Music 270a: Signal Analysis

Tamara Smyth, trsmyth@ucsd.edu
Department of Music,
University of California, San Diego (UCSD)

November 23, 2015

Signal Analysis

- Some tools we may want to use to automate analysis are:
  1. Amplitude Envelope Follower
  2. Peak Detection (attacks or harmonics), surfboard method
  3. Pitch Detection, harmonics vs. chaotic signal
  4. Frequency/Spectral Envelope (formant tracking, mccs or lpc)
  5. Constant overlap-add (COLA)

Envelope Follower (Amplitude detection)

- An envelope follower will essentially determine the amplitude envelope without dipping down into the valleys/zero crossings.
- It is a reduction of the information, representing the overal shape of the signal's amplitude (i.e. amplidtude envelpe) without the higher frequency information.
- The amplitude envelope $y(n)$ is given by

  $$y(n) = (1 - \nu)|x(n)| + \nu y(n-1),$$

  where ν determines how quickly changes in $x(n)$ are tracked:
  - if ν is close to one, changes are tracked slowly
  - if ν is close to zero, $x(n)$ has an immediate influence on $y(n)$.
- In order to capture attacks in the signal, the value for ν is usually smaller for an increasing signal and large for one that is decreasing.
- See envfollower.m

Peak Detection

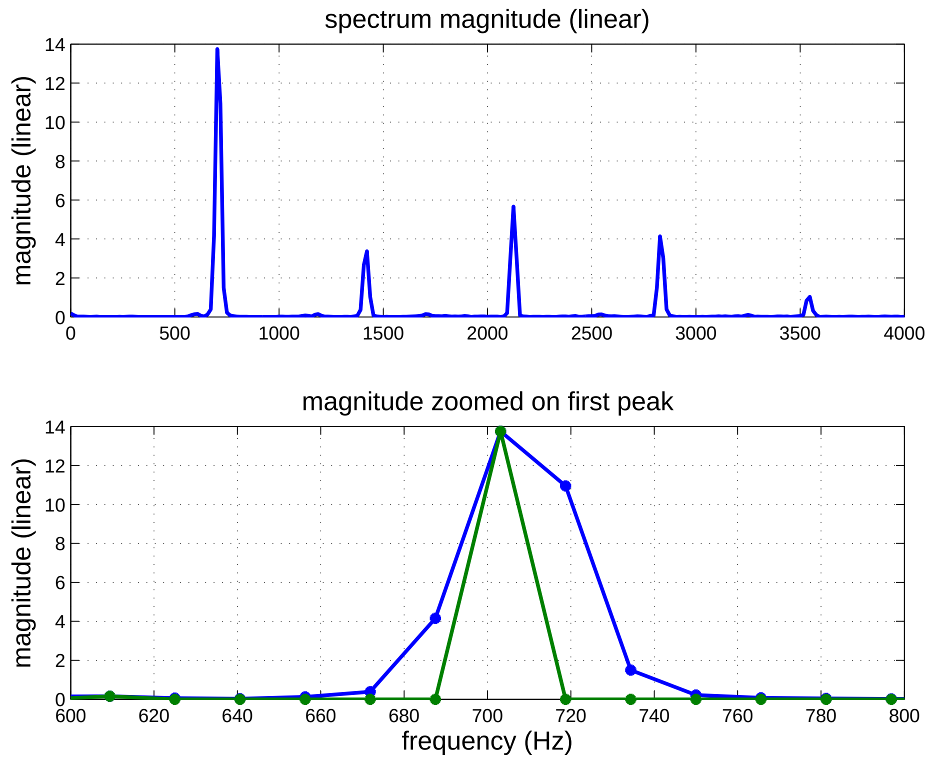

Figure 1: Peak Detection

```
% threshhold
th = 0;

% filter descending values
uslope = Ymag > [Ymag(1); Ymag(1:end-1)];

% filter ascending values
dslope = Ymag >= [Ymag(2:end); 1+Ymag(end)];

% only indeces at maxima retain non-zero value
Ymax = Ymag .* (Ymag > th) .* uslope .* dslope;

% peak indeces
maxixs = find(Ymax);
```

Quadratic interpolation

- The position of the peak is limited by the resolution of the DFT/FFT and its estimation can be improved using quadratic interpolation.
- The general equation for a parabola is given by

$$y(x) \triangleq a(x-p)^2 + b,$$

where p is the peak location and $b = y(p)$.

- Considering the parabola at points $x = 0, 1, -1$ yields 3 equations,

$$\begin{aligned} y(0) &= ap^2 + b = \beta \\ y(1) &= a(1 + p^2 - 2p) + b = \gamma \\ y(-1) &= a(1 + p^2 + 2p) + b = \alpha, \end{aligned}$$

and 3 unknowns a, p, b.

