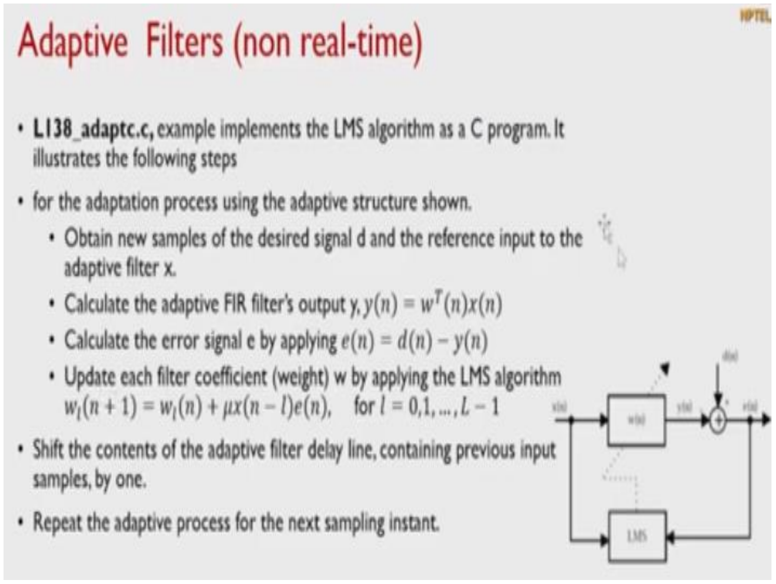
Real-Time Digital Signal Processing
Prof. Rathna G. N.
Department Of Electrical Engineering
Indian Institute Of Science, Bengaluru

Module No # 11
Lecture No # 52
Adaptive Filters (CCS) 2

Welcome back to real-time digital signal processing a lab basically. So, last class we had a problem with the code loading into the thing. So, today I will show you how you if there is any problem how you can load it.

(Refer Slide Time: 00:40)

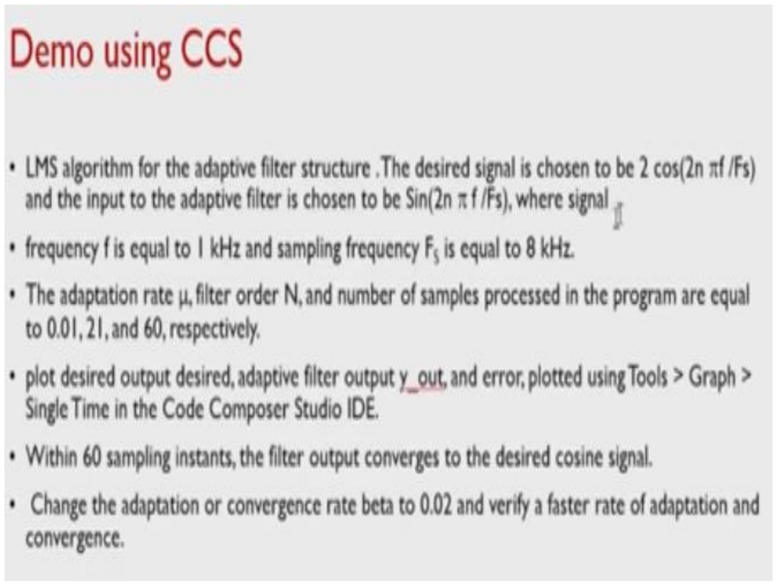


Adaptive Filters (non real-time)

- **L138_adaptc.c,** example implements the LMS algorithm as a C program. It illustrates the following steps
- for the adaptation process using the adaptive structure shown.
  - Obtain new samples of the desired signal d and the reference input to the adaptive filter x.
  - Calculate the adaptive FIR filter's output y, $y(n) = w^T(n)x(n)$
  - Calculate the error signal e by applying $e(n) = d(n) - y(n)$
  - Update each filter coefficient (weight) w by applying the LMS algorithm $w_l(n+1) = w_l(n) + \mu x(n-l)e(n)$, for $l = 0,1,\ldots,L-1$
- Shift the contents of the adaptive filter delay line, containing previous input samples, by one.
- Repeat the adaptive process for the next sampling instant.

So, we are discussing about the adaptive filters in non-real time. So, these were the equations that are updation of the weights here and error calculation and then the output calculation is here. And this is our LMS algorithm which will be running it in non-real time.

