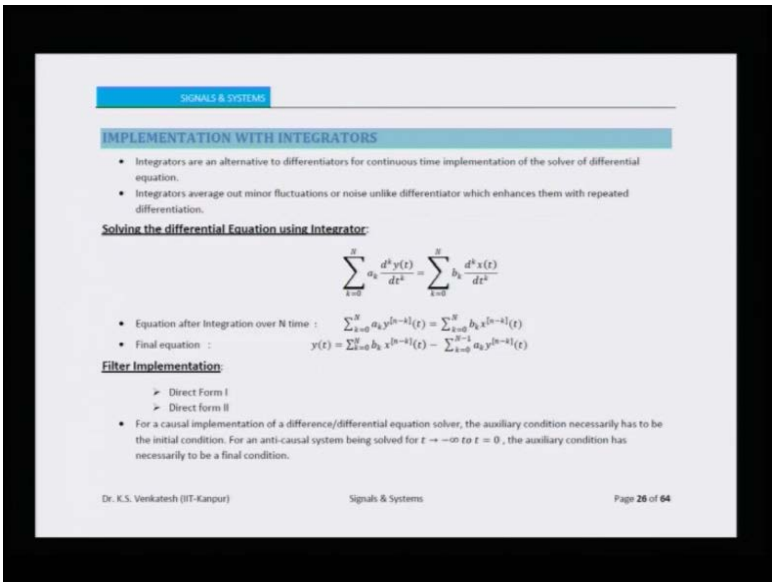
Signals and System
Prof. K. S. Venktesh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture - 23
Implementation with Integrators

(Refer Slide Time: 00:14)

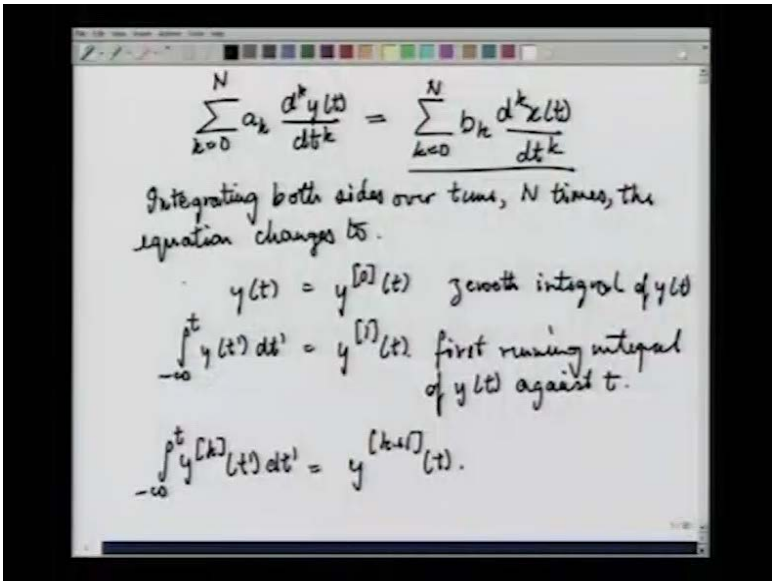


In connection with the continuous time implementations of the solvers of what would in that case be differential equations, there is one alternative implementation that can be spoken off. This alternative implementation is significant only for the context of the continuous time case, because for the discrete time case there is no problem implementing it using delay blocks of the in the manner that we have shown. For the continuous time case, however, there is a practical problem that arises and that practical problem is simply that whenever you differentiate a continuous time signal, which is corrupted even slightly by noise then the noise tends to get amplified under differentiation. So, repeated differentiation of a signal can be suicidal, it increases the noise level in signal to greater and greater extents and makes reduces the performance of the entire differential equation solving system.

The solution to that is to see if instead of differentiating we can carry out a series of integrations. Now this clearly must still solve the same equation that is a requirement that cannot be relinquished. The advantage of using an integrator is that instead of amplifying

the noise, the way a differentiation does the integration actually averages out minor fluctuations. Thus let us see if we can take the original differential equation and implement it using integrators instead of differentiators.

