The following content is provided under a Creative Commons license. Your support will help MIT OpenCourseWare continue to offer high quality educational resources for free. To make a donation or view additional materials from hundreds of MIT courses, visit MIT OpenCourseWare at ocw.mit.edu.

ALAN V. OPPENHEIM: Last time, we introduced the notion of mapping continuous time filters to discrete time filters, and we developed impulse invariance as one useful technique for carrying out that type of mapping. What I'd like to do in this lecture is illustrate impulse invariance as a design procedure in the context of one specific class of filters, namely Butterworth filters. And then following that, we'll proceed on to discuss another very important and useful mapping or design procedure referred to as the bilinear transformation.

So to begin, let me just discuss briefly what the class of Butterworth filters is. And specifically, the Butterworth filters are defined through their frequency response or transfer function. And I'm using capital B to denote a Butterworth filter. And by definition, the magnitude squared of the frequency response of a Butterworth filter is given by this expression. And for example, if N were equal to 1, then this would simply correspond to the magnitude squared of the frequency response for a first order differential equation.

Now if you look at the frequency response, B of omega for this class of filters-- I've illustrated that below-- and what we see is that the frequency response starts at unity, because that's the way that it's normalized, and it has a monotonic characteristic in the pass band and in the stop band. At a frequency equal to the parameter omega sub c, up here, which is referred to as the cutoff frequency, the Butterworth filter frequency response always goes through the same point, namely 0.707.

And as the order of the Butterworth filter, capital N, increases, the transition from the pass band to the stop band becomes sharper and sharper. So for higher order filters, then, the frequency response is flatter in the pass band and drops off more

quickly, and attenuates more in the stop band.

Now in designing Butterworth filters, what we want to look at is the location of the poles of the system function. And we can infer those from our definition of what the frequency response for the Butterworth filter is. In particular, we have this expression for the magnitude squared of the frequency response. And we recognize that, of course, as B of j omega times B of minus j omega, that's what the magnitude squared will be equal to. And in order to convert this to an expression describing the system function, or in terms of the more general Laplace transform variable s, what we recognize is that j omega, in the more general setting, simply corresponds to the Laplace transform variable s.

So this product, in fact, is the Laplace transform for s equal to j omega. More generally, then, this is the result of evaluating B of s times B of minus s at s equals j omega. Consequently, comparing this expression with this statement leads us to the conclusion that the transfer function B of s of the Butterworth filter, times B of minus s, is given by the expression that I indicate here, simply replacing j omega by s.

Now what we want to look at are the poles of B of s. That's what we'll want to get as we design a Butterworth filter. And we can recognize the poles of this product simply by looking at the roots of the denominative polynomial. And those roots are-- just taking account of this factor 2N-- those roots are at j omega sub c times the 2N roots of minus 1.

Those roots, in fact, all lie on a circle, and the consequence of that is that the poles of this expression are on a circle. The circle is of radius omega sub c, and the poles are distributed around the circle. So here I've illustrated the poles of B of s times B of minus s, for this specific case where capital N is equal to 3. So there are a total of six poles around this circle. And then for this specific case, the poles are spaced by 60 degrees.

Now we have B of s times B of minus s. To get the system function for the Butterworth filter, we'd like to get B of s, and the question now is how do we get that? Well, the thing to recognize is that wherever this factor has a root, this factor

has to have a root at the negative location. So in fact, when we look at these poles, we compare them with this, for example, associated with B of s, this associated with B of minus s. And likewise, we compare these two together, likewise we compare these two together. And so we can extract B of s from this product, simply by taking one pole out of each of those pairs.

Now a question, of course, is out of each pair, which one do we associate with B of s, and which do we associate with B of minus s? And the answer drops out fairly simply if we recognize that if we want to design filters that are stable, then B of s, the transfer function that we're designing, must have all its poles in the left half of the s plane.

So in fact, out of each of these pairs we would associate the left half plane pole with B of s, and so the transfer function for the Butterworth filter for this particular case, where the this designates the parameter omega sub c and capital N is 3, namely a third order Butterworth filter, is this set of pole locations. Given those, of course, we can figure out, simply through Algebraic means, what the transfer function B of s is.

All right, so that's what Butterworth filters are. And now what I'd like to do is talk about the design of a digital Butterworth filter using the design technique that we introduced last time, namely impulse invariance. And the context in which I will phrase the design is the context of mapping a continuous time signal to a discrete time signal, carrying out filtering using the discrete time filter that we're designing, and then mapping back.

