Signal Processing Algorithms & Architecture
Dr. Anirban Dasgupta
Department of Electronics & Electrical Engineering
Indian Institute of Technology Guwahati
Lec 24: Filter Design

Hello everyone, welcome to the lecture on filter design. This is Dr. Anirban Dasgupta, and let us get started. So, so far in these filter structures we have seen how we can implement the filters, specifically the FIR filters and the IIR filters. And what are we implementing? So, in the FIR filter, we are given the H(Z), or in short, a polynomial in H(Z), and we are implementing the filter structures as per our requirement and often, what is our H(z)? This is basically a polynomial in Z, or if you take the inverse, which is h[n], called the impulse response, this is nothing but h_0, h_1, and up to some h_n.

So, now the question is what these h[n] are? Of course, you will say the impulse response. But how do we get this? And directly, the problem of filtering is that we have our h[n]; we give the input, and we get the output. But the inverse problem, or the design problem, is how do we get these values of h such that I get my desired response? If I want to make a low pass filter, can I arbitrarily choose any value of h? No. So, design is the problem where you have to get your filter values correct.

So, the filter design problem is estimating your H(n) given the filter specifications. And what are the filter specifications? So when I say filter, I mean that I need to keep this range of frequencies and remove the other range of frequencies. For example, if I want to design a low-pass filter, I want to design the filter in such a manner that it retains all the frequencies from the range 0 to fc hertz, and beyond f_c hertz, it should discard all the frequencies.

$$H(f) = \begin{cases} 1, |f| \leq f_c \\ 0, |f| > f_c \end{cases}$$

If it is a high-pass filter, I should say that it should be designed such that it retains everything beyond f_c and discards everything below f_c. Similarly, there can be a bandpass filter.

So, this is low pass, this is high pass, this is band pass, and these are basically very trivial things. So I am just giving you a hint. So, f_1 and f_2 are two cutoff frequencies for this purpose. Similarly, we can have a band-reject filter that will reject a specific band of frequencies; a band-reject or band-stop filter is fine. So, the problem is that we should give these values regardless of the type of the filter and what the cutoff frequencies are; for low pass and high pass, we should have one cutoff frequency, and for band pass and band reject, we should have two cutoff frequencies.

But these filters are typically ideal filters, and in the real world, we will not get to design ideal filters. So for practical filters, our response will not be exactly like this. Let me change the color. So in the real world, our response, I think I have explained this, but still, I will get some kind of this. So, this is called ripple; this is pass band ripple, this is stop band ripple, and then there is this transition band.

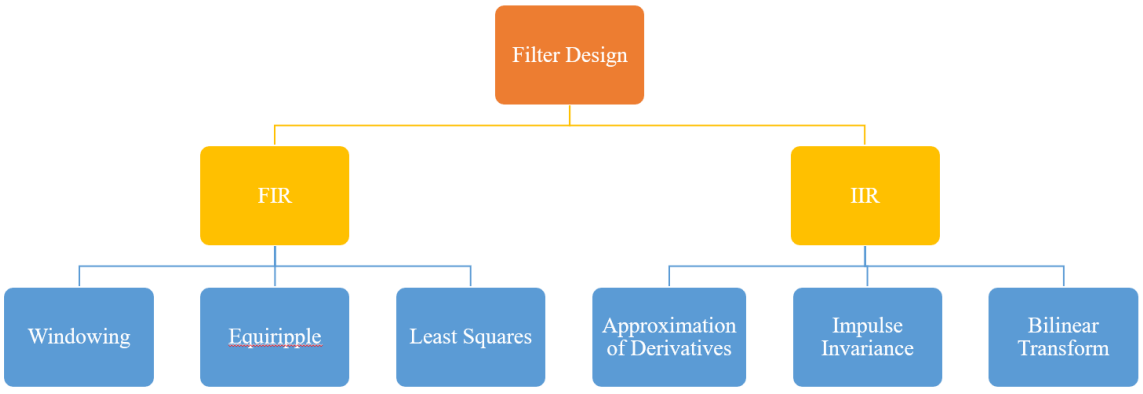
So, all these specifications that you provide, I have to design my values or design my filter in such a way as to get the values of h[n] that give me a response closer to my desired response. In the desired response, I should have some cutoff frequency, some transition band, and some pass band ripples from stop band ripples and often the type of filter, whether it is IIR or FIR, and if it is FIR specifically, may determine the minimum order filter we need to design. Like with a minimum number of coefficients, what is the best response I can get? So typically, these filter characteristics are designed in the frequency domain, as I said, and this is in terms of the magnitude response. I can give you the magnitude response of the filter or the phase response of the filter, and you have to get the h[n].

