Digital Communication using GNU Radio

Prof. Kumar Appaiah

Department of Electrical Engineering

Indian Institute of Technology Bombay

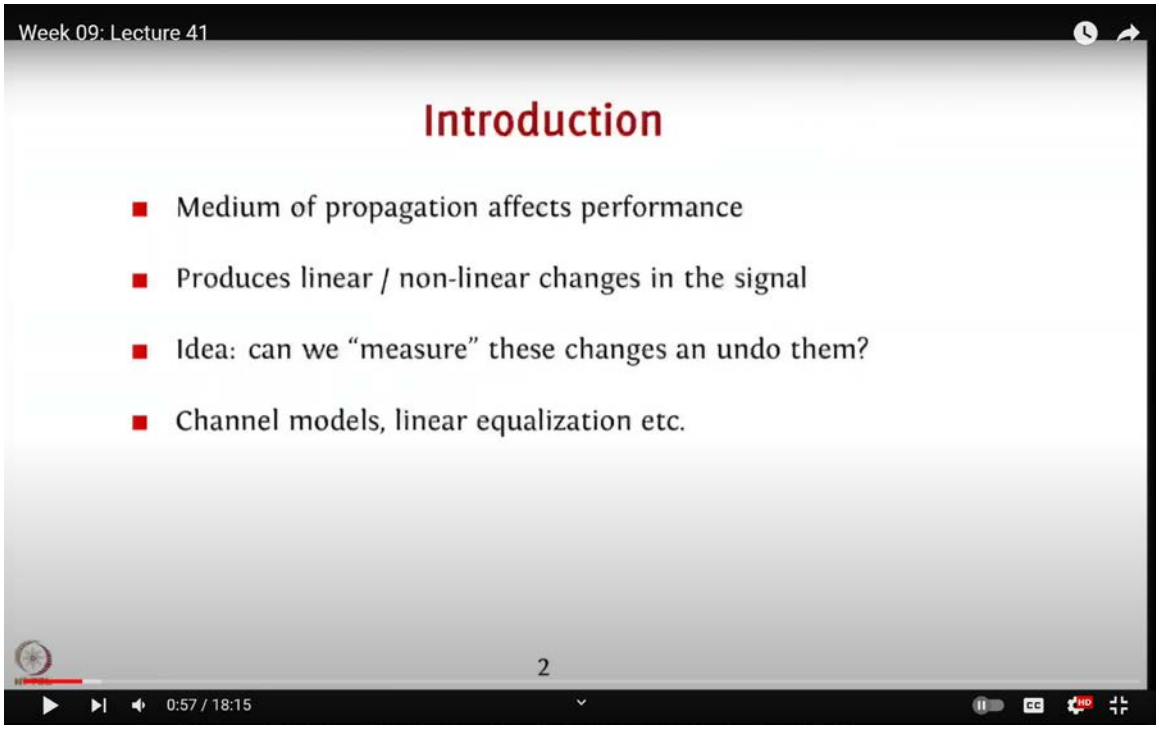
Week-09

Lecture-41

Channel Equalisation

Hello, and welcome to this lecture on Digital Communication using GNU Radio. My name is Kumar Appiah, and I am part of the Department of Electrical Engineering at IIT Bombay. In this lecture, we will delve into the topic of channel equalization, focusing on how different communication mediums affect signal propagation.

(Refer Slide Time: 00:57)