(Refer Slide Time: 01:00)



Demo using CCS

- LMS algorithm for the adaptive filter structure .The desired signal is chosen to be 2 cos(2n πf /Fs) and the input to the adaptive filter is chosen to be Sin(2n π f /Fs), where signal
- frequency f is equal to 1 kHz and sampling frequency F_s is equal to 8 kHz.
- The adaptation rate μ, filter order N, and number of samples processed in the program are equal to 0.01, 21, and 60, respectively.
- plot desired output desired, adaptive filter output y_out, and error, plotted using Tools > Graph > Single Time in the Code Composer Studio IDE.
- Within 60 sampling instants, the filter output converges to the desired cosine signal.
- Change the adaptation or convergence rate beta to 0.02 and verify a faster rate of adaptation and convergence.

So, these are the parameters what we have it that is 2 $\cos(2n\ \mu f\ /Fs)$ are this thing desired signal. And the input is chosen as a sign of a signal itself that is $2\pi\ \mu f\ /Fs$. So, here f = 1 kilohertz and sampling frequency chosen is 8 kilohertz so adaptation rate of mu is going to be set with 0.01 and order of the filter is 21 and we know within 60 or the iterations. So, the system will converge to the error are going to come down and then output will be equal into input.

So, this will be the desired plot what you can see it so this is going to happen in non-real time. So, we can change the adaptation or convergence rate and then see how it is going to behave.

(Video Starts: 02:01)

So, what we will do is we will go to the code composer studio as I said there was a problem in the previous class what we had it. So, all the directories if you go into the thing I have loaded all the labs things in my D drive basically. So, that is in CCS9 what I have all the files basically running, so this is my workspace basically for the code composer studio. So, yesterday I was seeing this and then I was having a little problem.

So, what I did was so go to the file if your by mistake all the complete workspace is cleared you can go and then you will be opening the projects from the file system. So, if you give this directory automatically all the files get reinstated in the directory as you can see here. I am having from the day one whatever demos I have been doing it is available here. So this is how you can get back all your directories and then you can run, so this is a non-real time what the demo is going to be.

So, what we have in this this is a main dot C file what we have it as it is pointed out this is the design signal what we want is $2\cos(2\pi T) * 1000Fs$ we said it is 1 kilohertz. So, the noise signal given as a sign that is $2 * \pi T$ or $1000/fs$ so this noise signal has to be eliminated and then we should be getting the desired size signal at the output. So, we will see how we are going to get the thing so this is desired and I have a y_0 and then error signal so fine.

In this case the w as we said the weight function so it is assigned to 0 initially even the x_0 will be getting the first noise signal here new noise sample; and D will be our desired signal; and Y is going to be 0. And the variable that is a filter output what we will be calculating y plus that is whatever weight you have computed into $x(i)$. And error is computed as d - y desired signal minus y output.

And you will be updating your weight $w(I)$ with $w(i) + \mu$ into error basically what you have computed here into input that is $x(i)$. So, in this case μ is assumed as 0.01 in this case that is what the initial so you can change it 0.02 to next and then see the convergence will be faster. And in Matlab π value automatically takes but here in code composer studio in C file you have to define your π value also.

So, it is defined up to 7 decimal places whatever length you want to define your pi you can define it. And sampling frequency we said 8000 order of the filter is 21 and then our number of iterations is chosen as 60. So, we will be calculating the update the error function and the delay because x also has to be further this thing delay line has to be a input line also has to be updated this is how we will be updating the delay line.

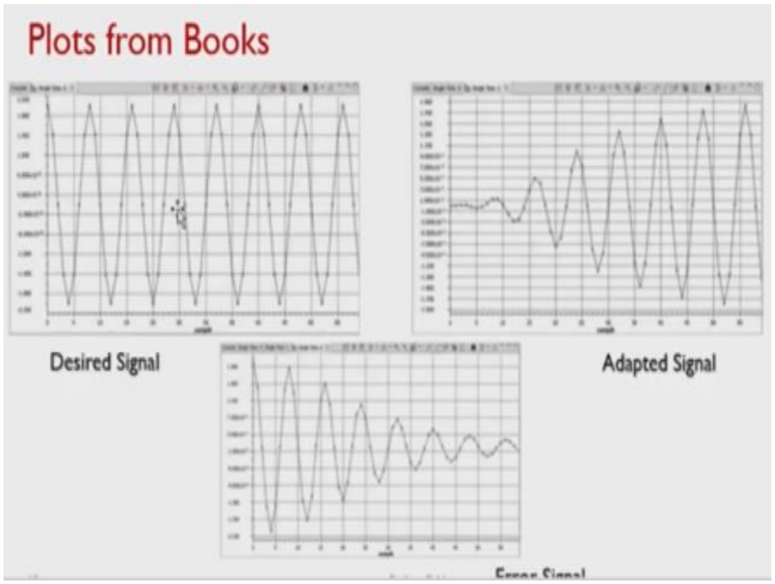
And we will be putting last desired of $t = d$ and then our y out will be y and then error function will be e. And then once it is done you can call it as done so what we do is because as we said the written function the handle is written elsewhere it is not in this memory so we will put a break point here. And then since it has been tested so we will what we will do is we will directly go for debugging.

