New Mexico State University
Klipsch School of Electrical Engineering

EE589 Digital Speech Processing
Spring 2011 – Project #4
Due: 8.55am Thu. Mar. 17

Name: _____

Grade: _____

Project

The goal of this project is to construct a voice coder or "vocoder" based on the Linear Predictive Analysis (LPA) for the purpose of compressing speech signals. The project consists of an encoder and decoder. The encoder is constructed from three components: the clipping autocorrelation pitch detector (Figure 4.36 of Rabiner and Schafer), the linear predictive analyzer (Chapter 5 of Quatieri), and the data formatter. The decoder is also constructed from three components: the data extracter, the excitation signal generator, and the spectral shaper.

Directions

This document will guide the student through several programming steps in constructing the vocoder. Please submit the cover sheet of this project and an email to `pdeleon@nmsu.edu` with a .zip file attachment. The .zip file should contain an archive which has two directories: "program" containing all program codes (step1.m, encode.m, decode.m, and support files) necessary for completing steps 1 – 3 and "data" containing several data files. These data files include:

1. plots and commentary from Step 1
2. encoded_speech.bin (output of your encoder with s1.wav as the input) from Step 2
3. decoded_speech.wav (output of your decoder with the encoded speech data file as the input) from Step 3

The input speech signal and code for PESQ measurements can be found at

`http://www.ece.nmsu.edu/~pdeleon/Teaching/EE589/Related_Links.html`

Evaluation

Evaluation of the project will consist of two measurements.

1. Compression factor

$$C = \frac{\text{size(s1.bin)}}{\text{size(encoded\_speech.bin)}}$$

   where s1.bin is a file of 16-bit samples from s1.wav. The compression factor will be quantized as shown.

2. Perceptual Evaluation of Speech Quality (PESQ). Using the supplied code, a PESQ score will be computed.

The project grade will then be

$$S = 0.13(\hat{C} + 2 * \text{PESQ})$$

C (compression factor)	$\hat{C}$ (point value)
$1.5 < C \leq 2$	1
$2 < C \leq 5$	2
$5 < C \leq 10$	3
$10 < C \leq 25$	4
$25 < C$	5

subject to a maximum of 100%. Note that you can trade off high compression ratios for (presumably) a lower PESQ or vice versa. One scoring example would be $\hat{C} = 4$ and PESQ $= 1.75$ which leads to $S = 98\%$.

It is assumed your code will execute in a *reasonable* amount of time on a standard PC. It would be unreasonable to require more than 10 minutes total to encode and decode 1 minute of speech. If your project is only partially working, then a grade will be assigned based on progress through the various programming steps. A partially working project may also require a separate demonstration by the student. Finally, Prof. De Leon reserves the right to increase (curve) project grades.

Step 1: Center Clipper

Create a 3-level center clipper function as described in Rabiner and Schafer by coding the

`center_clipper.m` tool in the workbook.

(a) With $C_L = \sqrt{2}/2$, apply the center clipper to one cycle of a sinusoid (100 samples)

$$x[n] \quad = \quad \sin(2\pi n/100). \tag{1}$$

Plot your result, save the plot as a .eps/.pdf file, and copy to the data directory.

(b) Plot the autocorrelation function of your /a/ phoneme and estimate the pitch period and associated fundamental frequency. Plot the periodogram of your /a/ phoneme and estimate the fundamental frequency and associated pitch period. Save the plots to the data directory.

(c) With an appropriate C_L, apply the center clipper to your /a/ phoneme. Plot the autocorrelation function of the center clipped /a/. The autocorrelation function should have periodic peaks and resemble Fig. 4.34 in Rabiner and Schafer. Estimate the pitch period and associated fundamental frequency and comment on the results as compared to the results in (b). Save the plot to the data directory. Good operation here should enable good decisions regarding voiced sections.

(d) With the C_L used in (b), apply the center clipper to your /f/ phoneme. Plot the autocorrelation of the center clipped /f/. The autocorrelation function should not have any periodic peaks. Save the plot to the data directory. Good operation here should enable good decisions regarding unvoiced sections.

(e) Comment (1/2 - 1 page) on your experimental work, procedures, and results from (a) - (d).

Step 2: ENCODE.M Outline

Program the encoder portion of main.m using the block diagrams of the LPC encoder and clipping autocorrelation pitch detector as illustrated in Figs. **??**–**??**. Following the illustrations, is a suggested outline of the encode portion of the main code.

