

Digital Communication using GNU Radio

Prof. Kumar Appaiah

Department of Electrical Engineering

Indian Institute of Technology Bombay

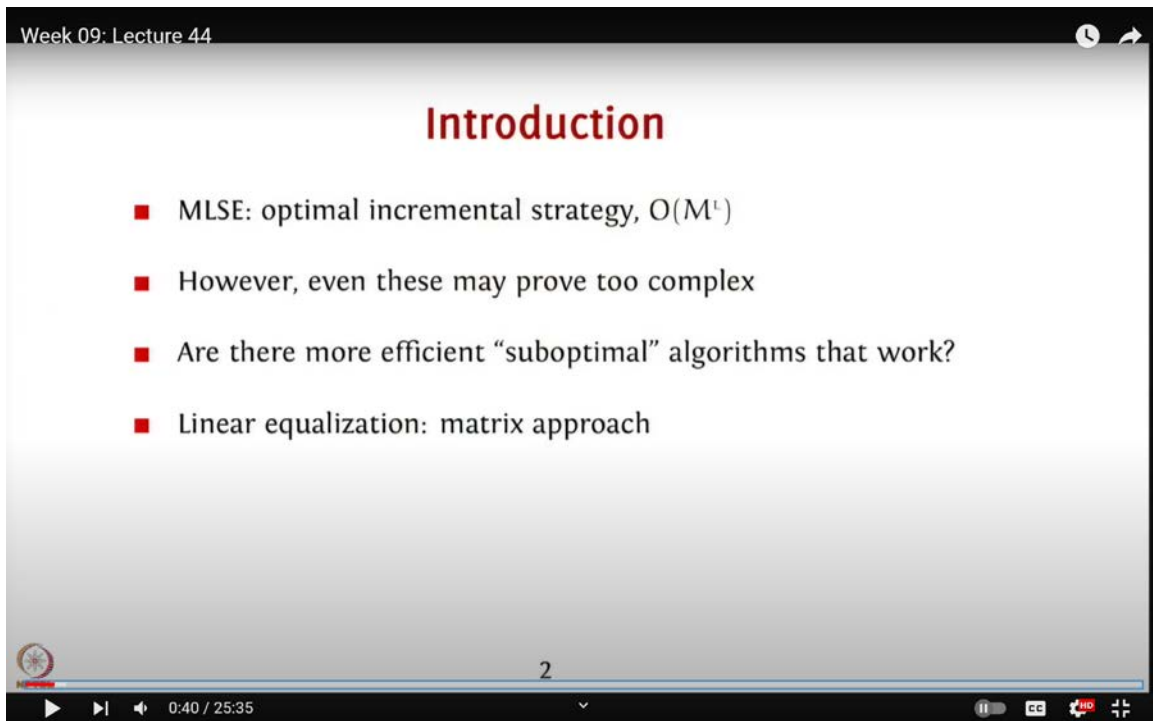
Week-09

Lecture-44

Suboptimal Channel Equalisation: Zero-forcing Receiver

Hello, and welcome to this lecture on digital communication using GNU Radio. I'm Prof. Kumar Appaiah from the Department of Electrical Engineering at IIT Bombay. In today's session, we will delve into the concept of suboptimal channel equalization. In the past few lectures, we discussed the optimal detection strategy, Maximum Likelihood Sequence Estimation (MLSE).

(Refer Slide Time: 00:40)



The screenshot shows a video player interface for a lecture. The title bar at the top left reads "Week 09: Lecture 44". The main content area has a red heading "Introduction" and a bulleted list of four points:

- MLSE: optimal incremental strategy, $O(M^L)$
- However, even these may prove too complex
- Are there more efficient "suboptimal" algorithms that work?
- Linear equalization: matrix approach

