[MUSIC PLAYING]

The last lecture we consider some basic structures for infinite impulse response and actually also for finite impulse response systems. In this lecture, I would like to focus specifically on finite impulse response systems and indicate, to show, that for that class of systems, there are some structures that specifically exploit properties of FIR, or finite impulse response, systems.

First of all, let's consider the general form of a finite impulse response system. One of the characteristics of a finite impulse response system, of course, is the fact that the impulse response has only a finite number of nonzero samples, or equivalently the system function involves a sum only over a finite range. So the system function in general for a finite impulse response system is of this form. And the difference equation for this system is that the form of a sum from k will 0 to capital N minus 1 of h of k, the unit sample response, x of n minus k.

We've stressed the fact, of course, in a variety of the earlier lectures, that the difference equation for a finite impulse response system, in fact, corresponds to the convolution sum for finite impulse response systems, or the coefficients in the difference equation are, in fact, the values of the unit sample response.

Now, for a finite impulse response system we can, of course, talk about the same types of structures that we talked about in a previous lecture. For example, we can generate a direct form structure from the difference equation, or equivalently from the z transform. And we see that the direct form structure corresponds to generating a weighted combination of delayed inputs where the weighting coefficients are the values in the unit sample response h of n.

So for a general direct form structure, we have x of n, x of n minus 1 at the output of this delay, x of n minus 2, et cetera, x of n minus 1 multiplied by h of 1, x of n multiplied by h of 0, down through the chain. The results added up at the output to produce the overall output y of n.

So this is what would correspond to applying the notion of a direct form structure to a finite impulse response system. Likewise, we could talk about a transposed form of this network. We could also talk about a cascade form realization of a finite impulse response filter, simply by factoring this into a product, not a poles and zeros, because, of course, there aren't poles for a finite impulse response system, but as a cascade of zeros, either implemented as first or second order sections.

Something to give some thought to incidentally is whether for a finite impulse response system you can generate

a parallel form structure like the parallel form structure that we talked about last time. The answer to that, in fact, is no. And it's tied to the fact that an FIR system doesn't have any poles. Although as we'll see later on in this lecture, there is a structure that is of the form of a parallel form for implementing FIR systems.

So the first point then is that we can apply many of the ideas of the previous lecture to FIR systems. But in addition, there are some properties of some FIR systems that lead to some specific network structures that in general can't be used for implementing IIR, or recursive, systems.

The first of these that I'd like to talk about is directed specifically toward the implementation of linear phase FIR systems. Now recall from several lectures ago-- and we've again raised this point a number of times-- that one of the differences between IIR and FIR systems is that because of the fact that the impulse response of an FIR system is a finite length, those systems can in fact be designed to have exactly linear phase, whereas the corresponding statement of course is not true for the case of infinite impulse response systems.

In particular, a linear phase finite impulse response system is one for which the values in the unit sample response satisfy a certain symmetry condition, namely the symmetry condition as I've indicated it here, h of n equal to h of capital N minus 1 minus little n, where capital N is the length of the finite length impulse response. For example, a finite, a linear phase FIR impulse response as one example might look, as I've indicated here, where I've drawn it for capital N equal to 9. And this symmetry property says that h of 0, if I substitute n equals 0, is equal to h of capital N minus 1, or h of 8. And we can see that that's the symmetry that we have.

And then likewise, the linear phase property says that as I go through the impulse response from both ends, I see identical values so that these values are identical. Then these values are identical. Then these values are identical. These, and, of course, this one is identical to itself. And that then is the general symmetry of a linear phase FIR impulse response.

Well, in fact, to stress once again that the phase is indeed linear, we can consider generating a unit sample response or impulse response h1 of n, which corresponds to this unit sample response shifted so that it's centered around the origin, in which case h of n is equal to h1 of n plus capital N minus 1. Or in this case, h of n is h1 of n shifted to the right by 4. And consequently, the frequency response for this system, or the Fourier transform of the unit sample response is the Fourier transform of the unit sample response of this system multiplied by e to the j omega capital N minus 1 over 2. That is just due to the fact that these two unit sample responses are related to each other by a linear shift.