Now, you may feel that, okay, that is pretty simple because when I know the magnitude and phase response, I know my H (ω) , right? And then if I take the IDTFT, then I can get my h[n], right?

$$H(\omega) = \begin{cases} 1, & |\omega| \leq \omega_c \\ 0, & \omega_c > |\omega| > \pi \end{cases}$$

But sometimes it is not so straightforward because, since this is a continuous time function, it is not very easy or straightforward to directly do the integration and obtain these specific values of h[n], mainly when H (ω) is not a very well-defined analytical function. Again, if you need linear phase characteristics, like my phase should vary with frequency linearly, then you have to design FIR filters. You cannot design an IIR filter. Also, IIR filters have lower side lobes in the stop band than an FIR filter with the same number of parameters. An IIR filter implementation also involves fewer parameters, requires less memory, and has lower computational complexity compared to an FIR filter for the same specification.

And this is the reason that if this phase linearity is not a mandatory criterion and you are okay with some phase distortion, then an IIR filter design is always preferred. But if phase is a very vital characteristic in your design, then you should go for FIR filters. And if your FIR filters are linear phase, then the coefficients will be symmetric in nature. So filter design, as I said, can be an FIR filter or it can be an IIR filter, and you have to select, at the first step of your design,



Whether you want to choose an FIR filter or an IIR filter. So, in FIR filters, there are several methods like windowing, equiripple, and least squares.

There are other methods, such as constrained least squares, but these are the three main, popular, or I would say basic methods. In IIR, the methods involve the approximation of derivatives, then impulse invariance and bilinear transform. So, let us try to understand what these methods are. So, I will give you the basics of each method. I will not go into really great detail about this, but you should have a good idea of what these methods are and how to design them if you are given the specifications.

$$H(f) = \begin{cases} 1, |f| \leq f_c \\ 0, |f| > f_c \end{cases}$$

So we will start with an ideal filter, and in an ideal filter, this is our desired response, which I said should have a gain of 1 in the passband, that is, f mod f less than or equal to f_c; f_c is my critical frequency or cutoff frequency, whatever you say, and it is 0 beyond the cutoff frequency, and this is. I am taking the example of a low-pass filter, and this general concept can be easily extended to other forms of filters. So, what should the impulse response be? So this is typically a box filter, and a box in frequency is a sync in time and vice versa. So, if you see this, this is with some adjustments; this is a sinc function, and the sinc function is-

$$Sinc(x) = \frac{\sin(\pi x)}{x}$$

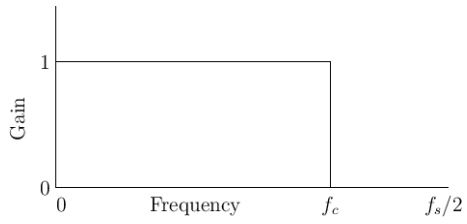
So, if you look at this function, this is my h(t) in the continuous time domain.

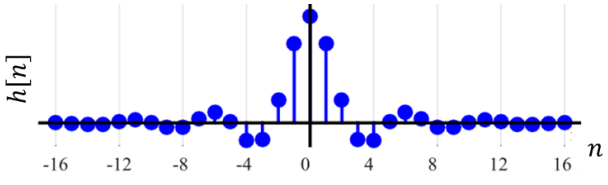
$$h(t) = \frac{\sin(2\pi f_c t)}{\pi t}$$

So, what should my response be in the digital domain? Well, this is my response in the digital domain, and here is what I see in this case: my analog frequency is non-limited; it is going from $-\infty$ to $+\infty$. But my digital frequency is limited from $-\pi \; to + \pi$. So it will have a gain of 1 from $-\omega_c$ to $+\omega_c$, and from that ω_c to π range, it will have a gain of 0, and this is the ideal filter and its impulse response in discrete time will be, well, this is the value because this sinc function will become 1 because of this limit, and so at $n = 0$, this is the factor which is $\frac{\omega_c}{\pi}$, and I have taken this out just to make this a proper sinc function, and at n ≠ 0, this is my gain or the response.