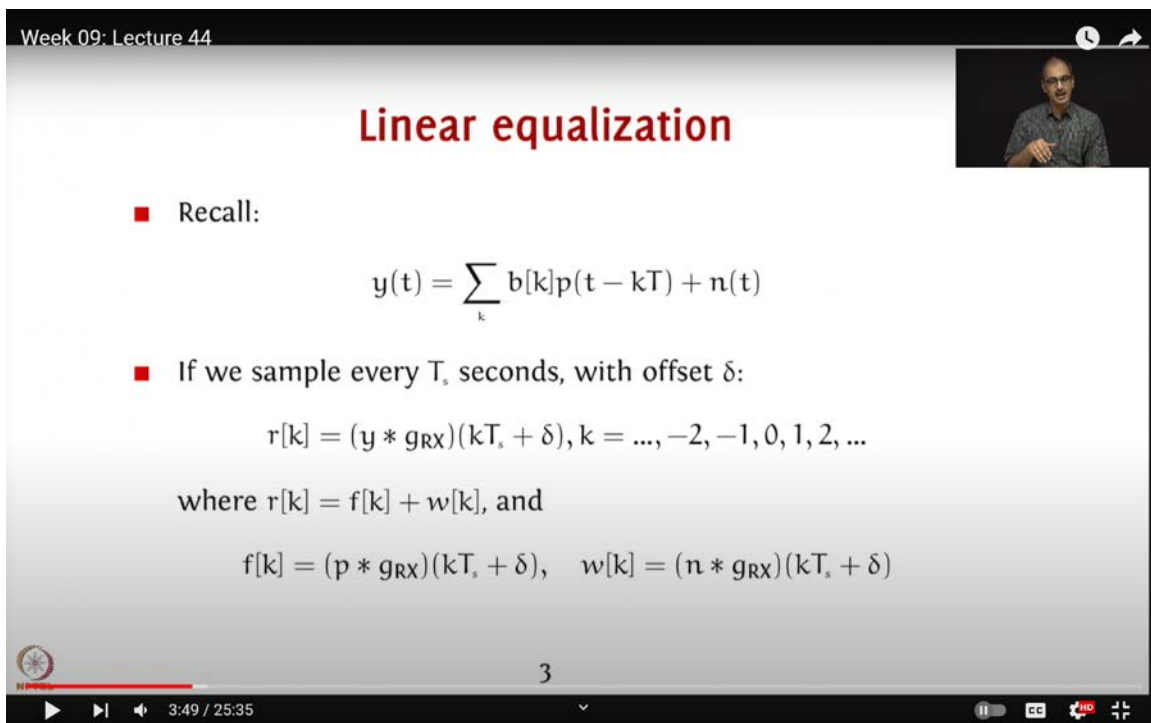
At the bottom of the slide, the number "2" is centered. The video player controls at the bottom show a play button, a progress bar at 0:40 / 25:35, and various icons for volume, closed captions, and full screen.

We also touched on the incremental approach, wherein we considered whether it was possible to make decisions progressively, based on the symbols observed so far, rather than

evaluating all possible combinations. This led us to the Viterbi algorithm, which helps in making decisions under certain conditions, while keeping track of surviving paths. The Viterbi algorithm is a powerful tool, and it generally works quite well.

However, there are instances where even the Viterbi algorithm can become too complex to implement practically. The algorithm necessitates the retention of multiple survivor paths in memory, which can lead to complications. In such cases, particularly when you have a high Signal-to-Noise Ratio (SNR), you might not want to deal with such complex algorithms. This raises an important question: Are there simpler, more efficient, and less computationally expensive suboptimal algorithms that still deliver satisfactory performance? The answer is yes, and that is where our discussion on linear equalization begins.

(Refer Slide Time: 03:49)



The image shows a video player interface for a lecture slide. The slide title is "Linear equalization". The content includes a "Recall:" section with the equation $y(t) = \sum_k b[k]p(t - kT) + n(t)$. Below that, it states "If we sample every T_s seconds, with offset δ :" followed by the equation $r[k] = (y * g_{RX})(kT_s + \delta), k = \dots, -2, -1, 0, 1, 2, \dots$. It then says "where $r[k] = f[k] + w[k]$, and" followed by the equations $f[k] = (p * g_{RX})(kT_s + \delta), w[k] = (n * g_{RX})(kT_s + \delta)$. The slide is part of "Week 09: Lecture 44" and is slide number 3. The video player shows a progress bar at 3:49 / 25:35.

Linear equalization can be thought of as a matrix-based or filtering-based approach, where we attempt to construct something akin to an inverse filter for the channel. Some popular techniques within this framework include the zero-forcing equalizer and the minimum

mean square error (MMSE) equalizer. These methods offer lower complexity and are useful in scenarios where fully optimal solutions may be overkill. Today, we will focus on the zero-forcing equalizer.

Let's revisit the channel model we've been using. The received signal, $y(t)$, can be expressed as a sum of delayed pulses, $b_k p(t - kT)$, with additive white Gaussian noise (AWGN). Here, T represents the symbol duration, the time per symbol. Now, suppose that at the receiver, we sample the signal at a different rate, say every T_s seconds, potentially with an offset δ . The received samples can then be expressed as the convolution of the received signal y with the receive filter $g_{RX}(t)$, evaluated at $kT_s + \delta$, where δ is a timing offset between zero and T_s . Thus, we obtain samples at intervals of T_s .

