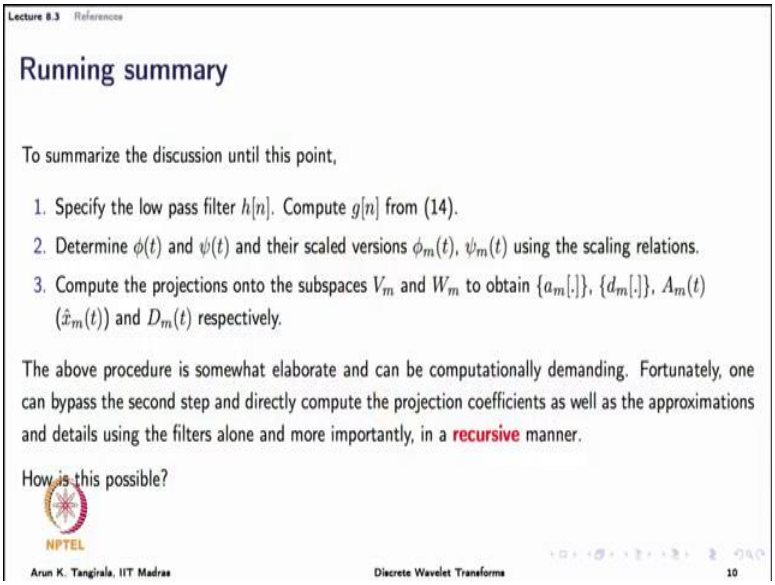
Introduction to Time Frequency Analysis and Wavelet Transforms

Prof. Arun K. Tangirala
Department of Chemical Engineering
Indian Institute of Technology, Madras

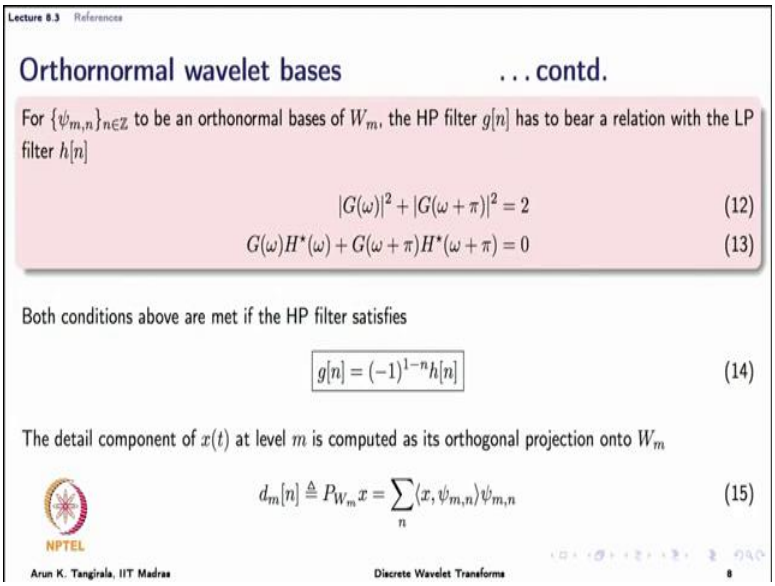
Lecture - 8.3
Wavelets Filters and Fast DWT Algorithm
Part 2/3

(Refer Slide Time: 00:16)



To summarize the discussion that we had until now, there are essentially 3 steps in the computation of DWT. Although there are 3 steps, as far as the user is concerned the user has to specify only the low pass filter which is the first step. And the low pass filter is denoted by h (n) as for our notation.

(Refer Slide Time: 00:40)



And then one computes the high pass filter using equation 14 which is this g (n) is minus 1 raise to 1 minus n h (n). In fact, I would ask you to do a simple exercise to show that if h is a low pass filter indeed g is a high pass filter. How would you do that? Well, there are several ways of doing it. But one simple way perhaps is to consider these filters in the frequency domain that is the frequency response functions of this low and high pass filters, and show that g of omega at omega equals 0 is 0, because that is one of the key requirements of the high pass filter.