(Refer Slide Time: 02:21)



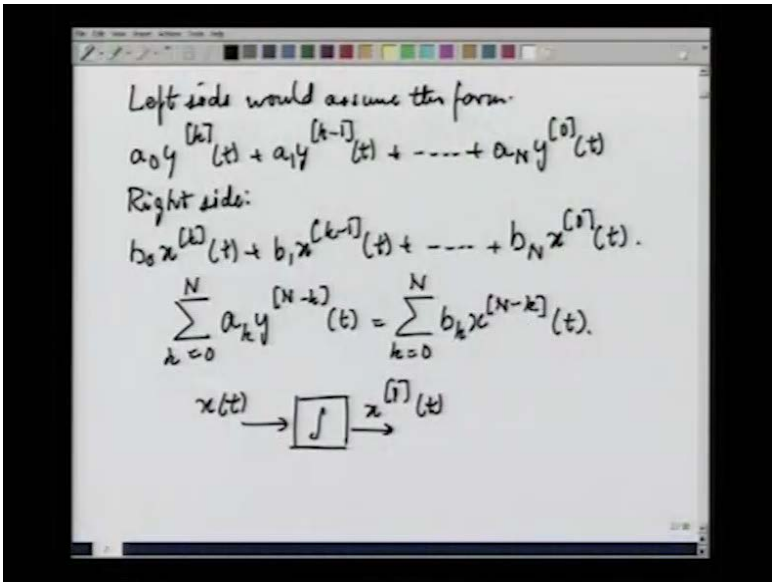
The original equation of course, only consisted of a set of terms on the left side starting from k equals 0 to N a k, and on the right side, conventionally we just had a forcing function. But in our more modern more generalized formulation, we allow also derivatives of the forcing function to exist, so that we actually write it in this form. So, this is our general formulation. Just as I said for the case of the discrete time version of this problem, the solution of difference equations the derivatives on the right side as was the case with the differences on the right side, do not in any way make the problem more complex.

Since x(t) is a forcing function is an input function its form is completely known, for computing its derivatives is extremely straight forward, there is no challenge in it at all. In fact, the entire equation, we have written here could just together, we returned to the previous form by writing this entire function as sum x dash of t where dash of course, does not represent differentiation, but just some other signal. And now with this form, it would revert to exactly the same approach that we followed earlier for the solution of differential equations, but this is worth retaining. So, let us do it.

Now, suppose we have as shown here, N derivatives a linear sum of n derivatives of the solution, adding up to be equal to a linear sum with different coefficients b of N derivatives of the input. We want to solve this without actually using derivatives if this is the constraint we put upon ourselves how do we do it. We do it by recognizing that repeated integration will cancel out the differentiation. So, if we integrate this entire equation on both sides, capital N times, what do we get? The equation now changes to now having integrated N times what we will get is the N eth derivative of y(t) will now become just y(t) because the N differentiations have been canceled by N integrations. And the lowest derivative of y t namely a naught y(t) this is the last term on the left side for k equal to 0 that will now become the N th integral of y t the integral of y t versus against time N times.

We need a notation for this because writing it with the integral notation becomes extremely clumsy. So, we will just write this in the following manner let us say that the zero eth the integration of y(t) with itself is written as y(t) itself equal to y(0) of t. The first integral would be integral minus infinity to t y(t) dash dt dash is what I will call y 1 of t. This is the first integral, first running integral of y(t) against t. Now we only need to generalize this and say that integral minus infinity to t of y(k) of t dash against t dash is what we denote by y k plus one of t. So, this is the notation we will use. With this notation let us see what happens.

