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

**Module No # 08**
**Lecture No # 42**
**Adaptive Filters (MATLAB)**

Welcome back to real time digital signal processing lab basically.

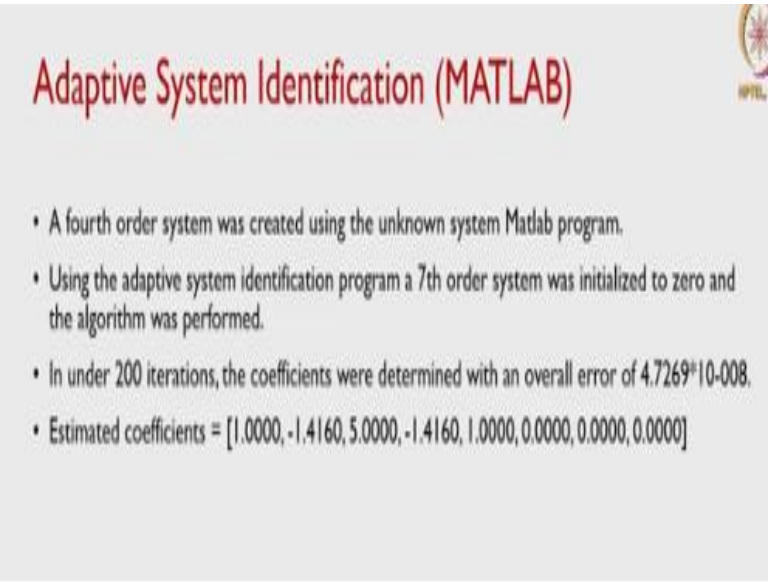
**(Refer Slide Time: 00:32)**


So we were discussing about the adaptive filters in the last class. So today we will see how we can use MATLAB first to generate different kinds of adaptive filters for different applications.

**(Refer Slide Time: 00:46)**


So this is what we discussed 2 of the adaptive filters in the last class.

**(Refer Slide Time: 00:52)**




And today we will see first how we can use the adaptive system identification; so here it is a fourth order system was created using the unknown system in the MATLAB program. And using the adaptive system identification program a seventh order system was initialized to zero initially and the algorithm was performed. So in under, 200 iterations the coefficients were determined with an overall error of 4.7269 to the power of 10 power - 8 what it shows. And the estimated coefficients values what will be seeing it here.

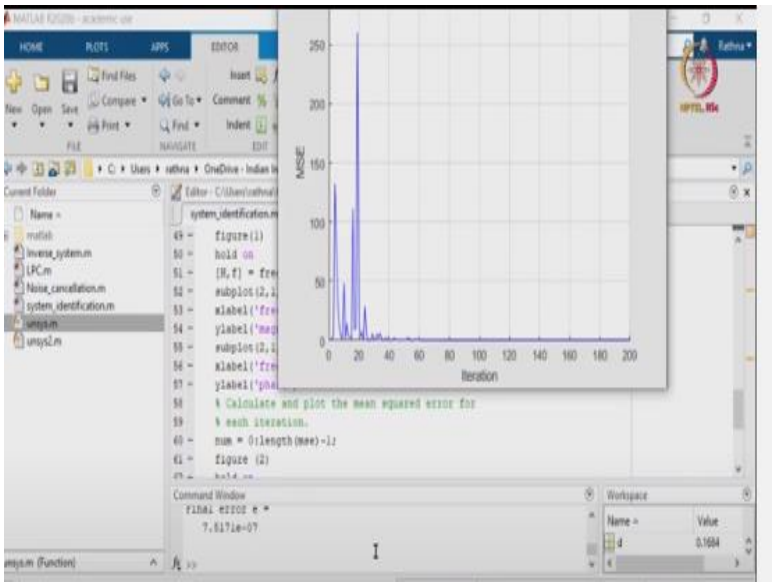
**(Video Starts: 01:43)**

So we will go back to the MATLAB and then see whether we are going to get these results, we will see the system identification so this is one of the available codes, to show the application so you can use the book codes to do the thing. This is written by Thomas Drumright it is you can see it is in 97 what it is written, and still it can be used. So what the program does is it initializes, input vector to 0, and then you have the mu is set to 0.1 that is convergence factor, and then the length it is taken as 8 that is filter length.

And then now the data value that is iteration number of iterations what it is selected is 200, and then input is a random number that is input white noise what it has been selected, and then mean square error is set to 0. And these are the initial input, and initial coefficient, filter coefficients basically the length of it, what you will be making it 0 w, and you will be start calculating the that is unknown system which is given as un-sys.m in this case.

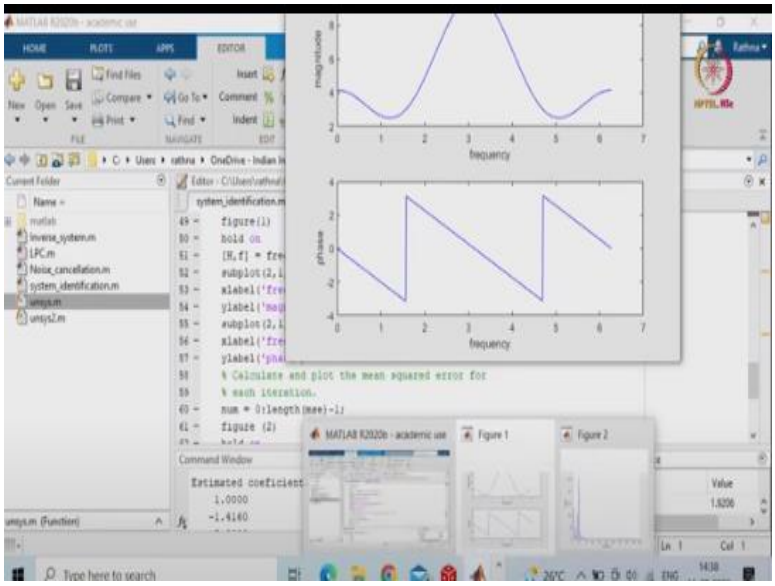
So here you will be seeing this is the system coefficients what it is defined and what the, your system this thing identification has to converge to this. So w will be starting weight vector will be 0 and output will also be 0. So then you will be going on and then this is the desired signal what you have it, and then error function you are trying to minimize that is d – y; y what you are considering with the x values and w is going to be updated as usual. And then we will see that how it is going to converge to the required system and what is the value we will be getting it so.

**(Refer Slide Time: 03:55)**



As you can see there will be little cut off if I increase the size of the font you will be losing some of the things on the top actually. So you will be seeing mean square error in between what it is high and then it is settling down to almost 200, it is almost it has become we can say it as nearer to 0. So that is what it shows for 200 iterations what it will be settling down and you will be seeing the values, so we said that you.