Now, this unit sample response is even. That is it's even about n equals 0. And consequently, its Fourier transform is a real function of omega. So this is real. And consequently, the phase associated with the causal system h of n

is just simply this linear phase term, e to the j omega N minus 1 over 2. So this is an argument, again, that stresses the fact that if the unit sample response has this symmetry, then the phase of the system is linear.

Now, how can we exploit this in the implementation of finite impulse response systems? Well, let's look, first of all, at the general expression for the z transform of the system, h of z given by a sum of h of n z to the minus n. And essentially the idea in exploiting this symmetry for the implementation of FIR systems is to take advantage of the fact that because roughly half of the values in h of n are identical, we should be able to implement the system with essentially only half the number of multiplies that would be required in a more general FIR system.

Well, to show how this works, let's work through this for the case where we assume that capital N, the number of non-zero points in the unit sample response, is even. In the previous example, we'd considered capital N odd. Here, in working through this let's assume that capital N is even. And a similar derivation would apply and has gone through in the text for capital N odd.

OK, well, we can express the transfer function h of z as a sum from n equals 0 to capital N over 2 minus 1 of h of n z to the minus n-- those are the first half points, the first half of the non-zero points-- plus the sum over the second half of the non-zero points from capital N over 2 to capital N minus 1. And basically where we want to get to is to exploit the fact that these points are numerically equal to these, except that the index runs in the other direction.

Well to develop that formally, let's manipulate this sum by using a substitution of variables. In particular, let's make the substitution of variables that r is equal to capital N minus 1 minus little n, or equivalently that little n is equal to capital N minus 1 minus r. If we do that, then changing this to a summation on r, where we have that little n is equal to capital N over 2, little n equal to capital N over 2, r is equal to capital N over 2 minus 1, as I've indicated here. And when little n is equal to capital N minus 1, little r is equal to 0.

So this gets converted to a sum from r equals 0 to capital N over 2 minus 1. h of what was n, which is now capital N minus 1 minus r substituted in here. And then capital N minus 1 minus r again substituted for little n. Basically, what we've done is reduce this to a sum over the same limits as we have here and with the indices running in the opposite direction.

And now, we can recognize that because of the symmetry property, because of the fact that we're talking about a linear phase system, h of capital N minus 1 minus r is equal to h of r. That's the symmetry property that corresponds to stating that the system is linear phase.

All right, well, so here we have a sum from n equals 0 to N over 2 minus 1. Some things involving h of n. Here we have a sum from r equals 0 to N over 2 minus 1, some things involving h of r. This index of summation, of course,

is just a dummy variable. And we can change it back to n if we want to, which we do.

So rewriting these two sum, in particular rewriting this one in terms now of a summation on n, this becomes the sum from n equals 0 to capital N over 2 minus 1, h of n z to the minus capital N minus 1 minus little n. This sum, of course, stays the same. And now, because we have h of n appearing in both of these sums, we can combine this together into a single summation, from 0 to N over 2 minus 1 of h of n, the values in the unit sample response, times z to the minus n plus z to the minus capital N minus 1 minus little n.

Notice now that we only have in this summation half the terms that we had originally. In other words, there are capital N over 2 different coefficients that we're using, whereas in the direct form implementation, there were capital N coefficients that we were using. And here, we're exploiting the symmetry in the coefficients.

Well, let's see what this implies in terms of a filter structure. Here, I've simply rewritten, again, the general form for the transfer function assuming that the system is linear phase, as we've just previously worked out. This is just simply a rewrite of the previous expression.

And let's look at this, for example, I had assumed that that capital N was even. So let's choose as an example capital N equal to 8, in which case this expression becomes a sum from n equals 0 to 3, h of n, z to the minus n plus z to the minus-- capital N minus 1 is 7, so minus 7 minus n.

Well, how many delays are involved in implementing this system function? Apparently, there are seven delays involved whether or not the system is linear phase. And that shouldn't be surprising because the length of the impulse response is still 8. So seven delays in general are required, but only four coefficients are required.

And to see what structure is implied by this, we recognize that h of z is constructed by summing a system which corresponds to a delay. Looking at this expression for n equal 0, no delay. n equal 0 substituted in gives us a transfer function of 1. Plus a delay with n equal 0 substituted in of 7. The output of those two delays added together and multiplied by h of 0, and consequently, if we consider a chain of delays as I've drawn here, we have the output before we go through any delays added to the output from the seventh delay-- that's looking at this expression for n equal 0-- and the result of that summation multiplied by h of 0 to form the first term in this sum.

