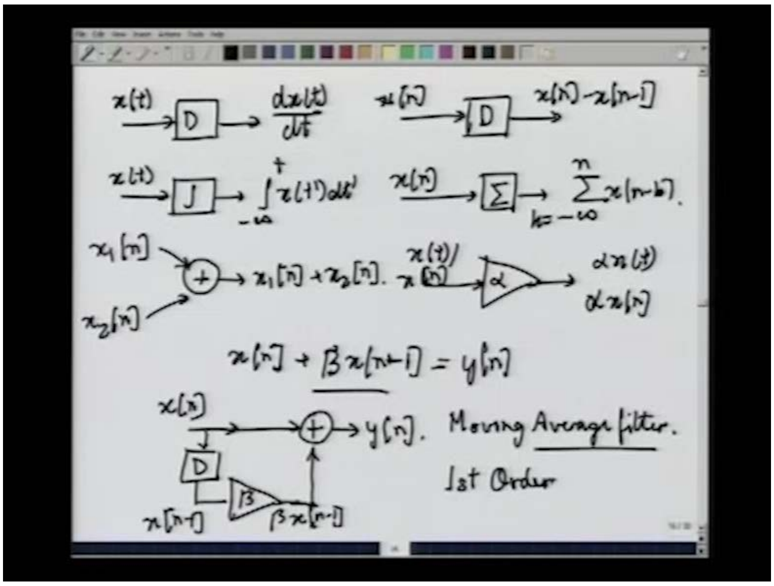
Signals and System
Prof. K. S. Venktesh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture - 22
Filters

We now, know that we have a means of differencing a discrete sequence, we have a means of carrying out a running form of a discrete sequence, we can differentiate a continuous time function, we can running integrate a continuous time function, we can do all these things. Suppose, without regard or prejudice to the manner in which these circuits are constructed which can do these calculations. Suppose, we nearly represent them by a block.

(Refer Slide Time: 01:11)



Suppose, we say that we can have a system, which takes a continuous time signal x t and yields at the output d x t by d t or we add a similar discrete system, which took in x n as the input and generated at the output is first difference x n minus x n minus 1.

We would represent both these by D and D the context would make it very clear, whether we are concerned with a discrete system or a continuous time system. Similarly, we could have a running integrator or a running summer x t goes inside and what you get at the output is integral minus infinity to t x t dash D t dash, fine here you would have this. We wound denote by this symbol and this we would denote by this symbol over here, it would be the running summer x n would yield summation minus infinity k equals minus infinity to nx n minus k.

Suppose we have these blocks let us construct some more blocks, let us have a summing block, this block takes two inputs let us say x 1 n or x 1 t it does not matter, the symbol assume is the same x 2 n and the output it produces is x 1 n plus x 2 n this is what we would call a summing block.

We could also have a special block for a system, which nearly multiplies by a scalar constant we call that an amplifier in conventional electrical engineering. So, we could have an amplifier, which takes a sequence x t or x n as the input this depending upon whether it is discrete or continuous. And produces us the output let us say, if this has a gain of alpha you would get alpha x t or alpha x n as the output.

So, these are the four kinds of blocks that will constitute the objects of our next discussion. Our next discussion will be to actually build systems, which can solve differential equations or difference equations. Suppose we want to solve the difference equation that we have been playing around with till recently suppose, we have let us say even simpler system. Suppose we just had x n plus beta x n minus 1 equals y n can be build an interconnection of these elementary blocks that we have over here to get y n, if we have the sequence x n supply to us, that is not very hard.

