AMATH 482 Winter 2020
Homework 2: Gábor Transforms

Ozan Erdal

February 7, 2020

1 Introduction and Overview

The frequency signature of a signal can be analyzed using Fourier transforms, however the transform lacks necessary information about the signal. When viewing the signal in frequency space, all information about time is lost. For a signal that is periodic and relatively uniform, Fourier analysis is sufficient, but with most real-world data and signals, processing and analyzing them without time information is not enough. For example, given a signal corresponding to a sound or a song, the amplitude of the signal is discrete and generally different at different points in time.

In this case, more advanced techniques, like the Gábor transform and the Wavelet transform can be used to gather information on both time and frequency. This style of signal processing is referred to as *time-frequency analysis*, and for the signals in question, graphs called spectrograms are also utilized for providing a time-frequency "fingerprint" for a signal.

For the purpose of this experiment, three different sound files were used, one of which comes downloaded with MATLAB, two of which were downloaded from the course page. The first piece was an 8 second excerpt from George Frideric Handel's *Messiah*. The second and third songs are the classic tune *Mary had a little lamb* played on the recorder and piano. By using time-frequency analysis, a trove of information about the music can be discovered. In particular, by determining the frequency at the time of each note being played, a musical score can be generated. Since *Mary had a little lamb* is a relatively simple piece, the "fingerprint" of the frequencies will place each note at a specific frequency once the piece has been filtered and transformed into a time-frequency signature.

2 Theoretical Background

The fundamental idea behind time-frequency analysis results from the biggest shortcoming of standard Fourier analysis - a lack of time information. By transforming the entire signal, all frequency information is captured but with no localization as to the time which the frequencies are occurring at. Additionally, by applying a filter to the signal, parts of the signal can be removed to isolate sections of interest. A narrower filter will remove greater portions of the signal, and an infinitely wide filter will result in an unaltered signal. Time-frequency analysis combines these concepts. By systematically isolating and analyzing the Fourier transform at various times, a more complete picture can be constructed. The effects of various filters applied to the signal corresponding to Handel's *Messiah* are shown in **Figure 1** with their respective transformations.

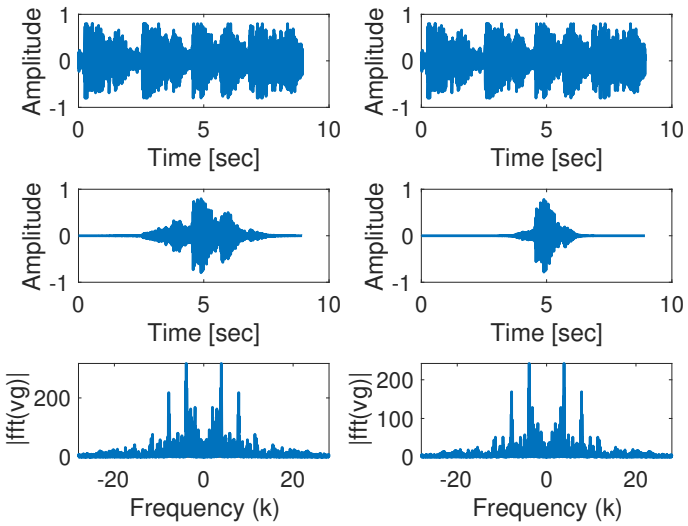

Figure 1: Filters of 2 different widths applied.

Additionally, the Wavelet transform already makes use of varying width filters. Instead of using a sliding window (changing t iteratively), the Wavelet transform addresses an issue with the Gábor transform in which different window sizes must be used to gather different information by using a scaling and sliding window. Wavelets themselves are simply periodic functions other than sine and cosine which have been progressively refined to provide more effective time-frequency analysis. A few examples of wavelets are shown below in **Figure 2**, and depending on the signal being filtered and transformed, a different wavelet may be more or less effective. A general rule of thumb being that a more similar wavelet to the frequencies present will be better. Additionally, multiplying a wavelet with an integrable function produces a different wavelet. In theory, Wavelets can be used to provide greater information on time in areas of high frequency and more information on the actual frequency for the lower frequencies.

