3.1)._ Data Acquisition

Question 1. What is the speech segment that you have chosen? Include the time plot with your attempt at labeling the phonemes.

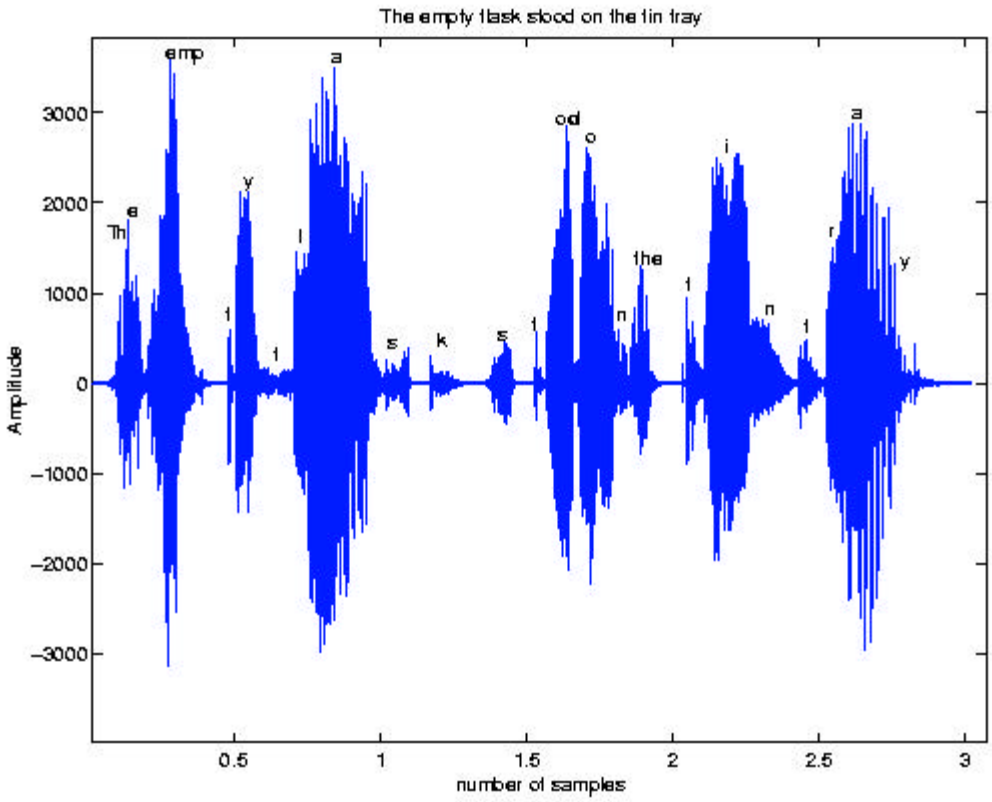

Figure 1. The speech signal contained in the file CW161 and an attempt to locate the different phonemes present in the speech signal.

In figure 1, we can see the plot of the speech signal contained in the CW161 file. The speech signal is the one corresponding to the text 'The empty flask stood on the tin tray' with a duration of 3 seconds. Figure 1 also shows the my attempt to locate the different phonemes present in the speech signal.

3.2)._ Pitch Detection

1a._ First, I extracted a 100mSec segment of data containing a vowel from the CW161 file. The vowel segment that I selected was the one corresponding to the 'e' in the word 'empty'. Figure 2 shows the plot of the extracted vowel.

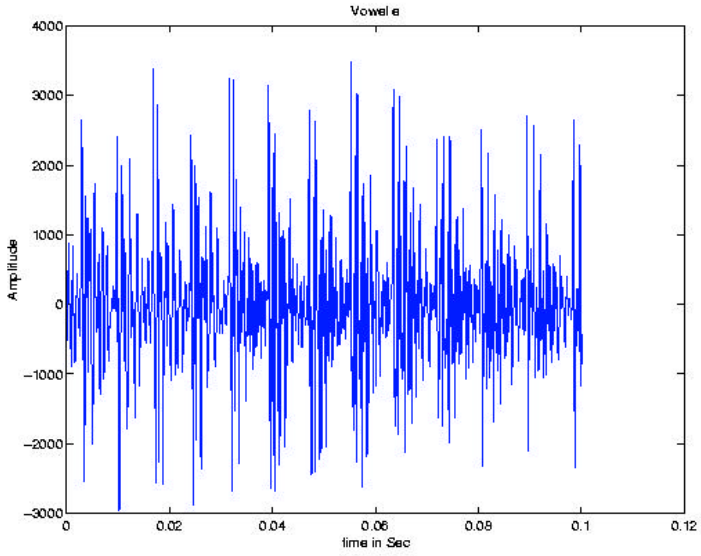

Figure 2. Vowel 'e' in the word 'empty' extracted from the CW161 file.

1b._Then, since the final objective of this section is to develop a function to detect the pitch of the signal, I filtered the signal of the vowel with a low pass filter with a cutoff frequency of 500Hz. I chose this cutoff frequency because the pitch in a man goes from 80 to 160Hz, and the pitch in a woman from 160 to 320 Hz. Thus, filtering at 500 Hz should be enough. Figure 3 shows the plot of the filtered signal.

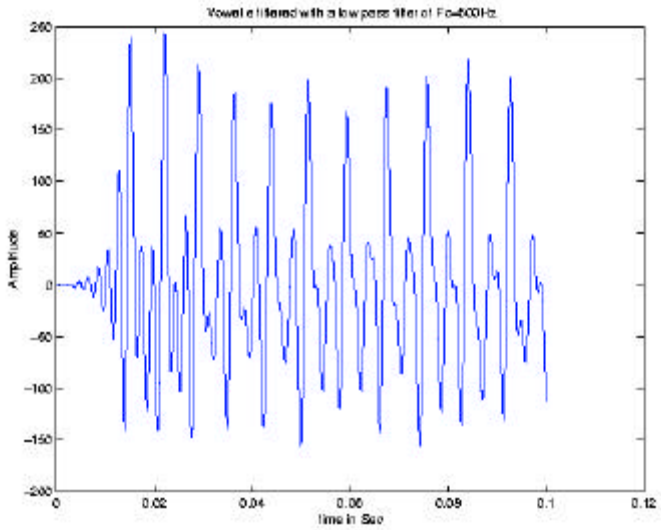

Figure 3. Vowel 'e' filtered with a low pass filter with a cutoff frequency of 500Hz.

2._Later, I calculated the autocorrelation function of the vowel. I did the necessary modifications to make the maximum of the autocorrelation function be at t=0, since this is always the case for autocorrelation functions. I also changed the time axis to show the autocorrelation function from -30 to 30 mSec. The plot of the autocorrelation function of the vowel is shown in figure 4. selected the -30 to 30mSec interval because it allow me to manually calculate pitch values as low as 33.33Hz (t=30mSec)

and as high as 500Hz or more (t<=2mSec). I calculated the pitch of the vowel manually. The pitch is 222.22Hz.

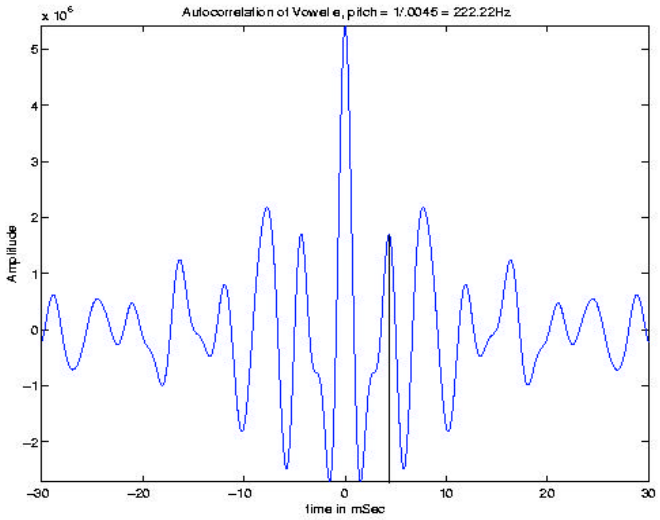

Figure 4 Autocorrelation function of the vowel 'e'

3._In order to eliminate the peaks due to the vocal tract transfer function from the autocorrelation function, the signal is often signal is first center clipped. The clipping limits often used are 75% of the minimum and maximum values of the speech segment. This are the clipping limits I used. In Figure 5, we can see the plot of the center clipped vowel. Figure 6 shows the plot of the autocorrelation function of the signal center clipped. Before center clipping the signal, I eliminated the DC value of the vowel signal. I also calculated the pitch manually for this center clipped vowel. The pitch is 147Hz.