So, you will be seeing that if there are errors it will ask you whether with errors it has to go for the debug since no errors so it will go for the debug. So, now I have as you can see the handle is in the main the flower bracket where it is flashing so it is ready to run with the thing. So, I have put the breakpoint as I said so we will run this code and then you will see it is going to run up to print f.

So, that will not have any error so you will be seeing that the some of the values what it has got changed and where location if you want to check in the variables what we have the values here. So, now what we will do is we have to plot that is what we have in the thing output what we have to see it.

(Video Ends: 07:09)

(Refer Slide Time: 07:10)



Plots from Books

So what is the thing this is the desired signal what it should look like and this is the adapted signal and this is the error you will be seeing it is initially high and then it is coming down.

(Video Starts: 07:20)

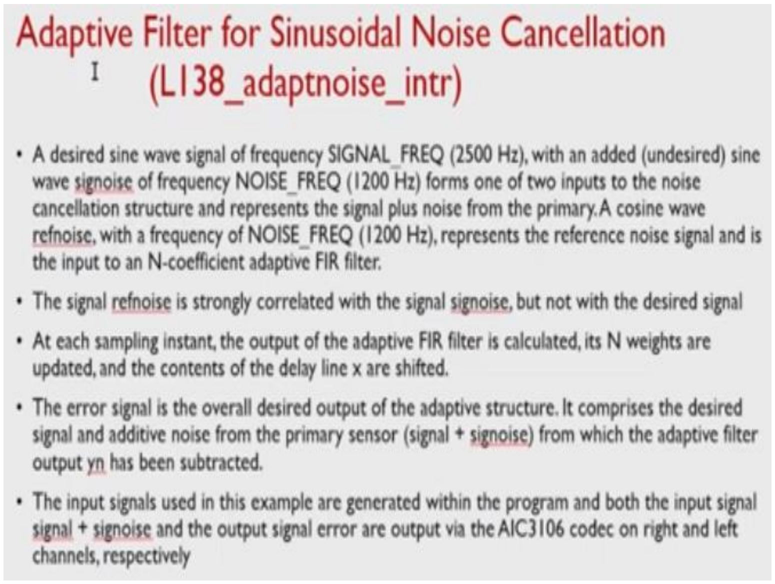
So, we will observe whether we have got these results, go to tools and then we will put the graph single time what we want is iterations is 60 and all of them have been declared as float. So, will it is a single precision 32-bit floating point so the first one will be calling it as desired. So, you are seeing this is a cos what we have it so now again we will plot the other one single time so this is also 60.

Same thing it is a floating point what it has been chosen here floating point and start address is going to be I will call it as y out is what we have want to have the thing so I will give y out. So, you will be seeing that initially it is low and then after that it will settles down so now we will see how the error is going to look like. Again graph single time I will give this also 60 iterations and this also declared as floating point and start address will be error function what I want to plot it.

So, you will be seeing that initially error is very high you are seeing get and then later on it is coming down. So, it is 60 samples is enough to get your desired output that is what it is shown here it is there and then you are settling down so it has whatever the desired signal so what you are getting at the output. So you will be seeing this is the desired signal so what we are trying to achieve initially we had the noise as sine so which comes out as the cosine here.

So, that is the desired signal this dot this is one of the demo what we had the thing so we will stop here and what is the next demo we will be taking in our code composer studio.

