

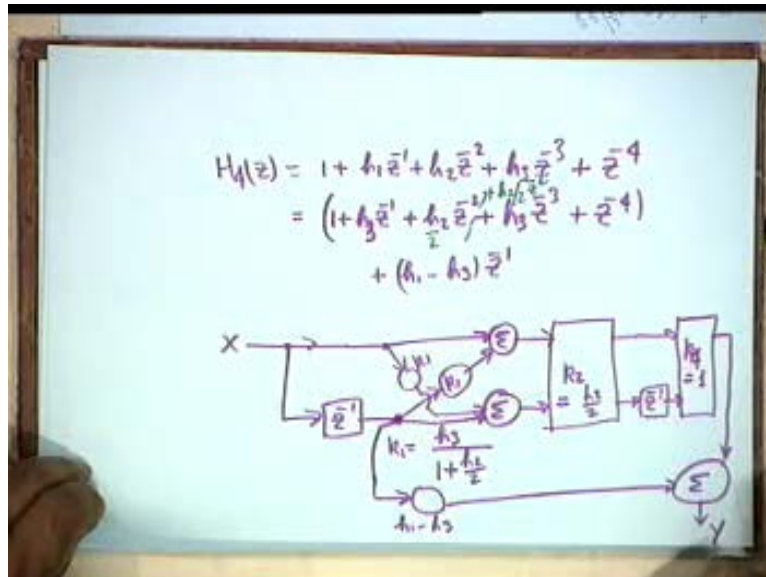
Digital Signal Processing
Prof. S.C. Dutta Roy
Department of Electrical Engineering
Indian Institute of Technology, Delhi
Lecture – 33
FIR Lattice (Contd.) and Digital Filter Design

This is the 33rd lecture on DSP and our topic for today is FIR lattice. We also start digital filter design today. In the 32nd lecture, we talked about general FIR lattice synthesis and then we went to linear phase transfer functions.

We considered all possible cases, namely symmetrical, asymmetrical, even length and odd length. Then, we started discussing general FIR, non linear phase filter with the highest power coefficient = + 1 or – 1, because $k_m = + 1$ or – 1 creates a problem in going to the next stage. We had taken one example. A single multiplier FIR lattice incidentally does not exist so far; unless you find one by the end of this course, it shall remain an open problem in future also. For IIR, it exists but for FIR it does not. To continue with the problems, last time we considered $H_4(z) = 1 + h_1z^{-1} + h_2z^{-2} + h_3z^{-3} + z^{-4}$ which is of odd length and highest power coefficient = + 1. We said that the decomposition would be $1 + h_3z^{-1} + h_2z^{-2} + h_3z^{-3} + z^{-4}$ which is linear phase, plus $(h_1 - h_3)z^{-1}$. That is, we use the higher power coefficient in the linear phase part. I could have used h_1z^{-3} for symmetry, but then I have to add $(h_3 - h_1)z^{-3}$.

In other words the tap had to be after three delays. But after three delays, calculation of the tap weight becomes a little more involved. It is much easier to use $(h_1 - h_3)z^{-1}$. The structure would be like that shown in the next figure. Note that in the linear phase part, the middle term has been broken into two equal parts, as we illustrated earlier. Also note that $k_3 = 0$ and $k_4 = 1$. The output of the linear phase part has to be added to $(h_1 - h_3)z^{-1}$, which we derive by tapping at the end of the first delay.

(Refer Slide Time: 03:02 -07:33)



Let us consider the same case with the highest power coefficient = - 1. Now we shall have to bring in asymmetry to decompose into a linear phase plus some other transfer function. Specifically, let $H_4(z) = 1 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} - z^{-4}$. We take $-h_3 z^{-1}$ instead of $+h_1 z^{-1}$ and would have to add $(h_1 + h_3) z^{-1}$. If the filter is anti-symmetrical, then the middle coefficient must be 0 so that $h_2 z^{-2}$ has to be added. As far as realization is concerned, I now require two tappings, one after z^{-1} and the other after z^{-2} . Let us see the realization of these two terms which no longer constitute a linear phase transfer function.