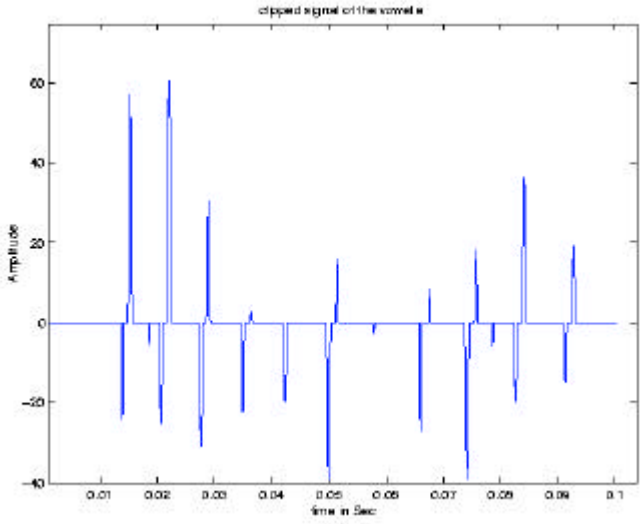

Figure 5. Plot of the vowel 'e' center clipped

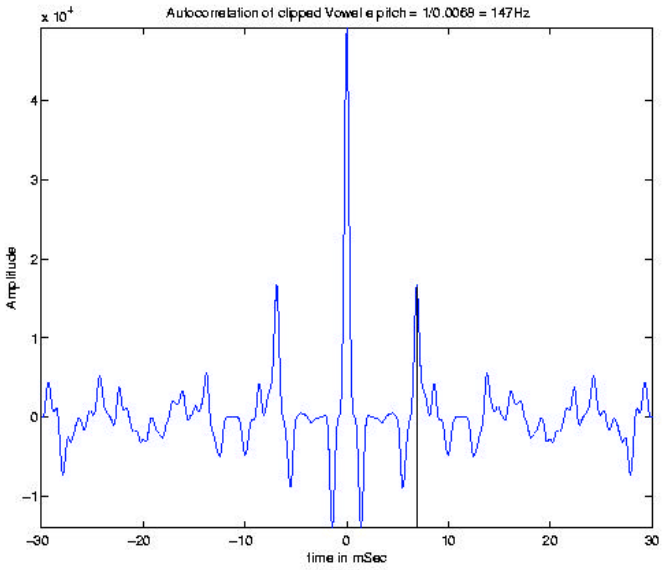

Figure 6. Autocorrelation function of the center clipped vowel signal

4._ I developed a function in matlab to automatically detect the pitch. The function takes the following parameters:

[pitch] = pitch_detect(x)

where x is a vector containing the frame of filtered speech signal for which we want to calculate the pitch. The function returns pitch, a scalar containing the pitch of the frame in Hz or 0 if unvoiced

The steps followed by the pitch_detect function in order to calculate the pitch are the following:

- First, remove the dc value of the frame by subtracting the mean
- Then find the minimum and maximum samples and center clip to 75% of those values
- Compute the autocorrelation of the frame
- Normalize the maximum of the autocorrelation to be one.
- extracting the positive (t = positive) part of the correlation
- Find the maximum peak following Rx[0]
- Calculating what percentage of the maximum at zero represents the founded maximum. This is, the ratio of the autocorrelation function at the peak pitch lag to the autocorrelation function lag=0
- Determine if the segment is unvoiced based on the 'voicing strength'. If voicing strength is less than 0.25, call it unvoiced and set pitch = 0, otherwise compute the pitch from the index of the peak

the code of the pitch_detector can be found in Section I of the appendix.

5._I applied the automatic pitch_detector that I coded to the filtered vowel 'e' of Figure 3. The calculated pitch for this speech frame was

vowelpitch = 142.8571Hz

Question 2. Describe how you determined the pitch manually. Include a plot of one of autocorrelation sequence from either step2 or step3. How does the pitch value determined by the automated pitch detector compare to the values estimated manually in steps 2 and 3? Based on the manual and automatic estimates, what do you think is the pitch of the vowel segment? Is this value reasonable, given what you know about speech?

In order to calculate the pitch manually in steps 2 and 3, I just searched the plot of the autocorrelation functions looking for the second highest (maximum) peak after Rx[0] then, I just took its time value and calculated the frequency inverting this value (F=1/T).

The manually calculated pitch value for the filtered vowel was 222.22Hz for step 2 and 147Hz for step 3. The output of the automatic pitch detector for the vowel segment was 142.8571Hz. I think that the pitch calculated for step 2 is not completely correct because it contains the peaks due to the vocal tract transfer function. The pitch calculated in step 3 is more accurate, but still not completely correct because I calculated the frequency looking at the time values using my naked eye. Finally, I think that the better approcimation to the pitch of the vowel is the one calculated using the automatic pitch detector function and thus, 142.8571Hz.

I think that this pitch value is reasonable, since the person who is speaking in the speech signal from which the vowel was extracted was a man and we know that the pitch range of a man goes from 80 to 160Hz. The pitch founded falls in these range.

3.3)._Channel Vocoder

1._I wrote a matlab function to generate a bandpass filter bank of 18 bands each with a 200Hz bandwidth, centered at evenly spaced frequencies from 100Hz to 3500Hz.

The function is called myownfilter_bank and takes the following parameters.

bank=filt_bank(N,L)

where N is the number of bands that we want, L is the length or order of the filter. The function returns a LxN matrix called bank. Each of the N columns contains an L-point FIR filter. The Sampling frequency and the Bandwidth are hardcoded within the function. By default, Fs=8000Hz and B=200;

The code for myownfilter_bank.m can be found in section II of the appendix. The steps followed for generating the filter bank are the same that we used in the first part of problem set 2 and are shown here.

- Design a prototype lowpass filter with Fc=200Hz and order L-1 using a 65 point kaiser window with beta =3.
- Make a n vector equal to the filter length for argument to cosine. The low pass filter samples are separated by 1/Fs seconds. We want to multiply it by a cosine of the same length whose samples are also separated by 1/Fs, then we have n = ([0:L-1]/Fs)';

✍In loop, design filter for each band:

i._Compute desired center frequency cf = i*B-start;

ii._Shift low pass prototype to center frequency. If it is the first band of the filter bank, then keep the original low pass filter designed lpf. Otherwise, shift the original lpf by the cosine using lpf .* cos(2*pi*cf*n)*2;

Figure 7 shows the fundamental frequency response of the filter bank. Figure 8 shows the composite frequency response of the filter bank for an order of 65 and Figure 9 shows the composite frequency response of the filter bank of an order of 1000.