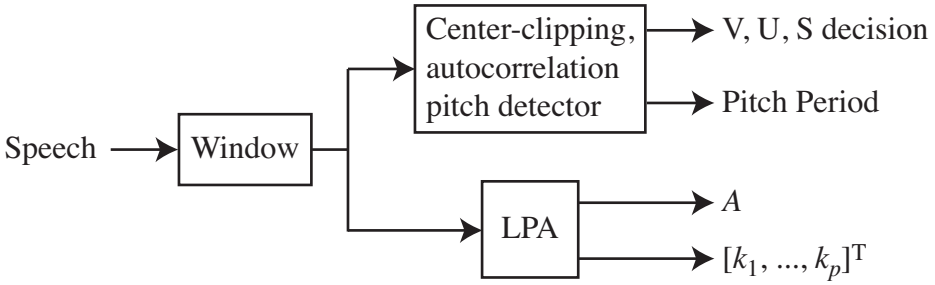

Figure 1: LPC encoder

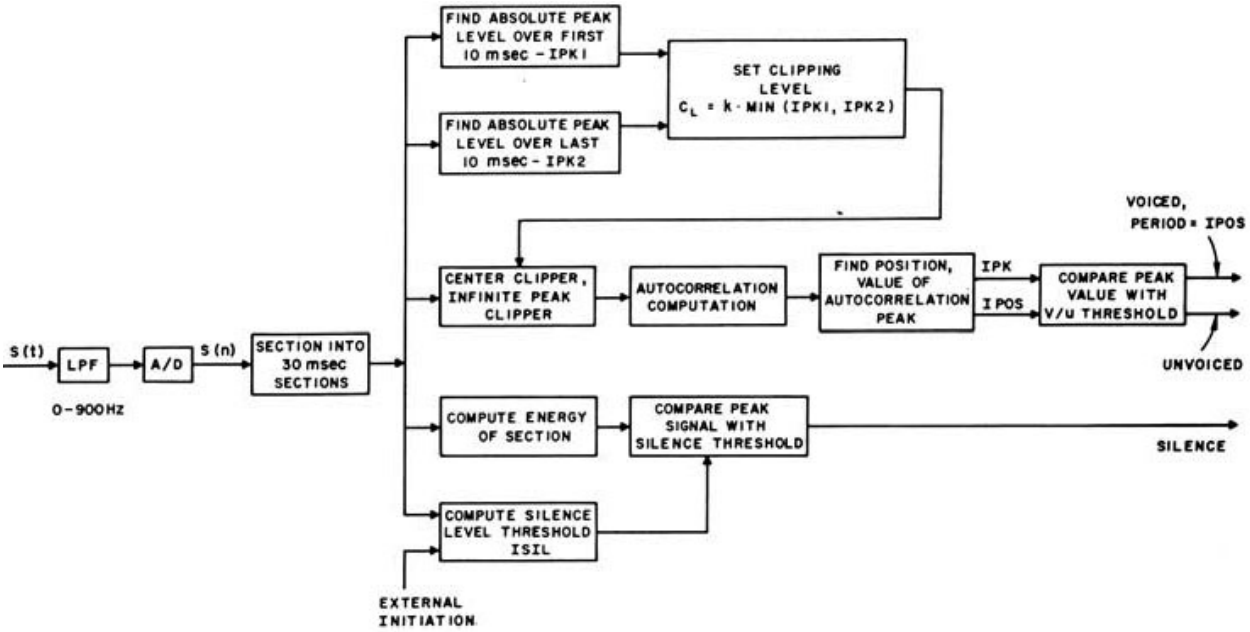

Figure 2: Center clipping, autocorrelation pitch detector

```
%------------
% LPC Encoder
%------------

% User parameters--these are only a few and only suggested values.
[x,fs,bits] = wavread('speech.wav');  % load speech file
L = 0.030*fs; % 30ms window/frame size
overlap = 0.015*fs; % 15ms window overlap
```

```
k = 0.50; % factor to set clipping level, 0.50
VU_Thresh = 0.60; % Voiced/Unvoiced threshold, 0.60
N = 10; % AR model order for LPC coefficients

% Initialize
(compute or estimate average background noise level, ISIL over initial frames of signal;
other initializations as required)
start = 1; stop = L;  % initial frame indices

% Encoder
for m=1:num_frames  % loop over the frames
   frame = x(start:stop);  % get the frame

   % --- 1) Clipping autocorrelation pitch detector ---
   (compute peak or average signal level of frame, peak_x)
   if (peak_x < ISIL)
      (decide silence)
   else
      (determine clipping level, c_L)
      (apply center clipper to frame)
      (compute autocorrelations, r of center clipped frame)
      (get value, position of first autocorrelation peak IPK, IPOS)
      if (IPK >= VU_Thresh)
         (decide voiced and pitch period)
      else
         (decide unvoiced)
      end;
   end;

   % --- 2) Linear predictive analysis ---
   (get A and PARCORs of the frame from Levinson)

   % --- 3) Data formatter ---
   (data record includes: 1) voiced, unvoiced, or silence parameter, 2) pitch period (if
   applicable), 3) model parameters (if applicable). Use fwrite to write data as int8 or int16)

   start = start+(L-overlap); stop = start + L -1;  % next frame indices
end; % loop over the frames
(arrange data)
(save file)
```