To form the next term in the sum for n equals 1, we have the output of one delay plus the output from the 7th minus 1th delay. That is the 6th delay. So we have this output, the output from the first delay, and this output, the output from the 6th delay. And now that is multiplied by h of 1 coefficient, which is h of 1, to form the next term in the sum.

And if we go through the remaining terms in the sum, we have the output from the 2nd delay and the output from the 5th delay added together. That multiplied by h of 2. The output from the 3rd delay and the output from the 4th

delay added together and that multiplied by h of 3. That forms the individual terms in this sum. And now we want finally to add those together. So we can simply add those together at their outputs to form the output y of n.

So here is a structure which implements a finite impulse response system of order 8, or of length 8. It involves seven delays, which of course the direct form also involves. But the important aspect of it is that it involves only half the number of multiplies that the direct form would normally require. Taking advantage of the fact that it's a linear phase system then, we're able to implement the system with half the number of multiplies than would be required in general.

All right, so this is one structure. I stress again that it's not the only structure for implementing a linear phase system. Clearly, we can generate another structure, linear phase structure, by simply applying the transposition theorem to this structure. And we would end up with a transposed configuration. And an example of that is shown in the text. I won't bother going through it. But the important aspect here is that we were able to exploit-- in the case a finite impulse response systems we're able to exploit the linear phase characteristics of the system, if we're talking about a system that is linear phase.

The second structure that is peculiar to find an impulse response systems and in some instances has some advantages is a structure that is often referred to as a frequency sampling structure. The reason for that being that the coefficients in the structure correspond to samples of the frequency response of the system. And I'll indicate in a few minutes why that might be an advantage. But first let's look at what the specific structure is that's involved.

A frequency sampling structure is derived by essentially implementing the system, or inquiry as to what the implementation would be, if we specify the system in terms, rather than of the unit sample response, of the discrete Fourier transform coefficients. We recall, and I hope that this is second nature to you by now, in general we have a unit sample response. This is a finite impulse response system. So this is a finite length sequence.

So we can describe it in terms of it's discrete Fourier transform. And the discrete Fourier transform coefficients are referred to as h of k. Or we can describe it as we can describe almost all sequences in terms of its z transform, or a system in terms of its system function.

The z transform is the sum of h of n times e to the minus, n as it always is. And the unit sample response can be expressed in terms of the discrete Fourier transform coefficients in terms of the sum of H of k W sub N to the minus n k. We have the 1/N out in front. And again, this term R sub capital N then to just simply truncate h of n after capital N points. That is to extract one period from the periodic function that this sum generates.

Now, we can substitute h of n expressed in terms of its discrete Fourier transform into this expression. The R sub

N of n will disappear because we're only summing from n equal 0 to capital N minus 1. And if we substitute this in here and interchange the order in which we evaluate the sums, then the resulting expression for the z transform for the system function is the sum from K equal 0 to N minus 1 of the DFT values. And this term and this term are involved in the sum on n. And they result in the sum from 0 to capital N minus 1 of z to the minus 1, W capital N to the minus k. That raised to the n-th power.

Well, this is the kind of summation that we've been carrying out again and again, particularly in talking about finite length sequences. And hopefully, this is another kind of summation that by now you carry around in your hip pocket. It's the sum from 0 to capital N minus 1 of something of the form alpha to the n, where the alpha happens to be that. And that sums up to 1 minus z to the minus capital N times-- let's see, 1 minus z to the minus capital N and actually W sub capital N to the minus k capital N.

That is it sums up to this alpha raised to the capital N-th power divided by 1 minus W sub N to the minus k times z to the minus 1. Well, what does W sub capital N to any integer multiple of capital N? That's just simply equal to unity. So this term disappears. And we end up simply with 1 minus z to the minus n divided by this.

All right, so combining this together, we can then express h of z as 1 minus z to the minus capital N, we have this 1/N, which we have to account for somehow, and then finally a sum from 0 to N minus 1 of H of k, divided by this factor. And so this is an expression for the z transform of the system, or the transfer function of the system, where the coefficients in that expression are the discrete Fourier transform values H of k.