And in showing that keep in mind that h of omega at omega equals 0 is not 0, because it is a low pass filter. In fact h of omega at omega equals 0, as per our requirements on the h is root 2. So, if you recall look at this equation 6, clearly tells me that the frequency response function of the low pass filter at 0 frequency is root 2. Using that and this relation in 14, I would like you to show that g of omega at omega equals 0 is 0; it should be a fairly a simple exercise.

So, let us move on with the 3 steps for computing DWT. So, as a user I specify the low pass filter compute the high pass filter; and in the second step I compute the scaling function and the wavelet function as well as the scaled versions using the scaling relations that we had seen earlier. So, specifically this scaling relations that we are referring to is 10, which tells me how to compute the wavelet given this scaling function and the high pass filter.

As well as equation 3, which tells me how to compute the scaling function given the low pass filter as well as the scaling function at a finer scale. So, we use this iteratively and we can compute the scaling function and wavelets at different scales because I know h n g. And of course, I have to start of with the scaling function at some scale, but we will talk about that in the next lecture where I will show you how to construct this scaling functions and wavelets, given h and g.

So, let us say you have done that. Going back to the 3 steps, we have done step 2 starting from h and g; we learnt how to compute this phi and psi, and scaled versions. Then remains the third step which is that of computing the projections, and the coefficients of course, on to the subspaces V m and W m. That is essentially we want to compute the approximations and details at level a m; m is generic.

So, we will compute approximation details at all levels. And in computing those approximations and details we would invariably compute the coefficients, indicate, denoted by the small a and small d at the respective scales. So, these are the essential 3 steps for computing the DWT. It is quite different from the way you compute CWT.

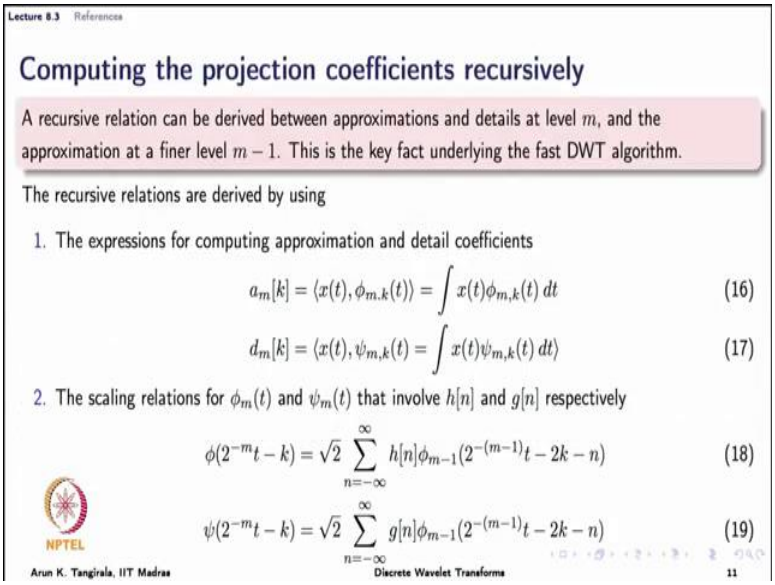
Now, the main difficulty with this 3 steps is, it is computationally quite demanding. And we should find means of computing this in an efficient manner. Unfortunately, for us there exists an efficient manner as I had mentioned earlier, where you can bypass step 2 which is the computation of the scaling function and the wavelet function from the filters. Without actually computing the phi and psi I can directly compute the projections; essentially, the approximations and details, and the coefficients from the filters themselves.

And that is the key; one of the key ideas in the fast algorithm. But, a more important point is that we can do this in a recursive manner. What do we mean by recursive manner? So, given the scaling function coefficients or the scaling coefficients, that is what we call the small a m, and the detail coefficients of the wavelet coefficients which again we denote by d m, given these 2 at a level m, the scaling coefficients and the wavelet coefficients, I can compute the scaling functions and the detail coefficients at a coarser level, that is a m plus 1, d m plus 1.

In fact, I do not even need d m, I just need a m, as we will see shortly. Once I have a recursive relation like that, then all I need to do is compute the coefficients at one scale, and then recursively compute the coefficients at all the subsequent coarser scales. Once I

have the coefficients, obtaining the approximations and details, that is the A and D, involves simply reconstructions. And we will talk about that, and we will see that there exists a fast algorithm for doing that as well. Now, the question is, how is it possible to derive a recursive relation? What is that relation?

