Real-Time Digital Signal Processing

Prof. Rathna G N

Department of Electrical Engineering

Indian Institute of Science - Bengaluru

Lecture - 46

Lab: LPC for speech synthesis

Welcome back to real time digital signal processing lab. So, in the previous class we discussed about the speech coding. Today we will be seeing the demo of the linear predictive coding for speech synthesis. And then we will see little bit of theory and then we will go for the demo both on MATLAB as well as in the board actually that is DSK board. So, first the introduction because to give the feel of LPC what we discussed in the theory. So, this is linear predictive coding is widely used in speech coding, synthesis, speaker recognition and for speech storage.

So, these methods provide extremely accurate estimates of speech parameters and does it efficiently. So, the basic idea here is to closely approximate the speech sample as a linear combination of past samples. So, as you can see in the equation. So, what we have is $s(n) = \sum_{k=1}^{p} a_k\, s(n-k)\ \ for\ some\ values\ of\ p, a_k,\ s.$ So, coming to the architecture for both analysis and synthesis our speech is what is shown in the figure here. So, we have the input speech we can select depending on which speech we want to have the thing. And we have the digital filter that is FIR filter. So, which is going to do the pre emphasis on the thing and this can be directly given as input to our moving average filter which is shown as lattice moving average filter.

So, one can go to the net and then check what is moving average filter. So, as an input. So, the other way is what you are going to give to overlap analysis window what it goes into the section. So, next is it is passed through the hamming window. So, this is the FIR filter what we are going to design.

and then we are going to check for the autocorrelation basically. So, we have discussed the correlation. So, here it is going to correlate with whether it is going to match with the present scenario and it uses the Levinson Durbin algorithm to predict the coefficients basically and this is given as input our lattice moving average filter. So, we are going to do time varying analysis basically over the fiber net what it can be connected here we will be doing both analysis is this equation and synthesis is this part. So, we will be incorporating both of it and then see the demo.

Otherwise in the normal case this will be becoming a channel and you will be sending it out. And, then at the synthesis part that is at the receiving part. So, we have to depredict the thing that is we have to run the lattice filter this which is time varying moving average filter here. So, analysis receiver what it is going to have and then the output is connected to our de-emphasis and then output is going to be predicted. So, whatever input speech which is there which should be coming out.

So, what is the how it has been implemented little details have been given here. So, first is because it is a speech which has to be segmented and taking the windows of it. So, the speech signal is divided into overlapping segments for further computation. So, the parameters used in this test scenario is window is the hamming window. and window length chosen as 80 and window overlap is 40.

So, as you can see this is the speech signal which is a input signal and we will be considering the overlapping window. So, you will be seeing that this is the window length and this is the hop length what it is called and this is the overlap length. So, approximately you can take it as 50 percent of overlap what it has happened as you can see here it is given as 40. So, that is what the overlap length what will be taken and these are the segments as you can see in this window length. So, we will be cutting this speech into smaller segments.

So, and so on and then till the end of the speech and processing is going to happen. So, coming to the next is the autocorrelation. So, what we have to do it on this windowed signal that is this is done to minimize the expected value of squared error between the predicted signal and the true signal value. So, it is computed using equation as you know $R_{xx}[k] = \sum_{m=-\infty}^{\infty} x[m]x[m-k]$

So, next is the Levinson Durbin algorithm for linear predictive coding what we have to use the thing. So, the equations of the Levinson Durbin recursion basically it is a recursive algorithm which are used to compute the corresponding reflection coefficients and LPC parameters. So, the equations you will be seeing that given by a to as you will be seeing up to e. So, this is your $E^{(0)} = \varphi[0]$ and $k_i = \frac{\varphi[0]-\sum_{j=1}^{i-1} \alpha_j^{(i-1)} \varphi[i-j]}{E^{(i-1)}}$. So, for this thing is $1 \leq i \leq P$ and then $\alpha_i^{(i)} = k_i$ and then after substituting in the equation. So, for $1 \leq j \leq i-1$ this is the recursion what will be doing it. So, expected value of i is calculated with this equation. And, then the updation α_j is given by $\alpha_j^{(P)}$ for $1 \leq j \leq P$ that is the final solution what it is given. So, when it matches.

