Digital Communication using GNU Radio

Prof. Kumar Appaiah

Department of Electrical Engineering

Indian Institute of Technology Bombay

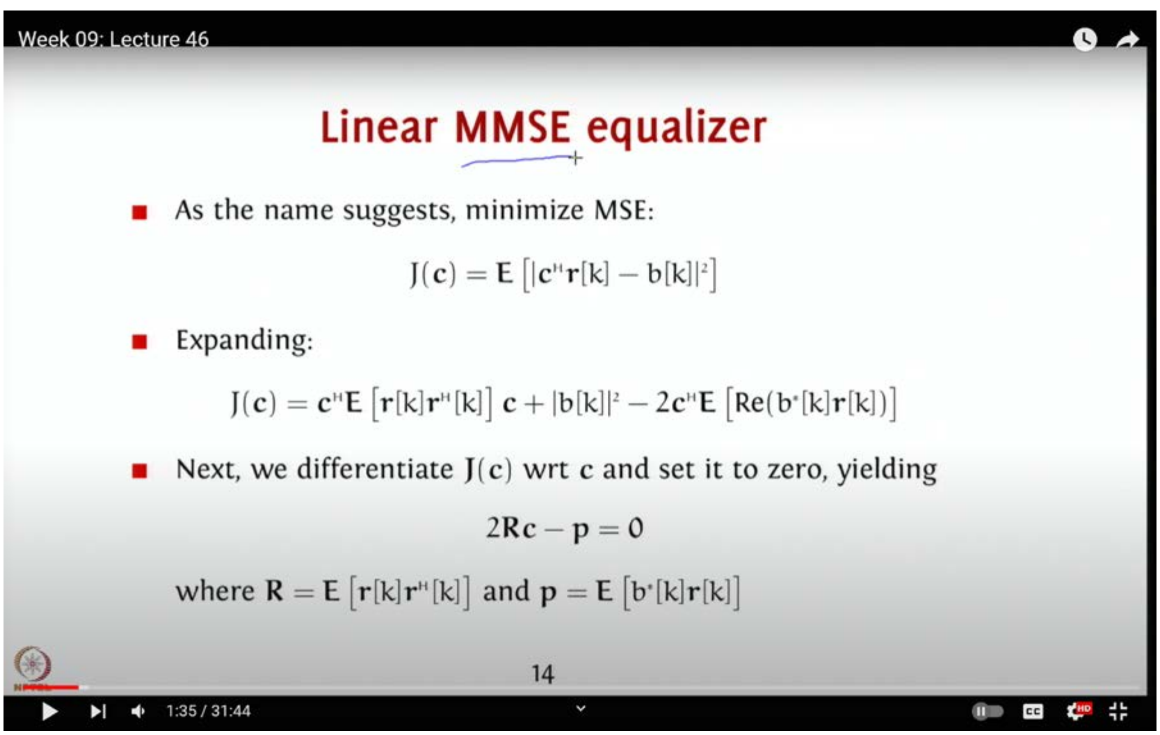
Week-09

Lecture-46

Suboptimal Channel Equalisation: Linear Minimum Mean-Square Error

Hello and welcome to this lecture on Digital Communication using GNU Radio. My name is Kumar Appiah, and I am from the Department of Electrical Engineering at the Indian Institute of Technology, Bombay. In this session, we will continue our exploration of suboptimal equalization techniques. Over the past few lectures, we have focused on zero-forcing equalization. In zero-forcing, the primary goal is to design an equalization filter that completely eliminates intersymbol interference (ISI) by forcing the interference to zero.

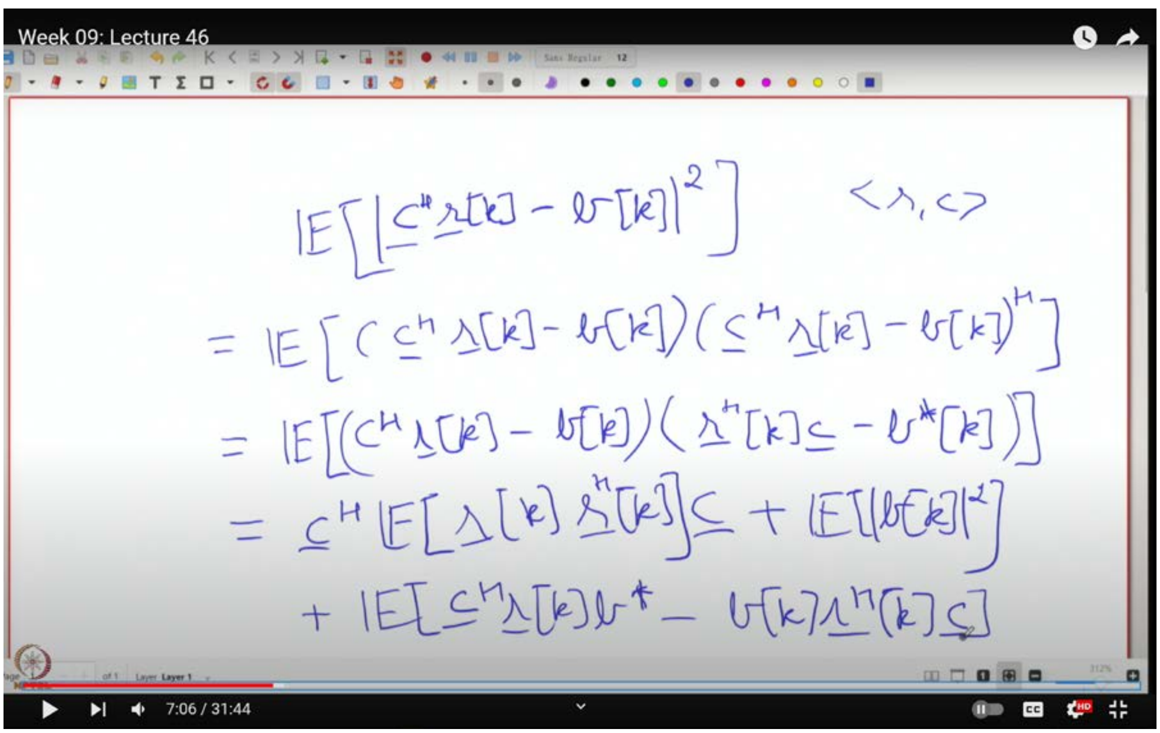
(Refer Slide Time: 01:35)



However, as we observed both in our theoretical discussions and through GNU Radio simulations, zero-forcing comes with a significant drawback: it can lead to substantial noise enhancement. This noise enhancement is particularly detrimental when operating in environments with low signal-to-noise ratios (SNR). Consequently, the zero-forcing equalizer may not be the best choice in scenarios where the SNR is very low.