(Refer Slide Time: 06:23)



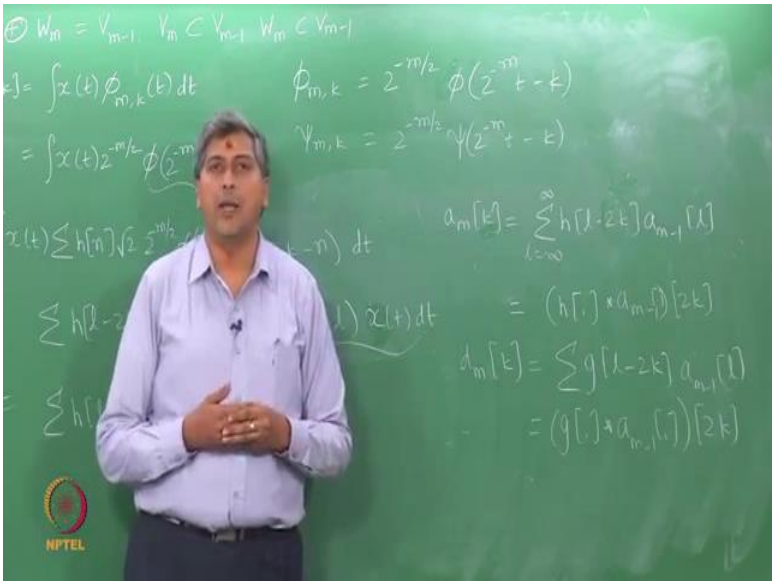
Now, to compute this recursive relation we use 2 sets of equations. The first set of equations are the expressions for the coefficients themselves which I have given in equation 16 and 17 here. We know already that the approximation coefficient is the inner product of x and this scaling function at that scale, right. And the inner product is given by this integral.

Strictly speaking, in integral I should have a phi star, but since most of the times in DWT we use real valued wavelets, I have dropped the star for you. So, its integral x (t) phi m subscript k (t) d t; now, k is the index that is, that keeps track of the time point at the m th scale. Likewise, the detailed coefficient d m (k) is a inner product of x and the wavelet at that scale. Once again I have the integral expression here. So, this is one set of equations.

And the other set of equations now, that I am going to use to derive a recursive relation are the scaling relations themselves. Remember, the scaling relations allow me to compute the basis functions for a coarser scale in terms of the scaling functions at a finer scale. So, as equation 18 and 19 show us, the phi at this scale m, in fact, it is not exactly phi at scale m, it is phi of 2 to the minus m t minus k. The phi at scale m would be what

you have on the left hand side, multiplied with 2 to the minus m by 2, right. So, let me just show that to you.

(Refer Slide Time: 08:23)



So, phi m comma k is 2 to the minus m by 2, times phi at 2 to the minus m t minus k. And likewise, the wavelet at scale m and at location k is 2 to the minus m by 2, psi 2 to the minus m t minus k. Once again I should remind you that this k helps me keep track of the time index at the scale m. Remember, the indices, the translation indices change with scales because I am moving the windows proportional to the widths or the scales of the basis functions or the scaling and wavelet functions.

So, what you have in equations 18 and 19 are relations for this, right. And the relations in 18 and 19 are nothing but generalization of these scaling relations that we have seen earlier. In fact, what we had seen earlier is that phi (t) by 2 is root 2 sigma; what we had seen earlier is phi (t) by 2 is root 2 sigma h n phi (t) minus m. So, this is nothing but the special case of this, where m is 1 and k is 0. So, as you can see here, a substitute in that equation 18, m equals 1 and k equals 0, I will get this relation.

And likewise, I had the scaling relation for the wavelet that we have seen earlier as well. This is once again a special case here. Of course, if I bring the root 2 here, then I have root 2 to the left here or to the left here, then I have psi m comma k. But, as you can see, the relation that you have in 19 gives you this when you evaluate it at m equals 1 and k equals 0, alright. So, the set of equations that I have for the approximation coefficients

and the detailed coefficients and the scaling relations, put together will give me the recursive relations.