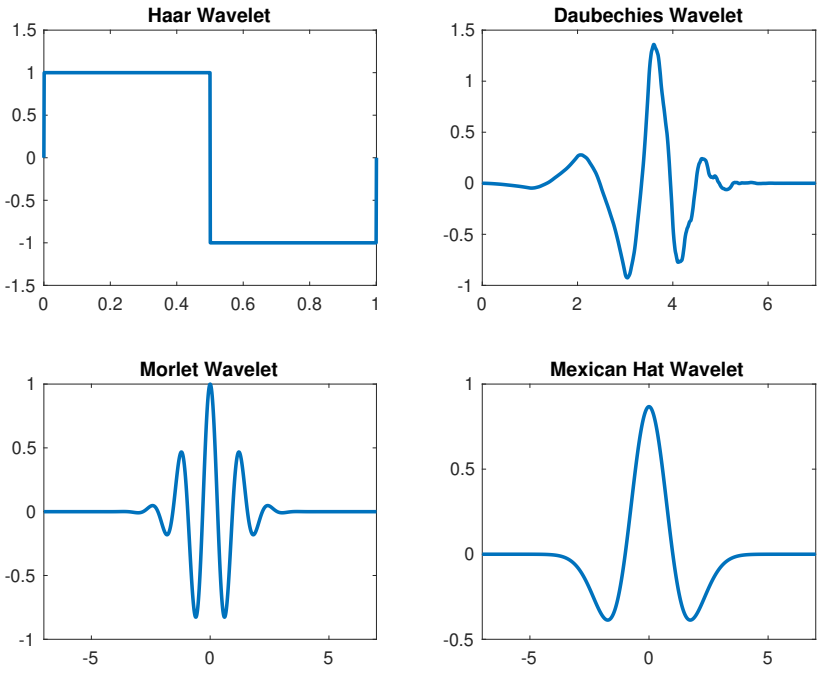

Figure 2: 4 common Wavelet shapes.

For the purposes of the Handel piece, both Gábor and Wavelet transforms, as well as MATLAB's *Signal Processing Toolbox* and *Wavelet Toolbox* were used to find patterns and other information about the music. For the purpose of *Mary had a little lamb*, only the Gábor transform and the spectrograms generated from sliding and repeatedly extracting information at different window centers were needed to ascertain the score.

The `.wav` files contain the audio for *Mary had a little lamb* on the piano and record respectively, which would have a key difference even if they were to be played on the same exact notes. A piano is a instrument that plays notes by striking strings, even if it may not seem as though from how it appears to be played. A recorder, however, is a blown instrument that sounds very different. This "difference" in sound when playing the same notes is a concept referred to as *timbre*, and it is related to the different frequencies produced by the two instruments. Wind instruments are among the many that produce *overtones*, which appear in the frequency signature of a piece as integer multiples of the *fundamental frequency*. The fundamental frequency is the only frequency which appears in a time-frequency analysis of a piano piece. Although these overtones can be misleading in a time-frequency analysis, they can either be filtered out or cropped from the spectrogram manually since the fundamental frequency or *zeroth harmonic* is indicative of the frequency of the note actually being played. Overtones appear quite faintly in a spectrogram and can be ignored, but it is important to note their existence.

Algorithm Implementation and Development

Handel's *Messiah*

For this piece, a variety of techniques were used. One of these was the Gábor transform, and spectrograms were generated for a variety of window sizes. Using a list of values for a instead of a single one gives a better picture, and since this piece had significantly less samples in total, a smaller step size was able to be used without sacrificing too much computational strength (i.e. without lagging the computer too much), allowing for 200 windows. Scaling parameters of $a = [0.1, 0.5, 1, 5]$ and $\Delta t = 0.0902$ were used.

Mary had a little lamb

Through usage of various filter widths, a scaling parameter of $a = 20$ was determined to provide a sufficient amount of time and frequency information, although plots of widths greater and less than this value were analyzed to better confirm the lengths and frequencies of the notes. Using this value for a and 100 windows, a sliding step of $\Delta t = 0.1607s$ was calculated. To calculate the spectrogram, a Gaussian filter was created with the selected value of a and applied across the entire signal at steps of Δt. At each step, the resulting filtered signal was then transformed into the Fourier domain using the *fft()* function. Storing each filtered and transformed signal and then plotting them using *pcolor()* produces a spectrogram.