In today's lecture, we will explore another suboptimal equalization technique known as the Linear Minimum Mean Squared Error (MMSE) equalizer. Here, MMSE stands for Minimum Mean Squared Error. This approach takes a statistical perspective, aiming to minimize the expected squared error between the detected symbol and the transmitted symbol. This minimization essentially defines our equalizer.

(Refer Slide Time: 07:06)



Similar to other methods that involve minimizing squared error, the use of squared error in MMSE allows for a straightforward computation of the equalizer. By differentiating the squared error with respect to the filter coefficients and setting the result to zero, we can derive the filter coefficients. This is precisely what we will examine in this lecture.

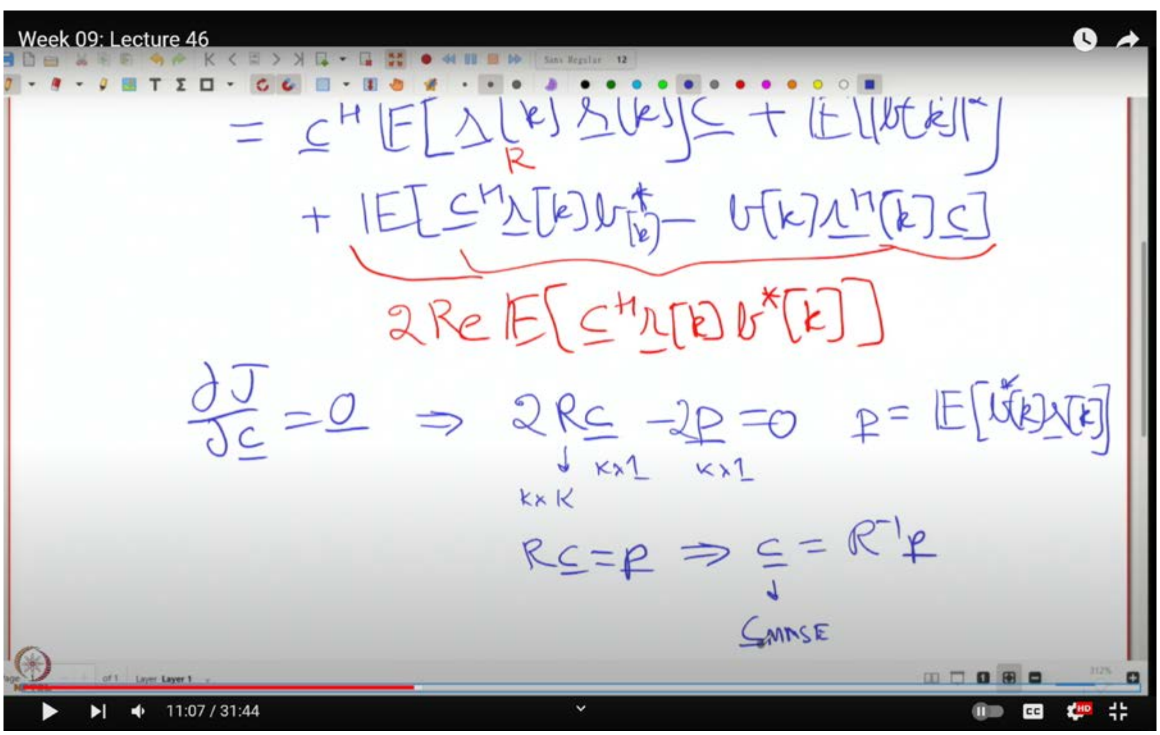
As the name suggests, our objective is to minimize the mean squared error. We can think of this error as our objective function, denoted by J. The function J represents the mean squared error and is a function of the filter coefficient vector c. The vector c contains the filter coefficients, much like in previous discussions.

To define the error, we consider the expectation $E[|c^H r_k - b_k|^2]$, where c^H represents the Hermitian (or conjugate transpose) of c, r_k is the received symbol vector, and b_k is the transmitted symbol, which is typically a complex number such as a QPSK symbol. Here, both c and r_k are vectors, so I use the underscore notation to represent them.

When we expand the expression for J(c), we obtain a specific form that we will delve into further in this lecture. For clarity, let's start with the expression $c^H r_k - b_k$.

As mentioned earlier, b_k is a symbol, a complex number like a QPSK symbol, whereas c and r_k are vectors. Thus, you might sometimes see $c^H r_k$ represented as $r_k^T c$, which is another notation that may be used in different contexts.

(Refer Slide Time: 11:07)



When dealing with the expression $c^H r^2$, one effective approach to handle these types of expectations is to express the modulus square as $c^H r$ multiplied by its Hermitian counterpart. Remember that this is a scalar quantity, but we will still use the Hermitian notation, which is essentially the same as the conjugate in this context. So, this expression can be written as:

$$E[(c^H r_k - b_k)(c^H r_k - b_k)^H]$$

I am deliberately using the Hermitian operator here because it allows us to conveniently swap the order of r and r_k, facilitating a more straightforward representation. Expanding this, we have:

$$E[c^H r_k - b_k] = E[r_k^H c - b_k^*]$$

Here, since b_k is a scalar, its Hermitian is simply its complex conjugate, denoted as b_k^*. Applying the Hermitian operator to $c^H r_k$ is equivalent to taking the conjugate, so we can write it as $r_k^H c$. You can verify that this is indeed correct.

