Digital Communication using GNU Radio

Prof. Kumar Appaiah

Department of Electrical Engineering

Indian Institute of Technology Bombay

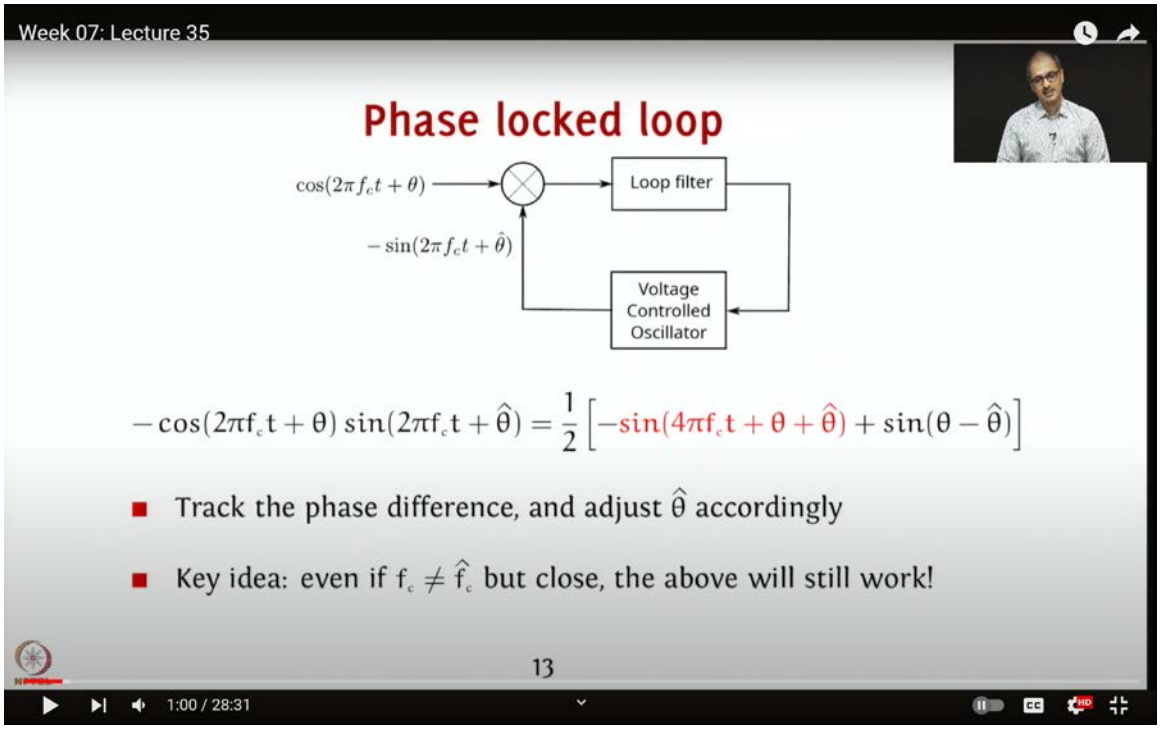
Week-07

Lecture-35

Phase Locked Loop and Differential Modulation

Welcome to this lecture on Digital Communication Using GNU Radio. I am Kumar Appiah from the Department of Electrical Engineering at IIT Bombay. In this session, which is the third in our series, we will continue relaxing the assumption that all receiver parameters are known. Specifically, we'll delve into synchronization and parameter estimation, focusing on how to estimate the delay, amplitude, phase, and frequency at the receiver.

(Refer Slide Time: 01:00)

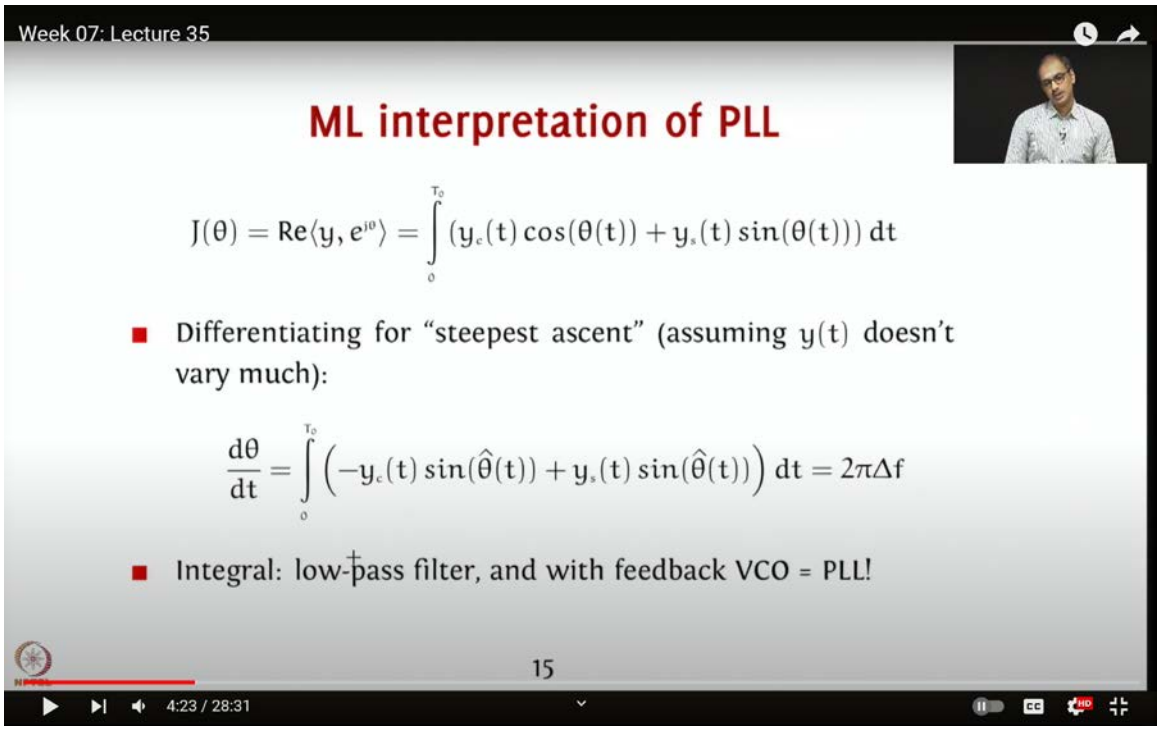


In our previous lecture, we touched upon the phase-locked loop (PLL). To quickly recap, a PLL is a feedback-based structure designed to track phase differences between two

frequencies. When these phases are tracked, the phase difference can be leveraged to adjust the voltage-controlled oscillator (VCO) so that it matches the carrier present in the received signal.

