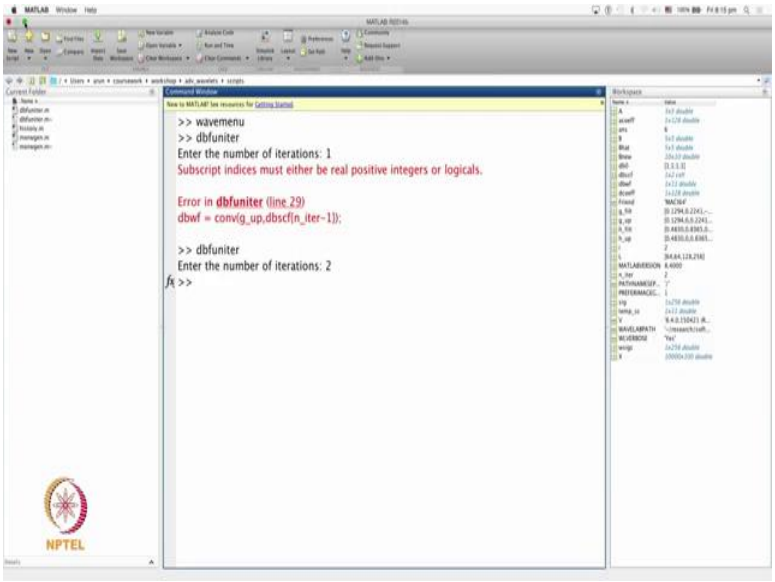
Introduction to Time-Frequency Analysis and Wavelet Transforms
Prof. Arun K. Tangirala
Department of Chemical Engineering
Indian Institute of Technology, Madras

Lecture - 8.4
Wavelets for DWT

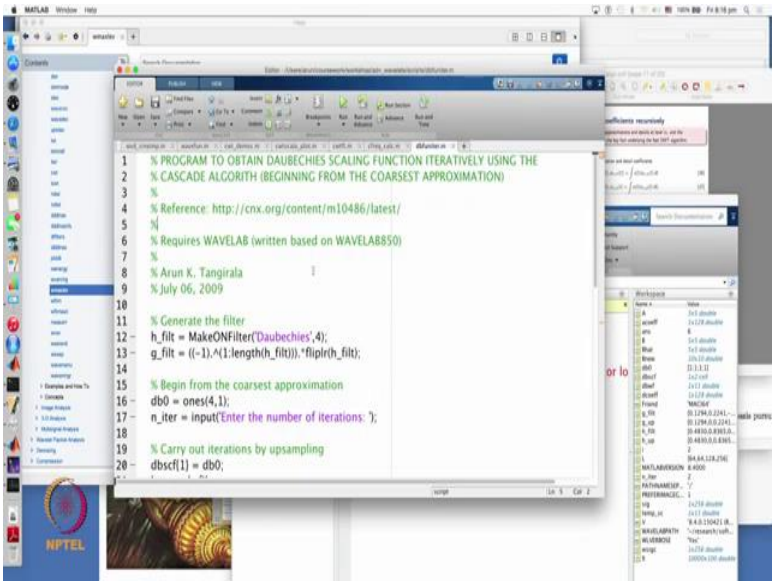
So, I have written a script here that performs the iterative algorithm that I just outline.

(Refer Slide Time: 00:17)



The script here… The script is shown here and you should be able to get the script from the web. When we post the lecture notes, we will also post the script. So, let me run the script and show you how the iterative algorithm works.

(Refer Slide Time: 00:48)