Expanding further, we get:

$$c^H E[r_k r_k^H] c + E[b_k b_k^*]$$

Notice that I can take c outside the expectation because c is a fixed vector, not dependent on the specific values of r or b. It is a constant vector determined by the filter design, not by the individual random variables.

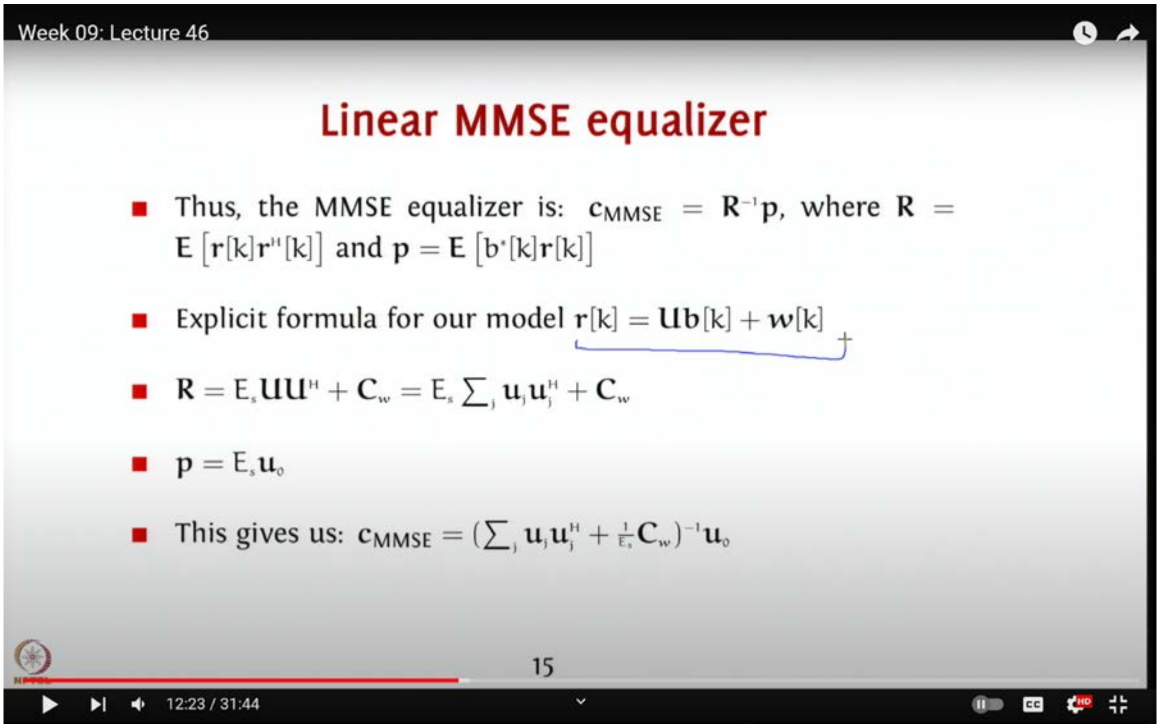
Now, when expanding the terms, I separate the cross terms:

$$E[c^H r_k b_k^* - b_k r_k^H c]$$

We have now introduced the cross term $c^H r_k b_k^* - b_k r_k^H c$. If you look closely, this term is exactly what we've written before, but it's often convenient to express it as twice the real part:

$$2\mathrm{Re}([c^H r_k b_k^*])$$

(Refer Slide Time: 12:23)



This expression is useful because it simplifies the differentiation process. You can treat the real part of $c^H r_k b_k^*$ as $r_k^H c$, and it allows us to move the expectation operator outside, which simplifies our calculations.

Now, focusing on our objective function J, we need to differentiate it with respect to c. By setting the derivative $\frac{\partial J}{\partial c} = 0$, we essentially differentiate each entry of c individually, its real and imaginary parts, and set them to zero. This approach is well-documented in linear algebra or matrix calculus references.

Upon differentiating, we obtain:

$$c^H E[r_k r_k^H]c - 2E[b_k^* r_k] = 0$$

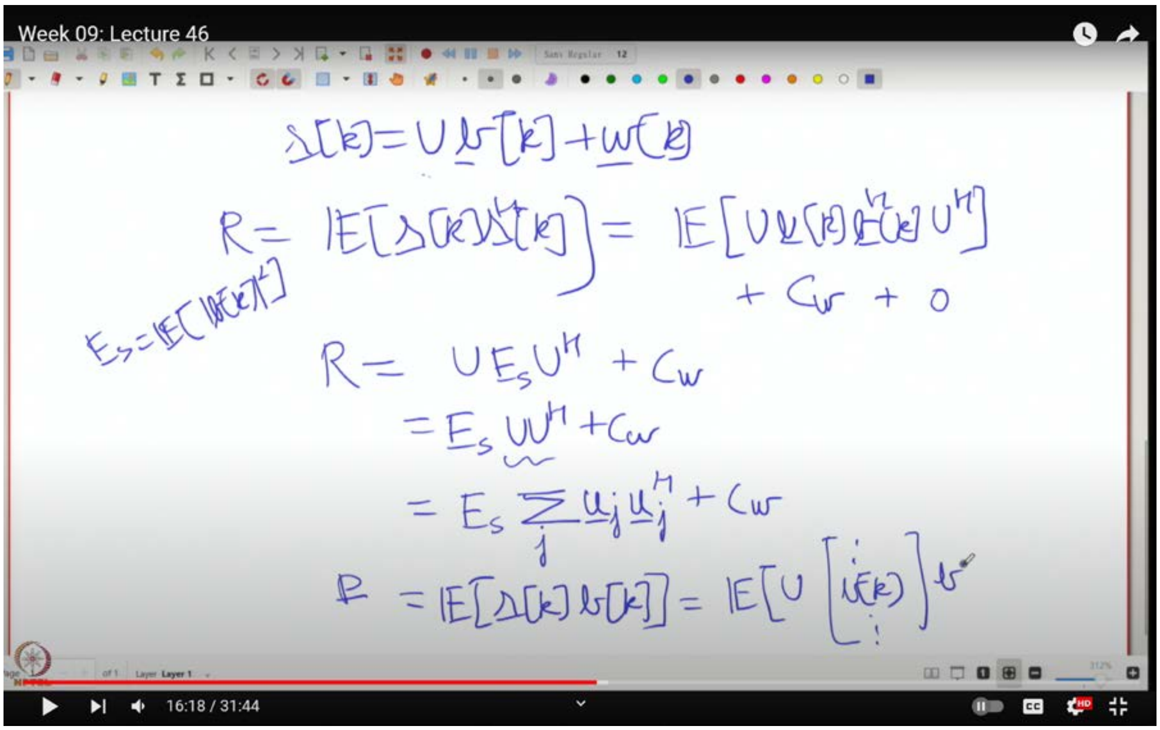
Here, p represents the expectation $E[b_k^* r_k]$. By expanding each entry of c into its real and imaginary parts, performing the differentiation with respect to these parts, and setting the resulting expressions to zero, you can derive this form:

$$p = E[b_k^* r_k]$$

This gives us a clear path to computing the filter coefficients based on the MMSE criterion.

Let's break this down for better clarity and understanding.

(Refer Slide Time: 16:18)



Consider a scenario where you have a matrix R that is $k \times k$, and vectors c and p that are $k \times 1$. To solve the equation Rc = p, if the matrix R is invertible, then c can be obtained by computing $c = R^{-1}p$. This type of equation is quite common in communication and signal processing applications.

The matrix R often exhibits certain properties that simplify its evaluation. For example, the entries of R may follow a particular pattern, like being of a Toeplitz form, which is a structured matrix with constant diagonals. When R is invertible, the solution to c is given by $c = R^{-1}p$.

To ensure that this solution represents a minimum, you can evaluate the second derivative of the objective function J with respect to c. The resulting values will confirm that c indeed corresponds to a global minimum, meaning that this solution is optimal across all possible values of c.

This optimal c is referred to as c_{MMSE}. While you can select various filters, c_{MMSE} is the specific filter that minimizes the Minimum Mean Squared Error (MMSE), which is our objective function J.

Now, let's revisit the expression $c^H Rc + |b_k|^2$. Here, we note that the expectation of $|b_k|^2$ is denoted by E_s, which represents the signal energy. Since E_s does not depend on the realization, its derivative with respect to c is zero. So, when you differentiate the equation $J = c^H Rc + |b_k|^2$ with respect to c, you arrive at:

$$2Rc - 2p = 0$$

Here, $R = E[rr^H]$ and $p = E[b_k^* r_k^H]$, where p is known as the cross-correlation matrix. Therefore, the solution for the MMSE equalizer is:

$$c_{\text{MMSE}} = R^{-1} p$$

where $R = E[rr^H]$ and $p = E[b_k^* r_k]$.

If you recall from our previous discussions, we had a structure in which we grouped several symbols together, such as grouping five symbols in our running example. We expressed r_k as:

$$r_k = ub_k + w_k$$

Here, b is a column vector containing the symbols b_{k-1}, b_k, and b_{k+1}. Let's evaluate the matrix $R = E[r_k r_k^H]$. Since we are assuming that the processes are wide-sense stationary, the matrix R simplifies to:

$$R = E[ub_k b_k^H u^H] + E[w_k w_k^H]$$

Since the cross-terms involving w_k and b_k are zero due to their uncorrelated nature, the matrix simplifies to:

$$R = uE_s u^H + C_w$$

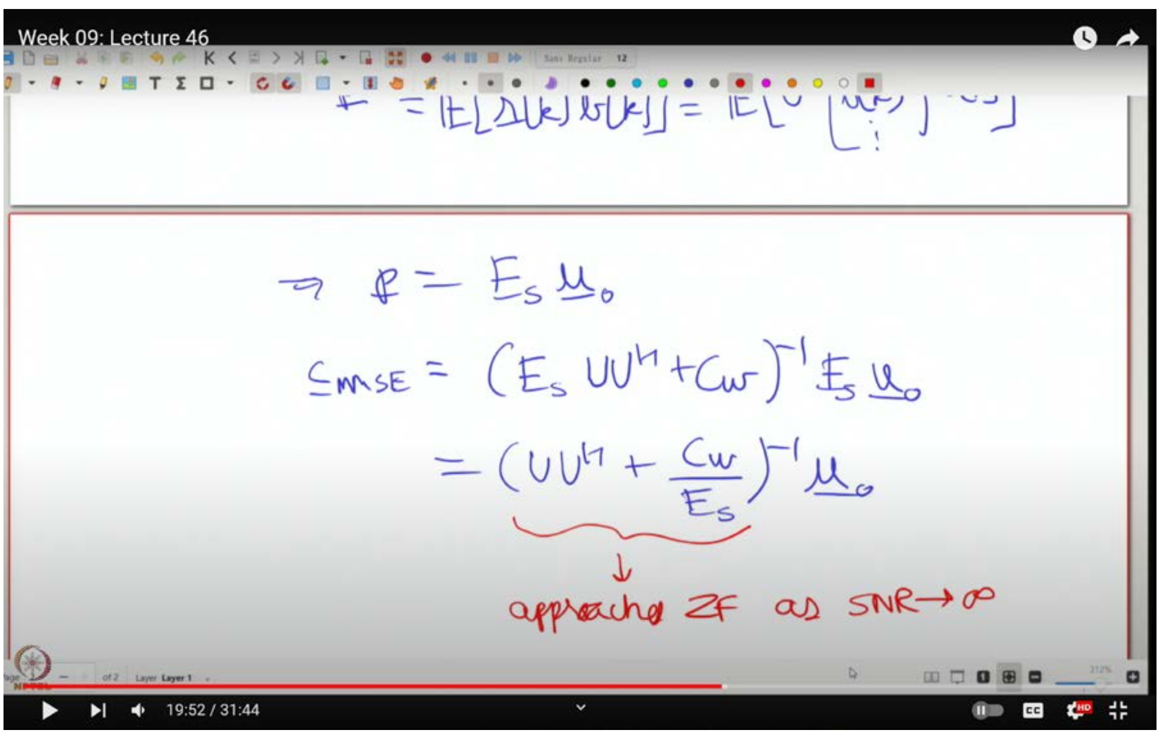
where $E_s = E[|b_k|^2]$ is the signal energy, and C_w is the noise covariance matrix.