One important point I mentioned earlier is that the PLL functions exceptionally well with a pure carrier, meaning an unmodulated carrier signal. This is true even in the presence of Gaussian noise. However, when the carrier has a signal modulated onto it, extra care must be taken. In such cases, you might need to explore alternative methods to retrieve the local oscillator that matches the transmitter's carrier.

(Refer Slide Time: 04:23)



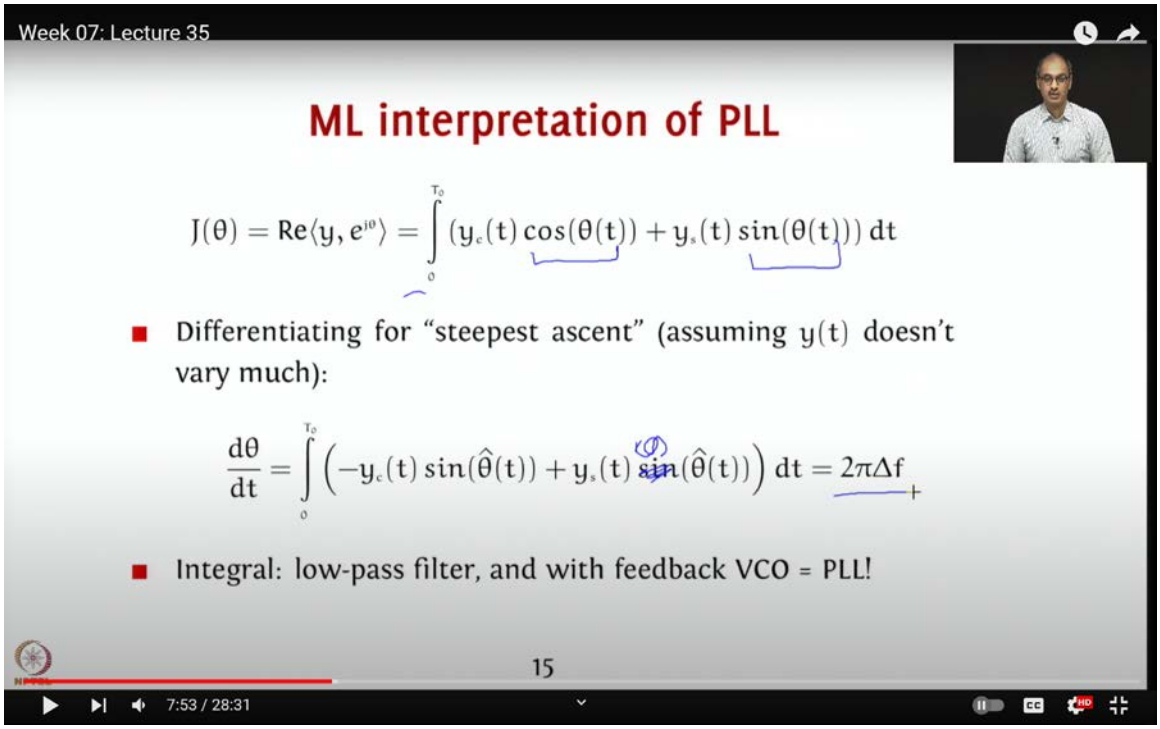
Next, we will discuss the maximum likelihood interpretation of the phase-locked loop. Imagine we consider an observation interval of length t_0, ensuring it's long enough to capture the variations of the carrier. During this interval, your received signal y(t) can be represented as $e^{j\theta} \times n(t)$.

Now, let's consider the phase variation, which I've been implicitly ignoring. Assume that $e^{j\theta(t)}$ represents this variation, where the phase θ(t) is not constant but varies with time.

As you're aware, the variation of phase is directly related to frequency, if you differentiate the phase θ(t), you obtain the angular frequency ω. Therefore, θ(t) is not just a static value but a time-varying quantity, and we need to determine the frequency offset that causes this variation.

The likelihood function L(y|θ) can be expressed as $\exp\left(\frac{1}{\sigma^2}\mathrm{Re}\left(ye^{j\theta(t)} - \frac{t_0}{2}\right)\right)$. Here's how this comes about: if you integrate the noise n(t) over an interval [0, t_0], it becomes a random variable with a variance proportional to t_0. When you consider this, t_0 is just an auxiliary parameter, not critical to our main concern. We need to focus on maximizing the expression $\int_0^{t_0} -y_c(t)$.

(Refer Slide Time: 07:53)



To break it down further, if we represent y(t) as $y_c(t)$ + $jy_s(t)$, this expression becomes $-\int_0^{t_0}[y_c(t)\cos\theta\,(t) + y_s(t)\sin\theta\,(t)]\;dt$. Essentially, this involves multiplying by cos θ(t) + j sin θ(t) and taking the real part. Notice that in this integral, terms like $y_c(t)$ cos θ(t) appear, and our goal is to maximize this to accurately track the phase over time. The

challenge here, compared to standard phase tracking, is that $\theta(t)$ is varying during the interval, requiring us to match this varying phase as precisely as possible.

To achieve this, I've rewritten the expression for clarity. If we apply the method of steepest ascent to maximize this function, why steepest ascent, you might ask? Whenever you want to maximize a function, you compute its derivative and move in the direction of the derivative to reach the peak. Imagine a function shaped like a hill; you find the slope (derivative) and move upwards in that direction until you reach the peak, where the derivative equals zero. This is the approach we'll use to match the varying phase $\theta(t)$ over time.

So, this is why we refer to this process as steepest ascent. Essentially, what we're doing is differentiating the function $J(\theta)$, but there's a subtle trick involved here. When you differentiate $J(\theta)$, you obtain the derivatives of Y_c and Y_s. However, in this context, we're not focusing on the derivatives of Y_c and Y_s themselves because we're assuming that they vary slowly. This assumption is crucial because it implies that the frequency the phase-locked loop (PLL) is tracking is much more significant than any variation in Y_c and Y_s.

Therefore, we only perform the differentiation with respect to these terms. Upon differentiation, you get $-Y_c \sin(\theta) + Y_s \cos(\theta)$, which is proportional to $2\pi\Delta f$. A common question is how this result corresponds to $\frac{d\theta}{dt}$. Well, when you differentiate $J(\theta)$, you apply the chain rule, resulting in $\frac{dJ}{d\theta} \times \frac{d\theta}{dt}$. By isolating $\frac{d\theta}{dt}$, this is essentially what you obtain. You can explore the details further, but for now, I'll move on since we've spent quite some time on this topic.