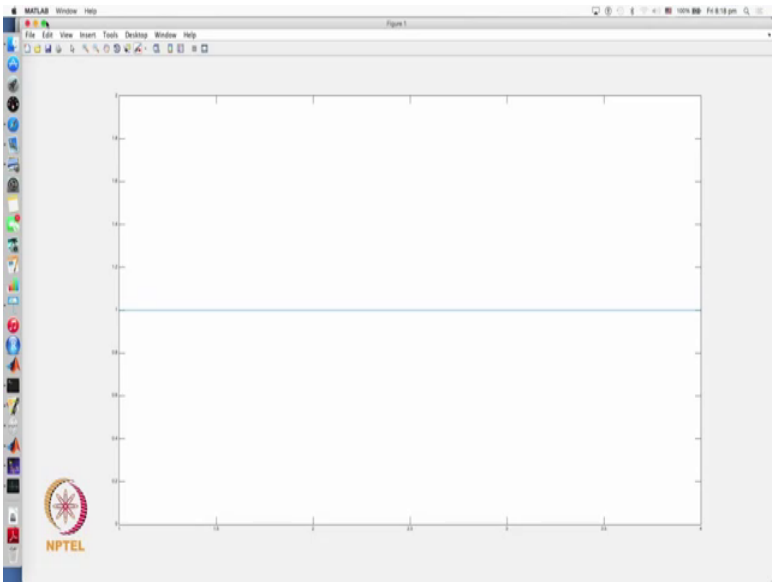


So, what I am doing here is, I am actually using wave lab to get the low-pass filter coefficients. I can use the wavelet tool box as well, does not matter. And, what I am interested in here is generating the Daubechies wavelet of vanishing moments too. In fact, if you look at the line here, it says that… It is asking for the filter coefficients corresponding to Daubechies family with the number 4 here. This number is not the vanishing moment, but the length of the filter. That is why I said there are conventions where people use the length as the specification or half the length which is a vanishing moment as a specification. Here in wavelab, the MakeONFilter; ON means orthonormal. That routine requires you to specify the length rather than the vanishing moments. So, I am asking for the filter coefficients here. And, I am interested in generating the phi of t corresponding to this db2. And, the accuracy of phi of t depends on how many times you run through that algorithm that we just discussed.

How do we initialize the phi of t? One thing I know for sure that, phi of t cannot have a zero average because it is a low-pass filter. So, what I am going to do is I am going to initialize the phi of t as a bunch of 1's – just a vector of 1's. In fact, ideally, I should be initializing such that it is not 1's, but 1 over n – 1 over 4, so that the sum of it. In fact, the area under that of phi of t should come out to be 1 and so on. But, we will not do that; that normalization we will do later on. So, as you can see here, my initial guess is that,

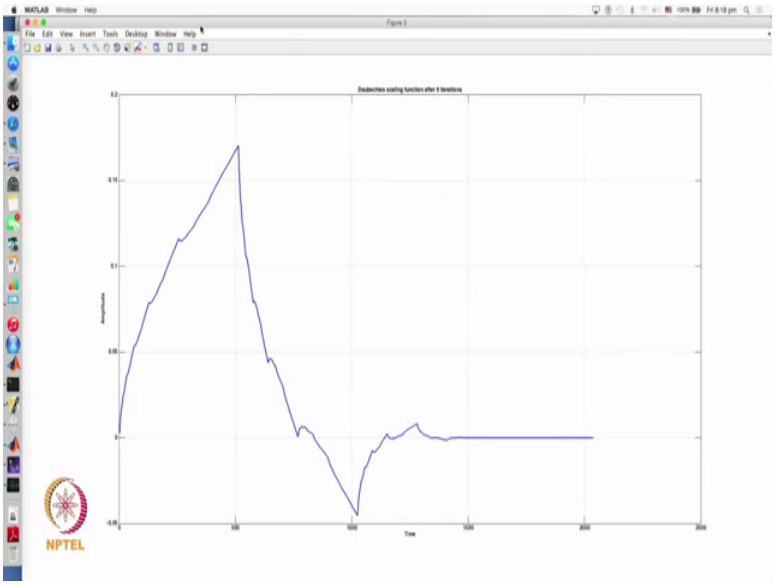
phi of t is taking on a value of one over a certain interval. So, this is how the initial one looks like.

(Refer Slide Time: 03:08)



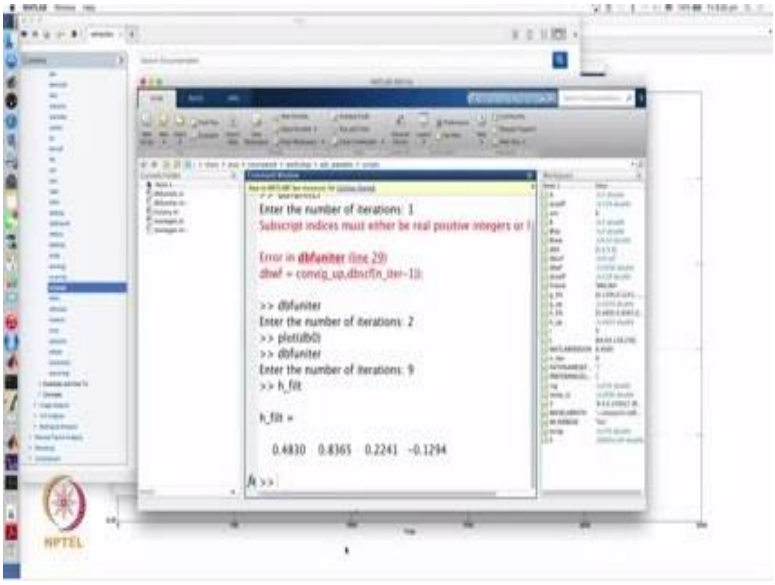
Let me show you the initial Daubechies wavelet – initial guess of the Daubechies wavelet, is just a plane straight line; that is it. Obviously, I know this is not db 2. But, the iterative algorithm magically constructs the db 2. Let us quickly see how it looks like. When I run through the iterations, I will take you… Let me run this here and then I will take you through the animation.