To write this expression in an expanded form, note that uu^H can be expressed as the sum of the outer products of its columns. This expansion is straightforward and aligns with what we have discussed so far.

Now, considering the vector p, recall that r_k was expressed as $ub_k + w_k$. The vector b contains three entries, and only one of these corresponds to b_k. Therefore, when substituting r_k into p = $E[b_k^* r_k^H]$, the term involving w_k vanishes due to the uncorrelation between w_k and b_k.

Let's go through this concept step by step, ensuring both clarity and a deeper understanding.

(Refer Slide Time: 19:52)



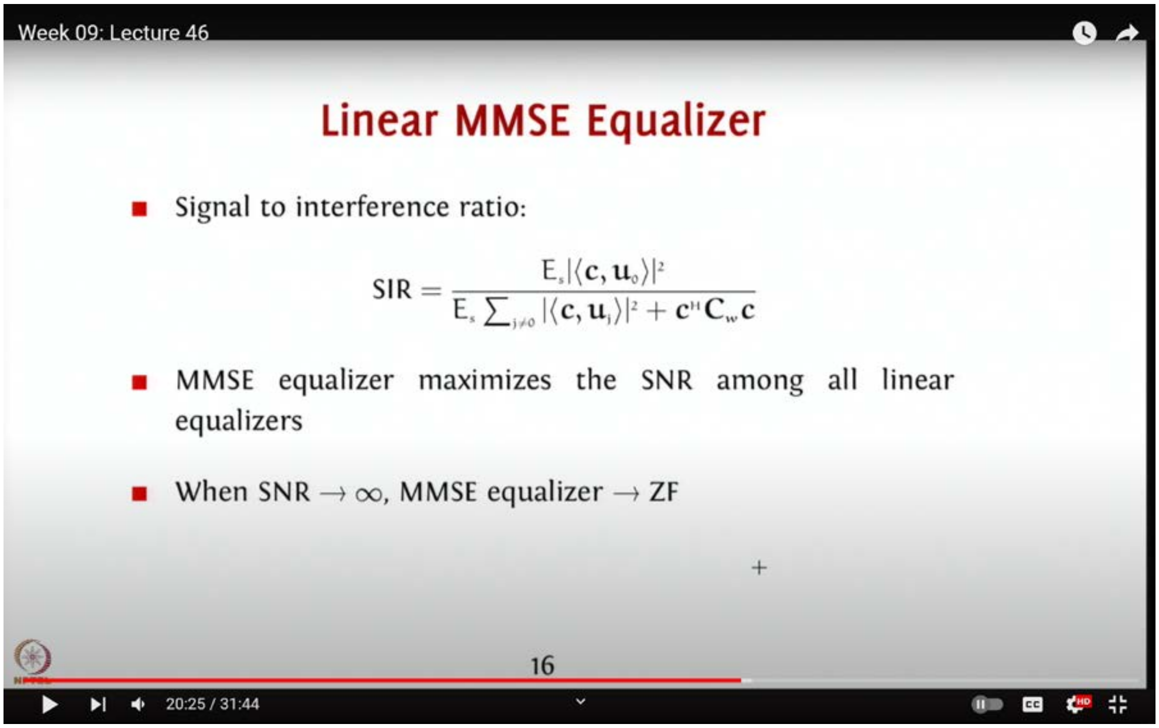
When we consider the expression ub_k, only the term involving b_k remains significant, with all other terms effectively going to zero. This is because p will only select the column of u

corresponding to b_k. We can denote this specific column as u_0. To visualize this, imagine that each column of u corresponds to a different entry of the vector b, which includes terms like b_{k-2}, b_{k-1}, b_k, b_{k+1}, and b_{k+2}. The only column that matters for p is the one associated with b_k, which is precisely what we're observing.

Now, let's build on this and derive the Minimum Mean Square Error (MMSE) solution, c_{MMSE}. Recall that $c_{\text{MMSE}} = R^{-1}p$. Here's a neat trick: since E_s (the signal energy) is just a scalar, we can introduce it into the expression by multiplying through by $\frac{1}{E_s}$. This allows us to rewrite the equation as:

$$c_{\text{MMSE}} = \left(uu^H + \frac{C_w}{E_s}\right)^{-1} u_0$$

(Refer Slide Time: 20:25)