If you examine this closely, it resembles the process we discussed earlier, where we mixed a cosine signal with a negative sine. The similarity lies in how this approach leads you to the expression $2\pi\Delta f$. Now, think of an integral as a low-pass filter. Recall that we performed an integration with respect to t_0. Here, this integral acts as a low-pass filter. With the help of feedback, the voltage-controlled oscillator (VCO) corrects for the $2\pi\Delta f$, which is the frequency deviation, thereby forming a phase-locked loop.

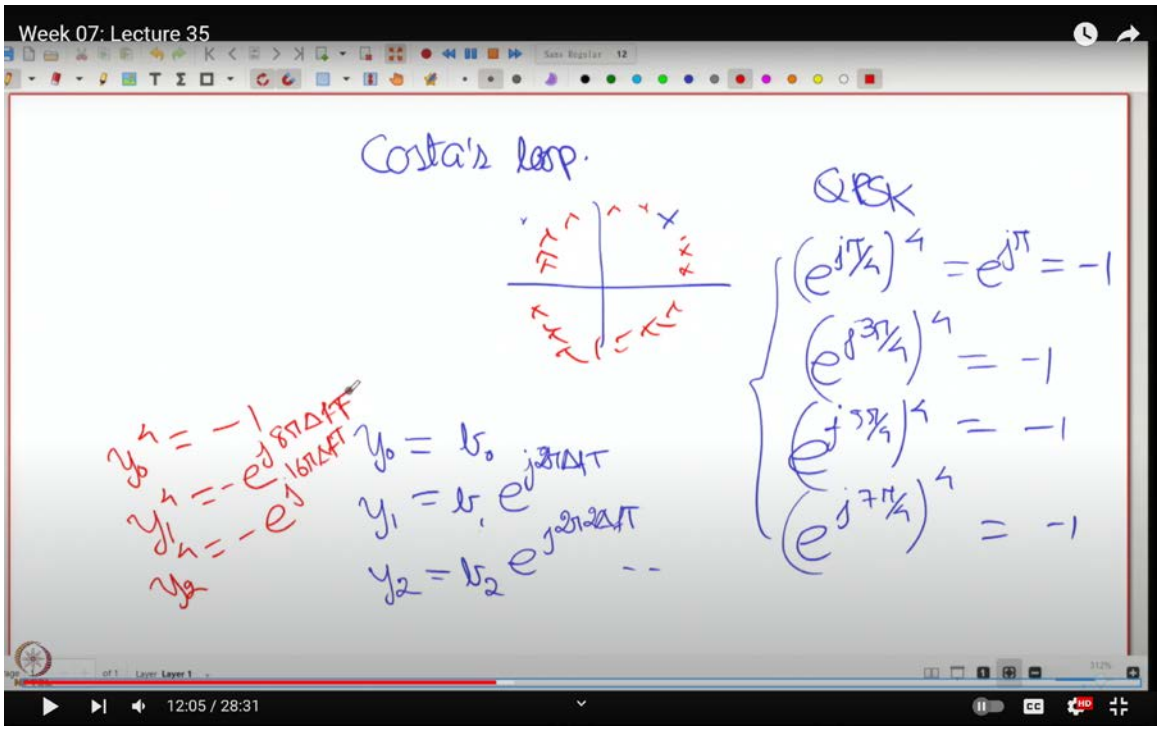
The loop filter and the entire mechanism we've described are, in essence, performing an

averaging or integration process, which then adjusts the VCO to yield the correct output frequency. This system simultaneously tracks both the frequency and the phase.

Intuitively, a PLL aims to track the phase. However, since it continuously tracks the phase over time, the phase variation over time corresponds directly to the frequency offset. This is the key insight for understanding how a PLL operates, even from a maximum likelihood perspective. We initially set up J(θ) as an optimization metric, and maximizing it translates to optimal frequency tracking. This concept can be explored further in specialized references.

In fact, the design of PLLs is a well-researched area, and there's a wealth of circuit-related literature available that covers efficient implementations of PLLs under various constraints and modulation formats. For those interested, these references provide a deep dive into the topic.

(Refer Slide Time: 12:05)



In our exploration of GNU Radio, we will delve into certain aspects of Phase-Locked Loops (PLLs), including simulations to evaluate their performance under noisy conditions.

As a brief aside, there's something known as Costas Loop, which is particularly used for Phase Shift Keying (PSK). Costas Loop is designed specifically to handle PSK, allowing us to perform symbol sampling while simultaneously tracking frequency offset by monitoring the phase offset.

Let me illustrate this with a practical scenario. Suppose you are transmitting PSK symbols, and let's consider the case of Quadrature Phase Shift Keying (QPSK). Ideally, you'd expect your QPSK symbols to be received accurately. However, due to the inevitable frequency offset, which manifests as a phase offset, what actually happens at the receiver is a continuous rotation of these symbols.

Now, why does this rotation occur? As we've previously discussed, when there's a phase offset, your signal is multiplied by $e^{j\varphi}$. If this phase offset changes over time, you then have $e^{j\varphi(t)}$, which introduces a dynamic, time-varying phase shift to your PSK symbols. This rotation can severely degrade the performance of your system, as it causes the received symbols to deviate from their expected positions.

So, how do we counteract this problem? We can employ a clever technique. Consider QPSK again, where you take $e^{j\frac{\pi}{4}}$, which represents one of the QPSK symbols. When you raise this to the power of 4, $\left(e^{j\frac{\pi}{4}}\right)^4$, you obtain $e^{j\pi}$, which is equivalent to -1. Similarly, for the other QPSK symbols like $e^{j\frac{3\pi}{4}}$, $e^{j\frac{5\pi}{4}}$, and $e^{j\frac{7\pi}{4}}$, they also map to -1.