(Refer Slide Time: 04:29)

Week 09: Lecture 44

Linear equalization

- To find the statistics of the noise $w[k]$, we note that $w(t) = (n * g_{RX})(t)$ is zero mean Gaussian with autocorrelation

$$2\sigma^2 \int g_{RX}(t) g_{RX}^*(t - \tau) dt = 2\sigma^2 (g_{RX} * g_{R,MF})(t)$$
 where $g_{R,MF}(t) = g_{RX}^*(-t)$. So, $w[k] = (n * g_{RX})(kT_s + \delta)$ has autocovariance

$$C_w[k] = \text{cov}(w_{n+k}, w_n) = 2\sigma^2 \int g_{RX}(t) g_{RX}^*(t - kT_s) dt$$
- Note: depends on *sample spacing* T_s , not symbol spacing T !

Scroll for details

4:29 / 25:35

Note that T_s could be equal to T , or it could be less than T , but typically it is not greater than T , since oversampling beyond T would cause loss of information. Now, the received sample r_k can be expressed as $r_k = f_k + w_k$, where f_k is the signal component, specifically,

p convolved with g_{rx} , evaluated at $kT_s + \delta$, and w_k is the noise component, given by the convolution of the noise with g_{rx} , also evaluated at $kT_s + \delta$.

A key aspect to remember is that after the convolution with the receive filter, the noise may no longer be independent and identically distributed (i.i.d.). In fact, the noise could become correlated, which requires careful consideration. To understand the noise statistics, we examine the term n convolved with $g_{rx}(t)$. The expected value of this noise component is zero (since the mean of AWGN is zero), and the autocorrelation function can be defined as $2\sigma^2 \int g_{rx}(t)g_{rx}^*(t - \tau)dt$. This expression for the autocorrelation involves the factor $2\sigma^2$ because we are dealing with the power spectral density N_0 , where $N_0/2 = \sigma^2$, the noise variance per dimension. For complex-valued signals, this becomes $2\sigma^2 g_{rx} * g_{mf}(t)$, where g_{mf} is the matched filter, which is defined as $g_{rx}^*(-t)$.

Therefore, the noise component w_k , which is n convolved with g_{rx} and evaluated at $kT_s + \delta$, has an autocovariance given by $2\sigma^2 \int g_{rx}(t)g_{rx}^*(t - kT_s)dt$. It's important to note that this autocovariance depends on the sample spacing T_s , not on the symbol spacing T . If you wish to revisit how we derived these expressions, they directly follow from our earlier discussions on signal space concepts.

By using the signal space framework, we can further analyze and understand the impact of noise and interference, leading to more effective suboptimal equalization strategies like the zero-forcing receiver.

Essentially, you are filtering the noise and leveraging that filtering to calculate the discrete covariance appropriately. That's the key idea here, filtering the noise and working out the statistics. A crucial point we're emphasizing is that it's the sample spacing, not the symbol spacing, that matters in this context. Now, let's walk through an illustrative example to make this clearer.

In this example, our symbol duration, T , is set to 2 seconds because the transmit pulse, $g_{tx}(t)$, is defined over the interval from 0 to 2 seconds. Essentially, we're sending one symbol every 2 seconds. However, at the receiver, we choose the sampling interval, T_s , to

be 1 second, so we're effectively sampling at twice the rate. Now, for simplicity, let's assume we're sending only one symbol, $b_0 = 1$, and nothing else.

(Refer Slide Time: 07:37)

Week 09: Lecture 44

Linear equalization

- For our running example, recall

$g_{TX}(t)$

$g_C(t)$

$p(t)$

- $T = 2$. Let us choose $T_s = 1$
- Easy to see that $C_w[k] = 2\sigma^2\delta[k]$ ←
- Sampled response for $b[0]$ is: $\{\dots, 0, 1, \frac{1}{2}, -\frac{1}{2}, 0, \dots\}$

5

7:37 / 25:35

When you send $b_0 = 1$, the received signal will be a shifted version of the pulse. If you sample the received signal using an appropriate matched filter, say, a rectangular pulse over $T_s = 1$, you will get specific sample values. For example, after filtering, your first sample will be 1, the second will be 0.5, the third will be -0.5, and the fourth will be 0. These values represent the sample response to b_0 , which is essentially shifted by one time instant, then followed by the output of the pulse $p(t)$.

If you sample this received signal using the appropriately chosen matched filter, you would obtain the samples 1, 0.5, and -0.5. One important detail to highlight is that if you choose your matched filter pulse as a rectangular window over the interval $-T_s/2$ to $T_s/2$, and then sample the signal, the noise will be scaled by $2\sigma^2 \delta(k)$. Because of this careful choice of sampling interval, the noise remains independent and identically distributed (i.i.d.), even though I previously warned that it might not be i.i.d. This holds true because we are

sampling at intervals of T_s . Essentially, what's happening is that with $T = 2$ seconds, the channel is mixing across symbols. For instance, the effect of the transmitted symbol is felt after one second, but by sampling at exactly one-second intervals, you avoid the impact of noise correlation. This pulse structure is particularly effective.

