[MUSIC PLAYING]

PROFESSOR: When we introduced the discrete Fourier transform, one of the important aspects of that transform that I stressed was the fact that the discrete Fourier transform, as opposed to the Fourier or z transform, actually has implications in terms of implementing discrete time linear systems. I commented, or suggested, that in fact, a linear system, or a digital filter could be implemented explicitly by computing the discrete Fourier transform of a unit sample response and an input sequence, multiplying these transforms together, and then computing the inverse discrete Fourier transform.

There is a very efficient algorithm, in fact, for computing the discrete Fourier transform. And so it will turn out, and we won't see this right away, but we'll see it toward the end of this set of lectures, it will turn out that, in fact, that's a very efficient way, often of implementing a convolution or implementing a digital filter.

Well, in the last lecture, we saw that computing discrete Fourier transforms, multiplying those together, and then computing the inverse discrete Fourier transform doesn't give us a convolution in the usual sense. In other words, it doesn't give us a linear convolution, it gives us a circular convolution. And one of the important considerations in using the discrete Fourier transform to implement a digital filter or a discrete time system is to account for this circularity, or to modify the circular convolution so that it looks like a linear convolution.

So in this lecture, what I would like to concentrate on, in fact entirely for the whole lecture, is a discussion of circular convolution, with the hope that at the end of this lecture there will be a very intuitive feeling for what circular convolution means. And then also, how you can use circular convolution to implement a linear convolution, and consequently, to implement a discrete time linear system, for example, to implement a digital filter.

Well, first of all, to remind you of the convolution property that we derived last time, we found that multiplying discrete Fourier transforms leads to a circular convolution, which we denoted with an n and a circle around it, meaning an N point circular convolution. And what that basically corresponded to was a periodic convolution of two periodic sequences, which are generated from these finite length sequences, by periodically repeating them with a period of capital N. And then finally, extracting one N point period from that. The result being the circular convolution of x1 of n with x2 of n.

Notationally, also sometimes we wrote this in a slightly different way where the periodic sequence, x1 tilde of m or n could also be written with a modular notation, a double parentheses, and a subscript capital N, indicating that x1 of n or m-- x1 tilde of m is generated from x1 of m by simply taking this argument, modulo capital M-- capital N.

So this is an alternative way of writing this. And finally, I pointed out in the last lecture, that if we just look at this expression, we see that x1 tilde of m, or equivalently x1 of m, modulo capital N, only involves values of this argument from 0 to capital N minus 1 and consequently as an alternative, we can replace here x1 of m. So this was an expression for circular convolution of the two finite length sequences x1 of n and x2 of n.

To remind you of how the mechanics of the circular convolution are carried out, I showed you last time the circular convolution of one finite length sequence, x1 of m and x2 of m. And let me re-stress a couple of important points. First of all, we form from x1 of m, rather from x2 two of m, x2 of n minus m modulo capital N. And for example, if little n equals 0, so that we're looking at x2 of minus m modulo capital N, that corresponds to flipping this over and then taking the values, module capital N, which basically corresponds to flipping this over, the points that come off the end get wrapped around back onto this side, as I've illustrated here. Then if we shift, and I've indicated with these dashed lines what would correspond to the periodic extension of this, although it's the interval from 0 to capital N minus 1, which is the interval that I want to focus on.

All right. Well, now as we change n, if this is generally n minus m, that corresponds to a shift left or right of this sequence. And if, let's say we look at x2 of 1 minus m, modulo capital N. That corresponds to shifting this sequence to the right by one point. And as this point falls off the end, it wraps around to the other side, which is what I've illustrated here. And then if we shift so that little n is equal to 2, another point wraps around the end, as I've indicated, et cetera. And that's the kind of shifting that gets carried out. In other words, it's a circular shift that's implemented.

To form the convolution, then for example, for little n equals 1, we multiply this set of values by this set of values and carry out the sum. And as I indicated last time, it's straightforward to see that the answer comes out to be unity over the interval 0 to capital N minus 1. But the important point, and the important picture that I tried to stress last time, was the notion of thinking of the sequences as being wrapped around a cylinder, x1 of n wrapped around a cylinder, x2 of minus n wrapped around a cylinder, one cylinder put inside the other, and then as we form the convolution, the cylinders are turned with respect to each other, and the values multiplied-- successive values multiplied and then added around the cylinder. And that's the picture of circular convolution.