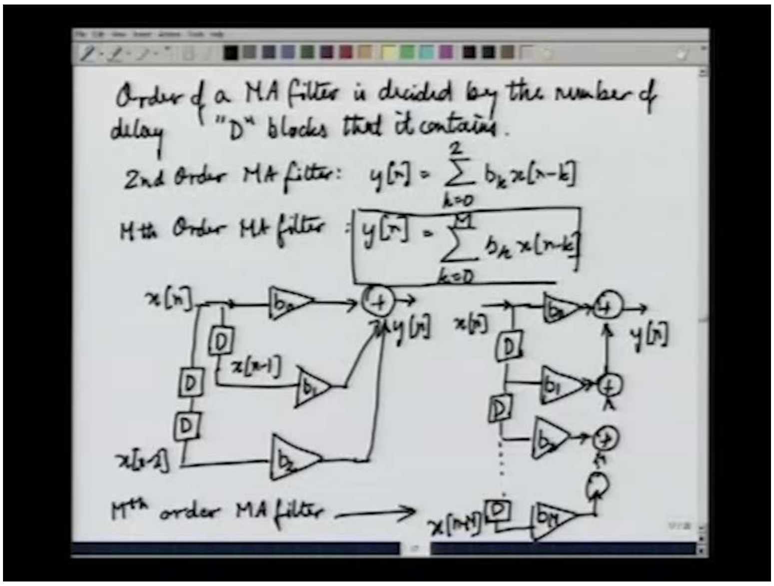
Let x n be available over here we need to add x n to something, namely to this term let us do that by constructing a summing block. Now, what is the other thing that we need to add to it is beta times x n minus one, but we do not have x n minus 1 available right now, we will have to make it by taking x n out of here putting it in a D block. So, that what you get here is x n minus 1.Now, this is only x n minus 1 we want beta x n minus one. So, what we do is to append another amplifier block with a gain of beta and this now is beta x n minus 1.

So, you have x n coming from this side beta x n minus 1 coming from this side and we now, add the 2 and what you get here, would actually the y n. This is a very simple kind of filter, which just takes the present value of x n and a certain scaled value of the previous x n that is x n minus 1and adds them to yield the current value of y n in short y n is said to be the moving average of x n.

This is called a moving average filter, it essentially creates a gradually moving average at every instant of time, and the window over which the input sequence is being averaged is being shifted by 1 position. So, you get this moving average filter. More precisely this is called a first order moving average filter, if you want to know why the fact is that the order of a moving average filter, which we will simply call an MA filter, is decided by the number of delays or D blocks that it contains. In this case it contains only one. So, it is a first order filter.

If you wish to generalize this to a second order filter all you would have is an additional D block and therefore, a second delayed version of x n namely x n minus 2, which would also be scaled by some factor and would get added to x n and a scaled version of x n minus 1 and so on to get y n.

(Refer Slide Time: 08:14)



So, let us write the equation for this I will not draw the block diagram of this instead I will go ahead and define the n th order moving average filters or the M th order moving order filter directly. If you wish to see how this is implemented well, there is a naïve way of implementing it and the other slightly smarter way of implementing it. You will apply the naïve approach for a second order implementation that is to say for a second order filter M equal to 2 and then we are going to see how we can make it slightly smarter.

We have x n as before and we need both x n minus 1 and x n minus 2. So, 1 way would be to take x n minus or to take x n and delay it by 1 unit of time to get x n minus 1 this x n minus 1 would then be scaled by b 1. x n itself would be scaled by b 0 according to the equation and you also need x n minus 2. So, you could start with x n right away put 1 delays over here and yet another delay over here. And get x n minus 2 to which you would attach the appropriate scale factor b 2. Now, that you have all these you could take a summing block over here add this, add this and add this and all this together would give you y n.

Now, this kind of a diagram is right for just a few very very straight forward simplifications. First of all let us say, that we want to keep the number of blocks of whatever kind to the minimum be the delay blocks or gain blocks such as b 0, b 1, b 2 or the summing blocks that we have over here. In the case of the summing block, which is the circle with the plus inside, we want to ensure that no summing block requires is at any time involved with summing more than 2 numbers, 2 quantities, 2 functions whatever.