The same process was repeated for the signal corresponding to the recorder, adjusting the axes for plotting in both cases. Since the recorder file was of different length, a different step size of $\Delta t = 0.1438$ was calculated.

Computational Results

Handel's *Messiah*

By using a variety of widths, multiple spectrograms were generated. These are shown below in **Figure 3** and labeled with the associated value of a.

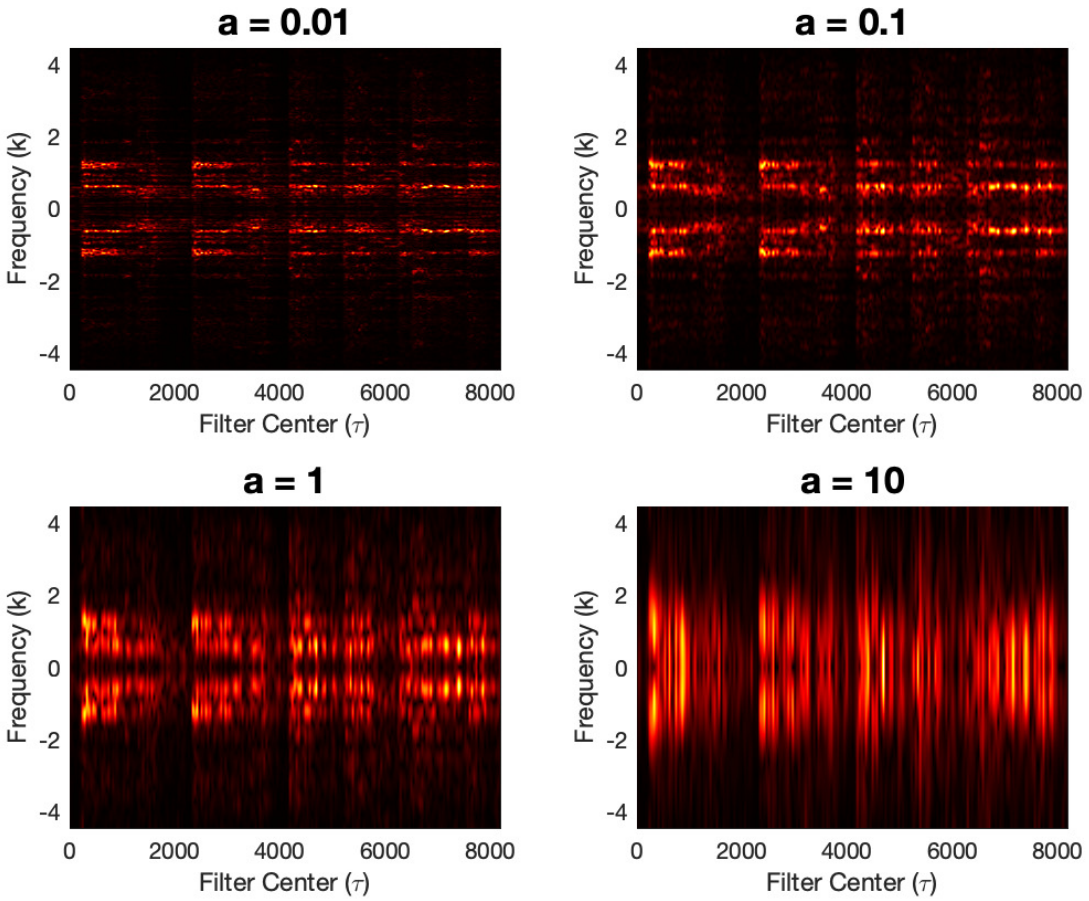

Figure 3: Various spectrograms for Handel's *Messiah*.

Mary had a little lamb

The piano spectrogram is shown below in **Figure 4**.