**(Refer Slide Time: 04:40)**



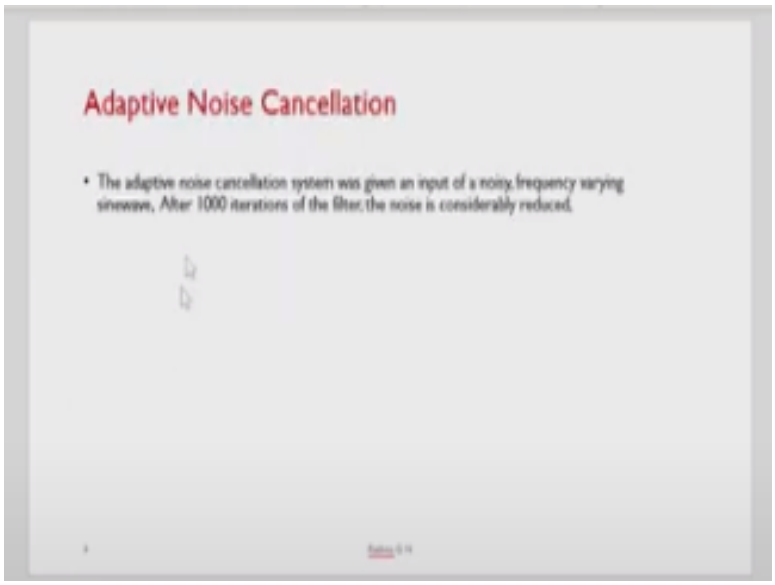

Will be seeing that this is the magnitude and this is the frequency response of the filter and you are seeing that whatever in the past band you are seeing this has a linear phase basically, this is frequency versus phase what it has been plotted. So what we can do is we can close these 2, so that we can go back and then see what was the initial value it was given, and then how the output estimated coefficients what we w that is system identification what you have done the thing. So it is given 1 - 1.416, 5 - 1.416 and then 1.

So you will be seeing that this is what it has converged to, so you are able to identify what system you have given as input it is a band pass filter in this case, so the coefficient has merged with this.

**(Video Ends: 05:50)**

So then going on to the next this thing this is the first application what we have seen it.

**(Refer Slide Time: 05:58)**



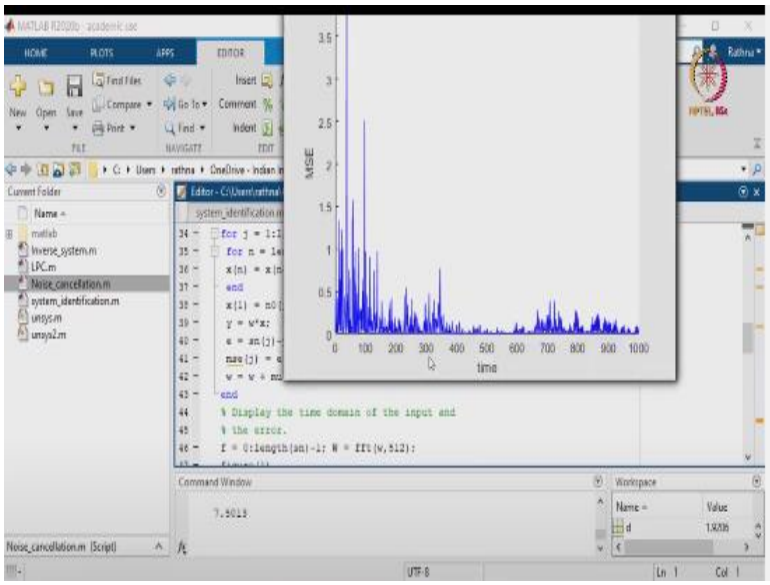

So the next one what we will see is in the MATLAB how we can do the adaptive noise cancellation. So our adaptive noise cancellation system was given an input of a noisy frequency varying that is sinewave after 1000 iteration you will be of the filter the noise is considerably reduced, that is what what it is written.

**(Video Starts: 06:22)**

So we will go back and then see noise cancellation in this case. So we will open this so this is adaptive noise cancellation so you will be seeing that mu is given as .01. So you can definitely expect that number of iterations is going to be more, here it is running up to 1000 iterations. And then you have been given the input white noises with 0.7 magnitude what it has been chosen. And you will be generating the sine wave along with random noise and then we will see how the noise is going to get cancelled.

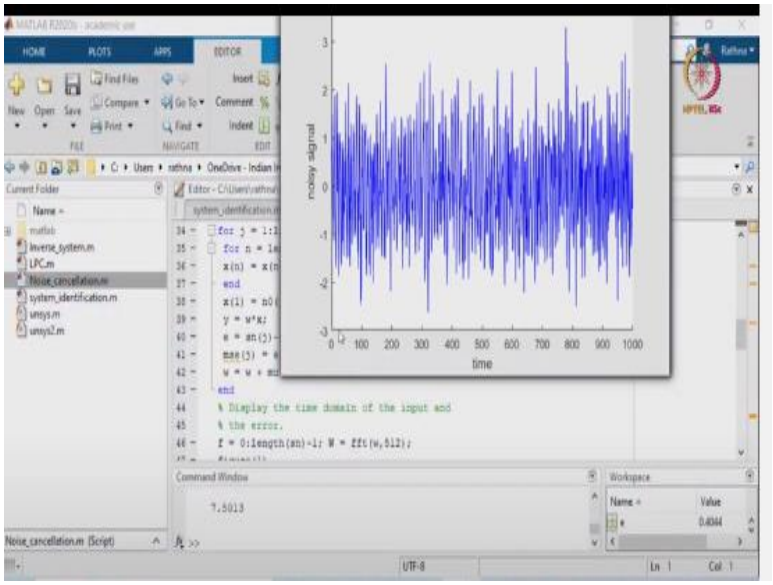
This is built in for function sine wave generation and then you have your error function also which is going to be reduced. So the rest of the thing remains the same thing, so you will be displaying what is the error even the wait function you start from 0. And then you will be trying to adjust your weights based on your application, so we will run the thing.

**(Refer Slide Time: 07:34)**



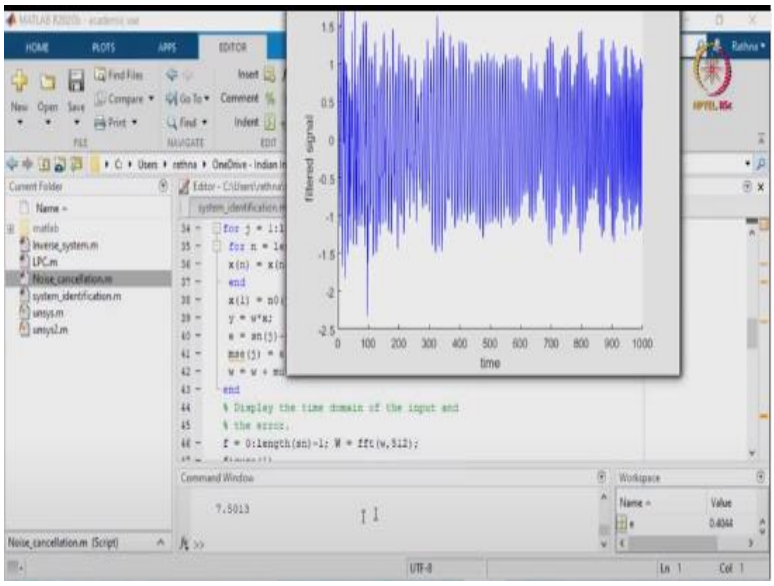

So you will be seeing that so many wave forms have got generated.