(Refer Slide Time: 07:40 – 10: 07)

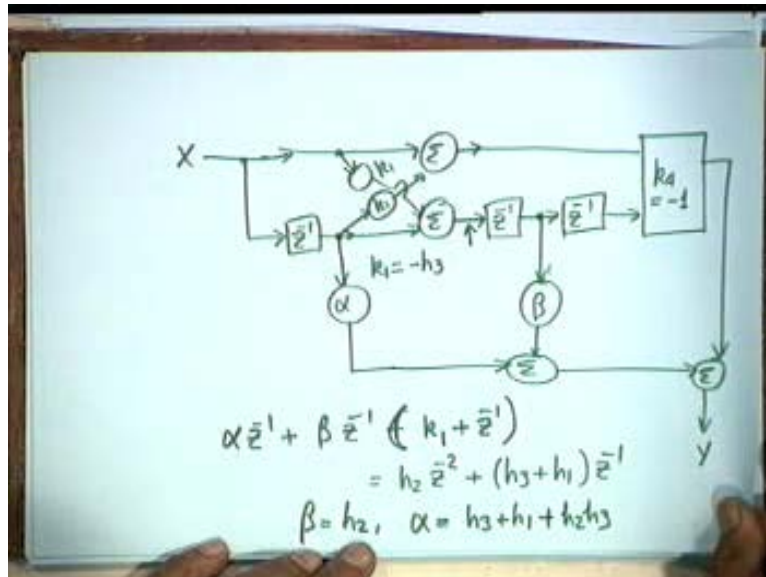
$$\begin{aligned} \text{Odd length; Asymmetric coeff of } z^{-4} &= -1 \\ H_d(z) &= 1 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} - z^{-4} \\ &= (1 - h_3 z^{-1} + h_3 z^{-3} - z^{-4}) \\ &\quad + h_2 z^{-2} + (h_3 + h_1) z^{-1} \end{aligned}$$

x ———

The structure is shown in the figure. Note that here $k_2 = k_3 = 0$. So we shall have two delays after the lattice section, and finally we shall have $k_4 = -1$. We have to take two tappings now. We have synthesized the linear phase transfer function by observation. Let the first tap weight be α and the second one, after the second delay be β . The only thing that remains to be found are the values of α and β .

The signal $\alpha z^{-1} + \beta z^{-1} (k_1 + z^{-1})$ should be equal to $h_2 z^{-2} + (h_3 + h_1) z^{-1}$. Now match the coefficients and obviously $\beta = h_2$, $\alpha = h_3 + h_1 - h_2 k_1$, and k_1 is $-h_3$.

(Refer Slide Time: 10:18 - 13:54)



Let us take the other two cases now. Next case is, even length and highest power coefficient = + 1. Let us take a 5th order transfer function $H_5(z) = 1 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + h_4 z^{-4} + z^{-5}$ and the decomposition now would be $1 + h_4 z^{-1} + h_3 z^{-2} + h_3 z^{-2} + h_4 z^{-4} + z^{-5} + (h_1 - h_4) z^{-1} + (h_2 - h_3) z^{-2}$. The first one is linear phase and I can synthesize it in the usual manner. I have to show the first two lattices explicitly because I want to tap the signals after the first two delays; and I cannot show them as boxes.

(Refer Slide Time: 14:03 - 16:58)

Ex Even length: H.P. coefft = +1

$$H_5(z) = 1 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + h_4 z^{-4} + z^{-5}$$

$$= (1 + h_4 z^{-1} + h_3 z^{-2} + h_3 z^{-3} + h_4 z^{-4} + z^{-5})$$

$$+ (h_1 - h_4) z^{-1} + (h_2 - h_3) z^{-2}$$

$k_1 = h_4 / (1 + h_3)$ $k_2 = h_3$

Now k_1 is $h_4 / (1 + h_3)$ and k_2 is h_3 . What are the next two? The next two are k_3 and k_4 which are 0 and therefore I shall have two delays, and finally $k_5 = +1$. The output shall combine with the two tapped signals. Let us call the tap weights as α and β .