(Refer Slide Time: 03:40)



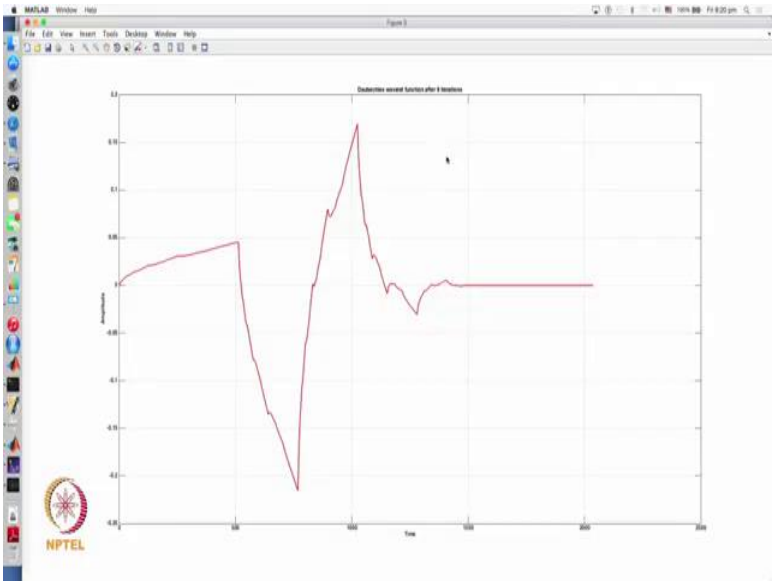
So, suppose I run it for about 9 iterations, what I have with me here are the Daubechies scaling function. Look at how it magically constructed the Daubechies wavelet from a guess. But, that guess is very important; you have to give a guess that make sense. You cannot give a guess such that it has a zero average; that is all. So, as long as you give a phi of t that has non-zero average, it should work fine. And, of course, here what you see on the x-axis; it says time. But, it is not time really; it is a number of iterate points over which I have computed iteratively. I started off with four points; but, with every iteration, it is doubled; the number of points is doubled. The time interval over which it exists is actually 0 to 3. Why? Because db 2 has four filter coefficients. Therefore, its support is over n 1 comma n 2; 0; n 1 is 0; n 2 is 3; that is, I have four filter coefficients at n equals 0, 1, 2, 3.

(Refer Slide Time: 04:54)



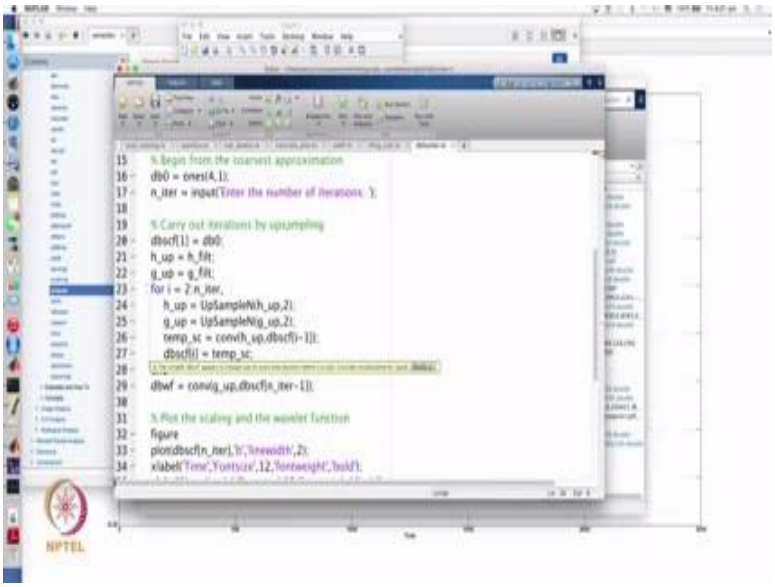
And, what are those filter coefficients? Those filter coefficients are here. You can also obtain these filter coefficients from matlab's wavelet tool box. There are four filter coefficients at n equal to 0, 1, 2, 3. So, the support of h is over 0 comma 3. By the result that we are seen earlier, support of phi is also over 0 comma 3. What you see as a plot here is actually over 0 comma 3. So, the time… This is only the number of points from 0 to 3. All right.