Now there are other ways of looking at circular convolution. And one of them can be suggested by referring back to this view graph. We saw that we could express the circular convolution in this form, or equivalently, what this means, what this equation means, or can be interpreted as, is as the linear convolution of x1 of n and x2 tilde of n, or equivalently, x2 to of n modulo capital N, and then one period extracted from that. So in fact, another way of looking at circular convolution would be to form the linear convolution of x1 of n with the periodic counterpart of x2 of n, carry out that linear convolution and then extract one period, which is what this function r sub capital N of n is doing.

How can we get the periodic sequence from x2 of n? Well, we could do that, for example, by convolving x2 of n with a pulse train where the spacing of the pulses is capital N. And that's illustrated on the next view graph. So that from the equation that we just saw, one possible way of thinking of circular convolution is as the generation of the periodic function x2 of n. By convolving x2 of n with a pulse train, then a linear convolution of that with x1 of n, and then finally, extracting one period, which is what this function, multiplying x3 tilde of n does.

So we could generate a pulse train by taking a system whose unit sample response is 1 at a spacing of capital N, and it's 0 otherwise. Kicking that system with the unit sample, in which case, we get this pulse train out. Using that as the excitation to a system whose unit sample response is x2 of n, and what we'll get then is x2 of n convolved with this pulse train, or x2 of n repeated at intervals of capital N, but that's x2 tilde of n, that's the periodic sequence that we want. That's the periodic counterpart of x2 of n. And so that is what we have here.

Using that as the input to a system whose unit sample response is x1 of n, and then the output is the periodic convolution of x1 of n with x2 of n. We extract one period and we get finally, the circular convolution of the finite length sequences x1 of n an x2 of n.

Well, the fact that we could view the circular convolution in terms of a cascade of systems of this type leads to a very important and interesting interpretation of circular convolution. In particular, let's look at these systems, cascaded as they are here. So we have then, the cascade of these three systems to generate the circular convolution, which is x3 of n, the circular convolution of x1 of n and x2 of n. So this then is three systems linear shift invariant systems and cascade.

And one of the important things that we know about linear shift invariant systems which are cascaded is that the order in which they're cascaded is irrelevant. That's something we showed in one of the early lectures. So in fact, we can interchange the order of these systems any way that we want to. And in particular, let's interchange the order of these systems by taking this system which generates the pulse train, putting that at the end of this cascade chain so that instead of this cascade then, we have x2 of n cascaded with x1 of n, cascaded with p sub capital N of n. And then of course, r sub n to extract the single period.

Well, what does this mean? This means that I can equivalently form the circular convolution of x1 of n and x2 of n by first forming the linear convolution of x1 and x2. That's of course, what the impulse response of these two systems and cascade is. And then using that as the excitation to a system whose unit sample response is a pulse train. So I can think of this then as the linear convolution. That convolved with a pulse train, one period extracted, and the result is a circular convolution. That's a very useful and important interpretation of circular convolution. And the way to sort of say it in words, this isn't exactly correct, but more or less the way to say it in words is that the circular convolution can be formed from a linear convolution plus aliasing.

You see this linear convolution in general, is going to be longer than capital N. The output of this system is this linear convolution repeated over and over again, added up over and over again with a spacing of capital N. So there is an interference, if you want to call it that, between the successive replicas of this linear convolution as they come out of this system. That's the aliasing, that's why aliasing is stuck into that phrase. But the way to summarize it or think about it is as circular convolution corresponding to linear convolution plus aliasing.

So here it is said again, circular convolution equals linear convolution plus aliasing. What I mean by that is that if I have the linear convolution of x1 of n and x2 of n, which I'm denoting by x3 hat of n, and I want the circular convolution, the N point circular convolution of x1 of n with x2 of n, I can get the circular convolution from the linear convolution by taking the linear convolution and convolving it with a pulse train with a spacing of capital N, or equivalently adding it to itself, delayed by capital N, 2 capital N minus capital N, minus 2 capital N, et cetera, and then extracting a single period.

And often when you carry out a circular convolution of two sequences or want to examine properties of circular convolution, the notion or interpretation of circular convolution as linear convolution plus aliasing is an extremely useful sort of picture to have. So it's the kind of phrase that you should repeat to yourself two or three times for a week before you go to bed. And in a short time, you'll understand the relationship between circular convolution and linear convolution very well.

Well, let's look at a couple of examples of this. Here is a sequence, x1 of n, it's a six-point sequence. And let's carry out the circular convolution of this sequence with itself, that makes it a relatively straightforward example. And let's do that by using the notion that circular convolution is linear convolution plus aliasing.