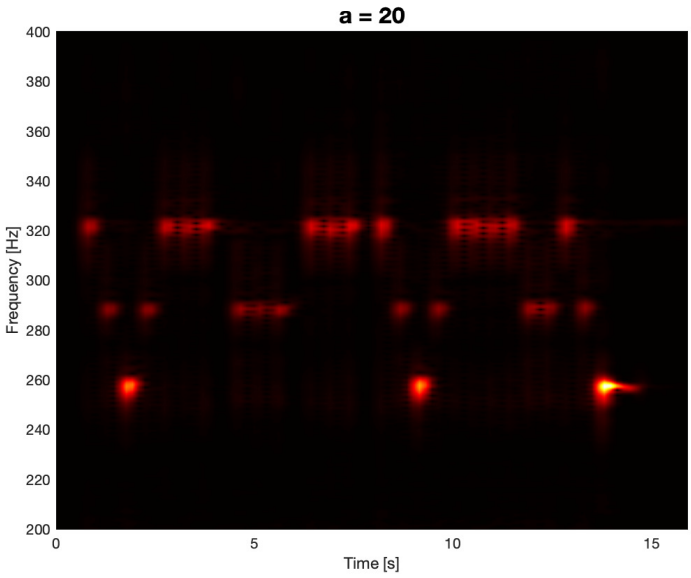

Figure 4: Spectrogram for *Mary had a little lamb* on the piano.

From this spectrogram, the frequencies of the notes are clearly observed to occur at 3 distinct frequencies: $f \approx 320Hz, 290Hz, 260Hz$. These notes correspond to E,D, and C respectively.

The recorder spectrogram with and without overtones are shown in **Figure 5** and **Figured 6** respectively.

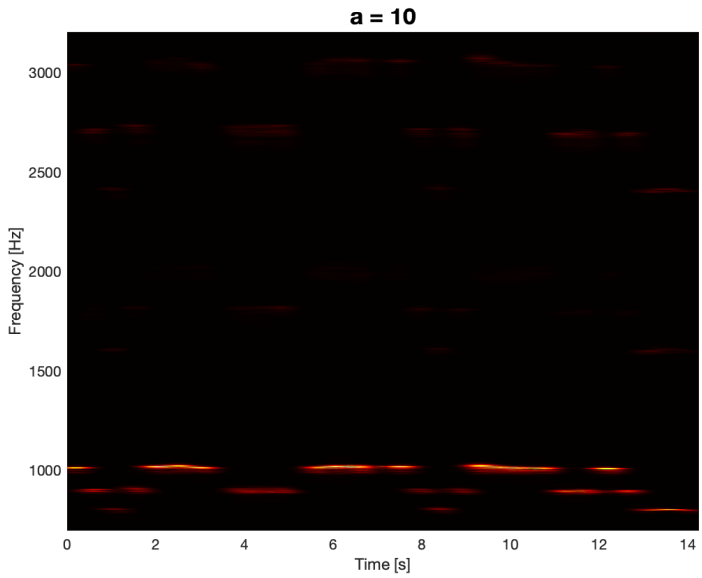

Figure 5: Spectrogram for *Mary had a little lamb* on the recorder with overtones.

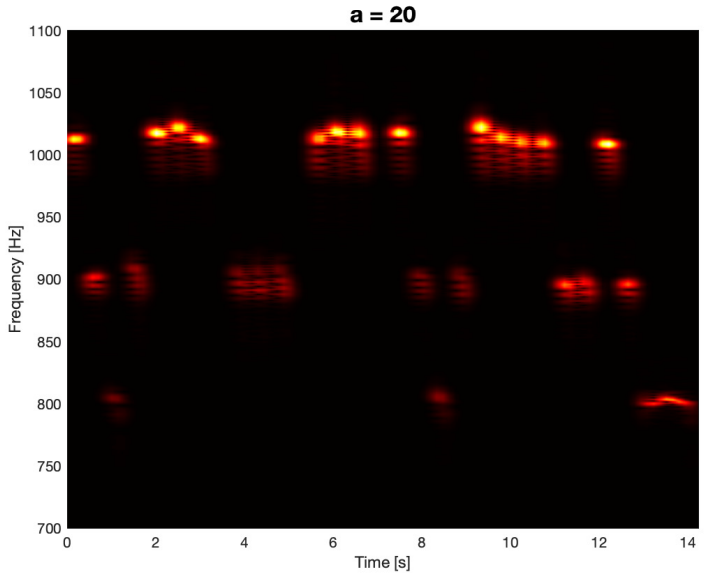

Figure 6: Spectrogram for *Mary had a little lamb* on the recorder without overtones.