(Refer Slide Time: 07:57)

Week 09: Lecture 44

Linear equalization

- Consider a block of five consecutive samples:

$$\mathbf{r}[k] = b[k-1] \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ 0 \\ 0 \\ 0 \end{bmatrix} + b[k] \begin{bmatrix} 0 \\ 1 \\ \frac{1}{2} \\ -\frac{1}{2} \\ 0 \end{bmatrix} + b[k+1] \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \frac{1}{2} \end{bmatrix} + \mathbf{w}[k] = \mathbf{U}\mathbf{b}[k] + \mathbf{w}[k]$$

where $\mathbf{w}[k]$ are Gaussian noise samples and
 $\mathbf{b}[k] = [b[k-1], b[k], b[k+1]]^T$.

6

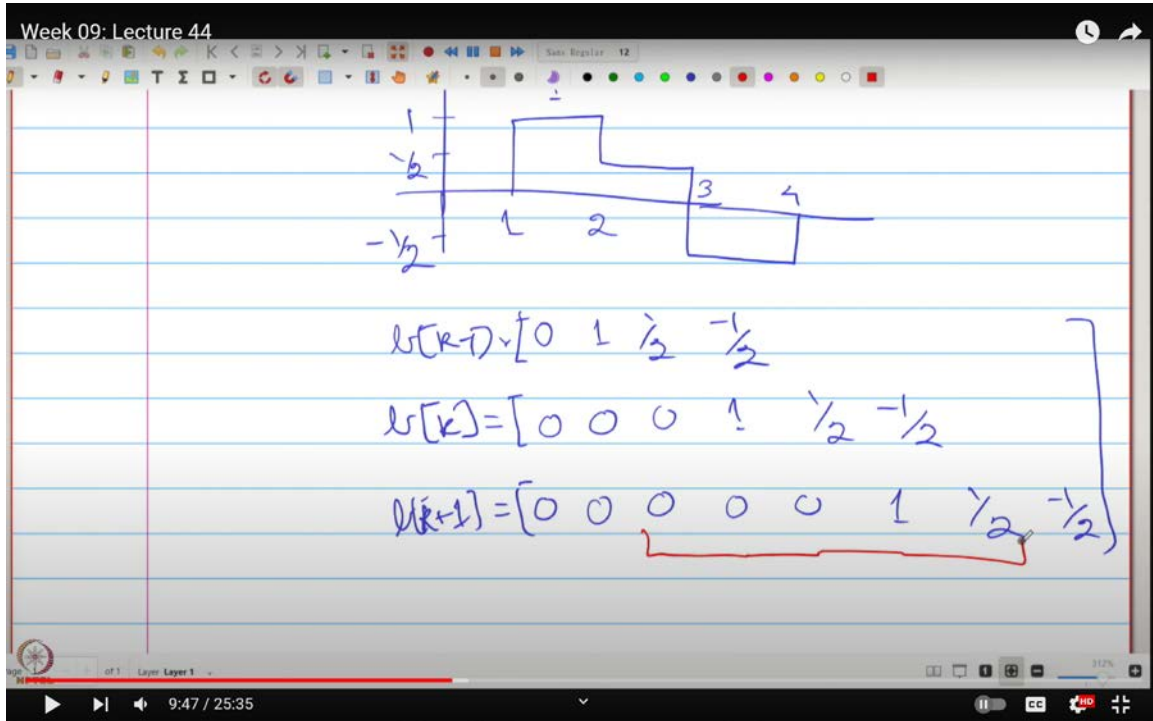
7:57 / 25:35

Now, let's consider a block of five consecutive samples to better understand how the system behaves. When you adopt this sampling strategy, the received signal, R_k , will be a combination of the contributions from neighboring symbols. For example, R_k will be influenced by b_{k-1} , with a response of $[0.5, -0.5, 0, 0, 0]$, and by b_k , with a response of $[0, 1, 0.5, -0.5, 0]$. So how do we get this? Let's step through it carefully.

First, recall that our channel response looks like this: $1, 0.5, -0.5$. These values occur at sample times $t = 1, 2, 3, \dots$, corresponding to the symbol duration. For instance, if we sample using a rectangular window with $T_s = 1$, the channel response would be $[0, 1, 0.5, -0.5]$.