This observation leads us to a useful trick. Let's denote the received symbols as follows:

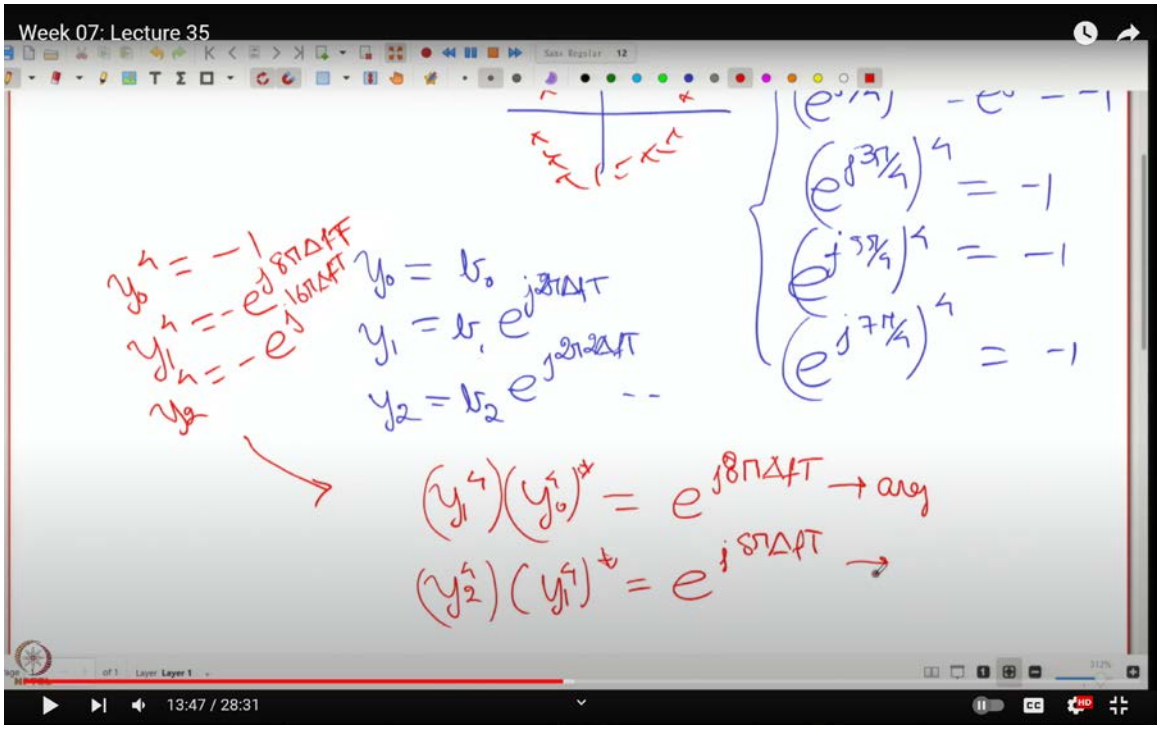
- $y_0 = b_0$ (assuming no noise for simplicity),
- $y_1 = b_1 e^{j2\pi\Delta f \cdot t}$, where Δf is the frequency offset and t is the duration of a symbol,
- $y_2 = b_2 e^{j2\pi 2\Delta f \cdot t}$, and so on.

The challenge, however, is that we don't know the exact values of b_0, b_1, and b_2, though we do know that they are QPSK symbols. Despite this uncertainty, we can still leverage the periodic properties of QPSK symbols to mitigate the impact of the frequency offset.

Let's explore the trick we're about to apply. When you take the fourth power of y_0, it results

in -1. Why? Because y_0 is one of the four QPSK symbols, and raising any of these symbols to the fourth power yields -1. Similarly, raising y_1 to the fourth power gives you b_1^4, which equals -1, and this result becomes $-e^{j8\pi\Delta fT}$. For y_2^4, it becomes $-e^{j16\pi\Delta fT}$.

(Refer Slide Time: 13:47)

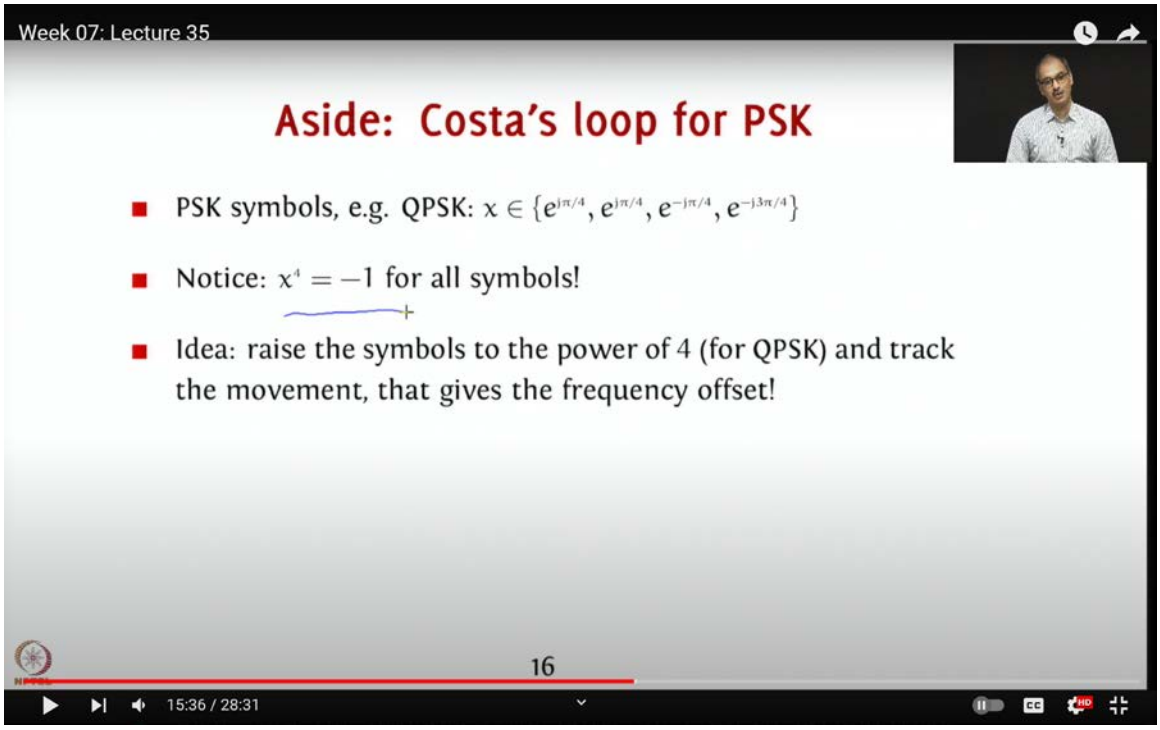


Now, let's analyze this further. If you multiply y_1^4 by the conjugate of y_0^4, what do you obtain? You get $e^{j8\pi\Delta fT}$. Likewise, if you multiply y_2^4 by the conjugate of y_1^4, the result is again $e^{j8\pi\Delta fT}$. Notice how consistent this is. At the receiver, if you know the symbol duration (which we'll assume you've already determined), you can extract the frequency offset by taking the argument of this result. Specifically, the argument is 8π Δf T, and when you divide by 8π T, you're left with the frequency offset Δf.

