Signal Processing Algorithms & Architecture
Dr. Anirban Dasgupta
Department of Electronics & Electrical Engineering
Indian Institute of Technology Guwahati
Lec 23: Filter Structures – II

Hello everyone, welcome to a fresh new lecture on the topic of filter structures II. This is Dr. Anirban Dasgupta and let us get started. So, we have already covered the structures of FIR filters And now we will look at the structures of IIR filters And specifically, we will study these structures: the direct form I, the direct form II , the transposed form I, the transposed form II, cascade form, parallel form, and lattice and lattice ladder forms.

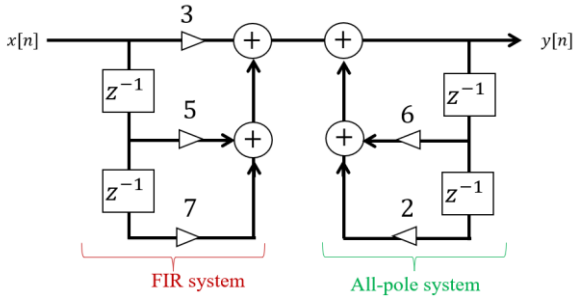
So, we have already seen that an FIR filter in the Z domain is nothing but a polynomial in Z inverse. What about the IIR filter? So, the IIR filter can be represented as a ratio of two polynomials in Z inverse. The first is $N(z)$, which is the numerator part, and the second is D(Z), which is the denominator part. Now, simply N(Z) can be written as one transfer function $H_1(z)$ or one filter, and $D(z)$ can be written as another filter, which is $H_2(z)$.

$$H(z) = \frac{N(z)}{D(z)} = H_1(z).H_2(z)$$

Now, the reason I am doing this is that we can see that $H_1(z)$ consists of the zeros of the filter $H(z)$, whereas $H_2(z)$ consists of the poles of $H(z)$. So, in other words, $H_1(z)$is a polynomial in z inverse, which is basically an FIR filter. Whereas $H_2(z)$is a polynomial, it is not a polynomial; it is the inverse of a polynomial of z inverse, or it is basically an all-pole filter. So, it is clearly shown that $H_1(z)$is an FIR filter whereas $H_2(z)$is an all-pole system. So, now we can see that since they are multiplied in the Z domain, you can actually cascade these two filters $H_1(z)$and $H_2(z)$; or in other words, our IIR filter is a cascade of an FIR part and an all-pole part.

Now, based on which comes first—whether I put H_1 first after H_2 or H_2 comes after H_1—we can get two such structures, and these are called the direct form structures.

$$y[n] = \underbrace{3x[n] + 5x[n-1] + 7x[n-2]}_{\text{FIR system}} + \underbrace{6y[n-1] + 2y[n-2]}_{\text{All-pole system}}$$



Why the direct form? Because we directly obtain them from the transfer function without doing any manipulation or simplification. So, in direct form I, what we do is basically have this filter, which is defined by this difference equation. Now in this case, you can see that this filter Y(Z) is basically some combination of the present input and some past values of the input, and this is the FIR system. and this is an all-pole system where we have the previous values of output.

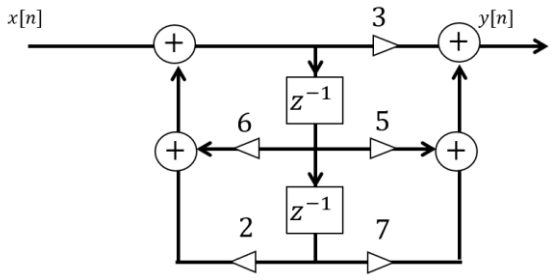
This is kind of recursion. So, this is a structure that is a direct form I where I have first the FIR system and then we have the all-pole system. Now, here you see that this FIR structure, if you see, this x[n] comes here and then this is multiplied by 3, so this becomes 3 times x[n], and then this is x[n-1]after this delay, which is multiplied by 5, so this part becomes 5 times x[n-1], and then this is 7 times x[n-2], and this all gets added to make the output of the FIR system, and this is again added to the y[n]. Now y[n] is again delayed, so I get y[n-1], which is multiplied by 6 here, so this is 6 y[n-1], and this is basically twice y[n-2], and all these are added together to form my y[n], which is this difference equation.

$$y[n] = 3x[n] + 5x[n-1] + 7x[n-2] + 6y[n-1] + 2y[n-2]$$

So here we have M plus N plus 1 multipliers. What is M? M is basically the highest lag in the FIR system. What is N? N is the highest lag in the all-pole system. So here in this case, my highest lag in the FIR system is 2, and so M is 2 here and N is also 2 here.