The filter, which is $\frac{\omega_c}{\pi}$ and $\sin(\omega_c n)$, has a response in the frequency domain, and in the time domain, this is the response.

$$h[n] = \begin{cases} \frac{\omega_c}{\pi} \frac{\sin(\omega_c n)}{\omega_c n}, & n \neq 0 \\ \frac{\omega_c}{\pi}, & n = 0 \end{cases}, \quad -\infty < n < \infty,$$

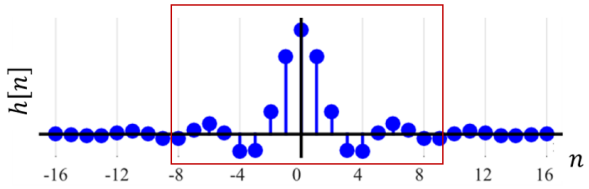




And here this is not limited to 16; I have just shown up to 16. Here, the frequency is going up to infinity, and n is going up to ∞, and of course, down to −∞. Now the first problem with this filter is that it has infinite length, but okay, IIR filters have infinite length; that is not a big deal. The big deal is that this system is highly non-causal because causal filters should not have any values below n = 0.
If I have something at n = - 1, that means it is taking one future value of the input. So this is not possible to achieve in real life. So, what is the windowing method? So, we see that this ideal low pass filter h[n] is non-causal; it is not only non-causal, it is highly non-causal; it goes fully up to −∞. So, it cannot be realized in practice for real-time signal processing applications. So, what can we do? So, we can select a few samples by multiplying them with a window function.

Now, typically, if I say that I will only select this -8 to +8, that is this 17, 16 plus 1, 0th sample, 17 samples, and this will be my filter of order 17. Well, that is possible, and although this is non-causal, this is the filter; this is of finite length. What we will do is put a shift of plus 8 so that this 8 becomes 0. So first of all, I get a symmetric filter, which means it will have a linear phase response, and second, I will have a causal filter by shifting this. To the positive side, that is to introduce a large delay, not, and in this case, my delay will be 8 samples.

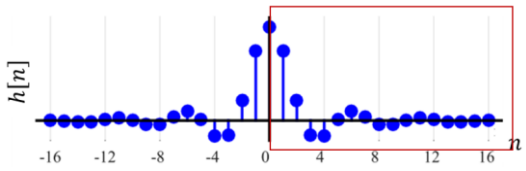
And then, after shifting this, everything beyond this window is set to 0, and this method, since I am multiplying this, means I am multiplying these regions by 1 and the rest with 0s. So, I am multiplying it with a rectangular window that has a value of 1 in this region and the rest are 0s. But what is the problem? This looks so good, but the resulting system no longer has an ideal frequency response like the one we desired. So, this is what I was talking about. So, you shift this and then take the window.



So, basically what we are doing is multiplying the desired response. So here, just the window has shifted. So basically, I will shift the whole thing, the whole signal, and that becomes your h[n], which is my desired response. This is my desired response, which is shifted and then multiplied with the window like this, like this. So here, if this comes to the 0 index and this comes to your index, whatever it is, say 17.