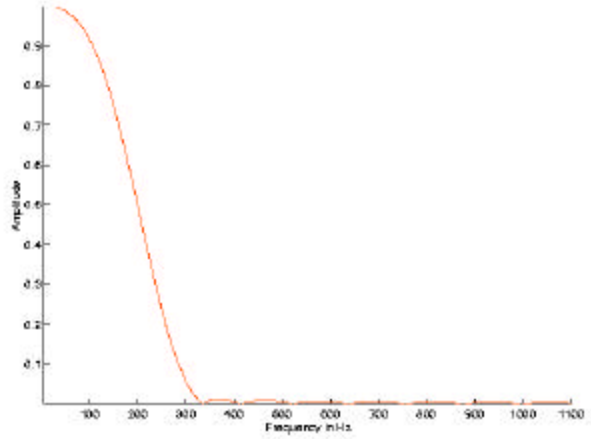

Figure 7. Fundamental frequency response of the filter bank

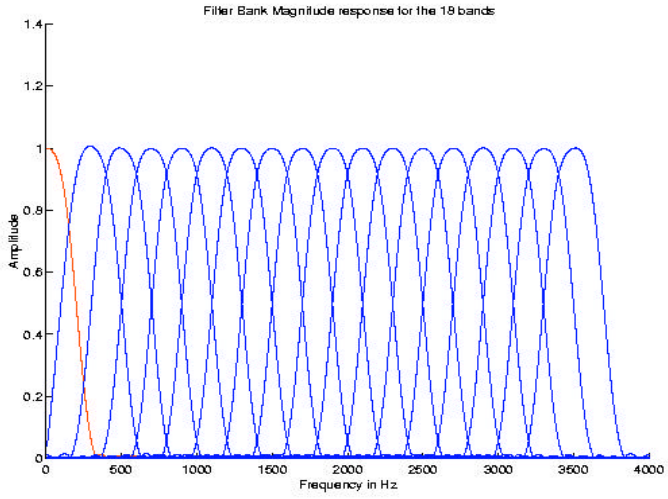

Figure 8. Composite frequency response of the filter bank fo order=65

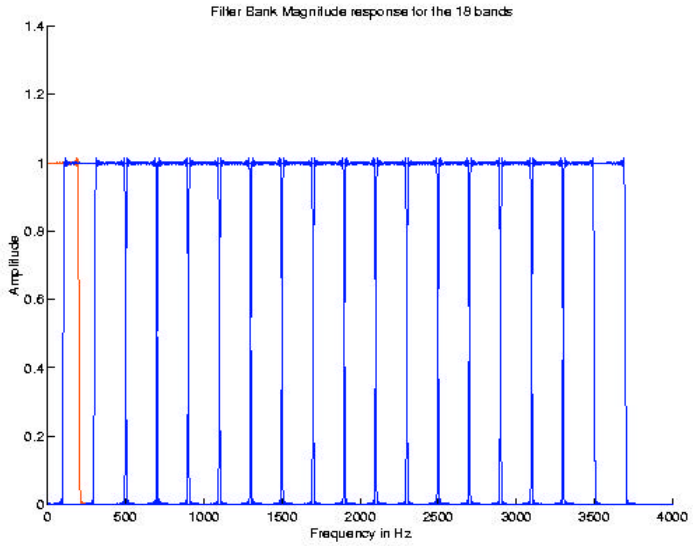

Figure 9. Composite frequency response of the filter bank fo order=1000

Question 3. Select a few representative frequency bands in the filter bank and plot their frequency responses. Also, plot the composite frequency response of the filter bank. Is the composite frequency response what you expected?

The plots are already shown in figures 7, 8 and 9. The only unexpected thing is that little overlapping present in the first two bands of our filter bank. I tried to eliminate this working together with Leonardo, but it seems that it is not possible to eliminate this overlapping and still be consistent with the approach followed in the first part of problem set 2 for the filter bank design.

2._I created the encoder portion of the Channel vocoder. The function name is chvocod_ana.m and take the following parameters.

[y,p]=chvocod_ana(x, D, N)

where x is a vector containing the speech signal to be encoded, D is the decimation rate to be used and N is the number of bands for our filter bank. Y is an output matrix containing the band envelope signals. Each column contains output of one band and each row contains output of one frame. p is an output vector containing pitch signal with one
pitch value per frame

The code for the chvocod_ana.m function can be found on section III of the appendix.
The steps followed in the development of this function are the following ones:

- Design a lowpass filter with 500Hz cutoff frequency
- Apply the filter to the speech signal and store it in xlpf
- loop. Each iteration processes one frame of data:
    i._Get next 30mSec segment of data from the filtered sinal xlpf and call it seg
    ii._Call the pitch detector function using seg as paramater
    iii._ store the result

iv._avance amount corresponding to the decimation rate to next frame

✍apply a median filter of order=3 to the pitch signal p in order to eliminate spurious values
✍Design a filter bank of N bands and order=65.
✍In loop, process each band:
i._Apply filter for this band
ii._Take magnitude of signal and decimate. (using abs)
iii._decimate and low pass filter the signal for this band and store the result.
at this point, each row in Y has a length of D (number of points)resulting from the decimation. Since we also have D number of segments, Y is a matrix in which each column represents a segment of the signal and each row represents a band in the filter bank

3._I processed the utterance and examined the outputs.

4._Initially, I tested the encoder using a decimation rate of 100. This was the result that I saved using the submit_lab2 function. It sounded pretty good as you are going confirm.

5._Testing my Channel encoder with different decimation rates, I found out that the higher the decimation rate I use, the worse the encoded signal sounds. On the other hand, the smaller the decimation rate is, the better the encoded signal sounds. I tested this for decimation values in the range of 10 to 1000. I also realized that the higher the decimation rate is, the longer it takes to calculate the encoded signal. The opposite is also true.

6._I produced monotone speech from the synthesizer by replacing the pitch signal calculated by the chvocod_ana.m function with a vector of constant values. The constant value that I selected was 100Hz. When I listened to the output, the result was unexpected for me. It sounds like a Robot speaking. Like a computer speaking in the early Robot movies of Hollywood.

7._I produced whispered speech by replacing the pitch signal with a vector of zeros. I was amazed by the results when listening to the output of this signal. It really sounds like a whisper.

8._I changed the male voice of the CW161 file to a female voice by scaling the pitch values by a factor of two. When I listened to this signal, It sound like a female voice, but it lacks the correct intonation. Even though the pitch values are the ones for females, I think that this signal does not sound that good because women have a different pronunciation than men (At least in my point of view).

Question 4. Plot the pitch values produced for the entire utterance. How well does your automated pitch detector perform? (Note that it is not necessary for your pitch detector to work perfectly), so at long as it produces reasonable pitch values for most frame utterances.

Figure 10 shows the plot of the pitch values used by the channel vocoder encoder that I designed. We can see a high peak located at t=1.65 seconds. This peak is eliminated applying the median filter of order 3. Figure 11 Shows a comparison between the pitch values used by channel vocoder encoder and the original signal. We can have an idea of the performance of my pitch detector by inspecting this plot. We can see that is working and that it only commit one mistake. We can also see that the vowels are always

detected by the pitch detector. Remember that the vowels are the high amplitude peaks present in the speech signal.

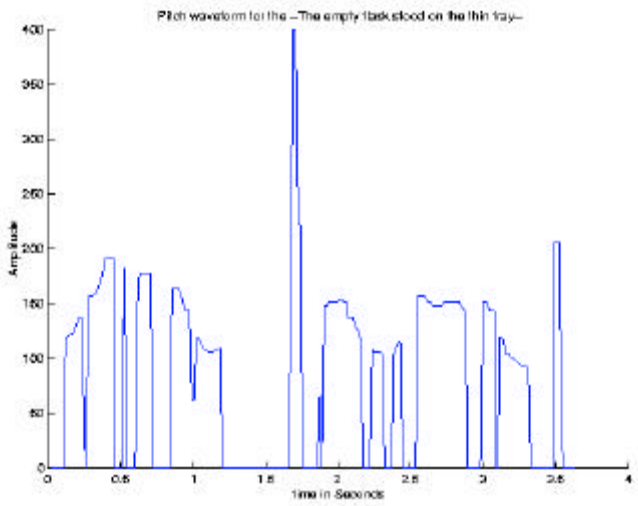

Figure 10. Pitch Values produced for the CW161 speech file

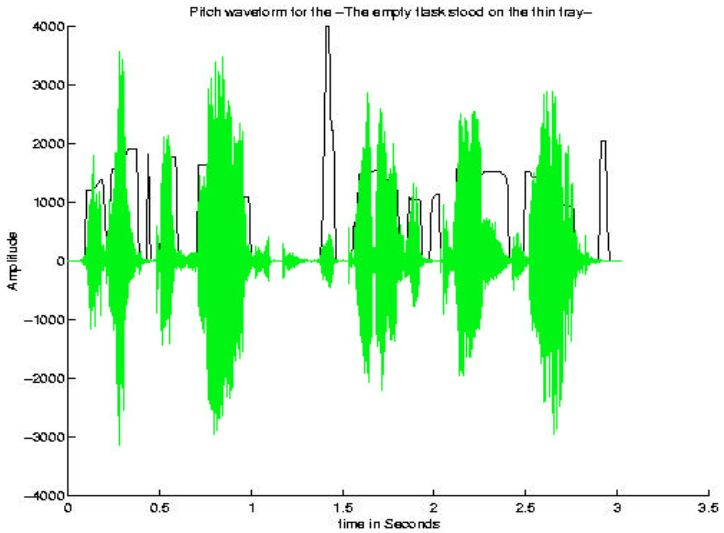

Figure 11. Plot of the original speech signal contained in the file CW161 and the pitch values calculated by the Channel analyzer encoder.

Question 5 If you have attempted any of these optional parts. Briefly describe your results and observations.

I have described all my observations in the previous sections numbered from 4 to 8. All the code for this optional part of the lab can be found in Section IV of the appendix.

3.4)._Linear Prediction analysis of a vowel

1._The plot for the vowel with the hamming window applied is shown in Figure 12. I know that for the this lab and the development of the LP encoder/decoder, it is not necessary to apply the hamming window since this screws up the pitch detection code since the vowel contains the envelope of the hamming window.