Now, let's apply the principle of superposition. Suppose only b_{k-1} is active; then, the response we get is $[0.5, -0.5, 0, 0, 0]$. If only b_k is active, we would see $[0, 1, 0.5, -0.5, 0]$. Similarly, if only b_{k+1} is present, the response would be shifted by two time units, resulting in $[0, 0, 0, 1, 0.5, -0.5]$.

(Refer Slide Time: 09:47)



Next, let's analyze five consecutive symbols. When you consider these five symbols and express the output in matrix form, the received signal R_k will be represented as a combination of the contributions from each symbol. The contribution from b_{k-1} will be $[0.5, -0.5, 0, 0, 0]$, from b_k will be $[0, 1, 0.5, -0.5, 0]$, and from b_{k+1} will be $[0, 0, 0, 1, 0.5, -0.5]$. So, in total, the received signal R_k is a superposition of these three responses.

You might wonder why we are choosing these specific five symbols and not others. You are correct, there is no special reason for focusing on these five symbols in particular. We could have selected different symbols, and in fact, we will relax this assumption later. For now, we are simply using these five symbols to illustrate the process.

Let's revisit the process for a clearer understanding. If you want to verify the working step by step, we are essentially dealing with specific terms like $\frac{1}{2}$, $-\frac{1}{2}$, 0, 0, 0 and similar values. So, starting from this, the operations involve multiplying b_{k-1} , b_k , and b_{k+1} by specific matrices. For example, we have $b_{k-1} \times \left[\frac{1}{2}, -\frac{1}{2}, 0, 0, 0\right]$, $b_k \times \left[0, 1, \frac{1}{2}, -\frac{1}{2}, 0\right]$, and $b_{k+1} \times \left[0, 0, 0, 1, \frac{1}{2}\right]$, plus Gaussian noise contributions.

(Refer Slide Time: 11:04)

Handwritten equations from the video lecture:

$$r[k-1] = \begin{bmatrix} 0 & 1 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 \end{bmatrix}$$

$$r[k] = \begin{bmatrix} 0 & 0 & 0 & 1 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

$$r[k+1] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

$$r[k-1] \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ 0 \\ 0 \\ 0 \end{bmatrix} + r[k] \begin{bmatrix} 0 \\ \frac{1}{2} \\ -\frac{1}{2} \\ 0 \end{bmatrix} + r[k+1] \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \frac{1}{2} \end{bmatrix}$$

You can then express this system as $U \cdot b_k + W_k$, where U is a matrix formed by combining these columns, and W_k represents the Gaussian noise samples. Importantly, $W_k, W_{k+1}, W_{k+2}, \dots$, are independent and identically distributed (i.i.d.), owing to the properties of Gaussian noise. Now, we define a vector $b_k = [b_{k-1}, b_k, b_{k+1}]$, which allows us to write the system as R_k .

Thus, $R_k = [R_{k-1}, R_k, R_{k+1}, R_{k+2}, R_{k+3}]$, and by using matrix multiplication, we can express it in terms of the transmitted symbols as:

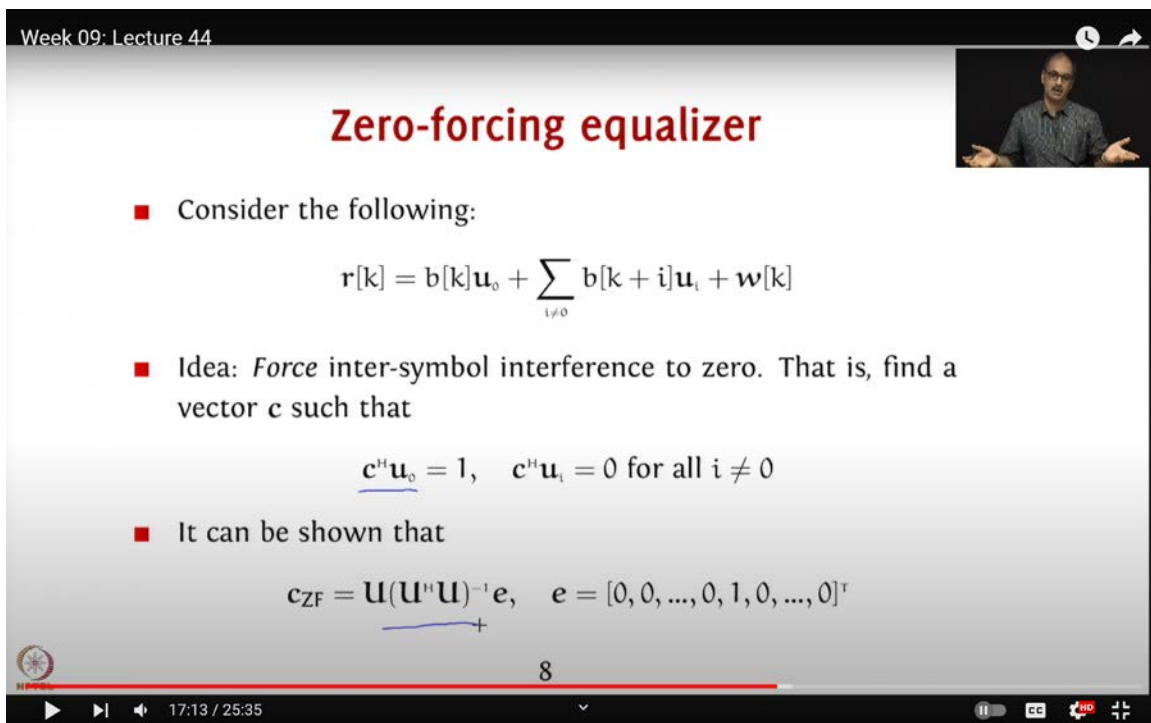
$$R_k = u \times [b_{k-1}, b_k, b_{k+1}] + W_k.$$