The speech parameters in the data record should be quantized and written to a binary file.

```
fwrite(fid,pitch_period,'int8');
PARCOR_int16 = floor((2^15)*PARCOR); % -1 < x < +1 to -2^(B-1) < x_int < +2^(B-1)
fwrite(fid, PARCOR_int16,'int16');
fwrite(fid,A,'float32');
```

If the frame is silence, there is no need to encode pitch period and vocal tract parameters thus allowing for higher compression factors.

Step 3: DECODE.M Outline

Program the decoder portion of main.m using the block diagram of the LPC decoder as illustrated in Fig. **??**. Following the illustration, is a suggested outline of the decode portion of the main code.

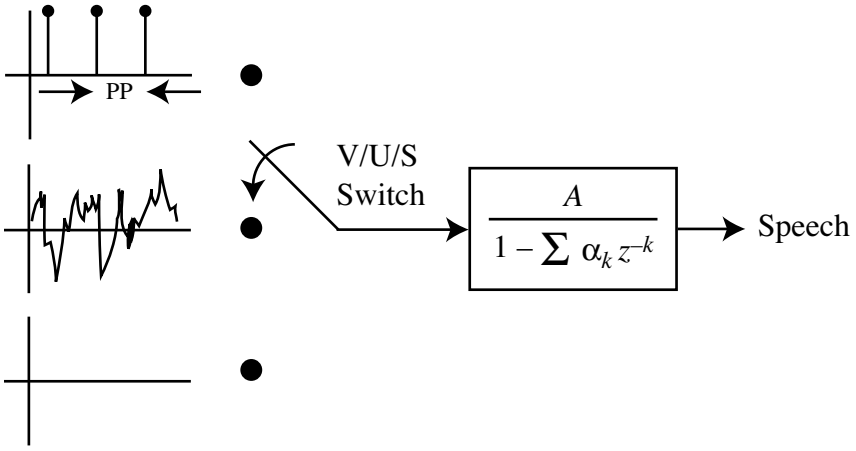

Figure 1: LPC decoder

```
% User parameters--these are only a few and only suggested values.
(load encoded speech file)

% Initialize

% Decoder
for m=1:num_frames  % loop over the data vectors

   % --- 1) Data extractor ---
   (extract parameters from data record)

   % --- 2) Generate excitation signal ---
   if (frame is voiced)
      (create impulse train with period = pitch period)
   elseif (frame is unvoiced)
      (create random noise U[-1,1])
   else
      (create silence)
   end;

   % --- 3) Spectrally shape excitation signal ---
   (filter impulse train or noise excitation using parameters from LPA)
   (overlap synthesized frames and add)

end; % loop over the data vectors

(remove DC and amplitude normalize)
(save .WAV file)
```

Hints

Here are a few hints which may improve compression and/or quality.

- Since there will be many encode, decode, listen/PESQ cycles, you may want to work with only a few seconds of the speech signal in order to speed the design process.
- Plotting the estimated pitch periods can facilitate evaluation of the accuracy and the setting of the associated parameters for the voiced decision and pitch period estimation.
- Occasionally pitch period estimations can be very wrong. Maintaining an average pitch period can assist in evaluating accuracy of pitch period estimation.
- Reflection coefficients, k_i and filter gain, A may be obtained with either

  ```
  [alpha,e] = lpc(x,p);
  k = tf2latc(1,alpha);
  A = sqrt(e);
  ```

  or

  ```
  r = xcorr(x,'biased');
  M=length(x);
  [alpha,e,k] = levinson(r(M:end),p);
  A = e;
  ```

  Students should verify that results with both methods are equivalent.
- Lattice filtering using reflection coefficients and gain is accomplished with

  ```
  [s,g] = latcfilt(k,A,u);
  ```

  or

  ```
  [b,a] = latc2tf(k,'allpole');
  s = filter(b*A,a,u);
  ```
- A simple LPF can be used to remove musical artifacts and other noise from the decoded signal

  ```
  h = fir1(31,0.75);
  x_hat = filter(h,1,s);
  ```

  however, there is no substitute for having a well-designed codec.

Bonus

(+10%) Using different compression factors, submit three encoded speech files with PESQ approximately 1.0, 1.5, and 2.0.