(Refer Slide Time: 10:00)



Adaptive Filter for Sinusoidal Noise Cancellation (L138_adaptnoise_intr)

- A desired sine wave signal of frequency SIGNAL_FREQ (2500 Hz), with an added (undesired) sine wave signoise of frequency NOISE_FREQ (1200 Hz) forms one of two inputs to the noise cancellation structure and represents the signal plus noise from the primary. A cosine wave refnoise, with a frequency of NOISE_FREQ (1200 Hz), represents the reference noise signal and is the input to an N-coefficient adaptive FIR filter.
- The signal refnoise is strongly correlated with the signal signoise, but not with the desired signal
- At each sampling instant, the output of the adaptive FIR filter is calculated, its N weights are updated, and the contents of the delay line x are shifted.
- The error signal is the overall desired output of the adaptive structure. It comprises the desired signal and additive noise from the primary sensor (signal + signoise) from which the adaptive filter output yn has been subtracted.
- The input signals used in this example are generated within the program and both the input signal signal + signoise and the output signal error are output via the AIC3106 codec on right and left channels, respectively

So, this is adaptive filter for sinusoidal noise calculation so here it is the adaptive noise interrupt driven what we will be doing it. So, what happens here is the same thing signal frequency is 2500 Hertz what it is chosen with an added undesired sine wave signal noise frequency which is 1200 Hertz. So one of the 2 inputs to our board basically here to the noise cancellation structure and then present the signal plus noise from the primary.

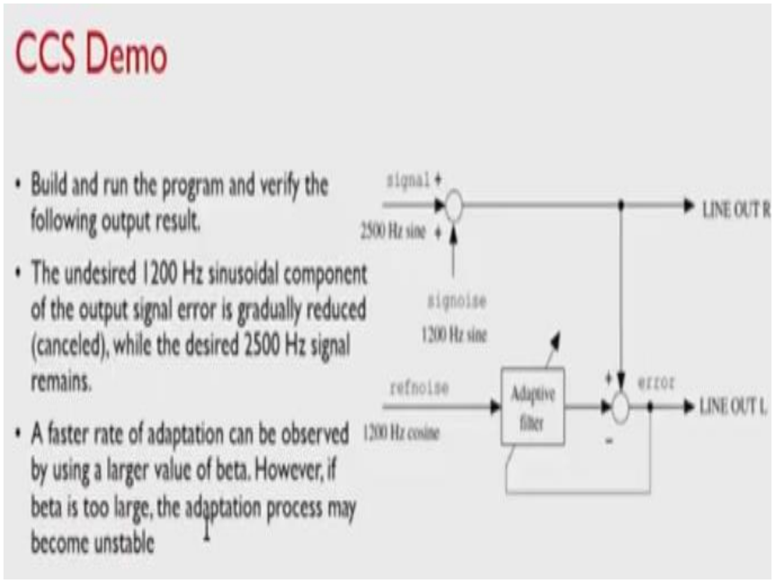
A cosine wave that is reference noise with the frequency of noise frequency that is what we have chosen 1200 Hertz represents the reference noise signal. And is the input to n coefficient adaptive

FIR filter in this case we want to remove this as a noise and then it will be removing the noise from the signal and you will be hearing only 1 tool. So, signal reference noise is strongly correlated with signal noise but not with the desired signal.

At each sampling instant the output of adaptive FIR filter is calculated and its N bytes are updated and the contents of the delay line x are going to be shifted. So, same way will be calculating the error signal is overall desired output of the adaptive structure. So it consists of desired signal and additive noise from the primary sensor that is signal plus signal noise is included from which the adaptive filter output y n has been subtracted.

And our input signal used in this examples are generated with the program and both the input signal plus noise and the output signal error are output via the AIC3106 codec on right and left channels respectively.

(Refer Slide Time: 11:58)



CCS Demo

- Build and run the program and verify the following output result.
- The undesired 1200 Hz sinusoidal component of the output signal error is gradually reduced (canceled), while the desired 2500 Hz signal remains.
- A faster rate of adaptation can be observed by using a larger value of beta. However, if beta is too large, the adaptation process may become unstable

This is how what will be a running the thing so you will be putting the demo here. So, what happens here is signal what we have it, the undesired 1200 hertz sinusoidal component of the output signal error is gradually is going to reduced canceled while the desired 2500 Hertz signal is going to remain. So, to faster rate of adaptation your beta or your mu can be changed it is too large that application process may become unstable.

So, this is your signal and this is the reference noise what you will be generating it and putting it and you will be hearing that line out right and then left what you can hear the error separately. So, if you have a separation of left and right you can independently look at your CRO also and then see that. So what we will do is we will go for the demo what i need is the adaptive noise interrupt driven.