So, the parameter is number of LPC coefficients chosen as 10 in this case. So, coming to the analysis filter here it is FIR filter B is being used, uses the LPC coefficients as filter weights and this filtering removes formants from speech signal and the remaining signal after the subtraction of filtered model signal is called residue. So, what is $y(n) = a_0 +$

$a_1x(n-1) + a_2x(n-2) + a_3x(n-3)$ you may be wondering why a coefficients has been chosen usually we choose a coefficients for poles and then b coefficients for poles. 0s basically in this case you can see that a coefficients have been used for 0s. So, you will be having a_0 plus as you can see this is $y(n) = a_0 + a_1x(n-1) + a_2x(n-2) + a_3x(n-3)$

And, in time domain is expressed in this fashion and in the z domain as you can see $H(z) = a_0 + a_1z^{(-1)} + a_2z^{(-2)} + a_3z^{(-3)}$ and filter length chosen is 10. So, you will be seeing $x(n)$ is the input these are the coefficients and then you are delaying the signal and you the other coefficients $a_1x(n-1)$ or $a_1z^{(-1)}$ what it is going to be. coming from this length basically and you will be adding with your a_0 and then you will be getting the $y(n)$. This is the filter structure what it shows that how it is getting calculated.

Now, in the synthesis part of it, it is the IIR filter. So, now, it must be triggering because the coefficients we have to keep the same a_0, a_1, a_2, a_3. So, here it is IIR filter whereas, in the analysis we use the FIR filter. Speed signal by reversing the analysis filter using the residue and LPC coefficients to recreate the formants. So, we will be using the all pole model of filter has the ability to describe most type of speed signals quite well.

So, whatever 0s in the analysis part now they become poles in the synthesis part as you will be seeing it here that is a reason why it was named as a coefficients. So, you will be seeing that impulse response $H(z) = \frac{Y(z)}{X(z)}$. So, which is given by some constant $\frac{G}{1-\sum_{k=1}^{P} \alpha_k^{(P)} z^{-k}}$. So, $y[n]$ is given as if we write in the time domain it is going to be $Gz[n] + \sum_{k=1}^{P} \alpha_k^{(P)} y[n-k]$. So, these are the feedback path and even here it is parameter what is it filter order is chosen as 10 equivalent to our analysis part.

So, coming to this the where all the LPC coding is going to be used applications what you will be seeing it in the speech compression example as GSM standard and then speech encryption in the voice codec also example is electronic music you can use the LPC coding. and tonal analysis of musical instruments you can use and even in the speaker recognition. So, these are the references to create the files what it has been used ok. Now, we will first see the demo in the MATLAB and then go to the demo.

hardware to check the thing. So, first I will show you the LPC or this thing DSP because this student has chosen as the project to implement it in the course as I say students will be taking as a mini project or they can use it as an assignment. So, what is it? Frame size what it is chosen 40 as theory says frame length is 80 and LPC coefficient number is chosen as 10. So, what is this part is going to do? System object to read from an audio file and determine the files audio sampling rate. So, when you are using it you may not know the thing. So, what we will do is first we will read one of the file and then we will test it on the other two also.

So, this is a Kannada. So, hold on what we are going to get through this. It is a M4 this thing format students own voice is recorded and then it is stored in the MP4 format. So, we will be taking input as we have to do the resample whatever the sampling rate may be the thing what we need is the 16 kilohertz. So, input with 16 kilohertz and then sampling frequency what we want to resample for 16 kilohertz. So, fs is going to be defined with 16 kilohertz here.

and then you will be starting and then some of the specific length of speech segment as 80 and then you have to do this zero padding to align on our this thing what is that length of it and then you will be making some of the variables 0. So, you will be creating a buffer system object and set its properties at that you get an output of twice the length of the frame size with an overlap length of frame size. So, what you want to have is although you are selecting 80 samples, 40 of them have to be overlapped. So, that is why you will be choosing the buffer length twice that of it. And, then in that there will be 40 right side and 40 left side overlap will be there.

So, this is the signal buffer what you are defining with twice that length. Next is each frame of input signal is windowed using a hamming window. So, you will be calling DSP dot window which is given with hamming window. So, the 10th order autocorrelation coefficients are found.

using this equations. So, you will be calling the function again DSP dot autocorrelator and then what is the maximum lag source property and then lag and LPAC or coefficient minus 1 and then scaling and biased or the So, you will be this section what it does is calculates the reflection coefficients from the autocorrelation results using the LPC spectrum. So, you will be calling the Levinson solver to get the coefficients as you are doing it. So, then you will be creating an FIR digital filter used for our adaptive purpose. In the analysis case also create two all pole digital filter system objects used for synthesis. So, you have to create the filters both for analysis and then synthesis.

So, this is the analysis filter which is FIR filter which is declared and then you will be seeing that it is a moving average lattice filter which is being used and then you will be getting the coefficient here and in the synthesis filter it is a all pole filter what you are designing it you will be giving the structure and then a lattice all pole value basically that is auto regression what you will be setting it. So, how you are going to play the thing it is going to be written audio writer. So, you will be writing into the audio device or writer. So, whatever the sample rate which is selected as fs in this case ok. So, main code where LPC analysis and synthesis of the input audio signal using instantiations done whatever you have set it ok.