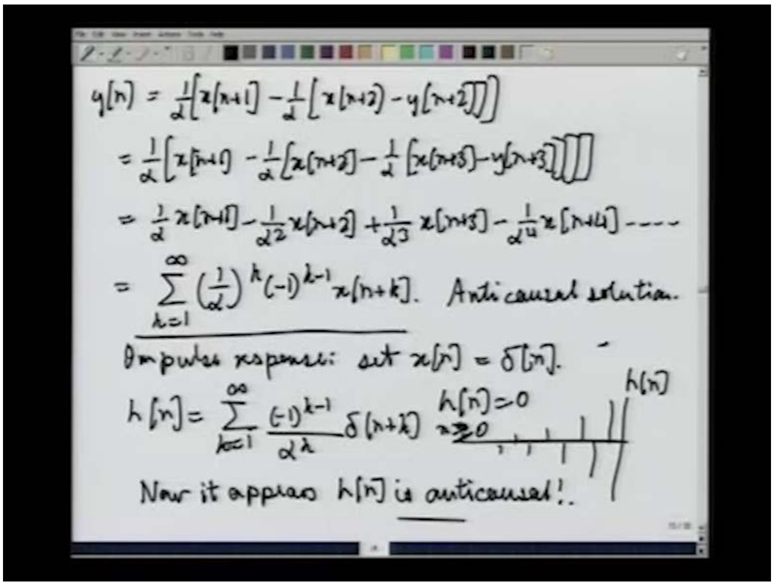
So, keeping these constraints in mind, we can make a simplified version of this diagram which does; however, exactly the same job and that is as follows. Now, we have the 3 delayed and appropriately scaled versions of x n. x n times B0, x n minus 1 times b 1 and x n minus 2 times b 2 with this. We now, add these 2 at a time first we add x n minus 2 b 2 and x n minus 1 b 1 to get this ,to this sum in turn, we add b 0 x n to get this and this of course, is now y n.

The second block is much more regular in appearance has 1 delay block less and has 2 summing blocks instead of one, but each summing block have only 2 inputs and 1 output. So, this is the more standardized form of a second order moving average filter which could easily be generalized in the manner that I am about to proceed to do.

All you do to generalize this further is to have n delays instead of just two. So, that you go on like this here of course, you will have b 2 and you will add this through another summing block you go down like this, and you have the last delay and the output of this delay would be x n minus M. This would go to b M and would go to a summing block which attach which adds it to the previous 1 this now, is an M th order moving average filter.

Now, you see moving average filters, while they are nice to look at are not really related to our differential equations. In short they are a different class together and their general form is summarized by the expression that we had earlier over here in this page namely, this is the general form of the M th order moving average filter of the discrete kind of course.

(Refer Slide Time: 17:21)



we are now, presently involved in only discussions about discrete systems moving average filters always have a finite impulse response, that is because the moving average filter simply sums up certain previous values of x n with appropriate scales, that is all it does and the number of previous values of x n that are added is simply M that many previous values of x n are being added together. That is why the impulse response of this is very easily calculated h n, which is equal to y n. When x n equals delta n is given by you just have to substitute delta n for x n on the right side, k equals 0 to M, b k delta n minus k, this is the impulse response as simple as that and therefore.

It will have only M different non 0 points in the impulse response actually, M plus 1 because even at M equal to 0 you have. So, its M plus 1 at most it could have less than M non 0 values if any of the b k's is 0 for k less than M. Then it would still be called as M th order filter, because there are M delays in the circuit M delay blocks in the circuit in the diagram, but it could have a smaller number of non 0 values alright.

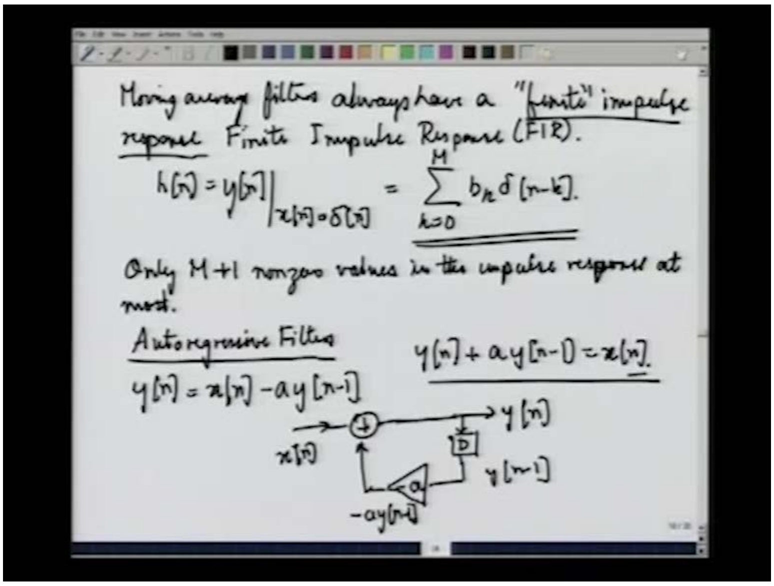
Next, suppose we want to solve differential equations like, we hope we would be able to or as we are now, concerned with difference equations. If you wish to solve them let us see, how we can solve a difference equation of the simplest kind that is the study of what are called auto regressive filters. Auto regressive filters are genuinely different from moving average filters. Moving average filters as I said have only a finite impulse response, in this sense that they have only finitely many non 0 values in the impulse response, that is why they are also called F I R filters finite impulse response filters. The auto regressive filters as it turns out will have an infinite number of non 0 values in the impulse response. An auto regressive filter the simplest case of, which would be first order differential equation. First order difference equation would be as follows an example, y n plus say a y n minus 1 equals x n. So, here we have not cross values of x n involved in the computation at least not explicitly, but certainly cross values of y n inward in this computation.