(Video Starts: 13:09)

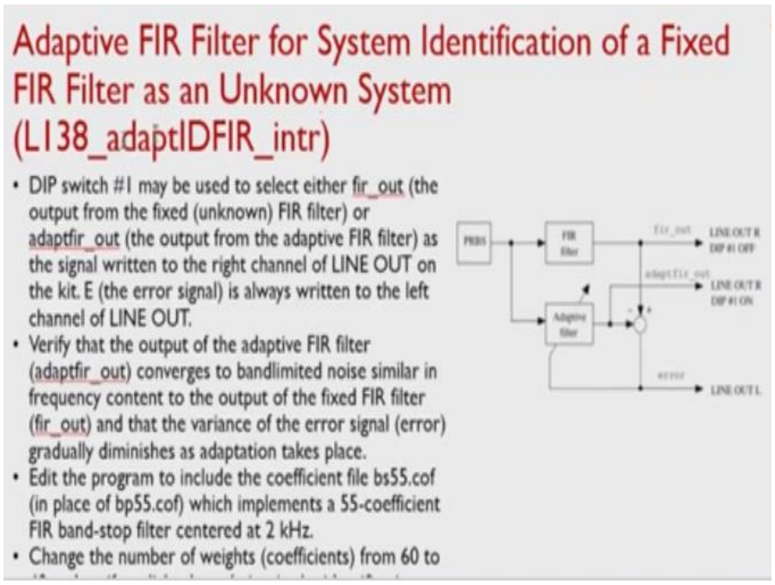
So, I have the thing already built in so if I click on it becomes active debug so we will see that this is a interrupt driven what we have it so we will be getting the data on the right and left channel and then you will be, it uses the delay adaptation what you have it. And in this case we will be using the GPU I will get input from left channel or right channel. So you can use the dip value 0 then it is going to output fixed out. So if it is otherwise if it is 1 it is going to put adaptive out. So, what we will do is?

We will run the thing will compile it debug it and then run. I can close all these things, so I will run the thing initially you may not hear the both the sounds we are hearing the noise that is what the thing. So, by changing the dip switch we can hear the clear sine wave basically so what I will do is I will show you by a graph basically what is the thing we have it out type will have adapt out hopefully it will plot it some I will plot some 200 samples.

So, it is in this case it is showing you a flat, so that is the error what it is coming out of our this thing audio signal. Dips which I have not taken the thing you can experiment it by changing the dip switch so you can hear the other sound.

(Video Ends: 16:01)

(Refer Slide Time: 16:02)


PRBS
FIR filter
Adaptive filter
fir_out
adaptfir_out
error
LINE OUT R DIP #1 OFF
LINE OUT R DIP #1 ON
LINE OUT L

Adaptive FIR Filter for System Identification of a Fixed FIR Filter as an Unknown System (L138_adaptIDFIR_intr)

- DIP switch #1 may be used to select either fir_out (the output from the fixed (unknown) FIR filter) or adaptfir_out (the output from the adaptive FIR filter) as the signal written to the right channel of LINE OUT on the kit. E (the error signal) is always written to the left channel of LINE OUT.
- Verify that the output of the adaptive FIR filter (adaptfir_out) converges to bandlimited noise similar in frequency content to the output of the fixed FIR filter (fir_out) and that the variance of the error signal (error) gradually diminishes as adaptation takes place.
- Edit the program to include the coefficient file bs55.cof (in place of bp55.cof) which implements a 55-coefficient FIR band-stop filter centered at 2 kHz.
- Change the number of weights (coefficients) from 60 to

So, the next one what will continue with the thing is adaptive FIR filter for system identification of a fixed FIR filter as an unknown system. So, what is it here also you have a dip switch basically to select either FIR out the output from the fixed unknown fire filter or adaptive FIR out. So, the outputs from this FIR filter what you will be getting it as output. Return to the right channel of line out on the kit and what you have to do is the error signal is always written to the left channel of line out.

So, that is what you will be one of the thing what you can select but your left channel always error what it will be coming. So, this is an unknown system you are trying to identify by adapting the weight for the FIR filter so it uses a band pass filter that is 55 dot coefficients that is 55 coefficients FIR band stop filter centered at 2 kilohertz. So, you can change the number of weights coefficients from 60 to 40 and verified slight degradation is going to happen in the identification process.