**(Refer Slide Time: 07:45)**



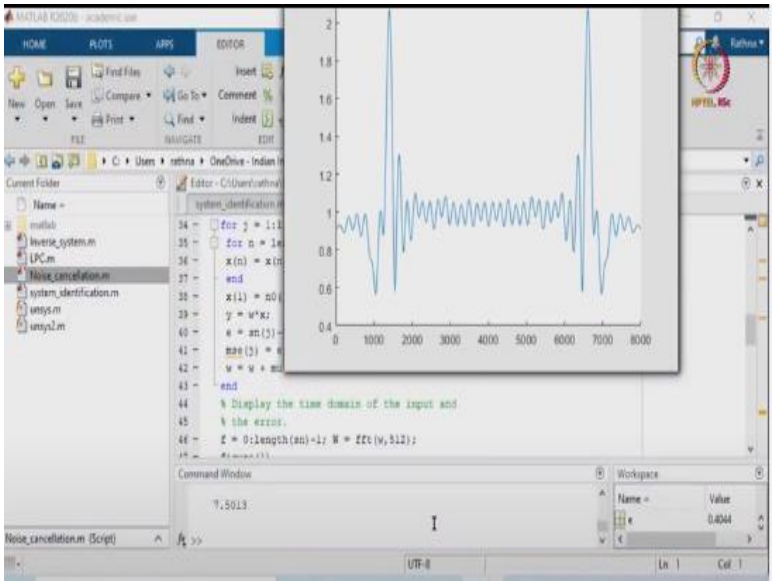

So the figure 1 shows the, in the time domain this is time and this is the noisy signal what we have it and we will see the next figure.

**(Refer Slide Time: 07:59)**



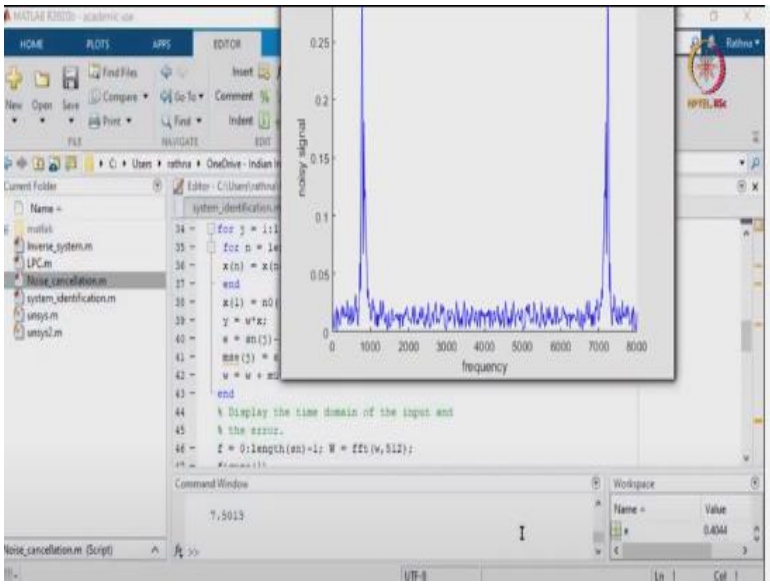

This is figure 2 this is a time domain how does it look like a filter signal in the time domain.

**(Refer Slide Time: 08:12)**



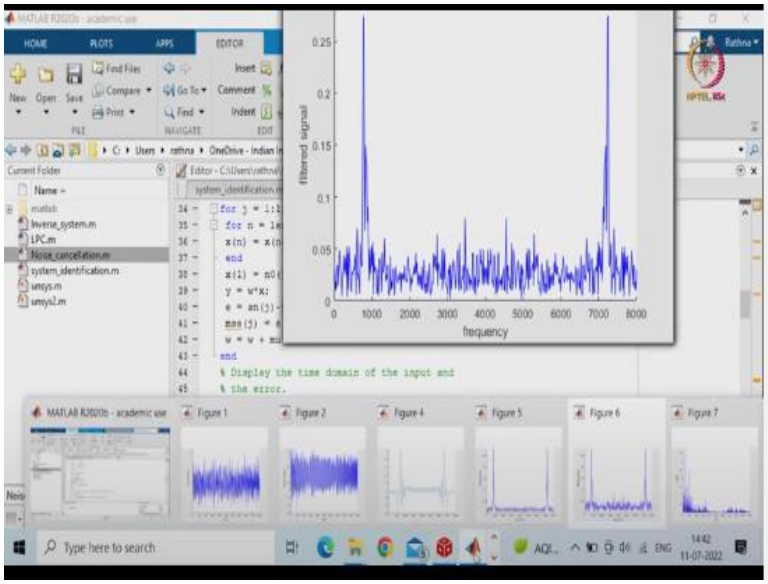
So the next one is what we have is figure 4 shows that this is the sine wave which was generated along with the noise basically. So this is in the frequency domain as you can see but naming has not been done here. And this is the magnitude of it what you will be seeing on the y axis and x axis is in the frequency domain response.

**(Refer Slide Time: 08:39)**



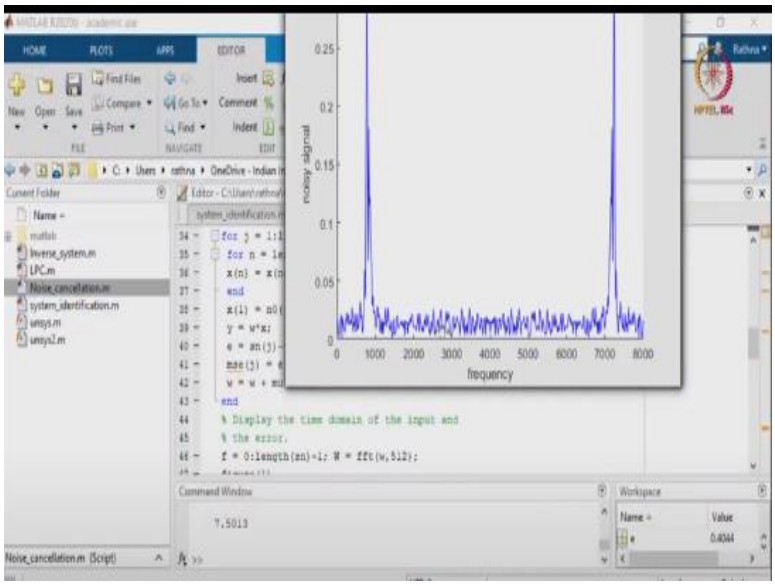

And then the fifth one how the noise is introduced with this is the noisy signal in the frequency domain along with that is sine wave you have the noise present also.

**(Refer Slide Time: 09:00)**



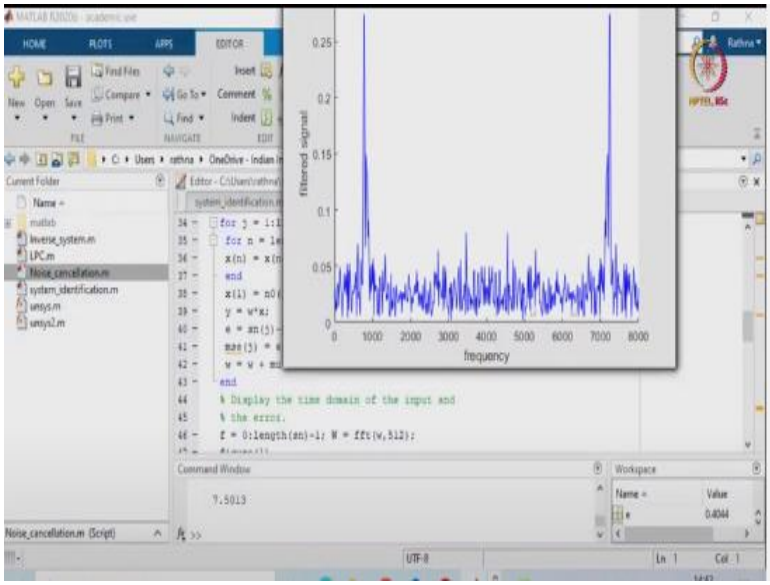

So now we will see figure 6 you will be seeing that this is a frequency so you are seeing that this shows I think this filter signal and this has got little bit exchange as you can see.

