3.Synchronization

3.1 Introduction to Synchronization

In almost every discussion of receiver or demodulator performance, some level of signal synchronization is assumed, although this assumption is often not explicitly stated. For example, in the case of coherent phase demodulation (PSK), the receiver is assumed to be able to generate reference signals whose phases are identical (except perhaps for a constant offset) to those of the signaling alphabet at the transmitter. These reference signals are compared with the incoming signals in the process of making maximum-likelihood symbol decisions.

In order to be able to generate these reference signals, the receiver has to be in synchronization with the received carrier. This means that there has to be phase concurrence between the incoming carrier and a replica of it in the receiver. In other words, if there were no information modulated on the incoming carrier, the incoming carrier and the replica in the receiver would pass through zero simultaneously. This is what is known as being in phase lock and is a condition that must be closely approximated if coherently modulated signals are going to be accurately de- modulated at the receiver. Being in phase lock means that the receiver's local oscillator is synchronized in both frequency and phase with the received signal. If the information-bearing signal is not modulated directly on the carrier but indirectly through the use of a subcarrier, both the phase of the carrier and that of the subcarrier must be determined. If the carrier and subcarrier are not kept in phase synchronism by the transmitter (they typically are not), this will require the generation of a replica of the subcarrier by the receiver, where the phase of the subcarrier replica is controlled separately from that of the carrier replica. This will enable the receiver to achieve phase lock on both the carrier and subcarrier.

It is also assumed that the receiver has accurate knowledge of when an in-coming symbol started and when it is over. This knowledge is required in order to know the proper symbol integration interval-the interval over which energy is integrated prior to making symbol decisions. Clearly if the receiver integrates over an interval of an inappropriate length, or over an interval that spans two symbols. the ability to make accurate symbol decisions will be degraded.

It can be seen that symbol synchronization and phase synchronization are similar in that they both involve producing in the receiver a replica of a portion of the transmitted signal. For phase synchronization, it was an accurate replica of the carrier. For symbol synchronization, it is a square wave at the symbol transition rate. The receiver must, in effect, be able to produce a square wave that will transition through zero simultaneously with the incoming signal's transitions between symbols. A receiver that is able to do this can be said to have symbol synchronization, or to be in symbol lock. Since there are typically a very large number of carrier cycles per symbol period, this second level of synchronization is much coarser than phase synchronization and is usually done with different circuitry than that used for phase synchronization.

In many communication systems an even higher level of synchronization is required. This is usually called frame synchronization. Frame synchronization is required when the information is organized in blocks, or messages of some uniform number of symbols. This will occur, for example, if a block code is used for forward error control, or if the communications channel is being time-shared, on a regular basis, by several users (TDMA). In the case of block coding, the decoder needs to know the location of boundaries between code words in order to decode the message correctly. In the case of a time-shared channel, it is necessary to know where the location of boundaries between channel users are, in order to route the information appropriately. Similar to symbol synchronization, frame synchronization is equivalent to being able to generate a square wave at the frame rate, with the zero crossings coincident with the transitions from one frame to the next.

Most digital communications systems using coherent modulation require all three levels of synchronization: phase, symbol, and frame. Systems using non-coherent modulation techniques will typically require symbol and frame synchronization. but since the modulation is not coherent, accurate phase lock is not required. In- stead, non coherent systems require frequency synchronization. Frequency synchronization differs from phase synchronization in that the replica of the carrier that is generated by the receiver is allowed to have an arbitrary constant phase offset from the received carrier. Receiver designs can be simplified by removing the requirement to determine the exact value of the incoming carrier phase. Unfortunately, as is shown in the discussion of modulation techniques, this simplification carries a penalty in terms of degraded performance versus signal-to-noise ratio. The relative trade-offs of synchronization levels versus performance and system versatility are discussed further in the next section.

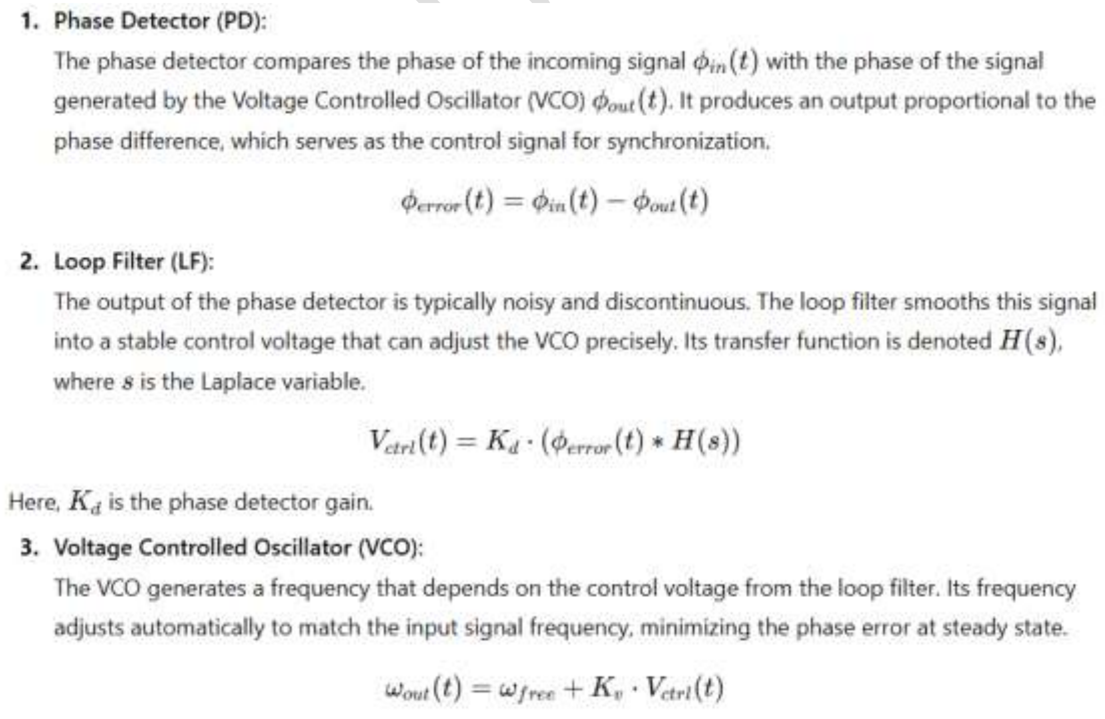
All of the discussion thus far had been oriented toward the receiving end of a communication link. There are instances, however, when the transmitter assumes the more active role in synchronization-by varying the timing and frequency of its transmissions to correspond to the expectations of the receiver. An example of this situation is a satellite communication network, where many terrestrial terminals are beaming signals toward a single satellite receiver. In most of these cases the transmitter relies on a return path from the receiver to determine the accuracy of its synchronization. Thus, transmitter synchronization often implies two-way communications or a network in order to be successful. Thus, transmitter synchronization is often called network synchronization. Transmitter or network synchronization is discussed later in this chapter.

3.2 Phase Locked Loop (PLL) Recovery – Signal Recovery Using PLL

A **Phase Locked Loop (PLL)** is an advanced electronic circuit used to synchronize an electrical signal with a reference signal in terms of **frequency and phase**. It is widely used in modern digital and communication systems to recover lost signals or restore timing (clock recovery) from received data without requiring an explicit clock reference.

How a PLL Works

A PLL consists of three main components that work together to achieve precise synchronization between the input signal and the generated signal:



1. **Phase Detector (PD):**
   The phase detector compares the phase of the incoming signal $\phi_{in}(t)$ with the phase of the signal generated by the Voltage Controlled Oscillator (VCO) $\phi_{out}(t)$. It produces an output proportional to the phase difference, which serves as the control signal for synchronization.

$$\phi_{error}(t) = \phi_{in}(t) - \phi_{out}(t)$$

2. **Loop Filter (LF):**
   The output of the phase detector is typically noisy and discontinuous. The loop filter smooths this signal into a stable control voltage that can adjust the VCO precisely. Its transfer function is denoted $H(s)$, where s is the Laplace variable.

$$V_{ctrl}(t) = K_d \cdot (\phi_{error}(t) * H(s))$$

Here, K_d is the phase detector gain.