So we want first of all, to form the linear convolution of x1 of n and x2 of n. And as you should recall from similar examples in the earlier lectures, and examples that you've worked in the study guide, the linear convolution of this sequence with itself is a triangular sequence. We of course, take this sequence, flip it over, and slide it past itself. And as we slide along, we accumulate more and more points. And so, the linear convolution of x1 of n with x2 of n is a triangular sequence, as I've indicated here.

Well, now to form the circular convolution of x of n and x2 of n. What we need to do is repeat this triangular sequence over and over again, add it to itself with a delay of capital N. And capital N in this case is 6. So we want this sequence repeated over and over again and added to itself with a spacing of six and then 12 and then 18, et cetera. Or equivalently, we want the linear convolution convolved with a pulse train where the spacing of the pulses is capital N, which in this case is 6.

So I've indicated here, the envelope of the triangular sequence, that's the envelope here, the envelope of the triangular sequence. And you can see that if we overlap these triangles by a spacing of six, then in fact, these points will get added to these points. Or when we add up the triangles, they'll always add up to 1.

So when we carry out this convolution, what we end up with are values that are constant from minus infinity to plus infinity, that's due to the convolution with this term. Finally, to form the N point, or six-point in this case, circular convolution, what we want to extract are the six points of this sequence one, two, three, four, five, six, those six points, set the rest to 0, and the result is, as I've indicated here. So this then is the N point circular convolution of x1 of n with x2 of n. This is the linear convolution of x1 of n with x2 of n.

It should be clear that, or relatively clear that this is, in fact, the way the circular convolution comes out, because if you imagine flipping this and sliding it, modulo capital N, or equivalently taking this sequence and wrapping it around the cylinder, no matter how you twist the cylinder you always see the same points. So in fact, by evaluating the circular convolution using modulus arguments, you would of course, come up with the same result. But we see here, how it comes out of this notion of linear convolution plus aliasing.

All right, well let's look at another example. Actually, it's the same example, but with one slight change to it. Here again, I have the example of x1 of n equal to x2 of n. And again, we take both of them as a boxcar sequence. In other words, rectangular or constant value in the interval 0 to n minus 1, and 0 outside that interval. Again of course, then we have the linear convolution of these two as a triangle, which is what I've denoted here.

And now let's look, not at the N point circular convolution, but at the 2n point circular convolution of x1 of with x2 of n. In that case, we want this triangular sequence repeated and added to itself, not with a spacing of capital N as in the previous example, but with a spacing of 2 capital N because we're doing a 2 capital N point circular convolution. Well if we do that, then we end up with this triangular sequence added to itself starting here, finishing over there. Or in general, if we convolve it with this pulse train, then the resulting periodic sequence that we end up with has triangular periods to it with a period now, not have capital N as we had in the previous example, but of 2 capital N.

Finally, to get the circular convolution, we see that, or we recall that we want to extract one period of this. So we extract a period from 0 to 2 capital N minus 1. So we have 0 to 2 capital N minus 1. And this result then is the 2 N point circular convolution of x1 of n with x2 two of n.

It's interesting to note that in this case, the circular convolution in fact, came out looking like the linear convolution of x of n and x2 of n. And the reason for that we see, is that when we did the convolution with this pulse train, we chose the spacing of the pulse train large enough so that there was no interference from successive replicas of this linear convolution.

Well, that suggests an interesting thing. It suggests that we could in fact, implement a linear convolution from a circular convolution if we're careful enough to pick the length of the circular convolution long enough. Now what does long enough mean? Well, long enough means that the length of the circular convolution has to be longer than the length of the linear convolution.

You can show incidentally, and you'll see this in more detail in the text, that if you have sequence of length capital N and another sequence of length capital M, then the linear convolution of the two is always going to be shorter than or equal to n plus m minus one points. So if we do the circular convolution on the basis of n plus m minus 1, or n plus m-- well, take n plus m, that will certainly work, then the circular convolution will end up corresponding to a linear convolution. And this is the way that we can carry out a linear convolution using circular convolution.

This is commonly referred to as padding with zeros. And what that means is treating this N point sequence not as an N point sequence when we compute the DFT, but as a sequence of length 2 capital N, so that if we computed a 2 capital N point DFT of this sequence, padding out with capital N zeroes, multiplied that DFT together with itself and then computed the inverse DFT, what we would get, in fact, because of the fact that we padded out with zeroes is a 2 N point circular convolution, which would correspond to the linear convolution.