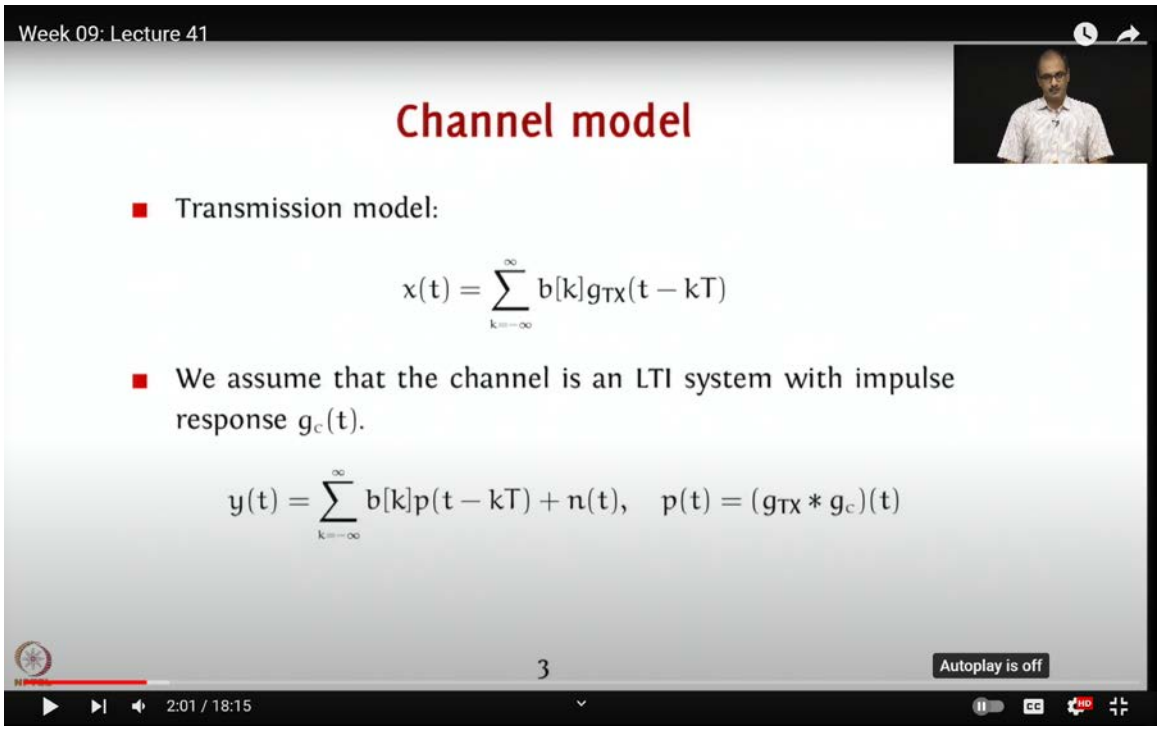


In addition to dealing with noise and receiver impairments, we need to consider how the communication medium itself alters the signal. The medium of propagation can significantly impact performance. For example, if the medium is a wire, distortion may

occur at high frequencies. If it's a wireless medium, issues such as reflection, noise, and phase changes can arise. Similarly, underwater or fiber optic channels introduce their own unique signal modifications. These changes can affect our ability to accurately detect symbols.

Mediums introduce both linear and non-linear alterations to the signal. The key question is: can we measure and correct for these changes? Fortunately, the answer is yes. Your everyday devices, such as telephones and cell phones, demonstrate this capability, as they calibrate to the medium and correct for its effects.

(Refer Slide Time: 02:01)



In this lecture, we'll examine basic channel models and explore linear equalization techniques. This involves understanding the channel distortion and attempting to reverse it, particularly when the channel behaves as a Linear Time-Invariant (LTI) system. This will be the focus of the upcoming lectures.