From the spectrogram without overtones, the frequencies of the notes are clearly observed to occur at 3 frequencies that are much less well-defined, but averaging at around: $f \approx 1000Hz, 900Hz, 800Hz$. The closest notes that these notes correspond to are C,B, and A respectively. Although the recorder is not as clear-cut as the piano, listening to the audio file and by using an online recorder playing software to play the determined score, the notes determined are the closest in terms of frequency and auditory similarity.

The scores constructed for the two pieces are shown below in **Figure 7**.

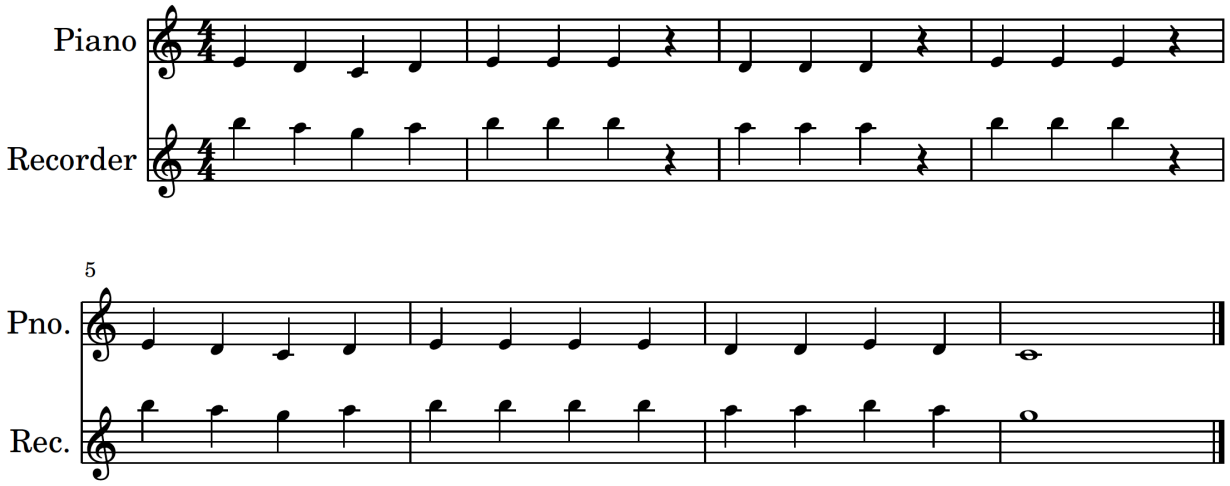

Figure 7: Score for *Mary had a little lamb* on the piano and recorder.

Summary and Conclusions

As evidenced in the previous project and this one alike, Fourier transforms and the information obtainable from viewing a signal in frequency and time-frequency space is significant. For relatively simple songs, the score can be determined from even the most basic time-frequency analysis techniques. More advanced techniques like Wavelet transforms and Multi-Resolution Analysis were not even required. For individual musicians, trends show themselves in the spectrograms and time-frequency analysis can be used as a means for distinguishing certain composers from others. However, it is unlikely that a Gábor transform would be sufficient to unearth the score of a more complicated piece, like Handel's *Messiah*.

Wavelet transforms are also very useful for classical music in particular since there is a wider range of frequencies present in a symphony or philharmonic. By utilizing different types of Wavelets, one that fits well best for a situation can be used in the eventual analysis.

Appendix A

audioread(), audioplayer(), and playblocking()

Used to read-in and generate an audioplayer from a vector of samples and amplitudes and subsequently play them from MATLAB.

linspace(start, end, n)

Generates 100 evenly spaced points from the first parameter to the second. If the third parameter is passed, instead generate n points from *start* to *end*.

pcolor(x, y, c)

Plots a pseudocolor plot on the x-y coordinate grid. The values of c determine the intensity of the color.

Setting *colormap('hot')* produces the black-red-white coloring.

wavefun()

Uses *Wavelet Toolkit* to generate from a handful of common wavelets.

