# Halleluia: The Gabor Transform

Kristen Drummey

Department of Physiology and Biophysics, University of Washington

Github: github.com/kldrummey/AMATH582

**Abstract**

The Gabor transform (GT) is a vital tool for time-series analysis, as it provides a way to maintain temporal resolution while transforming signals into the frequency domain. One obvious use of the GT is in analyzing pieces of music - compositions made up of instruments producing different frequencies, constructed in a particular way over time. In this study, the GT was used to analyze a segment of Handel's *Messiah* and to reconstruct the score of the classic *Mary had a Little Lamb*.

## 1 Introduction and Overview

Although the Fourier transform (FT) is extremely useful in determining frequency signatures present in a dataset, it is unable to provide any information about when those frequencies occurred in time. For moving signals, or signals such as musical compositions, in which frequencies are changing constantly as the signal progresses, a lot of meaningful data is lost by simply taking the FT of the dataset.

The Gabor transform (GT), also known as the short-time Fourier transform, extracts frequency information from a signal while also maintaining temporal resolution. The GT does this by taking the FT of the signal at multiple windows along the signal's length. This allows for analysis of a signal's frequency content while also allowing for examination of how that frequency content changes over time.

In this study, the GT was used to analyze two pieces of music - Handel's *Messiah* and the classic children's song *Mary Had A Little Lamb*. Different variations of time step, filter width, and filter type were applied to *Messiah* to further probe how these variations in the GT change the final spectrogram output of a signal. Using this analysis method, the score for *Mary Had A Little Lamb* was determined, and the effect of overtones in a piano rendition of the song was compared to the less-harmonic rendition of the song that was played using a recorder.

## 2 Theoretical Background

The FT is an excellent tool for looking at the frequency components of a signal. This allows for determination of the signature frequency of a signal of interest and provides a method for filtering out unwanted noise. However, transforming a signal entirely into the frequency domain erases any sense of the signal's temporal behavior. For signals that are moving or changing over time, this temporal resolution is essential for deriving meaning from the data.

The GT solves this problem inherent in the FT by introducing windows. This is accomplished by the Gabor kernel:

$$g_{t,\omega}(\tau) = e^{i\omega\tau} g(\tau - t) \tag{1}$$

This kernel, when introduced into the broader equation shown in Eq. 2 below, takes windows of the data and applies the FT sequentially. This windowing maintains temporal resolution while still transforming the data into the frequency domain for further filtering and analysis.

$$G[f](t,\omega) = \tilde{f}_g(t\omega) = \int_{\infty}^{\infty} f(\tau) g(\tau - t) e^{-i\omega\tau} d\tau \tag{2}$$

Although the GT does maintain both temporal and frequency resolution, the extent to which it does so is limited by the size of the window. Large windows maintain a larger amount of the frequency components, but since the size of the window is larger some temporal information is lost. In contrast, smaller windows maintain more precise temporal information, at the expense of losing frequency information.

# 3 Algorithm Development and Implementation

Handel's Messiah and Mary Had A Little Lamb were both examined using the GT¡, with varying window sizes and time steps. Initially, data was loaded in and the parameters of the recording were determined.

**Algorithm 1:** Establish parameters

Load data using `load` or `audioread`
Determine sampling rate, Fs
Determine length of the sample using `1:length(sample)/Fs`
Set Fourier modes L to be `length(sample)/Fs`
Establish frequency components as `ks=[0:(Fs/2)/(n/2-1):Fs/2`

The Nyquist frequency, or half of the sampling rate, was used to determine the frequency components. After establishing the parameters, Gabor filters were applied to the data using a designated time step.

**Algorithm 2:** Apply Gabor transform to data

Determine timestep and set sliding window as `tslide=0:timestep:L`
Establish empty matrix of zeros as `spec=zeroslength(tslide),length(signal)`
**for** `jj=1:length(tslide)` **do**
  Establish Gabor filter
  Multiply signal by Gabor filter
  Apply FT to filtered data
  Add filtered data to empty matrix
**end for**

After populating the matrix with transformed values from each window, a spectrogram was constructed to visualize the data.

**Algorithm 3:** Plot spectrogram of transformed data

Use `pcolor` to plot frequency components, tslide, and transformed/filtered data.