So, M plus N, which is 4 plus 1, is 5. So, we have 5 multipliers: 1, 2, 3, 4, and 5. That extra 1 is for the x[n] without any delay. Adders are M plus N, so these are the adders, and delays are also M+N. These are the four delays.

Now the next structure can be done if we place the all-pole filter first and then the all-zero filter in this manner. So, what is the advantage I get? I can combine the delays. So, this is very similar to the direct form I structure in terms of multipliers and adders. But the delays are being shared. So, if there were one more branch here on the output side or even on the input side, we would have to use one more delay.

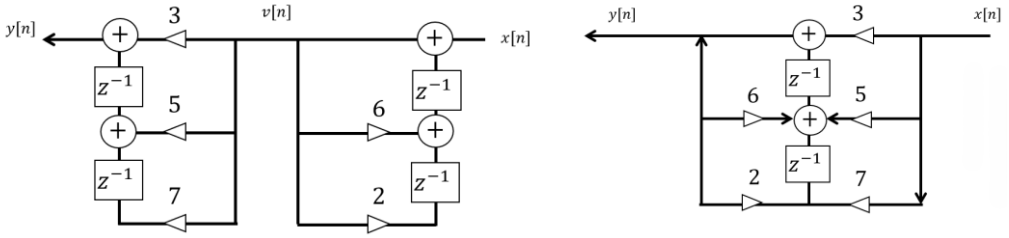


So, in general, it is the maximum of M and N. Whichever has the highest number of delays is the number of delays in your direct form 2 structure. Okay, so next is the transposed form, and this comes from the concept of signal flow graphs. In this transposed form, we will use the same difference equation,

$$y[n] = 3x[n] + 5x[n-1] + 7x[n-2] + 6y[n-1] + 2y[n-2]$$

and what changes we will make to the direct form I are that first, we will interchange the input and output, and next, we will reverse all the directions. And third, we will interchange the adders and the branching elements, and vice versa.

So, if you see here initially, if you quickly look at the direct form I, this is the direct form I. So here you see, this is the input; this is the output. So here y will come; here x will come. Second is the directions will get reversed and third, the adders will be replaced with branching elements; adders will go here, and then branching elements will come here. So, they will change their place. So, this is basically the transposed structure of my direct form I. And again, if you verify, we get the same thing. So just do a quick verification. So here I have written just an intermediate signal v[n], which shows that this is the signal in between the cascades.



So, if you use this V[n], it is easier to do the analysis. Otherwise, you will still come up with the same results and needless to say, the transposed form will be just the transposed version of the direct form II, where we do these operations to the direct form II structure. So now we have studied four structures for the IIR system. Now let us look at the cascade form of the IIR, which is very similar to the cascade form for the FIR.

$$H(z) = \prod_{k=0}^{N-1} H_k(z) = H_0(z).H_1(z).H_2(z) \dots H_{N-1}(z)$$

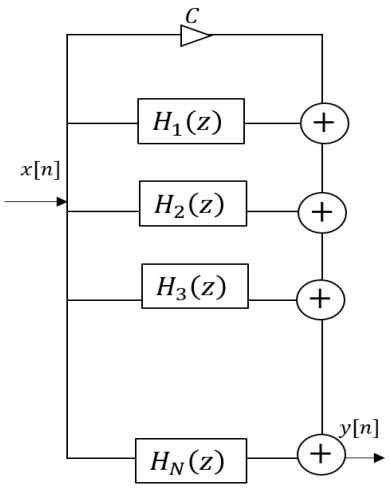
So, suppose you have a big transfer function H(Z) and this H(Z) can be factored into a lot of such filters, I say $H_k(z)$where k varies from 0 to N - 1, which means there are N cascades like this. H_0 to H_{N-1}. So, such a filter, and each is an IIR filter of smaller order, maybe 1 or 2. So in this manner, you can actually form different small IIR filters. So, this big H(Z), which is a huge filter of very high order, is broken down into small filters that are also IIR.
Typically, there may be one FIR as well that is not a big problem, but typically they are in cascades. So, what about the next form? The next form is basically the parallel form. So, what about the parallel form? So, as we know, what is parallel? So, in a cascade, cassettes are basically sequential or series forms. So here, one block is entering the signal for processing, and then the output of the first block is going to the second block, and so on. So, things are actually not happening as fast as they look because there will be lags while each stage is processing.