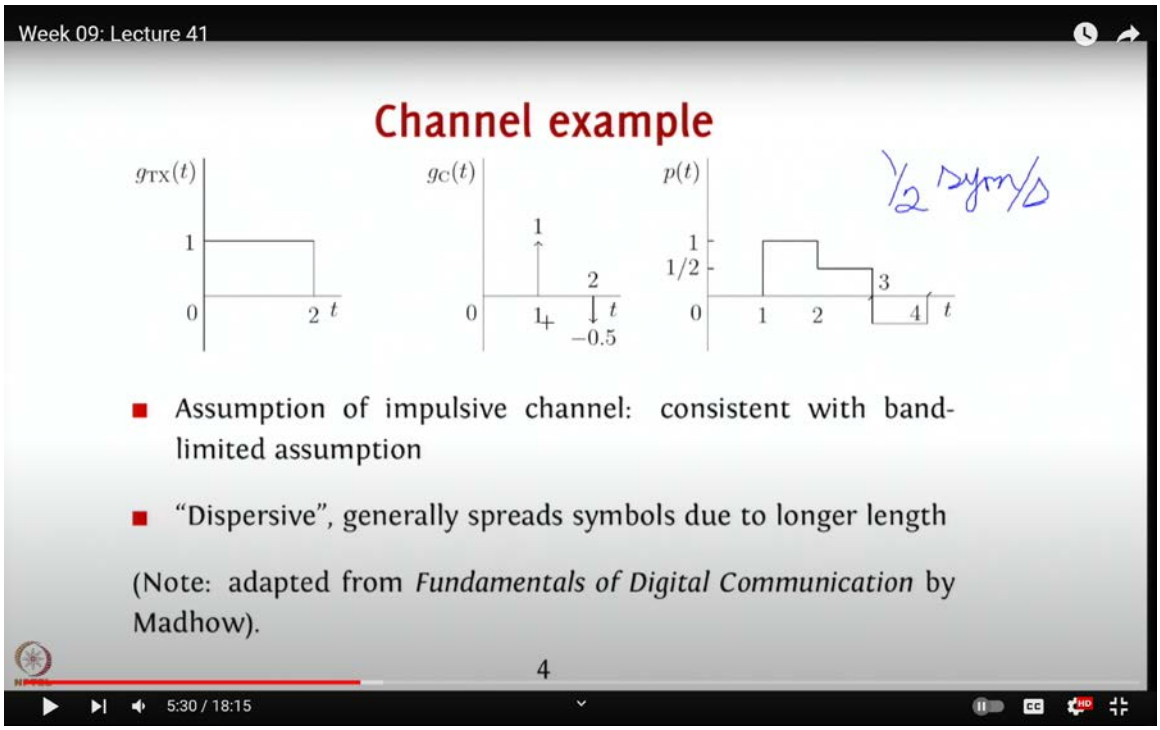
We will start with the transmission model we've used throughout this course. The transmitted signal, denoted as x(t) (or s(t) in some contexts), is given by:

$$x(t) = \sum_{k=-\infty}^{\infty} b_k g(t - kT)$$

Here, b_k represents the data points or constellation points, and g(t - kT) is the pulse shape used, which could be a sinc pulse, a raised cosine pulse, or a rectangular pulse, depending on your power and bandwidth constraints. This signal is then transmitted through the medium.

For this lecture, we'll assume that the channel is an LTI system with an impulse response $g_c(t)$. However, in reality, channels are not always linear and time-invariant. For instance, fiber optic cables can exhibit non-linear effects if significant power is used. Similarly, in a wireless medium, moving in a vehicle like a car or train changes the reflections the signal encounters, affecting the medium's characteristics.

(Refer Slide Time: 05:30)