# 4 Computational Results

The first part of this study examined the effects of different Gabor window widths and timesteps on spectrograms of a short clip from Handel's *Messiah*. Figure 1 shows the transformed song clip, H(n), and the effect of applying Gabor windows of width 1, 10, 50, and 100 onto the transformed data. As the window increased in size, the amount of resolution of the frequency data appeared to decrease, with fewer points visible in the data that had a Gabor window of 100 applied, as opposed to the data that had window widths of 1 or 10 applied.

Figure 2 shows the spectrogram results of varying window size and timestep. Spectrograms are plots that show the intensity of a frequency signal over time, and are particularly useful in visualizing data that has undergone the GT. The top row of Figure 2 shows the effect of varying the window width while maintaing a constant timestep. As window width increases, the resolution of frequencies per time bin appears to increase. For example, the window width of 1 shows significant blurring of the data, which gets resolved as the window

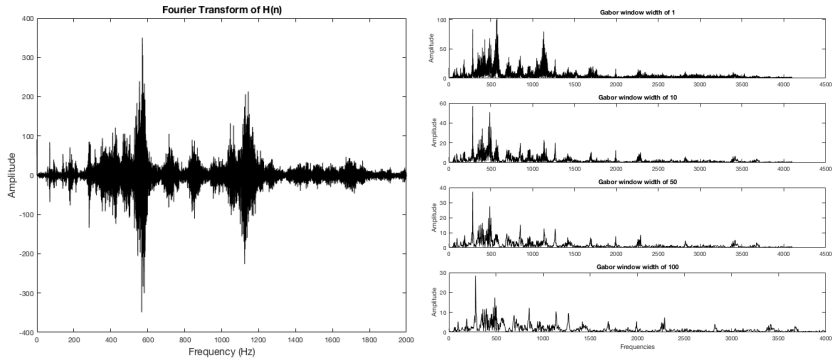


Figure 1: Fourier transform and effect of Gabor window sizes on the sample of Handel's *Messiah*. The FT in the left panel shows increased frequency content in expected areas, such as those frequencies associated with human voices and trumpets, both of which are evident upon listening to the music. Increasing the window width of the GT in the right panel shows that increasing window width reduces resolution of the data.

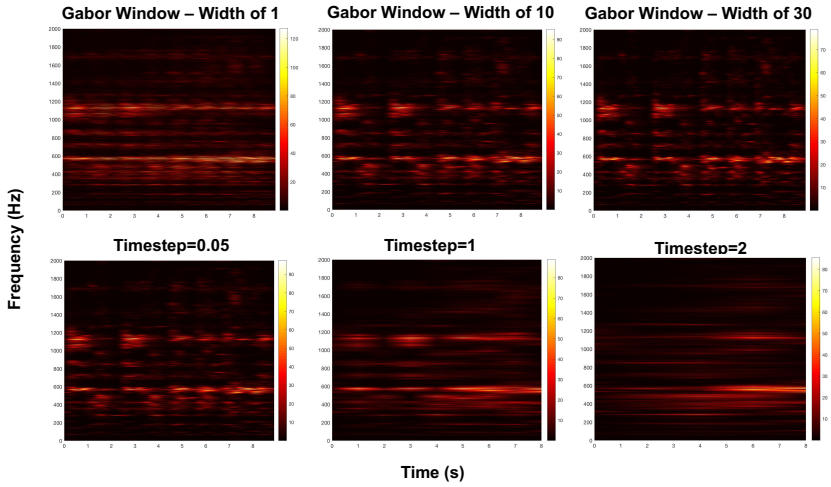


Figure 2: Spectrograms of Handel's *Messiah*. The top row shows the effect of altering the window width of the GT applied to the sample. The bottom row shows the effect of maintaining the same window width (equal to 10) and altering the timestep used in the GT.

size increases to 10 or 30. The bottom row of graphs in Figure 2 show the effect of changing the timestep while window width is kept constant at 10. Increasing the timestep also blurs the data - for example, when the timestep was increased to 2, it is nearly impossible to discern individual frequencies being played at that moment in time.