The discrete Fourier transform values, as we've stressed several times, in fact are samples on the unit circle of the z transform. So these coefficients, the H of k's are in effect samples of the frequency response of the system. In other words, they are frequency samples. And consequently, the structure that this implies is usually referred to as a frequency sampling structure.

Well, let's look at what the specific structure is. We can view this as a cascade of the system that implements this term in cascade with the system that implements this summation. This term is simply implemented by a gain of unity in the forward direction, and then a gain of minus z to the minus capital N-- I happen to have the arrow here in the reverse direction, which is wrong. This is as an arrow, again, in the forward direction. That just simply implements the system, 1 minus z to the minus capital N.

And that I guess makes enough of a mess out of it, so that hopefully you can at least see what direction that arrow goes in.

This then implements that piece. To implement this sum, well, this sum implies a parallel combination of systems, each system implementing one of these pieces. So if we look at this, say, for k equal 0, we have a coefficient of 0

as I've indicated here and a pole. And where is the pole? Well, the pole for k equal 0 is at W sub N to the minus 0. For k equal to 1, it's at W sub N to the minus 1 and on through.

So the system that implements this sum is then a parallel combination of first order poles, where the coefficient in the feedback path is W sub N to the minus k. And the gain in the forward path, which can be placed either before or after the pole, are the frequency samples H of 0, H of 1, down through H of n minus 1.

And then, of course, we have the gain of 1/N, which has to go someplace. And for now, let's just leave it at the end.

I stress also that these, of course, are complex poles, because W sub N to an integer power is in general complex. We can, of course, redraw this structure by combining complex conjugate poles together. And the structure that results is indicated in the text. I won't bother to do that here. But there are a couple of points about this structure that I'd like to draw your attention to.

First of all, we started with a finite impulse response system. Obviously, we've ended up with a finite impulse response system. But the system appears to have some poles in it. That is its a recursive structure, and it has in fact capital N poles involved in the implementation, although presumably the overall system function has only zeros.

Well, how can that be? Very simply because if you look at this term, this term has capital N zeros equally distributed around the unit circle in angle. And in fact, a zero is occurring exactly at the location where each of these poles occur. So that, in fact, this piece in cascade with this cancels out this feedback loop, or this pole simply cancels out one of the zeros from this term. This one cancels out another zero and on down through the chain.

So in the overall system function, there are no poles. There are only zeros. And what happens in effect is that these poles are used to cancel out different zeros appearing in this first path.

That suggests one of the problems, in fact, with the frequency sampling structure because it requires poles, as I've drawn it here actually on the unit circle-- in an actual implementation obviously that would be slightly of the unit circle-- but it involves implementing poles to cancel out the zeros appearing in this term 1 minus z to the minus n.

And that has some drawbacks from a number of points of view. In fact, it has some drawbacks-- we won't analyze this, but it turns out to be true that due to finite register length effects there are some drawbacks.

But there are also some advantages to this kind of structure. And the advantages in some situations outweigh the

disadvantage, the main disadvantage, of canceling out zeros with poles or poles with zeroes.

One of the first advantages is that the coefficients that are involved involve frequency samples. Now, in many situations what we would like to implement is a system in which perhaps a large number of the frequency samples happen to be zero. Imagine for example, implementing a very narrow band, low pass or bandpass, filter and specifying the filter in such a way that the frequency samples in-band are non-zero and the frequency samples out of band are equal to zero.

In that case, most of the coefficients are coefficients which are zero coefficients. And consequently, there will be only a few of these paths that we have to implement, so that in that situation where perhaps a large number of frequency samples are zero-- in fact, the number of multiplications involved in implementing the system, and in fact, the number of delays involved in implementing the system, might be significantly reduced.

Another advantage to a frequency sampling structure is the fact that it is basically a parallel form structure. And that can be exploited in terms of implementation of a large number of filters, or in terms of the implementation of a filter bank. Let me state more specifically what I mean by that.

In this system function, this piece and this piece, except for the H of k's, are tied only to the fact that the unit sample response of the system is of length capital N. So that if I implement a large number of systems whose impulse response is of length capital N, the systems will look identical from the input up to the point where I do the multiplication by the frequency samples.