(Refer Slide Time: 17:28 - 17:48)

Ex Even length: H.P. coefft = +1

$$H_5(z) = 1 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + h_4 z^{-4} + z^{-5}$$

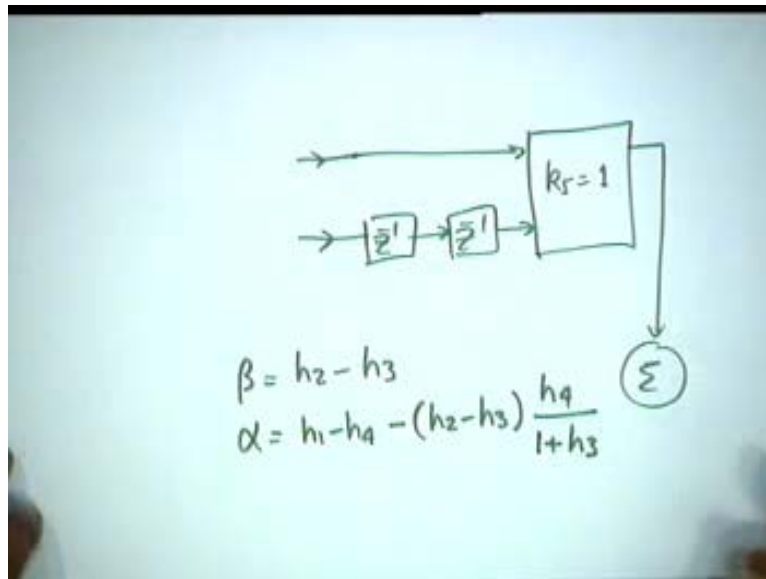
$$= (1 + h_4 z^{-1} + h_3 z^{-2} + h_3 z^{-3} + h_4 z^{-4} + z^{-5})$$

$$+ (h_1 - h_4) z^{-1} + (h_2 - h_3) z^{-2}$$

α β $k_1 = h_4 / (1 + h_3)$ $k_2 = h_3$

And if you follow the same procedure you get $\beta = h_2 - h_3$ and $\alpha = h_1 - h_4 - (h_2 - h_3) \times h_4 / (1 + h_3)$.

(Refer Slide Time: 17:07 - 8:25)



Finally, consider the case of even length and coefficient of highest power term = - 1; so we have to go for anti-symmetry. Let $H_5(z) = 1 + h_1z^{-1} + h_2z^{-2} + h_3z^{-3} + h_4z^{-4} - z^{-5}$ and the decomposition now would be $(1 - h_4z^{-1} - h_3z^{-2} + h_3z^{-3} + h_4z^{-4} - z^{-5}) + (h_4 + h_1)z^{-1} + (h_2 + h_3)z^{-2}$. Also we have $k_2 = -h_3$, $k_1 = -h_4/(1 - h_3)$, $k_3 = k_4 = 0$ and $k_5 = -1$.

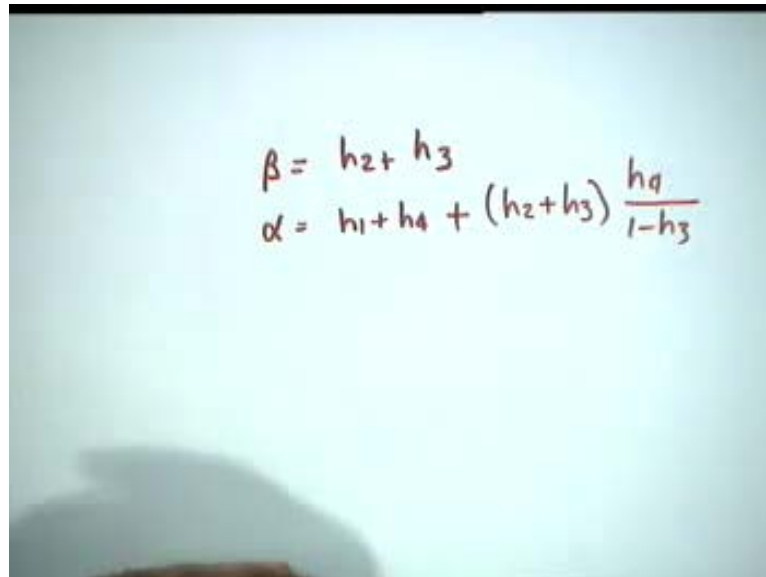
(Refer Slide Time: 18:42 - 20:51)