So we're talking about now a discrete time filter that we want to design through impulse invariance from continuous time Butterworth filters, and we're going to get our design specifications in the context of having considered discrete time processing of continuous time signals, where we will map from a continuous time signal to a sequence, carry out the filtering with the discrete time filter that we're going to design.

And then we will take the resulting filtered output and map it back to the continuous time signal. But this discrete time filter is the one that we're talking about designing.

And for a choice of parameter, there's a sampling frequency, of course, involved in this process. And the value that I'll pick for the sampling frequency is 10 kilohertz. OK, all fairly straightforward so far.

And so since we have a sampling rate of 10 kilohertz, we want to first look at our specifications on the desired continuous time filter and then map those to appropriate specifications on the discrete time filter. And what I'll pick for the desired specifications on the continuous time filter is that at 1 kilohertz, I will ask that the continuous time frequency response be down by no more than 1 db in comparison with its value at omega equals 0. So that, in effect, specifies the behavior in the pass band, or the specifications on the pass band. And for the stop band, I'll specify that the filter is down by 15 db by the time we've gotten to 1.5 kilohertz.

So we have, essentially, the beginning of the transition band-- the end of the pass band--at 1 kilohertz, and the beginning of the stop band at 1.5 kilohertz. And since we're talking about designing a Butterworth filter, we know that the Butterworth filter is monotonic in the pass band and stop band, and so we'll have a filter specification, something as I show here. This represents the allowable pass band tolerance. This is the allowable stop band tolerance. And if I can draw this without getting myself into trouble, essentially we're looking for a filter, then, that always stays between the specified boundaries here.

Now what we have to figure out is what the corresponding specifications are for the digital filter. And the strategy, let me emphasize, is that we have a situation where we're doing discrete time processing of continuous time signals, and we have a set of specifications associated with that. That imposes specifications on our discrete time filter, and then we want to design the discrete time filter using impulse invariance, and that's the discrete time filter that we'll use in the overall system.

All right, now we want specifications on the discrete time filter, and we want the continuous overall equivalent system to meet certain specifications at certain frequencies related to continuous time frequencies. Recall that when we sample a continuous time signal, there's a very specific mapping from the continuous time

frequency axis to the discrete time frequency axis. In particular, the sampling frequency gets mapped to 2 pi. Well, that means that our other critical frequencies get mapped in proportion to that. So 1 kilohertz, which is a 1/10 of the sampling frequency, would then convert to a discrete time frequency of 0.2 pi. And 1.5 kilohertz will convert to a discrete time frequency of 0.3 pi.

So what this says is that for the discrete time filter, we would like the same behavior, but at frequencies-- or the same specifications, but at frequencies normalized to the discrete time frequency axis. That means that we want the discrete time frequency response magnitude to be greater than or equal to minus 1 db at 2/10 pi-- corresponding to the 1 kilohertz in continuous time-- and for the beginning of the stop band, that would occur at 0.3 pi, at which point we want this less than or equal to minus 15.

So those are our discrete time specifications. And we now want to design the discrete time filter using impulse invariance. Now in impulse invariance, as you recall, it corresponds to generating an impulse response which is a sampled version of the continuous time impulse response, and there is a temptation naturally to think all of this parameter, capital T, as necessarily identical to the sampling in the system in which the filter is going to be used.

Now this is a fairly subtle, complicated, tongue twisting issue, but the bottom line on it, the essential point, is that the parameter capital T that we use in impulse invariant design is a totally different, unrelated, and in fact, as it turns out, arbitrary parameter, which is not necessarily pegged to the sampling frequency. And I think it would be difficult for me to totally clarify that during the lecture. It's discussed more in the book, and certainly, you should take plenty of time to reflect on it.

All right, but let's now look, then, at where we are in our design procedure, and specifically, what it is that we need to do in order to design the digital Butterworth filter. Now we have a set of specifications that we've generated relating, essentially, to how we want the pass band of the digital filter and the stop band of the digital filter to behave. Of course, since this isn't an ideal filter, it has some transition from

pass band to stop band, and as we discussed last time, there is aliasing, which we need to at least be aware of.

We specified certain frequencies along this axis which are easily converted by relating to the two axes through this mapping, are easily related back to the continuous time frequency axis, as we have here. And in particular, now if we were to simply pick that parameter in the impulse invariant design, capital T, as equal to unity, and I indicated just a minute ago that we can pick it arbitrarily, if I pick it as unity, then the procedure would consist of designing the continuous time Butterworth filter with meeting or exceeding the appropriate specifications and then going through the impulse invariant procedure.

All right, so let's do that then. We want the discrete time impulse response to be the continuous time impulse response sampled, and for convenience, I'm going to pick this parameter capital T equal to 1. That means that the frequency normalization between the discrete time frequency axis and the continuous time frequency axis in fact is-- those axes are scaled identically, because capital T is equal to 1.