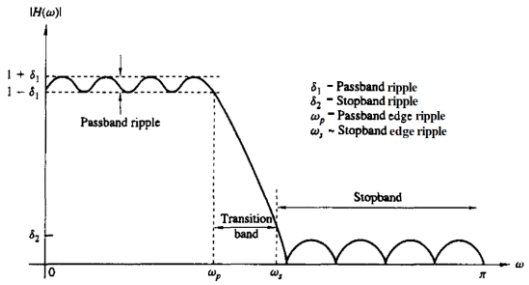
So, it is symmetric and has a finite length. So this is my final design. So if I do this, this is my actual response. So this h[n] is my actual response, $h_d[n]$ is my desired response, or you can say the ideal response, which is the infinite non-causal response, and this w[n] is the rectangular window.

$$h[n] = h_d[n].w[n]$$
$$H(\omega) = H_d(\omega)W(\omega)$$



This is w[n] and is basically having 1 for this window range that is 0 to your n, and this is after you set a delay to the signal, and it is 0 otherwise elsewhere.

So, when I do this multiplication in the time domain, it will become convolution in the frequency domain. So, this is a convolution. What is happening is that this was my desired response in the frequency domain right at ω_c, and then I am multiplying it with the window function. So, if this is my window, what is the spectrum of the window? The window is a box function. So, I will get the spectra, and this is the magnitude spectrum, say.



So, the magnitude spectra will be a sinc. So sinc is like this, but since this is magnitude, everything will be plus. So I feel like this. Now, if I do a convolution of this with that, what will happen? We get something like this, where the ideal response is no longer ideal, but this is a practical response. So, what is happening here is that, instead of getting a sharp cutoff at ω_c, we get a smooth cutoff, and this is called ω_p, which is the pass band representation, pass band frequency, and this is the stop band frequency, and this is called my transition band.

And so, for a good filter design, this transition band should be as low as possible. Similarly, there are some ripples in the stop band and there are some ripples in the pass band. So, you can say this ripple is + or - δ_1 ; this is called the passband ripple. So, this range is $2\delta_1$, and similarly, this δ_2 is the stopband ripple. So, now if you see that this rectangular window is bringing this kind of response, which is not a very good response.

So, can we do better? How can we do better? Maybe I should reduce the transition band. Or maybe I will reduce the ripples in the pass band and stop band. So, based on that, there are different windows. So the rectangular window that I have explained is easier to understand. But this may not be giving you the best response because of the sharp cutoff.

Because of the sharp cutoff, you are getting that kind of sink response, which is making the actual response very poor. So there are some modifications to this window, like the Hamming window,

which is given by this equation. There is a Hanning window, which is also sometimes called the cosine window. Then there is a Barlet window, which is like a triangular window.

Rectangular Window

$$w[n] = \begin{cases} 1, 0 \leq n \leq N-1 \\ 0, \quad \text{otherwise} \end{cases}$$

Hamming Window

$$w[n] = 0.54 - 0.46\cos(N - 12\pi n), n = 0,1,2,\dots,N-1$$

Hanning Window

$$w[n] = 0.5(1 - \cos(N - 12\pi n)), n = 0,1,2,\dots,N-1$$

There is a Kaiser window. So these window functions have been developed, and there are some advantages of each window function over the others. So, this is basically the windowing method for designing your FIR filter. Similarly, you can design your FIR filter using least squares or equiripple methods. So, equiripple as a method suggests that the energy in the ripples is distributed equally to get a better response. So, I will not cover the equi-ripple and least squares in this lecture, but I will just cover this one method, which is the windowing method, so that you know at least how to design FIR filters using the window method.

So, you are highly recommended to read the least squares and equiripple methods as well from any standard textbook. So we will now quickly go to the IIR filter design. So in IIR filter design, the catch is that in FIR filters, we know exactly how many values of h(n) we require. But in IIR, we do not know because there is an infinite number of points of h(n). So naturally, it would be foolishness if I tried to design all the infinite values of h[n], right? In the difference equation, do we also write infinitely many times for x[n]? No.

For example, take the accumulator as an example.

$$y[n] = \sum_{k=-\infty}^{n} x[k]$$

So, it is accumulating everything from the past. So, y[n] is given by the sum of your k from $-\infty$ to n x[k]. So if I represent this as an FIR system directly, it is very difficult to get, right? But if I represent y[n] as

$$y[n] = x[n] + y[n-1]$$

right? Because I can just take out this x[n], and the remaining part is up to n-1, which is my y[n-1]. Like this step if I write one more step that y[n] is mine, so I take this n and write x of n plus this becomes what? This becomes So n-1 because I have taken n out, and this is xi.