So, the loop stops when we reach the end of the input file. So, which is detected by the audio file reader as a system object. So, you will be saying that frame size and then length of input minus 2 star frame size and then taking input in segmentation you will be taking it and then window each segment and then perform the autocorrelation here what you are doing it, then calculate the Levinson-Dubbin algorithm in this place and then synthesis filter is IR filter basically what it is being used and then you will be writing in this thing and then you can play output audio. So, writer what you are doing it and you will be. after everything you have to close the thing.

So, you will be releasing the audio writer part of it. So, we will as I said this is the first file what has been taken for reading and we will see the after synthesis sorry analysis and synthesis what output you would be getting it ok. So, we will run the thing. were you able to hear it.

So, the selected one is telling Namaskara. So, we can see once again running it. Namaskara. So, what we will see is we will run the other part of it. So, we will open something what it has been stored wave file.

So, we will run this. Good evening. Did you hear the thing again I will play it. so this is one more so you can as I said we can play with whatever your voice you want to have the thing so we will see this one go away so we can play the original and then see how it is going to look like go away So, now what I will do is this is from the MATLAB what we have played the thing. So, I can show you how it is going to be from the original way file whatever is it and then what we were able to reconstruct. So it takes a little time to play through this thing.

Go away. Go away. Go away. Go away. Go away.

Go away. Go away. Go away. Go away. Go away. This was the original. So, what I will play here is from the this thing what is it analysis and then synthesis LPC coding what we have done the thing. So, you would be seeing that there is a little difference with the original just you will hear the thing again. But, still it has a little meaning in the thing. So, as you know LPC coding is 64 kilobits per second what we are going to have it.

So, the original voice in all the thing what you can hear it from this place. I will play the other one original. Good evening, good evening, good evening, good evening, good evening, good evening. So, as you can see this is a female voice now we have to take mean score of opinion here. So, what is it going to be? So, you will be hearing this good evening how it is going to be it has got reconstructed through our LPC coding.

So, this is what what you heard ok. So, there is a difference between the voice pitch and other things. So, you have to move on to the other standards and other things and see whether we are going to get the correctly ok that is what what you have to do it and then matlab. This is the original voice. So, you will be seeing that after you have already heard once again I am playing with the original and then the reconstructed. Still it was able to do the thing with the some of the functions.

So, you can recreate much more. So, before moving on to our demonstration in code composer studio just I thought I will show you one of the application of our filter. So, here it is usually we call it as multirate signal processing which those who are interested can look into the thing when I want to convert my CD quality that is 44.1 kilohertz file to a midi quality what we call it as 48 kilohertz. So, this is the input audio rate and then output audio we want it at 48 kilohertz. So, how we are going to do it is going to be in stages what we will be reconstructing.

So, we will be having some filters in between when we are up sampling it. And then we will be using the FFT that is audio and then FFT what it will be coming and then we will run and see this is one of the application of a filter. So, here it is going to happen in three stages basically. So, some of the filter you will be seeing it used here it is for. decimation and then interpolation and then these are the two ones will be configuring to two the other one will be last one will be this one.

There will be three stages of it and we will see that each stage how it is going to run with the thing ok. So, we will run this algorithm. So, and then see. So, what happened? So, you will be seeing that if I make it maximize on the thing.

So, you will be seeing here. these are the three stages through the filter what it has got passed ok. So, from 44.1 what will be going to 48 kilohertz which is depicted with respect to this. So, you will be seeing that initially the red one was at 44.

1 kilohertz. Then, based on our this thing both decimation and interpolation together we will be converting in 3 stages to 48 kilohertz. So, you will be seeing this is the black is the first stage. and then green is what it is going to be second stage and this will be third stage what waveform it is going to look like. So, as you heard from the signal this is a sine wave there is no difference between your stages from one to the other one. So, without compromising on any input data so, we can reconstruct from one frequency to the other frequency.

So, those who are interested so, this code will be up. So, you can look into the thing and then see that if you want to do from one why do you why do you want to convert it into one frequency at this frequency because as you we are seeing it. or speech is at 8 kilohertz and then from 8 kilohertz narrowband speech to wideband if I want to convert it from 8 to 16 kilohertz what I have to convert. So, this requires multirate processing basically. So, those who are interested as a beyond this course material you can go through and then get it.