And so now we want the analog, or continuous time, specifications. And so what we need to do then is design a Butterworth filter. I'm using capital B here again to denote the frequency response of the Butterworth filter. The Butterworth filter to have a magnitude which is greater than or equal to minus 1 db prior to the frequency 0.2 pi, and less than or equal to minus 15 at a frequency beyond 0.3 pi. And so now what we need to do is determine capital N and omega sub c in order to meet or exceed those specifications.

Now if you go through the associated algebra in doing that, let's say that you decide that you want to pick capital N and omega sub c to exactly meet those inequalities at the two frequencies, 0.2 pi and 0.3 pi, what you'll find after going through the algebra is that they're exactly met if capital N is 5.88 and omega sub c is 0.7047. And this obviously isn't satisfactory as parameters for the Butterworth filter. Why is that? The reason is that capital N isn't an integer, and in the class of Butterworth filters, to generate a rational transfer function, capital N, this parameter, must be an

integer.

So since it can't be 5.88, the natural thing to do is to move it up to the next closest integer, namely 6. And that means that we'll end up with a filter that does even better than the specifications. On the other hand, there's something kind of underneath the surface that is inherent in the impulse invariant design procedure, namely the fact that there will always be some aliasing.

So one strategy, and a natural one, often, in impulse invariant design, is to choose N as the next highest integer as I've done, and then choose the parameter omega sub c so that the pass band specifications are exactly met, and the stop band specifications are then slightly exceeded, and that will leave some margin for aliasing.

All right, now continuing this example, then, how would we complete the design? Well, we know what our two parameters, capital N and omega sub c are. That means that we can determine B of s times B of minus s. Those poles are located on a circle in the complex s plane. And the poles on that circle are paired, some being associated with B of s and some with B of minus s. And in particular, to determine B of s, we would simply take the poles on the portion of the circle that's in the left half of the s plane.

Now what that gives us is a Butterworth continuous time filter, which we are then mapping through impulse invariance with capital T equal to 1 to a discrete time filter. That discrete time filter to be used in a system which has an associated sampling frequency which is 10 kilohertz. So to get the discrete time filter, then, we first determine B of s, as we just did-- or at least I indicated how to do it.

We would then expand that out in a partial fraction expansion, and then apply the impulse invariant procedure, which consists of mapping the poles in the s plane to poles in the z plane at locations e to the s of k, capital T, capital T is equal to 1. And retaining the residue, or the coefficient, in the expansion. And this, then, will give us the transfer function for the discrete time filter.

So if in fact we did that, then the resulting frequency response that we would get is what I've indicated here. And I indicated first on a magnitude scale, linear magnitude scale, and second on a logarithmic scale. And as we had originally specified, the digital filter is supposed to be greater than or equal to minus 1 db at point 2 pi, and less than or equal to minus 15 db at 0.3 pi. And in fact, this slightly exceeds those specifications since we had purposely allowed some margin in the stop band.

Now this is an illustration of the impulse invariant procedure, and it has a number of very nice properties, one of which is that it takes a continuous time filter frequency response and it converts it to a discrete time frequency response, which in the absence of aliasing, looks identical, except for a linear frequency scale change. And in fact, if we picked capital T equal to 1, then there is no scale change. It has the major disadvantage that there is always aliasing, and for some problems, for example, if the filter that we're trying to design is not band limited or low pass, then the aliasing will naturally become intolerable.

Well, there's another design procedure which I now want to introduce, which totally avoids the problems of aliasing, but, obviously, then has its own costs associated with it. And that's procedure is referred to as the bilinear transformation. The bilinear transformation, which I won't try to derive here in any detail, is a mapping of continuous time filters to discrete time filters, corresponding to taking the Laplace transform variable s in the continuous time filter and replacing it by what is referred to as a bilinear function of z. And so if I substitute this in here, that will give me the discrete time frequency response.

Again in this procedure, there's a parameter capital T, which again is totally irrelevant given the approach that we're taking, and which we will generally tend to normalize out to unity. And let me just say quickly and in passing that although we won't go through this, the notion that the bilinear transformation can be tied to the concept of taking the differential equation for the continuous time filter, converting it to an integral equation by integrating enough times on both sides, and then converting that to a difference equation by approximating the integrals with the

trapezoidal rule. And that, in effect, will correspond to mapping the continuous time filter to a discrete time filter with the bilinear transformation.