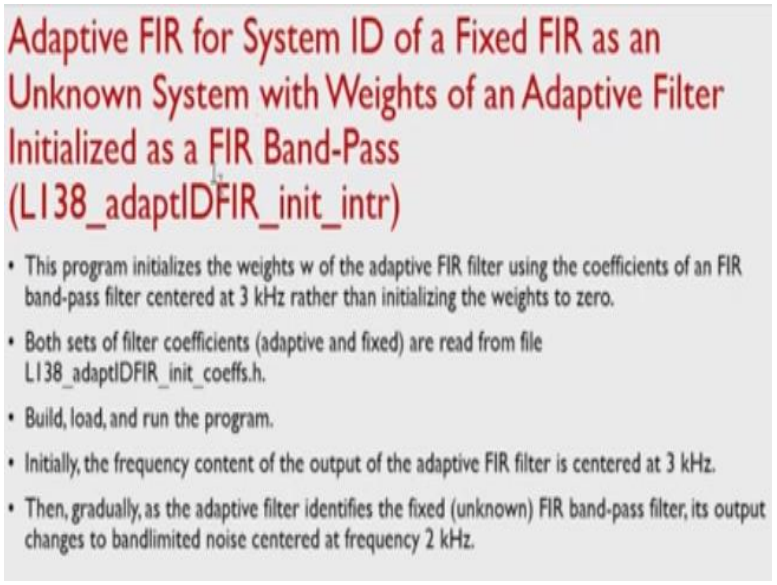
(Video Starts: 17:29)

So, we will see the thing using our CCS, this is the system identification, so you have the system ID interrupt dot C the code. What I will do is? I will show you this code, so here the weight is that is W length is 256 and then the beta value or mu is 10 into E power - 12 the learning rate has been selected. And you will be getting the left sample what you are going to get from the board and then you will be delaying it and then working on it.

So, this is the adaptation what will be that is initialize the adaptive filter weights and delay line. So, w i is also made 0 and delay line is also made 0 and you will be adapting it to the system so we will run this code. You will be hearing the noise I am unable to change the dip switch in this case you can try it yourself and then see that you will be getting the output correctly.

(Video Ends: 19:23)

(Refer Slide Time: 19:26)



Adaptive FIR for System ID of a Fixed FIR as an Unknown System with Weights of an Adaptive Filter Initialized as a FIR Band-Pass (L138_adaptIDFIR_init_intr)

- This program initializes the weights w of the adaptive FIR filter using the coefficients of an FIR band-pass filter centered at 3 kHz rather than initializing the weights to zero.
- Both sets of filter coefficients (adaptive and fixed) are read from file L138_adaptIDFIR_init_coeffs.h.
- Build, load, and run the program.
- Initially, the frequency content of the output of the adaptive FIR filter is centered at 3 kHz.
- Then, gradually, as the adaptive filter identifies the fixed (unknown) FIR band-pass filter, its output changes to bandlimited noise centered at frequency 2 kHz.

So, the next one what we will have is the same thing there are different this thing what is it fixed FIR as an unknown system with weights of an adaptive filter initialized as FIRr band pass filter in this case.

(Video Starts: 19:48)

So, that is IDFIR in it coefficients dot h what it will be using it and then what is coming out of the left sample. This is the adaptive filter output left sample you are having the left sample onto the L DAC what it is going. One thing I forgot to give the thing is it is going to read from the ADC and then this is the DAC output so what I have to provide is one of the sine wave what I have to provide it as an input.

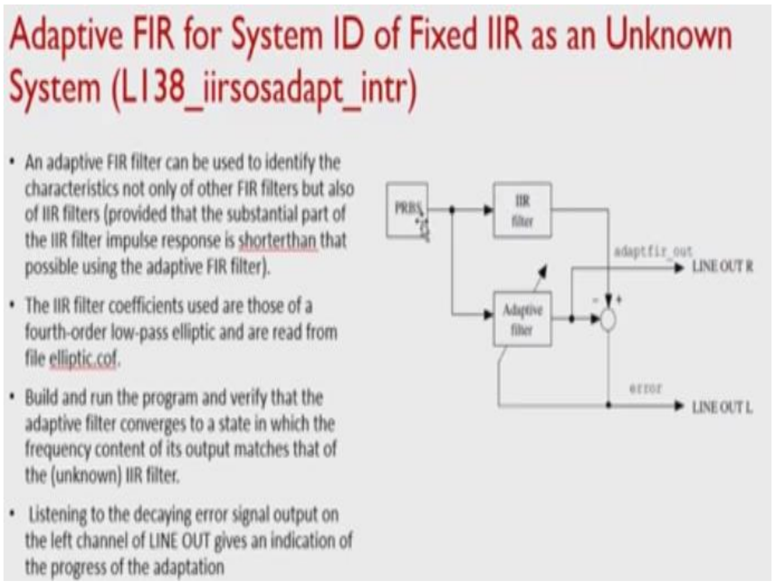
So for running this test signals it is given some of the test signals will be known some of the tones are there. So, I can provide the sine 1000 this is a sound input to the system. We will be seeing that it is continuously running so we will see whether the left sample has come or not in this case. So, we will output it and then check sometimes what happens whatever input I have taken the thing whether I have got it out or not this way what you will be checking the thing so we will recompile it. So, this is the input which is coming out. So, that is how you will be checking whether you have got the input into your system or not.