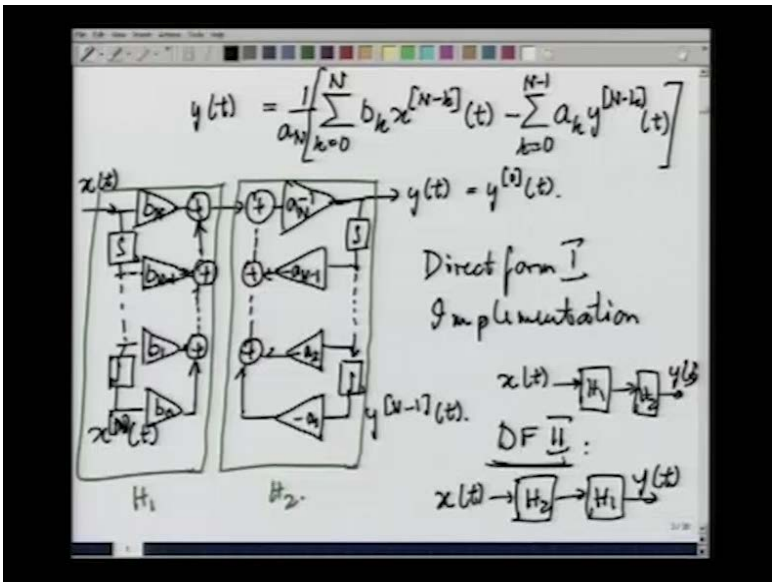
(Refer Slide Time: 09:00)



We can rewrite that equation or rather the equation that arises after N integrations would look like this. The left side would take the form, the left side will assume the form a 0 y k of t plus a 1 y k minus 1 of t plus so on up to a N y 0 of t, the last term being equal to a N of y t. Similarly, on the right side you would get b 0 x k of t plus b one x k minus 1 of t plus so on up to b N x 0 of t, where the last term again is just equal to x t. Now, let us write this in using the sigma notation to see what we get, this would be equal to say summation k equals 0 to N a k y N minus k of t equals a similar expression on the right side. This is what we would get, thus to obtain successive members of this sequence on the left side sequence of terms on the left side or on the right, we need what we will call an integrator block in place of a differentiator block.

In short, what we would require is a block that does the following a block that we have already introduced. You apply to it a function like x(t), now what you get over here is x 1 t, this is what you should get on the right side. This is a block that we will have this symbol inside to indentify this nature of this block. Now, with this kind of a block, can we do something, can we construct an expression for this yes we can. Let us see how we do that we need to isolate the term corresponding to y(t), because that is the solution that is what we want. So, how do we do that the coefficient of y(t) was a N.

(Refer Slide Time: 13:00)



So, you have a N y(t) equals the entire expression on the left side that is k equals 0 to N b k x N minus k of t minus k equals 0 to N minus 1 a k N minus k of t. Thus now the

highest integral on the right side, the lowest integral on the right side would be N y to the N minus N minus 1 that is y 1 of t, because y 0 of t is out here on the left side. So, this is what we want to implement of course, after making the change that we take away this a N coefficient over here that I am pointing out to right now, and place it over here on the right side. In order to do that we simply erase this here and write instead 1 by a N on the right side then close the rest of the expression in brackets, this is the expression we want to evaluate.

The way proceed with it is not unique, we could do it using direct form one we could use it we could develop it using direct form two, but since we already know direct form two, we probably need indulge in using direct form one with its more expensive manner of doing things. So, all we need to do now is to see how to build this system. We will assume that as usual we have y(t) available to us. So, let us say that this is y(t); y(t) which is equal to y 0 of t. What do we need we also the successive integrals of y(t) we also need successive integrals of x(t), and if you recall how we did it last time then we will recognize that the coefficients of x(t) all appeared on the right side the coefficients of y(t) all appeared on the left side. Or if that is too much effort let us initially develop the system using our conventional direct form one implementation and then make the required alteration.