3. **Voltage Controlled Oscillator (VCO):**
   The VCO generates a frequency that depends on the control voltage from the loop filter. Its frequency adjusts automatically to match the input signal frequency, minimizing the phase error at steady state.

$$\omega_{out}(t) = \omega_{free} + K_v \cdot V_{ctrl}(t)$$

Where:

- K_v is the VCO sensitivity in $(\mathrm{rad/s})/V$
- ω_{free} is the free-running frequency of the VCO

At steady state (locked condition):

$$\omega_{out} \approx \omega_{in}, \quad \phi_{error} \approx 0$$

Digital Signal Recovery (Clock Recovery)

In digital communication systems, data is often transmitted **without an explicit clock signal**. A PLL is used to **recover the timing of the data**:

Example: Suppose we have NRZ binary data with a bit rate f_b=1 MHz:

1. The PD compares the incoming data signal with the VCO output, calculating the phase difference.
2. The loop filter converts the phase difference into a smooth control voltage.
3. The VCO adjusts its frequency to match the incoming data rate.

After a short time, the PLL locks, and the VCO output frequency matches the input data frequency, effectively **recovering the original clock signal**:

$$\omega_{VCO} \approx \omega_{data} = 2\pi \cdot 1\,\mathrm{MHz}$$

This allows the system to correctly sample and interpret the incoming digital data without any external clock reference.

Practical Applications

PLL recovery is critical in:

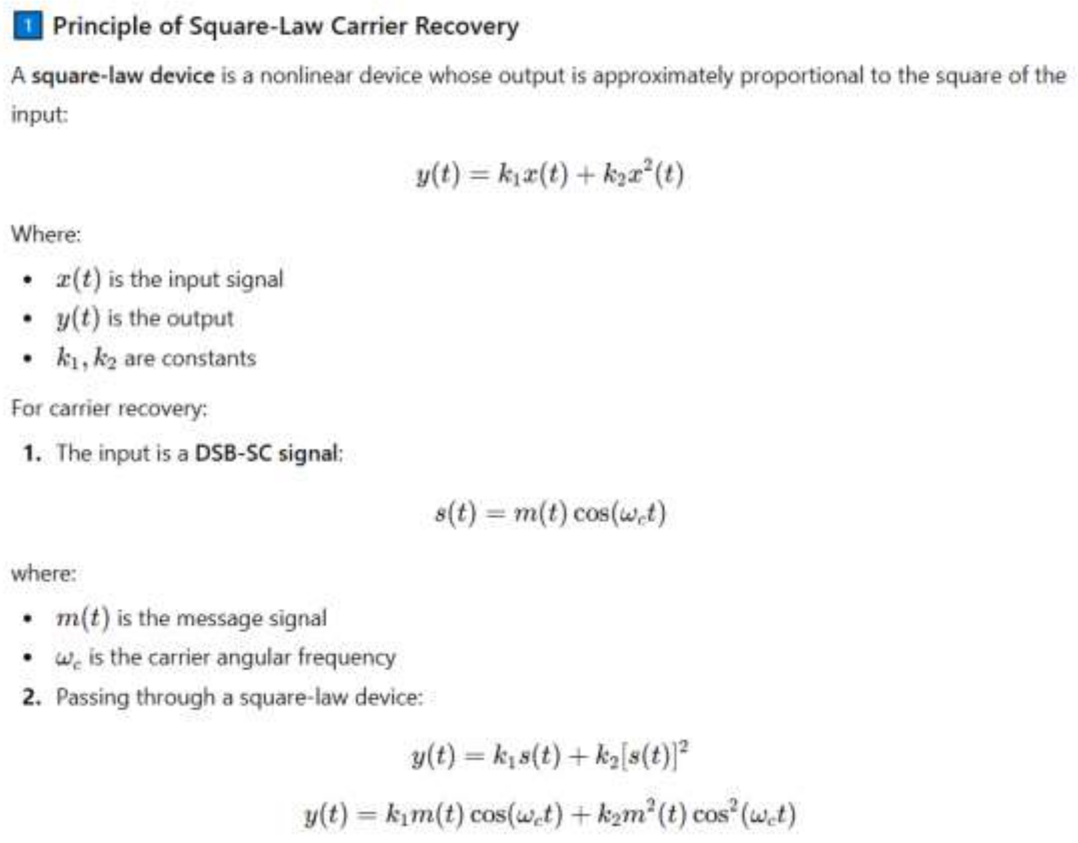
- **Clock recovery** from digital communication signals such as Ethernet or DSL.
- **Transmitter-receiver synchronization** in radio and radar systems.
- **Stable frequency generation** in wireless and communication systems.

Practical Example:
To recover a clock from 1 MHz NRZ data:

- Connect the data to a PD along with the VCO output.
- Pass the phase error through a simple RC loop filter.
- The VCO converts the filtered voltage into a frequency matching the incoming data, providing a recovered clock perfectly synchronized with the received data.

3.3 Carrier Recovery Using a Square-Law Device

In communication systems, especially **Amplitude Modulated (AM)** or **Double-Sideband Suppressed Carrier (DSB-SC)** systems, the receiver often needs to **regenerate the carrier** for proper demodulation. When the carrier is suppressed or unknown, **carrier recovery** circuits are required. One simple and widely studied method uses a **Square-Law Device (SLD)**.



1 Principle of Square-Law Carrier Recovery

A **square-law device** is a nonlinear device whose output is approximately proportional to the square of the input:

$$y(t) = k_1 x(t) + k_2 x^2(t)$$

Where:

- $x(t)$ is the input signal
- $y(t)$ is the output
- k_1, k_2 are constants

For carrier recovery:

1. The input is a **DSB-SC signal**:

$$s(t) = m(t)\cos(\omega_c t)$$

where:

- $m(t)$ is the message signal
- ω_c is the carrier angular frequency

2. Passing through a square-law device:

$$y(t) = k_1 s(t) + k_2 [s(t)]^2$$

$$y(t) = k_1 m(t)\cos(\omega_c t) + k_2 m^2(t)\cos^2(\omega_c t)$$

Using the identity $\cos^2(\omega_c t) = \frac{1}{2} + \frac{1}{2}\cos(2\omega_c t)$:

$$y(t) = k_1 m(t)\cos(\omega_c t) + \frac{k_2}{2} m^2(t) + \frac{k_2}{2} m^2(t)\cos(2\omega_c t)$$

2 Extracting the Carrier

- The term $\frac{k_2}{2} m^2(t)\cos(2\omega_c t)$ is at **double the carrier frequency** $(2\omega_c)$
- The term $\frac{k_2}{2} m^2(t)$ is **DC and low-frequency components**
- There is **no term directly at** ω_c.

To **regenerate the carrier**:

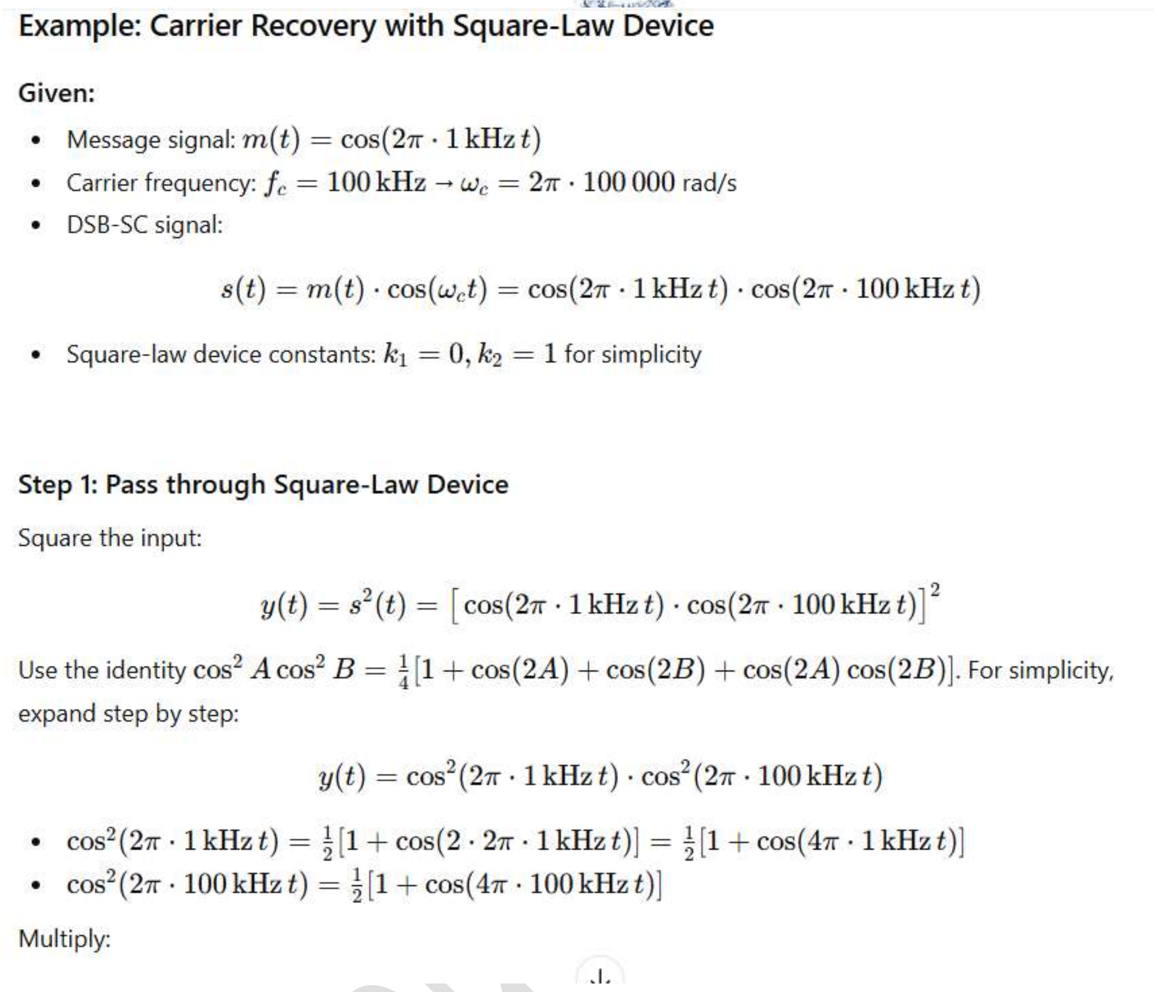
1. Pass $y(t)$ through a **narrow band-pass filter centered at** ω_c.
2. The filtered signal approximates the carrier phase:

$$\hat{c}(t) \approx A_c \cos(\omega_c t)$$

This recovered carrier can then be used in a **coherent demodulator** to extract $m(t)$ from $s(t)$.

Important Notes

- The **square-law device** is suitable for **low-frequency messages** where $m^2(t)$ is small compared to carrier.
- The recovered carrier is not perfect; **phase error** may exist depending on the symmetry of the message signal.
- This technique is simpler than PLL-based carrier recovery but is mainly used for **DSB-SC** and **AM** systems with **low modulation indices**.



Example: Carrier Recovery with Square-Law Device

Given:

- Message signal: $m(t) = \cos(2\pi \cdot 1\,\text{kHz}\,t)$
- Carrier frequency: $f_c = 100\,\text{kHz} \rightarrow \omega_c = 2\pi \cdot 100\,000$ rad/s
- DSB-SC signal:

$$s(t) = m(t) \cdot \cos(\omega_c t) = \cos(2\pi \cdot 1\,\text{kHz}\,t) \cdot \cos(2\pi \cdot 100\,\text{kHz}\,t)$$

- Square-law device constants: $k_1 = 0, k_2 = 1$ for simplicity

Step 1: Pass through Square-Law Device

Square the input:

$$y(t) = s^2(t) = \left[\cos(2\pi \cdot 1\,\text{kHz}\,t) \cdot \cos(2\pi \cdot 100\,\text{kHz}\,t)\right]^2$$

Use the identity $\cos^2 A \cos^2 B = \frac{1}{4}[1 + \cos(2A) + \cos(2B) + \cos(2A)\cos(2B)]$. For simplicity, expand step by step:

$$y(t) = \cos^2(2\pi \cdot 1\,\text{kHz}\,t) \cdot \cos^2(2\pi \cdot 100\,\text{kHz}\,t)$$

- $\cos^2(2\pi \cdot 1\,\text{kHz}\,t) = \frac{1}{2}[1 + \cos(2 \cdot 2\pi \cdot 1\,\text{kHz}\,t)] = \frac{1}{2}[1 + \cos(4\pi \cdot 1\,\text{kHz}\,t)]$
- $\cos^2(2\pi \cdot 100\,\text{kHz}\,t) = \frac{1}{2}[1 + \cos(4\pi \cdot 100\,\text{kHz}\,t)]$

Multiply:

$$y(t) = \frac{1}{4}[1 + \cos(4\pi \cdot 1\,\mathrm{kHz}\,t)][1 + \cos(4\pi \cdot 100\,\mathrm{kHz}\,t)]$$

Expand:

$$y(t) = \frac{1}{4}\left[1 + \cos(4\pi \cdot 1\,\mathrm{kHz}\,t) + \cos(4\pi \cdot 100\,\mathrm{kHz}\,t) + \cos(4\pi \cdot 1\,\mathrm{kHz}\,t)\cos(4\pi \cdot 100\,\mathrm{kHz}\,t)\right]$$

- The last term can be rewritten using $\cos A \cos B = \frac{1}{2}[\cos(A - B) + \cos(A + B)]$:

$$\cos(4\pi \cdot 1\,\mathrm{kHz}\,t)\cos(4\pi \cdot 100\,\mathrm{kHz}\,t) = \frac{1}{2}[\cos(4\pi \cdot 99\,\mathrm{kHz}\,t) + \cos(4\pi \cdot 101\,\mathrm{kHz}\,t)]$$

So the output contains frequencies:

- DC component → 0 Hz
- 2 kHz → $2 \cdot f_m$
- 200 kHz → $2 \cdot f_c$
- 198 kHz and 202 kHz → $2f_c \pm 2f_m$

Step 2: Band-Pass Filter at Carrier Frequency

We want to **recover the carrier**, so we use a **narrow band-pass filter** centered at $f_c = 100\,\mathrm{kHz}$.

- After filtering, only the component at f_c survives.
- Due to the square-law device, the output will be **proportional to cos(ω_c t)** with possibly a small DC offset.

$$\hat{c}(t) \approx A_c \cos(2\pi \cdot 100\,\mathrm{kHz}\,t)$$

This is our **recovered carrier**, now suitable for **coherent demodulation** of the original DSB-SC signal.

Step 3: Coherent Demodulation

Multiply the received DSB-SC signal with the recovered carrier:

$$s(t) \cdot \hat{c}(t) = \cos(2\pi \cdot 1\,\mathrm{kHz}\,t)\cos(2\pi \cdot 100\,\mathrm{kHz}\,t) \cdot \cos(2\pi \cdot 100\,\mathrm{kHz}\,t)$$

Use $\cos^2(\theta) = \frac{1}{2}[1 + \cos(2\theta)]$:

$$s(t) \cdot \hat{c}(t) = \frac{1}{2}\cos(2\pi \cdot 1\,\mathrm{kHz}\,t)[1 + \cos(4\pi \cdot 100\,\mathrm{kHz}\,t)]$$

- Low-pass filtering removes the high-frequency component ($200\,\mathrm{kHz}$)
- Output:

$$m(t) = \frac{1}{2}\cos(2\pi \cdot 1\,\mathrm{kHz}\,t)$$

3.4 Costas Loop

Definition

A **Costas Loop** is a specialized type of **Phase-Locked Loop (PLL)** used primarily for **carrier recovery in suppressed-carrier modulation systems**, such as **BPSK, QPSK, and other PSK signals**. Unlike a simple PLL, the Costas Loop can recover both the **frequency and phase of a carrier** from a modulated signal where the carrier is not explicitly transmitted.

Principle of Operation

1. Input signal (BPSK example):

$$s(t) = m(t) \cdot \cos(\omega_c t)$$

Where $m(t) = \pm 1$ is the BPSK data.

2. The loop splits the incoming signal into **in-phase (I)** and **quadrature (Q)** paths using a **90° phase shift**.

$$I(t) = s(t) \cdot \cos(\hat{\omega}_c t)$$

$$Q(t) = s(t) \cdot \sin(\hat{\omega}_c t)$$

Here, $\hat{\omega}_c$ is the local oscillator frequency generated by the loop's VCO.

3. Multiply $I(t)$ by $Q(t)$ to generate an **error signal**:

$$e(t) = I(t) \cdot Q(t) \approx m^2(t) \sin(\phi_e)$$

Where $\phi_e = \phi_{in} - \phi_{VCO}$ is the phase error.

4. The error signal passes through a **loop filter**, producing a control voltage to adjust the **VCO**.

Mathematical Representation

For BPSK:

$$s(t) = m(t) \cos(\omega_c t + \phi)$$

Costas Loop Error Signal:

$$e(t) = I(t) \cdot Q(t) = m(t)\cos(\omega_c t + \phi) \cdot m(t)\sin(\omega_c t + \phi) = m^2(t)\frac{\sin(2(\omega_c t + \phi))}{2}$$

- After low-pass filtering (removing $2\omega_c$ component):

$$e_{LPF}(t) \approx m^2(t)\sin(2\phi) \approx \sin(2\phi)$$

- This drives the **VCO** toward the correct phase:

$$\omega_{VCO}(t) = \omega_{free} + K_v \cdot e_{LPF}(t)$$

Applications

- Carrier recovery for **BPSK, QPSK**
- Coherent demodulation in **satellite communications**
- Digital radio receivers

Decision-Feedback PLL (DF-PLL)

Definition

A **Decision-Feedback PLL (DF-PLL)** is used in digital communications for **clock or phase recovery** when the received signal contains **high-speed data transitions**. Unlike classical PLLs, it uses **decisions on the received data** to generate the phase error, improving performance in **low SNR** or **high-data-rate scenarios**.