Where,

$$\mathbf{U} = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 1 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}$$

This expression describes how the transmitted symbols and noise are combined. Now, we define the zero-forcing equalizer. The goal here is to isolate b_k , eliminating the effects of intersymbol interference (ISI). We achieve this by finding a vector c such that $c \cdot U_1 = 0$, $c \cdot U_{-1} = 0$, and $c \cdot U_0 = 1$. This ensures that all interference from neighboring symbols is canceled out.

(Refer Slide Time: 17:13)



Week 09: Lecture 44

Zero-forcing equalizer

- Consider the following:
$$\mathbf{r}[k] = b[k]\mathbf{u}_0 + \sum_{i \neq 0} b[k+i]\mathbf{u}_i + \mathbf{w}[k]$$
- Idea: Force inter-symbol interference to zero. That is, find a vector c such that
$$\mathbf{c}^H \mathbf{u}_0 = 1, \quad \mathbf{c}^H \mathbf{u}_i = 0 \text{ for all } i \neq 0$$
- It can be shown that
$$\mathbf{c}_{ZF} = \mathbf{U}(\mathbf{U}^H \mathbf{U})^{-1} \mathbf{e}, \quad \mathbf{e} = [0, 0, \dots, 0, 1, 0, \dots, 0]^T$$

8

17:13 / 25:35

To implement this, the zero-forcing equalizer c satisfies the conditions:

$$c^\dagger U_0 = 1 \quad \text{and} \quad c^\dagger U_i = 0 \quad \text{for all } i \neq 0.$$

This is the essence of the zero-forcing equalizer, which forces the contribution of all symbols except b_k to zero. The mathematical expression for this zero-forcing equalizer is:

$$c_{ZF} = U^\dagger (U U^\dagger)^{-1} e,$$

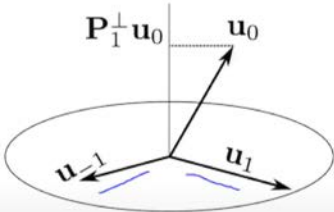
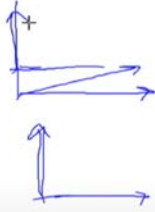
where e is a vector like $[0, 1, 0]$. This formulation aligns with the idea of projecting a vector onto a particular subspace while ensuring that interference from other subspaces is nullified.

(Refer Slide Time: 19:10)

Week 09: Lecture 44

Zero-forcing equalizer

- Geometrically, ZF equalizer projects received vector onto signal subspace

- In other words, $c = \alpha P^\perp u_0$, with $\alpha = 1 / \|P^\perp u_0\|^2$
- Smaller the orthogonal projection, larger the scale factor α ; can boost noise!

9

19:10 / 25:35

Why exactly is this form used? Intuitively, when you multiply U by its conjugate transpose, U^\dagger , the result is the identity matrix. The vector e effectively selects the part of the matrix corresponding to U_0 , ensuring that its contribution is scaled to 1 while eliminating others.

To verify this, consider the product:

$$c_{ZF}^\dagger U = U^\dagger (U U^\dagger)^{-1} e^\dagger U.$$

The key here is that the inverse matrix effectively removes the contributions from the other symbols, leaving us with:

$$c_{ZF}^\dagger U_0 = 1,$$

which confirms that only the desired symbol, b_k , remains.

One caveat, however, is that zero-forcing equalizers are not always feasible. For instance, there are certain linear independence conditions on the columns of U that must be satisfied for the zero-forcing approach to work. Whenever these conditions are met, the equalizer can successfully retrieve b_k while eliminating b_{k-1} and b_{k+1} .