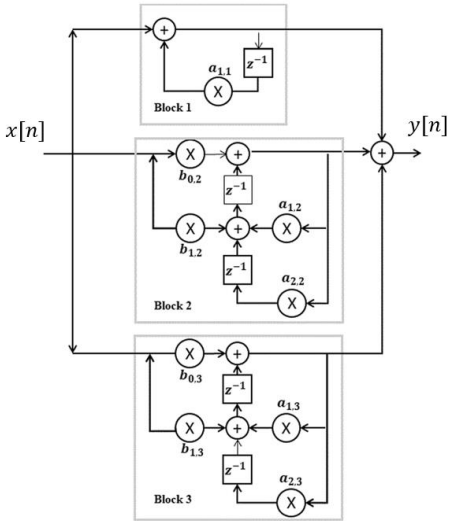
So, what about parallel? So, in our signal processing architectures, we also learned that if we implement some kind of parallelism in the architecture during the operations, then things are faster. So that brings us to the parallel form. Now the question is this: if we look here, I have this expression for H(Z). Now in cascade form, what do we get? The cascades were products of different H's(Z), right? But in parallel form, typically what we get is that they have to be the sum of some terms. If you see this equation, H(Z)can be expressed as some constant and the sum of H(Z), which I denote as $H_1(Z)$, $H_2(Z)$, $H_3(Z)$, up to $H_N(Z)$, and then there is a constant that is just a simple multiplier, and here everything is added. So, these operations are happening in parallel and hence they are faster. Now, what is most important here is that typically these P_Ks are our poles.

$$H(z) = C + \sum_{k=1}^{N} \frac{A_k}{1 - p_k z^{-1}}$$

Now the question is whether, if I am given an H(Z), I can break it down into sums. Well, the answer to the problem is that I hope most of you, or maybe all of you, have done this concept of partial fraction expansion. So partial fraction expansion means you have to divide a big polynomial into some of its factors, and this is what I have done.

So, if they are divided, like in, say, "I have one H(Z)." Which is to say something of $\frac{A_k}{1-p_k z^{-1}}$. Now this can be polynomials of Z as well, say 1 minus 2 Z inverse. So, this is basically our cascade form, right? Now, in parallel form, what we need to do is get basically the partial fraction expansion for this cascade form.



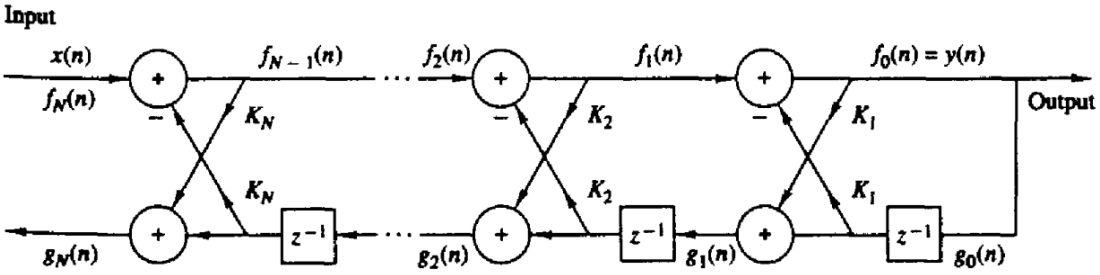


And now there are methods to obtain these values C_0, C_1, C_2, and this again may not be just constants; they may be C_z inverse plus some d, something like that. So now the question is how can you get them? So, there are rules to solve these equations to obtain the coefficients of the partial fraction expansion. Now, typically what I was saying is that, in this cascade form, what are these coefficients or values giving? These are basically giving you the poles because, say in this case, the pole is at 1, the pole is at z equals 2, and how do you know? Because at the pole, this term becomes 0. So, hence, the partial fraction expansion, if you know the poles, is a very easy way to factorize by giving the poles directly here and what you finally get is a parallel bank of single-pole filters. So, this is one example where we have a bank of three such IIR filters.

Now, so far, the direct form was present in the FIR case; we obtained two direct forms in the IIR case because there are cascades of an FIR and an all-pole filter, and by the interchange of these filters, we get two direct forms. Again, similarly, by changing these branchings, we got the transpose form. The cascade form was also present in the FIR system, and in the IIR, we also got the cascade form. We are studying something new as the parallel form in the IIR case.