**(Refer Slide Time: 09:13)**



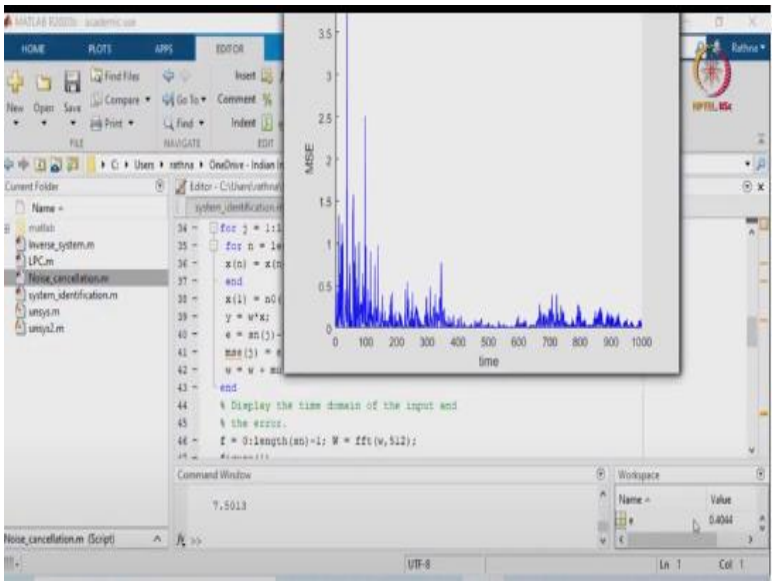

That here the noise is less after filtering so the naming is a little bit interchange.

**(Refer Slide Time: 09:20)**




That this, is the noisy signal and then the filter signal is little less sorry about it I done the mistake in y-axis.

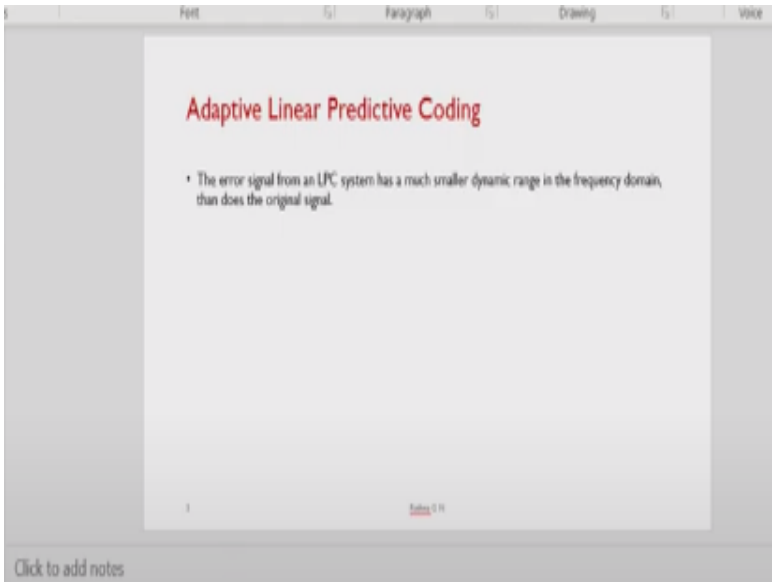
**(Refer Slide Time: 09:34)**




So; the next one is you will be seeing the error that is time versus mean square error. So you are seeing initially which was high and then which it has got considerably reduced. So this is how you would be doing the noise cancellation so using adaptive filter so the next one what we will see is after the noise cancellation.

**(Video Ends: 10:02)**

**(Refer Slide Time: 10:11)**




How your linear predictive coding is going to work, that is error signal from the linear predictive system basically what we have it so, that is system has a much smaller dynamic range in the

frequency domain than does the original signal. So how this can be implemented what we looking using the MATLAB.

**(Video Starts: 10:35)**

So we will run the this thing what it has is you will be seeing that for the prediction you will be seeing that convergence trend is very small, and the filter length chosen is 100 in this case; and the number of iterations as you can see it has gone up to 5000. So you will be seeing that the variation in iterations what it has been chosen from one application to the other application it is increasing, because it takes more time to predict the thing.

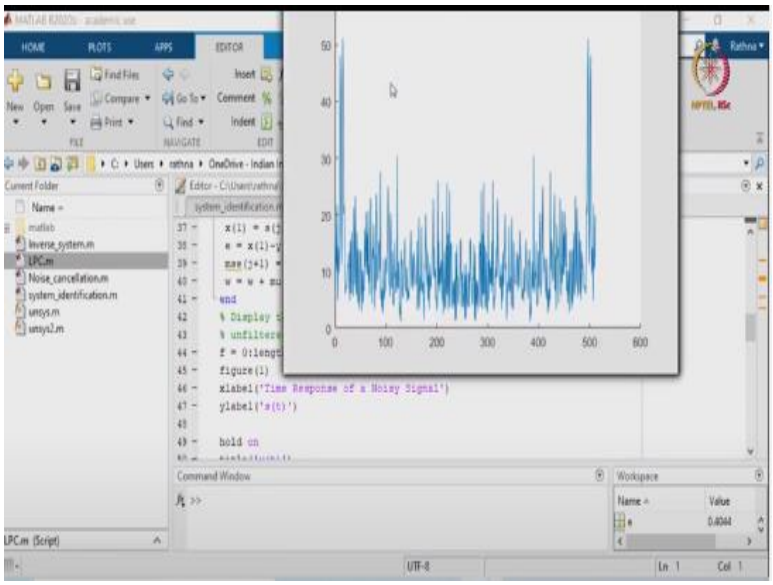
That is what it says this is a linear predictive coding what it is going to be. It recursively calculated the coefficients of an unknown system, the value mu this thing specified the rate of convergence of the adaptive algorithm, Must be less that the smallest Eigenvalues of the unknown transfer function for this adaptive filter to be properly conditioned. So for that you have to choose the lesser one.

And length is the order of the filter and then LIN is number of iterations to take place, x is our input signal to the system n 0, n 1 are the two independent noise sources in this case. So this is a sign which has this thing you are seeing that there are three sine waves what you are seeing in here with different frequencies what you have. This is your signal s 1 what it has been given and then your n naught, is the random noise white noise which is getting added with the signal.

Apply the adaptive algorithm what you will be doing it and x and w are the initial input as well as the filter coefficients. And you will be running the code for your this thing what is it multiple times. So you will be monitoring the mean square error what you will be setting it the j + 1 will be set to the previous error. And then your weight function is given with w + whatever the previous value with respect to updated, this is how your w vector is getting updated.

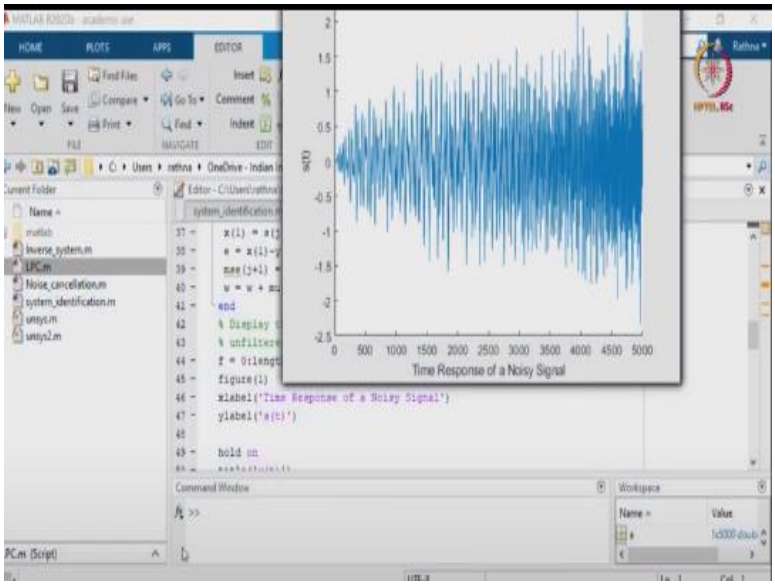
And you will be seeing that these are the displays what you will be looking at it. So we will run and then see the algorithm how it is going to behave.