(Refer Slide Time: 05:29)



And, the associated wavelet has also been constructed. The moment I have the scaling function, I can also run the same algorithm by starting off with the guess for the wavelet. And, this is how the wavelet is built.

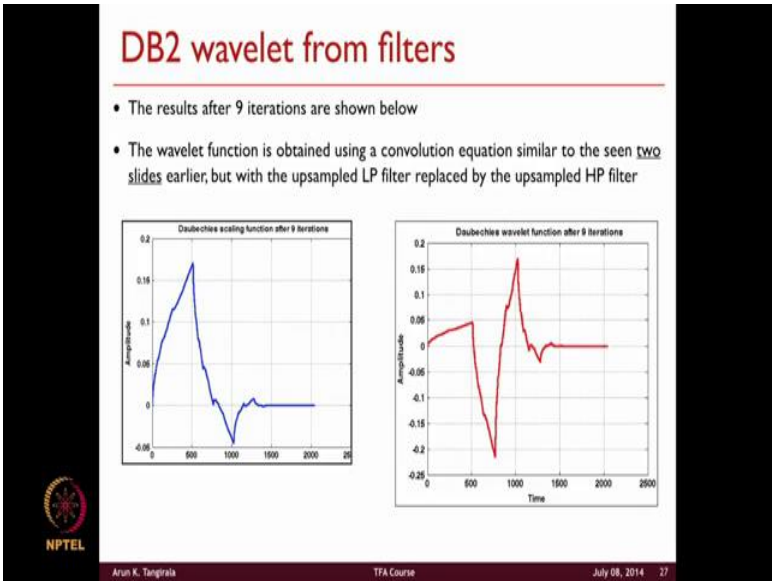
(Refer Slide Time: 05:48)



So, if you look at the code, the way I have written is… This is the wavelet function. And,

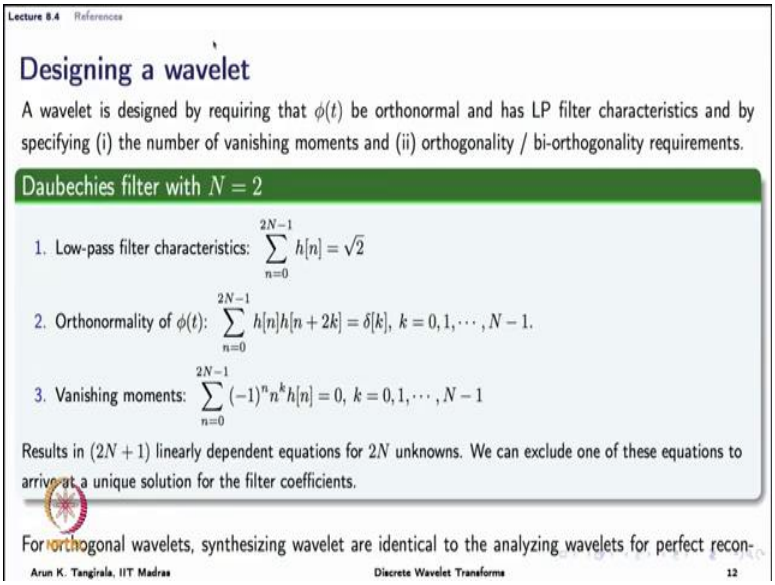
all I do is… Remember the wavelet at any stage is a convolution of g with the scaling function, because that is from the scaling relation. The psi of t is integral g of n phi of t minus n; sigma g of n phi of t minus n; that is it. So, I use that convolution expression and compute the wavelet. I do not have to run another iterative algorithm at all. All I need to do is generate phi of t and I will get phi of t. So, I hope you understand what is being done here. Essentially, this is a scaling relation that we have seen earlier.