So this is the way, then that we can modify a circular convolution so that it implements for us a linear convolution. And when we actually implement a digital filter, or a discrete time system using an explicit computation of the discrete Fourier transform, then of course, this is what we have to do to avoid the circular convolution since what we want in that case is a linear convolution.

Well, it's important, very important to keep straight first of all, the properties of circular convolution. Sometimes in fact, circular convolution is what you would like to get. The notion, the relationship between circular and linear convolution, mainly this notion of circular convolution being equal to linear convolution plus aliasing. And then finally the way in which you can use a circular convolution to actually implement a linear convolution.

Now what I'd like to do is really cement this by illustrating these notions, finally, with a film which will depict a number of things. It's actually the continuation of a film that you saw in the first lecture that we had on discrete time convolution. We'll show first of all, linear convolution just exactly as you had seen before. Second of all, periodic or circular convolution and the difference between them. The third thing that the film will illustrate is the relationship between linear convolution and periodic or circular convolution, namely, the notion of circular convolution as linear convolution plus aliasing. And finally, the fourth thing that it will illustrate is this notion of padding with zeros so that a circular convolution can be used to implement a linear convolution.

So let's go to the film now and go through these various examples with the hope that at the end of this, the notion and relationships will really be very, vivid.

OK the convolutions sum then, we have is the sum of x of k, h of n minus k. And so what we would like to illustrate is the operation of evaluating this sum. On the top we have x of k, on the bottom we have h of k, h of k being an exponential. And now we see h of minus k, namely h of k flipped over. As we shift h of k to the left, corresponding to n equals minus 1, then back to 0, and now to the right. So we have h of 1 minus k. And then to form y of n, we want the product of h of minus k shifted with x of k, that product summed from minus infinity to plus infinity.

" here we see x of k times h of 1 minus k, 2 minus k, 3 minus k, et cetera, those multiplied and then summed from minus infinity to plus infinity. We see during this portion that more and more values of h are engaged with non-zero values of x. And so y of n grows until we reach the point where values fall off the end of the non-zero values of x, where y of n decays. So this then is an illustration of the linear convolution of x of k with h of k.

Now what we'd like to compare this with is periodic convolution where the sum is of a similar form, but the arguments are taken modulo capital M. And in this case then, the shifting is a circular shifting rather than a linear shifting. On the top again, we have x of k, modulo M. On the bottom, h of k, modulo m. Now h minus k, modulo m, and we see that as we shift, the shift is circular modulo capital M.

So there's 0, h of 0 minus k, modulo capital M, h of 1 minus k, modulo capital M that was shifted to the right circularly by 1. And now we want to look at the sum from 0 to m minus 1 of the product of those. And the thing to notice is that since x of k is of length capital M, as we implement the circular shift of h of minus k, the values of h of k, or h if minus k are always engaged with unity values of x, and consequently, y always comes out to have the same value.

To see the relationship between the periodic and aperiodic convolution, namely the aliasing relationship, we can simply look at the aperiodic convolution, or linear convolution, take the second m values, added to the first m values, and when we carry out that sum, what we get is a constant. In general, the relationship is that the circular convolution is y of n plus y of n plus capital M, y of n plus 2 capital M, et cetera.

Next, what we would like to illustrate is the fact that the linear convolution can be obtained from a circular convolution if we pad out with zeros. So now we're going to implement the circular convolution, but notice that we've padded x of k out with a number of zeros. In that case, as we do the circular shifting of h, the values on the right are multiplied only by the zero values that we appended x with, and so this looks exactly the same as if we were sliding h in from the left-hand end. And so what results then is the linear convolution. Although we're implementing a circular convolution, because of the fact that we've augmented with zeros, the resulting answer is exactly the same as the linear convolution of x of n with h of n.

All right, well hopefully, through the film you got to see the dynamics of convolution in these relationships that I've been trying so hard to stress. And now if we return to the example that we were talking about before the film, it should be clear that through this notion of padding with zeros, we can implement a linear convolution, and thereby implement a discrete time linear shift invariant system using circular convolution, or equivalently, computing DFTs, multiplying and computing the inverse DFT.

Well there's only one hitch to that, and that's the following, suppose that what I wanted to implement was a discrete time system. Let's take the impulse response as this sequence x1 of n. And let's say that the input to the filter was also a sequence of length capital N, or capital M. Then, on the basis of everything we've been saying in this lecture, clearly we could do the convolution by padding out with enough zeros, computing the discrete Fourier transform, multiplying and inverse transforming.