**(Refer Slide Time: 13:22)**



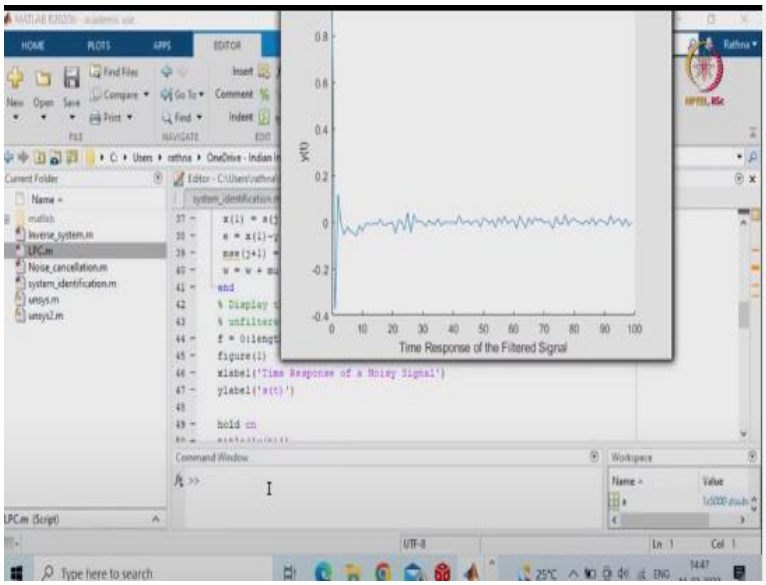
So we will be seeing that what all the figures.

**(Refer Slide Time: 13:33)**



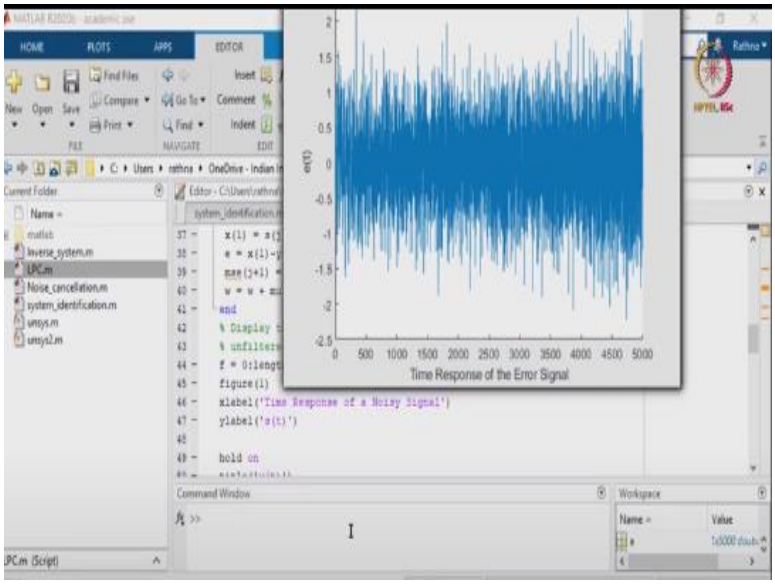

This is our input wave here also so you will be seeing that s of t represents our in the time domain input wave. That is time response of a noisy signal what we are considering it a.

**(Refer Slide Time: 13:46)**



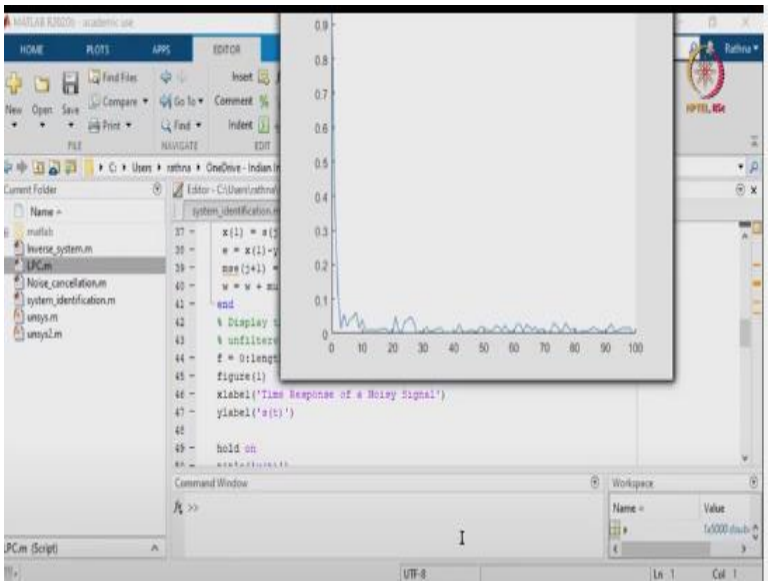

And then figure 2 will be giving a filtered signal basically. How it is represented that is y of t in terms of time domain thing.

**(Refer Slide Time: 13:59)**



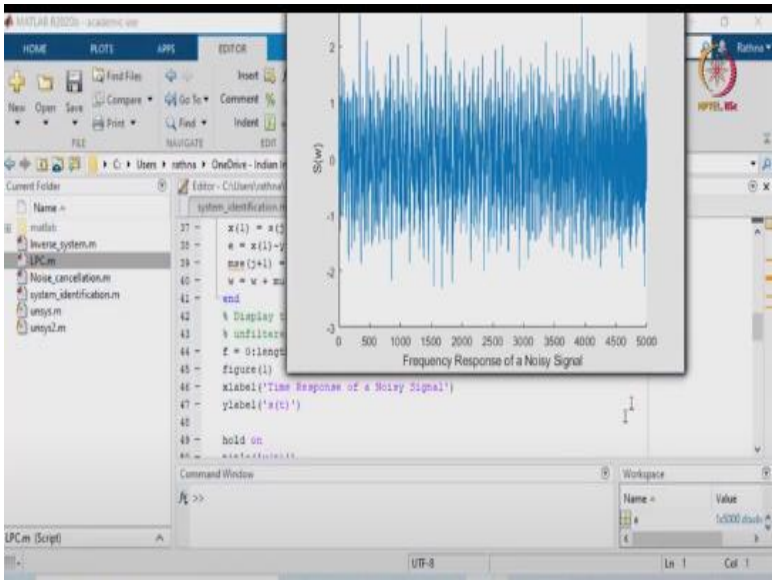

And then figure 3 will represent time response of the error signal that is e of t what you are looking at random white noise how it is going to be.