That is how; so what is the key idea? The key idea is I start with the equation 16, because I want to compute a m that is the approximation coefficients at level m, given the approximation coefficients at level m minus 1. Remember, we know that V m and W m put together, give me V m minus 1. In other words, V m is a subspace of V m minus 1, and so is W m. Therefore, to compute a m and d m, I only need a m minus 1 which live in V m minus 1.

So, I need a recursive relation for a m in terms of a m minus one. What do I do? I begin with the expression that I have in 16 which is for the projection coefficient, and observed that this is x (t) times phi m k (t) d t, alright. And what do I do now? I expand c of m comma k which is 2 to the minus m by 2; I know this already here; phi of 2 to the minus m t minus k d t. And at this point I bring in equation 18 which tells me how this term is related to the basis functions for level m minus 1, right.

So, using equation 18, I rewrite this integral as x (t) times sigma h (n) times root 2. So, now, let us not; so let us write sigma h n here root 2. And then I have 2 to the minus, m by 2 already here, times phi 2 to the minus m minus 1 times t minus, what else do I have? I have 2 k, and I have an n. So, that mean, minus 2 k minus m d t. So, all I have done is, I have substituted for phi using equation 18.

What we shall do is, first interchange the order of summation and the integral, right. So, when I do that, I get sigma h of, what; an additional step that we shall do is, we shall call this 2 k plus n as some l, so that I realize n is l minus 2. So, instead of n, here I write, l minus 2 k. And then I have 2 to the minus m minus 1 by 2, where I have clubbed root 2 with the 2 to the minus m by 2.

And then now I have phi 2 to the minus m minus 1 t, minus l d t. Now, this is very nice. Of course, there is an integral here, I am sorry; so integral of this times x (t) d t. The reason for changing the order of the summation and integral is, will now become obvious. I recognize this part here, what is this? This part here is nothing but, a m minus 1, right. This is, a m minus 1; at what time?

At lag l or at the translation index l, because that is by definition if you look at 16, that is exactly what we have, but we have it for m right. We have it for m, but if I evaluate for m

minus 1, now, l is the translation index that will keep track of the time locations at scale m minus 1 or level m minus 1; that is it.

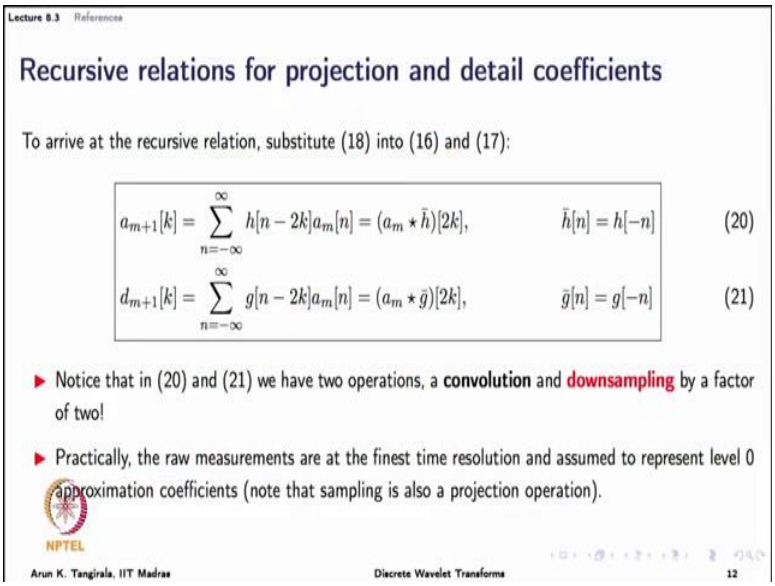
So, I have now this familiar relation for a m k in terms of a m minus 1. So, if I have to write here the relation, the final relation that I have; I have a m of k as sigma h l minus 2 k times a m minus 1 l. It is fairly obvious that this is a convolution expression, right. It is a discrete time convolution. And here, l runs from minus infinity to infinity. But, the only difference between the regular convolution and this one, if this was only a convolution then I should have had only a k, then that makes it a perfect convolution, but I have a 2 k here, and therefore, this is convolution plus down sampler.