Even Length: Coefft of H.P. Term = -1

$$H_5(z) = 1 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + h_4 z^{-4} - z^{-5}$$
$$= (1 - h_4 z^{-4} - h_3 z^{-2} + h_3 z^{-3} + h_4 z^{-4} - z^{-5})$$
$$+ (h_4 + h_1) z^{-1} + (h_2 + h_3) z^{-2}$$
$$k_2 = -h_3$$
$$k_1 = \frac{-h_4}{1-h_3} = \frac{-h_4}{1-h_3}$$

The values of α and β can be calculated easily. β has to be $h_2 + h_3$ and $\alpha = h_1 + h_4 - \beta k_1$. With a little bit of maturity, you will not have to do the intermediate steps but write these values directly. The highest power coefficient being either + 1 or - 1 actually is a welcome step, because it reduces your effort. If it is linear phase, it reduces the effort considerably. If it is not a linear phase, and neither is the coefficient of the highest power term equal to + 1 or - 1, even then with a little bit of ingenuity, you can apply some of these simplifications.

(Refer Slide Time: 20:56 - 23:39)



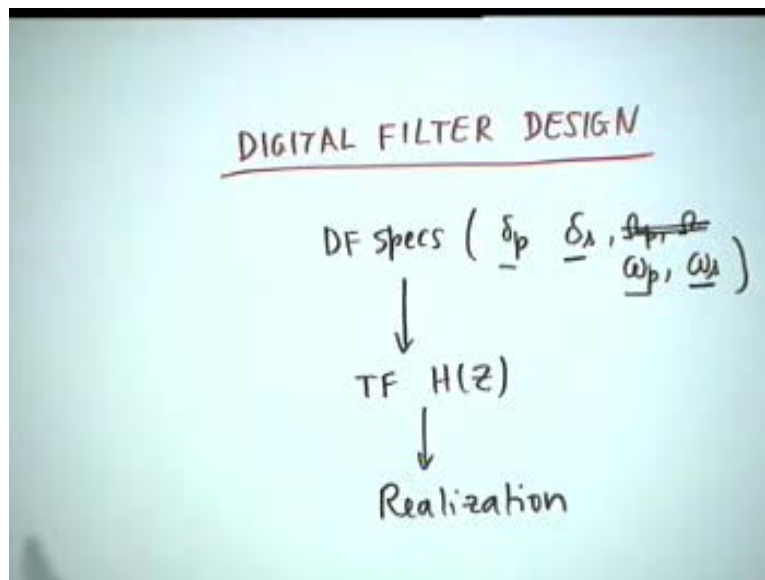
The image shows a whiteboard with two handwritten equations in red ink. The first equation is $\beta = h_2 + h_3$. The second equation is $\alpha = h_1 + h_4 + (h_2 + h_3) \frac{h_4}{1 - h_3}$.

And with that, we close our discussion on FIR realization of digital filters. We next go to the most important topic, that is Digital Filter Design. We have already done some examples. We have done first order low pass, first order high pass, and then second order (the minimum order) band pass and band stop. Given a problem, we should be able to solve it even without sophisticated digital filter design techniques. But if the specifications are very tight, for example, if Δ_p is of the order of 0.1dB and Δ_s is 120dB, the order has to be high and it may not be possible to do it by cascading lower order ones, by trial and error. You shall have to know what the order is and how to go about it. But given a practical situation, most of the practical engineers use synthesis by analysis. That is, they cascade first and second order filters and find out whether it does the job. They use MATLAB to find out the performance and if it does not satisfy the specs, then they keep changing. So, the problem is that the specifications are given in terms of tolerance in the pass band, tolerance in the stop band and the band edges; you have to find the transfer function.

