MITOCW | watch?v=ZbYAZLQHXSg

[MUSIC PLAYING]

ALAN OPPENHEIM: Last time, we introduced the topic of digital filter design, and we discussed two design techniques which permitted us to obtain digital filter designs from analog or continuous time designs. In particular, the first technique that we discussed was basically mapping differentials in a linear constant coefficient differential equation to differences in a linear constant coefficient difference equation. We found that that corresponded basically to a mapping from the s plane to the z plane. But we saw also that it was not a good technique for two basic reasons. One is that it didn't correspond to mapping the frequency characteristic from the j omega axis to the unit circle. And second of all, and perhaps even more serious, it didn't result in a mapping of stable analog filters to stable digital filters.

The second technique that we discussed was the technique of impulse and invariance, where basically, the discrete time filter is designed by sampling periodically the impulse response of the continuous time filter. And we saw that what this corresponded to was basically a mapping of the system function for the continuous time filter to a system function for the digital filter, which preserved the residues and mapped the poles of the continuous time filter at poles at s sub k to poles of the digital filter at e to the s sub k capital T.

Well, let me remind you again of the advantages and disadvantages of this technique by illustrating it with a simple example. Let's take the case of a simple, second order analog filter, let's say with one 0 and two poles, so that the analog system function is given by S plus a divided by S plus a squared plus b squared. For the impulse invariant design procedure, we want to expand this into a partial fraction expansion. And in that form, this system function becomes a 1/2-- 1/2 as the residu3-- divided by S plus a plus jb. The pole is then at minus a minus jb, plus a residue of 1/2 over S plus a minus jb, in other words, a pole at minus a plus jb.

All right. Well, mapping this analog system function to the digital system function, we maintain the residue and map the pole at minus a minus jb to a pole in the z plane at e to the minus a capital T e to the minus jb capital T. So the resulting digital transfer function has these same residues, and it has poles which are at e to the minus aT e to the minus jgT, and e to the minus aT e to the plus jbT. And this system function, we can now recombine. We can recombine the two terms. And the resulting digital transfer function is as I've indicated here.

Now, this illustrates one of the important points, which is the fact that the poles got mapped according to a mapping which is of the form z equals e to the s capital T. But notice that, in fact, the 0 didn't get mapped in the same way. The 0 at minus a got mapped, for this particular example, to a 0 at e to the minus aT t cosine bT. So in fact, it's not a mapping of S to z, although the poles of the analog system function get mapped to poles of the digital system function according to the mapping z equals e to the sT.

Well, what this example looks like in the frequency domain, I've indicated here. Here is the s plane plot of the two poles and the 0 from the analog system function. And the result of the mapping or the result of applying the impulse invariant procedure are the poles and 0s, which are indicated here in the z plane.

Well, to see what happens to the frequency response, for this particular analog example, we have a resonance at a frequency corresponding to the vertical position of these poles. And that's the resonance that we have here in the magnitude of the analog system function. And one of the important aspects of the impulse invariant procedure, which I pointed out last time, is that, except for aliasing-- and that's a big except for-- except for aliasing, the frequency response has the same general shape in the discrete domain as it does in the continuous domain.

So the corresponding discrete time frequency response is as I've indicated here. We have a resonance corresponding to these pole locations. But whereas this frequency response drops off asymptotically to 0, this frequency response drops off to some value, non-zero value, and then continues back up again, basically because this frequency response has to be periodic. But in particular, the value which this frequency response decays to is dictated in part by the effect of aliasing.

And you can see here that in comparing this value to this one, this is higher. And the reason it's higher is because of the effect of aliasing. So there is the advantage to impulse invariance that the frequency axis is a scaled replica of the analog frequency axis. But then there's a distortion that's introduced, which is the effect of aliasing.

There's another technique which is a very important and commonly used technique for designing digital filters, which avoids the problem the problem of aliasing, but it avoids that problem at a price. That technique is referred to as the bilinear transformation and basically involves mapping from the s plane to the z plane by applying-- by relating s and z according to the bilinear transformation, a bilinear transformation. There, in fact, is an interesting way of deriving this technique, which is somewhat similar to the technique involving mapping differentials to differences. But of course, the bilinear transformation will turn out to be a much more useful technique.