(Refer Slide Time: 20:10)

The screenshot shows a video player interface for a lecture titled "Week 09: Lecture 44". The main content is a slide with the title "Zero-forcing equalizer" in red. It contains three bullet points:

- Noise variance per dimension (with ZF equalizer):

$$v_{ZF}^2 = \sigma^2 \|c_{ZF}\|^2 = \sigma^2 \alpha^2 \|P^\perp u_0\|^2 = \frac{\sigma^2}{\|P^\perp u_0\|^2}$$
- Corresponding noise variance without ISI:

$$v_{MF}^2 = \frac{\sigma^2}{\|u_0\|^2}$$
- Thus, noise enhancement due to ZF equalization is:

$$\frac{v_{ZF}^2}{v_{MF}^2} = \frac{\|u_0\|^2}{\|P^\perp u_0\|^2} + 1$$

The slide also features a small video inset of the lecturer in the top right corner and a progress bar at the bottom showing 20:10 / 25:35.

However, there's a geometric aspect to consider. Think of the columns U_{-1} and U_1 as vectors in a vector space. The zero-forcing equalizer is essentially projecting onto a space orthogonal to these vectors, ensuring that their contribution is nullified. Once this projection is completed, we scale U_0 to ensure it has a magnitude of 1. This scaling is critical because it can amplify the noise, which is a potential drawback.

For example, if the columns U_{-1} and U_1 are nearly orthogonal, the projection process is straightforward. But if the vector spaces are mixed up, the projection can lead to significant noise enhancement, a problem we'll observe in our experiments with GNU Radio.

The noise variance per dimension when using the zero-forcing equalizer is given by:

$$\sigma_{ZF}^2 = \sigma^2 \cdot |c_{ZF}|^2,$$

where c_{ZF} multiplies the noise. This results in:

$$\sigma_{ZF}^2 = \frac{\sigma^2}{|P_{\perp} U_0|^2},$$

where P_{\perp} denotes the projection matrix perpendicular to the interference space. In contrast, if there were no ISI, the noise variance would simply be:

$$\sigma^2 / |U_0|^2.$$

(Refer Slide Time: 21:47)

The screenshot shows a video player interface for a lecture. The title bar reads 'Week 09: Lecture 44'. The main slide content is as follows:

Zero-forcing equalizer

- For the running example $c_{ZF} = [5/8, 5/8, 5/8, -1/8, 1/4]^T$
- We note that noise is complex normal, variance $2.5\sigma^2$. So, for BPSK, only real-part matters, with variance $1.25\sigma^2$, $E_b = 1.5$.
- Symbol error rate is $Q\left(\sqrt{\frac{\alpha E_b}{N_0}}\right)$, with $\alpha = 16/15$ for ZF, and $\alpha = 2$ for ISI-free BPSK. Loss in SNR terms: 2.73 dB

$10 \log_{10}\left(\frac{15}{8}\right)$

The video player shows a progress bar at 21:47 / 25:35 and a slide number '11' at the bottom center.

Thus, the noise enhancement factor caused by zero-forcing is:

$$\text{Enhancement Factor} = \frac{|U_0|^2}{|P_{\perp}U_0|^2}.$$

This projection matrix, P_{\perp} , can introduce significant noise amplification if the vectors are not well-separated, causing performance degradation. In our running example, you can check that the zero-forcing coefficients (CZF) are:

$$\text{CZF} = \left[\frac{5}{8}, \frac{5}{8}, \frac{5}{8}, -\frac{1}{8}, \frac{1}{4} \right]^T.$$

This outlines how the zero-forcing equalizer works and highlights the challenges related to noise enhancement.

Now, you can observe that these values are all less than one, but they aren't particularly small. As we discussed earlier, the noise is modeled as complex normal with a variance of $2.5\sigma^2$. However, for BPSK, only the real part of the noise matters since we adopted a complex model. In this case, the real noise variance becomes $1.25\sigma^2$.

Regarding the energy per bit, E_b is 1.5. Why? Because the pulse itself has amplitudes of $\frac{1}{2}$ and $-\frac{1}{2}$. If you sum up the energies, you get:

$$1 + \left(\frac{1}{4}\right) + \left(\frac{1}{4}\right) = \frac{3}{2}.$$

The symbol error rate, assuming no intersymbol interference, follows the well-known expression $Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$ for BPSK. However, in this context, the factor A comes into play, and when you compute it, A turns out to be $\frac{16}{15}$, which is very close to one.

As a result, the performance loss, in terms of signal-to-noise ratio (SNR), is approximately $10 \log_{10}\left(\frac{15}{8}\right)$. This comes out to roughly 2.73 dB. Now, 2.73 dB may not sound significant at first glance, but it's actually quite a lot, nearly 3 dB, which represents a doubling of power. This loss is what you're experiencing due to noise enhancement caused by using the zero-forcing equalizer.