Now, if it is a simple low pass filter, then you know Δ_p , Δ_s , ω_p and ω_s , which are the digital filter specs. On the other hand, if it is a bandpass filter then you require ω_{p1} , ω_{p2} or the bandwidth, the center frequency and also ω_{s1} and ω_{s2} . Δ_s may differ from one stop band to another stop band.

You can have multi band pass or multi band stop or a combination filter which is, for example, low pass plus something else. From the specifications, your job is to find out a transfer function $H(z)$. The specifications we are talking about are magnitude specs. Usually we do not design digital filters to satisfy phase specifications, although there are situations where this is to be done. But for phase, what is our objective in digital filter design? The phase should be linear; we cannot obtain this with an IIR filter but we can obtain this exactly with FIR. In IIR, phase is a consideration but usually we design for satisfying the magnitude specs; whatever phase comes, we accept it and then try to compensate for the non linearity of phase by cascading all pass filters. All pass filters are everywhere. Whichever way you want to go, all pass filter shall accompany you, whether you like it or not. So the transfer function has to be found and once you obtain $H(z)$, then with the knowledge you have already acquired, you realize the same either by software (write up a programme with a dedicated PC or use a programmable chip) or by hardware. If it is very complicated, then you design your own chip and fabricate.

(Refer Slide Time: 23:52 - 29:00)



Now, IIR and FIR design follow entirely different routes and they are very different from each other. IIR design usually can be made with a degree of confidence, but not FIR. Although FIR is advantageous from linear phase and stability considerations, it is not easy to design it. With IIR,

you have to check the stability. But the advantage of IIR is that you can use much lower orders. That is, the hardware complexity or the software complexity is much lower than that of the corresponding FIR filter. And the other advantage is that IIR filters usually can be designed analytically to the given specifications, to satisfy the specifications exactly, except for word length errors i.e. except for the errors arising out of using a finite number of bits. FIR filters can also be designed analytically but the level of confidence is much lower. The filter that you design may or may not satisfy the specs. What you have to do is that after you complete the design, you have to compute its performance and find out if it satisfies the specs. If it does not, then you go back and change the parameters. So it is an iterative process, whereas in IIR, it is a one step process and this comes about because IIR specs can be transformed to an analog filter specs by using a transformation from ω (normalized digital frequency) to analog frequency Ω .

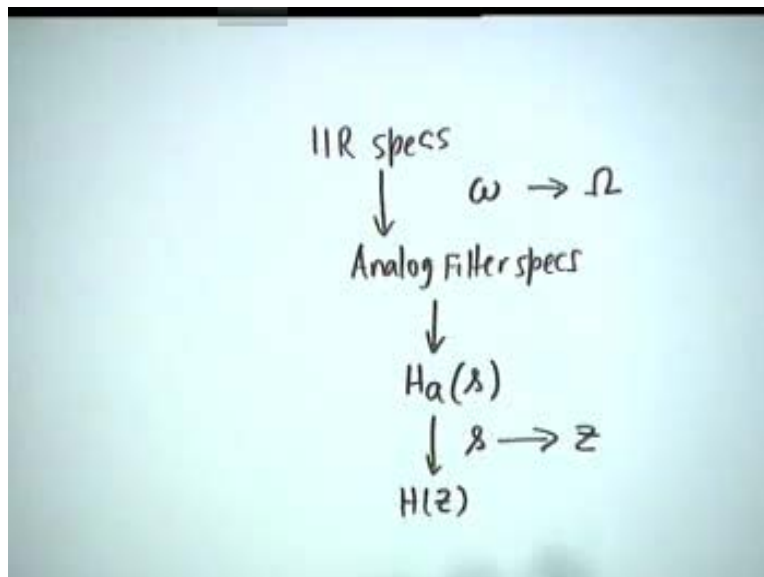
From the analog filter specs, by the well established approximation techniques, you can carry out the design to satisfy the specs. And this is normally done by transforming the analog filter specs to a normalized analog low pass filter and then using the transformation back. So you design $H_a(s)$ and then apply another transformation to go from s to the z domain to obtain $H(z)$. This is the route that is followed for IIR analytical digital filter design. One might ask, in this age of computers, when very large capability computers are available, why should we go through this analytical technique at all? Why can we not use a very blind approach?