As in part one, the GT was used to examine two renditions of *Mary Had A Little Lamb* - one played on a piano, and one played on a recorder. The spectrograms of both of these renditions are shown in Figure 3. Both spectrograms showed clear delineation of individual notes that were played, allowing for discernment of the score of the pieces. For the piano, the lowest note played corresponding to a frequency of around 260 Hz, which is middle C. The other notes played on the piano were around 330 Hz, which corresponds to the E above middle C, and 294 Hz, which corresponds to the D above middle C. The full score of the piano rendition was found to be E,D,C,D,E,E,E/D,D,D,E,E,E/E,D,C,D,E,E,E/E,D,D,E,D,C. The recorder registered notes at a higher pitch, with the lowest note recorder to be the A two octaves above middle C, which corresponds to a frequency of around 880 Hz. The other two notes played on the recorder were B ( 988 Hz) and C( 1050 Hz). The score for the recorder was found to be C,B,A,B,C,C,C/B,B,B,C,C,C/C,B,A,B,C,C,C/B,B,C,C,A,A.

Many instruments resonate at the note that is played but also at that note's harmonics, which are equal to 2,3, or 4 times the wavelength of the played note. This provides a "richer" music, which can be qualitatively appreciate when listening to a piano vs. a recorder. The left panel Figure 4 shows the amplitude of the frequencies present in the piano and recorder renditions of the song. The right panel shows the number of instances for the piano and recorder in which values equal to 2,3, or 4 times the first note in the song (E

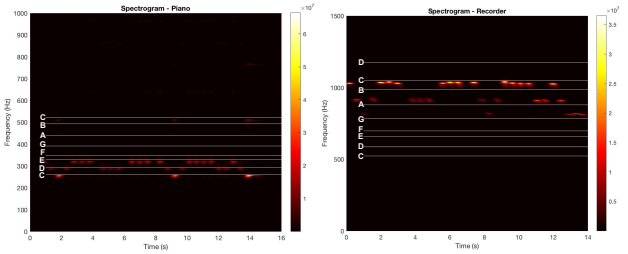


Figure 3: Spectrograms of Mary Had A Little Lamb, piano and recorder. Applying the GT to both datasets allows for clear visualization of the notes being played. The white lines and white letters on the graph indicate the approximate frequencies of the notes at that point.

for the piano, C for the recorder) were present in the spectrogram data. This roughly shows the resonant harmonics present in each sample. Interesting, the recorder shows a large instance of the first harmonic, although this decreases rapidly with each subsequent harmonic. The piano shows a moderate amount of each harmonic, which remains more consistent across.

# 5 Summary and Conclusions

The GT and the spectrogram of its output are extremely useful tools for examining how frequencies in a sample change over time. In this study, the effects of varying window width and time step were shown for a short sample of Handel's *Messiah*. Additionally, using only the GT and visualization with spectrograms, the scores of *Mary Had A Little Lamb* were able to be determined for two different renditions on two different instruments.

The GT takes the uniquely powerful tool of the FT and makes it even more useful by preserving both frequency and temporal resolution. This time-series analysis is extremely useful for looking at things like musical scores, but is also applicable to many other types of real-world data, such as how neural activity in the brain changes over time.

# Appendix A MATLAB Functions

Notable functions used in this code:

- `audioplayer` Creates object to play audio.
- `fft(X)` Performs the fast Fourier transform on X.
- `exp(X)` Returns the exponential $e$ for X.
- `abs(X)` Returns the absolute values of values in X.
- `plot(X,Y)` Plots 2D graph of X vs. Y.
- `zeros` Creates a matrix of zeros of designated input values.
- `pcolor(X,Y,C)` Creates a pseudocolor plot along the axes X and Y using values from matrix C to determine color intensity.
- `shading interp` Sets shading of a color graph by interpolating colormap.
- `colormap` Sets the colormap of a graph. The colormap "hot" was used for spectrograms in this study.
- `colorbar` Attaches a color bar, indicating the colors the correspond to intensity values, to a plot.
- `audioread` Reads in audio data from given file and returns sample and its sampling frequency, Fs.

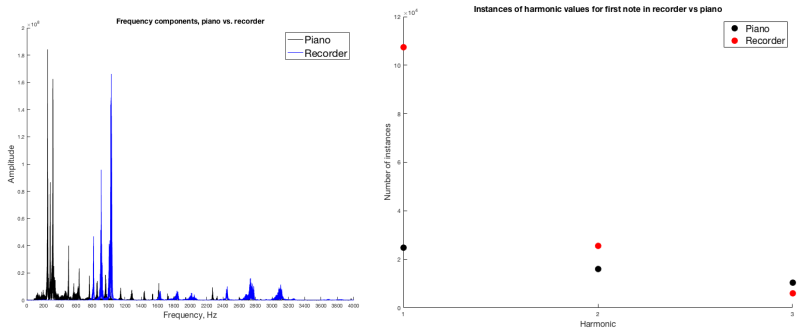


Figure 4: Frequency content and instances of harmonics in both renditions of Mary Had A Little Lamb. The left panel shows the Fourier transformed data for the piano (black) and recorder (blue), showing that the piano has overall a lower set of frequencies than the recorder. The right panel shows the instances of harmonics, that is, the total number of times in each spectrogram that a number 2, 3, or 4 times the frequency of the first note that was played in each song (E, 330Hz for piano and C, 1050Hz, recorder). The recorder has more instances of the first harmonic than the piano does, but sees a sharp decrease with the second and third harmonics as compared to the piano.