This technique can be derived-- and we won't be going through this in detail here, although we do in the text-- this can be derived essentially by beginning with a continuous time differential equation, integrating the equation to obtain an integral equation, and then approximating the integrals, using the trapezoidal rule. In other words, replacing the integrals by sums, using the trapezoidal rule for integration, the resulting equation is then a difference equation. And that difference equation will then represent the digital filter, and that digital filter, as it turns out, can be related back to the analog filter. The system functions can be related through a transformation from s to z, which is, in fact, the bilinear transformation.

So the technique then is basically to begin with an analog system function, convert that to a digital system function, and do that by replacing s by a bilinear transformation, where the bilinear transformation is as I've indicated here. There's a parameter, capital T, that I've introduced, and it's a parameter that comes in, basically, because of what's involved in applying the integration or applying the trapezoidal rule to implementing the integration. But this then is the general form of the transformation. Or expressing z instead as a function of s, we obtain z equals 1 plus sT over 2 divided by 1 minus sT over 2.

So one important aspect of this then, which, in fact, is different than the impulse invariant procedure, is that it, in fact, does correspond to a mapping from s to z. It's a mapping of the s plane to the z plane, which is not quite what the impulse invariant method is. Well, recalling that there are two guidelines that we set down last time, we'd like to verify, first of all, whether the bilinear transformation in fact preserves the characteristics, the frequency characteristics-- that is, whether it maps from the j omega axis to the unit circle, first. And second of all, does it map stable analog filters to stable digital filters?

Well, let's look at the first question first. Let's look at the unit circle and inquire as to what part of the s plane the unit circle came from. So we have, then, z equals e to the j omega on the unit circle. And so we want to inquire as to what values of s land on the unit circle through the bilinear transformation. And we do that simply by substituting z equals e to the j omega, so that s is equal to 2 over capital T times 1 minus e to the minus j omega divided by 1 plus e to the minus j omega.

And now we can change this term and likewise the denominator term by factoring out e to the minus j omega over 2. And then this numerator term becomes e to the plus j omega over 2 minus e to the minus j omega over 2. And we recognize this numerator then as 2j times sine omega over 2.

Well, likewise, we can carry out a similar manipulation with the denominator term. The result then is that this can be rewritten as 2 over capital T times j times the tangent of omega over 2. Well, since omega ranges from 0 to 2 pi, this is always a real number, tangent omega over 2. And consequently, we can rewrite this as j times a real number capital Omega.

All right. Well, what does that say? It says that if z is on the unit circle, then s is going to be of the form j times a real number. And that has to have been, then, the j omega axis. So s equal to j omega means z equals e to the j omega, where capital Omega and little omega are related by capital Omega equal to 2 over capital T times the tangent of little omega over 2.

Well, that says that, indeed, the j omega axis gets mapped to the unit circle, and vice versa. And in fact, the entire j omega axis gets mapped to exactly once around the unit circle. Well, let's look at the mapping in a little more detail. Here we have the s plane, and here we have the z plane. We have the relationship between the j omega axis and the unit circle, that the j omega axis gets mapped to the unit circle.

And in fact, as capital Omega runs from minus infinity to plus infinity, we map exactly once around the unit circle. Now, that's an important point for two reasons. One is that, obviously, there's no aliasing involved anymore, because the entire j omega axis ended up once around the unit circle. That is, there was nothing that got added on to something else that got added on to something else, as it did with the impulse invariant method.

The second point is that that axis is infinitely long. The unit circle only has a finite circumference. So clearly, something has to happen. Something has to be distorted in order to fit all of that onto just this little bit. In particular, the mapping, if we draw it, between little omega, angular frequency around the unit circle, and distance along the j omega axis, looks as I've indicated here, which, of course, is a nonlinear curve. Well, let me return to that in a second.

An additional point about the bilinear transformation, which I'll state simply and leave it to you to verify, is that the bilinear transformation has the property that the left 1/2 of the s plane gets mapped to the interior of the unit circle, and the right half of the s plane gets mapped to the exterior of the unit circle. Well, that's good, because that says that, if we have poles or 0s-- but poles is what we really care about right now-- we have poles in the left 1/2 of the s plane, they'll end up inside the unit circle. So if we have a stable analog filter, we'll end up with a stable digital filter. If we have an unstable analog filter, corresponding to poles on the right 1/2 of the s plane, then we'll end up with an unstable digital filter, corresponding to poles outside the unit circle.