So, let us go back to direct form one all. So, I have y t I need successive integrals of y(t). So, I do this and then I go on like this until I have the last integrator at which point I get y N minus 1 of t now I have to use all these. The coefficient of y N minus 1 will be a 1. So, this has to be multiplied by minus a 1 then previous term has to be multiplied by minus a two and so on until this one is multiplied by minus a N minus 1. What you would see here obviously is a N to the minus 1 right. This would give me y(t) if I already had the rest of the terms, but this has to be added to several other things. In fact, this will not come right here, this will come a little later.

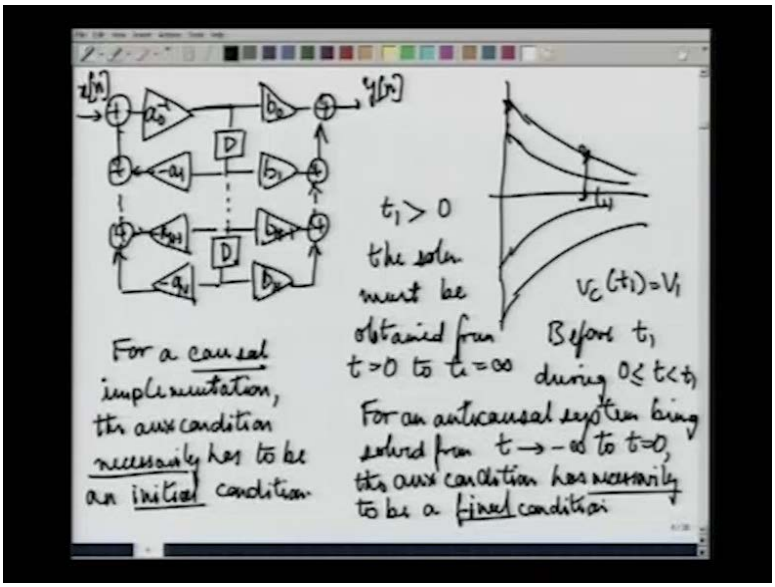
Lets first generate the rest of the part, we have to add all these terms, now let us add them. This is a sum that we are generating this should also be appended to all the terms arising out of x. So, we start with x(t) and have a series of integrals of x(t) as follows first integral and so on until the N th integral so that here. This is x(t) over here and here we will have x N of t x N of t has to be multiplied by b 0, x N minus 1 t has to be multiplied by b 1; these two have to be added and so on until here you have b N minus 1 t as the

scaling coefficient. And this gets added to all the terms met along the way. And finally, you have here b N, and this gets added to all the things that are coming from before and these whole thing gets added over here.

Now, at this sum, you have practically everything that should equal to y(t) except for the scale factor and that scale factor as we know is one by a to the N or a to the N to the minus 1 a N to the minus 1, this gives us the direct form one implementation. If you want the direct form two implementation, you know how to go about it. We do not have to summarize that thing here; you just have to literally replace this block over here to the right side. Let me just call this for simplicity, let me just call this say H 1 and let me call this block H 2.

So, the system we have now is we have x(t) going into H 1 this going into H 2 and this finally, yielding y(t). The direct form two implementation would be just to take x(t) pass it to H 2 first, and then pass it to H 1, and then getting y(t) out of it. With of course, the also implementing the savings that you get by collapsing parallel sets of integrators into a single set of integrators into a single sequence of integrators that is all you need to do get to get this direct form two implementation of the same system. There is one question that needs still to be asked. See suppose we go back to the direct form to implementation for the discrete case of an N th order differential equation difference equation; for the N th order difference equation we had a series of derives going down like this.

(Refer Slide Time: 23:31)



And on the right side we had b 1 b 0 and so on until we had b N. On the left side, we had minus a 1 onto minus a N, and we had a lot of summations. On the left side, similarly, we had x N y N coming out of here; and finally, 1 by a 0 or a 0 inverse being the term over here this is what we had. This was the direct form implantation two of an N th order differential equation solver. Where do the initial conditions or the auxiliary conditions go? Suppose we ask this question, we now have to qualify what we did and all the things we said about the auxiliary conditions.