As an example, I estimate that a tenth order IIR filter should do for the given specs. Then I require 20 unknown constants, 10 in the numerator and 10 in the denominator. This number can be reduced by one by making the constant term in the denominator equal to unity; so there are 19 coefficients. I guess these coefficients and I start with some initial values and I put it to the MATLAB and find out what the results are. If it does not satisfy the specs, then we keep on changing. We do not know whether the process will converge or diverge.

In the initial stages of development of digital filters, this is what was done. However, at the present time, sophisticated computer programmes are available for digital filter design, be it IIR or FIR. Companies do use them. They are very costly because they have been derived by putting

many man hours. They are usually proprietary for a company who do not divulge it to others. There are some open programmes available on the internet which you will have to modify to suit your designs. But then if you can design it analytically, there is nothing like it because there is a measure of confidence which cannot be matched by any other filter design. The people in industry argue that for an engineer, something which works is good enough; why bother about analytical design? And fortunately since analytical designs are available, we shall use computer aided design algorithms only if we have to. If you do not have that much of man hours at your disposal, then go ahead and use analytical techniques.

(Refer Slide Time: 31:10 - 36:34)



Now, let us look at why is it that IIR filters are designed from analog filters. This is also historical. Analog filter design is an advanced art; almost everything is known about analog filters. They usually give rise to closed form expressions. We have discussed the Butterworth and Chebyshev filters; they give rise to elegant expressions and you can find out the transfer function exactly. Not only that, for elliptic filters, tabulated handbooks of filter designs are available and therefore you may not have to do the intermediate steps. You may simply consult a handbook and say these are the specs and there is the design.

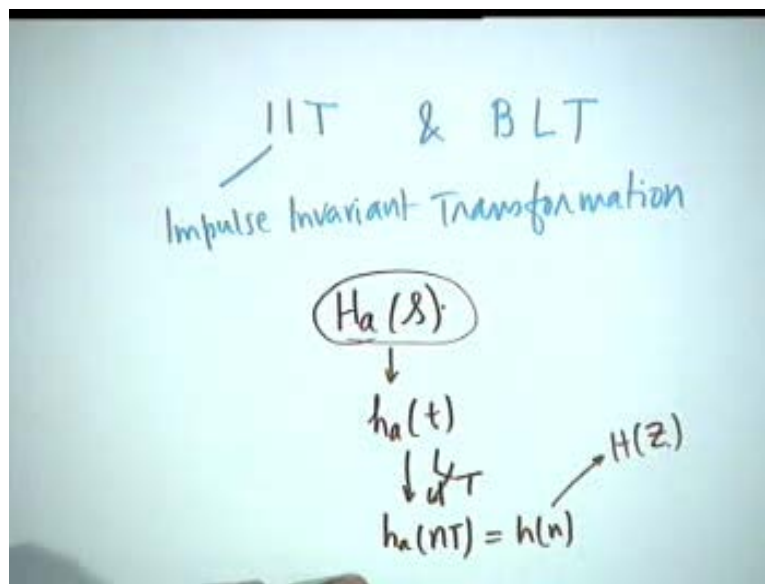
So, you can take the analytical design of the analog filter, do your transformation from s to z plane, obtain the filter transfer function, realize it, and make an analysis with the finite number of bits. Find out whether they satisfy the specs or not and then make incremental adjustments. In analog filters also, after putting it on hardware you have to check experimentally to see if it satisfies the specs; if not then you have to tune the filter. Tuning is required in digital filters because of finite word length. Otherwise, if infinite number of bits were available, your design would have been perfect. So these are the reasons why analog filters form the parent filter for IIR digital filter design.

There are various kinds of transformations that are available. The two transformations that are generally used go by the names of IIT and BLT. IIT stands for Impulse Invariant Transformation which is very simple conceptually but has its own disadvantages. And BLT is the Bilinear Transformation. Impulse Invariant Transformation is always amenable to aliasing distortion. Aliasing cannot be avoided in impulse invariant transformation whereas bilinear transformation is free from aliasing error. Even then why do we discuss IIT? It is because IIT is very simple conceptually and the calculations are also much simpler; so if the distortion can be contained within a reasonable limit then it does the job. For most of the IIR filters for bio medical signal processing, for example, people do not want to go into the complication of bilinear transformation. They simply use impulse invariant transformation and for an engineer, anything that works is good enough. So people still use IIT.