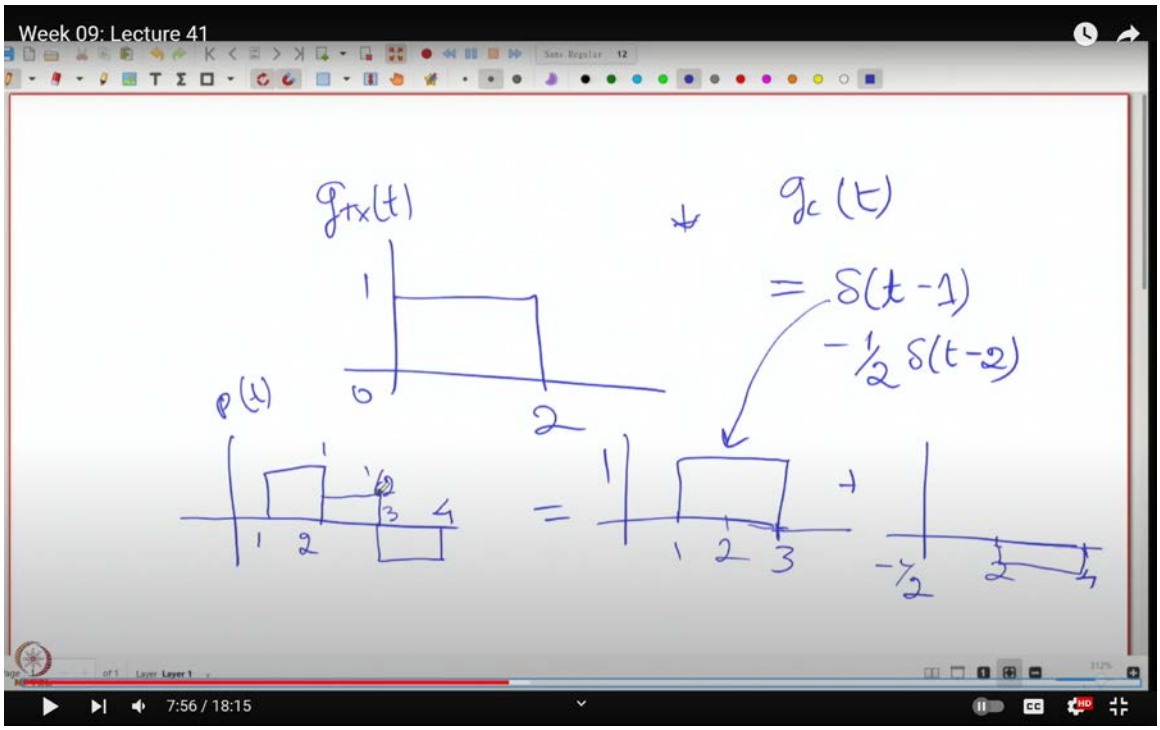


While the assumption of linearity and time invariance may not hold true in all cases, it is useful for analyzing and correcting the effects of the medium over short time snapshots. This assumption simplifies our understanding of how the medium alters the signal and

helps us develop effective correction strategies. Although you might encounter media that don't conform to these conditions, this channel model serves as a useful instructional example, adapted from reference materials for this course.

You will transmit symbols at a rate of half a symbol per second, using a rectangular pulse $g_{tx}(t)$ that spans from 0 to 2 seconds. In other words, your symbols will be sent at a rate of 0.5 symbols per second. The channel, which we are assuming to be an impulsive channel, has an impulse response characterized by an impulse at t = 1 with a weight of 1, and another impulse at t = 2 with a weight of -0.5.

(Refer Slide Time: 07:56)