Now, one way to approach this problem would be to go along the lines that we followed, when we try to find the impulse response of a first order a system described by an equation. Such as, this that was there in some previous slide a little while ago let us just takes a look. Here is an example, there must be more examples, if we go further back here is the other example, this one is the causal version, causal solution and the next was the anti causal solution.

Now, for that causal solution or for that anti causal solution, this should be delta over here and likewise let us go to the previous one and see what we have done h n equals this is. So, we could keep recursively using older and older values of x n and not using the past values of y n, but as you saw in the process by which we obtained the impulse response of solutions for causal and non causal, anti causal solutions for difference equations. You still require some one past value of y n. So, the live way of going about solving an equation such as, this would be to express y n in terms of y n minus 1 and x n and n then y n minus 1 in terms of x n minus 1 and x n and y n minus 2 and so on.

So, if we keep doing this we will get an expression for y n that involves an infinite number of terms. Such as, this expression has an infinite number of terms because k equals 1 to infinity. It goes from 1 to infinity; this also has an infinite number of terms. Now, this is not convenient, we if it has an infinite number of terms; that means, it requires an infinite number of delay blocks, D blocks in that very very clumsy sense of the term. Any auto regressive filter would be an infinite order filter, but that is not, how we are going to look at an auto regressive filter. We are not going to treat it as some kind of a distorted or mutated F I R filter with an infinite number of delay elements. There is a much nicer and simpler way of doing it, but in order to understand this one has to temporarily live with a paradox, what I am going to assume in this case well.

(Refer Slide Time: 25:53)



Let me first rearrange the equation. So, that it is in the convenient form of y n equal to something. So, you can write this as y n equals x n minus a y n minus one suppose, this is the equation, you have then what is the best way of implementing this.

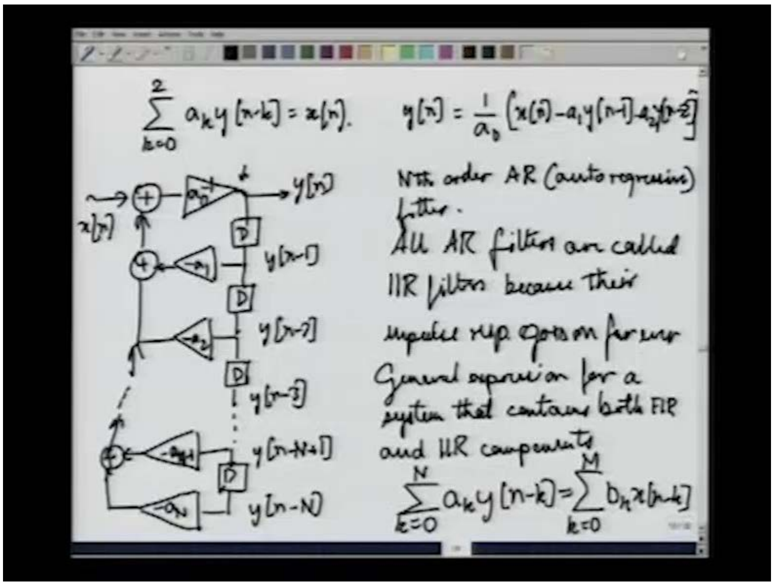
You need to know y n. You have to compute y n, but unfortunately to compute y n, you need to know both x n as well as y n minus 1. a is known now. So, you need y n minus 1 to find out y n alright, but y n minus 1 is not known either y n minus 1 is the x n minus 1 minus a y n minus two. So, you would have to know y n minus 2. So, you have to keep going further and further back, because there is always one term of y n, which you do not know, such as y n minus 2 or y n minus 3 or y n minus 4. The problem only goes further and further away, but it does not go away altogether. So, what is the solution? The solution is to pretend temporarily that you have somehow solved this equation and have actually evaluated y n, suppose you make that temporary hypothetical belief, assumption that y n has somehow been found.

