Real Time Digital Signal Processing

Prof. Rathna G N
Department of Electrical Engineering
Indian Institute of Technology – Bengaluru

Lecture - 36
Error Surface and Error Contour

Welcome back to real time digital signal processing lab. So, today we will see that how we are going to find the contour basically error surface and error contours.

(Video Starts: 00:31)

And then see that how adaptive filter is going to work, the examples what we have taken is from the book as you will be seeing that it is from real time digital signal processing fundamentals, implementation and application this is the 3rd edition book that is from Sen M Kuo Bob H Lee and then Wenshun Tian publisher is John Wiley and Sons, so, here you will be seeing that compute and plot the 3 dimensional error surface for $L = 2$.

So, how it is going to do the thing you will be seeing that $w0$ $w1$ $L = 2$ coefficients what it is being chosen and then you will be considering the meshgrid given by this define w0 and w1 arrays for 3 D plots what you are giving it and error is given with the equation point 3 4 minus 0.6 into $w0 - w1 + w0 * w0 + w1 * w1$. So, we know that this is the error signal what we want it to plot what we have considered in the class also.

So, you will be doing that mesh w0 and w1 with error that is what, what it says plot the mesh using where color is proportional to mesh height add major grid lines what it says. So, you will be adding on the grid and title is error surface what is given and then label x label x axis is represented with $w0$, and y label is $w1$ and then the z label we have is MSE that is mean square error what it is going to put. So we will run this code and then see that how it is going to look like.

So, you can see that this is x axis is $w0$, y axis is $w1$ and then mean squared error what you have given it on the z axis. So, you will be seeing that our whatever we saw in the theory. So, you will be seeing the x points y and then zee. So, this is the mesh on the grid what you have plotted. So as we have increased the resolution, so, we had to delete from here. Next one is how the contours looks like, error contours of mean square error whatever given in the example, we have taken the same example same from the same book.

So, you will be seeing that it is instead of as you have seen the thing here it is mesh grid with this and then this you will be finding out the contour of this error that is weights basically $w0$ and $w1$ with error and then we said for the 15 is the iteration what we will be fixing it. So, this also will run it and then see. So, you will be seeing 15. What is it? Concentric circle, this is how you will be implementing to see that your weight vectors what it is given if it is more than you can get your contour and then surface basically using this function.

So now the book, we have seen in the last lab that is students how they write their codes and other things. So we will see that how the book is going to give adaptive filter because this is from the Welch book what I have taken the thing. So, you will be seeing that different books have been referred for different applications or for any of the solutions, you can refer to different books and then you can get the code and then but you have to understand how they have be implemented and what is the thing happening.

So, here you will be seeing that you are taking the order of the filter is 20 that is number of adaptive filter coefficients what you are assuming it and mu step size is selected as 0.01, that is the convergence of factor. So, and then f naught is 1000 here it is going to generate a chirp start frequency and f1 = 5000 chirp stop frequency. So, chirp is a basically like you can birds chirp what we call it, so, they will be having different tones what they will be creating it same way between 1000 and 5000 Hertz you can create in steps of it using this function.

And then our step whatever stop is 20 that is time for the chirp to be at $f1$ what it says. So, that is you will be having steps of 20 with varying thing from 1000 to 5000. So, what you are going to give is your voice with the sampling frequency what you will be taking it from this voice recording dot wav, so, you can record your own voice and then store it as a dot wav and then you can use it then we will be using the audio read function in the latest MATLAB versions.

And then you will be convert the column to a row vector in this case, the output is going to be in column format. So, you are doing the transpose so, you will be getting it as in the row format, then what happens, M is the length of the voice what you are taking it so, what was the duration of voice you will take that length that is number of samples to be simulated and t is the time which is going to go that is create a time vector for the chirp command. So, you will be creating 1 to M/f_s.