So, the input is coming so because I am taking a left sample and then I am outputting it. Whether it has adapted or not one has to look at it because there is a problem at the, so this is the noise what you have it so you are outputting the noise in this case basically. So, you will hear the noise anyway already you have heard it so I am showing that what is the output happening. So, the noise is coming out of the thing so once it is, will push this little down so that I can output my either filter output what we can output it here this is of you will be changing the thing.

So, FIR out what I want to put the thing so we will do debug and then see whether I am getting the so as you can see I have a faced this thing error in the thing. So, unless I solve this there is no point in line 27 what it says so I have to come out of the debug mode. I will go to the thing, as you can see I have not removed it was 0 what it was taking so there was a mismatch in the thing so I have to correct the error and then recompile it. So, when there are errors do not go and then load it onto the board as you can see that there is little error in whatever the FIR filter what it is getting adapted to that is a filter coefficients trying to remove this noise and then it should I should be getting clean sine wave. But still there is an error so this order is may not be sufficient for it so you may have to go to higher order to remove the noise from the thing. So, it is a little bit reduced but it is still getting overlapped so you have to fine tune and then work on it.

(Video Ends: 25:35)

(Refer Slide Time: 25:41)



Adaptive FIR for System ID of Fixed IIR as an Unknown System (L138_iirsosadapt_intr)

- An adaptive FIR filter can be used to identify the characteristics not only of other FIR filters but also of IIR filters (provided that the substantial part of the IIR filter impulse response is shorterthan that possible using the adaptive FIR filter).
- The IIR filter coefficients used are those of a fourth-order low-pass elliptic and are read from file elliptic.cof.
- Build and run the program and verify that the adaptive filter converges to a state in which the frequency content of its output matches that of the (unknown) IIR filter.
- Listening to the decaying error signal output on the left channel of LINE OUT gives an indication of the progress of the adaptation

so what is the next application what we will see using this thing in the board that is a ID of fixed IIR as an unknown system in this case so earlier we have seen it as FIR so this is the pseudorandom generation what you are doing it IIR filter through which you are passing and then looking into the thing. So, here it is used as elliptic dot coefficient what it is going to be used for this purpose the filter length that is because we know that it has to be one of the thing either I can use a Chebyshev or Butterworth or Elliptic.

Elliptic has the minimum what we have seen in the class is that it has both ripple in the past bend and then stop band so which has a very minimum order for the thing. So, we will go and then see whether we can work on this so this is IIR so I have adaptive IIR here so we will make this is an active debug directory and then I will close the C file and open IIR adaptation.

(Video Starts: 26:55)

So, you will be seeing that elliptic dot coefficient what it needs a thing. When I open it so you will be seeing that number of sections what it has is 3 that is numerator sections are 2 basically as you can see the thing and then denominator what you have it. So, number of sections what we have it is that is second order this is the numerator the denominator of the first section and numerator and then denominator of the last section what we have it.

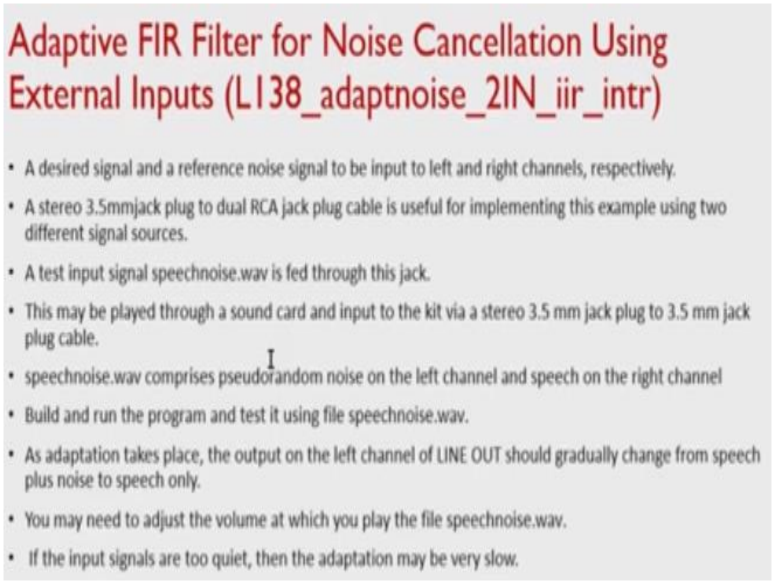
That is number of sections what it has is 3D, order is 2 sections what we have it that is fourth order filters first is the second order second stage is the second order section what we have it. So, these are the B coefficients and these are the A coefficients what we have it. So, when we run the thing we will build the thing here adaptation rate is chosen as 1 E power - 13 basically as you can see in the code value.