In the presence of noise, you simply take a larger number of these symbols, compute their arguments, and average out the noise to effectively determine the frequency offset. This method is highly versatile, working seamlessly for various PSK constellations, whether it's BPSK, QPSK, 8-PSK, or 16-PSK, as long as the constellation points are equi-spaced and lie on a circle.

This technique also extends to other QPSK variants. For instance, if you use a QPSK constellation rotated by π/4, where the points are 1, -1, j, and -j, raising these to the fourth power yields 1 instead of -1. Thus, the technique remains effective for this rotated version as well.

(Refer Slide Time: 15:36)

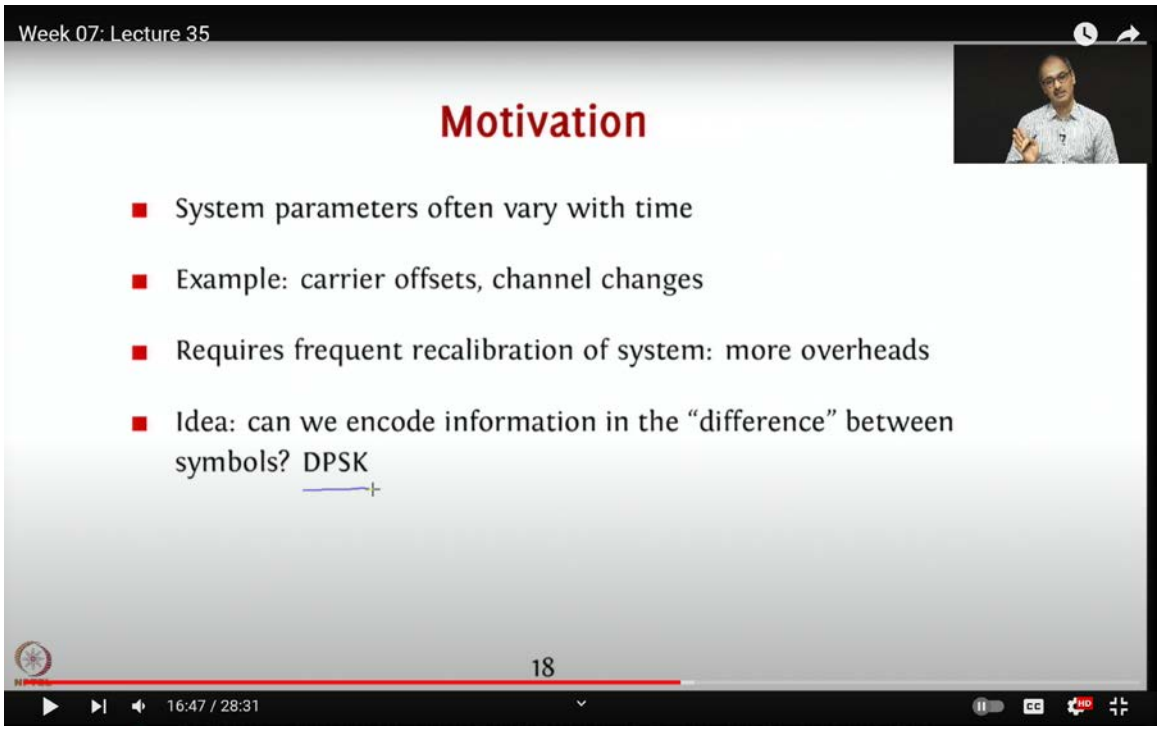


The Costas Loop, in particular, allows for very straightforward tracking of the frequency offset in PSK constellations. By raising the symbols to the fourth power and tracking their movement, you can accurately determine the frequency offset. However, it's important to note that while the Costas Loop is intuitive and easy to implement, it may falter in the presence of other impairments, such as channel effects.

Let's consider a scenario where your symbols, unfortunately, get convolved with some channel or experience other distortions. In such cases, adjustments are necessary for the Costas loop to function effectively. For instance, when facing the challenge of jointly estimating parameters like delay, amplitude, frequency offset, phase, and even the channel itself, the Costas loop might not be the most effective solution. A more traditional

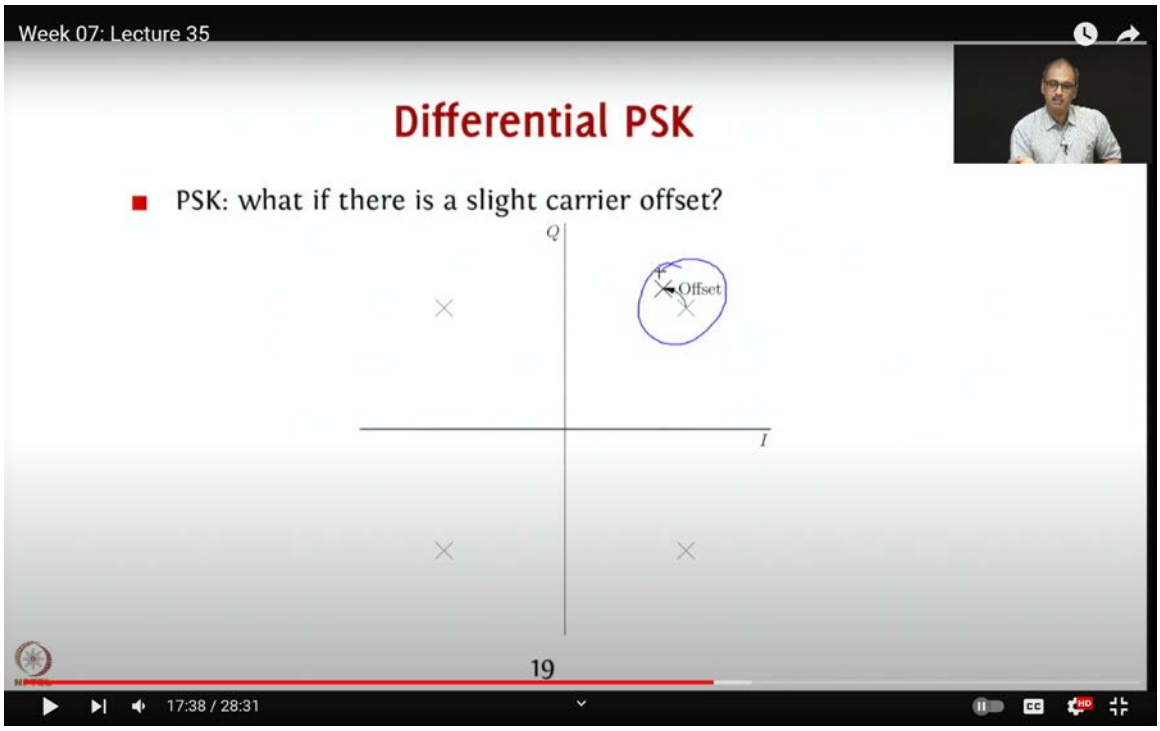
approach, such as a phase-locked loop (PLL), can be easier to implement because it allows for more straightforward extraction of these frequency components. While the Costas loop is a powerful conceptual tool, especially in understanding how the PSK constellation behaves in the presence of frequency offsets, it may require more complex adjustments when additional impairments are present. You'll explore the application of the Costas loop through simulations in GNU Radio, where its practical strengths and limitations will become clear.