Principle of Operation

1. **Input signal:** NRZ or PSK data r(t)r(t)r(t)
2. **Decision device:** Compares the incoming symbol with a threshold to determine the most likely transmitted bit.
3. **Feedback generation:** The estimated bit is used to compute the **phase error** between the received signal and the local oscillator.
4. **Loop filter:** Smooths the phase error to control the **VCO**.

Mathematical Description

For BPSK:

$$r(t) = m(t)\cos(\omega_c t + \phi) + n(t)$$

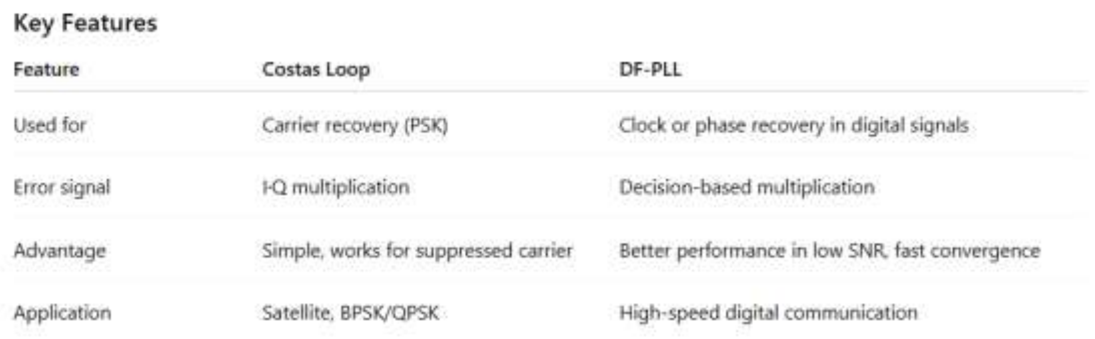
- Decision: $\hat{m}(t) = \text{sign}[r(t)]$
- Phase error:

$$\phi_e(t) = r(t) \cdot \hat{m}(t)$$

- Loop update:

$$\omega_{VCO}(t) = \omega_{free} + K_v \cdot \phi_e(t)$$

- The loop converges faster and is less sensitive to amplitude noise, because the **decision eliminates some data-dependent fluctuations**.



Key Features

Feature	Costas Loop	DF-PLL
Used for	Carrier recovery (PSK)	Clock or phase recovery in digital signals
Error signal	I-Q multiplication	Decision-based multiplication
Advantage	Simple, works for suppressed carrier	Better performance in low SNR, fast convergence
Application	Satellite, BPSK/QPSK	High-speed digital communication

Example: BPSK Costas Loop Carrier Recovery

Given:

- BPSK data: $m(t) = \pm 1$
- Bit rate: $f_b = 1\,\text{kHz}$
- Carrier frequency: $f_c = 100\,\text{kHz} \to \omega_c = 2\pi \cdot 100\,000$ rad/s
- Initial VCO frequency: $f_{VCO0} = 99.5\,\text{kHz}$
- VCO sensitivity: $K_v = 2\pi \cdot 10^6$ rad/s/V
- Loop filter: simple RC low-pass filter (time constant $\tau = 1\,\mu s$)

Step 1: BPSK Signal

Input BPSK signal:

$$s(t) = m(t) \cdot \cos(\omega_c t)$$

Example: if the first bit $m_1 = +1$ (first 1 ms),

$$s(t) = \cos(2\pi \cdot 100\,\text{kHz}\,t), \quad 0 < t < 1\text{ms}$$

Step 2: Split Into I and Q Paths

- In-phase: $I(t) = s(t) \cdot \cos(\hat{\omega}_c t)$
- Quadrature: $Q(t) = s(t) \cdot \sin(\hat{\omega}_c t)$

Assume initial VCO frequency slightly off: $f_{VCO0} = 99.5\,\text{kHz}$,
$\hat{\omega}_c = 2\pi \cdot 99.5\,\text{kHz}$

$$I(t) \approx \cos(2\pi \cdot 100\,\text{kHz}\,t) \cdot \cos(2\pi \cdot 99.5\,\text{kHz}\,t)$$

$$Q(t) \approx \cos(2\pi \cdot 100\,\text{kHz}\,t) \cdot \sin(2\pi \cdot 99.5\,\text{kHz}\,t)$$

Use trig identities:

$$I(t) = \frac{1}{2}[\cos(2\pi \cdot 0.5\,\text{kHz}\,t) + \cos(2\pi \cdot 199.5\,\text{kHz}\,t)]$$

$$Q(t) = \frac{1}{2}[\sin(2\pi \cdot 0.5\,\text{kHz}\,t) + \sin(2\pi \cdot 199.5\,\text{kHz}\,t)]$$

- **Low-pass filter** removes high-frequency term $\approx 199.5\text{kHz}$
- Remaining signals:

$$I_{LPF}(t) \approx \frac{1}{2}\cos(2\pi \cdot 0.5\,\text{kHz}\,t)$$

$$Q_{LPF}(t) \approx \frac{1}{2}\sin(2\pi \cdot 0.5\,\text{kHz}\,t)$$

Step 3: Error Signal Generation

Costas Loop multiplies I and Q to produce **phase error**:

$$e(t) = I_{LPF}(t) \cdot Q_{LPF}(t) = \frac{1}{4}\cos(2\pi \cdot 0.5\,\text{kHz}\,t) \cdot \sin(2\pi \cdot 0.5\,\text{kHz}\,t)$$

$$e(t) = \frac{1}{8}\sin(2\pi \cdot 1\,\text{kHz}\,t)$$

- The error signal oscillates at the **frequency difference between carrier and VCO** ($f_c - f_{VCO0} = 0.5\,\text{kHz}$)

Step 4: Loop Filter and VCO Update

- Low-pass filter smooths $e(t)$, producing a **control voltage** V_{ctrl}
- VCO frequency update:

$$\omega_{VCO}(t + \Delta t) = \omega_{VCO}(t) + K_v \cdot V_{ctrl}(t)$$

- Over a few milliseconds, f_{VCO} converges to **exact carrier frequency** 100 kHz
- Phase error reduces to nearly 0 → **carrier is locked**

Step 5: Coherent Demodulation

- Multiply BPSK signal with recovered carrier:

$$r(t) \cdot \cos(\omega_c t) \approx m(t)\cos^2(\omega_c t) = \frac{m(t)}{2}[1 + \cos(2\omega_c t)]$$

- Low-pass filtering removes $2f_c$ term → **recovered message**:

$$\hat{m}(t) = \frac{1}{2}m(t)$$

Example: DF-PLL for BPSK/NRZ Clock Recovery

Given:

- NRZ/BPSK data: $m(t) = \pm 1$
- Bit rate: $f_b = 1\,\text{kHz}$
- Carrier frequency: $f_c = 100\,\text{kHz} \rightarrow \omega_c = 2\pi \cdot 100\,000$ rad/s
- Initial VCO frequency: $f_{VCO0} = 99.5\,\text{kHz}$
- VCO sensitivity: $K_v = 2\pi \cdot 10^6$ rad/s/V
- Loop filter: simple RC low-pass filter, $\tau = 1\,\mu s$

Step 1: Received Signal

BPSK signal:

$$r(t) = m(t) \cdot \cos(\omega_c t)$$

Example: first bit $m_1 = +1$ (0–1 ms):

$$r(t) = \cos(2\pi \cdot 100\,\text{kHz}\,t)$$

Step 2: Make a Decision

- The **decision device** estimates the transmitted bit based on the received signal sampled at the bit center:

$$\hat{m}(t) = \text{sign}[r(t) \cdot \cos(\hat{\omega}_c t)]$$

- Initial VCO slightly off: $f_{VCO0} = 99.5\,\text{kHz}$ → phase not aligned yet.
- After first bit:

$$r(t) \cdot \cos(\hat{\omega}_c t) \approx \cos(2\pi \cdot 100\,\text{kHz}\,t) \cdot \cos(2\pi \cdot 99.5\,\text{kHz}\,t)$$