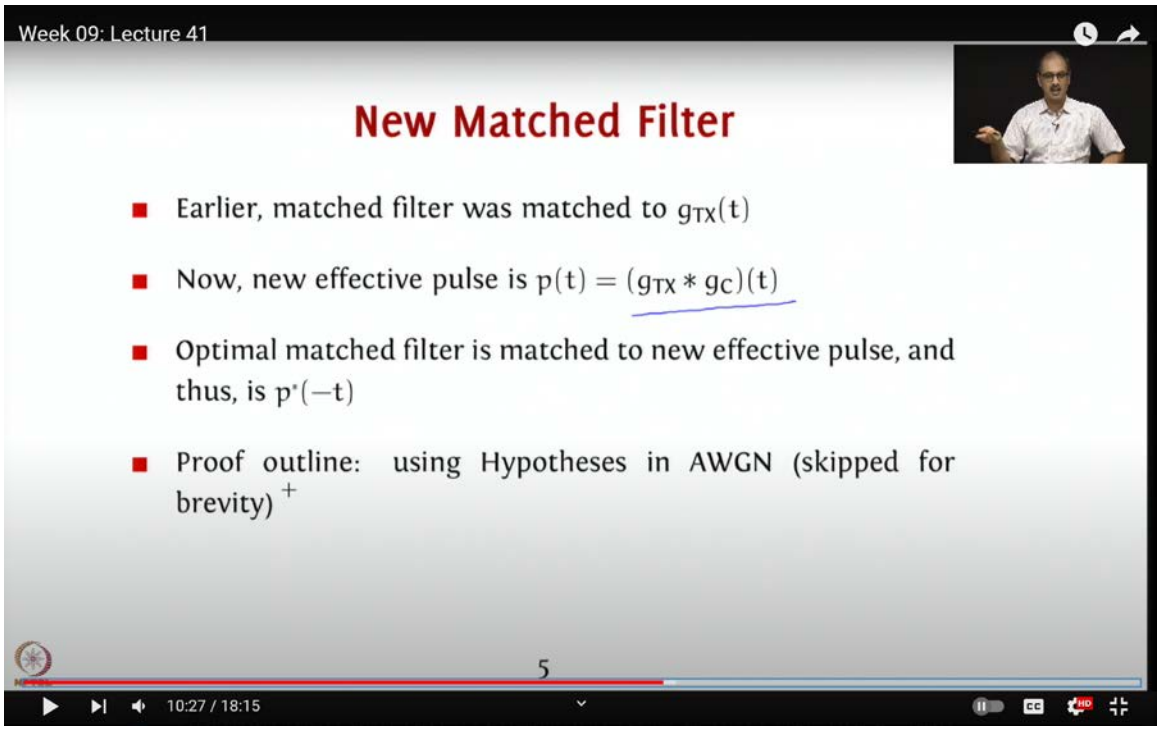


You might wonder why we refer to this as an impulsive channel and whether this is realistic. In practical terms, an impulsive channel does not necessarily mean the channel is literally impulsive. Instead, treating it as an impulse response is a good approximation within the bandwidth of interest because the channel essentially shifts and modifies the signal. Specifically, this channel can be represented as:

$$g_c(t) = \delta(t-1) - \frac{1}{2}\delta(t-2)$$

This representation indicates that the channel delays the signal and then adds another delayed version of the signal with a negative weight. This simplification is reasonable for modeling purposes, and it justifies the use of impulses in our channel model.

(Refer Slide Time: 10:27)



The effective pulse p(t) that you obtain after passing $g_{tx}(t)$ through the channel is given by the convolution of $g_{tx}(t)$ and $g_c(t)$. In the previous scenario, you only had $g_{tx}(t)$, but now the pulse after propagating through the channel is:

$$p(t) = g_{tx}(t) * g_c(t)$$

The optimal matched filter for this new effective pulse p(t) will be matched to $p^*(-t)$, which can be proven. Before proving this, let's verify that p(t) is indeed the result of the convolution of $g_{tx}(t)$ and $g_c(t)$.

Given $g_{tx}(t)$ and $g_c(t)$, where:

$$g_{tx}(t) = \text{rect}(t)$$

and

$$g_c(t) = \delta(t-1) - \frac{1}{2}\delta(t-2)$$

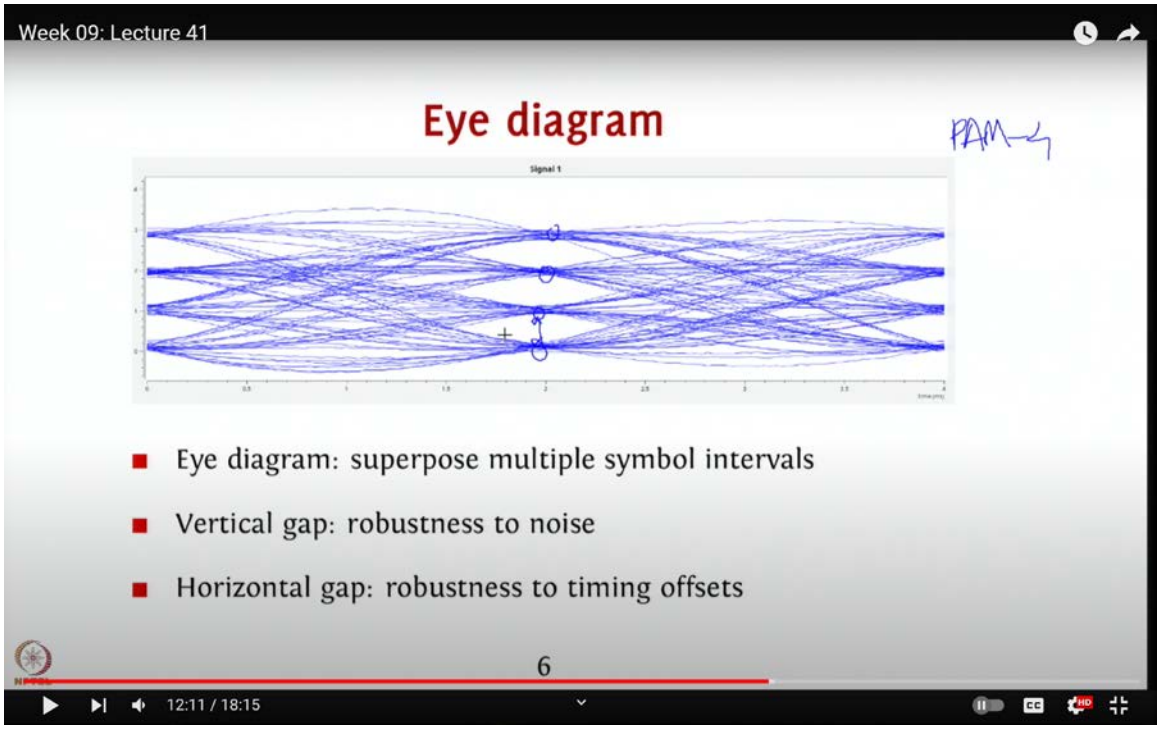
we perform the convolution:

$$p(t) = g_{tx}(t) * g_c(t)$$

Since $g_{tx}(t)$ is a rectangular pulse from 0 to 2 seconds, and $g_c(t)$ is an impulse function, the convolution process involves delaying $g_{tx}(t)$ by the amounts specified in $g_c(t)$ and combining them accordingly. Specifically:

- Delaying $g_{tx}(t)$ by 1 second and keeping the amplitude as 1.
- Delaying $g_{tx}(t)$ by 2 seconds and reducing the amplitude by 0.5.

(Refer Slide Time: 12:11)



The result of this convolution is:

- From 0 to 1 second, the convolution output is unaffected.
- From 1 to 2 seconds, the amplitude is 1 (due to the first impulse at t = 1).
- From 2 to 4 seconds, the amplitude is -0.5 (due to the second impulse at t = 2).

Thus, combining these effects, you get an effective pulse p(t) where the amplitude from 1 to 2 seconds remains 1.

From 1 to 2 seconds, the amplitude of the signal is 1. Between 2 and 3 seconds, you have an amplitude of 1 from $g_{tx}(t)$ and -0.5 from $g_c(t)$, resulting in a net amplitude of 0.5. From 3 to 4 seconds, there is no contribution from $g_{tx}(t)$, but $g_c(t)$ provides an amplitude of -0.5. Consequently, the effective pulse p(t) shows an amplitude of 1 from 1 to 2 seconds, 0.5 from 2 to 3 seconds, and -0.5 from 3 to 4 seconds. This is the p(t) we have derived.

This pulse p(t) represents the effective signal received. The key observation here is that while the transmitter's symbol was confined to 0 to 2 seconds, the received symbol now spans from 1 to 4 seconds. Essentially, there is a delay of 1 second and a spreading effect that extends the signal duration.