So, we are passing the same sine wave what we have kept it so it is continuously running, so we will see that how this is going to react to the noise and then input signal. You have the adaptive FIR out and adaptive FIR both of them are output both right and left what you are seeing the thing. Almost it is getting the noise it is not completely eliminated with fourth order what you are seeing it.

So, you are using the update adaptive FIR filter coefficients in this case delay line what you are doing it. So, still the noise is left out so it needs further training and then the order of the filter is very low in this case so it is unable to adapt itself to the noise.

(Video Ends: 29:29)

(Refer Slide Time: 29:33)

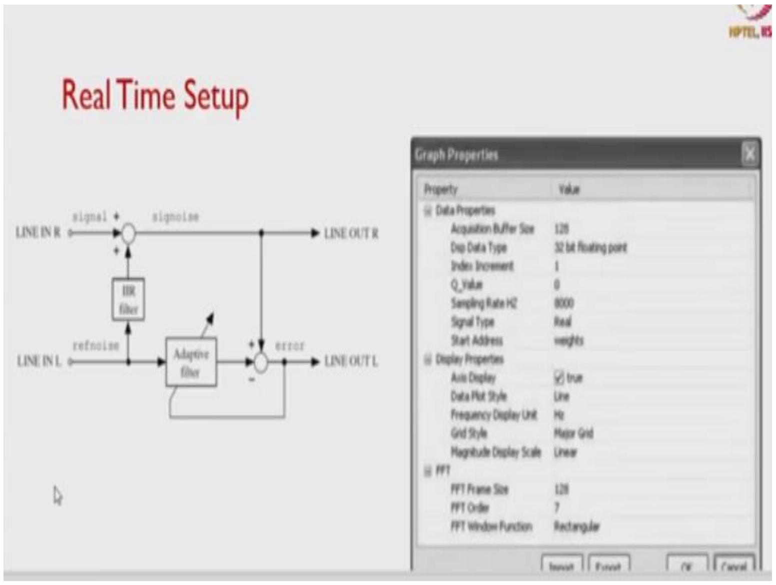


Adaptive FIR Filter for Noise Cancellation Using External Inputs (L138_adaptnoise_2IN_iir_intr)

- A desired signal and a reference noise signal to be input to left and right channels, respectively.
- A stereo 3.5mmjack plug to dual RCA jack plug cable is useful for implementing this example using two different signal sources.
- A test input signal speechnoise.wav is fed through this jack.
- This may be played through a sound card and input to the kit via a stereo 3.5 mm jack plug to 3.5 mm jack plug cable.
- speechnoise.wav comprises pseudorandom noise on the left channel and speech on the right channel
- Build and run the program and test it using file speechnoise.wav.
- As adaptation takes place, the output on the left channel of LINE OUT should gradually change from speech plus noise to speech only.
- You may need to adjust the volume at which you play the file speechnoise.wav.
- If the input signals are too quiet, then the adaptation may be very slow.

So, this example because I said it is an audio a wave files what we generated in the last class. Because it needs a longer time to adapt itself so it was taking longer time that is why I shown this demo in the last class.

(Refer Slide Time: 29:52)



Real Time Setup

Property	Value
Data Properties	
Acquisition Buffer Size	128
Dsp Data Type	32 bit floating point
Index Increment	1
Q_Value	0
Sampling Rate HZ	8000
Signal Type	Real
Start Address	weights
Display Properties	
Axis Display	✓ true
Data Plot Style	Line
Frequency Display Unit	Hz
Grid Style	Major Grid
Magnitude Display Scale	Linear
FFT	
FFT Frame Size	128
FFT Order	7
FFT Window Function	Rectangular

So, this completes the adaptive filter up what I will say for different applications how we are going to run it using the board in real time. So, whatever data how a system ID identification it happened and how the noise is going to be canceled and it can be non-real time or real time what you can give the thing. So, one more way of doing it is you can give separately the noise in one channel here it is mixed and then sent it as an input to the system.

So, one of them can be mic input the other one can be line input what you can give and then mix and match. So, that is what I was telling so you can play around this board the way you want to give the input and then what you want to hear at the output. So, thank you for listening to this lab.