What is this? This is y[n-1]. So this IIR way of writing this is a better representation, and that is why we get that numerator-denominator part in the H(z).

So, this process might be very difficult, like the methods we discussed in the FIR case. So, what people typically do is, or how it is designed, is that first the analog filter $H_a(s)$ is designed, and

then this analog filter is converted into a digital filter H(Z) using a mapping from S to Z, and based on the mapping, there are different methods. So, why is this taken is because analog filters have been widely used before the birth of digital filters. So, the topic of analog filter design is quite mature and well developed.

So, you can design any analog filter very nicely based on the specifications and now, once you have designed the analog filter, you just need to convert this H(S) into H(Z) using some standard mapping.

$$H_a(s) = \frac{N(s)}{D(s)}$$

An analog system or an analog filter can be described by this function N(s) and H(s) in the Laplace domain. So, the HA of s is stable if its poles lie in the left half of the s-plane. Now, if you want to design a stable IIR filter, the analog filter's poles, if it is stable, will be in the left half.

So, in the digital domain, it will also be stable if my $j\omega$ axis in the s plane is mapped onto the unit circle, and the second is that the left half of the s plane should be marked inside the unit circle because these are the values where, if the poles lie, the system is stable.

So, this is the condition for stability in the z-domain. So, what we will do is use the first method, which is the approximation of the derivative, and in this case, we will first approximate a derivative, which is dx by dt.

$$\frac{dx(t)}{dt} \approx \frac{x[n] - x[n-1]}{T_s}$$

If $H(s) \rightarrow H(z)$, then $s = \frac{1-z^{-1}}{T_s}$

$$\frac{d^2x(t)}{dt^2} \approx \frac{x[n] - 2x[n-1] + x[n-2]}{T_s^2}$$

If $H(s) \rightarrow H(z)$, then $s^2 = \left(\frac{1-z^{-1}}{T_s}\right)^2$

$$H(z) = H_a(s)|_{s=\frac{1-z^{-1}}{T_s}}$$

How do we approximate this? So, this is $x[n] - x[n-1]$, and this difference is accounted for by how much duration? That is my sampling duration, T_s. So, if I need to do this conversion, what will I do? So, I have to map this. So, basically, what is happening here? So, this is basically if you see this as 1 and this as a delay. So, the delay is z inverse, right? So, this is the factor T_s. This is because this is s into x(s) in the Laplace domain. So, this s into x(s) is basically mapped as, so this 1 minus z inverse by T_s into H(z).

$$\frac{1 - z^{-1}}{T_s}$$

So let us see the second order differentiation. So if I take the second order derivative, I get this.

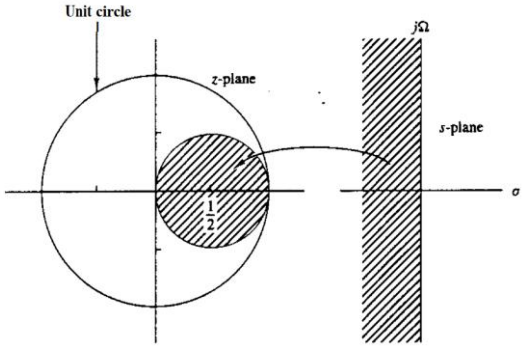
If $H(s) \rightarrow H(z)$, then $s = \frac{1-z^{-1}}{T_s}$

$$\frac{d^2x(t)}{dt^2} \approx \frac{x[n] - 2x[n-1] + x[n-2]}{T_s^2}$$

So now what we get is basically, if you see that this is $1 - 2z^{-2} + z^{-2}$ if I transform it into the z domain. So this is what I am getting. So, S squared is mapped to 1 minus z inverse by T_s squared.

$$s^2 = \left(\frac{1 - z^{-1}}{T_s}\right)^2$$

So, clearly S is mapped to $\frac{1-z^{-1}}{T_s}$; S squared is mapped to $\left(\frac{1-z^{-1}}{T_s}\right)^2$, So similarly, if you see S to the power of any value k, it will be mapped to this, and you can prove it by induction. So the approximation of the derivative is nothing but this mapping, this mapping. So, this is what we are getting. So, if you see, this desirable property is that the left half of the S plane is mapped onto the unit circle, and not only the unit circle, but it is also much less than the unit circle.