The spreading of the signal, caused by the channel's characteristics, leads to inter-symbol interference (ISI). This occurs because the signal from one symbol overlaps with the next symbol's duration. For example, the first symbol, initially from 0 to 2 seconds, now occupies from 1 to 4 seconds due to the delay. The second symbol, originally from 2 to 4 seconds, will extend from 3 to 6 seconds. Thus, you have overlap between symbols 1 and 2 from 1 to 4 seconds and between symbols 2 and 3 from 3 to 6 seconds. This overlap causes inter-symbol interference, where the symbols mix, making it necessary to manage this interference.

To recap, the new effective pulse p(t) was derived from the convolution of $g_{tx}(t)$ and $g_c(t)$. The optimal matched filter at the receiver should be matched to the conjugate time-reversed version of this pulse p(t), denoted as $p^*(-t)$. Although the detailed proof involves hypothesis testing in additive white Gaussian noise (AWGN), the intuitive approach is to minimize the squared norm of the difference between the received and transmitted signals. Expanding this norm ultimately leads to terms involving $g_{tx}(t)$ convolved with $g_c(t)$. This

process highlights that p(t) accounts for the spreading effect, so $p^*(-t)$ is the appropriate matched filter. Therefore, when dealing with a dispersive channel, the pulses get spread, leading to inter-symbol interference that needs to be addressed for accurate signal recovery.

Let's consider a multilevel channel, specifically a four-level channel used for PAM4 (Pulse Amplitude Modulation with 4 levels). In PAM4, the levels are 0, 1, 2, and 3. When you use a root raised cosine pulse for PAM4, you transmit pulses at these four levels. Each pulse overlaps with the next one, and this continuous overlap can be visualized using an eye diagram.

An eye diagram is created by taking symbol intervals and stacking them on top of each other. This visualization helps assess the signal's robustness to noise and timing offsets. In the eye diagram, each level corresponds to one of the PAM4 levels: 0, 1, 2, and 3. The vertical spacing between these levels indicates the system's robustness to noise, larger gaps suggest better resistance to noise. Horizontal spacing shows the impact of timing offsets on signal detectability.

An eye diagram is crucial for evaluating channel performance. Open eyes in the diagram indicate a well-performing channel where minimal correction is needed. Conversely, closed eyes suggest significant challenges in data recovery, necessitating substantial corrective measures. This visualization provides an intuitive sense of how well the channel is performing and helps determine the channel's impact on signal detectability.

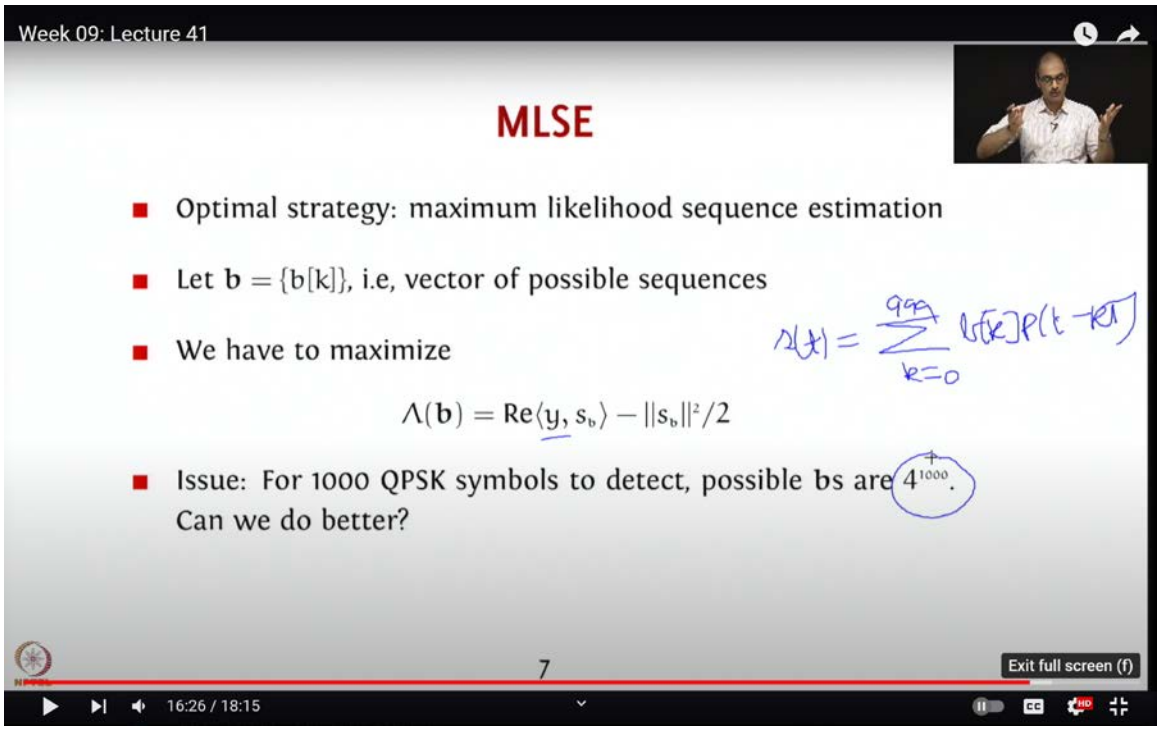
In GNU Radio, generating eye diagrams is straightforward, and we will cover this in a future lecture. The key takeaway is that the state of the eye diagram reflects the channel's effect on your signal and your ability to detect it.