Now, if y n has been found then you know the value of the sequence y for the point n minus 1 and n minus 2, and everything only then you will say that. You know the entire sequence n. So, well take this and put a delay on it, the sequence that appears at this point will give y n minus 1 this of course, is a delay block as I said it is a delay block.

Now, that you have y n minus 1. Let us, try to proceed remember this entire thing is founded on the belief that we have y n, somehow and we will just have to live with that belief for a little longer. Since, we have y n minus 1. Now, we will make the appropriately scaled version of y n minus 1, which is minus a. So, what you have, here is minus a y n minus 1. So, if you have minus a, y n minus 1 and you have x n. All you have to do is to add these 2 and the sum will itself be equal to y n. So, all I do is put up a summing block like this, and let x n going to it and minus a y n minus 1 go into it, what should I get over here according to the equation, I should get y n over here and well though I got y n. Assuming that I already had it I am happy to see that I do have it. So, I just go ahead and connect these 2 points. So, it produces y n and feeds upon itself to produce y n continually in future in perpetuity. This is how a first order auto regressive filter functions you are able to get y n, if you start, if you start by pretending that you have y n .

A second order auto regressive filter will be 1, which can solve a second order difference equation. Second order difference equation would be of the form.

(Refer Slide Time: 30:18)



Let me put it in the more general form with all kinds of coefficients in it summation k equals 0 to 2 in this case a k y n minus k equals x n. You just have to extend the previous model, a little further and write y n equals 1 by a 0 times x n minus a 1 y n minus 1 minus a 2 y n minus 2 this would give me y n.

Now, this also can be implemented, if you follow the same trick that we followed a little while ago, which is to pretend that you have y n, then you put 1 delay on this and get y n minus 1. Put another delay on it and get y n minus 2. So, you have these, all these hypothetical for the time being because y n itself is not known, but now that you have these or in a position to put in some of these coefficients this will be minus a 2 this will be minus a 1, and you have to add all these things.

So, you put a summing point over here now you have these 2 things. These 2 things plus x n complete the picture in some sense x n plus minus a 1 times y n minus 1 plus minus a 2 times y n minus 2. What is all this equal to this is, just a0 times y n, but we do not want a0 y n. We want y n. So, we just now, here put a forward looking scaling block and write here a0 to the power minus 1 that is 1 by a0.

What is do what do you find at this point you just find y n. So, you have got y n, once again. The more general equation for the n th order auto regressive filter, which we just write as AR filter, this found by just generalizing this diagram, a little bit which I will do without rewriting it all over again, I have this already. So, I will put another delay block over here. So, I will get y n minus 3 and so on … until, I come to the last delay block before, which I have y n minus n plus 1 and with this delay finally, you get y n minus n, and you take this and scale it with minus aN, this will be minus aN minus 1, minus a N minus 1, and then you add all these using the text for shortcuts, that you are already familiar with.

So, this is it you now have an, n th order auto regressive filter, which will give the output y n, if you have x n and these may be delays scale multiplier scalars like a0 inverse a1 and a2 minus a1 minus a2 and so on until the last summation minus a n, this gives you an n th order AR filter.