- `double(X)` Converts data in X from original data type to a double data type.
- `length(X)` Gives the length of X.
- `for` Iterative statement that loops over given values.
- `if...elseif` Logical statement that evaluates data based on given parameters and executes a function if the parameters are true.

# Appendix B MATLAB Code

```
    %% AMATH582 - Homework 2

%% Handel's Messiah
clear all; close all; clc;

%Load in data
load handel; H=y'/2;
haudio=audioplayer(H,Fs); playblocking(haudio);

figure(1)
plot((1:length(H))/Fs,H,'k'); xlabel('Time (s)')
ylabel('Amplitude'); title('Raw signal of interest, H(n)')

%Establish parameters and frequency range
n=length(H); L=n/Fs; t=(1:n)/Fs;
k=[0:(Fs/2)/(n/2-1):Fs/2];

%Fourier transform and plot of unfiltered data
```

```
Ht=fft(H);

figure(2)
plot(k,Ht(1:length(Hts)/2),'k'); xlabel('Frequency (Hz)','FontSize',[14])
ylabel('Amplitude','FontSize',[14]); title('Fourier Transform of H(n)','FontSize',[14])
xlim([0 2000]);

%Gabor windowing and plot - filter widths of 1, 10, 50, and 100
g1=exp(-1.*(t-4).^2); Hg1=g1.*H; Hg1t=fft(Hg1);

g2=exp(-10.*(t-4).^2); Hg2=g2.*H; Hg2t=fft(Hg2);

g3=exp(-50.*(t-4).^2); Hg3=g3.*H; Hg3t=fft(Hg3);

g4=exp(-100.*(t-4).^2);Hg4=g4.*H; Hg4t=fft(Hg4);

figure(3)
subplot(4,1,1)
plot(k,abs(Hg1t(1:length(Hg1t)/2)),'k'); title('Gabor window width of 1'); ylabel('Amplitude')
subplot(4,1,2)
plot(k,abs(Hg2t(1:length(Hg2t)/2)),'k'); title('Gabor window width of 10'); ylabel('Amplitude')
subplot(4,1,3)
plot(k,abs(Hg3t(1:length(Hg3t)/2)),'k'); title('Gabor window width of 50'); ylabel('Amplitude')
subplot(4,1,4)
plot(k,abs(Hg4t(1:length(Hg4t)/2)),'k'); xlabel('Frequencies'); title('Gabor window width of 100'); ylab
xlim([0 4000])

%% Handel spectrograms using different window widths

%Spectrogram with width 1
tslide=0:0.1:L;
Hg1_spec=zeros(length(tslide),length(H));
for jj=1:length(tslide)
    g1=exp((-1*(t-tslide(jj)).^2)).*cos((t-tslide(jj))*pi);
    Hg1=g1.*H;
    Hg1t=fft(Hg1);
    Hg1_spec(jj,:)=abs(Hg1t);
end
Hg1_spec=Hg1_spec(1:length(tslide),1:length(Hg1_spec)/2);

f=figure('Visible',false);
pcolor(tslide,k,Hg1_spec.'); ylim([0 2000])
shading interp; colormap(hot); colorbar;
print('Handel_Width1.jpeg','-djpeg')
close(f)

%Spectrogram with width 10
tslide=0:0.1:L;
Hg10_spec=zeros(length(tslide),length(H));
for jj=1:length(tslide)
    g10=exp((-10*(t-tslide(jj)).^2)).*cos((t-tslide(jj))*pi);
    Hg10=g10.*H;
    Hg10t=fft(Hg10);
    Hg10_spec(jj,:)=abs(Hg10t);
```

```
end
Hg10_spec=Hg10_spec(1:length(tslide),1:length(Hg10_spec)/2);

f=figure('Visible',false);
pcolor(tslide,k,Hg10_spec.'); ylim([0 2000])
shading interp; colormap(hot); colorbar;
print('Handel_Width10.jpeg','-djpeg')
close(f)

%Spectrogram with width 30
tslide=0:0.1:L;
Hg30_spec=zeros(length(tslide),length(H));
for jj=1:length(tslide)
    g30=exp((-30*(t-tslide(jj)).^2)).*cos((t-tslide(jj))*pi);
    Hg30=g30.*H;