So, noise is what you are going to have it as a chirp, which is t, f_0, t stop comma f_1. So, you will have 20 points that is creating a chirp signal here the function in MATLAB what it is called in as chirp and then what you will be doing your create the noise storage array x is going from 2 to N noise of whatever N - 1 previous one minus 1 you will be down sampling and then 1 this is the step size what you will be taking it. Then initially your weight vector is going to be zeros 1 to N the length of the filter that is what it says initialize the adaptive filter coefficients.

Then you will be doing the storage of your voice plus noise, create the signal plus noise in this case. And then you will be what is it? You are going to normalize the storage how we are going to do it? That is divided by maximum of absolute of d storage what you are taking that so that is the positive highest value dividing by that what you are going to do it. Then you are y storage we will have 1 to M zeros that is storage array for filtered noise and e storage is going to be your errors you can make I think you would have got the hint of it.

So which are going to be zeros 1 to M, that is storage array for the cleaned up signal what you are going to have it. So what is the algorithm for filtering, So you will be doing j is defined N to M what you are going to do in this case, that is interrupts service routine simulation starts here input the 2 channels of data, that is $x(1)$ noise j what you are going to have it interference of noise signal and d will be the desired signal from a d storage of j that is voice signal plus interference what you will be taking it as the desired signal.

Now, you are going to have adaptively filter the interference signal. So, you will be making it $y = 0$ if the length of the filter what you will go N - 1, y will be y plus you can see the weight $i + 1$ what will be considering it multiplied by $x(N - i)$. So, then end the thing and your error function is e is given by $d - y$. So, then update the filter coefficients, so, you will be seeing that $w(i)$ is updated as $w(i)$ instead of calling $i + 1$ so, we can update in the present state itself.

So, that our storage is going to be reduced $w(i)$ star 2 into $\mu *$ our error $x(N - i) + 1$, what will be taking it. So, this is the loop which will go and next is prepare the x array for the next input sample. So, you will be taking the next sample from this and then we will be working on it and then the interrupt service routine simulation is going to end here and you will be doing a storage after the post simulation.

So, you will be putting y storage of j will be y, whatever the output after finishing your up to filter length and e storage of j is going to give you error e then you will be listening to the results basically. So, with the original voice what you will see it voice comma Fs signal with interference you will be pausing for 24 seconds and then d storage which is added with noise you will be seeing with the same sampling frequency then you will be hearing the adaptive filters output that is where it has stored e storage will give you the adaptive filter output there.

So, these are the plots what you will be seeing it normalized one, how the original voice and then the record voice signal they are overlapped and then you will be seeing the output. So we will run the code because we have understood how it is adaptive filter is going to work so we will run it. So you are bored first was the original voice second one now was with the added chirp signal. So you saw that how the chirp signal is generated it had varying frequencies and then later on after adaptive filter you have seen that how the noise is eliminated almost what you can put it. Because we not be able to go up to 0, what we said the error cannot be completely 0. So you are seeing this is the audio signal overlap with chirp signal what you are seeing it so if you want to see a portion of it. So I can increase I think I may have to go back to original resolution because we try to increase so that we you can see the thing so, the filter thing is plot is going little up, in any case, you can see yourself by that is expanding the thing how the chirp signal is going to look like. So now we will see the input and output.

So, you will be seeing a blue is the input signal and then red is going to be the recovered signal. So, some places you will be seeing that red is little normalized both of them so, you will be seeing that it is a little low. So, can you guess what will be delay of our output in this case, the order of the filter is 20. So there will be 20 samples delay in the case of adaptive filter to start our output. So, you can see initial stages, what delay we are going to get it. So, this as you can see, it is marked recovered voice signal from the thing.

So you will be seeing that some of it is whatever you will see, see the thing, the magnitude is lessened in some places. So this is the output what you will get it and then but most of the cases we were able to recover from the noise signal our output. So in the next class, we will be seeing how to implement the same thing in Code Composer Studio, and few more applications. What we will be taking up using adaptive filter, both in MATLAB and then Code Composer Studio. Thank you.

(Video ends: 16:50)