Use trig identity:

$$\cos A \cos B = \frac{1}{2}[\cos(A - B) + \cos(A + B)]$$

$$r(t) \cdot \cos(\hat{\omega}_c t) \approx \frac{1}{2}[\cos(2\pi \cdot 0.5\,\text{kHz}\,t) + \cos(2\pi \cdot 199.5\,\text{kHz}\,t)]$$

- Low-pass filtering removes high-frequency term →

$$\hat{m}(t) \approx \frac{1}{2}\cos(2\pi \cdot 0.5\,\text{kHz}\,t)$$

- Decision: $\hat{m} = +1$

Step 3: Generate Phase Error Using Feedback

- Phase error is computed using **decision-based feedback**:

$$\phi_e(t) = r(t) \cdot \hat{m}(t)$$

- With $\hat{m} = +1$,

$$\phi_e(t) \approx r(t) \cdot 1 = r(t)$$

- Low-pass filter smooths $\phi_e(t)$ to get control voltage:

$$V_{ctrl}(t) = LPF[\phi_e(t)]$$

Step 4: Update VCO Frequency

$$\omega_{VCO}(t + \Delta t) = \omega_{VCO}(t) + K_v \cdot V_{ctrl}(t)$$

- Initial frequency difference: $f_c - f_{VCO0} = 0.5\,\text{kHz}$
- Control voltage gradually increases VCO frequency toward 100 kHz
- After several bits, VCO locks exactly to carrier → phase error → 0

Step 5: Recover Data / Clock

- Multiply received signal $r(t)$ by **locked VCO** to demodulate:

$$r(t) \cdot \cos(\omega_c t) = m(t)\cos^2(\omega_c t) = \frac{m(t)}{2}[1 + \cos(2\omega_c t)]$$

- Low-pass filter removes $2f_c$ term → recovered NRZ/BPSK data:

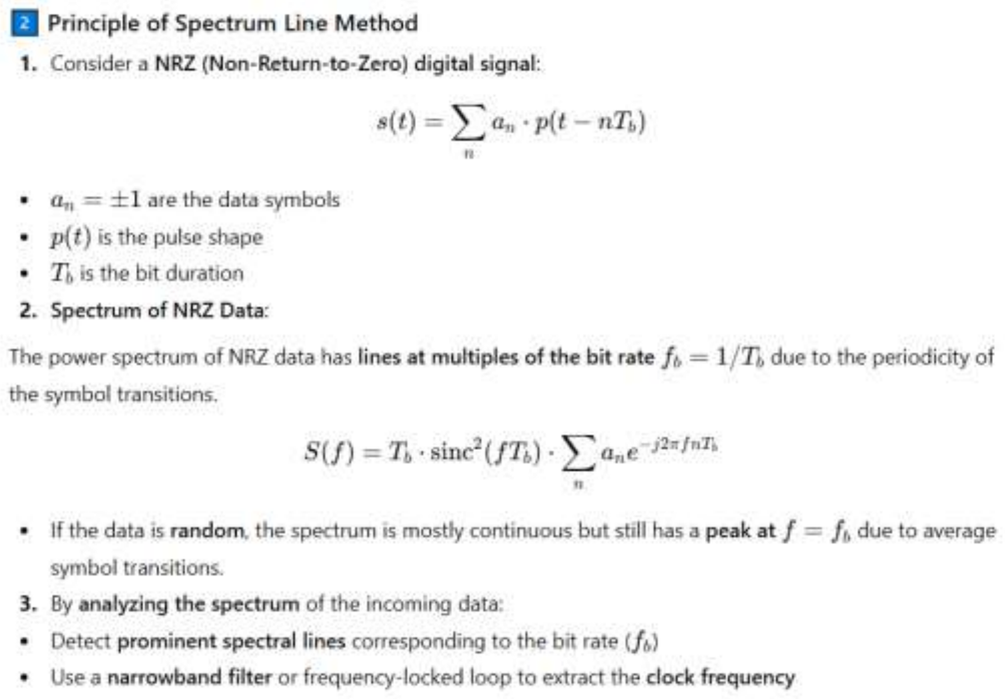
$$\hat{m}(t) = \frac{1}{2}m(t)$$

Clock Recovery – Spectrum Line Method

1- Introduction

In digital communication systems, the receiver must extract timing information from a received **data stream** in order to sample symbols accurately. This is called **clock recovery**.

The **Spectrum Line Method** is a technique that uses **frequency domain analysis** of the received data to recover the clock by detecting **spectral components corresponding to the data rate**.



2 Principle of Spectrum Line Method

1. Consider a **NRZ (Non-Return-to-Zero) digital signal**:

$$s(t) = \sum_n a_n \cdot p(t - nT_b)$$

- $a_n = \pm 1$ are the data symbols
- $p(t)$ is the pulse shape
- T_b is the bit duration

2. **Spectrum of NRZ Data**:

The power spectrum of NRZ data has **lines at multiples of the bit rate** $f_b = 1/T_b$ due to the periodicity of the symbol transitions.

$$S(f) = T_b \cdot \text{sinc}^2(fT_b) \cdot \sum_n a_n e^{-j2\pi f n T_b}$$

- If the data is **random**, the spectrum is mostly continuous but still has a **peak at** $f = f_b$ due to average symbol transitions.

3. By **analyzing the spectrum** of the incoming data:

- Detect **prominent spectral lines** corresponding to the bit rate (f_b)
- Use a **narrowband filter** or frequency-locked loop to extract the **clock frequency**

3 Implementation Steps

1. **Received Data:**

$$r(t) = \sum_n a_n p(t - nT_b)$$

2. **Perform FFT or use a bandpass filter** at $f \approx f_b$ to detect the **line corresponding to bit transitions.**
3. **Extract Phase Information:**

- The phase of the spectral line corresponds to the **timing of the bits.**
- Adjust a **VCO or clock generator** to match this phase → recovered clock.

4 Example

Suppose we have:

- NRZ signal at $f_b = 1\ \text{kHz}$
- Symbol sequence: $+1, -1, +1, +1, -1, \ldots$
- Pulse shape: rectangular with duration $T_b = 1/f_b = 1\ \text{ms}$

Step 1 – Spectrum Analysis:

- Compute FFT of received signal → spectral peaks observed at:

$$f = 1\,\text{kHz}, 2\,\text{kHz}, 3\,\text{kHz}, \ldots$$

- **Fundamental line at $f_b = 1\,\text{kHz}$** corresponds to **bit rate.**

Step 2 – Bandpass Filter:

- Pass the signal through a narrowband filter centered at 1 kHz → output is approximately sinusoidal:

$$c(t) \approx A\cos(2\pi f_b t + \phi)$$

- This is the **recovered clock.**

Step 3 – Sampling Data:

- Use $c(t)$ to trigger **sample points at each bit center** → recover transmitted symbols.

Key Notes

- Advantage: Simple and works for **periodic or repetitive data patterns**
- Limitation: **Random data reduces spectral line amplitude**, so SNR must be sufficient
- Can be combined with **PLL** for more accurate clock recovery in noisy channels
- Often used in **SONET/SDH, Ethernet, and optical communications**

Example: Clock Recovery – Spectrum Line Method

Given:

- NRZ signal
- Bit rate: $f_b = 1\,\text{kHz} \to T_b = 1\,\text{ms}$
- Symbol sequence: $+1, -1, +1, +1, -1, +1, -1, -1$
- Pulse shape: rectangular pulses of duration T_b

Step 1: Generate NRZ Signal

NRZ waveform can be written as:

$$s(t) = \sum_n a_n p(t - nT_b)$$

For rectangular pulses:

$$p(t) = \begin{cases} 1 & 0 \le t < T_b \\ 0 & \text{otherwise} \end{cases}$$

So for first 4 bits:

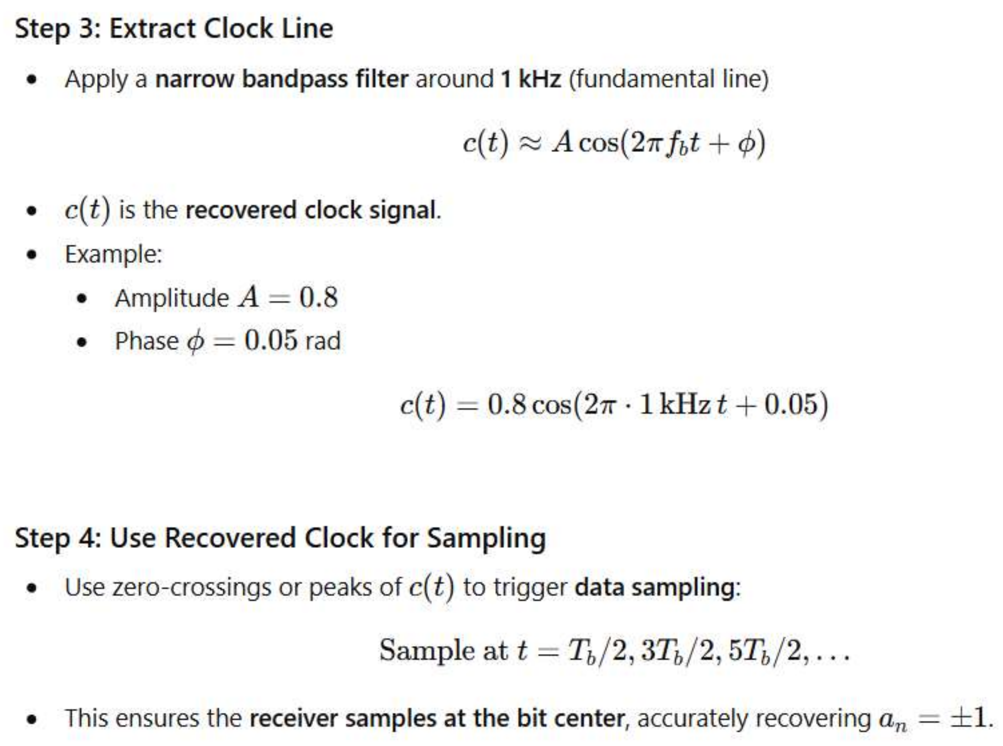
$$s(t) = +1 \text{ for } 0 \le t < 1\,\text{ms}, -1 \text{ for } 1 \le t < 2\,\text{ms}, +1 \text{ for } 2 \le t < 3\,\text{ms}, +1 \text{ for } 3 \le t < 4\,\text{ms}, \ldots$$

Step 2: Compute Spectrum

- The **FFT** of the NRZ signal shows peaks at multiples of the **bit rate** $f_b = 1\,\text{kHz}$.
- Fundamental spectral line at $f = 1\,\text{kHz}$ corresponds to **repetition of bits**.
- Higher harmonics at $2f_b, 3f_b, \ldots$

Numerical example:

Frequency (Hz)	Amplitude
0	0.1
1 kHz	0.8
2 kHz	0.5
3 kHz	0.3
4 kHz	0.2



Step 3: Extract Clock Line

- Apply a **narrow bandpass filter** around **1 kHz** (fundamental line)

$$c(t) \approx A\cos(2\pi f_b t + \phi)$$

- $c(t)$ is the **recovered clock signal**.
- Example:
  - Amplitude $A = 0.8$
  - Phase $\phi = 0.05$ rad

$$c(t) = 0.8\cos(2\pi \cdot 1\,\mathrm{kHz}\,t + 0.05)$$

Step 4: Use Recovered Clock for Sampling

- Use zero-crossings or peaks of $c(t)$ to trigger **data sampling**:

$$\text{Sample at } t = T_b/2, 3T_b/2, 5T_b/2, \ldots$$

- This ensures the **receiver samples at the bit center**, accurately recovering $a_n = \pm 1$.

Key Notes

1. **Fundamental spectral line** corresponds to **bit rate**.
2. **SNR** must be high enough for spectral peak detection.
3. This method is especially useful for **repetitive or synchronous data streams** (like SONET, Ethernet).
4. For **random data**, the spectral line amplitude decreases, so sometimes it is combined with **PLL** for better accuracy.

1 Definition

The **Minimum Mean Square Error (MMSE)** is a statistical criterion used in estimation and signal processing to **find an estimator that minimizes the expected value of the squared error** between the true signal and its estimate.

Formally, if x is the desired signal and $\hat{x}$ is its estimate:

$$\text{MSE} = E\left[(x - \hat{x})^2\right]$$

The MMSE estimator is the one that **minimizes this expected squared error**:

$$\hat{x}_{\text{MMSE}} = \arg\min_{\hat{x}} E\left[(x - \hat{x})^2\right]$$

2 Principle

1. Let x be the desired signal and y be the observed signal (possibly noisy or distorted).
2. We want a function $\hat{x} = f(y)$ that estimates x from y.
3. The **mean square error** is:

$$\text{MSE} = E[(x - f(y))^2]$$

4. **MMSE estimator** chooses $f(y)$ to **minimize MSE**.

3 Mathematical Formulation

If x and y are random variables:

$$\hat{x}_{\text{MMSE}} = E[x|y]$$

- That is, the **conditional expectation of x given y**
- This is the **optimal estimator in the MMSE sense**, achieving the **lowest possible MSE**.

Proof outline:

Let $\hat{x} = g(y)$, then MSE is:

$$E[(x - g(y))^2] = E[E[(x - g(y))^2|y]]$$

- For fixed y, the inner expectation is minimized by $g(y) = E[x|y]$.
- Hence, **MMSE estimator** = **conditional mean**.

4 Example: Linear MMSE Estimator

Suppose we have a **linear model**:

$$y = hx + n$$

- x is zero-mean with variance σ_x^2
- n is zero-mean noise with variance σ_n^2, independent of x

We want **linear estimator**:

$$\hat{x} = ky$$

Step 1 – Mean square error:

$$\text{MSE} = E[(x-ky)^2] = E[(x-k(hx+n))^2]$$
$$= E[(x-khx-kn)^2] = E[((1-kh)x-kn)^2]$$
$$= (1-kh)^2\sigma_x^2 + k^2\sigma_n^2$$

Step 2 – Minimize MSE w.r.t k:

$$\frac{d}{dk}\text{MSE} = -2h\sigma_x^2(1-kh) + 2k\sigma_n^2 = 0$$
$$-2h\sigma_x^2 + 2kh^2\sigma_x^2 + 2k\sigma_n^2 = 0$$
$$k(h^2\sigma_x^2 + \sigma_n^2) = h\sigma_x^2$$
$$\boxed{k_{\text{MMSE}} = \frac{h\sigma_x^2}{h^2\sigma_x^2 + \sigma_n^2}}$$

$$\text{MSE}_{\text{min}} = \sigma_x^2 - kh\sigma_x^2 = \frac{\sigma_x^2\sigma_n^2}{h^2\sigma_x^2 + \sigma_n^2}$$

Applications of MMSE

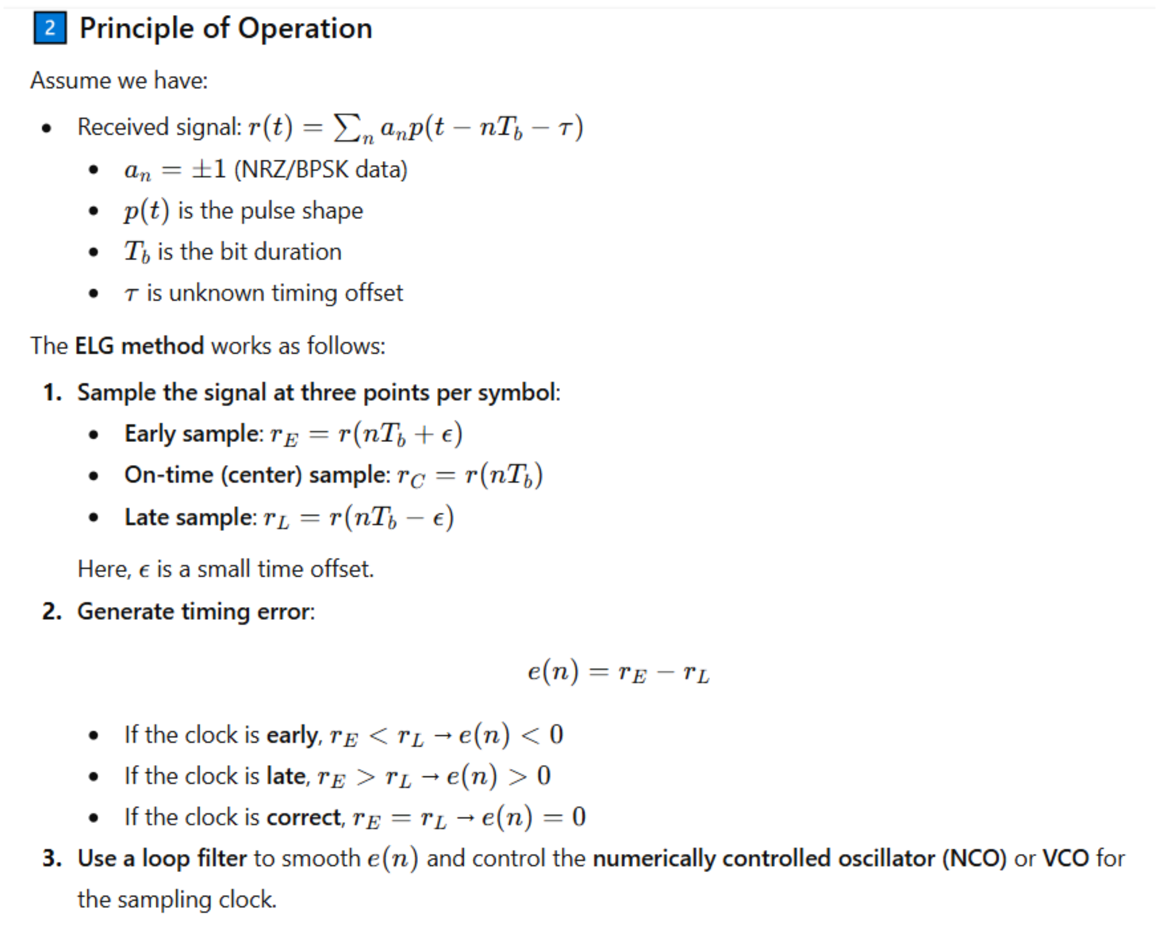
1. **Channel equalization** in digital communication
2. **Adaptive filtering** (Wiener filter, LMS algorithm)
3. **Noise reduction** in signals and images
4. **Data estimation** in wireless communication (MIMO detection)