(Refer Slide Time: 07:01)



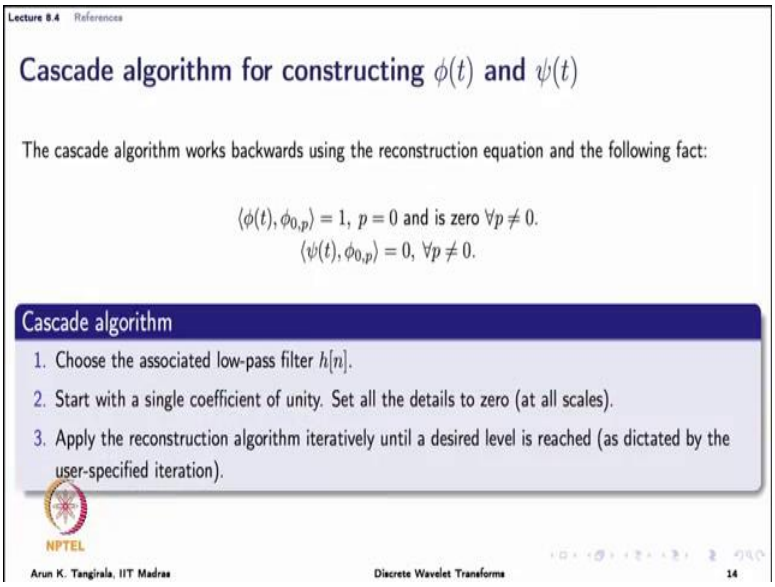
So, let me also show you just to make the thing very effective; that is, for you to see gradual guess – refinement of phi of t across the iterations, I am going to take you to another presentation that I have from previous workshop; where, this is the initial guess. Second guess – this is the development in fact after three iterations. And then, gradually, it develops fourth iteration, fifth iteration, sixth, seventh, eighth; that is it. So, then, you have a wavelet. So, that is called… That is the iterative algorithm.

(Refer Slide Time: 07:34)



Now, as I said earlier, there is also a second algorithm.

(Refer Slide Time: 07:37)

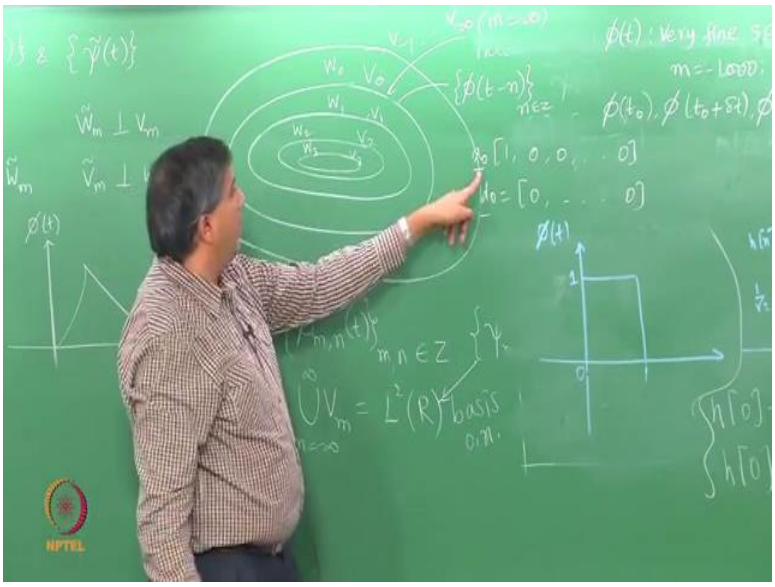


That is known a cascade the algorithm, which has a very beautiful thing to it. It is a very nice way of reconstructing phi of t; it is very elegant. This algorithm works backwards using the reconstruction equation and the following fact. Now, remember that, in the

MRA, what I am doing is I am constructing approximations of signals that are of finite energy. What I do here is I set the signal as a scaling function itself. Why not? See even a scaling function is some signal. Now, when I set the signal as a scaling function itself, then I know by the orthonormality property that, the inner product between the scaling function and the scaled scaling function is one. Here I am right now I am looking at scale 1, that is, at m equals 0. As you can see from the subscript here; and, p here refers to the translate. So, the inner product between phi of t, which is the scaling function – the continuous one and the phi of 0 comma p, that is, its own translate in other words. So, it is other way of saying that, the phi of t is orthonormal; that is all I am using. That is a property that I am using. But, I am rewriting it as an inner product.

And, I also used the property that, the phi and psi are orthonormal at any scale. And, here I am choosing zero scale. But, if you look at this result from an approximation coefficient view point; now, think carefully; I start from the signal, which is phi of t and I start projecting it onto different subspaces v 0 or v minus 1 or v minus 2 and so on. Ideally, where does phi of t live? phi of t lives in the finest scale; it lives at v minus infinity; phi of t lives in v minus infinity; that is, when m goes to infinity, that is where phi of t is residing. You can project this phi of t onto any of the subspaces. Let me just show that to you on the graph.