**(Refer Slide Time: 14:10)**



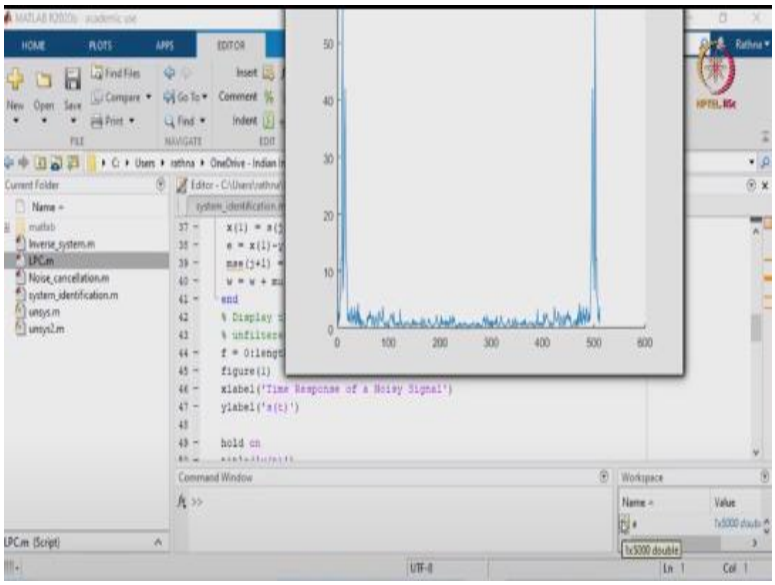
And then next one is figure shows that how the error signal is reduced, somewhat at 100 it is coming down.

**(Refer Slide Time: 14:25)**



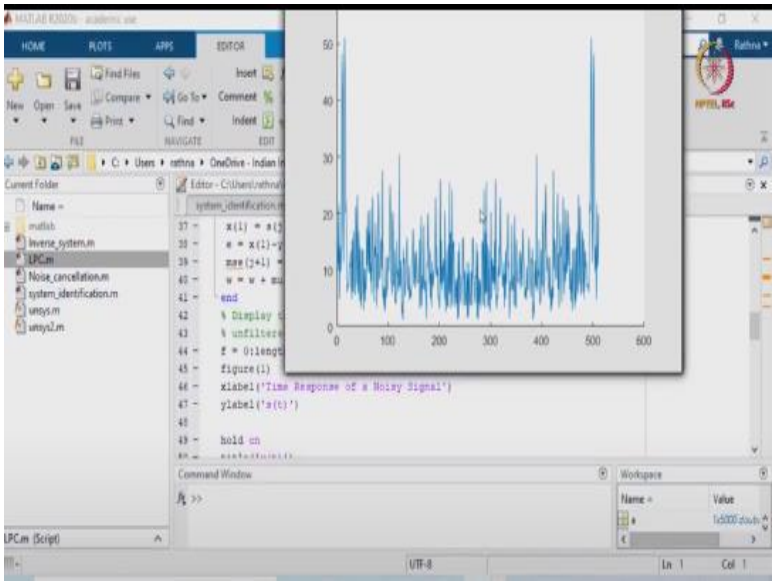

And then next one will be frequency response of a noisy signal in the frequency domain, what you are seeing it; s of w what you are putting it with respect to weight vector what you are seeing the thing.

**(Refer Slide Time: 14:40)**



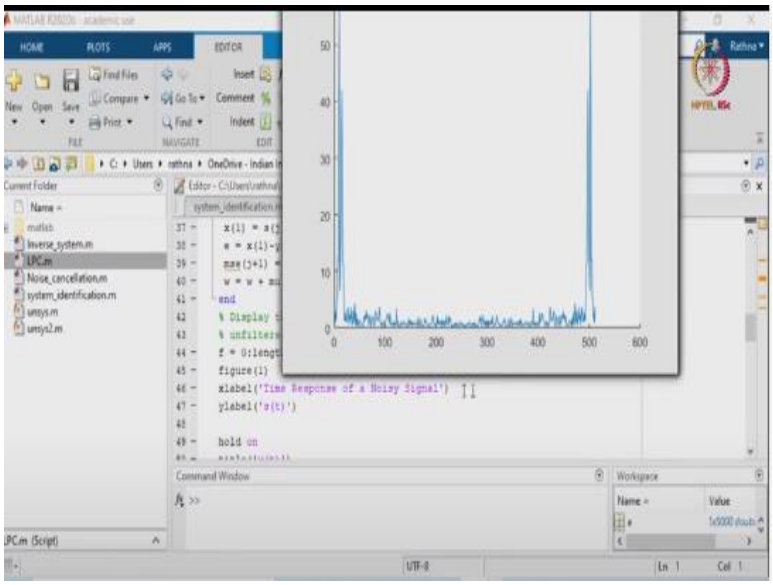

And then 6 will be whatever sign wave and then this is according to this we have done at 1000 iterations. So you will be seeing that approximately we are able to achieve the, this thing error signal. How it is coming down and then your sine waves are what it is seen at the output it has adapted.

**(Refer Slide Time: 15:04)**



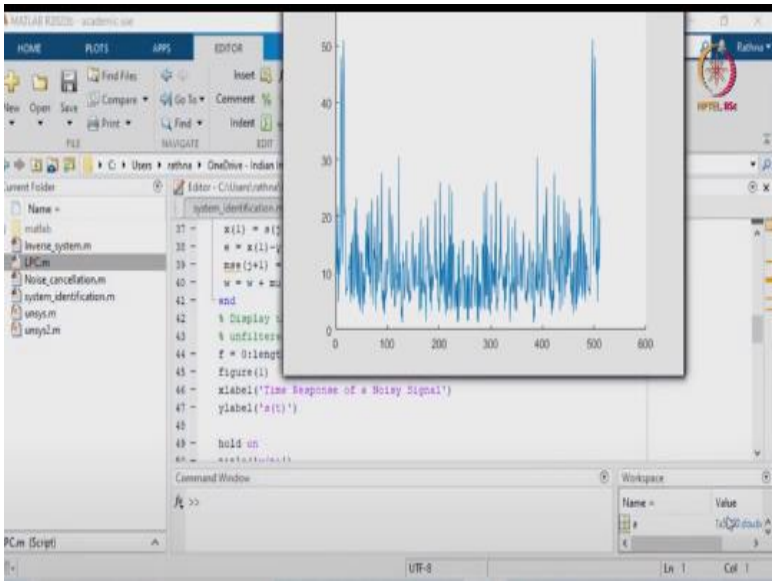

And this will be showing errors along with the in the frequency domain how the error is much more compared to the previous one which has got filtered out.

**(Refer Slide Time: 15:20)**




So this is what after filtering.

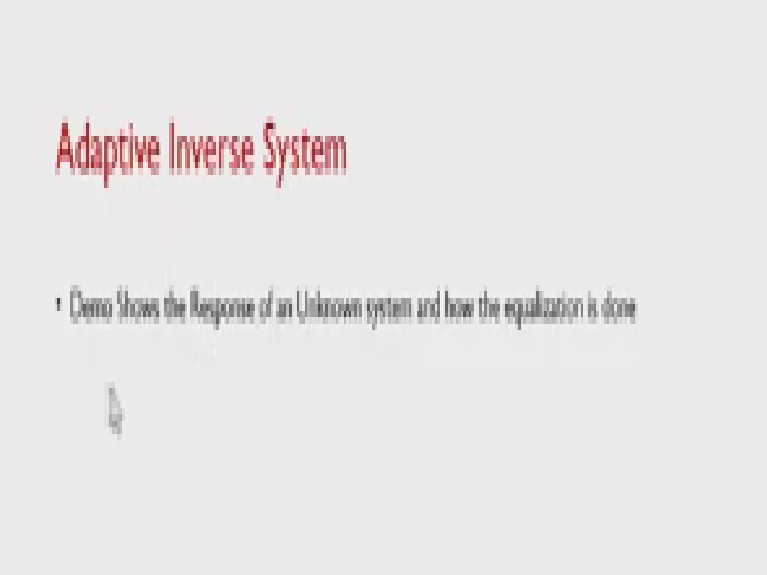
**(Refer Slide Time: 15:25)**




And this is the before filtering how the noise was present in the input signal. So it is coding how LPC coding has been done to receive the, our input basically correctly that is although there is a noise channel noise has been added. As a white noise and the receipt signal is exactly what we have receipt; and coming to the next application.