So, think of this convolution being implemented in 2 steps; 1 - a regular convolution. So, think of this as k prime; and k prime is 2 k, right. So, in fact, you can say this, if this is k prime, this is k prime by 2; k is k prime by 2. So, either way you look at it. Anyway, so we will; now that you have understood hopefully that this is convolution followed by down sampling.

I can write this as convolution of h with, a m minus 1, but followed by down sampling. So, a m of k, now is very easy to obtain. All I do is, given a m minus 1, I convolve with h; that is a beauty; h does not change with scale at all, we have noticed that earlier; h remains the same as long as I am looking at 2 successive scales. So, I am looking at 2 successive scales which is the same h, and I convolve and then down sample by factor of 2; down sampling would mean essentially throwing away every other sample.

Similarly, you can derive a recursive relation for d m going by the same procedure. And here, instead of h you would have a g. This is once again convolution of g with a m minus 1, here also have, but down sample again by a factor of 2; that is it. So, I have the recursive relations that I want. I start of at a certain level; what level do I start off with? We will talk about it very soon. But, I start off with a certain level at which I know the approximation coefficients, and then obtain a and d recursively; it is a beautiful algorithm therefore, because convolutions can be implemented in a very efficient manner.

(Refer Slide Time: 19:02)

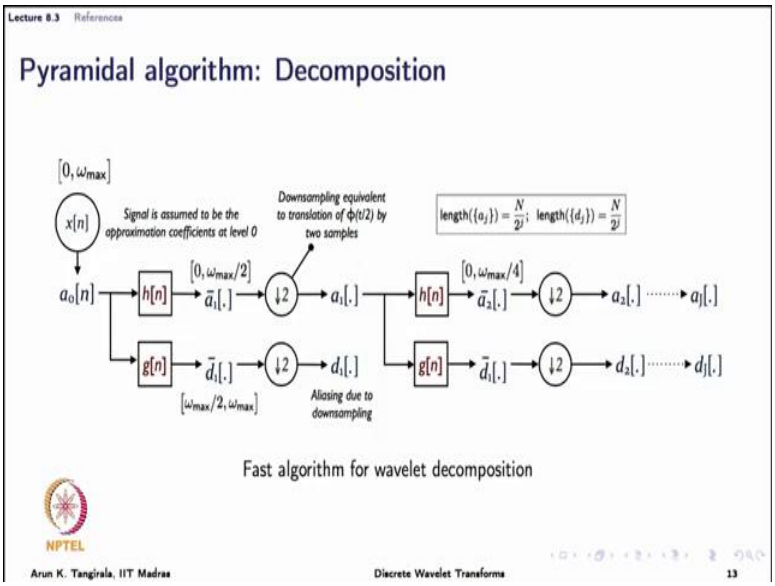


So, equations 20 and 21, essentially summarize, or give you the relations. In fact, there is a slight modification that I wanted to make, let me make that in light of what do you see in 20 and 21. It is not exactly convolution here because convolution would have been 2 k minus l, I am sorry about that. So, it should be convolution of a with h, reflected versions of h and g. So, there are 3 things that are different, earlier we said 2, but there are 3 things that are different.

One is that a regular convolution would have, if I denote this as k prime, regular convolution would have h k prime minus l, a minus 1 l. But I have l minus k prime; pretty much like what we have seen in continuous wavelet transforms and so on. So, I have this; and likewise, for this expression as well. Therefore, it is not a convolution with h, but reflected version of h, and that is what you see in equations 20 and 21; apologies for the mistake.

So, we have 2 operations; of course, convolution and down sampling, in addition to the reflection. Now the question is, what scale or what level do I begin at? The level at which I begin is the level at which the raw measurements are available. The raw measurements are available in form of a sample signal. I never have x (t) with me. I only have a sample version of x (t).