Well, we'll just focus on the properties of the mapping. And in particular, if we were to substitute into the expression z equal to e to the j omega corresponding to the unit circle, we would find that the unit circle in discrete time corresponds to mapping the j omega axis in continuous time, which is exactly what we want. Now the mapping between the continuous time frequency and the discrete time frequency is a nonlinear mapping, which is given by the algebraic expression that I indicate here.

And if we plot this mapping, what we have is this curve. And what this corresponds to, then, is a mapping of the j omega axis, or continuous time frequency, to discrete time frequency. And if we think more generally of the mapping represented by the bilinear transformation in the context of the s-plane and the z-plane, it corresponds to mapping the j omega axis in the s-plane to once around the unit circle in the z-plane. And you could also convince yourself that the left half of the s-plane maps to the inside of the unit circle. And so that means that stable continuous time filters will always map to stable discrete time filters, which is exactly what we desire.

Now notice in this that there's no issue of aliasing. What's happened is that we've replaced s by a function of z, and it corresponds to mapping the s-plane to the z-plane. In fact, the whole j omega axis has mapped to once around the unit circle, which obviously requires some type of nonlinear mapping, because, look, the j omega axis is infinitely long. The unit circle has a finite radius.

And so essentially, what has to happen is if you think of walking along the continuous time frequency axis, and simultaneously walking around the unit circle, if you walk at a constant rate around the unit circle and you're simultaneously walking up the j omega axis, if you want to get around to pi by the time, here, you've gotten to infinity along the continuous time frequency axis, you better start walking faster, and faster, and faster, because you've got an infinite distance to cover while you just go over a finite distance here.

Well, what all that says, really, is that that's why, in fact, we're taking the entire-- or

we're able to take the entire omega axis and just map it into an interval of length pi. All right, now that means that there is a nonlinear distortion of the frequency axis if we were to take a continuous time filter and convert it to a discrete time filter with the bilinear transformation. How do we account for that, or for that matter, when can we really use it?

And we can see how to both take account of it, and what its limitations are, by recognizing the following. Suppose that I want to design a discrete time filter and what is going to happen is that it will be mapped to a continuous time filter, or the relationship between the two frequency axes will be given by this curve.

So let's suppose that the continuous time frequency response looks as I've shown here, with a pass band cutoff frequency and a stop band cutoff frequency. If this were mapped through the bilinear transformation to a discrete time filter, then this cutoff frequency would fall over here, which is related through this curve, and this cutoff frequency, the stop band edge would be here. Again, it's these frequencies reflected through this curve.

So let's suppose now that what I'd like to do is design a discrete time filter, where the discrete time filter has certain frequency specifications, which is what our previous example has. It has, let's say, a specified stop band edge and a specified pass band edge, and a specified stop band edge. The design procedure would then correspond to mapping those frequencies to the corresponding continuous time frequencies, designing the continuous time filter to meet the specifications based on those critical frequencies, then taking the continuous time design and mapping it back to a discrete time filter through the bilinear transformation.

All right, now what we want to do is again map a Butterworth filter, continuous time Butterworth filter to a digital filter in such a way that the digital filter approximately meets the specifications that we had before. And let me just remind you of where we were. The critical frequencies were at 0.2 pi and 0.3 pi. And what we had asked for is that the frequency response be down by no more than 1 db up to 0.2 pi, and down by at least 15 db at 0.3 pi.

So we want to design the same discrete time filter as we did before with impulse invariance. We now want to do it with the bilinear transformation applied to an appropriate Butterworth design. OK, well, let's see what kind of specifications we have. We know that the frequency, the critical frequencies, are mapped through this curve, or this equation. And I indicated again that this parameter capital T is arbitrary in the design procedure. That may seem confusing, initially, but there's some further discussion of it in the book, and it's true and important to sort out. So we're going to pick capital T equal to unity.

And that means then that the corresponding critical frequencies of the continuous time filter are at twice the tangent of 0.2 pi over 2, and twice the tangent of 0.3 pi over 2. So that means then that the specifications on our continuous time filter are given by this, essentially pass band and stop band edges warped through that nonlinear curve-- this nonlinear curve. And I'm using capital G here to denote the system function for the resulting continuous time filter. And so now we could think of designing a Butterworth filter that, let's say, exactly meets these specifications.

Well, if you do that, what you'll find is that you get exact equality here if you pick capital N equal to 5.3. And again, we have the issue that if we want to meet or exceed the specifications, we can't make the filter order lower, we have to make it higher. And so we would make it, instead of 5.3, we would make the filter order equal to 6. And now, again, we have several options and trade offs.

