Assignment 12: Noise Source & Stop Formant Transitions

Reading:


section 7.4.2.1 (pp. 340-344), Labial Stop Transitions
section 7.4.3.1 (pp. 355-357), Alveolar Stop Transitions
section 7.4.4.1 (pp. 365-367), Velar Stop Transitions


Locus Equations:

LPC Re-synthesis with Voiceless segments

(1) Write a MATLAB function to calculate the dominant frequency based on zero-crossing rate of a sequence of analysis frames:

```matlab
function f = get_zcr (signal, srate, window_ms, slide_ms)
```

Output arguments:
- \( f \) (dominant frequency in Hz, based on zero-crossing rate; vector of length nframes)

Input arguments:
- `signal` input signal
- `srate` sampling rate in Hz
- `window_ms` duration of analysis window (milliseconds)
- `slide_ms` number of milliseconds between successive analyses (frames)

Write a loop that goes through the data in steps of `slide_ms`, and analyzes a window of `window_ms`. Try using the same values for these as you used in `get_lpc`. There is no point applying Hamming window. (why?)

To compute the frequency estimate using zero-crossing for a given frame, do the following. Find the number of times the signal crosses through zero in the window
(see example in zcr_demo). Call that Zn. That is the number of zero-crossings per frame. You can get the number of zero-crossings per second (Hz) by converting as follows:

\[
zcr = Zn \times \text{srate} / \text{samps_per_frame}
\]

Then, to get the estimate of dominant frequency (f) based on zero-crossing, divide zcr by 2.

(2) Record a sentence with several voiceless segments. Create a script that does the following: Analyze the sentence using get_lpc, get_f0 and get_zcr to derive parameters for analysis and re-synthesis. Run ftime_zcr to determine threshold value for voiced vs. voiceless segments of the signal. Re-synthesize using syn_lpc. Send me the script including all the steps.

```matlab
function signal = syn_lpc (srate,f0,frame_dur,A,G,zcr,thresh, AH_gain)
    % Louis Goldstein
    % synthesize a signal from LPC coefficients
    % switch between voiced and noise source if zero-crossing vector is input
    % input arguments:
    % srate     sampling rate (in Hz)
    % f0        fundamental frequency (in Hz)
    %           can be either a scalar (for flat f0)
    %           or a vector specifying f0 values for each frame
    % frame_dur duration of each frame in milliseconds
    % A         LPC A coefficients (M X nframes)
    % G         LPC G coefficients (nframes)
    % zcr       zero-crossing rate (Hz) vector (nframes)
    % thresh    zcr threshold for voicelessness (def. = 1000 Hz)
    % AH_gain   overall gain factor for voiceless source (def. = .05)
    % output argument:
    % signal    vector with synthesized waveform samples
    % fill in defaults
    if nargin < 7, thresh = 1000; end;
    if nargin < 8, AH_gain = .05; end;
```
Formant synthesis with noise sources

syn3.m

The next version of the synthesizer provides two additional capabilities: modulating the amplitude of the voiced source (amplitude of the pulses returned by make_impulse), and creating a noise source of variable amplitude. It is also possible to use a combination of voiced and noise sources. Two additional rows are now added to the Ftable, AV (Amplitude of Voicing) and AH (Amplitude of Hiss). The source can be a continuously weighed combination of the voiced source and the noise source, with AV and AH being the weighting factors. So now the Ftable has 3 required rows at the top, which are in order: AV, f0, and AH.

Source algorithm:

Voiced source. The voiced source should not change values of AV within a period (this would create an unnatural waveform). So AV needs to be set for each impulse. To do this, I added the AV vector as a fourth input argument to make_impulse (now called make_impulse_AV), and interpolated AV to the sampling rate as for f0. Then, when setting the value of the impulse vector to be non-zero at some sample, instead of setting it to 1 as before, I set it to the value of the interpolated AV at that sample. Finally, the resulting AV-scaled impulse train is filtered through the low-pass glottal resonator.

Noise source. I filled a vector equal in length to the voiced source vector with random Gaussian noise, which is a good approximation to a pressure noise source at the glottis. The noise must then be low-pass filtered to approximate a velocity sources (which is the basis for the transfer function). I filtered with a feedforward filter with b coefficients = [.5 -5]. Then the noise source has to be scaled by the appropriate value of AH. To do so, I interpolated the set of AH frame specifications to the signal sampling rate (as for f0 and AV). The interpolated AH vector is then multiplied by the filtered noise vector.