    Hg30t=fft(Hg30);
    Hg30_spec(jj,:)=abs(Hg30t);
end
Hg30_spec=Hg30_spec(1:length(tslide),1:length(Hg30_spec)/2);

f=figure('Visible',false);
pcolor(tslide,k,Hg30_spec.'); ylim([0 2000])
shading interp; colormap(hot); colorbar;
print('Handel_Width30.jpeg','-djpeg')
close(f)

%% Handel spectrograms with varying timesteps, window width of 10
tslide5=0:0.05:L; %Timestep of 0.05
Hg5_spec=zeros(length(tslide5),length(H));
for jj=1:length(tslide5)
    g5=exp((-10*(t-tslide5(jj)).^2)).*cos((t-tslide5(jj))*pi);
    Hg5=g5.*H;
    Hg5t=fft(Hg5);
    Hg5_spec(jj,:)=abs(Hg5t);
end
Hg5_spec=Hg5_spec(1:length(tslide5),1:length(Hg5_spec)/2);

f=figure('Visible',false);
pcolor(tslide5,k,Hg5_spec.'); ylim([0 2000])
shading interp; colormap(hot); colorbar;
print('Handel_TimeStep0.05.jpeg','-djpeg')
close(f)

%Timestep of 1
tslide1=0:1:L;
Hg1_spec=zeros(length(tslide1),length(H)); %Spectrogram with width 10
for jj=1:length(tslide1)
    g1=exp((-10*(t-tslide1(jj)).^2)).*cos((t-tslide1(jj))*pi);
    Hg1=g1.*H;
    Hg1t=fft(Hg1);
    Hg1_spec(jj,:)=abs(Hg1t);
end
Hg1_spec=Hg1_spec(1:length(tslide1),1:length(Hg1_spec)/2);
```

```
f=figure('Visible',false);
pcolor(tslide1,k,Hg1_spec.'); ylim([0 2000])
shading interp; colormap(hot); colorbar;
print('Handel_TimeStep1.jpeg','-djpeg')
close(f)

%Timestep of 2
tslide2=0:2:L;
Hg2_spec=zeros(length(tslide2),length(H)); %Spectrogram with width 10
for jj=1:length(tslide2)
    g2=exp((-10*(t-tslide2(jj)).^2)).*cos((t-tslide2(jj))*pi);
    Hg2=g2.*H;
    Hg2t=fft(Hg2);
    Hg2_spec(jj,:)=abs(Hg2t);
end
Hg2_spec=Hg2_spec(1:length(tslide2),1:length(Hg2_spec)/2);

f=figure('Visible',false);
pcolor(tslide2,k,Hg2_spec.'); ylim([0 2000])
shading interp; colormap(hot); colorbar;
print('Handel_TimeStep2.jpeg','-djpeg')
close(f)

%% Mary had a little lamb
%% Setting up parameters and determining frequency components
close all; clear all; clc;

tp=16;  % record time in seconds - piano
p=audioread('music1.wav','native');
p=double(p); np=length(p'); Fsp=np/tp; Lp=np/Fsp; tp=(1:np)/Fsp;
kp=[0:(Fsp/2)/(np/2-1):Fsp/2];
plot((1:np)/Fsp,p); xlabel('Time [sec]'); ylabel('Amplitude'); title('Mary had a little lamb (piano)');
% p8 = audioplayer(p,Fs); playblocking(p8);

tr=14;  % record time in seconds
r=audioread('music2.wav','native');
r=double(r); nr=length(r'); Fsr=nr/tr; Lr=nr/Fsr; tr=(1:nr)/Fsr;
kr=[0:(Fsr/2)/(nr/2-1):Fsr/2];
plot((1:nr)/Fsr,r); xlabel('Time [sec]'); ylabel('Amplitude'); title('Mary had a little lamb (recorder)
% p8 = audioplayer(y,Fs); playblocking(p8);

%Look at frequenices included in the songs
pt=fft(p); rt=fft(r);

figure(1)
hold on
plot(kp,abs(pt(1:length(pt)/2)),'k')
plot(kr,abs(rt(1:length(rt)/2)),'b')
xlim([0 4000]);title('Frequency components, piano vs. recorder','Fontsize',[14])
ylabel('Amplitude','Fontsize',[16]); xlabel('Frequency, Hz','Fontsize',[16])
xticks([0:200:4000]); legend('Piano','Recorder','Fontsize',[20]);