In a theoretical level, at a theoretical level, I said that if you have a difference equation or a differential equation of N th order you will have an infinite number of solutions, and in order to pick a particular solution out of the infinite number of the solutions, you simply had to specify a set of auxiliary conditions. I further said that those auxiliary conditions could be specified at arbitrarily chosen instants of time not necessarily the starting instants, but it does make a difference when you are implementing a causal system. Whatever you built using the circuit like this that you see on the board right now is always a causal system. As I said the anti causal system is looks good on paper, but it cannot be implemented, because in a physical system one thing are forced to be causal. Now when things are forced to be casual, what happens to the specification of auxiliary conditions; the simple answer is the following.

Let me illustrate what I mean to say by drawing the series of curves that represented the capacitors charging curve for the case of the first ordered differential equation. We had a series of curves going like this and we had to pick one of them. As I said in order to pick one of them you could specify the initial condition at on the y axis that is to say the voltage axis at t equal to 0 that is to say or as I said we could specified it at any chosen instant of time over here. The problem that I now introduce to you is the following; if you choose an arbitrarily instant of time such as this; that means, that time is still in the future when this curve has to be drawn at t equal to 0; let us call this time t 1; t 1 is greater than zero. Now if you want to solve the differential equation, and get a unique solution, a unique curve that represents the solution. Then that curve should be drawn from t equal to 0 to t equal to infinity; that means, the solution must be obtained from t equal to 0 to t equal to infinity, let us say this is obviously, what we require.

Now, mathematically it is all very nice to say that the value of v c of t 1 must be equal to v 1 - that is a perfectly valid auxiliary condition. In as much as it identifies a specific

curve out of the set of all possible curves that could have solved to solve the differential equation. But before t 1 that is to say during 0 less than t less than t 1 the system that we actually manufacture and place on the table must be able to tell us what is v c of t. Since we are going to specify the condition only at t 1, the system still does not know what you are going to specify as v 1, it does not know the value of v 1. Therefore, it is not possible for the system to anticipate the value of v 1, and generate the curve accordingly because v 1 is still in the future; v 1 which is v c at t 1 is still in the future.

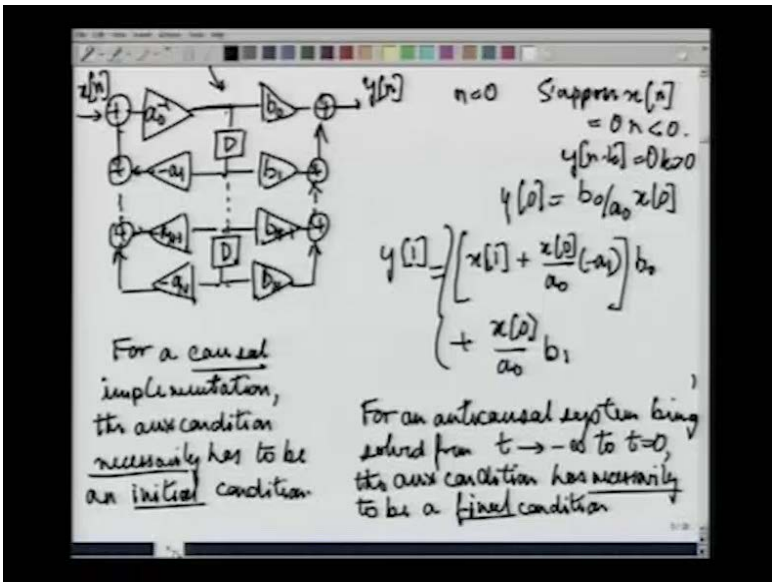
Hence for a causal implementation of a difference equation or differential equation solver, the auxiliary condition necessarily has to be initial condition. This takes nothing away from the arguments we proposed at that time, they were not wrong, they were perfectly right. It simply the fact that in a causal implementation, it becomes necessary to specify the auxiliary condition right at the beginning, so that the correct curve can be chosen out the possible curves and developed over time until t equal to infinity. This does not negate anything that was said before.