These types of situations arise especially when you have either a low SNR or a significant amount of intersymbol interference. This is important to keep in mind as you work through equalization strategies.

(Refer Slide Time: 23:11)

The screenshot shows a video player interface for a lecture. The title bar at the top left reads "Week 09: Lecture 44". The main content area has a red heading "Zero-forcing equalizer". Below the heading are two bullet points: "Block based approach: can be extended to large block lengths" and "Can we have a 'filtering' based approach? In our example, $T = 2$, $T_s = 1$, so we have the following channels:". Below the text, the channel transfer functions are given as $H_1(z) = 1 - \frac{1}{2}z^{-1}$ and $H_2(z) = \frac{1}{2}$. A blue underline is drawn under the text "we have the following channels:". Below the equations, the text "We need:" is followed by the equation $\sum_{i=1}^m H_i(z)G_i(z) = z^{-d}$. A blue underline is drawn under the summation symbol. At the bottom of the slide, the number "12" is centered. The video player controls at the bottom show a progress bar at 23:11 / 25:35.

One issue I previously mentioned was that we used a block-based approach with a block length of five. Now, the question arises: Can we extend this to larger block lengths? Of course, with larger blocks, you'd be dealing with bigger matrices, but can we take a more continuous filtering-based approach? After all, our channel behaves like a filter. So, could we just use filtering instead?

In this particular example, you can verify that we have $T = 2$ and $T_s = 1$. This allows us to represent the channel as:

$$H_1(z) = 1 - \frac{1}{2}z^{-1}, \quad H_2(z) = \frac{1}{2}.$$

Why these forms? If you recall, our pulse $p(t)$ was $\frac{1}{2}, -\frac{1}{2}$. Since we're sampling at twice the rate, the effective split between these two filters follows that pattern. The first filter captures the alternating 1 and $-\frac{1}{2}$ values, while the second filter picks up the residual $\frac{1}{2}$.

To achieve proper equalization, we need to satisfy the condition,

$$H_1(z) \times G_1(z) + H_2(z) \times G_2(z) = z^{-1}.$$

This means that the overall effect of the filtering should ideally be just a delay, delays are acceptable to a certain extent, but you don't want any additional filtering effects.

(Refer Slide Time: 24:13)

Week 09: Lecture 44

Zero-forcing equalizer

- Recall $\mathbf{c}_{ZF} = [5/8, 5/8, 5/8, -1/8, 2/8]^T$. This translates to:

$$G_1(z) = \frac{1}{8}(-1 + 5z^{-1}), \quad G_2(z) = \frac{1}{8}(2 + 5z^{-1} + 5z^{-2})$$
- Result:

$$H_1(z)G_1(z) + H_2(z)G_2(z) = z^{-1}$$
- For fractionally spaced equalizers, sufficient condition: $\{H_i(z)\}$ should not have common zeros.
- For symbol spaced: $1/H(z)$ is the only equalizer

13

24:13 / 25:35

If you can find $G_i(z)$ that satisfy this condition, you can very easily construct a filter-based equalizer. Now, let's see if that's possible. Our zero-forcing equalizer (ZF) has a specific structure. If you reverse the elements, you'll find terms like:

$$\frac{2}{8} + \frac{5}{8}z^{-1} + \frac{5}{8}z^{-2},$$

which corresponds to one filter, and:

$$-\frac{1}{8} + \frac{5}{8}z^{-1},$$

which corresponds to another filter.

You can confirm that $H_1(z) \times G_1(z) + H_2(z) \times G_2(z) = z^{-1}$ holds true. For fractional-phase space equalizers, there's a sufficient condition: the filters $H_i(z)$ should not share common zeros. If they don't have common zeros, you can always find $G_1(z)$ and $G_2(z)$ that give you proper equalization. This approach isn't necessarily better or worse than the block-based method, it's just a continuous implementation of the same concept.

However, for symbol-space equalizers, where the sample rate equals the symbol rate, your only equalizer option is $1/H(z)$. That can be problematic. So, while the zero-forcing equalizer is a possible solution, keep in mind that it tends to work well only under conditions of high SNR or when the vector spaces are clearly separable.

Another significant issue is noise enhancement. The Gaussian noise gets amplified significantly by the zero-forcing approach. In the next class, we will introduce the minimum mean square error (MMSE) equalizer, which takes noise into account and mitigates these problems. The MMSE equalizer provides a more balanced performance, even in high noise scenarios. Thank you for your attention, and we will continue with this discussion in the next session.