Mitra's second edition, does not discuss IIT at all. In Impulse Invariant Transformation, one takes a given analog filter obtained in the same manner as discussed earlier. That is, by using some transformation, you transform ω to Ω ; then you decide on an analog filter $H_a(s)$. Obtain its impulse response $h_a(t)$ and then sample it. Sample the impulse response to get $h_a(nT)$ and call this $h(n)$. From $h(n)$, take z transform and obtain your $H(z)$. The distinctive characteristic of this procedure is that the impulse response remains invariant. That is, the digital filter impulse response is the sampled version of the analog filter impulse response. And if sampling theorem has been respected in the process, then they should be a one to one representation. That is, given $h(n)$, you should be able to construct $h_a(t)$ and given $h_a(t)$, you should of course be able to obtain

$h(n)$. The problem of aliasing occurs because no transfer function in practice is band limited. If you want to respect the sampling theorem, the sampling must be such that the sampling frequency is greater than or equal to twice the highest frequency contained in $H_a(s)$. No transfer function can stop abruptly at a particular frequency because they would disobey the fundamental Paley-Wiener criterion. They would be non causal and unrealizable. No transfer function is therefore band limited which means that some amount of aliasing distortion is bound to occur. So long as you contain this to within the tolerable limits, it can be used.

(Refer Slide Time: 39:07 - 44:01)



The procedure is extremely simple. Suppose $H_a(s)$, for simplicity, has only distinct poles p_i , $i = 1$ to N . Then $H_a(s)$ can be expanded in partial fractions as $\sum A_i/(s - p_i)$. What happens when I have repeated poles? We will come back to it later. Suppose we have N number of poles in the transfer function which are distinct; p_i can be real or can be complex. But the real part of p_i , if it is complex, must be negative, i.e. poles must be in the left half of s plane. Now for simplicity, I shall write only summation, I will not put the limits which shall be from 1 to N . $h_a(t)$ therefore shall be $\sum A_i e^{p_i t} u(t)$. And if I sample this at intervals of T , then obviously what I get is, $\sum A_i e^{p_i nT} u(nT)$. I identify this as my $h(n)$, so my transfer function becomes $H(z) = \sum A_i / (1 - e^{p_i T} z^{-1})$.

$e^{p_i T} z^{-1}$). From the given analog transfer function, I go to the digital transfer function in a very simple manner. What do I do? I replace $1/(s - p_i)$ by $1/(1 - e^{p_i T} z^{-1})$.

(Refer Slide Time: 44:09 - 46:40)

The image shows a whiteboard with the following handwritten equations:

$$H_a(s) = \sum_{i=1}^N \frac{A_i}{s - p_i}$$

$$h_a(t) = \left(\sum A_i e^{p_i t} \right) u(t)$$

$$h(n) \triangleq h_a(nT) = \sum A_i \left(e^{p_i T} \right)^n u(n)$$

$$H(z) = \sum_{i=1}^N \frac{A_i}{1 - e^{p_i T} z^{-1}}$$

This means that the pole at $s = p_i$ is transformed to the pole at $z = e^{p_i T}$. If this transformation is to work, then a stable analog filter should be transformed to a stable digital filter. The transformation we are using is $z = e^{sT}$, which I can write as $e^{\sigma T} e^{j\Omega T}$. And therefore magnitude $z \geq$ or < 1 for $\sigma \geq 0$, or $\sigma < 0$. This means that a pole in the left half of the s plane transforms to a pole inside the unit circle. The $j\Omega$ axis transforms to the unit circle and the right half plane transforms to outside the unit circle. Thus a stable analog filter shall transform to a stable digital filter. We look at it again. When σ is < 0 , that is the pole is in the left half plane, then magnitude $z < 1$; therefore this transforms to a point inside the unit circle. Thus a stable pole goes to a stable pole. The $j\omega$ axis for which $\sigma = 0$ goes to magnitude $z = 1$ which is the unit circle. And the right half plane where no poles should be located goes outside the unit circle. So a stable filter transforms into a stable filter.