(Refer Slide Time: 16:47)



Now, let's transition to a related topic: differential modulation. I'll start by providing some context. So far, we've discussed how system parameters like phase and frequency can vary over time, leading to differences between the transmitter and receiver. Typically, these differences require calibration, such as through a phase-locked loop, to keep the system aligned. However, frequent recalibration introduces overhead. But what if we could take advantage of the fact that these system parameters change only slowly? Could we encode information in the difference between successive symbols, rather than continuously recalibrating? This is where differential phase shift keying (DPSK) comes into play.

(Refer Slide Time: 17:38)

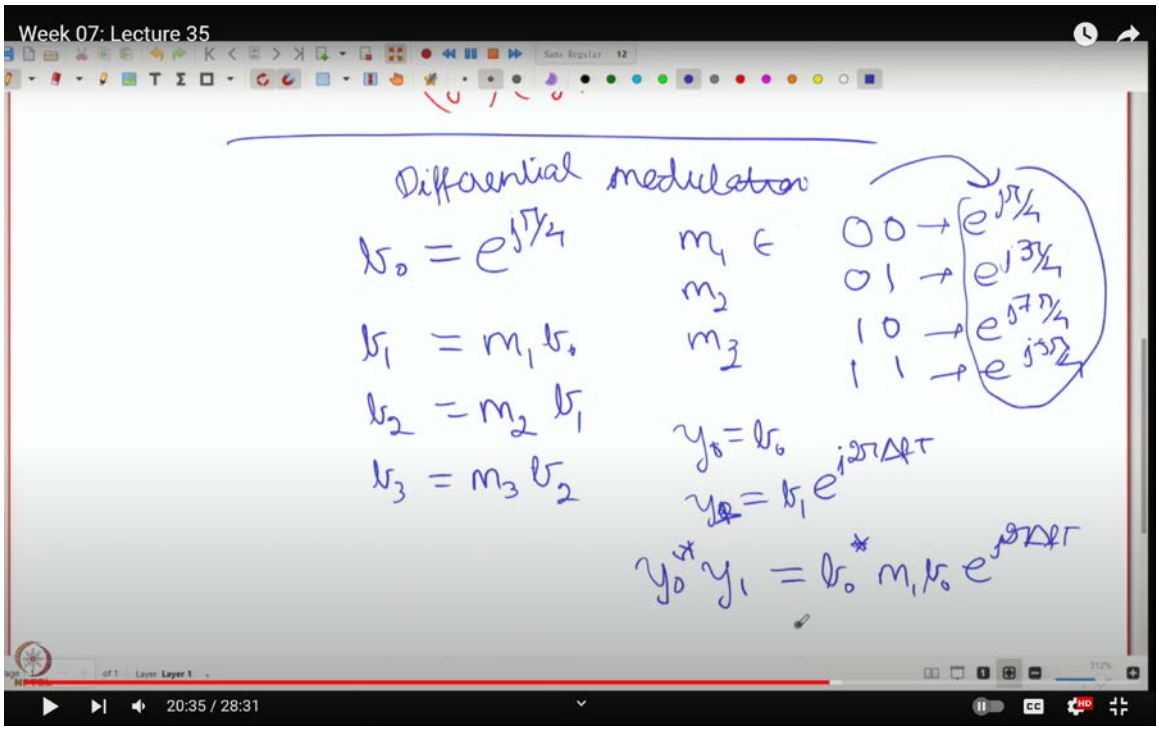


DPSK is a form of non-coherent communication, meaning that it doesn't rely on traditional carrier recovery methods. Instead, it exploits the fact that the carrier at the receiver is nominally close to that at the transmitter, allowing for communication without precise synchronization. How does this work? Imagine you're using QPSK, but there's a frequency offset. In differential PSK, instead of decoding the absolute phase of each symbol, you focus on the phase difference between successive symbols. This approach allows the system to handle frequency offsets more gracefully, as it relies on relative phase changes rather than absolute values.

The frequency offset results in a rotation of the constellation points. For example, if you transmit the symbol 1 + 1j, it will be slightly shifted, moving along the circle but away from its original position. This shift makes the system less robust to noise because the constellation's movement exacerbates the impact of noise. The traditional solution is to use a phase-locked loop (PLL) or some other method to estimate the carrier frequency. However, suppose we aim to avoid carrier estimation altogether and rely on the fact that the offset doesn't change significantly. In that case, we can apply a similar strategy as we

discussed with the Costas loop.

(Refer Slide Time: 20:35)