All right. Well, now, returning to the mapping of the j omega axis to digital frequency, the mapping from analog frequency to digital frequency, one of the important aspects of the bilinear transformation is that that mapping ends up as a nonlinear mapping. Well, what are the consequences of that? Let's take a look at it in a little more detail. And in addition to pointing out some of the problems that that raises, this offers us an opportunity to indicate how, in fact, this distortion can be taken into account in implementing a digital filter design.

So here is the frequency mapping of the bilinear transformation. This is digital frequency. Here is analog frequency. And this curve, of course, goes out to infinity in capital Omega and asymptotically approaches pi, as it does that, in little omega. That says that the top 1/2 of the j omega axis gets mapped to the upper 1/2 of the unit circle.

Well, suppose that we had, say, an analog filter frequency response that was, let's say, linear in frequency. Would the corresponding digital frequency response also be linear in frequency? Well, the answer to that, unfortunately, is no. In other words, a frequency characteristic along this axis will get distorted as it gets reflected through this nonlinear curve. For example, let's look specifically at a linear analog frequency characteristic, as I've indicated here which might, for example, be the frequency response of a differentiator. That, in fact, is the kind of frequency response that a differentiator might have.

As we reflect that through this nonlinear curve, because of the fact that that's nonlinear, we don't maintain the linearity of the analog frequency response. Now, if it happened that this was the kind of frequency response that we wanted, we could get that from an analog differentiator. But more typically, what we might be interested in is a digital differentiator which would have, say, a linear frequency characteristic. And one of the important aspects of the bilinear transformation is that we couldn't design that by designing-- by mapping over an analog differentiator to a digital differentiator, using the bilinear transformation.

Now, that's in contrast to the impulse invariant design procedure, where, in fact, for an impulse invariant design, if we have a band-limited filter that is linear over some part of the frequency band, impulse invariance would maintain the linearity of the frequency characteristics. So that's an important distinction between these two methods and, in fact, a drawback to the bilinear transformation.

Well, where might we be able to tolerate this kind of a nonlinear distortion in the frequency axis? One place we could is when the desired digital filter is piecewise constant, in which case that will get reflected into a piecewise constant analog filter. And let me indicate that more explicitly.

Let's suppose that we were interested in designing a digital filter, which was a low pass filter. Here's the magnitude axis, and here's the frequency axis. And the digital low pass filter was unity, say, in the passband, a rapid cutoff at the passband edge, and then 0. Well, clearly, this part being constant gets reflected through this nonlinear frequency curve into a constant analog frequency characteristic. Similarly, the stopband gets reflected into a constant amplitude, in this case, 0, in the analog frequency response.

And what's the effect of this nonlinear distortion in the frequency axis? The only effect that it has is that the cutoff frequency of the filter or, in the more general case, the frequencies at which the piecewise constant characteristic changes from one piece to another gets changed according to this frequency warping characteristic. So this cutoff frequency gets converted into a new cutoff frequency, where the analog cutoff frequency is related to the digital cutoff frequency according to that curve.

Now-- and so one of the things that that says is that for the filter design, if we wanted to design an ideal digital filter, an ideal digital low pass filter, then to decide on the specifications for the ideal analog filter, we would simply pre-warp-- what's called pre-warping-- pre-warp the cutoff frequency to an analog cutoff frequency, which would get unraveled back to the right cutoff frequency when we go through the bilinear transformation. Now, we know that there aren't ideal filters. And so, in fact, we're, in general, faced with the problem of approximating a characteristic like that.

But if the specifications for the filter are still piecewise constant, then, in fact, we can still concentrate just simply on a pre-warping of the critical frequencies. And the piecewise constant portions of the frequency response will then fall in the correct places. And let me illustrate that by showing a more realistic kind of filter frequency response. Well, that's pretty close. All right.

Here we have what would be more typical of the digital filter that we would design and implement-- some allowed tolerance in the passband, some allowed tolerance in the stopband, a passband cutoff frequency, and a stopband cutoff frequency. Well, if that was our desired digital filter, then how would those specifications translate to an analog filter? Well, if the deviation in the digital filter was between 1 and 1 minus delta sub p, then that would get converted into specifications on the analog filter, likewise between 1 and 1 minus delta sub p. The passband cutoff edge would get reflected through this nonlinear curve to a new analog cutoff frequency.