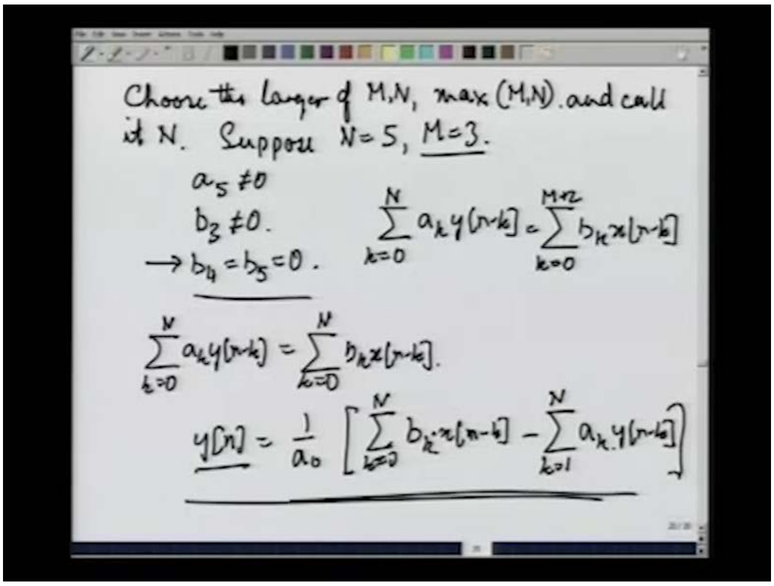
Now, see the point within AR filter, it does not even have to be n th order even a first order AR filter is full of surprises, what we want to see is whether, if the impulse response is really infinite, it is not hard to see suppose you apply x n as just delta n. Then what would happen in a system like this, delta n would add to something and come over here, and a delayed delta n would be delta n minus 1. It would get scaled and it would come back here and even though input is no longer being given this would now, act as the input delta n or rather a minus a1 minus a delta n would appear as the next output and this would go on and on, on and on, on and on. So, that you would get an infinite impulse response. An impulse response with a support that is infinite.

So, these filters are called I I R filters, all AR filters are called I I R filters, because their impulse response goes on forever that is why it is called an infinity impulse response filter or I I R filter. Now, you could in principle have a filter, which contains both F I R and I I R elements. It would of course, jointly be I I R only it would not be F I R under any circumstances, but we will see how one can combine these 2 and make a block diagram for a more general filter, which contains both an I I R and an F I R component.

So, let us look at the equation of a system, which contains both I I R and F I R components. Such a system would have a general expression like this, on the left side. We would have the part that is a difference equation that is like; the left side of a difference equation k equals 0 to N. a k y n minus k and this would be equal to a similar sum weighted sum of past values, and the present value of x n that is to say k equals 0 to M , b k x n minus k, this is what you have in the right side.

So, alright suppose, you have this, then M in general could be different from n, but it would be convenient for us to keep both numbers the same. And in order to keep both numbers the same there is a very simple trick 1 can use.

(Refer Slide Time: 40:13)



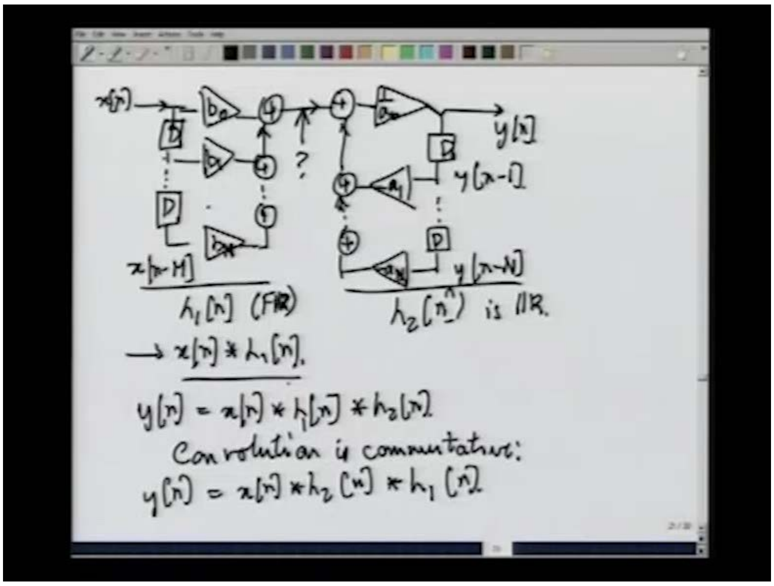
The trick 1 uses is to choose the larger of M and N that is to say max M, N. Choose the larger of the 2 and call it N, then we rewrite the equation. So, that there are exactly n terms on either side. Now, the extra terms that need to be added on the side, that was earlier possess of a smaller number of terms will essentially be terms. Where the coefficients are 0. So, suppose N were equal to 5 and M were equal to 3 then. Since N is equal to 5. I would have a 5 not equal to 0 and I cannot comment on a4 to a 0 sum of those terms might possibly be 0 on the other hand for the right side. Since M is equal to 3, we can write that b3 is not equal to 0.