Before, with impulse invariance, we essentially decided to meet the pass band specifications and exceed the stop band specifications to provide some margin for aliasing. Here, we don't have any aliasing, and we can trade things off anyway we'd like. The way that I've chosen to do it is to exactly meet the stop band cut off, and exceed the pass band specifications. And the result of doing that is to choose a filter order capital N equal to 6, and the parameter omega sub c in the Butterworth filter is given by 0.76622.

All right, so now we have the parameters for the continuous time Butterworth filter, which when mapped to the discrete time filter through the bilinear transformation,

will exceed our requirements in the pass band and just meet the stop band cutoff. And so now we want to complete the design procedure. And given the parameters of the Butterworth filter, we can draw our appropriate circle in the s-plane, which happens to be the same circle as before, but with a different radius.

And then associate poles on this circle with B of s and B of minus s. In particular, take the ones in the left half of the s-plane-- same ones we had before except we have a different value for omega sub c-- and those, then, represent B of s. So we can now determine B of s. And once we have the transfer function B of s, we then map that to a discrete time filter by mapping through the bilinear transformation. And that then is the design procedure which we would follow this example.

All right, well, let's just see what the result looks like when we're done. Here I show, again, on both a linear magnitude scale and on a logarithmic or DB scale, the frequency response of the resulting filter. And recall that the specifications were that at 0.2 pi, we should be down by no more than 1 db, and clearly we've exceeded that considerably. And at 0.3 pi, we should be down by at least 15 db, and we chose the design so that we would exactly meet that edge. So we've met this point and exceeded this point, and this is our design.

Now let's just compare this with the impulse invariant design that we saw a few minutes ago. The impulse invariant design is what I indicate here. The bilinear transformation design is now overlaid on top of it. It's hard to see much difference on a linear magnitude scale, but let's look on a logarithmic scale. And this one that I'm lifting is the bilinear transformation, and this is impulse invariance.

And notice, in fact, something very interesting, which is that the filter that we obtained by mapping the Butterworth filter to a digital filter through the bilinear transformation in fact falls off in frequency much more rapidly than the one that we got through impulse invariance. The question is why? Now one thought that might come to mind is, well, impulse invariance has aliasing, the bilinear transformation doesn't have aliasing, that must be a consequence of aliasing. In fact, that's not the reason. Aliasing, as it turns out in this particular design, was relatively minimal in the

impulse invariant design. The reason has to do with this nonlinear mapping in the bilinear transformation from the continuous time frequency to the discrete time frequency.

Keep in mind that through that mapping, as you start walking around the unit circle and moving up to j omega axis, as you move up the j omega axis, you have to move faster, and faster, and faster, and faster. And what's in continuous time frequency at infinity is what you get to in the discrete time frequency by the time you get around to pi. So in fact, what we're looking at as we look out along this frequency axis is we're seeing higher, and higher, and higher frequencies in the continuous time filter. By the time we get to pi, we should in fact be in the continuous time filter equivalently off to infinity, which sounds like a pretty uncomfortable place to be.

OK, now this was a fairly rapid trip through a number of issues, in particular, some of the issues associated with the bilinear transformation, and also this issue of how you pick this parameter capital T, and how it might be associated with a sampling frequency if you're doing discrete time processing of continuous time signals. And we don't have time to explore some of those issues more fully in this lecture.

But I'd like to conclude by making a couple of comments. One comment is that the two techniques that we've talked about, impulse invariance and the bilinear transformation, are the two techniques that are principally used when one thinks of mapping continuous time filters to discrete time filters for whatever application. And I stress again that you may want to do that mapping whether or not the discrete time filter is eventually going to be used for processing continuous time signals.

Now impulse invariance had the very nice characteristic that it corresponds to a linear mapping between the two frequency axes, except for the issue of aliasing, and that's a problem with it, and in particular, limits its usefulness to filter designs, or for mapping continuous time filters that are band limited. On the other hand, we have the bilinear transformation as a design procedure, which totally avoids aliasing, but has the disadvantage or difficulty that it represents a nonlinear mapping from the continuous time filter to the discrete time filter.

Now this nonlinear distortion is perfectly acceptable if we're designing or attempting to design filters that have flat frequency characteristics. It's not acceptable if, for example, we had a linear frequency characteristic that we wanted to map to a discrete time filter and end up with a linear frequency characteristic. It won't come out to be linear because of this nonlinear mapping of the frequency axis.

Now there are also a number of other design procedures, which we won't go into, for designing discrete time filters, and among them are a variety of techniques, including, for example, computer aided design procedures. And I invite you, if you're interested and want to dig into that in more detail and more deeply, to explore that topic by making reference to various of the books listed in the bibliography in the text. Thank you.