**(Video Ends: 15:56)**

**(Refer Slide Time: 16:02)**




So what we will be seeing with the MATLAB is adaptive inverse system so that is a demo will be showing the response of an unknown system and how the equalization can be done, with the inverse system. So this is what the demo will look at it using MATLAB.

**(Video Starts: 16:21)**

So we have the inverse system here so we will run the thing, that is calculated the coefficients of an unknown system mu specify the rate of convergence all the same thing. So what we have is here there is an unknown system 2 what it has been chosen for this application. So which has you will be seeing that this is the weight system coefficients what it is modified compared to the other unknown system dot m.

So this is the unknown system and then the output will be identifying the system basically it will be the inverse system how you can do the equalization to get the original system. So we will run the code, sorry because that is a function which is called from here, so that is the reason why what you are getting the error it is telling that it does not have input arguments because I had to call this function from other source. So I would not be able to run it directly so you will be seeing that will run from the inverse system.

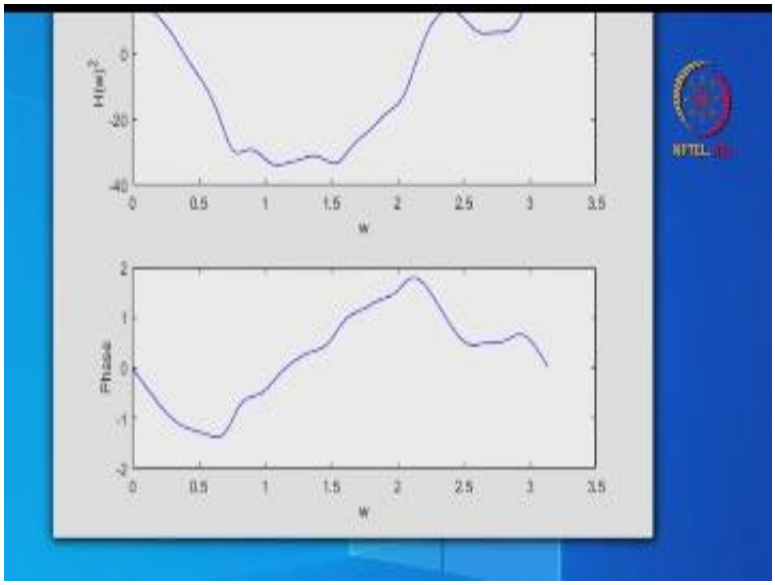
So here what it shows is, we will get the detail of it, and then come back to show the output, here mu is chosen 0.005 and length of the filter chosen as 20 and the iteration is 5000 in this case. So you will be seeing that initial input vector what you have it and this is a Gaussian white noise

what it has been added to our noisy signal. And then you will be calculating your sine wave what you have it and the, you will be adapting to your unknown system basically.

Inverse of the system what I have to get it and then your weight vector as usual will be updated in this manner and then you will be performing mu what is it frequency that is filtering what you will do it. And this is the filter what it is being used you will be calling it is the IIR response of the filter and then f is the frequency what it is being calculated. You will be using the frequency z with your weight, vector w basically what it has.

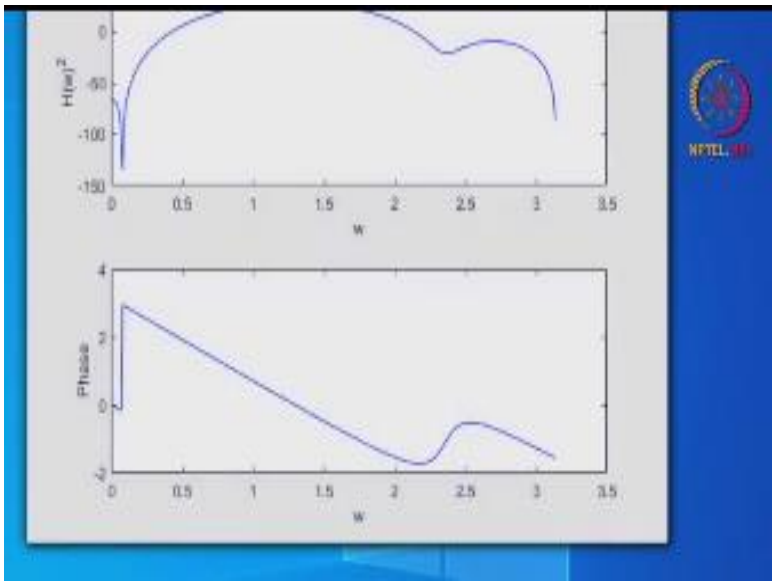
And then 1 and then 5, 12 what will be getting it and then you can do subplot and then it will be hold on and then rest of the thing what it will be giving you so this is the filter one has to identify the thing. So you will be seeing the values what error you will be seeing that which is equal to 0.2848 what it has come down to and you will be seeing that what is your answer for the thing what it has identified the unknown system that is inverse system.

**(Refer Slide Time: 19:56)**



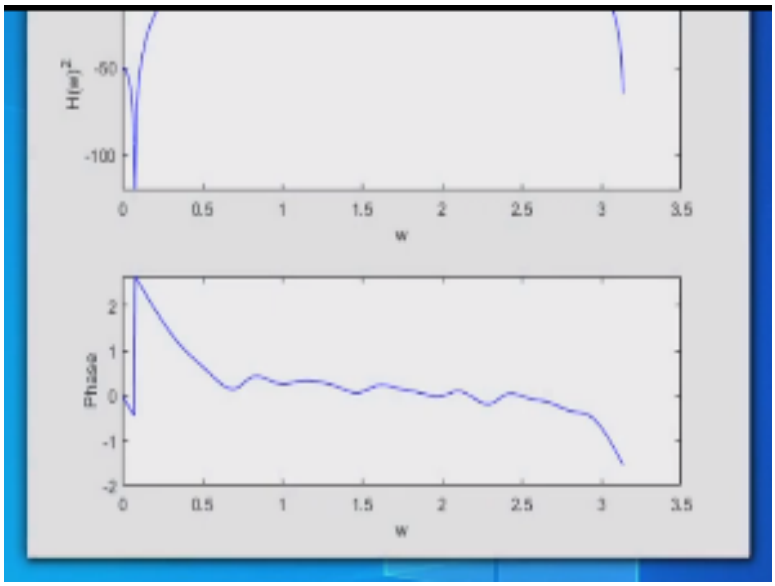

So this is the just hold on for a while I think or I will show from here you will be seeing that this is your inverse filter what it is working.