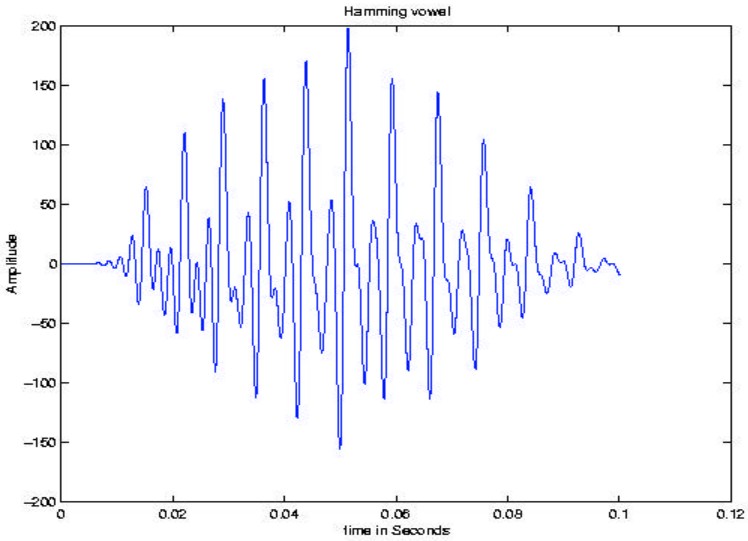

Figure 12. Vowel with a hamming window applied.

2._After applying the hamming window. I calculated the LP coefficients and gain of this segment of speech signal corresponding to the vowel. The results are as follow using a model order of 12.

coeff =

1.0000
-2.3447
1.2787
0.3071
0.0101
-0.0400
-0.2263
-0.1114
0.2330
-0.2700
-0.0310
0.4445
-0.2396

gain = 21.2603

3. and 4._I ploted the frequency response of the all pole LP model filter (dBs v Hz) and the DFT of the original speech segment in the same coordinates. This can be seen in Figure 13 for a model order of 12. Figure 14 shows the same for a model order of 8, Figure 15 for a model order of 20 and Figure 16 for a model order of 50.

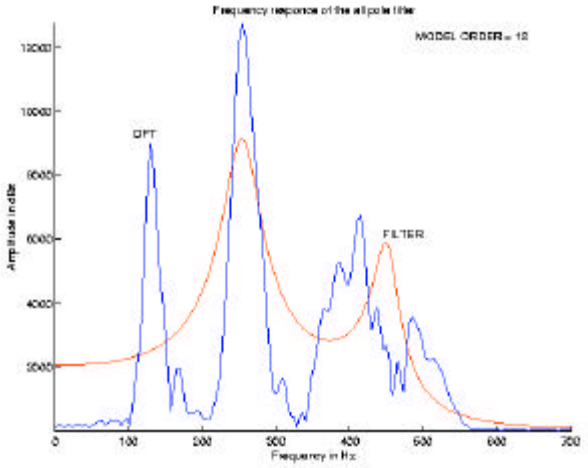

Figure 13. Frequency response of the all pole filter for a model order of 12

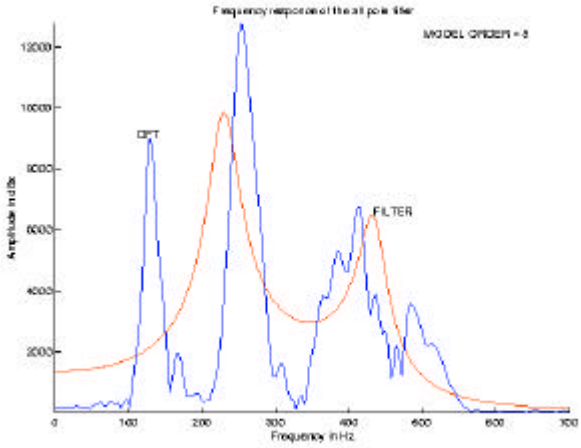

Figure 14 Frequency response of the all pole filter for a model order of 8

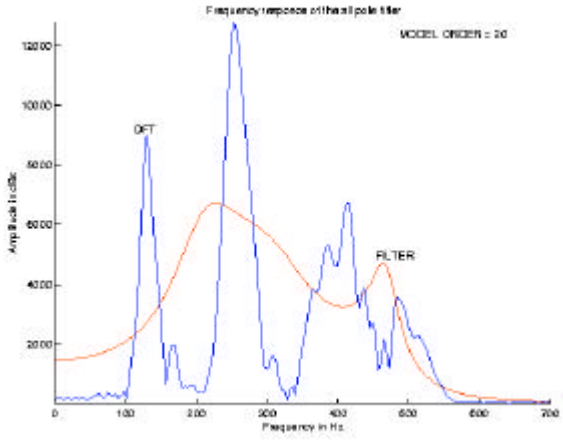

Figure 15 Frequency response of the all pole filter for a model order of 20

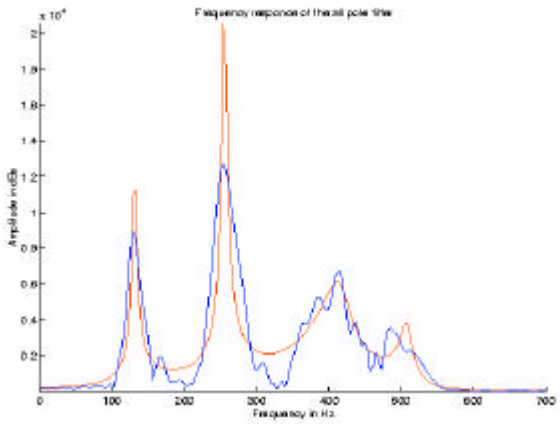

Figure 16. Frequency response of the all pole filter for a model order of 50

Question 6. Compare the magnitudes of the LP and DFT spectra, including plots of both. Which spectrum gives a better representation of the formants? Which spectrum gives a better representation of the fundamental frequency? Which model order gives the best representation of the formant frequencies?

Before answering to all these questions, let me remember what is the definition of formants. A formant are the sharp peaks in the frequency response of the filter representing the resonance frequencies. The first formant correlates with the degree of opening in the vocal tract. The second formant correlates with the front-back position of the tongue. Please also remember that the fundamental frequency is the smallest frequency, that is the frequency of the first peak. Having this in mind, now we can start answering all the questions.

Which spectrum gives the better representation of the formants?
Since the formants are the peaks in the frequency response of the filter, the model that better represent the formats is the one with the highest model order. In my case, even the model with order equal to 20 does not correctly (accurately) represent the formants. That is why I included an additional plot for a model order of 50 which fully represents all the formants for my vowel 'e'.

Which spectrum gives a better representation of the fundamental frequency?
Since the fundamental frequency is the frequency of the first peak, the model that best represent the fundamental frequency is the one with highest order. In my case is the model with order equal to 50.

Which model order gives the best representation of the formant frequencies?
It is also the model with highest order. In my case, the model with order 50.

Because of the computational time required to calculate filter models with high order, it is necessary to have a tradeoff between computational time and accuracy in the representation of the formant, fundamental and formant frequencies. For this lab, I selected the order of the model for the LP vocoder to be 12.

5._I this optional part of the lab, I calculated the error signal resulting from applying the best order from step 3 and 4 (This is, the model order of 12 since this gives us a good approximation of the formant frequencies and at the same time, it doesn't require a lot of computational time) by applying the inverse of theLP model filter.

The plot of the error signal is shown in Figure 17. The error signal has a very similar waveform to the one of the original vowel but with less amplitude. This is correct because we know from the theory that the error signal is the estimate of the source for the LP vocoder model.

I verified that the LP gain equal to the square root of the energy in the error signal. I first calculated the squared each value contained in the error vector. Then I summed up all the values and I calculated the square root of the sum. The result for this operation is 21.24

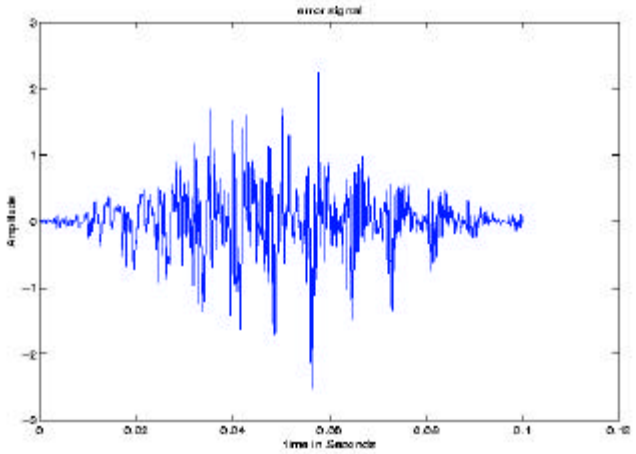

Figure 17. Error signal resulting from applying the LP model filter to the speech signal

Question 7. If you did this optional part, include plots and briefly describe your results and observations.
The plot of the error signal is the one shown in Figure 17 and I put all the comments in the previous section (section 5).