(Refer Slide Time: 46:43 - 49:00)

$$s = p_i \rightarrow z = e^{p_i T}$$

$$z = e^{\sigma T} = e^{\sigma T} e^{j\Omega T}$$

$$|z| \begin{matrix} \geq 1 \\ < 1 \end{matrix} \text{ for } \sigma \begin{matrix} \geq 0 \\ < 0 \end{matrix}$$

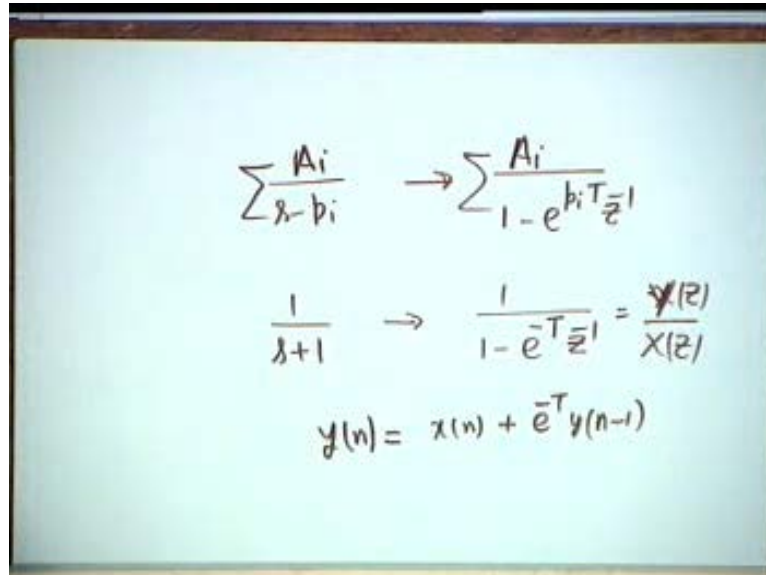
The weakness of this transformation is that $H_a(s)$ may not be band limited. How does it reflect in the transformation $1/(s - p_i)$ to $1/(1 - e^{p_i T} z^{-1})$? For the pole at $s = p_i$, we find the corresponding pole in z plane; we have no control over the zeros of the transfer function. Zeros from the analog plane are not transformed according to this relation. Whatever zeros come, we will have to accept and these arise because $H_a(s)$ is not band limited. This is how it is reflected. Since we have no control over the zeros, the transfer function we get shall not be exactly what we wanted. There are two ways of looking at it. One is aliasing point of view and the other is that we have no control over the zeros.

On the other hand, in bilinear transformation, we transform poles as well as zeros. In the Impulse Invariant Transformation, if I have a normalized first order analog filter, it transforms to $1/(1 - e^{-T} z^{-1}) = Y(z)/X(z)$. The difference equation would be $y(n) = x(n) + e^{-T} y(n - 1)$; this is of first order. First order digital filter is derived from first order analog filter. There is no order change.

Similarly, in the bilinear transformation also, there is no order change as we shall see. If the impulse response is invariant, does it imply that the response to a unit step is also invariant? In

other words is the step response invariant? The answer is no. The step invariant transformation is quite different. We shall demonstrate this in the next class and that is where we stop today.

(Refer Slide Time: 49:56 – 54:06)



The image shows a whiteboard with three lines of handwritten mathematical equations. The first line shows a partial fraction decomposition in the s-domain being transformed into a geometric series in the z-domain. The second line shows a specific example of this transformation for a pole at s = -1. The third line shows the resulting difference equation for the step response.

$$\sum \frac{A_i}{s - p_i} \rightarrow \sum \frac{A_i}{1 - e^{p_i T} z^{-1}}$$
$$\frac{1}{s+1} \rightarrow \frac{1}{1 - e^{-T} z^{-1}} = \frac{Y(z)}{X(z)}$$
$$y(n) = x(n) + e^{-T} y(n-1)$$