So, we will stop the MATLAB demo here. So, we will go into the code composer studio and then check the demo of LPC coding. will see how to proceed using the code composer studio. So, this is the synthesis using LPC as it has been named and then we need some include files, Ivo files and some math library and then because we are going to use the codec in this case input and then output codec. So, we need this ac3106 init dot h is our codec.

So, we will be including the output. So, hamming window as even in the FIR filter when I was demonstrating usually we create using the MATLAB and all our filter coefficients and then keep it. So, you can see here hamming dot H what I will open. So, if this is the 80 order what it has been chosen ok, ham 80 and these are the coefficients in fixed point format what you have taken. So, fixed point format basically equivalent to what it represents in our board is integer format. So, you will be seeing 80 coefficients what it has been stored here and which is used for our scamming window coefficients basically this is the scaled version of it.

And here also again the overlap is defined with 40 length and then the frame length is defined with 80 that is twice that of it. So, you can include this thing speech samples what it is being stored. So, initially I have taken Kannada 2 dot hedge and we have the other hedge file. So, we will see how does it look like because.

you will be seeing that total length is 19520. So, this is the total length of the speech samples what we have it and this is how what it has got stored with respect to what I will say is in the fixed point format. So, you can see all of them are integers. So, the first one. So, after that what it is going to do? So, we have to define some data section we when we want to have in particular data place will define as hash pragma that what we want this final output in the dot output. So, that is store the output in external memory and we want the input to be also in the external memory then we will be putting it this way because it is length input and output are more.

So, the internal memory is less in the case. So, we want to store it in external memory. And, you will be initializing basically. So, to show that print is to debug basically in between if you are unable to get the output usually because this is a C code. So, in between to debug it.

So, you give print commands and then see whether it is matching with MATLAB. First usually all the codes are written in MATLAB and then tested that it is working because we prove it there sometimes we have the built in functions in MATLAB for all these things. So, when we try it out and then it is working then. usually sometimes we may not use the built in function we will write our own code. So, that the code can be mapped into our code composer studio or any hardware for that matter even if you are designing your thing in FPGA usually in the MATLAB we test the thing.

So, algorithm or in using the python code one can write their own algorithm. then will be going into or transfer it to our whatever hardware which supports the languages. FPGA we use the Xilinx board basically there we use both VHDL as well as Verilog combinations to implement it even sometimes C will be front end and then back end will be VHDL coding what we will be doing it. So, here also you will be seeing the front end in C. I will as I have promised one assembly program I will show you that how the coding length is going to reduce and then how you can predict what is the time it is going to take in later lab demonstration.

So, this is the interrupt driven. So, usually here it is used is interrupt 4 actually and then you are getting some input sample and then you want to put it in some dummy like dummy file we say here dummy variable what it is going to be put and then we will be taking it out ok. And, if the index is less than the total then you have to modify your input and then you will be making it 0 because your window length is less and then your input length is more. So, you will be what I call it as circular buffer what you will be putting it and then taking into. So, next is this is the segmentation function input into frames of size whatever length it has been specified.

So, from the start what it is going to do. So, you will be allocating some memory basically and then you will be starting. So, you will be bifurcating your input wave and then if you want to print it that signal loop and other things what you can print it. So, that it is entered into the loop ok. Then now comes the window function.

So, this multiplies the segment with hamming window. So, you are seeing that here it is doing the hamming within the loop and then converting result from you can see Q format what it is shown. So, because the multiplication is going to result with. Q 30 format. So, result has to be in the Q 15 format. So, you are seeing that it is getting shifted by this thing 16 bits and then here because we have 2 what we call it as sign bits that is why one left shift is going to happen and then later on 16 bits and then will be storing that.

in our this thing basically output which is going to show that whether it is coming or not ok. So, then what it is going next one is your autocorrelation function which is being implemented in C as you can see. So, this is X correlation what although it named as cross correlation. So, you will be using same data. So, that it can be for two different data set input or it can be the same input what you can provide.

So, you will be initializing your correlation 0 and then you will be looping till your frame length minus i. So, that index does not spill over in this case. So, you are doing the basically what is it only to the size of your data and then you will be returning from this function and then next is the LPC function what it is defined here. It uses the Levinson Durbin algorithm and then it is going to calculate your LPC coefficients from autocorrelated segment. And, you will be defining some of the variables and then allocating some memory here and then what are the loop functions you will be seeing that some of the k and e and then alpha what it is being defined in the equation which has to be incorporated.