3.5)._LP Vocoder

1._In this part of the lab, I designed a matlab function to encode the speech signal using the Linear prediction model.

The function takes the following parameters: [coeff, gain, pitch] = lpvocod_ana(x, p)

Where x is the input signal to be coded and p is the desired order of the LPC model. The function returns coeff which is a matrix of LP coefficients (column index = frame number; row index = coefficient number), gain is the vector containing the gains (one gain per frame), and pitch is the vector of pitch values (also one pitch value per frame).

The steps that I followed in order to design this function are the following.

? Design a low pass filter with a cutoff frequency of 500Hz
? Chop the original signal in segments of 30msec overlapping by 15msec.
? Loop for all the frames of data (segments)
    i._compute the coefficients and gain for this segment (using lpcoef) and store the results
    ii._apply the inverse filter A to the segment signal to get error signal
    iii._filter the error signal using the low pass filter previously designed
    iv._Detect pitch of segment (frame) from the error signal
    v._avance to next frame or segment
? Apply a median filter to the pitch signal to eliminate spurious values.

The MATLAB code for the linear prediction encoder can be found in section V.

2._In this part of the lab, I designed a MATLAB function corresponding to the decoder or synthesizer using the linear prediction model.

The function uses the following parameters.

y = lpvocod_syn(coeff, gain, pitch, fr_int)

where coeff is a matrix of LP coefficients (column index = frame number; row index = coefficient number), gain is the vector containing the gains (one gain per frame), and pitch is the vector of pitch values (also one pitch value per frame) all of them calculated using the lpvocod_ana.m function. fr_int is the time between frames. In this case, it must be always .015. The funcion returns y which is the synthesized speech signal.

The algorithm or steps that I used in order to design this function are the following

? Loop over all the frames
    i._if the pitch value of the frame is greater than zero, then generate a pulse train with frequency corresponding to the pitch and keeping the delay generated to use in the next frame. Otherwise, generate a random signal with the same length of the frame. In this latter case, it is not necessary to keep the delay.
    ii._normalize the energy in the generated signal for this frame to be the unity. I did this by dividing all the signal generated by C, where C = sqrt(sum(p.^2)) and taking care of not dividing by zero.
    iii._filter the generated source signal with the LP model filter keeping the filter state to use for the next data frame.
    iv._insert the generated signal in the output vector y.

NOTE: the important thing in this function is to keep track of the delays in the signals generated by the function pulse_train and the filter state when filtering the signal with the function filter in order to eliminate perceptually undesired continuities.

The MATLAB code for the linear prediction decoder can be found in section VI.

3._ I used the LP analyzer and synthesizer to encode and decode the entire speech signal. Then, I submitted the signal using the submit_lab2 function.

Question 8 Based on your listening, describe how the synthesized speech from the channel and LP vocoders compare to each other and to the original speech. Submit your original speech, channel vocoder synthesized speech, and LP synthesized utterances (using the function submit lab2)so the instructors can listen to them when grading your lab write-up. You may reuse the submit function as many times as you wish, but only the last set of sentences submitted with each vocoder type will be saved.

comparing the utterances generated by the channel vocoder to the ones generated by the linear prediction model, I was able to confirm what we saw in theory in the classroom. The linear prediction utterance sounded better than the one generated by the channel vocoder. The main difference is that it was more clear and sounded with a better quality. I also sent the synthesizer speech signals for grading at this point.

Question 9 Assuming 16-bit quantization, what is the bit rate (in bits/sec)of the original speech? Now assume that each pitch value can be encoded in 8 bits and that each band envelope value requires 12 bits. What is the bit rate of your channel vocoder? What is the bit rate of your LP vocoder? How do these three bit rates compare?

Original Signal

$$\left(\frac{8000\ \text{Samples}}{Sec}\right)\left(\frac{16\ \text{bits}}{\text{Sample}}\right) = 128\frac{Kbits}{Sec}$$

Channel Vocoder (for 18 bands in the filter bank)

$$\left(\frac{200\ \text{pitch values}}{3\,Sec}\right)\left(\frac{8\ \text{bits}}{\text{pitch value}}\right) + \left(\frac{18 \cdot 200\ \text{bands}}{3\,Sec}\right)\left(\frac{12\ \text{bits}}{\text{bands}}\right) = 14.933K\frac{bits}{Sec}$$

Linear prediction Vocoder (for a LP model order of 12)

$$\left(\frac{200\ \text{pitch values}}{3\,Sec}\right)\left(\frac{8\ \text{bits}}{\text{pitch value}}\right) + \left(\frac{12 \cdot 200\ \text{coeff}}{3\,Sec}\right)\left(\frac{12\ \text{bits}}{\text{coeff}}\right) + \left(\frac{1 \cdot 200\ \text{gains}}{3\,Sec}\right)\left(\frac{12\ \text{bits}}{\text{gains}}\right) = 10.9K\frac{bits}{Sec}$$

The best vocoder in terms of bits per second rate and signal quality is the Linear Prediction vocoder.

3.6)._Spectrograms

1 and 2._I preemphasized the original speech signal and the two synthesizer outputs and then calculated the spectrogram of each signal using the function specgram6555. I selected a window length of 50 in order to have a resolution of 300Hz. This is because the f in f=2/L gives 300Hz for this L value.

Figure 18 show the spectrogram of the original signal, Figure 19 the one for the Channel signal and Figure 20 for the Linear prediction signal.

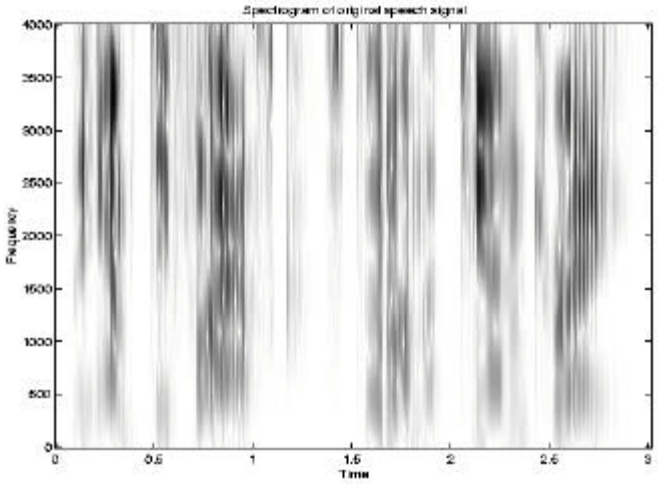

Figure 18 Spectrogram of the original speech signal

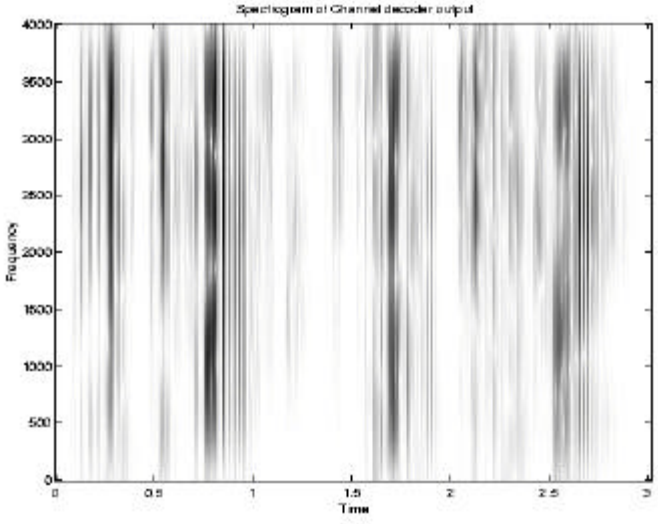

Figure 19 Spectrogram for the Channel vocoder synthesized signal

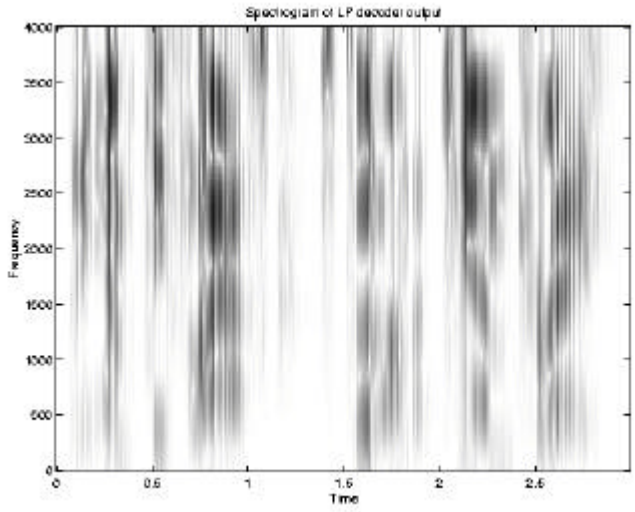

Figure 20 Spectrogram for the Linear prediction vocoder synthesized signal

How is a formant represented in a spectrogram?