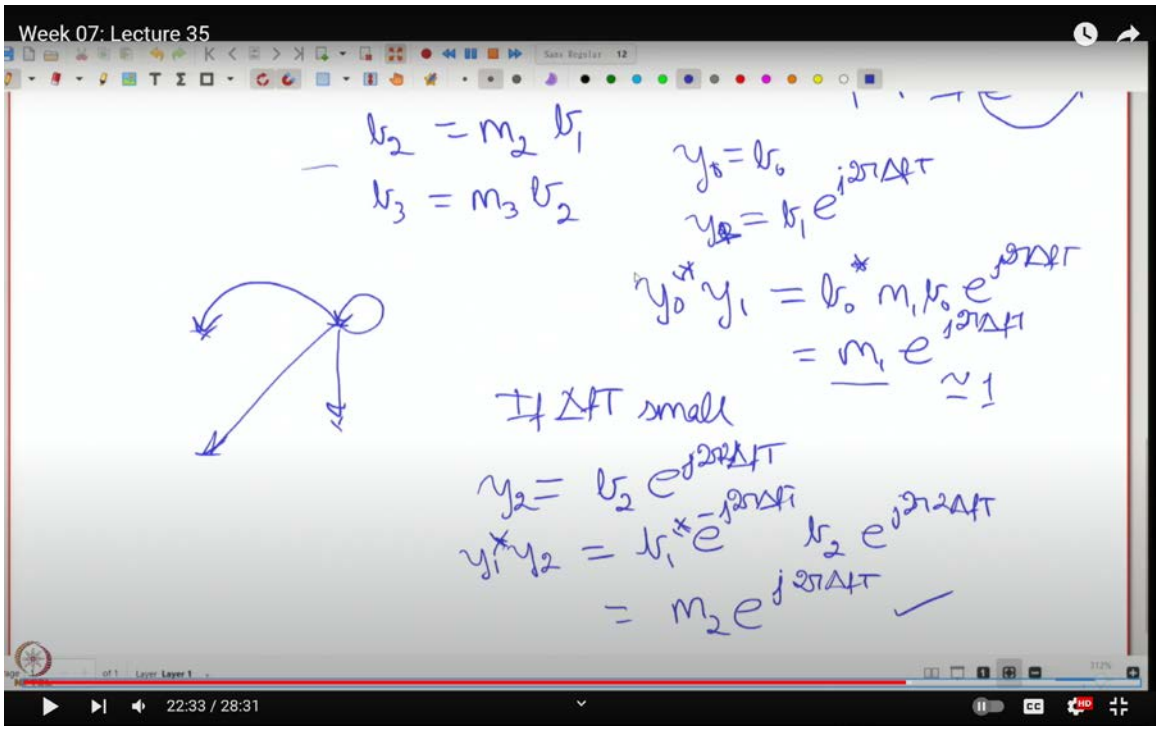
Here's what we can do: instead of encoding information in the constellation points themselves, we can encode it in the phase difference between successive points. Let me explain what I mean by phase difference. Assume b_0 is our reference point, always represented as $e^{j\pi/4}$. Recall the mapping 00, 01, 10, 11 that we discussed earlier with Gray coding, which helps minimize bit errors in the presence of symbol errors. This mapping could correspond to points like $e^{j\pi/4}$, $e^{j3\pi/4}$, $e^{j7\pi/4}$, and $e^{j5\pi/4}$.

Now, consider that for b_1, we write it as $m_1 \times b_0$, where m_1 is the symbol we want to transmit, belonging to a set of possible symbols m_1, m_2, and so on. Similarly, b_2 is encoded as $m_2 \times b_1$. This means we are always encoding the information in the phase difference. Following this logic, b_3 would be $m_3 \times b_2$.

Let's apply this to the scenario we discussed earlier. Suppose $y_0 = b_0$ and $y_1 = b_1 e^{j2\pi\Delta ft}$, where the phase shift $e^{j2\pi\Delta ft}$ is caused by the frequency offset. Although this phase shift is a problem, the way we've encoded the information, in the phase difference, allows us to

manage it.

(Refer Slide Time: 22:33)



Now, if we take the conjugate of y_0 and multiply it by y_1, we get:

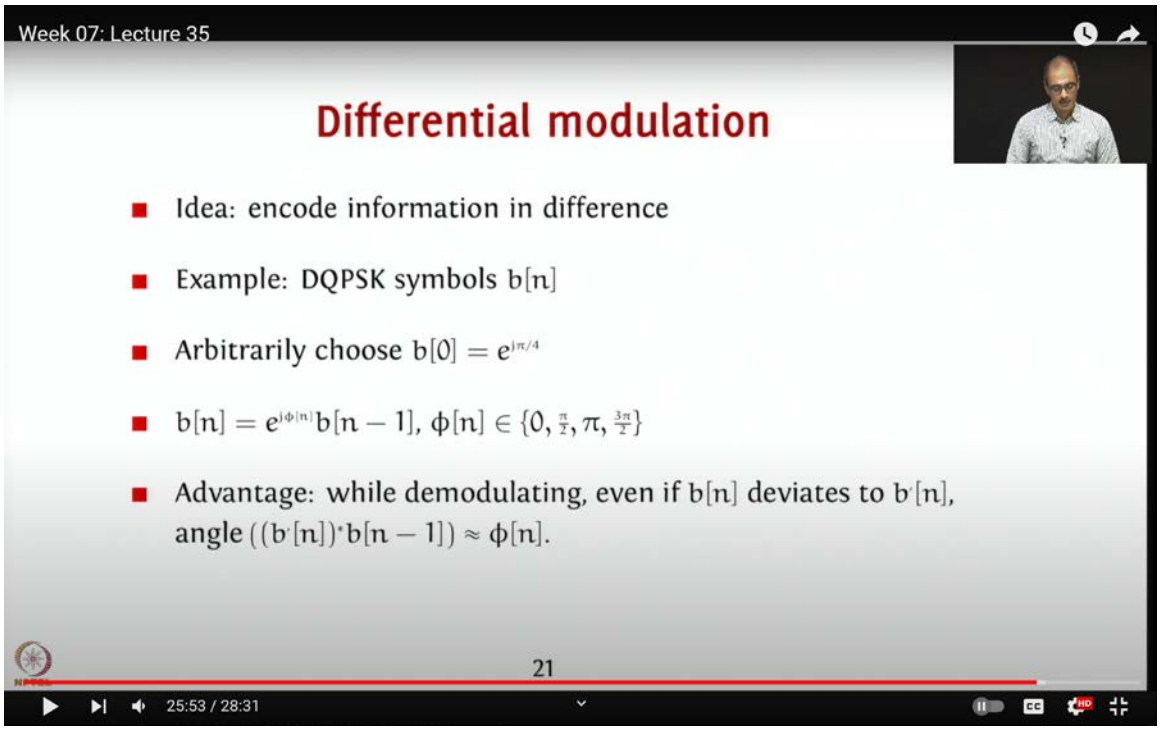
$$y_0^* y_1 = b_0^* \times \left(m_1 \times b_0 \times e^{j2\pi\Delta f t}\right) = m_1 \times e^{j2\pi\Delta f t}$$

If the product $\Delta f \times t$ is small, then $e^{j2\pi\Delta f t}$ is approximately equal to 1. This means the phase shift introduced by the frequency offset is minimal, and you can recover your symbol m_1 directly. This approach effectively mitigates the impact of frequency offsets without needing carrier recovery, making it a powerful method in scenarios where the offset is relatively small.