Now, let's address the question of optimal strategy in such a scenario. When detecting a sequence of symbols, the strategy shifts to Maximum Likelihood Sequence Estimation (MLSE) or Maximum Likelihood Sequence Detection (MLSD). The goal is to estimate the most likely sequence of symbols that was transmitted, based on the observed sequence of received symbols.

Previously, for single symbol detection, you could make decisions symbol by symbol

without worrying about overlap. For instance, with a Nyquist filter, you could satisfy the Nyquist criterion and sample at the right moments to eliminate the effects of other pulses. However, in a scenario where symbols overlap significantly, this approach is no longer sufficient. The impact of one symbol will affect the detection of the next, making MLSE or MLSD necessary to handle the sequence as a whole and accurately estimate the transmitted sequence.

(Refer Slide Time: 16:26)



In the context of our discussion, we are dealing with the convolution effect, which, as mentioned, causes symbols to overlap. For instance, the first symbol, spanning from 1 to 4 seconds, overlaps with the second symbol, which spans from 3 to 6 seconds. This results in a combination of the first and second symbols between 3 and 4 seconds.

To address this issue, we need to maximize a specific metric, denoted as $\Lambda(b)$, where b represents a column vector and $\mathrm{Re}(y_{sb})$ is the real part of y_{sb}. Here, s_b refers to the sequence of vectors, which can be expressed as $s(t) = \sum_k b_k p(t - kT)$. This sequence is essentially what is received.

To maximize this metric, you would typically use Gaussian noise assumptions. By minimizing the expression involving the exponential of Gaussian noise, considering the noise affects each value of b independently, you obtain the necessary result. This process, while straightforward, involves dealing with potentially huge numbers of possibilities.

For example, if you have 1000 QPSK symbols to detect, the number of possible sequences of s_b you need to consider is 4^{1000}. This arises because each symbol can be one of four possibilities, leading to a total of 4^{1000} possible sequences. Clearly, performing a brute force search across such a vast number of possibilities is infeasible, especially as you move to larger constellations like QAM-16 or QAM-2048.

Given the prohibitive nature of such brute force approaches, the question arises: can we avoid maximum likelihood sequence estimation (MLSE) due to its computational impracticality? The answer is nuanced. While some suboptimal methods may be used, a more efficient approach exists. This involves transforming the problem into an additive metric, allowing for decision-making as computations proceed.

The Viterbi algorithm is a particularly effective method for performing MLSE without resorting to infeasible brute force searches. This algorithm optimizes the sequence estimation process by reducing the computational complexity significantly. In the next lecture, we will derive the Viterbi algorithm, explore its branch metrics, and learn how to use it to make optimal decisions for sequences. Thank you.