Lab 5 – Linear Predictive Coding

Idea

When plain speech audio is recorded and needs to be transmitted over a channel with limited bandwidth it is often necessary to either compress or encode the audio data to meet the bandwidth specs. In this lab you will look at how Linear Predictive Coding works and how it can be used to compress speech audio.

Objectives

- Speech encoding
- Speech synthesis

Read the LPC.pdf document for background information on Linear Predictive Coding

Procedure

There are two major steps involved in this lab. In real life the first step would represent the side where the audio is recorded and encoded for transmission. The second step would represent the receiving side where that data is used to regenerate the audio through synthesis. In this lab you will be doing both the encoding/transmitting and receiving/synthesis parts. When you are done, you should have at least 4 separate MATLAB files. Two of these, the main file and the levinson function, are given to you. The LPC coding function and the Synthesis function are the files that you have to write. Below is an overview list of steps for this lab.

1. Read the *.wav file into MATLAB and pass the data to the LPC coding function
2. The LPC coding function will do the following:
    a. Divide the data into 30mSec frames
    b. For every frame, find the data necessary to reproduce the audio (voiced/unvoiced, gain, pitch, filter coefficients)
    c. The LPC coding function will return these data vectors
3. Pass the data from the LPC coding function to the Synthesis function
4. The Synthesis function will do the following:
    a. Regenerate each frame from the given data
    b. Reconnect all the frames
    c. The Synthesis function will return the reconstructed audio
5. Play the original audio
6. Play the synthesized audio

The above list of steps is already coded in the main MATLAB file. The parts that you need to do are the functions for step 2 and step 4. Please look at all the given code and instructions carefully before starting this lab.

The LPC coding function

As mentioned above the LPC coding function will take the speech audio signal and divide it info 30mSec frames. These frames start every 20mSec. Thus each frame overlaps with the previous and next frame. Shown in the figure below:

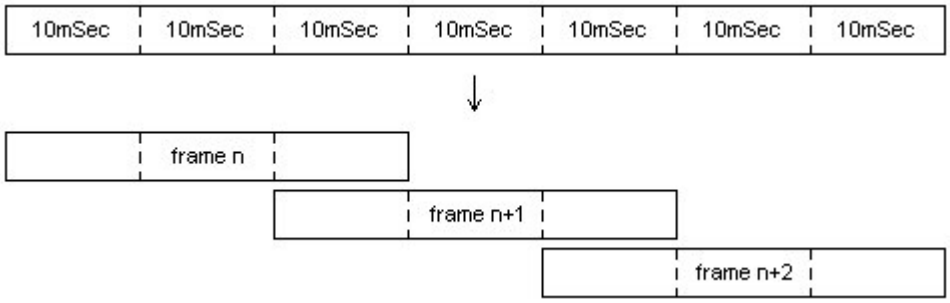

Figure 1: Audio signal to separate frames

After the frames have been separated, the LPC function will take every frame and extract the necessary information from it. This is the voiced/unvoiced, gain, pitch, and filter coefficients information.

To determine if the frame is voiced or unvoiced you need to find out if the frame has a dominant frequency. If it does, the frame is voiced. If there is no dominant frequency the frame is unvoiced. If the frame is voiced you can find the pitch. The pitch of an unvoiced frame is simply 0. The pitch of a voiced frame is in fact the dominant frequency in that frame. One way of finding the pitch is to cross correlate the frame. This will strengthen the dominant frequency components and cancel out most of the weaker ones. If the 2 biggest data point magnitudes are within a 100 times of each other, it means that there is some repetition and the distance between these two data points is the pitch.

The gain and the filter coefficients are found using Levinson's method. This code is already written for you. After downloading this function to your working directory you can do a "help levinson" to find out how to use the function.

After finding these variables for all the frames the function will pass them back to the main file as seen below:

[Coeff, pitch, G] = proclpc(inspeech, Fs, Order);

- Coeff is a matrix of size (number of coefficients x number of frames) containing the filter coefficients to all the frames
- Pitch is a vector of size (number of frames) containing the pitch information to all the frames
- G is a vector of size (number of frames) containing the gain information to all the frames
- Inspeech is the input data
- Fs is the sample frequency of the input data
- Order is the order of the approximation filter

The Synthesis function

The synthesis part is fairly easy compared to the coding function. Fist for each frame you need to create an initial signal to run through the filter. This initial signal is also of length 30mSec. Using the information from the variable passed into the synthesis function you will be able to synthesize each frame. After you have synthesized the frames you can put them together to form the synthesized speech signal.

The initial 30mSec signal in created based on the pitch information. Remember that if the pitch is zero, the frame is unvoiced. This means the 30mSec signal needs to be composed with white noise. (look at the MATLAB function randn) If the pitch is not zero, you need to create a 30mSec signal with pulses at the pitch frequency. (look at the MATLAB function pulstran)

Now that you have the initial signals all you have left to do is to filter them using the gain and filter coefficients and then connect them together. (look at the MATLAB function filter)

Putting the frames back together is also done in a special way. This is where the reason for having the frames overlap becomes clear. Because each frame has its own pitch, gain and filter, if we simply put them text to each other after synthesizing them all, it would sound very choppy. By making them overlap, you can smooth the transition from one frame to the next. The figure below shows how the frames are connected. The amplitude of the tip and tail of each frame's data is scaled and then simply added.

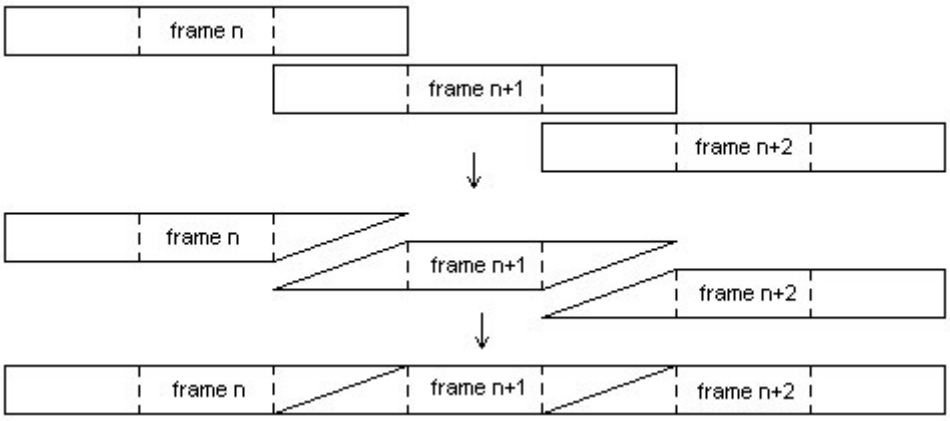

Figure 2: Addition of separate frames into final audio signal

Interpreting your results

There are several ways to look at the results but for the sake of simplicity we will rely on the human ear. In the main function the input speech audio and the synthesized speech audio is played back to back. You can listen to these playbacks and judge accordingly if your coding and synthesis functions are working.

Post Lab

Write a paragraph summarizing what you have learned in this Lab. Also include a copy of all MATLAB code written and a copy of all the plots and images created. Turn in your lab results to your Lab TA at the beginning Lab in three weeks.