(Refer Slide Time: 09:59)



So, what is happening here is as I go here, I get v minus infinity, which is corresponding to m equals infinity. So, when I have phi of t… phi of t resides in this space really; v of minus infinity is actually the finest scale. I can project that let us say to v 0, which will construct some approximation, because v 0 – what is it spanned by? v 0 is spanned by phi of t minus n; and, n belonging to the interior side. And, what that result says is whenever I project phi of t onto v naught, there is only going to be one approximation coefficient that is unity and all the other approximation coefficients are going to be 0. So, imagine instead of assuming v minus infinity, I have phi of t at a very fine scale. Let us say at very very fine scale, maybe m equals... In fact, this is not m equals minus infinity; m is minus infinity. So, let us say I have over very fine scale, m equals minus 1000. What does m equals minus 1000 mean? A very fine scale over a very fine grid. So, phi of t is also known over a very fine grid. So, we will just relax that it is continuous; rather imagine that, phi of t is now known over an extremely fine grid.

When I come from an extremely fine grid to v 0, what I would have generated is a bunch of approximation coefficient; it is like taking any signal and I decompose any signal to some level; I have a number of coefficients. That signal is also sampled signal. The number of approximation coefficients of course will be much lesser than the original signal. So, here also what I am doing is I am taking phi of t known at some very fine grid. Let us say t naught and then t naught plus delta t; phi at t naught 2 delta t and so on. So, a very fine grid and projecting it. Here I have a bunch of approximation coefficients. Those approximation coefficients will be of certain length. And, what that result says is when I feed; when I perform a decomposition of phi of t to a level 0 let us say; then, what happens? I have a bunch of approximation coefficients, which look like 1; and, rest of them are all 0's. The length of this of course, is determined by what length you are looking at.

In fact, if you have some n grid points here… Let me not confused with this m here. Let us say I have s grid points here. Then, by the time I come to 0, the number of coefficients here would be s over 2 power m – 2 power 1000 if that is at 1000. But, that does not matter. Essentially, this phi is known at some very fine scale over a grid and I have subjected that to an approximation or a decomposition, where this result says you start from the finest scale and perform a decomposition; you will always end up with these set

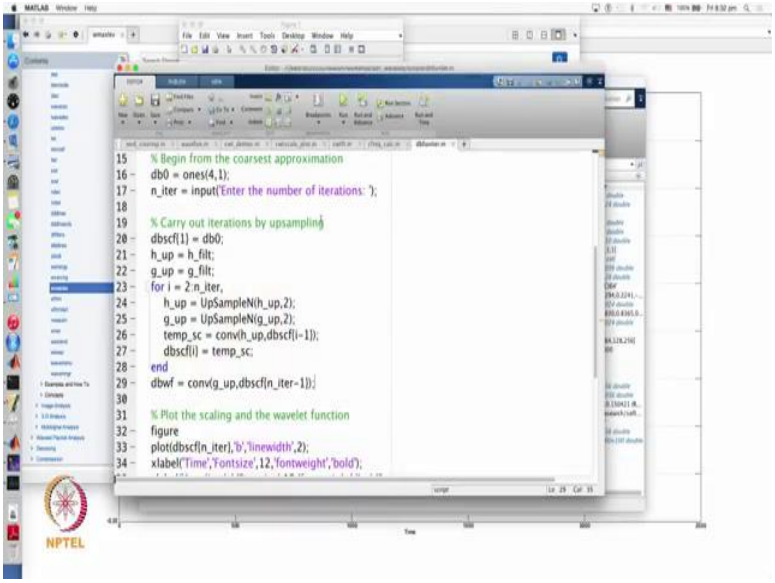
of approximation coefficients. And also, the detailed coefficients at this scale would be all 0's because the phi is orthonormal to the wavelet; which means the low-pass filter is orthonormal to the high-pass filter. Therefore, there are no high frequency coefficients at all.

Now, I do not know what is that phi of t; but I know for sure that, any phi of t associated with the low-pass filter will produce this approximation and detailed coefficient at level 0. Now, I reconstruct. So, I would like to go from v 0 to v minus 1. So, this here is the projection of phi of t onto a naught. I am going to now reconstruct a minus 1. I can do that; I have the reconstruction algorithm. So, I am going to run a naught through the reconstruction algorithm repeatedly as many times as I want to get phi of t. That is a basic idea behind the cascade algorithm; it is very simple one. But, you… And, uses the orthonormality property. So, what you do is you choose the associated low-pass filter whatever… For whatever family you want to construct the scaling function, start the low-pass filter, because it is all about constructing phi of t from the low-pass filter an then get your… Set initialize your guess for the… Not guess, initialize your approximation coefficients at level 0. Does not have to be level 0; but, just for discussion sake, we say it is level 0.

