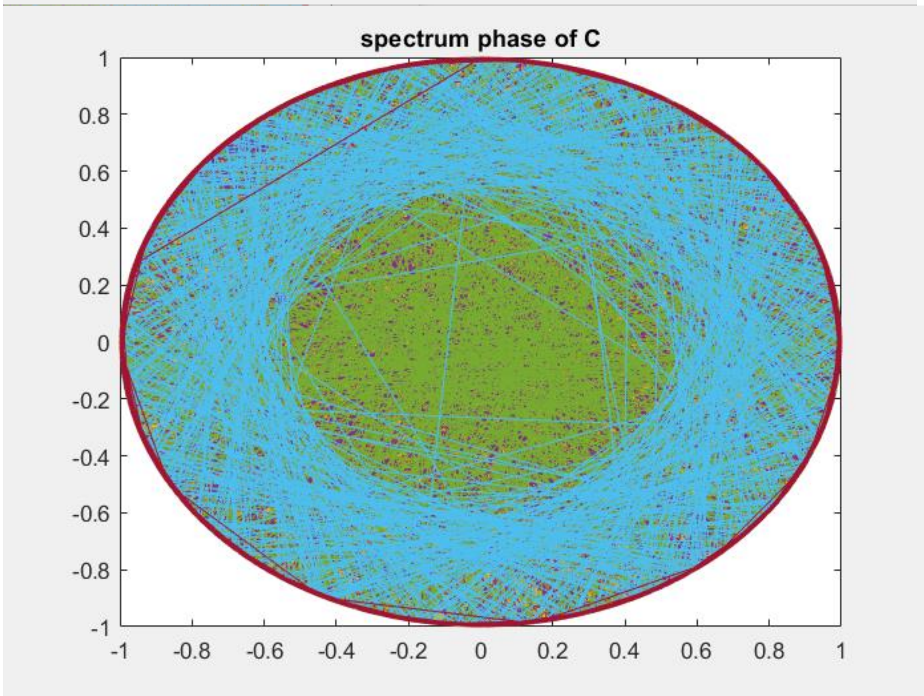
```
%Player of B is defined for the user to test/play
player_B = audioplayer(B,fsB);

%the same process is applied to audios A
[A,fsA] = audioread('silentnight_piano.aif');
A = resample(A,16000,fsA);
A = A(:,1);
spectrum_A = stft(A', 1024, 256, 0, hann(1024));
music_A = abs(spectrum_A);
figure; plot(music_A); title('spectrum magnitude of A');
phase_A = spectrum_A ./ (abs(spectrum_A) + eps);
figure; plot(phase_A); title('spectrum phase of A');
player_A = audioplayer(A,fsA);
```



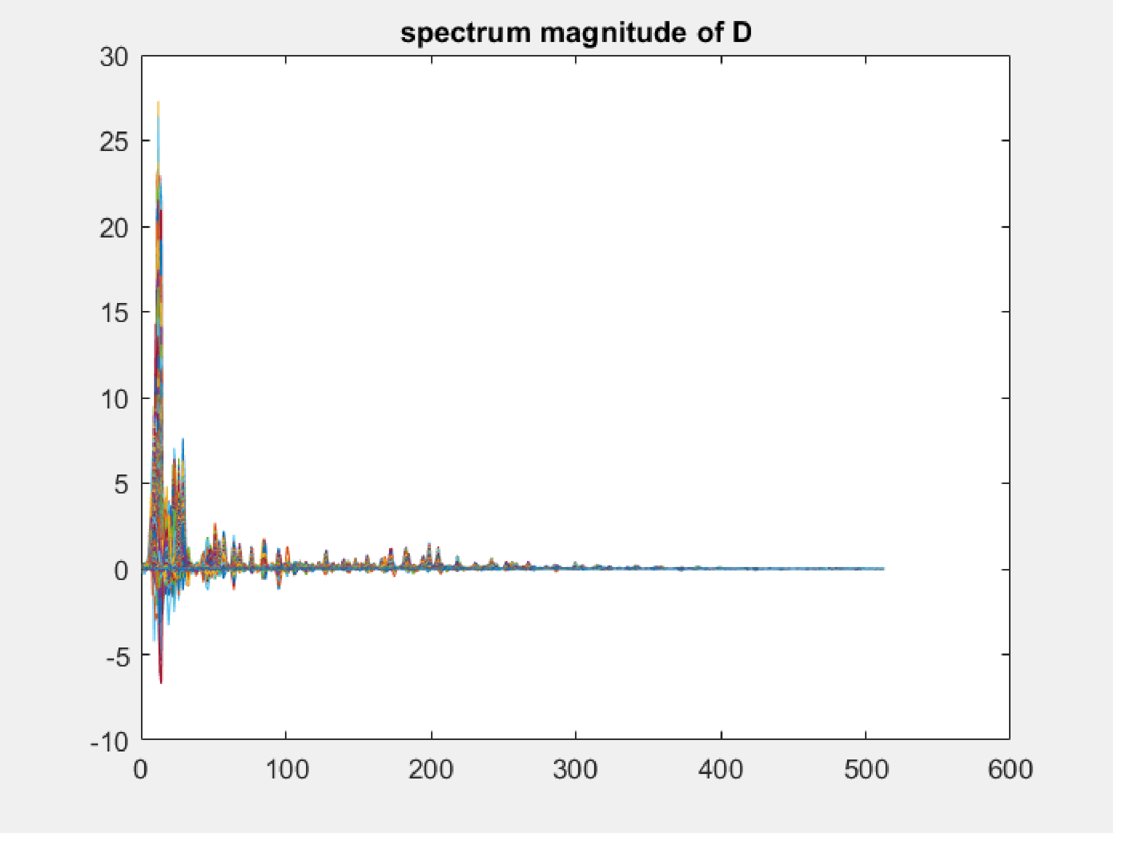


```
%the same process is applied to audios C
[C,fsC] = audioread('littlestar_piano.aif');
C = resample(C,16000,fsC);
C = C(:,1);
spectrum_C = stft(C', 1024, 256, 0, hann(1024));
music_C = abs(spectrum_C);
figure; plot(music_C); title('spectrum magnitude of C');
phase_C = spectrum_C ./ (abs(spectrum_C) + eps);
figure; plot(phase_C); title('spectrum phase of C');
player_C = audioplayer(C,fsC);
```





```
% The result of music_B * pinv(music_A) corresponds to a
"piano to a
% guitar" conversion matrix
% Therefore, the result of music_B * pinv(music_A) *
music_C corresponds to piano
% version of music C
% We use pinv() instead of inv() since music_A is not
square matrix
music_D = music_B * pinv(music_A) * music_C;

%music_D (spectrum magnitude of audio D) is plotted
figure; plot(music_D); title('spectrum magnitude of D');
```




```
%all negative values in music_D is converted to 0 since
there cannot be
%negative magnitude for a frequency
[i1,i2] = size(music_D);
for i=1:i1*i2
    if music_D(i)<=0
        music(i)=0;
    end
end

%audio_D is constructed with stft function
%the magnitudes in music_D is multiplied by phase_C since
our goal is to
```

```
%create a "guitar of version of C". Other parameters are
the same
audio_D = stft(music_D .* phase_C, 1024,256,0,hann(1024));

%we define player variable for audio_D using audioplayer
function
player_audio_D = audioplayer(audio_D, fsC);

%audio_D is played
play(player_audio_D);
```