Introduction

The **Early-Late Gate (ELG) method** is a widely used **timing recovery technique** in digital communication receivers.

- Its main goal is to **synchronize the receiver clock with the transmitter bit timing**.

- It works by **comparing the received signal sampled slightly earlier and slightly later than the expected sampling instant** and generating an **error signal** to adjust the clock phase.



2 Principle of Operation

Assume we have:

- Received signal: $r(t) = \sum_n a_n p(t - nT_b - \tau)$
  - $a_n = \pm 1$ (NRZ/BPSK data)
  - $p(t)$ is the pulse shape
  - T_b is the bit duration
  - τ is unknown timing offset

The **ELG method** works as follows:

1. **Sample the signal at three points per symbol**:
   - **Early sample**: $r_E = r(nT_b + \epsilon)$
   - **On-time (center) sample**: $r_C = r(nT_b)$
   - **Late sample**: $r_L = r(nT_b - \epsilon)$

   Here, ϵ is a small time offset.
2. **Generate timing error**:

$$e(n) = r_E - r_L$$

   - If the clock is **early**, $r_E < r_L \rightarrow e(n) < 0$
   - If the clock is **late**, $r_E > r_L \rightarrow e(n) > 0$
   - If the clock is **correct**, $r_E = r_L \rightarrow e(n) = 0$
3. **Use a loop filter** to smooth $e(n)$ and control the **numerically controlled oscillator (NCO)** or **VCO** for the sampling clock.

2 Principle of Operation

Assume we have:

- Received signal: $r(t) = \sum_n a_n p(t - nT_b - \tau)$
  - $a_n = \pm 1$ (NRZ/BPSK data)
  - $p(t)$ is the pulse shape
  - T_b is the bit duration
  - τ is unknown timing offset

The **ELG method** works as follows:

1. **Sample the signal at three points per symbol**:
   - **Early sample**: $r_E = r(nT_b + \epsilon)$
   - **On-time (center) sample**: $r_C = r(nT_b)$
   - **Late sample**: $r_L = r(nT_b - \epsilon)$

   Here, ϵ is a small time offset.
2. **Generate timing error**:

$$e(n) = r_E - r_L$$

   - If the clock is **early**, $r_E < r_L \rightarrow e(n) < 0$
   - If the clock is **late**, $r_E > r_L \rightarrow e(n) > 0$
   - If the clock is **correct**, $r_E = r_L \rightarrow e(n) = 0$
3. **Use a loop filter** to smooth $e(n)$ and control the **numerically controlled oscillator (NCO)** or **VCO** for the sampling clock.

5 Example

Assume:

- NRZ data: $+1, -1, +1, +1, -1$
- Bit duration: $T_b = 1\,\text{ms}$
- Pulse shape: rectangular

Suppose the receiver samples **slightly early**:

- $r_E = 0.8, r_L = 1.2$
- Timing error:

$$e = r_E - r_L = 0.8 - 1.2 = -0.4$$

- Negative error → sampling clock is **early** → **slow down the clock**

Next symbol:

- $r_E = 1.05, r_L = 0.95$
- Error: $e = +0.1$ → clock is **late** → **speed up the clock**

Over several symbols, the **loop converges**, and the clock locks at the **center of the bit**, minimizing intersymbol interference.

6 Key Features

- Works for **NRZ, RZ, BPSK, QPSK signals**
- Loop converges to **zero error** when sampling at bit center
- **Simple and robust**, widely used in **digital modems, optical communication, and UARTs**

3.2 Costs versus Benefits

There is a cost associated with the need for receiver synchronization. Each additional level of synchronization implies more cost. The most obvious cost is in the need for additional hardware or software in the receiver for acquisition and tracking. Possibly less obvious costs lie in the extra time required to achieve synchronization before commencing communications, or in the energy expended by the transmitter on signals to be used at the receiver as acquisition or tracking aids. In the face of these costs to the system, one might question why a communications system designer would consider a system design requiring a high degree of synchronization. The answer: improved performance and versatility.

Consider a standard commercial analog AM radio. This radio may be considered part of a broadcast communication system involving a central transmitter and many receivers. This communication system involves no synchronization. How- ever, the receiver passband must be wide enough to accommodate not only the information-bearing signal, but also any fluctuations in the carrier, due perhaps to Doppler shift* or drift in the transmitter's frequency reference. This requirement in the receiver passband means that additional noise energy is passed to the detector, over and above the amount theoretically required by the bandwidth of the information. A somewhat more complicated receiver that employs a carrier frequency tracking loop would be able to keep a narrow passband filter centered about the carrier, thereby substantially reducing the detected noise energy and improving the received signal-to-noise ratio. Thus, although a standard radio may be perfectly adequate for reception of signals from large transmitters a few tens of kilometers distant, it may prove totally inadequate under less benign conditions.

For digital communications, examples of the trade-off between performance and receiver complexity are often seen in the choice of modulation. Among the

simplest digital receivers are those designed to be used with non coherently detected binary FSK. The only synchronization requirements are bit timing and frequency tracking. however, the same bit error probability could be achieved with approximately 4 dB less signal-to-noise ratio if the modulation is coherent BPSK. The disadvantage of BPSK is that the receiver requires accurate phase tracking. which can present a complex design problem if the signals experience high Doppler rates** or fading.

A third cost-versus-performance trade-off involves the use of error-control coding. As was established in earlier chapters, there are substantial performance advantages in the use of appropriate error-control coding techniques. The cost. however, measured in receiver complexity, can be high. For a block decoder to operate properly requires the receiver to achieve block, message, or frame synchronism. This is a procedure over and above the usual decoding procedure, although some error-correcting codes have been designed with block synchronization aids built in [1]. Convolutional codes also require some degree of additional synchronization in order to provide optimum performance. Although the performance analysis of convolutional codes often makes the assumption that the input data sequence is infinitely long, in practice it is not. In order to provide the minimum error probability, the decoder must know the beginning state (usually all zeros) when the data sequence will begin, the eventual ending state, and when the ending state is to be reached. Knowing when the beginning state was left and when the ending state is to be reached, however, is equivalent to having frame synchronization. In addition, the decoder will have to know how to group the channel symbols in order to make branch decisions. This is also a synchronization requirement.

The trade-offs discussed thus far have been in terms of the performance versus complexity of individual links and receivers. The ability to synchronize has a large potential consequence in terms of system efficiency and versatility as well. Frame synchronization allows the use of advanced, versatile, multiple-access techniques, such as the variety of demand-assignment-multiple-access (DAMA) schemes, which have become increasingly popular as communication channel re- sources become increasingly scarce. In addition, the use of spread-spectrum techniques---both as multiple access schemes and for interference rejection-requires a high level of system synchronization. (Spread-spectrum techniques are treated It will be seen that these techniques provide the potential for a great deal of system versatility, which is a very valuable feature if the system encounters changing or unstable conditions, such as the effects of intentional and unintentional interference from external sources.

3.3 Approach and Assumptions

There have been at least two substantial developments in the general area of synchronization since the first edition of this text. One has been the emergence, and then near total dominance, of signal processing (including synchronizers) by sam-

The rate of change of the Doppler shift. This rate sets requirements on the tracking ability of the phase tracking loop.