And then, set all the details to 0 and apply the reconstruction algorithm repeatedly. And, how do you do that? There is a command called up c o e f in wavelet tool box of matlab, which will allow you to reconstruct repeatedly either from the approximation coefficients or the detailed coefficients. And then, you will be able to reconstruct it at any level. All right? That is nothing but the reconstruction algorithm with all the other band coefficients set to 0; that is all; it is nothing very great about it. In fact, this algorithm now can be applied to the wavelet also. For the wavelet, what happens is… The property of the wavelet is such that the detailed coefficients… If you were to feed not the scaling function as a signal; but, if you were to feed the wavelet as a signal, what kind of approximation and detailed coefficients would you get? You would get the reverse situation. So, I start from psi of t. That is what I would feed and then perform the decomposition many many many times. Then, when I come to a very coarse scale, what I would have is all zero approximation coefficients, because a wavelet has nothing in the low frequency band. And, the detailed coefficients would be 1 at lag 0; that is, at time 0

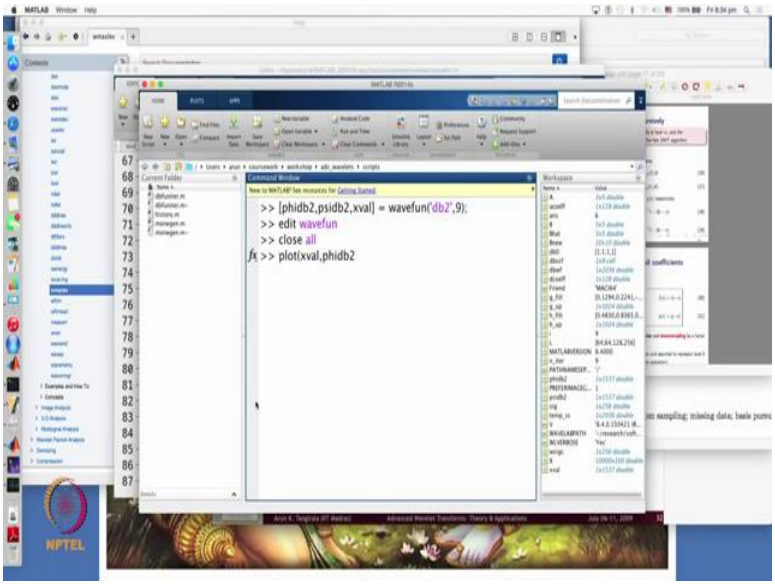
and 0 elsewhere, because of the orthonormality property of the psi; and, that is it. So, I can also apply this to generate the wavelet. And, this is exactly the algorithm that is implemented in the wavefun.

(Refer Slide Time: 17:20)



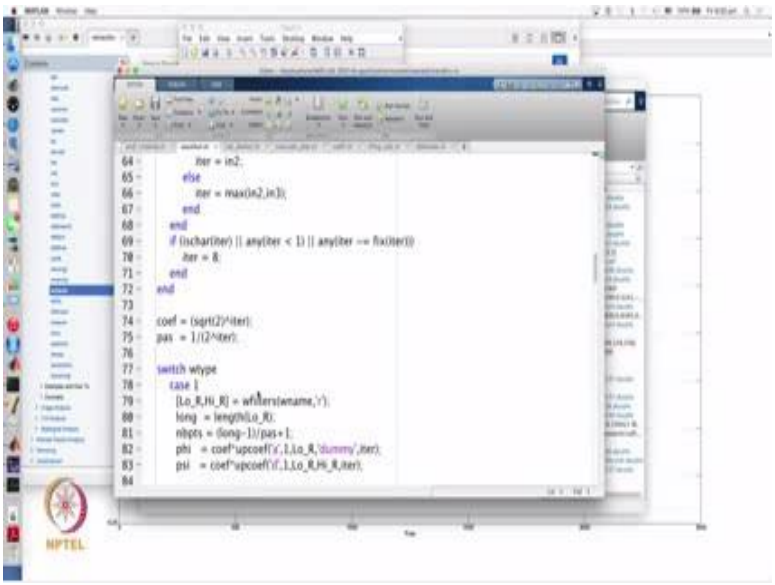
So, remember; we use this wavefun long ago in the CWT – lecture 7.2. When we talked about scaled frequency, we started with a specification of the low-pass and high-pass filters for the Daubechies. And then, we said… We specified the number of iterations to generate the scaling function and the wavelet. And, that is exactly what the wavefun does.