There is no higher coefficient on the right side than b3, because M equals 3, but that does not stop us from simply saying that b4 equals b5 equals 0. Thus, we would write summation keeping this in mind, we would write summation a k y n minus k, k equals 0 to n, which is equal to 5 equals k equals 0 to M plus 2, b k x n minus k. Now since, m equals M plus 2, we can call both of them n and simply write summation k equals 0 to n a k y, n minus k equals k equals 0 to n , b k x, n minus k keeping in mind that this is true.

So, we now have a more convenient form of this kind of an equation which we want to solve to find y n given x n. So, how do we do it, we have to rewrite this equation. So, that we have an explicit expression for y n and that is not hard to do, we can write y n equals 1 by a 0,which is the coefficient of y n times b k summation k equals 0 to N, x n minus k minus a k summation k equals 1 to n, y n minus k note that the y n minus 0 term is not incorporated on the right side, but on the left side namely y n itself.

All other terms of the left hand side have been pushed to the right. So, that you now have an explicit expression for y n with this explicit expression, we can see how this thing can be constructed, you have 1 by a 0 into b. b k summation sorry there is a mistake here(Refer Slide Time: 45:55) this should be over here this also should be over here let us just erase this and make it more spacious this is the correct expression sorry for the mistake. Now, how do you implement this, we look at these 2 blocks as 2 separate blocks and try to find these 2 intermediate quantities the result of the summation of all the terms in the first summation and the result of the second summation.

(Refer Slide Time: 46:25)



To get the first summation, we already have a certain mechanism namely, we take x n and delay it a sufficient number of times to get the different delayed versions of the signal. So, you have here x n minus M, here you have x n. So, you multiply all these by respective scales b 0 and this will be b M.

Now, let us add all these things up b 1 adding up over here, this another sum over here and. So, on till you have a summation over here, which gets an input from bM, x n minus m. So, you have all these things this is the summation of the first set of terms.

Now, we need the second set of terms after we add these 2 there is still a little more work to be done, but let us construct the second set of terms the second set of terms if you look back has y n minus k starting from k equal to 1 to k equals capital n. So, you have various delayed versions of y n starting with delay by 1 delay by 2 and so on.

So, now I will assume that y n is available over here the same trick that I played before y n is available here. So, now, if I put a delay over here you find that y n minus 1 is available and. So, on add infinite term. So, you get a further delay and here you get y n minus Nthat is what we have over here.

Now, let us scale all each of these terms by the appropriate scale factor y n minus N gets scaled by a n minus a n actually and y. So, on up to here where y n minus 1 is getting scaled by minus a 1. Lets now, add all these terms together separately like we are about to do. So, we have added all the terms in the second summation we have added all the terms in the first summation. Now, we have to put all these together now, how do you put them together not hard you have got all these terms in the sum over here you put another summing block over here and you now have a 0 y n.

So, in order to get all this to evaluate to y n, you just now multiply by 1 by a0. So, what you have is 1 by a 0 y n and that, when added to this or rather when connected to this will give you y n, this is a completely self consistent set of blocks arranged set of blocks, which will generate y n the solution to the differential equation, which had x n as the input.

Now, we have this you see each of these blocks is essentially implementing some kind of a convolution, because it is a linear time invariant block having constants at all these summation points at before all these summation points, which are scaling the original signal and then we are adding up.

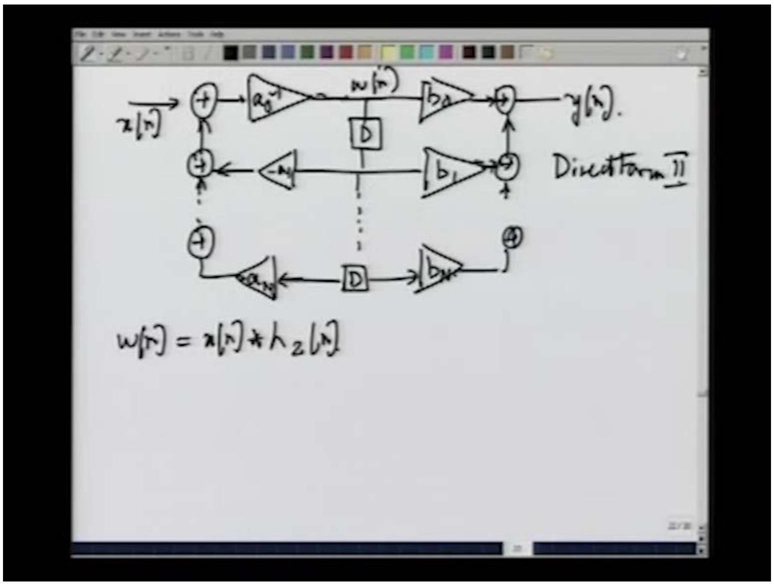
So, we are just generating a linear function or a quantity, which is linearly related to x n. Now, keeping this in mind what can we do because this whole thing is linear, let me say that this block has an impulse response, which is I will something I will call h 1 of n, likewise this will have an impulse response which I will call h 2 of n. Now, h 1 of n is F I R as you know h 2 of n is I I R.