The peaks that occur in the sound spectra of the vowels, independent of pitch, are called *formants*. Just three formants are typically distinguished. A formant is a dark band on a broadband spectrogram, which corresponds to a vocal tract resonance

Reading spectrograms

Voiced sounds: formants are large concentrations of energy that characterize voiced phonemes. Voiceless phonemes are not usually said to have formants. Vowels can usually be easily distinguished by the frequency values of the first two or three formants, which are called F1, F2, and F3

In the vowels, F1 can vary from 300 Hz to 1000 Hz. The lower it is, the closer the tongue is to the roof of the mouth. The vowel /i:/ as in the word 'beet' has one of the lowest F1 values - about 300 Hz; in contrast, the vowel /A/ as in the word 'bought' (or 'Bob' in speakers who distinguish the vowels in the two words) has the highest F1 value - about 950 Hz. Pronounce these two vowels and try to determine how your tongue is configured for each.

F2 can vary from 850 Hz to 2500 Hz; the F2 value is proportional to the frontness or backness of the highest part of the tongue during the production of the vowel. In addition, lip rounding causes a lower F2 than with unrounded lips. For example, /i:/ as in the word 'beet' has an F2 of 2200 Hz, the highest F2 of any vowel. In the production of this vowel the tongue tip is quite far forward and the lips are unrounded. At the opposite extreme, /u/ as in the word 'boot' has an F2 of 850 Hz; in this
vowel the tongue tip is very far back, and the lips are rounded.

F3 is also important is determining the phonemic quality of a given speech sound, and the higher formants such as F4 and F5 are thought to be significant in determining voice quality.

Voiceless sounds: are not usually said to have formants;

General approach

In the unvoiced fricative sounds, the energy is concentrated high up in the frequency band, and quite disorganized (noise-like) in its appearance. In other unvoiced sounds, e.g. the plosives, much of the speech sound actually consists of silence until strong energy appears at many frequency bands, as an "explosion". The voiced sounds appear more organized. The spectrum highs (dark spots) actually form bands across the spectrogram. These bands represent frequencies where the shape of the mouth gives resonance to sounds. The bands are called formants, and are numbered from the bottom up as F1, F2, F3 etc. The positions of the formants are different for different sounds and they can often be predicted for each phoneme

Question 10 How do the three spectrograms compare? Include your spectrogram plots. How do the two different vocoders affect formants?

How do the three spectrograms compare?

In the original one, we can see all the original energies of the speech signal. The vertical lines are clear, well separated and distinguishable.

In the channel vocoder spectrogram is the one with less energy through all the speech signal. This correlates with the fact that the channel vocoder does have a really good synthesizing quality. This spectrogram also shows that the channel vocoder eliminates (attenuates) the energy corresponding to high frequencies, since the grey areas in the top of the spectrogram are almost missing.

The spectrogram of the linear prediction vocoder shows more energy than the spectrogram for the channel vocoder. This also correlate with the fact that this vocoder have a better synthesizing quality. It seems to me that the LP vocoder is better than the channel vocoder in keeping high frequencies and eliminating less vertical lines.

Another characteristic is that the spectrogram of the synthesizers eliminate some vertical lines from the original spectrogram.

How do the two different vocoders affect formants?
The formants in the spectrograms of the synthesizers have less energy than the original ones. This can be seen in the spectrogram as areas with less gray level intensity. I think that this is normal, since we are able to reduce the bit rate of the speech signals by eliminating the energy in the speech signal. The good vocoders are those that just eliminate the energy (or signals) that are not listened by the human ear. In this case, I think that the LP vocoder makes a better job.

Question 11. Compare and contrast the basic approaches to speech coding used in the channel vocoder and in the linear prediction vocoder. In what ways are the designs of these systems similar? In what ways they differ?

To do this, I first, I am going to show the algorithms or steps in the construction of the encoder part for both vocoders.

Channel vocoder encoder

- Design a lowpass filter with 500Hz cutoff frequency
- Apply the filter to the speech signal and store it in xlpf
- loop. Each iteration processes one frame of data:
    i._Get next 30mSec segment of data from the filtered sinal xlpf and call it seg
    ii._Call the pitch detector function using seg as paramater
    iii._ store the result
    iv._avance amount corresponding to the decimation rate to next frame

- apply a median filter of order=3 to the pitch signal p in order to eliminate spurious values
- Design a filter bank of N bands and order=65.
- In loop, process each band:
    i._Apply filter for this band
    ii._Take magnitude of signal and decimate. (using abs)
    iii._decimate and low pass filter the signal for this band and store the result.
        at this point, each row in Y has a length of D (number of points)resulting from the decimation. Since we also have D number of segments, Y is a matrix in which each column represents a segment of the signal and each row represents a band in the filter bank

LP vocoder encoder

- Design a low pass filter with a cutoff frequency of 500Hz
- Chop the original signal in segments of 30msec overlapping by 15msec.
- Loop for all the frames of data (segments)
    i._compute the coefficients and gain for this segment (using lpcoef) and store the results
    ii._apply the inverse filter A to the segment signal to get error signal
    iii._filter the error signal using the low pass filter previously designed
    iv._Detect pitch of segment (frame) from the error signal
    v._avance to next frame or segment
- Apply a median filter to the pitch signal to eliminate spurious values.

The similarities are the following

Both chop the signal in overlapping frames
Both filter the frame before pitch detection
Both use the pitch information

The differences are the following

Channel vocoder: The way in which it reduces the bit rate is by the decimation stage. It uses a filter bank to separate the signal in frequency bands and get only a magnitude value for them.

LP vocoder: The way in which it reduces the bit rate of the signal is to separate the source from the filter in the speech signal. It only stores the information of the filter rater than the signal or parts of the speech signal. It calculates the pitch of the signal from the error signal instead of calculating it from the original filtered speech signal like the channel vocoder.

Question 12. What did you find most challenging about this lab exercise?
The LP synthesizer function was the most difficult for me to design and code, since I had no clue of how to keep track of the delays and filter state for a good sound quality. Also, there so many things to do in this lab exercise, that I had to work very hard

Question 13 What is the most important thing that you learned from this lab exercise.
Deep understanding of how vocoders work, and specially, vocoders of high quality such as the Linear prediction vocoder.

Question 14 What did you like/dislike the most about this lab exercise?
The fact that we developed a useful code that might be useful for my research on biomedical applications/technologies in the home setting of the future. I am going to use the pitch detection algorithm to detect mental illnesses in little babies

Appendix

Section I.
MATLAB code for the pitch detector function pitch_detector.m

```
function [pitch] = pitch_detect(x)
%PITCH_DETECT            Performs pitch detection on a speech waveform
%
%       p = pitch_detect(x);
%            x is vector containing frame of speech data
%       p is scalar containing pitch of frame in Hz or 0 if unvoiced

Fs = 8000;

%  First, remove the dc value of the frame by subtracting the mean . . .

x = x-mean(x);

%  Then find the minimum and maximum samples and center clip to
%  75% of those values (`cclip') . . .
x = cclip(x,min(x)*0.75,max(x)*0.75);

%  Compute the autocorrelation of the frame . . .
x = xcorr(x,'coeff'); %calculates the autocorrelation and also normalizes the maximum of the correlation to be 1
maxim = find(x== max(x)); %finding the index of the maximum value
poscorr = x(maxim:length(x)); %extracting the positive (t = positive) part of the correlation

%  Find the maximum peak following Rx[0] (`peak') . . .
[peakporc index] = peak(poscorr);
peaktime = index/Fs; % converting from index of sample to seconds


% Determine if the segment is unvoiced based on the 'voicing strength'
% (the ratio of the autocorrelation function at the peak pitch lag
% to the autocorrelation function lag=0) . . .
%calculating what porcentage of the maximum at zero represents the founded maximum (without mult by 100)


% If voicing strength is less than 0.25, call it unvoiced and set pitch = 0,
% otherwise compute the pitch from the % index of the peak . . .
if peakporc<.25,
   %the segment is unvoiced
   pitch = 0;
else
   %the segment is voiced
   pitch = 1/peaktime;
end
```