And, then you will be iterating with this thing i is equal to 1 to order as iteration is going to variable. So, you will be seeing that whether it has entered into this for debugging purposes what you will be seeing in between printf statements. Now, this will be to the order you will be looping this is the start step 1 here. So, you are calculating your sum and then k value and reference value and then you will be calculating your modifying your alpha and then you will be using it in your equation basically in the Levinson Durbin equation. And, you will be trying to minimize your error basically and then E of i and then you will be resetting your alpha value and other things and then you will be freeing all your memory here.

And, this is the residual function what it is being computed. Here, FIR filter to calculate residual signal from windowed signal and LPC coefficients. Numerator size minus number of filter coefficients is equal to our number of LPC coefficients what it is getting generated. So, this is the function what you will be writing it. So, then you will be summing it up and then to check whether it is doing it correctly you will be printing it.

Then next for the synthesis you have to run your IIR filter. So, it is a all pull synthesize your speech from whatever you have done the analysis part of it and then you will be reconstructing it. So, this is length here also IIR filter length is 10 what it has been considered. and then you will do your equation and then next is the buffering function. So, you want to accumulate because only for 80 samples what you are doing it and then you are doing it in concatenation that is there is a overlap and other things.

So, you have to reconstruct your signal with removal of overlap basically. So, you will be defragment and then put it in the concatenation using this function. So, finally, you are all those are the functions which are defined and then main function what we will be seeing it here ok. So, all your outputs and then have been defined with 0 and then the loop what you will be giving it that is I trace for every speed segment length here it is 80 and overlap what we have taken is 40. And every overlapping speed segment is going to be windowed and do the autocorrelation later and calculate your LPC parameters and calculate the residual signal calculating using the So, you will be doing the synthesis through IAR and output the accumulation based on overlap.

So, you will be seeing that these are the ones how you are going to free. And since we are using 3106 here L138 we have to initialize the interrupt service routine. So, we have to say what is the sampling rate we are providing it to the ADC, it is 16 kilohertz even DAC is at the same thing and gains what usually it is going to be 0 dB what we will be providing it and then input to this is going to be from LCDK line input. So, even the output as here it supports both mic and then line input mic is we have a differential input whereas, line will have both left channel and then right channel input what we can take it. So, that is how we will be working on one of the channel or if you have two signals we can work on left and then right channel if it has been separately given and then we will be looping back. So, now what we will do is if you are doing it for the first time this code it will take little time basically.

So, since I have already compiled and then done the thing directly I can go and then do the debugging part of it. So, you will be seeing that it has build has finished. So, the memory map what you are seeing it how it is getting mapped into the boards memory and then the code has to go and then load it on to the board.

So, you will be seeing that. the pointer you will be seeing in the C code. So, it has come to the main function where it is pointing to our program counter basically. So, it is now ready for the debugging is happened. I can cancel this memory map. So, you will be seeing so that your debug is going to be seen ok.

So, it has entered the main function here in the main. So, which is what we call in its you can see the code both in C as well as mixed assembly I will show you in a while. And this is where the entry point for the board basically C underscore int 00. So, this is the boot input what entry point it has reached the thing that is what it shows the thing. And then these are the what it says is we have used the ECDIS 110 USB debug cable for connecting it to the board.

So, we will run and then you will be hearing the synthesized output from the board. So, as you can hear that I am going to close my debugger as I did in the other example still the code is running on the board as you can hear it. So, unless I reset it will be continuously running. So, I can disconnect from the debugger. So, as you heard the thing I can disconnect from my this thing system and it can run directly as an input and output this thing system unit.

So, we will see with other example how it is going to be depicted that is input voice signal. take out from the this thing comment and then I will comment the other one. So, it if this it can be both in C and then C++. So, once I have done the modification I had to do this is going to do debug and then say any errors are present since I do not have any errors in the thing. So, what I will be doing is I can directly go into the debug mode. So, it will debug and then as you can see there are warnings the section which was specified it says it has not taken the things.

So, some dot h files have to be given. So, but still it is running. So, we sometimes if it does not run some of the warnings you cannot ignore you have to check with the thing. As you can see that it has become a male voice from the female voice. So, the last one we will see whether you would be able to hear from the board or not. I will be saving it and then I can debug.

So, to give different this thing we can create a gel file and from there we can provide different sections. So, that while running itself I can modify it, otherwise I have to do debug and then rerun the thing ok. So, good evening, good evening, good evening, good evening. Good evening. So, here it is you are able to at least hear the female voice, but for the good one of the voice almost getting I think swallowed or something it is coming out as a good evening.

We have seen the LPC speech coding basically both in MATLAB and then in Code Composer Studio using 6748 board. So, we will see other examples of scrambler, echo and equalizer in the using Code Composer Studio on the 6748 board in the next class. Thank you.