Likewise, the transition region would get reflected into a new transition region. If this frequency characteristic was, let's say, linear in the transition region, of course, it wouldn't end up linear in this transition region. But who cares? The stopband edge would get reflected into a new stopband edge. And finally, the stopband specifications, let's say, delta sub s, would get mapped to the same stopband specifications, delta sub s.

Furthermore, notice that if, let's say, we had an optimum filter which turns out to be equal ripple for the digital case, then that would correspond to an equal ripple filter in the analog case. Or said another way, if we were able to design an equal ripple analog filter, then map through the bilinear transformation, that would likewise result in an equal ripple digital filter. So the important point here then is that we can, for piecewise constant frequency characteristics and specifications, it really is only the critical frequencies that we care about, the edge of the passband, the edge of the stopband, and in the multiple band case, of course, there will be a number of those.

By pre-warping those through this arctangent curve into the corresponding analog critical frequencies, if this filter is then designed and mapped to a digital filter, using the bilinear transformation, then the passband and stopband edges will fall at the frequencies that we want. And this point again will be emphasized in the next lecture, when I show a design example comparing impulse invariance in the bilinear transformation. So the key point then about the bilinear transformation is that it avoids the problem of aliasing at a price. The price is that there's a distortion in the frequency axis. And so consequently, we can only use the bilinear transformation to design filters that can tolerate that distortion. Typically, that means designing piecewise constant digital filters.

All right. Well, that basically concludes the discussion of the design techniques that correspond to mapping analog filter designs to digital filter designs. There is one final class of techniques that I would like to just comment on somewhat briefly. And that is the class of techniques that correspond to algorithmic design.

Now, there are algorithmic design procedures for analog filters. And the same algorithmic procedures can likewise be used for digital filters. Since we have to go through an algorithmic procedure-- in fact, there is no advantage, in that case, to first designing the analog filter and then mapping that over to the digital filter. Rather, for these algorithmic techniques, it makes more sense to just simply apply them directly to the design of a digital filter.

Well, there are a number of techniques, actually a fairly long list, many of them simple modifications of others. There are only two that I want to point to in this lecture. When we talk about the design of finite impulse response filters, there are some additional algorithmic design procedures that we'll point to.

Well, the first algorithmic design procedure, which I am restricting, in my discussion here, to the design of infinite impulse response filters, the first is simply designing the filter to minimize the mean square error between the desired frequency characteristic, the desired frequency response, and the actual frequency response. So let's assume that we have some desired frequency response, H sub d of e to the j omega. We would like to match that with the frequency response of the rational transfer function.

It, of course, isn't possible to attempt to do that at an arbitrary number of frequencies. So we can pick out some set of frequencies, omega sub i, at which we would like to minimize the error between the desired frequency response and the actual frequency response. The error, then, which we'll be using, or one error which is used, is the error that corresponds to minimization of the mean square error is the mean square difference between the magnitude of the actual transfer function and the magnitude of the desired transfer function.

So this error, then, summed over the frequency points at which we specify the desired frequency response, this error, m is to be minimized. Well, how do we pick H? Well, the general form for H is as a rational function-- that's what we've been restricting ourselves to throughout the discussion-- the rational function of z, and expressed, for example, in cascade form, involving second order sections, that is, two coefficients for the 0s and two coefficients for the poles, with capital K sections all together.

So basically what the procedure then consists of-- it's somewhat brute force-- is to specify the order of the filter. We then have a set of coefficients which we need to solve for. We have a set of equations which result from minimizing this error function. And by solving that set of equations, by minimizing this error, what results then are values for the parameters, which provide the parameters for the filter that we want to implement.

Well, one of the issues with this method, one of the difficulties with it, of course, is that-- and you can kind of imagine that this is what's going to happen-- is that, if you take this rational function and put it into this equation and minimize this error, you end up with a set of equations for the a sub k's, b sub k, c sub k, and d sub k, but it's a set of nonlinear equations. And basically what the algorithmic part of this design procedure amounts to is algorithmically solving this nonlinear set of equations.

Well, let me just illustrate, with one simple example, how this procedure-- what type of filter this procedure might lead to. I don't want to, obviously, go through all the details of solving the equations. But let's just quickly look at an example. This is an example incidentally that is taken from a paper by Professor Kenneth Steiglitz, at Princeton University.