Section II.
MATLAB Code for the filter bank generator myownfilter_bank.m

```
function bank=filt_bank(N,L)
% FILT_BANK     Filter bank generator
%               bank = FILT_BANK(N,L);
%                       N       number of bands
%                       L       length of each FIR filter (order of filter)
%               Generates an LxN matrix
```

```
%                   Each of the N columns contains an L-point FIR filters
%                   The following parameters are specified within the function
%                   so they'll be the same for both the analysis and synthesis
%                   portions of the channel vocoder:
%                           Fs      sampling frequency in Hz
%                           start   center frequency of first band in Hz
%                           B       bandwidth of each band in Hz
%

%  Fill in parameter values . . .
Fs = 8000;                  % sampling frequency in Hz
Fny = Fs/2;
start = 100;        % center frequency of first band in Hz
B = 200;            % bandwidth in Hz
FL = B;


% Design a prototype lowpass filter . . .
%       Choose the cutoff frequency to obtain a bandwidth of B.
%       We suggest that you use a 65 point FIR Kaiser window with beta = 3.
lpf = fir1(L-1,FL/Fny,kaiser(L,3)); %design by the windowing method

% Need to make it a column vector
lpf = lpf(:);

% Make a DT vector equal to the filter length for argument to cosine . . .
% the low pass filter samples are separated by 1/Fs seconds. We want
% to multiply if by a cosine of the same length whose samples are also
%separated by 1/Fs, then we have
n = ([0:L-1]/Fs)';


% In loop, design filter for each band:
for i = 1:N
  % Compute desired center frequency from i, B, start, and Fs . . .
  cf = i*B-start;
  lol(i) = cf; %erase this
  % Shift lowpass prototype to center frequency  . . .
  % See (See Problem Set 2, #1)
  if i==1
     lol
   bank(:,i) = lpf;

  else
     bank(:,i) = lpf .* cos(2*pi*cf*n)*2;  %default calculations in matlab for cos are done in radians
  end
end
%extra code for plotting the frequency response of the filter bank
lol
F = 1:Fny;
figure;
hold;
Band = abs(freqz(bank(:,1),1,F,Fs));
plot(F,Band,'r');
for z=2:N
  Band = abs(freqz(bank(:,z),1,F,Fs));
  plot(F,Band,'b');
end
hold;
title('Filter Bank Magnitude response for the first 5 bandas');
xlabel('Frequency in Hz');
ylabel('Amplitude');
```

Section III.
MATLAB code for the Channel Vocoder analizer function chvocod_ana.m

```
function [y,p]=chvocod_ana(x, D, N)
%CHVOCOD_ANA           Analyzes speech waveform into pitch and band envelope signals
%                     [y,p] = CHVOCOD_ANA(x, D, N)
%
%                     x         vector containing speech signal to encode
%                     D         decimation rate
%                     N         number of bands
%
%                     y         output matrix of band envelope signals
%                               each column contains output of one band
%                               each row contains output of one frame
%                     p         output vector containing pitch signal with one
%                               sample for each frame

% Make x a column vector just to be sure.
x = x(:);

Fs = 8000;                                  % sampling frequency
Fny = Fs/2;        %nyquist frequency
FL = 500;  %low cutoff frequency of low pass filter for filtering the signal
order = 200; %order of filter to filter the speech signal
frlen = floor(0.030 * Fs);      % use 30ms frame length
nframes = ceil(length(x)/D);    % number of frames


% preallocate output for speed
pitch = zeros(nframes, 1);
% Preallocate output matrix "y" for efficiency.
y = zeros(nframes,N);

%--------------------- get "source parameters" (pitch detection)-------------

% First, lowpass filter the signal with 500Hz cutoff frequency,
% since we're interested in pitch, which is 80-320 Hz for adult voices.
% (Note that the lowpass filtered signal should be used as the input
% to the pitch detector, but not to the filter bank.)

%filtering the vowel to preserveonly frequencies below 1kHz
Bfir1 = fir1(order,FL/Fny); %design lowpass filter by the windowing method
xlpf = fftfilt(Bfir1,x); %filtering the speech signal


% Here's the loop. Each iteration processes one frame of data.

for i = 1:nframes

  %  Get the next segment.  The indexing has been done for you.
  startseg = (i-1)*D+1;
  endseg = startseg+frlen-1;
  if endseg > length(xlpf)
    endseg = length(xlpf);
  end
  seg = xlpf(startseg:endseg);

  % Call your pitch detector . . .
```

```
  p(i) = pitch_detect(seg);

end

%remove spurious values from pitch signal with a median filter
p = medfilt1(p,3);

%--------- get "filter" parameters (determine band envelope values)-----------

% Compute FIR coefficients for filter bank. Use 65-point filters(order = 65). . .
bank = myownfilt_bank(N,65);

%filtering the signal


% In loop, process each band:

for i=1:N,
    %      Apply filter for this band (bank(:,i)) to input x (not xlpf) . . .
    segbpf = fftfilt(bank(:,i),x); %filtering the speech signal

    %      Take magnitude of signal and decimate.
    segbpf = abs(segbpf);

    %      (The matlab function decimate includes lowpass filtering) . . .
    y(:,i) = decimate(segbpf,D);
    %at this point, each row in Y has a length of D (number of points)
    %resulting from the decimation. Since we also have D number of segments,
    %Y is a matrix in which each column represents a segment of the signal
    %and each row represents a band in the filter bank

end

%format of matrices in matlab is
%matrix(F,C)
```

Section IV
Optional code for generating the monotone, whisper and female sounds for the optional part of the lab

```
%testing the Channel Vocoder
load('/mit/6.555/data/vocod/cw161_8k.mat');
[y,p] = chvocod_ana(cw161,100,18);
voicesyn = chvocod_syn(y,p,100);
%listening to the syntetized version
soundsc(voicesyn);

pitch = ones(1,length(p)).*100;
voicesyn = chvocod_syn(y,pitch,100);
%listening to the monotone version
soundsc(voicesyn);

pitch = zeros(1,length(p));
voicesyn = chvocod_syn(y,pitch,100);
%listening to the whispered version
soundsc(voicesyn);

pitch = p.*2;
voicesyn = chvocod_syn(y,pitch,100);
%listening to the whispered version
soundsc(voicesyn);
```

Section V
MATLAB code for the LP encoder

```
function [coeff, gain, pitch] = lpvocod_ana(x, p)
%LPVOCOD_ANA            Analyzes speech waveform into pitch, gain, coefficients
%                   [coeff,gain,pitch] = LPVOCOD_ANA(x, p)
%
%                   x       input signal
%                   p       order of LPC model
%
%                   coeff   matrix of LP coefficients
%                           (column index = frame number;
%                       row index = coefficient number)
%                   gain    vector of gain values (one per frame)
%                   pitch   vector of pitch values (one per frame), 0=unvoiced
%
%

% Make x a column vector just to be sure.
x = x(:);

% Initialize variables for loop.

Fs = 8000;                              % sampling frequency
Fny = Fs/2;     %nyquist frequency
FL = 500;      %low cutoff frequency for the low pass filter to apply to each segment
frlen = round(0.03 * Fs);       % length of each data frame, 30ms
order = frlen/5; %order of the filter to apply to each segment
noverlap = fix(frlen/2);        % overlap of successive frames, half of frlen
hop = frlen-noverlap;                   % amount to advance for next data frame
nx = length(x);                 %length of input vector
len = fix((nx - (frlen-hop))/hop);          %length of output vector = total frames

% preallocate outputs for speed
gain = zeros(1, len);
pitch = zeros(1, len);
coeff = zeros(p+1, len);
coeff(1,:) = ones(1,len);
vstrength = zeros(1,len);

% also preallocate window . . .
Hwin = hamming(frlen);

% Design (but do not yet apply) a lowpass filter with 500Hz cutoff
% frequency, since we're interested in pitch, which is 80--320 Hz for
% adult voices.  Make the filter length(order) substantially shorter than the
% frame length . . .
Bfir1 = fir1(order,FL/Fny); %design lowpass filter by the windowing method


% Here's the loop. Each iteration processes one frame of data.

for i = 1:len
%  Get the next segment/frame of the unfiltered input signal.
%  The indexing has been done for you.
   seg = x(((i-1)*hop+1):((i-1)*hop+frlen));

%  Window the segment . . .
%seg = seg .* Hwin;
```

```
%  Compute the LPC coefficients and gain of the windowed segment (`lpcoef')
%  and store in coeff matrix and gain vector . . .
[coeffseg,gainseg] = lpcoef(seg,p);
coeff(:,i) = coeffseg;
gain(i) = gainseg;

% Compute the Linear Prediction error signal
%coefficients in position of B because they are already inverted
Scalc = filter(coeffseg,1,seg);  % aplying the filter A
e = Scalc; % I dont need to substract the signal because it is already done in the coefficients

% Detect voicing and pitch for this frame by applying 500Hz lowpass filter
% to error signal and then calling your pitch detector . . .
fe = fftfilt(Bfir1,e); %filtering the vowel signal
pitch(i) = pitch_detect(fe);
aux((i-1)*frlen+1:i*frlen) = fe;  %delete this


end                     %end for loop

% Remove spurious values from pitch signal with median filter (do not use order more than three)
pitch = medfilt1(pitch,3);

soundsc(aux);
```