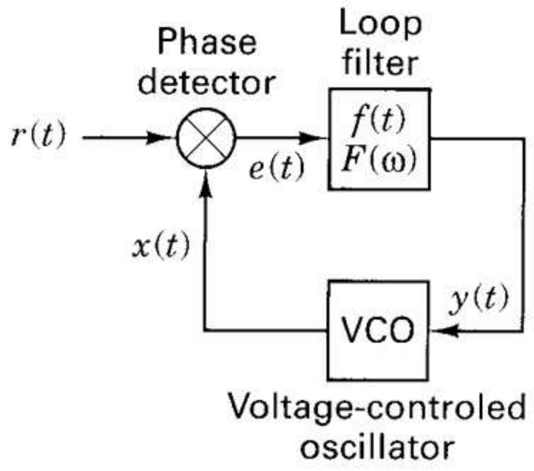
Thus, we will generally follow a traditional analog development knowing that these principles apply equally well to sampled-data systems, even if the implementation of the synchronizers differ. Phase-locked-loops are commercially avail- able as relatively small gate-count chips, or as one part of a larger signal-processing device. It is assumed that the reader interested in modern design implementations will be able to make the reasonable straightforward transition from the principles presented here to the basic sampled data representations.

3.4 RECEIVER SYNCHRONIZATION

All digital communication systems require some degree of synchronization to in- coming signals by the receivers. In this section the fundamentals of the various levels of receiver synchronization are discussed. The discussion begins with the basic levels of synchronization required for coherent reception-frequency and phase synchronization and a brief discussion of the principles of phase-locked-loop (PLL) operation and design. The discussion then broadens into the topic of symbol synchronization. Some degree of symbol synchronization is required for all digital communications reception, either coherent or non coherent. The final topics in the section are receiver frame synchronization and techniques for achieving and maintaining it.

3.5 Frequency and Phase Synchronization

At the heart of nearly all synchronization circuits is some version of a phase-locked-loop (PLL). In modern digital receivers this loop may be difficult to recognize, but the functional equivalent is essentially always present. A schematic diagram of the basic PLL is given in Figure 3.1 Phase-locked loops are servo- control loops, whose controlled parameter is the phase of a locally generated replica of the incoming carrier signal. Phase-locked loops have three basic components: a phase detector, a loop filter, and a voltage-controlled oscillator (VCO). The phase detector is a device that produces a measure of the difference in phase between an incoming signal and the local replica. As the incoming signal and the local replica change with respect to each other, the phase difference (or phase error) becomes a time-varying signal into the loop filter. The loop filter governs the PLL's response to these variations in the error signal. A well-designed loop should be able to track changes in the incoming signal's phase but not be overly responsive to receiver noise. The VCO is the device that produces the carrier replica. The VCO, as the name implies, is a sinusoidal oscillator whose frequency is controlled by a voltage level at the device input. In Figure 3.1 the phase detector is shown as a multiplier, the loop filter is described by its impulse response function f(1), with Fourier transform F(w), and the VCO is so indicated.



Figure(3.1) Schematic of the basic phase – locked loop

A VCO is an oscillator whose output frequency is a linear function of its input voltage over some range of input and output. A positive input voltage will cause the VCO output frequency to be greater than its uncontrolled value, w while a negative voltage will cause it to be less. Phase lock is achieved by feeding a filtered version of the phase difference (i.e., the phase error) between the incoming signal r(t) and the output of the VCO, x(t), back to the input of the VCO, y(t).

In the case of modern digital receivers, the error detector may be mathematically much more complicated than the simple multiplier shown in Figure 3.1. For example, the error detector might be a set of matched-filter correlators, each matched to a slightly different phase offset feeding a weighting or decision function. The output of the weighting function would be the phase error estimate. Such a function might be mathematically very complex, but it would be easily approximated using modern digital technology. The VCO may not appear to be a sinusoidal oscillator, but it may be implemented as a read-only memory whose pointers are controlled by a combination of a clock and the output of the error estimator. The feedback path may not be continuous (as shown in Figure 3.1), but phase corrections may only be applied once per frame, or once per packet, depending on the signal structure. A special header or known sequence of symbols may be inserted into the information stream for the expressed purpose of aiding the estimation process. These obvious differences notwithstanding, the basic principles are still illuminated with the simple model of Figure 3.1

Consider a normalized input signal of the form

$$r(t) = \cos[\omega_0 t + \theta(t)]$$

where ω_0 is the nominal carrier frequency and $\theta(t)$ is a slowly varying phase. Similarly, consider a normalized VCO output of the form

$$x(t) = -2\sin[\omega_0 t + \hat{\theta}(t)]$$

These signals will produce an output error signal at the phase detector output of the form

$$\begin{aligned} e(t) = x(t)r(t) &= 2\sin[\omega_0 t + \hat{\theta}(t)]\cos[\omega_0 t + \theta(t)] \\ &= \sin[\theta(t) - \hat{\theta}(t)] + \sin[2\omega_0 t + \theta(t) + \hat{\theta}(t)] \end{aligned}$$

A low-pass filter provides an error signal that is solely a function of the difference in phases between the input] and the VCO output This is exactly the error signal that is needed. The VCO output frequency is the time derivative of the argument of the sine function If we make the assumption that w, is the uncontrolled frequency of the VCO (the output frequency when the input voltage is zero), we can express the difference in the VCO output frequency from w, as the time differential of the phase term $\hat{\theta}$(t). The output frequency of the VCO is a linear function of the input voltage. Therefore, since an input voltage of zero produces an output frequency of wo, the difference in the output frequency from will be proportional to the value of the input voltage y(t), or

$$\begin{aligned} \Delta\omega(t) &= \frac{d}{dt}[\hat{\theta}(t)] = K_0 y(t) \\ &= K_0 e(t) * f(t) \\ &\approx K_0[\theta(t) - \hat{\theta}(t)] * f(t) \end{aligned}$$

where Aw(t) denotes the frequency difference, the notation indicates the convolution operation (see Appendix A), and the small-angle approximation [i.e., e(t) = sin [Θ (t) - $\hat{\Theta}$(t)] = Θ (t) - $\hat{\Theta}$(t)] The small-angle approximation will be accurate when the output phase error is small (the loop is close to phase lock). This will be the situation when the loop is operating normally.

The factor K, is the gain of the VCO, and f(r) is the loop-filter impulse response. This linear differential equation in $\widehat{\Theta}(t)$ (utilizing the small-angle approximation) is known as the linearized loop equation. It is the single most useful relationship in determining loop behavior during normal operation (where the phase error is small).

3.6 Costas Loops

An important form of a suppressed carrier loop is the Costas loop, shown schematically in Figure 3.2. This loop design is important because it eliminates the square-law device, which can be difficult to implement at carrier frequencies, and replaces it with a multiplier and relatively simple low-pass filters. Although the appearance of the circuits in Figures 3.1 and 3.2 is quite different, their theoretical performance can be shown to be the same . The main remaining implementation problem with Costas loops is that to achieve the theoretically optimum performance, the two low-pass arm filters must be perfectly matched. This can only be approximated in any analog hardware implementation. If the arm filters are implemented digitally, there will be no problem keeping them matched, but the designer will confront the usual sampled data design issues. Thus the decision as to whether to implement a Costas loop or the classical design of Figure 3.1 amounts to a design decision between the difficulty of implementing the squaring device and the difficulty of implementing closely matched arm filters. This design decision will depend on the parameters and requirements of the particular receiving system, and cannot be generalized here.

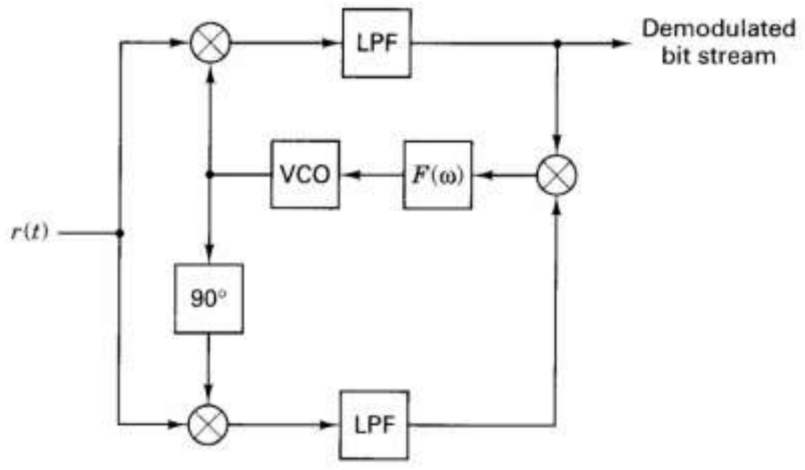


Figure (3.2) Costas Loop