And the objective here was to design what's called a format filter, which is used for speech synthesis. And the frequency characteristic-- the aspects of the frequency characteristic that were important were to have peaks at this frequency and this frequency and this frequency-- those are the peaks that, incidentally, correspond to the resonances of the vocal tract-- and to have valleys in between those peaks. Well, one of the things that you can see from the kind of filter that we're talking about designing here is that this isn't piecewise constant. And in fact, it's not band-limited. And it tends to have-- it's not arbitrary, of course, but it's a much more general frequency characteristic shape than we had been talking about in discussing the previous methods, in particular, than we had been talking about in discussing the bilinear transformation.

Well, the procedure that was used, then, was to specify the values of the desired frequency response and to try to specify them in such a way that the frequency characteristic was more or less forced to go through the proper values at the important frequencies. So what was specified, then, was a point here. The frequency response was specified to have that value there, one here to get us heading toward the resonance, three points at the peak of the resonance there, three points to force the frequency response back down again, another three to bring it back up for the second resonance, a few points distributed along here to provide a general characteristic to tend toward the next resonance, three points at this resonance, and then a few points to have the frequency response taper off.

Well, what was used was a total of three second order sections. And the resulting frequency response that was obtained is what is shown here in solid lines. And you can see that, in fact, that's a reasonably good fit to the points that were specified. This particular example I chose to illustrate primarily, because it is one of the harder examples involving the minimization of mean square error. And in fact, in terms of computer time on an IBM 36065, this particular example required approximately 2 and 1/2 minutes. So that's a reasonable amount of computation, but one of the important aspects to it is that it allows the design, and it allowed the design, of a somewhat general frequency shape, frequency characteristic.

All right. So that is one algorithmic method. And it, in fact, is a good method. It's one of the methods that is commonly used, particularly when the frequency-- when some of the other methods-- the bilinear transformation or impulse invariance can't be used, because the filter isn't of a simple form. But one of the major disadvantages to it is the fact that the equations that result are not linear equations. And in general, solving nonlinear equations is a fairly difficult algorithmic job.

Well, there's an alternative to that, which is a method which is commonly referred to as least squares inverse design. And in that design procedure, we can imagine specifying a desired impulse response. And in this case, we restrict the filter transfer function to have no 0s and only poles.

Well, it turns out-- and I don't want to elaborate on the details-- but basically, the way this procedure is set up is to consider a system whose system function is the inverse of H of z, so that that system is in all 0 system, and then choose as an input to that system the desired unit sample response. Now, we have the desired unit sample response is the input. We've got the reciprocal of the filter that we're going to implement in this as the system function.

And if, in fact, the realized unit sample response was exactly equal to the desired unit sample response, what would the output be? Well, it would be a unit sample, because this would be exactly the inverse of this unit sample response. Well, let's think of the output as g of n. And ideally, g of n should be an impulse, a unit sample. Well, in general, of course, it won't be, since H sub d of n we won't be able to fit exactly by a model of that form.

So let's consider the error, which is the mean square difference between g of n and a unit sample. And that then is the error that we minimize. So the idea behind the least squares inverse design is basically to design the inverse to the desired unit sample response, do that by minimizing the mean square error between the output of this inverse system and the desired output. And the desired output, of course, is a unit sample.

Well, there's the legitimate question as to whether that is a reasonable error criterion to use. It's, of course, a possible error criterion to use. But one could ask, why in the world would you want to set up the problem that way? Well, there's a very good reason. The reason is that, if you do that, minimize this error in terms of these coefficients, the equations that result are linear equations. And linear equations are simple equations to solve, as opposed to nonlinear equations.

So this is a technique that, in fact, turns out to be a good technique for a wide variety of filters. It has the important advantage that it leads to a set of linear equations, as opposed to the minimization of mean square error, which leads to a set of nonlinear equations.

Well, this is just a very quick glimpse at two algorithmic design procedures. As I said previously, there are some others, which we won't be going into, some of which are described in the text. And there are a few related to finite impulse response filters that we'll be discussing in a number of lectures. In the next lecture, what I would like to do is review the impulse invariant and bilinear design procedures, since those are the two major design procedures for infinite impulse response filters, review those two procedures through a design example. Thank you.

[MUSIC PLAYING]