(Refer Slide Time: 17:45)



So, let me do that; phi db; you can see here I am going to 12 or even 9 iterations; does not matter. And, that gets me the db 2. In fact, if you open up wavefun, you will see that, it is exactly implementing the algorithm that we just discussed.

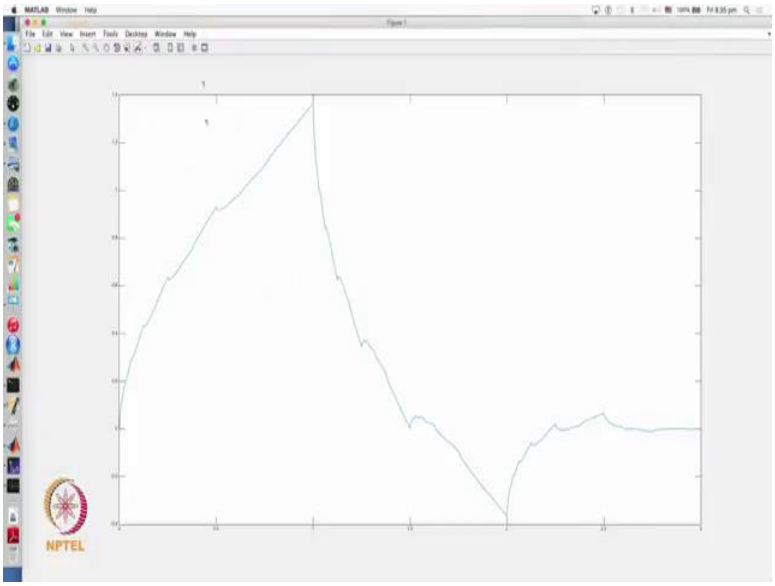
(Refer Slide Time: 18:12)



If you throw away all of this, what it does is here. Here it gets a filter coefficients and the

up c o e f is a key. To generate the scaling function, it uses an approximation coefficients. And, what are the approximation coefficients? If you go up, you will see… In fact, here itself, it says, the approximation coefficient is simply a single coefficient of value 1; that is, you have started from phi of t and you have decomposed it to such a level that, only a single coefficient is available; and, that single coefficient value is 1. Given any signal, I can perform the maximum decomposition such that I only have one coefficient left. And, if I… What the previous discussion tells us is if I start from the scaling function itself, that is, a decomposition scaling function itself, there will come a level at which there is a single approximation coefficient. And, that is the value of 1. Likewise, if I start from the wavelet and I perform the decomposition, then I can reach a level at which there is a single detailed coefficient; and, that is the value 1. So, the up c o e f performs a repeated reconstruction; that is at the core of wavefun. So, let us plot here.

(Refer Slide Time: 19:46)



So, as you can see, this is this scaling function now over the interval 0 to 3. As you can see, it is over 0 to 3 because that is the effective support. And, the result of course, looks identical at least in shape to what we had with the previous algorithm. And, if I look at the wavelet, this is how you iteratively generate the scaling functions and wavelets starting from filters.

Now, what we shall do in the next matlab session – the fresh matlab session is I am going to show you how to compute the wavelet transforms and how one can perform signal compression and how one can perform signal estimation and all of that. Primarily, it will be a matlab session; and, I will give you a brief theory of signal estimation. The basic idea and signal estimation has already been discussed. You decompose the signal and then you apply some criterion like we did on the CWT and the short-time fourier spectrogram and so on. You apply a criterion on the coefficients at all the scales or at some scales and so on. We call this as thresholding; that is, you throw away certain coefficients or you retain and so on, and then reconstruct. So, the basic idea is transform, perform some operation in the transformed domain, and then reconstruct. That operation in wavelet domain is nothing but thresholding. So, there are several flavors of thresholding and of this of this basic idea, which we will discuss in the next matlab session. So, it will be more practice-oriented and a bit of theory on signal estimation. That would be the final lecture on this topic of discrete wavelet transforms.

There are of course, other applications of dwt such as discontinuity detection. I will also briefly illustrate the idea of discontinuity detection in DWT; and, how choosing wavelets with higher vanishing moments – different vanishing moments makes a difference in discontinuity detection. So, what we are going to do is look at how to perform decomposition and reconstruction, detect discontinuity briefly, and then a spend lot more time on signal estimation, discuss the different thresholing methods. And, with that, we will conclude the topic of the discrete wavelet transforms. And finally, we will have a closing session, that is, a closing lecture for the entire course. Hopefully, you enjoyed this lecture and learnt quite a bit. And, see you in next matlab session.