This is a circle with a half radius. So, this is a problem that only a very small area utilizes. So, you can only design low-pass filters and band-pass filters with relatively small resonant frequencies because you are not utilizing the full stable region; you are only utilizing a very small portion. So, to solve that another method is developed which is called the impulse invariance method. So here the objective is to design an IIR filter that has this impulse response h[n] by sampling the analog version, which is HA of t.

$$H_a(s) = \sum_{k=0}^{N} \frac{c_k}{s - p_k}$$
$$h_a(t) = \sum_{k=0}^{N} c_k e^{p_k t}$$

So, if I see the transfer function of the analog filter, this is like a sum, and this C_k can be obtained using the partial fraction method.

Now this will typically be the poles, of course, because at these values of S our response will go to infinity. So if I take the inverse Laplace, I get that it is ck, which are the coefficients e raised to the power of pkt, and if you have any confusion, just check the inverse Laplace table. So if we sample this impulse response because that is what we are supposed to do, we obtain a sample

version of the impulse response. So we get h(n) from this, which is h(n) Ts for different values of n again, I will keep n from 0 to ∞, not -∞ infinity, to keep it causal.

$$h[n] = h_a(nT_s),\ n = 0,1,2,\dots$$

$$h[n] = \sum_{k=0}^{N} c_k e^{p_k nT_s}$$

And then I get this, which is basically just plugging in at t; we are plugging in nT_s and this is basically how we got the h[n]. So there are, say, n poles, so we will go up to n, and then the next thing is that we will use the formula for the Z transform. So here, this value of h[n], we will plug in this result to get this expression, and just by arranging, I take out the C_k outside, and this term is basically an infinite GP series.

$$h[n] = \sum_{k=0}^{N} c_k e^{p_k nT_s}$$

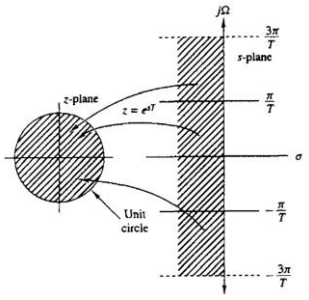
$$H(z) = \sum_{n=0}^{\infty} h[n] z^{-n} = \sum_{n=0}^{\infty} [\sum_{k=0}^{N} c_k e^{p_k T_s n}]\, z^{-n}$$

$$= \sum_{k=0}^{N} c_k \sum_{n=0}^{\infty} (e^{p_k T_s} z^{-1})^n = \sum_{k=0}^{N} \frac{c_k}{1 - e^{p_k T_s} z^{-1}}$$

$$z = e^{p_k T_s},\ k = 0,1,2,\dots,N$$

$$H(z)|_{z=e^{sT_s}} = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} H_a(s - j\frac{2\pi k}{T_s})$$

So, this can be evaluated to 1 by this, right? And this C_k is the factor. So technically, where are

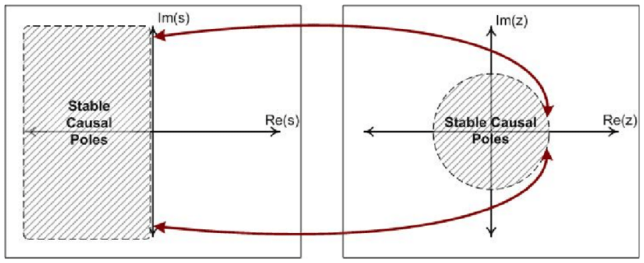


the poles? So poles are at e to the power of $P_k T_s$. So, in general, we can say that this is mapping. So, the poles of the analog are mapped to the poles of the digital. So, at z equals e power sts, we get the response. Now there is a situation here; the situation is that, in the continuous domain, if I have a frequency response like this h of f or say capital omega, which is the analog CTFT, what do we get in the DTFT? We get h of small omega, which is basically the replica and this is periodic with a period of 2π. So sampling makes replicas, periodic replicas of my spectra, right? And that is why these are repeated after one cycle; this is repeated, this is repeated, this is repeated. So if it is beyond this Nyquist, then there will be aliasing, and naturally, you have to keep it in the low-frequency region, which is half the sampling frequency. So what happens here is that I can design only low-pass filters or band-pass filters. I cannot design high-pass filters because of the spectrum analysis in the sampling process. Because if I design high-pass filters, they will merge and there will be aliasing.