- Solve for a:

$$\alpha - \gamma = 4ap \longrightarrow a = \frac{\alpha - \gamma}{4p}.$$

- Solve for p using a:

$$\begin{aligned}
\gamma - \beta &= a - 2ap \\
&= \frac{\alpha - \gamma}{4p} - 2\frac{\alpha - \gamma}{4p}p \\
4p(\gamma - \beta) &= \alpha - \gamma - 2(\alpha - \gamma)p \\
4p(\gamma - \beta) + 2p(\alpha - \gamma) &= \alpha - \gamma \\
2p(\gamma - 2\beta + \alpha) &= \alpha - \gamma \\
p &= \frac{\alpha - \gamma}{2(\gamma - 2\beta + \alpha)}.
\end{aligned}$$

- Finally, the height at peak p is given by

$$\begin{aligned}
y(p) &= b \\
&= \beta - ap^2 \\
&= \beta - \frac{\alpha - \gamma}{4p}p^2.
\end{aligned}$$

- Another way (Dan Ellis)

Pitch Detection

- Considerations for Computer Music applications:
  1. Signals are often noisy, eg: poor soundcards, other instruments/voices,
  2. How much frequency resolution is needed? Correct octave a must, but will a semitone suffice?
  3. What latency can be tolerated (what framesize should be used for analysis?)
  4. Does the instrument have well-defined/behaved harmonics?
- In the paper by de la Cuadra et al., "Efficient Pitch Detection Techniques for Interactive Music", four (4) pitch detection algorithms are summarized:
  1. Harmonic Product Spectrum
  2. Maximum Likelihood
  3. Cepstrum-Biased HPS
  4. Weighted Autocorrelation Function

Harmonic Product Spectrum (HPS)

- HPS (Noll 1969) measures the maximum coincidence for harmonics for each spectral frame according to

$$Y(\omega) = \prod_{r=1}^{R} |X(\omega r)|, \tag{1}$$

where R is the number of harmonics being considered.

- The resulting periodic correlation array $Y(\omega)$ is then searched for a maximum value of a range of possible fundamental frequencies ω_i

$$\hat{Y} = \max_{\omega_i} Y(\omega_i) \tag{2}$$

to obtain the fundamental frequency estimate.

- Octave errors are common (detection is sometimes an octave too high).
- To correct, apply this rule: if the second peak amplitude *below* initially chosent pitch is approximately 1/2 of the chosen pitch AND the ratio of amplitudes is above a threshold (e.g., 0.2 for 5 harmonics), THEN select the lower octave peak as the pitch for the current frame.

- Due to noise, frequencies below about 50 Hz should not be searched for a pitch.
- Pros: HPS is simple to implement, does well under a wide range of conditions, and **runs in real-time**.
- Cons: low frequency resolution must be enhanced by zero-padding, so that the spectrum can be interpolated to the nearest semitone. This means that high frequencies are also being unecessarily interpolated.
- See hps.m