Appendix B

Loading and plotting the various pieces

```
%% Handel
load handel
v = y';
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

%% Mary had a little lamb (Piano)
[y,Fs] = audioread('music1.wav');
tr_piano=length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)');

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

Plotting the different Gaussian filters

```
a1 = 0.5;
a2 = 2;
g1 = exp(-a1*(t-tau).^2);
g2 = exp(-a2*(t-tau).^2);

vg1 = v.*g1;
vg2 = v.*g2;
vg1t = fft(vg1);
vg2t = fft(vg2);

subplot(3,2,1)
plot(t,v,'LineWidth',2)
set(gca,'Fontsize',16), xlabel('Time [sec]'), ylabel('Amplitude')
subplot(3,2,3)
plot(t,vg1,'LineWidth',2)
set(gca,'Fontsize',16), xlabel('Time [sec]'), ylabel('Amplitude')
subplot(3,2,5)
plot(ks,abs(fftshift(vg1t)),'LineWidth',2)
set(gca,'Fontsize',16), xlabel('Frequency (k)'), ylabel('|fft(vg)|')
subplot(3,2,2)
plot(t,v,'LineWidth',2)
set(gca,'Fontsize',16), xlabel('Time [sec]'), ylabel('Amplitude')
subplot(3,2,4)
plot(t,vg2,'LineWidth',2)
set(gca,'Fontsize',16), xlabel('Time [sec]'), ylabel('Amplitude')
subplot(3,2,6)
plot(ks,abs(fftshift(vg2t)),'LineWidth',2)
set(gca,'Fontsize',16), xlabel('Frequency (k)'), ylabel('|fft(vg)|')
print -depsc gaussian_ex.eps
```

Plotting the Wavelets

```
close;
figure(2)
subplot(2,2,1)
[phi_haar,psi_haar,xval_haar] = wavefun('haar',10);
plot(xval_haar,psi_haar,'LineWidth',2); title('Haar Wavelet');
subplot(2,2,2)
[phi_db,psi_db,xval_db] = wavefun('db4',10);
plot(xval_db,psi_db,'LineWidth',2); title('Daubechies Wavelet'); xlim([0,7]);
subplot(2,2,3)
[psi_morl,xval_morl] = wavefun('morl',10);
plot(xval_morl,psi_morl,'LineWidth',2); title('Morlet Wavelet'); xlim([-7,7]);
subplot(2,2,4)
[psi_mexh,xval_mexh] = wavefun('mexh',10);
plot(xval_mexh,psi_mexh,'LineWidth',2); title('Mexican Hat Wavelet'); xlim
   ([-7,7]);
print -depsc wavelet_demo.eps
```

Constructing the Spectrogram

```
close;
v = y';
n = length(v);
L = tr_piano; % switch to 'L = tr_rec' for recorder
ts = linspace(0,L,n+1); t = ts(1:n);
k = (1/L)*[0:(n/2-1) -n/2:-1]; % use hertz instead of radians
ks = fftshift(k);
a = 20;
tslide = linspace(0,L);
vgt_spec = zeros(length(tslide),n);

for j = 1:length(tslide)
    g = exp(-a*(t-tslide(j)).^2);
    vg = v.*g;
    vgt = fft(vg);
    vgt_spec(j,:) = fftshift(abs(vgt));
end
```

Plotting the Spectrograms

```
%% Plot the Piano Spectrogram
figure(4)
pcolor(tslide,ks,vgt_spec.'), shading interp
title(['a = ', num2str(a)],'FontSize',16)
ylim([200,400])
xlabel('Time [s]'), ylabel('Frequency [Hz]')
colormap('hot')
print -depsc mary_spec.eps

%% Plot the Recorder Spectrogram
figure(5)
pcolor(tslide,ks,vgt_spec.'), shading interp
title(['a = ', num2str(a)],'FontSize',16)
ylim([700,3200])
xlabel('Time [s]'), ylabel('Frequency [Hz]')
colormap('hot')
print -depsc mary_spec_rec_overtones.eps

%% Plot the Recorder Spectrogram (without overtones)
```

```
figure(6)
pcolor(tslide,ks,vgt_spec.'), shading interp
title(['a = ', num2str(a)],'FontSize',16)
ylim([700,1100])
xlabel('Time [s]'), ylabel('Frequency [Hz]')
colormap('hot')
print -depsc mary_spec_rec.eps
```