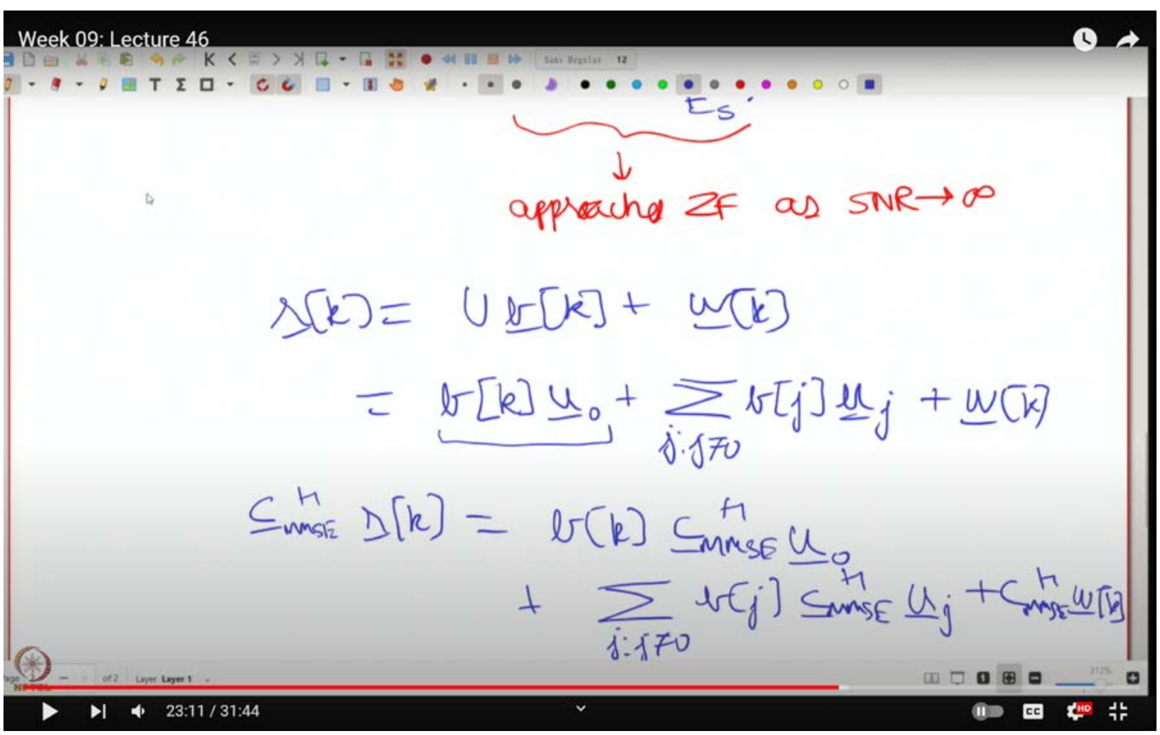
If C_w, the noise covariance matrix, is equal to $\frac{N_0}{2} \times I$, this expression takes on a particularly elegant form. Here's where things get interesting: the fundamental difference between Zero

Forcing (ZF) and MMSE is that the $\frac{1}{E_s}$ term in the MMSE solution essentially weights the equalization filter based on the noise level.

Let's engage in a thought experiment to understand this better. If the entries of C_w, which represent noise, have variances much larger than E_s, then the matrix $\frac{C_w}{E_s}$ will, upon inversion, yield values close to zero. This implies that in a low Signal-to-Noise Ratio (SNR) scenario, the MMSE equalizer might almost entirely suppress the signal, leading to no useful information being recovered.

On the other hand, in a high SNR regime, where the noise is minimal, C_w becomes nearly zero. In such cases, the MMSE solution begins to resemble the Zero Forcing approach. As the SNR approaches infinity, the MMSE equalizer essentially becomes equivalent to the ZF equalizer. This phenomenon will become clearer when we simulate it in GNU Radio. The key takeaway is that the MMSE equalizer offers a sort of "slider" that intuitively balances the weight given to the received signal, depending on the noise level.

(Refer Slide Time: 23:11)



In high SNR conditions, the MMSE equalizer behaves like a ZF equalizer, whereas in low SNR conditions, it avoids blindly inverting the channel, which would otherwise amplify the noise. Instead, it strikes a trade-off to achieve the minimum mean squared error, which, we hope, results in better performance.

Now, let's consider why the MMSE equalizer might be preferable. The MMSE equalizer maximizes the Signal-to-Interference Ratio (SIR) for linear equalizers. To explore this, we can break down the received signal r_k:

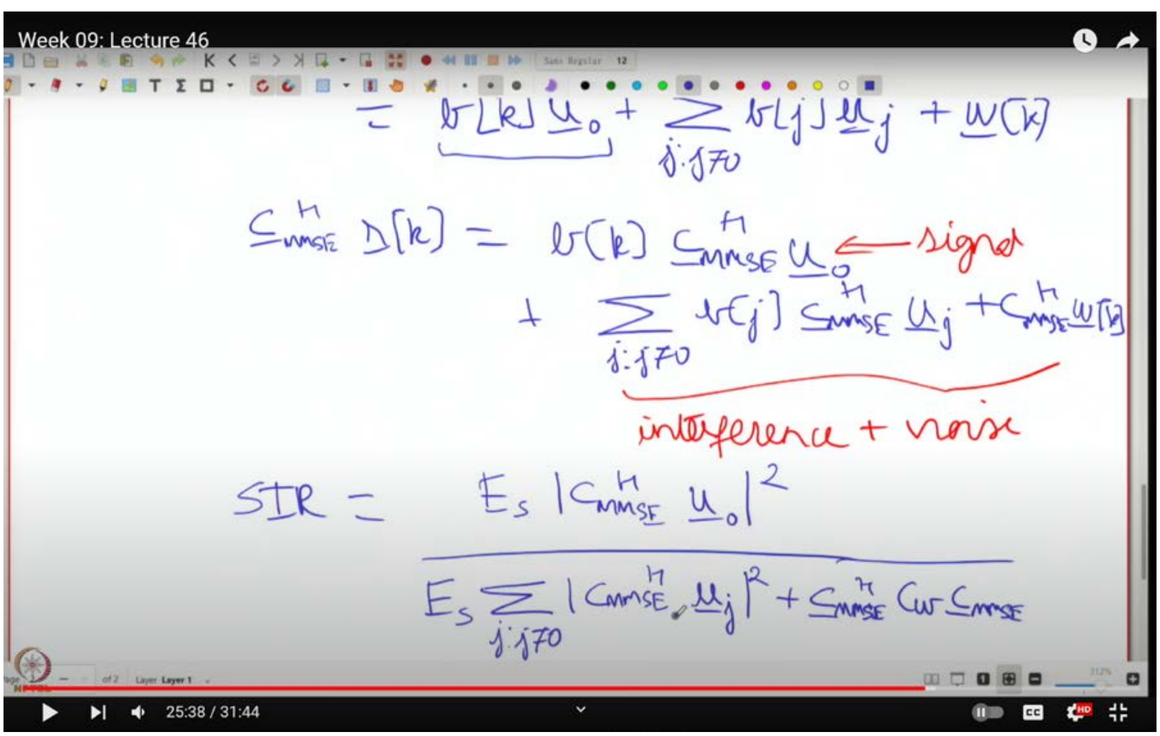
$$r_k = ub_k + w_k$$

This can be decomposed into two parts:

$$r_k = b_k u_0 + \sum_{j \neq 0} b_j u_j + w_k$$

Here, the first term corresponds to the desired signal power, while the remaining terms represent interference from neighboring symbols and noise.

(Refer Slide Time: 25:38)



When we compute $c_{\text{MMSE}}^{H} r_k$, we get:

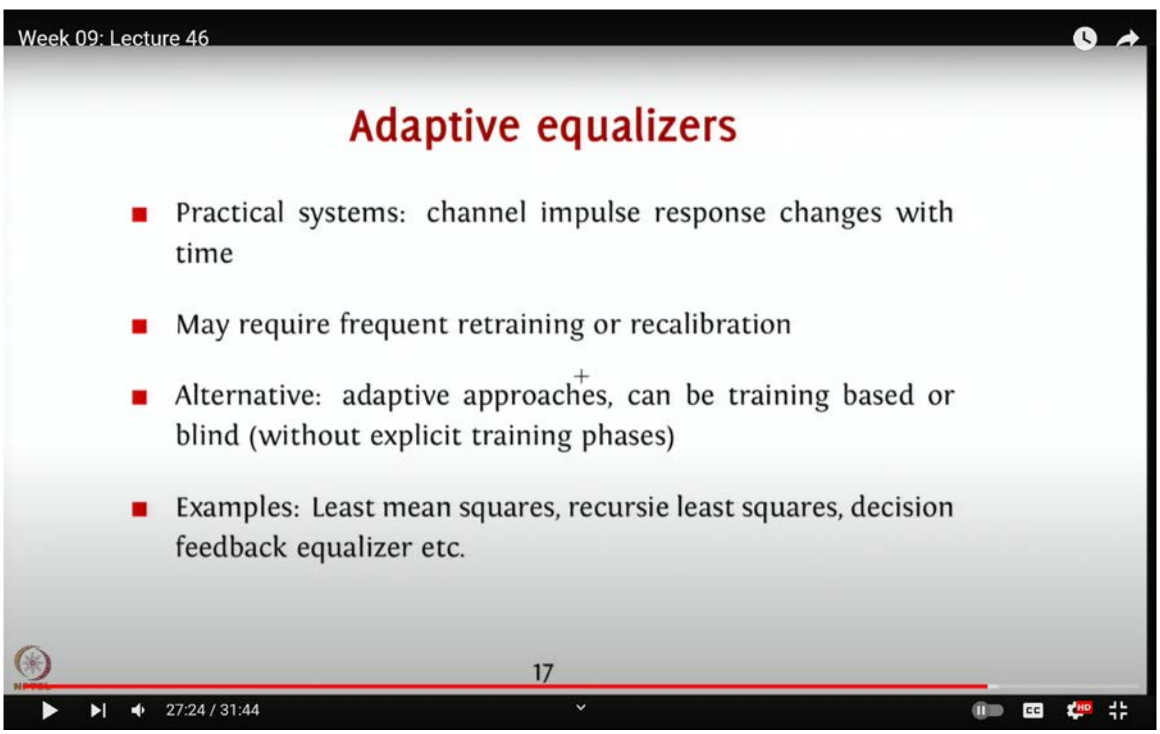
$$c_{\text{MMSE}}^{H} r_k = b_k c_{\text{MMSE}}^{H} u_0 + \sum_{j \neq 0} b_j c_{\text{MMSE}}^{H} u_j + c_{\text{MMSE}}^{H} w_k$$

Here, the first part corresponds to the desired signal, while the second part represents interference from other symbols, and the third part is due to noise. To compute the SIR, we take the ratio of the power of the desired signal to the power of the interference and noise. The expression for SIR becomes:

$$\text{SIR} = \frac{E_s \left| c_{\text{MMSE}}^{H} u_0 \right|^2}{E_s \sum_{j \neq 0} \left| c_{\text{MMSE}}^{H} u_j \right|^2 + c_{\text{MMSE}}^{H} C_w c_{\text{MMSE}}}$$

If C_w (the noise covariance matrix) approaches zero, the SIR simplifies to the ratio of the desired signal power to the interference power. In the case of ZF equalization, the filter c is chosen such that the interference terms are zero, meaning $c^H u_j = 0$ for all $j \neq 0$, aligning the filter with u_0 and ignoring the others.

(Refer Slide Time: 27:24)



However, in MMSE equalization, while the interference terms are not exactly zero, the filter also takes into account the noise, achieving a better balance between minimizing interference and controlling noise amplification. This is how MMSE improves the overall SIR, providing an advantage over ZF, especially in scenarios with varying SNR levels.

Now, let's discuss the MMSE equalizer in more detail. The key idea here is that the signal-to-interference ratio (SIR) is maximized when you choose C as C_{MMSE}. In fact, among all linear equalizers, it can be proven that C_{MMSE} is the optimal choice for maximizing the SIR. Although I won't go into the detailed proof here, the essential point is that minimizing the mean squared error is equivalent to maximizing the SIR.