**(Refer Slide Time: 20:04)**



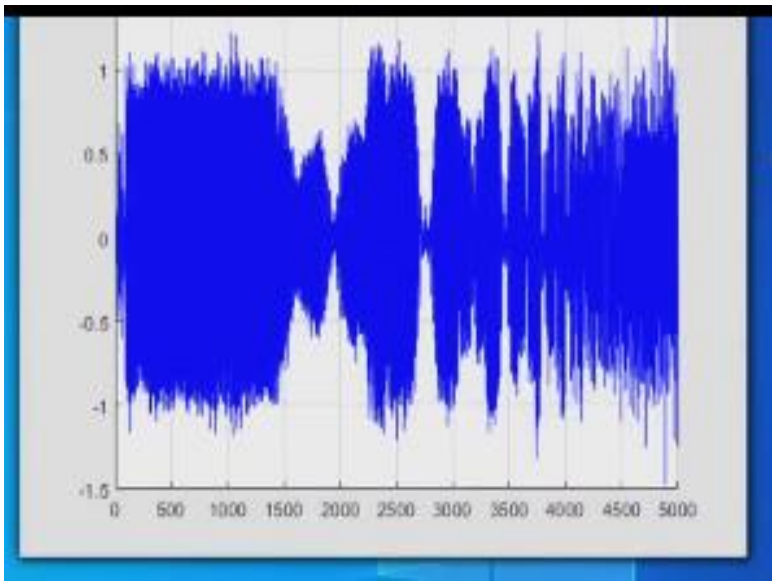

And this is the response you will be seeing that of the original this thing filter what we have it.

**(Refer Slide Time: 20:13)**



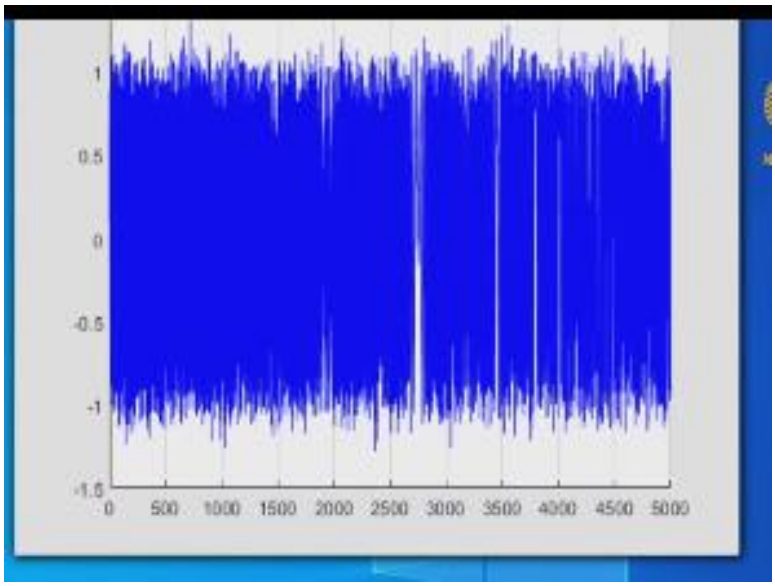

And how you can that is equalized from the inverse system to the output level what you will be seeing.

**(Refer Slide Time: 20:22)**




And this is the noise which is getting added to your system this is in the frequency domain.

**(Refer Slide Time: 20:22)**




And this is also in the frequency domain how your error signal is going to, that is noise is going to look at it. So from the noisy signal you can see that after your system has the, what is it input basically what you have given the thing the response of the system what you are looking at it. So you have identified from the noise also and then you are able to this was the original, and you are able to correct it to the original and you are seeing your phase how does it look like here in the original this is the phase of the system, and in the recovered signal this is the phase of the signal. So this is how you will be applying your different applications; that is adaptive filter in different fields of it, so that is the application of our adaptive filter. So now what we will look at it is will

go to equalizer what we discussed in the theory one of the equalizer developed by our own student what we have seen in the last class. We will see that how the MATLAB is has designed an equalizer.

So we will see or this is my equalizer dot m anybody can download from the MATLAB website and then what all it is a new my equalizer, or raises the existing singleton what it says. And h is the equalizer returns the handle to a new equalizer or the handle to the existing singleton, and it will be calling this functions basically and this is a GUI. So it has other thing what you can see guide or whatever one of the thing what you can use it.

So this was modified in 2008 which was available some of when you download these things one has to be careful. So wherever wave read has been used in the latest MATLAB version it has been modified as audio read. So if you, if it gives an error one has to modify it to audio read and then my play, has to be changed based on the whatever 2020 b supports. So you will be using it to audio play instead of my play, so these are the modifications one has to do it. So we will run this file.

So you are seeing that in this case by looking at it you must be able to identify how many control that is gain systems have been given it is a 5 band equalizer basically what you can see it. And you are seeing that what they call it as c 1, c 2, c 3, c 4, and c 5 are the 5 bands. And what are the frequencies, this is a low frequency and then this is band 4 to 8 kilo hertz, and this is 9 to 13 kilohertz what you can see, and this is 13 to 17 kilohertz.

And this is the high frequency as you can see that the frequency has gone up to 44.1 kilohertz what it has been chosen for the audio and you have the 5 bands what it has been chosen. So we will see that we have all of them have been placed your knobs in the center of it. And this is the response what it is showing I think I may have to reduce the thing and then come back because all the control is at the bottom which I am unable to see the thing how we can do that.

So by changing the resolution you will get back your what I put it is the complete GUI will be shown otherwise some of it as you were seeing it the bottom portion was getting blocked. So now we will see that load the of whatever wave file which has come with the default what I will be taking it they call it as bach 2. And then if you want to plot you can plot this is how the after

equalization what you will be seeing it, and we will vary and then see how it is going to look like.

So we will play the thing now with the all of them are in equal positions. Only this is the length of it, now what we will do is we will bring down the thing and then we will see plotting of it. So what you are seeing is almost low passes what is it? It is cut off, the low frequency has come down to -16 db. So this is the db setting and these are all 0 db, so if you want you can play and then hear to this.

So now what we will do is whether we can go up to positive, from 0 db we will increase the db to 16 db, now we will try to plot it so you are seeing that this is a low pass filter and rest of them this is at how much 16 db as you can see the thing. Magnitude response so it is a low pass filter what it is happening rest of them are not there, so we will try to play. So you can see how loud it became because more of low pass filter frequencies what it contains.

So now we will see the thing by increasing this to positive side, so we will plot the thing so you are seeing that it is going in steps. So this I have given 10 db see to these frequencies whatever you are seeing it up to 20 kilohertz as I said 20.05 will give you 44.1 kilohertz is the sampling frequency pi by 2 whatever f is by 2 what it is getting plotted. So you are seeing this is 16 db and this is 10 db, so hopefully it will not be so loud let us see.

We will go increasing this also, so we will put it as 6 db and then you can plot. So you will be seeing almost in steps what we are incrementing it here, and you can play the thing. So you can see this plot again we have increased the other one and we will do the other one little more than this one, and see how the plot looks like. This is what I said you can keep playing with the thing this has gone up to 2 db now here.

So it is not anymore 0 db for the high frequency and then will play the thing, so you can have your own music and then keep seeing it how it can vary, what are the components present in it, which one you want to highlight, and which one you want to suppress. As you would be seeing nowadays that most of you in mobile, so you will be having your own what is it audio basically for the whatever music you would be playing it, only orchestra will be there voice will be yours.

So voice of the original person is going to be separate and then you can have your voice inbuilt in that orchestra most of them you will be seeing single solo, they will be playing if without instrument. These people will be using the orchestra part of it and then voice will be this, so this is how you can play with your music that is what I wanted to show.

**(Video Ends: 30:30)**

So in the next class what we will do is continuing with our application of adaptive filter. So we will be seeing in code composer studio same adaptive filter with different application what we have seen in the MATLAB will be seeing in our board also thank you.