More importantly since, these two diagrams have a lot in common we can rearrange these two diagrams, but before we before we rearrange we should be very clear about not just the fact that this is FI R and this is I I R. But, the fact that this block is essentially implementing a linear time invariant system and therefore, can be equivalent to the computation of convolution, in short if this is what we call h 1 n and this is what we call x n, what signal do we have at this point. The answer we have is that is x n, convolved with h 1 of n that is what you get over here. Now, this is another block, which is also linear time invariant and this block is now, processing this signal and producing y n.

In short, I can express y n also as a convolution of h 2 n and the input to the second block therefore, I can write y n equals x n convolved with h n, this in term convolved with this was h 1 n, this in turn convolved with h 2 n this is a total convolution of the F I R part followed by the I I R part with the input signal to get y n the output signals.

An important thing that we can now do is remember that sorry convolution is commutative. So, we can as well write y n as equal to x n, what we want is to reverse the order of the 2 convolutions and write h 2 n over here and write h 1 n over here. What does this serve, what help does this give us what do we get out of doing this. That is what we will now, see I have a block over here and I have a second block over here this is h 1 and this is h 2. We are now going to exchange their positions you apply x n and now you have to process x n with the h 2 block.

(Refer Slide Time: 54:46)



First and the h 2 block consisted of a summation and here, you had a0 inverse here after this was actually some output from here you had a delay and so on. Until you had the last delay over here, you add a set of delays. Now, this delay was scaled by a or rather minus a N. This similarly would be scaled by minus a 1 and this would get into this and so on. This is just the same block that we had before except that now, the input is x n and the output is something that we call say u n or w n still better.

Now, w n is x n convolved with h 2 n, what do we do with w n, we now put the moving average part the F I R part on this side. So, that we take this over like this and scale it by b 0 take it down through a delay and scale this by b 1 take it further down until you finally, get b n.

Now, remember that the number of coefficients in both is same in both sides are same for both the set of terms is same except that we are always free to assume as per convenient to us, whether some of the higher end b n’s are 0 or some of the higher end a n’s are zero. So, now we have this and then again construct the intermediate sums and get since, we have just exchanged the blocks, the final result of all these must still be y n, which indeed it. So, you have y n over there.

But now, let us look at these blocks, you have something called w n over here and w n has been delayed n different times for the moving average part and n different times for the auto regressive part, but it is the same signal which is getting delayed 2 times through two sets of parallel delay blocks we need only one set of parallel delay blocks. So, we could actually simplify this diagram by combining all these delays together if this is w n this will be w dash n, but w dash n is required for this as well and so on. Until here, you will get w delayed by n. You do not get w dash n over here, what you would get is w n minus 1, which is required by both sides. Finally, here you would get w n minus n that is what you would get over here.

So, let us just knock off two parallel sets of delay blocks and replace it by one set of delay blocks. So, I am going to do that by removing these parts and putting a single signal this directly goes in over here. This gets delayed by 1 time this is w n actually and this goes to both sides. So, on until you finally, have again a delay 1 input going over here and of course, the other input going over here this is a further simplification of making block diagrams for systems described by differential equations the general systems that we had auto regressive as well as moving average systems.

So, this is the general block diagram this is called a direct form two implementation and the implementation you had before, where you had two sets of blocks 2 complete independent set of delays etcetera is called a direct form one realization.

So, which is better direct form two is defiantly better, because it does the same job as direct form one, but it does it using half the number of delays that the direct form one used .