Let's work through another example. Assume y_2 is given by $b_2 e^{j2\pi\Delta f t}$, with the assumption that the frequency offset Δf is constant. We want to evaluate $y_1^* y_2$, which can be computed as follows:

$$y_1^* y_2 = \left(b_1^* e^{j2\pi\Delta f t}\right)\left(b_2 e^{j2\pi 2\Delta f t}\right)$$

(Refer Slide Time: 25:53)



This simplifies to:

$$y_1^* y_2 = b_1^* b_2 e^{j2\pi\Delta ft} e^{j2\pi 2\Delta ft} = b_1^* b_2 e^{j2\pi 3\Delta ft}$$

Notice that $b_1^* b_2$ represents the product of the complex conjugate of b_1 and b_2. For PSK constellations, this product essentially equals m_2, so:

$$y_1^* y_2 = m_2 e^{j2\pi 3\Delta ft}$$

Here, $b_1^* b_2$ simplifies to m_2, as these are PSK symbols. The term $e^{j2\pi 3\Delta ft}$ represents the phase shift introduced by the frequency offset.

The core idea here is to encode information in the phase differences between symbols. By analyzing these phase shifts, you can extract the information from the constellation. For instance, if you consider phase jumps of π/2, you can account for the phase differences more effectively.

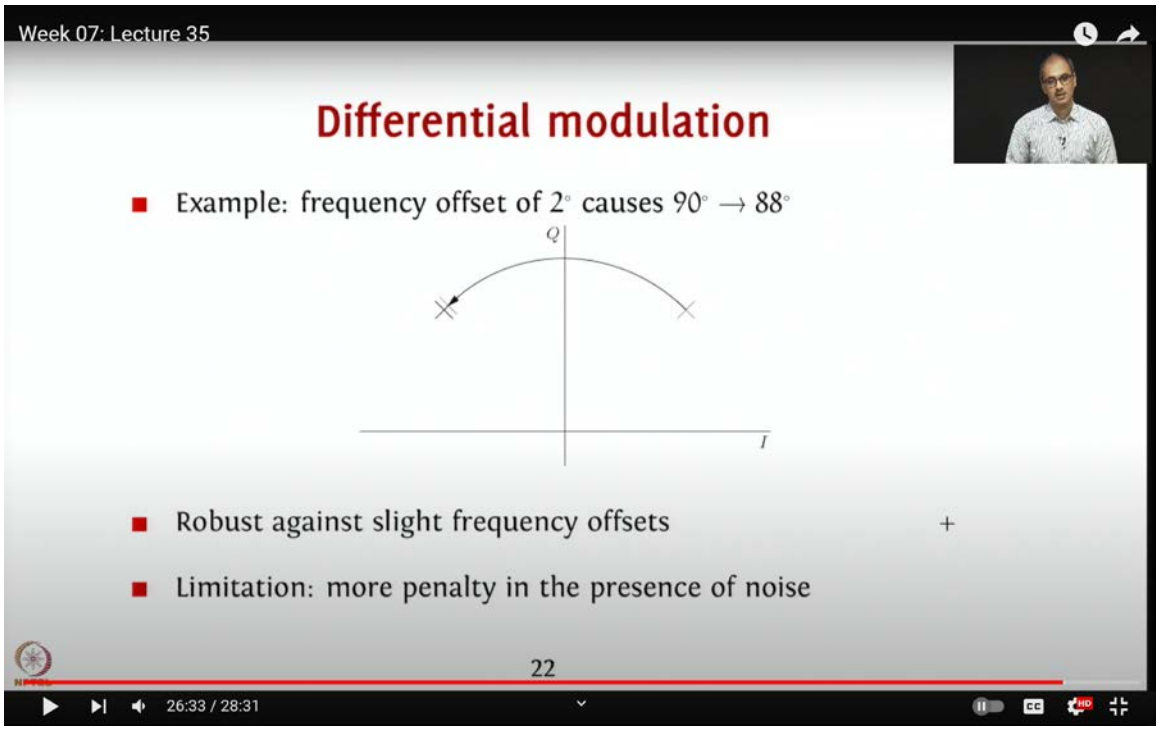
In practical terms, if you use differential modulation with jumps of π/4, the phase

transitions will encode the information. For example:

- The symbol 00 might correspond to e^{j0} (no phase jump).
- The symbol 01 might correspond to $e^{j\pi/2}$ (a π/2 phase jump).
- The symbol 11 could be $e^{j\pi}$ (a π phase jump).
- The symbol 10 might be $e^{-j\pi/2}$ (a -π/2 phase jump).

By focusing on these phase transitions, you effectively ignore the impact of the frequency offset, provided that Δf t remains small. Thus, differential phase shift keying (DPSK) allows you to encode information in the differences between symbols rather than their absolute phases, making the system more resilient to frequency offsets.

(Refer Slide Time: 26:33)



Let's discuss some of the potential pitfalls of this approach. One major issue arises when the frequency offset is significantly large. For instance, if your constellation point is ideally positioned, but the frequency offset shifts it considerably, you encounter a problem. In this case, your approach of encoding information based on phase differences might lead you astray. If the frequency offset causes your signal to move far from its intended position,

you will need to correct for this shift, which becomes challenging, especially in the presence of noise.

When the frequency offset is substantial and combined with high noise levels, the differential approach can struggle. Because this method does not accurately estimate the carrier phase but rather attempts to track it, it tends to be less robust against noise. Differential phase shift keying (DPSK), for example, has poorer signal-to-noise ratio (SNR) properties compared to other modulation schemes. You end up with a higher bit error rate or symbol error rate for the same SNR when using DPSK.

To formalize this, in differential QPSK, we arbitrarily choose b_0 as $e^{j\pi/4}$ and b_n as $e^{j(\varphi_n+\varphi_{n-1})}$, where φ_n represents possible phase shifts (0, $\pi/2$, π, or $3\pi/2$). The process involves calculating the argument of the ratio of the received signal and its previous symbol, which gives you the phase information. This approach is relatively robust against small frequency offsets but suffers from a higher penalty in noisy environments.

In summary, our lectures have highlighted the importance of accurate parameter estimation at the receiver. For example, in QPSK or 16-QAM, precise amplitude estimation is crucial for correct decision regions. Similarly, accurately determining the sampling location requires estimating the symbol start delay. Additionally, getting the phase and frequency offset right is essential for optimal performance. There are several approaches for handling frequency and phase offsets, such as the phase-locked loop (PLL) and the Costas loop. These methods have been tested and refined over the years to suit different scenarios.

One strategy is to forego precise frequency or phase offset estimation and instead transmit signals differentially across the phase. This approach works well when dealing with small frequency offsets. However, it struggles with large frequency offsets and significant noise. In such cases, differential modulation is less effective. Generally, coherent or carrier-based receivers, which rely on local oscillators, offer better signal-to-noise ratio (SNR) to bit rate performance compared to differential modulation. In the upcoming lectures, we will explore these impairments using GNU Radio and then move on to examine the next major issue: channel modeling. Thank you.