%% Spectrograms for recorder and piano pieces
```

```
middleCoctave=[261.63,293.66,329.63,349.23,392,440,493.88,523.55]; %Scale from middle C to 1 octave abo

tpslide=0:0.05:Lp;
pspec=zeros(length(tpslide),length(p));
for jj=1:length(tpslide)
    g_piano=exp((-20*(tp-tpslide(jj)).^2)).*cos((tp-tpslide(jj))*pi);
    pg=g_piano.*p';
    pgt=fft(pg);
    pspec(jj,:)=abs(pgt);
end
pspec=pspec(1:length(tpslide),1:length(pspec)/2);

octaveP=zeros(length(tpslide),length(middleCoctave));
for o=1:length(octaveP)
    octaveP(o,:)=middleCoctave(:);
end

f=figure('Visible',false);
hold on
pcolor(tpslide,kp,pspec.')
shading interp; colormap(hot); colorbar
plot(octaveP,'-w')
xlim([0 max(tpslide)]); ylim([0 1000]);
xlabel('Time (s)'); ylabel('Frequency (Hz)'); title('Spectrogram - Piano')
print('PianoSpec.jpeg','-djpeg')
close(f)

highCoctave=[523.55, 587.33,659.26,698.46,783.99,880,987.77,1049.5,1174.7];
trslide=0:0.05:Lr;
rspec=zeros(length(trslide),length(r));
for j=1:length(trslide)
    g_rec=exp((-20*(tr-trslide(j)).^2)).*cos((tr-trslide(j))*pi);
    rg=g_rec.*r';
    rgt=fft(rg);
    rspec(j,:)=abs(rgt);
end
rspec=rspec(1:length(trslide), 1:length(rspec)/2);

octaveR=zeros(length(trslide),length(highCoctave));
for o=1:length(octaveR)
    octaveR(o,:)=highCoctave(:);
end

f=figure('Visible',false);
hold on
pcolor(trslide,kr,rspec.');
shading interp; colormap(hot); colorbar; ylim([0 1500])
plot(octaveR,'-w')
xlim([0 max(trslide)]);
xlabel('Time (s)'); ylabel('Frequency (Hz)'); title('Spectrogram - Recorder')
print('RecorderSpec.jpeg','-djpeg')
close(f)
```

```
%% Determine overtones in piano and recorder signals
%Look at number of instances where spectrogram data is within 1, 2, or 3
%harmonics of first note in song

%~330Hz=E, first note in piano score; ~1050Hz=C, first note in recorder
%score
countph1=0; countph2=0; countph3=0;
for j=1:length(pspec)
    if 330<pspec(j) && pspec(j)<(330*2)
        countph1=countph1+1;
    elseif (330*2)<pspec(j) && pspec(j)<(330*3)
        countph2=countph2+1;
    elseif (330*3)<pspec(j) && pspec(j)<(330*4)
        countph3=countph3+1;
    end
end

countrh1=0; countrh2=0; countrh3=0;
for j=1:length(rspec)
    if 1050<rspec(j) && rspec(j)<(1050*2)
        countrh1=countrh1+1;
    elseif (1050*2)<rspec(j) && rspec(j)<(1050*3)
        countrh2=countrh2+1;
    elseif (1050*3)<rspec(j) && rspec(j)<(1050*4)
        countrh3=countrh3+1;
    end
end

countsp=[countph1 countph2 countph3]; countsr=[countrh1 countrh2 countrh3]; harmonics=[1 2 3];

figure()
hold on
plot(harmonics, countsp,'ok','MarkerSize',[10],'MarkerFaceColor','k')
plot(harmonics, countsr,'or','MarkerSize',[10],'MarkerFaceColor','r')
xlabel('Harmonic','FontSize',[14]); ylabel('Number of instances','FontSize',[14]);
title('Instances of harmonic values for first note in recorder vs piano','FontSize',[14])
legend('Piano','Recorder','FontSize',[16]); xticks([0:1:3]);
```