However, in order to do that, obviously for the input, before we compute the discrete Fourier transform, we have to wait until the entire input comes in. In other words, we need a buffer that's long enough to hold the entire input. Furthermore what we need to do is compute a Fourier transform, discrete Fourier transform, that is at least as long as the input data. In fact, as long as the input data, plus the length of the unit sample response.

Now obviously, if we want to filter some signal like, let's say, one of Beethoven's symphonies, clearly we're not first of all, going to want to buffer the entire input signal. Second of all, you can imagine computationally that at, say a 10 kilohertz or 20 kilohertz sampling rate, we'd be talking about a discrete Fourier transform that's astronomical in length.

So if we want to implement a digital filter, or a discrete time system using this kind of approach, then we would like to have some strategy that avoids the problem of having to buffer the entire input. And one way of doing that is by sectioning the input signal.

So that, suppose we had as an example, a unit sample response which was of length capital M, so it runs from 0 to capital M minus 1. We have an input signal that's arbitrarily long, starts at 0, let's say, and runs on for an indefinite length. We can imagine sectioning the input, sectioning x of n into sections of some length, and let's call the length capital L, so that we have a number of sections of length capital L.

Now we know, and you should recall from when we discussed convolution, that the convolution of a sum, the convolution of h of n with the sum of things is equal to the sum of the convolutions. So that if we generated a set of sequences, each one corresponding to a different section of the input, we could convolve each of those with h of n, and then add the resulting convolutions together to form the output y of n.

So if we section the input, x of n, we can then form a set of sequences, each one of which corresponds to one of these sections and is 0 outside the interval that I've indicated for each one of these colored sections. And we would end up with a set of sub sequences which would, for example, look like this, x 0 of n, corresponding to the first l points, and 0 afterwards. X1 of n corresponding to the second l points, and 0 afterwards and before. X two of n, the next set of l points. X3 of n, the next set of l points. And we could continue on like that.

Implement the convolution of each one of these sections with the unit sample response. And then add the resulting outputs together. In that case, the length transform that we would need to compute the linear convolution of this l point section with the input h of n, would be where h of n was of length m, would be a DFT of length l plus m minus 1. L plus m certainly works, but it turns out l plus m minus 1 is enough.

Now obviously we have to make sure that we line up. If we have the input sections delayed by a certain amount, we have to make sure that we line up the output sections. And what we end up with then is a set of output sub sequences which correspond to the convolution of each one of the input sections with h of n.

Now we started out with sections of length L. We convolved those with a unit sample response of length m. The resulting length of the convolution will be L plus m minus 1. We've seen that in a variety of ways. And consequently, when we consider adding up the outputs from the convolution with each one of the subsections, if this is the point that corresponds to l, this is just exactly the right delay, we notice that there's a tail out of the convolution in this section that overlaps into the first m minus 1, or N points of it's m minus 1 points, actually, of this section.

Likewise, there are m minus 1 points at the tail of this one that overlap into the first m points here. So that when we add up the output sub sequences, what we have is an overlap between the tail of this section and the beginning of this one, since this starts at l and this runs longer than l.

Similarly, an overlap from the tail of this into the beginning of this one. And that continues on. Because of that, this technique for suctioning and then adding the results together is often referred to as the overlap add method because we overlap the outputs and then add these together. These pieces will add. These blue pieces will add, et cetera, down the line.

There's also another method for sectioning using circular convolution and fitting the outputs together, which is a method that is referred to as the overlap save method. And that method is discussed in some detail in the text, and corresponds basically, to implementing a circular convolution of the sections rather than a linear convolution as we did here. The linear convolution, we could get from a circular convolution by padding with zeros. Alternatively, we can take the circular convolution and discard the points that don't correspond to a linear convolution.

Well, there are a variety of techniques of this type. The important point though, is that circular convolution can, in fact, be used to implement linear convolution. It can be used to implement digital filters or discrete time linear systems. And in fact, although this-- there's nothing that we've said that clearly makes this obvious, in fact, there are very efficient ways of computing the discrete Fourier transform, which in fact make this notion of computing the transform, multiplying and inverse transforming, an efficient method for implementing a digital filter.

All right, well this concludes our discussion of the discrete Fourier transform. Toward the end of this set of lectures, we'll return actually, to the discrete Fourier transform again, where at that time, what we'll want to talk about is specifically the computation of the discrete Fourier transform, which leads to notions such as the fast Fourier transform algorithm. It's that computational efficiency that makes some of these ideas that we've talked about in the last few lectures so important in terms of implementing digital filters.

Thank you.

[MUSIC PLAYING]