Section VI
MATLAB code for the LP decoder

```
function y = lpvocod_syn(coeff, gain, pitch, fr_int)
%LPVOCOD_SYN            Synthesize speech waveform from pitch, gain, and LPC
%                       coefficients
%
%                       y = LPVOCOD_SYN(coeff, gain, pitch, fr_int)
%
%                       coeff       matrix of LP coefficients
%                                   (column index = frame number;
%                               row index = coefficient number)
%                       gain        vector of gain values (one per frame)
%                       pitch       vector of pitch values (one per frame), 0=unvoiced
%                       fr_int      frame interval (sec)
%
%                       y           synthesized speech signal
%

% Error checking
if (nargin < 3), error('There must be 3 or 4 input arguments'); end;
[nrows nframes]=size(coeff);
if (nframes ~= length(pitch)), error('Pitch vector has illegal length'); end;
if (nframes ~= length(gain)), error('Gain vector has illegal length'); end;

% Initialize variables for loop.
Fs = 8000;                                  % sampling frequency
if (nargin < 4), fr_int = .015; end;  % assume 15-msec frame interval
frlen= round(fr_int*Fs);

% Preallocate output for efficiency
y=zeros(nframes*frlen,1);
delay=0;                        % delay to first pitch pulse
filt_state = zeros(size(coeff,1)-1,1);
```

```
% Loop over frames to generate total source signal
for i=1:nframes
   % Pitch value for each frame indicates if voicing source or noise source
   if pitch(i) > 0
      % Compute pitch period and generate impulse train (`pulse_train') . . .
      % (Be sure to save new delay for next frame)
      [p, delay] = pulse_train(round(Fs/pitch(i)),frlen,delay);
   else
      % Generate noise source (`randn') . . .
      p = randn(frlen,1);
   end

   % normalize source signal to have unit energy, taking care not
   % to divide by zero . . .
   c = sqrt(sum(p.^2));
   if c == 0,
      p = p;
   else
      p = p/c;
   end

   %  Now, filter source signal through LP model filter,
   %  keeping track of the filter state to avoid discontinuities . . .
   [res,filt_state] = filter(gain(i),coeff(:,i),p,filt_state);  % aplying the filter

   % Insert frame into output signal . . .
   y((i-1)*frlen+1:i*frlen) = res;

end
```

Section VII
MATLAB code for multiple purposes

```
%lab sesion number two speech VOCODER

%Variables%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
filename = '/mit/6.555/data/vocod/cw161_8k.mat';
Fs = 8000;  %sampling frequency of 8KHz
Fny = Fs/2;
order = 200; %order of the filter
FL = 500;  %low cutoff frequency of low pass filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;  %close all the graphical windows opened

%audiodata = auread(filename);  % audio vector with values between -1 and 1 otherwise clipped
load(filename);
%sound(audiodata,Fs);
%soundsc(audiodata); % scale the sound vector so that it can be played as loud as possible
%soundsc(cw161);


%plotting the signal
time = 1:length(cw161);
time = time * (1/Fs); % converting samples to seconds
figure;
plot(time,cw161)
%plot(time,audiodata);
```

```
title('The empty flask stood on the tin tray');
xlabel('number of samples');
ylabel('Amplitude');
soundsc(cw161(1:500));



%Extracting the vowel in a 100 msec segent
%the vowels are the segments of the signal with high peaks, which means
%that they have formants
vowel = cw161(6300:6300+800);  %correct interval of a vowel
soundsc(vowel);
time = 1:length(vowel);
time = time * (1/Fs); % converting samples to seconds
figure;
plot(time,vowel);
title('Vowel e');
xlabel('time in Sec');
ylabel('Amplitude');


%filtering the vowel to preserveonly frequencies below 500Hz
Bfir1 = fir1(order,FL/Fny); %design lowpass filter by the windowing method
fvowel = fftfilt(Bfir1,vowel); %filtering the vowel signal
figure;
plot(time,fvowel);
title('Vowel e filtered with a low pass filter of Fc=500Hz');
xlabel('time in Sec');
ylabel('Amplitude');



%calculating the autocorrelation of the vowel
%
vowelcorr = xcorr(fvowel);
figure;
time = 1:length(vowelcorr);
%we need to find the maximum of the correlation and make it
%correspond with zero
maxim = find(vowelcorr== max(vowelcorr));
time = time-maxim;     %making correspond the maximum with zero in the time scale
time = (time/Fs)*1000; %adjustment to show the scale in mSec
plot(time,vowelcorr);
axis([-30 30 -inf inf]);
title('Autocorrelation of Vowel e, pitch = 1/.0045 = 222.22Hz');
xlabel('time in mSec');
ylabel('Amplitude');



%clipping the signal to have a better way to find the pitch
%this suppress the peaks due to vocal tract transfer function
vowelclip = fvowel;
vowelclip = vowelclip - mean(vowelclip); %eliminating the mean of the filtered signal
vowelclip = cclip(vowelclip,min(fvowel)*0.75,max(fvowel)*0.75);
figure;
time = 1:length(vowel);
time = time * (1/Fs); % converting samples to seconds
plot(time,vowelclip);
title('clipped signal of the vowel e ');
xlabel('time in Sec');
```

```
ylabel('Amplitude');


%calculating the autocorrelation of the filtered and clipped vowel
vowelcorr = xcorr(vowelclip);
figure;
time = 1:length(vowelcorr);
%we need to find the maximum of the correlation and make it
%correspond with zero
maxim = find(vowelcorr== max(vowelcorr));
time = time-maxim;     %making correspond the maximum with zero in the time scale
time = (time/Fs)*1000; %adjustment to show the scale in mSec
plot(time,vowelcorr);
axis([-30 30 -inf inf]);
title('Autocorrelation of clipped Vowel e pitch = 1/0.0068 = 147Hz');
xlabel('time in mSec');
ylabel('Amplitude');


%testing the function to detect the pitch
vowelpitch = pitch_detect(fvowel);


%testing the Channel Vocoder
load('/mit/6.555/data/vocod/cw161_8k.mat');
[y,p] = chvocod_ana(cw161,100,18);
voicesyn = chvocod_syn(y,p,100);
soundsc(voicesyn);

%doing section 3.4
%applying a hamming window to the vowel
Hvowel = fvowel.*hamming(length(fvowel));
figure;
time = 1:length(Hvowel);
time = time * (1/Fs); % converting samples to seconds
plot(time,Hvowel);
title('Hamming vowel');
xlabel('time in Seconds');
ylabel('Amplitude');




[coeff,gain] = lpcoef(Hvowel,50);
F = 0:Fny/1024:Fny-Fny/1024;
%calculating the frequency response of the all pole filter
H = abs(freqz(gain,coeff,F,Fs)); %the order of coefficients is B,A
figure;
hold;
semilogy(F,H,'r');
title('Frequency response of All Pole filter');
fourier = abs(fft(Hvowel,2048));
semilogy(F,fourier(1:1024),'b'); %fourier(1:1024)
axis([0 700 -inf inf]);
hold;
title('Frequency responce of the all pole filter');
xlabel('Frequency in Hz');
ylabel('Amplitude in dBs');
```

```
%in order to calculate the error signal, we have to filter the signal
%using the coefficients and substract the result from the original signal
Scalc = filter(coeff,1,Hvowel);  % aplying the filter A
e = Scalc;
figure;
time = 1:length(Hvowel);
time = time * (1/Fs); % converting samples to seconds
plot(time,e)
title('error signal');
xlabel('time in Seconds');
ylabel('Amplitude');
```