So, the final method of this course or topic is the bilinear transform. And here what we do is convert the analog to digital while preserving the frequency response characteristics. So, this is the transformation that is

$$s = \frac{2}{T_s}\frac{1 - z^{-1}}{1 + z^{-1}}$$

So, here is a conformal mapping that will transform the $j\omega$ axis onto the unit circle only once. Unlike the simple z-invariance where you get copies like this, here the whole minus infinity to infinity will be mapped to the circle only once.



This will avoid aliasing of frequency components, and hence you can design high-pass filters using this method. So, this is what I was talking about: your stable causal poles will be inside the unit circle to make them stable causal poles in the Z domain. So, here what we will do is, instead of substituting a finite difference, which we did for the approximation of the derivative, we will integrate the derivative on the left side and approximate the integral using a numerical method, which is the trapezoidal rule. So, say we have dy/dt equal to x(t). So, if we integrate this and take one sample duration from $(n-1)ts$ to nts, we do the same for the right side.

$$\frac{dy(t)}{dt} = x(t)$$

$$\therefore \int_{(n-1)T_s}^{nT_s} dy(t) = \int_{(n-1)T_s}^{nT_s} x(t)dt$$

$$\therefore y(t)|_{(n-1)T_s}^{nT_s} = \int_{(n-1)T_s}^{nT_s} x(t)dt$$

$$\therefore y[n] - y[n-1] = \frac{x[n] + x[n-1]}{2}T_s$$

So, on the left side, we do the integration, so dy will be y, and the limits will be $(n-1)T_s$ to $n\,T_s$, and on the right side, we will evaluate this integration using the trapezoidal rule. So, here we get y($n\,T_s$) now, for y($n\,T_s$), it is basically y[n], and similarly, y[n-1] T_s is y[n-1]. And here the trapezoidal rule is basically the area under the curve, right? So this is you getting x[nTs], which is $x[n] + x[n-1]$ Ts, which is $\frac{x[n]+x[n-1]}{2}T_s$. So this is the formula to calculate the area of a trapezoid.

So, by 2 into T_s. So this is basically a trapezoid. So we are working in this area. So, this is nT_s , and this is, sorry, this is n T_s, and this is (n-1)T_s. So now this is what we have in the previous stage. So if we take the Z-transform of both sides, this is what we get from this, and here is what we get. And now, doing simplification by taking Y(Z) common,

$$y[n] - y[n-1] = \frac{x[n] + x[n-1]}{2} T_s$$

Taking Z-transform of both sides

$$Y(z) - Y(z)z^{-1} = [\frac{X(z) + z^{-1}X(z)}{2}]T_s$$

$$\therefore Y(z)(1 - z^{-1}) = \frac{1 + z^{-1}}{2} T_s X(z)$$

$$\therefore X(z) = \frac{2}{T_s}\frac{1 - z^{-1}}{1 + z^{-1}} Y(z)$$

$$X(s) = sY(s)$$

$$s = \frac{2}{T_s}\frac{1 - z^{-1}}{1 + z^{-1}}$$

$1 - z^{-1}$, and taking X(Z) common, this is what we get and if we rearrange, we get $\frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}}$. So this is what we get, and this is the thing that we got in the analog domain: since x(t) was dy/dt, in the Laplace domain, x(s) is sy(s). Now, if you compare these two, you get this mapping that we discussed in the beginning.

So, this is basically our bilinear transform to map S to Z. Thank you so much. Hope you have enjoyed it. Thank you very much. Have a nice day.