Consequently, if I were to implement a large number of filters simultaneously. I can use in common for all of those systems the part of this structure from the input to the output of the poles. And then simply draw off those outputs, multiply by the required frequency samples, sum the outputs, and that implements the filter.

Consequently, I'm able to share among all of the filters in the filter bank these delays, these delays, and the multiplications by powers of W. And the only additional coefficients and the only additional pieces that come in that differ between the filters and the filter bank are the coefficients that I multiply these outputs by. I need to multiply by the coefficients and then form the linear combination.

So that then are the two primary advantages to frequency sampling structures. And in fact, frequency sampling structures have found application primarily for those two reasons.

Well, there are lots of other structures that one can talk about, both for a finite impulse response an infinite impulse response systems. And as I stressed at the end of the last lecture, the discussion of structures in general and the kinds of structures that can be generated is very vast. In fact, it seems that new structures are generated faster than they can be evaluated. And I would like to terminate the discussion of specific structures with just the

ones that I've shown you in the previous lecture and the ones that I've shown for finite impulse response systems in this lecture.

To conclude that discussion of digital networks and digital filter structures, I'd like to focus finally on just a few of the issues, one primarily, involved in the comparison of filter structures. Now we've talked about different structures. One of the things that obviously is true is that from a strictly mathematical point of view, all of these structures, different ways of implementing the same transfer function, obviously implement the same transfer function. That is algebraically or theoretically, all of the structures from an input output point of view are indistinguishable. In other words, they generate the same transfer function.

On the other hand, there are a variety of issues when one considers the practical implementation of digital filters that have to be considered in choosing different filter structures. One of them I suggested last time to be the number of delays involved. And we saw the difference between non-canonic and canonic form structures. Delays mean registers. And registers mean hardware.

In talking about linear phase filters, we saw that if we exploit the linear phase property, then we can reduce the number of multiplications involved by half. And multiplications, again, mean hardware or speed or something of that sort in terms of the implementation.

Another common characteristic that can be focused on in comparing structures is what you could refer to as modularity. It's particularly important in hardware implementation of digital filters, where perhaps you would like your structure to have stages in it that are identical, so that the implications in terms of the hardware are that a section or sections can be multiplexed and therefore used over and over again.

And another issue, which is the one that I'd like to direct our attention to finally are the effect of finite register length on the characteristics of the system. And these characteristics, or the finite register length effects, in fact, are very heavily dependent on the filter structures that are used.

There are two kinds of finite register length effects. One of them has to do with the fact that arithmetic in a digital filter, specifically in a recursive digital filter, has to be done, of course, with finite register length, and it has to be truncated. In other words, multiplications, the result of B bits times B bits results in 2B bits. If this goes on indefinitely, obviously the register length grows indefinitely.

And so the results of arithmetic have to be truncated or rounded. And this generates some noise or essentially noise in the filter output. And the second thing is the fact that coefficients have to be represented in finite register length. That is coefficients or parameters have to be truncated.

What I would like to do in closing the lecture is just briefly give you a glimpse into some of the issues of the coefficient, or parameter, quantization and not delve any in any more detail into the relatively complex issue of finite register length effects. But let's focus just a little bit, just a little, on the effects of parameter quantization. In particular, we have, of course, the general form for a system function as a ratio of polynomials in z to the minus 1, where the denominator polynomial, as we've been writing it, is of the form 1 minus the sum of a sub k times z to the minus k.

Well, let's imagine that what we would like to do or that it's reasonable to tie the investigation of H of z in terms of parameter errors to the accuracy, with which the poles of the transfer function can be placed. We can express this polynomial, the denominator polynomial, in factored form, where, of course, each of the factors involves the poles of the system explicitly.

And now as we talk about implementing this system, the coefficients in the denominator polynomial, and in the numerator polynomial also, have errors due to the fact that they have to be represented in a finite register length. So we can represent that as the coefficient that we actually arrive at is the coefficient that we really want plus some error.

And consequently, the pole location, the location of each of the poles, the i-th pole, for example, is where we want it plus some deviation. Now, we can examine what this deviation is by essentially assuming that the variation in these parameters is small. And expressing this error in terms of the partial derivatives of the pole locations expressed in terms of the coefficients a sub k. In other words, we can express the movement of the pole in terms of the movement of the coefficients as a sum of the partial derivative of the pole, the i-th pole-- that's the one we're talking about here-- with respect to the k-th coefficient times the error in the k-th coefficient.