But is there a possible parallel form for the FIR? Well, technically, if you look here, since there are products and there is a denominator, that is why you are able to break them into this partial fraction. But IIR, that is the IIR case, but in the FIR case, if I have my H[Z] and if I want to represent it as the sum of 2, it is nothing new; this is the same FIR filter

only because there are no denominators. So, like, if I have a polynomial in z, if you just break this as one filter and this as one filter, or this as one and this as one, it does not really change. Meaning that the direct form realization of the FIR filter is already a parallel implementation, and as such, there is no separate parallel form for the FIR filter. Now coming to the last structure, which is the lattice ladder form.



So, to deal with this filter or this form, let us begin with an all-pole system whose system function H(Z) is given as this. Now, technically, this is an all-pole filter, as I said. So, it has only the denominator part as a polynomial in z inverse. And if you carefully look at the denominator, this is very similar to the lattice structure of the FIR. So, how do I use this information? Well, this is the filter in the time domain or the difference equation, correct? Now what I will do is carefully examine the structure of this filter and if it interchanges the roles of input and output, that is, make y[n] as x[n] and x[n] as y[n], what do we see? We see that we get this expression, and what is this? This is our FIR filter.

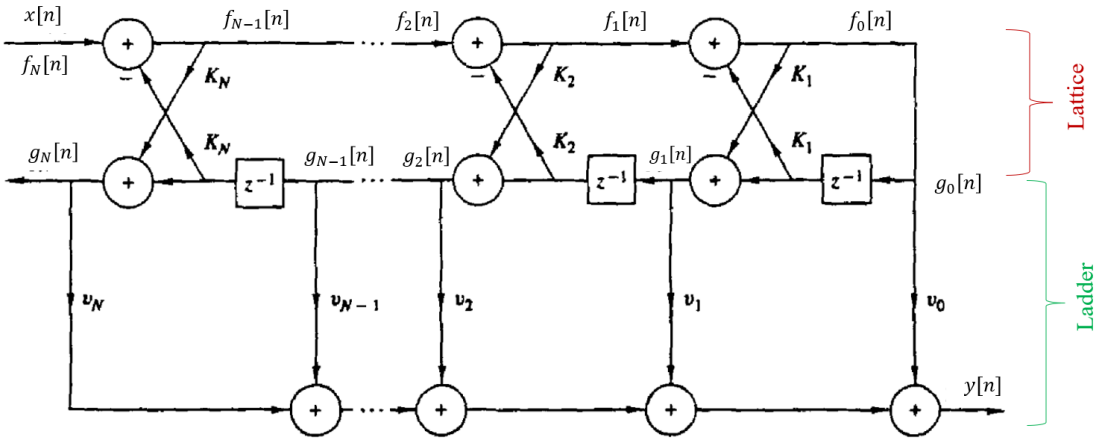
$$H(z) = \frac{1}{1 + \sum_{k=1}^{N} a_N[k] z^{-k}}$$

$$y[n] = -\sum_{k=1}^{N} a_N[k] y[n-k] + x[n]$$

Because y[n] is dependent on x[n] and our previous values of x[n] and we have already studied the lattice form for this FIR structure, haven't we? So, this means that this all-pole filter has a lattice structure that is very similar to the FIR filter. Just one condition is that the roles of inputs and outputs are changed. So, this is what I am getting. So here the roles are basically changed, and what we are getting is that this output is being fed back.

So, there is a reverse direction. So here you see that this k_1, k_2, and this k_n are computed exactly like the lattice structure of the FIR form. The only thing is that the output is fed here instead of the input. So, it is going in the backward direction, and the remaining things are exactly the same. So, this is basically what we are getting. Now again let us recall that the IIR system is a cascade of an FIR filter; this is the FIR, and this is the all-pole.

I can say that the FIR has all-zero filter, which is not a big deal, and in direct form, what we did is this: first, we created the all-pole filter, and then we cascaded the FIR version of the filter, resulting in a structure that we have already seen. Now, here in the lattice ladder form, what happens is we have completed the lattice. And what do we get in the lattice? We got the all-pole version, and with the all-pole, we now need to cascade the FIR part, and this FIR part we can just use the direct form of realization, which is basically this part. So, this is the lattice part, and this is the thing that is the direct form of the realization of the FIR part. And if you look at this, it looks like a ladder, and hence it is called a lattice ladder form or lattice ladder realization of the IIR filter.



So, you can see that this is actually very simple, although it looks very complicated. If you are given an IIR filter, first you look at the lattice part, which is basically the all-pole part. So, if you are well-equipped with the concepts of this lattice filter for FIR, then this is the same thing. This is exactly the same thing, just treating the output as input and vice versa. The remaining part is just the direct form implementation, which is the FIR part.

And this is basically your lattice ladder realization. So, thank you very much. We will meet again. Have a nice day.