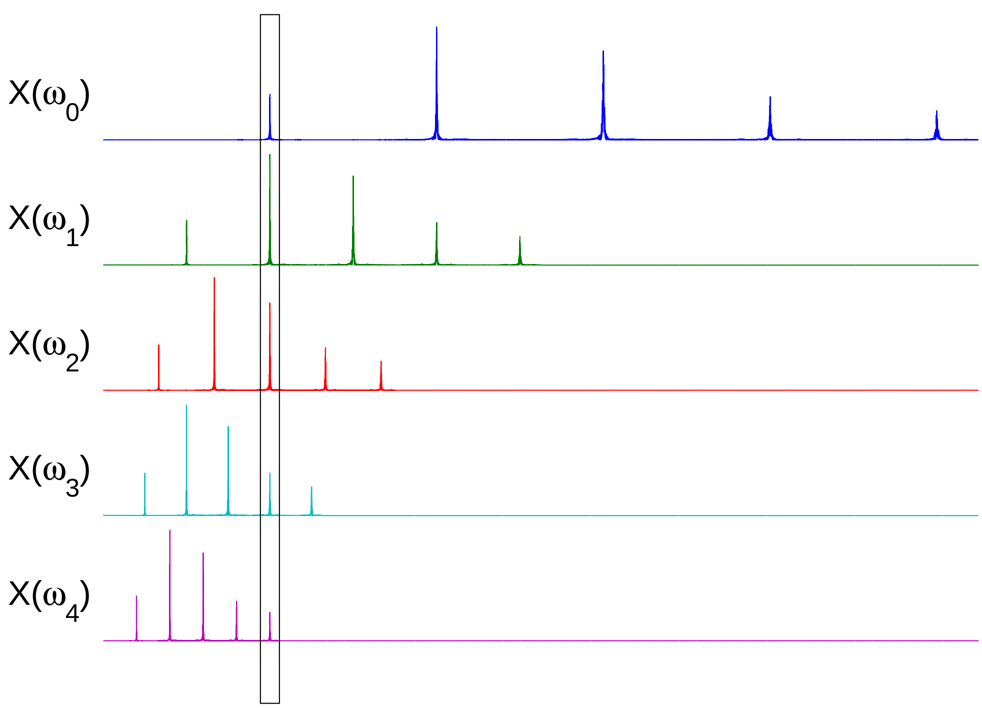

Figure 2: Harmonic Product Spectrum

Uses of Linear Predictive Coding (LPC)

- LPC, a statistical method for predicting future values of a waveform on the basis of its past values[1], is often used to obtain **a spectral envelope**.
- LPC differs from formant tracking in that:
  - the waveform remains in the time domain; resonances are described by the coefficients of an all-pole filter.
  - altering resonances is difficult since editing IIR filter coefficients can result in an unstable filter.
  - analysis may be applied to a wide range of sounds.
- LPC is often used to determine the filter in a source-filter model of speech[2] which:
  - characterizes the response of the vocal tract.
  - reconstitutes the speech waveform when driven by the correct source.

[1] Markel, J. D., and Gray, A. H., Hr. *Linear Prediction of Speech.* New York: Apringer-Verlag, 1976.

[2] In a source-filter model of speech, there is assumed to be no feedback dependency between the vibrating vocal folds and the vocal track

Concept of LPC

- Given a digital system, can the value of any sample be predicted by taking a linear combination of the previous N samples?
- Stated mathematically, can a set of coefficients, a_k, be determined such that

  $$y(n) = a_1 y(n-1) + a_2 y(n-2) + ... + a_N y(n-N).$$

- That is, can a signal be represented as coefficients of an all-pole (only feedback terms) IIR filter.
- If yes, then the coefficients and the first N samples would completely determine the remainder of the signal, because the rest of the samples can be calculated by the above equation.

LPC residual

- The answer is actually *"not precisely"* for a finite N.
- Rather, we determine the cofficients that give the *best* prediction by minimizing the difference, or error $e(n)$, between the actual sample values of the input waveform $y(n)$ and the waveform re-created using the derived predictors $\hat{y}(n)$.

$$\min_n\{e(n)\} = \min_n\{\hat{y}(n) - y(n)\}.$$

- The smaller the average value of the *error*, also called the *residual*, the better the set of predictors.
- The residual may be used to exactly reconstruct the original signal $y(n)$ by using it as an input to our all-pole filter, that is

$$y(n) = b_0e(n)+a_1y(n-1)+a_2y(n-2)+...+a_Ny(n-N)$$

  where b_0 is a scaling factor that gives the correct amplitude.
- When the speech is voiced, the residual is essentially a periodic pulse waveform with the same fundamental frequency as the speech. When unvoiced, the residual is similar to white noise.

LPC Order

- The accuracy of the predictor improves with an increase of order N:
- The smallest value of N that will yield sufficiently quality in the speech representation is related to
  1. the highest frequency in the speech,
  2. the number of formant peaks expected and
  3. the sampling rate.
- There is no exact relationship for determining what value of N should be used, but in general it's between 10 and 20 or the sampling rate in kHz plus 4 (for $fs = 15$kHz, you might use a $N = 19$).
- Schemes for determining predictors by minimizing the residual work either explicitly or implicitly.
- A more rigorous treatment of the subject can be found in J. Makhoul's tutorial[3]

[3] Makhoul, J. "Linear Prediction, a Tutorial Review." *Proceedinds of the institute of Electrical and Electronics Engineers*, 63, 1975, 561-580.