Well, this can be tracked through-- and this is done in the text. I won't go through the algebra-- but basically, the result is that this partial derivative can be expressed in this form where the important thing to focus on is that what's involved in the denominator is the product of the difference in the pole locations. Let's assume, for example, that all of the poles are clustered inside a circle of radius epsilon. In other words, the largest distance between the poles is epsilon.

What that says then is that when we substitute that in for this expression, the absolute value of the change in the pole location with respect to a change in the coefficient has in the denominator epsilon raised to the n-th power. Well, what does that mean?

What it means is that if the poles are clustered within a radius within a circle of radius epsilon, for any given pole the higher the order of the polynomial that we use to implement the pole, the greater the sensitivity of the pole location with respect to the coefficient. So this says that if we implement the poles of the system, or when we

implement the poles of the system, it obviously is better to implement the poles through a lower order polynomial rather than through a higher order polynomial.

What does this mean in terms of the filter structures we've been talking about? It means basically that the direct form structures are the most sensitive, that the cascade form structures are certainly less sensitive than the direct form structure, and the parallel form structure in terms of the implementation of the poles have the same type of hypersensitivity as the cascade form and likewise are less sensitive than the direct form.

Even in the cascade form structure, though, there's some flexibility. And there are some issues with regard to how the poles are implemented. And let me close the lecture by just giving you some indication of what some of the-- in particular what two of the possibilities might be.

Based on the discussion that we've just gone through, the indication is that we would want to implement, or it might be it advantageous from the point of view of parameter quantization, to implement a system as a cascade or parallel combination of first or second order sections. And let's make them second order sections.

Let's look once again at the implementation of a simple second order pole pair. I've neglected the implementation of the zeros. But clearly the point that I'm about to make extends also for the implementation of two poles and two zeros.

Here, I have implemented in what we had referred to as direct form, a second order section which we might use in a cascade form or parallel form realization, which implements the complex pole pair at radius r, an angle plus, and minus theta. In the actual implementation of a cascade or parallel form structure using a second order section of this form, it is these coefficients that are quantized to fit within a finite register length.

We see then that the quantization of the coefficients for this particular network form corresponds to quantizing the parameter r squared and quantizing the parameter 2r cosine theta. What this implies in the z-plane is that for, let's say, a choice of register length for this particular case equal to 3, for 3-bit register length, the pole locations which can be implemented with that system are restricted to fall at the intersection of the grid points that I've indicated here due to the fact that quantization of the parameter r cosine theta implies that we must lie along lines equally spaced along the horizontal axis.

Recall that r cosine theta is just the real part of the pole location. And quantization of the parameter r squared corresponds to the requirement that the poles must lie along circles, which are spaced according to the quantization of r square. So the realizable pole positions with a second order section in the form that I indicated on the previous slide are given by the constellation, or by the intersection of the grid points, that we have shown here.

Another possible implementation for a second order pole pair is the so-called coupled form implementation. I won't justify here by inspection that the transfer function is given by the same expression as for the direct form structure that I showed previously, but, in fact, it is. And the poles in this case fall at, again, are equal to e to the j theta and r e to the minus j theta. But now, the coefficients we see are r cosine theta and r sine theta.

Quantization of r cosine theta implies, as it did previously, that the poles must lie along lines equally spaced along the horizontal axis. Quantization of r sine theta implies that the poles must line along lines equally spaced along the vertical, or imaginary, axes. And so for that case, the choice of realizable pole positions are governed by the constellation of grid points that I've indicated here.

So this is just simply, this last issue, is just simply an introduction to the fact that even given the notion that the cascade form structure is better than the direct form structure, there are additional issues involved in specifically how to implement a pole pair, or a pole zero combination in order to account for parameter quantization effects.

There are lots of remaining issues, as you'll see in the text, related to both the issues of digital filter structures and the effects of parameter and arithmetic finite register length effects, which we won't be going into in this set of lectures. But you should at least be aware be aware of their existence. In the next several lectures, we'll begin on a new topic, namely a discussion of some of the issues involved in the design of digital filters. Thank you.

[MUSIC PLAYING]