In fact, I can say things the other way around for an anti causal system. For an anti causal system being solved from p equal to minus or rather t tending to a minus infinity to t equal to 0, and anti causal system cannot look at the past it can only look at the future. Just like a causal system cannot look at the future, it can only look at the past, it only knows the past; it does not know the future. An anti causal system only knows the future, it does not know the past. For such an anti causal system the opposite constraint would hold that is to say the auxiliary condition has necessarily to be a final condition, in this case the final instant is t equal to 0.

So, you still have to specify, you have to specify y h t equal to 0, because if you specify anything before it, suppose you specify y at t 1 less than 0 then the system can be solved by the anti causal system can solve and give you the output until t one. But after t 1 the auxiliary condition has moved to the past of the present; and it cannot see the past, it can only see the future. So, if you want the auxiliary condition to be always accessible to the system, it has to be solved it has specified for the final instant that is t equal to zero. So, these are important remarks that relate the specification of auxiliary conditions to the causality or non causality of a system. So, going back to this block diagram why did I draw this, very simple, you have all these delay blocks over here. Suppose this is a causal

system that I am trying to implement and I want to understand what happens, what are the values of the system at N equal to 0.

(Refer Slide Time: 34:56)



At N equal to 0, let me just erase this part, now that we are done, at N equal to 0 , these amplifiers do not take any time to compute. What will we get as y N, let us just take a look. We have x 0 being applied over here, I want to see what will y 0 be; y 0 at N equal to 0 that is to say will be affected by whatever we are applying over here as well as the sum of the scaled values of the delayed signal. Suppose we say x N equals 0 for N less than 0, suppose we say this. Also we say that all initial conditions are 0 that is these values after all if this is y N, these are the earlier values we have over here, we also say that y N minus k equals 0 for k greater than 0. Suppose we say this then the delays over here all of them have 0 at their outputs, and there is no contribution through any of these summing points, so that the only thing that you get at y 0 equals b 0 by a 0 times x 0.

Now that we have this, we will see what happens at y 1. Now let us see what happens for y 2 to y 1. You have x 1 over here; at this point, we had x 0 by b 0 1 instant ago that is at N equal to 0, this was x 0 divided by a 0 not b 0, y 0 x 0 by a 0 by a naught, x 0 by a naught is now available at this point, x 0 by a naught is the quantity that gets multiplied by b 1 on 1 side, and by minus a 1 on the other side and gets added to x 1. So, y 1 will be equal to x 1 plus x 0 by a naught multiplied by minus a 1 that is one side of it, that is what we have over here. And this gets now scaled by this quantity b 0 and then gets

added to x 0 by a 0 multiply by b 1 this then is y naught, y naught or rather y 1 this is the value of y 1. It has incorporated the value of x 1; it has incorporated the delayed functions that have a reason in the mean time.

Now, clearly you can see y 1 uses both x 1 and x 0; y 2 will use x 1, x 0 and x 2 and so on, so that after N instance where N is the order of the system we passed N values of x we will come into the picture. And even older values will come into the picture, because of the order nature of these blocks on the left side, and the system completely starts functioning as it suppose to. But in the initial states because we chose all the delay outputs to be equal to 0, the system has developed in this fashion. On the other hand, we could have said some initial conditions, we could have said that y minus 1 is equal to this y minus 2 equals this, y minus three equal to this.

And if you look at the direct form one implementation then we can clearly see how that system would take values of y minus 1 y minus 2 and so on has initial conditions and use them to contribute to the output right from y 0 to y 1 and so on. So, this more or less closes our discussion of differential equations, and their solutions. It has been a long discussion gone into several several hours; at this point of time, we will close this discussion, because it has been sufficiently thorough and we will now proceed to discuss a totally different issue.