(Refer Slide Time: 20:51)



And what I assume therefore, if you look at the, now the schematic for the so called pyramidal or fast algorithm due to Mallat, this was in fact a breakthrough in the computation of DWT, pretty much like the way the f f t was a breakthrough in the computation of DFT. So, again coming back to the point, we assume that the signal that is available to me, the sample signal that is available to me is nothing but a set of approximation coefficients at scale at level 0. Now, is this true? Does this make sense?

Well, first of all remember that x (t), it is a continuous time signal, is sampled to obtain x (n). Now, for those of you are familiar with sampling and linear algebra, sampling is nothing but a projection of x (t) on to some basis functions, right. And we can treat x (n) as some projection coefficients of x (t) on to some basis base. We do not know what is at x (t), but we will say that I now have the approximation coefficients of some signal x (t) which I do not know, on to the wavelet basis base that I am or the scaling function basis base that I am looking at.

The recursive relations are between 2 successive scale levels. Therefore, I do not have to worry whether I call the signal, available signal at being at level 0 or level minus 1 and. so on. But, for convenience, we will assume that it has level 0. So, the basic point is we assume the sample signal to be the approximation coefficients and that is the best approximation you can have. If you want a finer approximation beyond what you have in sample form, you have to go back sample faster and obtain the sampling sequence at a finer resolution in time.

The resolution at which the sample signal is available is t s. All the other approximations that I am going to construct, the coarser approximation coefficients that I am going to compute are going to be at poorer resolutions. But, what do I gain? I gain on the frequency localizations. So, the time localization versions as I go down this scale, go down the resolution or go up in m, but the frequency localization improves.

So, let us now understand the algorithm. We have already looked at the basic equations. I start with the sample signal which is assumed to be, a 0 of n, convolve with h; strictly speaking, it is h bar and g bar. And then obtain this intermediate sequences; this intermediate sequences are at the same resolution, time resolution as, a 0, itself because I am not perform any down sampling here.

The moment I down sample, then I obtain the desired coefficients, a and d, at a coarser scale that is at a higher m. By our notation we call them as, a 1 and d 1. Because of the down sampling, a 1, has half the length as, a 0, and d 1, has half the length as, a 0. So, if you put together, a 1 and d 1, that is you stack them up the total length of a 1 and d 1, is going to be the same as, a 0. So, you have not really increased the representation. The number of coefficients required, that are involved are still the same.

What we have done is, some kind of rearrangements. Think of, a 0, as being some bunch of colored balls; what we have done is we have rearranged the colours; and it is a nice example because the frequency bands correspond to some colours. What we have taken is, done is, we have taken range of colours, and taken balls in the range of colours and put them in, a 1, with some modification. And then we have taken balls in another range of colours and done some modifications and put it in, d 1.

Now, the nice thing is, ideally the frequency bands that, a 1 and d 1, live in are non over lapping. That is, and also the frequency bands corresponding to, a 1 and d 1, are different from the ones in, a 0; a 0, has, that is the sample signal has all frequencies from 0 to miqiz frequency that is omega max. Whereas, a 1, has all frequencies in the range 0 to omega max over 2, under ideal conditions. It is more or less true. And d 1, has the frequency range, omega max over 2 to omega max. So, we have gained on the frequency localization, but we have lost out on the time localization because, a 1 and d 1, are now spaced 2 t s apart. And I repeat this exercise.

Remember, I had only need, a 1, to obtain, a 2 and d 2. And I say, move down here as I come to, a 2, the frequency falls by a factor of 4. In fact, it is wrongly indicated that, a 1,

a 2 tilde here has 0 comma omega max of 4; it should be ideally on, a 2, not on, a 2 tilde. And d 2 would have frequency content in the range omega max over 4 to omega max over 2 and so on.

So, when do I stop this decomposition? Well, that differs from application to application, and it is a user specified value. Theoretically, if the number of samples n is a power of 2, let us say, 2 power J, then you can go up to J, so that in the end you have 1 coefficient in your courses decomposition, 1 coefficient in a 1, and 1 coefficient in d 1. So, that is the maximum level to which you can decompose.

So, these are now we have what is known as a signal decomposition in the scaling function and the wavelet domain, but normally we call this as wavelet decomposition. These are not approximations or details, these are approximation and detail coefficients; they are different as we have seen earlier.