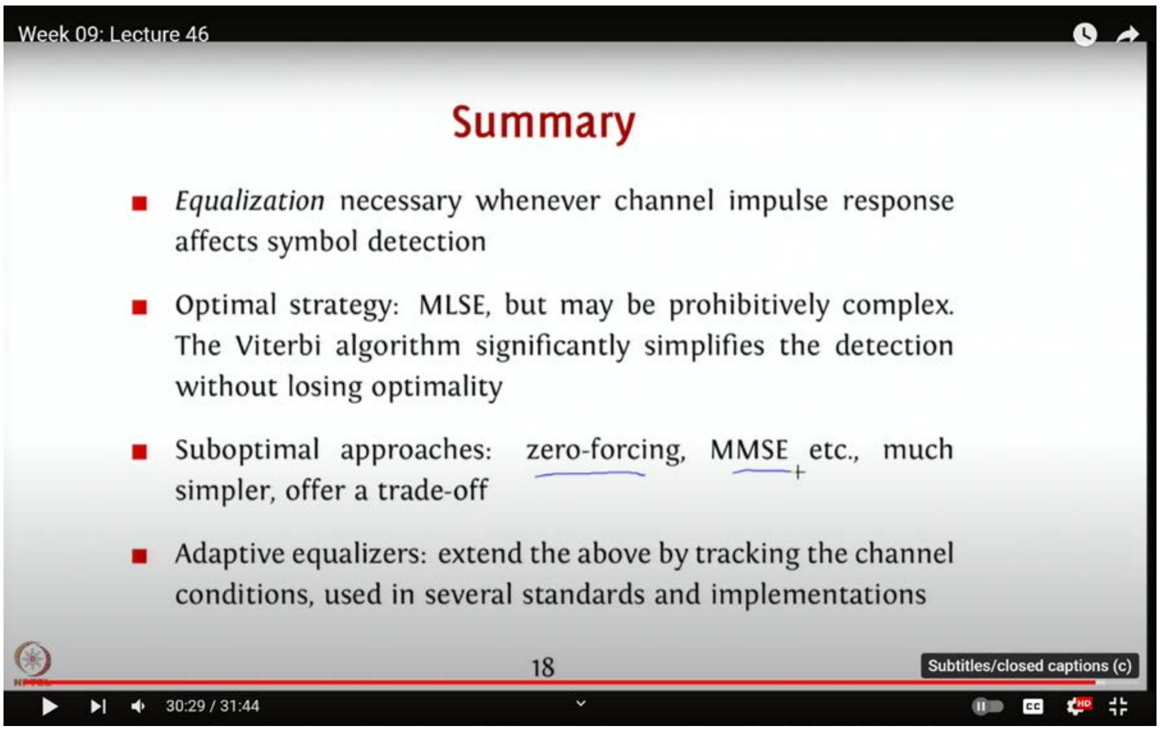
Another important point is that as the Signal-to-Noise Ratio (SNR) approaches infinity, the MMSE equalizer effectively becomes the Zero Forcing (ZF) equalizer. This makes intuitive sense: if the term $C^{\dagger}C_{w}C$ (where $C^{\dagger}$ represents the Hermitian transpose of C) approaches zero, you can select C such that it orthogonalizes with respect to the interfering signals and only has an inner product with the desired signal u_0. In such a scenario, the SIR becomes extremely large, approaching infinity. While this is an intuitive understanding, we will also demonstrate this numerically, and we'll explore it further in the next lecture.

An interesting extension of these concepts is the idea of adaptive equalization. In practical systems, the channel impulse response or frequency response varies over time. For instance, if you're on a call while moving around, the environment around you changes, causing the channel characteristics observed by your phone to change as well. This means that the equalizer might need frequent retraining or recalibration.

Rather than retraining the system every time the channel changes, an alternative approach is to track the channel dynamically. This adaptive method involves learning the changes in the channel gradually, rather than relearning it from scratch. For example, as you walk around during a call, your phone can learn and adapt to the evolving channel conditions in real-time. This method is known as adaptive equalization.

There are two primary types of adaptive equalization: training-based and blind. In training-based adaptive equalization, the transmitter (e.g., the base station) sends specific information to the receiver, allowing it to track changes in the channel. On the other hand, blind equalization requires the receiver to adapt without explicit training data, it must infer the channel characteristics from the received signal alone, without prior knowledge of the exact data sequence or modulation scheme.

(Refer Slide Time: 30:29)



In practical systems, both training-based and blind approaches are often used in combination. Some common algorithms for adaptive equalization include the Least Mean Squares (LMS) algorithm, the Recursive Least Squares (RLS) algorithm, and the Decision Feedback Equalizer (DFE). These methods build upon the equalization techniques we've discussed and allow for continuous adaptation to the channel, thereby enhancing performance without the need for constant retraining.

To summarize, equalization is essential whenever the channel impulse response affects symbol detection. The optimal strategy would be to use Maximum Likelihood Sequence

Detection (MLSD), but this can be prohibitively complex, especially with a large number of samples. While the Viterbi algorithm simplifies this process, it still requires state tracking and channel estimation, which can be challenging. As a result, suboptimal approaches like Zero Forcing (ZF) and MMSE are often used because they are simpler to implement, though they come with a trade-off in performance, particularly in low SNR conditions.

Adaptive equalization extends these techniques by allowing the system to learn and adapt to the channel dynamically. This approach is much more efficient and effective because it enables real-time adaptation without the need to stop and retrain the system continuously. These adaptive techniques are widely used in many communication standards and technologies that you encounter daily.

In our next lecture, we will implement the MMSE equalizer using GNU Radio and observe the differences between MMSE and ZF equalization. After that, we'll move on to other topics, including Orthogonal Frequency-Division Multiplexing (OFDM) in wireless communications. Thank you for your attention, and I look forward to our next session.