LPC Analysis in Matlab

- Matlab for generating original, lpc, and residual spectra.

```
%lpc
order = fs/1000 + 5; % order
xlpc = lpc(xw, order)'; % coefficients

%windowed speech frequency response
zpf = 3; Nfft = 2^nextpow2(N*zpf);
XW = fft(xw, Nfft);

% lpc spectrum
[H,w] = freqz(1, xlpc, Nfft, 'whole');

%inverse filter to obtain residual
E = XW./H;
e = real(ifft(E));
```

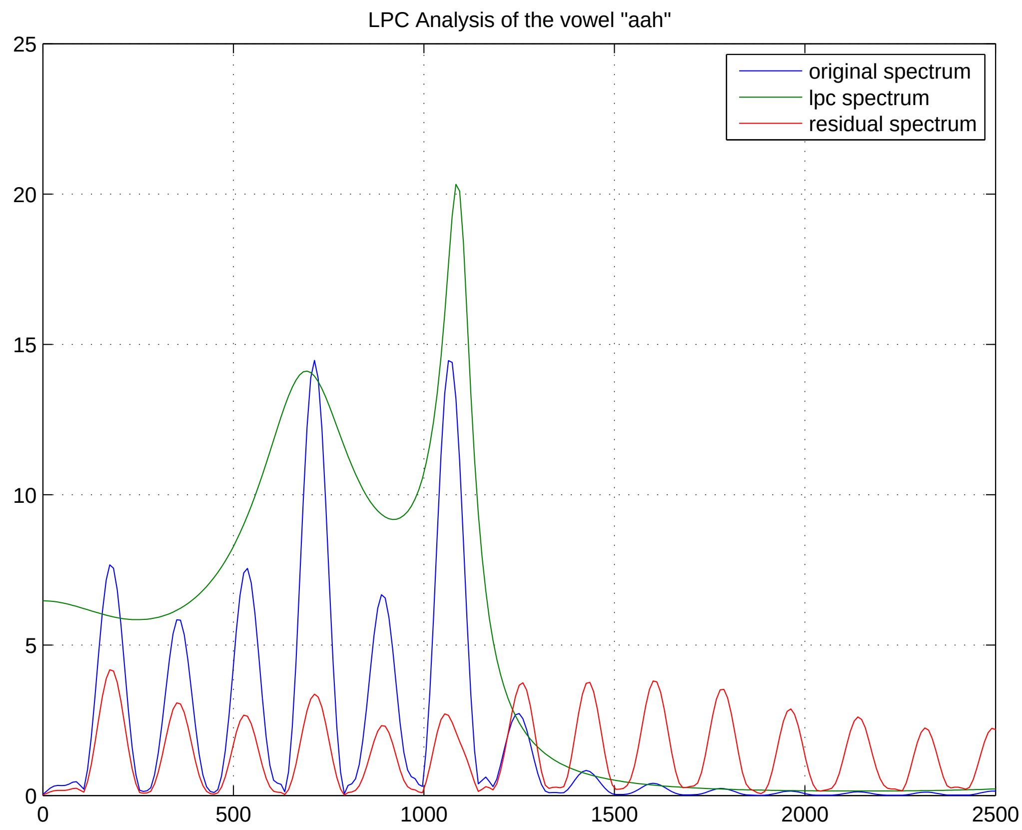

Figure 3: LPC Analysis Results for vowel ’a’.

Constant Overlap Add (COLA)

- Mathematical definition of the Short-time Fourier Transform (STFT) is given by

$$X_m(\omega) = \sum_{n=-\infty}^{\infty} x(n)w(n - mR)e^{-j\omega n},$$

where R is the hopsize, and m is the length of the window.

- The window used in the STFT, $w(n)$, must satisfy the Constant Overlap-Add (COLA) property:

$$\sum_{m=-\infty}^{\infty} w(n - mR) = 1.$$

- If COLA is satisfied, then the sum of successive DTFTs over time equals the DTFT of the whole signal $X(\omega)$, that is:

$$\begin{aligned}\sum_{m=-\infty}^{\infty} X_m(\omega) &\triangleq \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} x(n)w(n-mR)e^{-j\omega n} \\ &= \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}\sum_{m=-\infty}^{\infty} w(n-mR) \\ &= \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} = X(\omega), \quad \text{if COLA.}\end{aligned}$$

- Rectangle window is COLA if there is no overlap.
- Bartlett window, and all the Hamming family are COLA with 50 % overlap (when end points are handled correctly).

Matlab implementation

```
[x, fs, nbits] = wavread('...');
N = length(x);

Nwin = 256;        % window size
Noverlap = Nwin/2; % 50 percent overlap
zpf = 1;  Nfft = Nwin*zpf;
X = zeros(Nfft, round(N/Noverlap-1));
win = hanning(Nwin); %COLA window

for i=0:N/Noverlap-2
  ix = Noverlap*i+[1:Nwin];
  X(:,i+1) = fft(x(ix).*win, Nfft);
end

%Reconstruct
y = zeros(N,1);
for i=0:N/Noverlap-2
  ix = Noverlap*i+[1:Nwin];
  y(ix) = y(ix) + real(ifft(X(:,i+1)));
end
```