Combining sources. The next step is to add the voiced and noise sources. However, we also need gain factors, so that an input value of 1 (or 100) for either AV or AH will result in a signal of comparable amplitude. I am using AH_gain = .05 and AV_gain = 100, but they are not necessarily optimal. (The noise gain must be lower, since the noise source has non-zero values every sample, while the voiced source has only one non-zero value per fundamental period). The voiced source is multiplied by AV_gain and the noise source by AH_gain and then they are added. The combined source is used as input to the vocal tract formant filter, which works as before.
Exercise: modeling VCV with stop C

(1) Record and save the following nine utterances:
   aba  ada  aga
   ɛbe  ede  ɛde
   ubu  udu  ugu

(2) For /aba/, analyze formants using get_lpc, and f0 using get_f0, then examine the
    formant time functions using ftime2. This is a revised version of ftime that differs
    in the order of arguments, and in having a larger window for the formant display to
    make it easier to see formant transitions.

function ftime2 (signal,sr, A, G, f0, minG,BegFr, EndFr)
   % Louis Goldstein
   % November  2009
   % plot time functions of signal, gain, f0, F1-F4
   % Click on point in time for transfer function, formant values
   % Hit return to end
   %
   % Input arguments
   % signal            vector signal samples
   % sr                sampling rate in Hz
   % A                 matrix of lpc A coefficients
   % G                 vector of lpc Gains
   % f0                vector of f0 values
   % minG              Gain cutoff for formant plot (def. = 10^-8)
   % BegFr             beginning frame for plot (def. = 1)
   % EndFr             ending frame for plot (def. = last frame)

(3) Based on the formant and gain values obtained from ftime2, create parameter files
    for syn3 for each /aba/. The strategy is illustrated below. There is no need to include
    f0 variation (you can choose a value close to your mean f0, as observed in ftime2). If
    the values for the end of the VC transition and beginning of the CV transition do not
    seem to be working (it doesn’t sound like [aba]), choose values based on theoretical
    considerations (as in the slides).

(4) For /ada/ and /aga/, try to create convincing parameter files by starting with /aba/ as a
    template, and changing only the the values for the end of the VC transition and
    beginning of the CV transition based on theoretical considerations. Examine your
    recorded utterances as an additional guide for those values/.

(5) Repeat the same procedure for the other two vowel contexts. For the /u/ context,
    use /udu/ as your starting template. Note: for /ɛdɛ/, you may not get a good coronal
    stop percept. We will see why next week.

(6) For one of the 3 vowel contexts, produce voiceless aspirated versions of the VCVs.
    (Again, see strategy below).
Synthesis strategy:

[aba] -- Louis

>> ftime2 (aba, sr, A, G, f0, .5*10^-7);

I have added to the ftime2 display vertical lines indicating the frames that represent the beginning and end of the VC and CV transitions. Try to make sure that you don’t have any extra or missing formants in the region of the transitions. If you do, then re-run get_lpc with different values of M. Here I set M=44, which is fewer than the default (by 4).

1. Find values for the formants and bandwidths during the ‘steady-state’ vowel regions. You should be able to use a single set of constant values for the entire utterance, and in fact should be able to use the same values for all three VCVs with the same V. Set up a input file with just those values at frame 1 and frame N (which should be set equal to the number of frames you will need for the complete VCV). Try synthesizing with that, and make sure it sounds OK. The bandwidths can be rounded greatly, or you can even use values like those below:
2. Now find the frames where amplitude of voicing needs to be decreased for the stop consonant. Interpolate from 100 to a low value (0 or 10) over two frames. Then synthesize using that:

```
aba2.txt
```

3. Now find the frames where the VC transition starts, and interpolate the values of F1-F3 from the vowel values there to the values found at the end of the transition. Do the parallel interpolation at for the CV transition. Synthesize again.

```
aba3.txt
```
4. To generate an aspirated version, set the value of AH to be high (e.g. 50) during the interval of the CV transitions, and set the value of AV to be 0. You want relatively abrupt transitions of these values, so for example, you should specify a value of AH=0 two frames before the beginning of the CV transition, and the value AH=50 at the beginning of the transition. See bold values in apa.txt. It also improves naturalness if B1 (the bandwidth of the first formant) is widened to about 500 Hz during voiceless intervals.
apa.txt

F 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
AV 1 100 20 100 22 0 33 0 38 0 42 100 60 100
F0 1 100 60 100
AH 1 0 33 0 35 50 38 50 40 0 60 0
F1 1 650 14 650 21 550 35 275 40 650 60 650
F2 1 1350 14 1350 21 1140 35 875 40 1350 60 1350
F3 1 2450 14 2450 21 2300 35 2500 40 2450 60 2450
F4 1 3100 60 3100
F5 1 4650 60 4650
B1 1 100 33 100 35 500 40 500 42 100 60 100
B2 1 150 60 150
B3 1 200 60 200
B4 1 200 60 200
B5 1 400 60 400

>> sig = syn3(10000,10,5,'apa.txt');