Chapter 3

Synchronization

3.1 Introduction

The word Synchronization comes from *Chronos*, the Greek god of time. *Syn* is a prefix meaning with, along with, together, or at the same time. "To synchronize" thus means to cause one thing to occur or operate with exact coincidence in time or rate as another thing. As applied to digital communications, it usually means the process of causing one oscillator to oscillate with the same frequency and phase as another oscillator.

In the previous chapters, the effects of carrier phase offset and symbol timing offset have been shown. Conceptually, carrier phase synchronization is the process of forcing the local oscillators in the detector to oscillate in phase and frequency with the carrier oscillator used at the transmitter. Symbol timing synchronization is the process of forcing the symbol clock in the receiver to oscillate with the same phase and frequency as the symbol clock used at the transmitter. In either case, the detector must determine the phase and frequency of the oscillator embedded in the received, noisy modulated waveform.

The synchronizers presented in this chapter are all based on the phase-locked loop, or PLL. The fundamentals of PLL operation and analysis are reviewed, in detail, in the Appendix. The results are repeated here for continuity for those already familiar with PLLs.

3.2 Phase-Locked Loops

3.2.1 Continuous-Time Phase Locked Loops

All continuous-time phase-locked loops are characterized by three components: the phase detector, the loop filter, and the voltage controlled oscillator (VCO) arranged as shown in Figure 3.1 (a). PLL performance is usually characterized by how well the PLL tracks the phase of a sinuosoid. In this case, the input to the PLL is a sinusoid with radian frequency ω_0 rads/sec and time-varying phase $\theta(t)$. The output of the VCO is a sinusoid with radian frequency ω_0 rads/sec and time-varying phase $\hat{\theta}(t)$ which is an estimate of the input phase $\theta(t)$. The phase detector produces some function $g(\cdot)$ of the phase error $\theta_e(t) = \theta(t) - \hat{\theta}(t)$. The phase detector characteristic is usually non-linear and is characterized by a plot of $g(\theta_e)$ vs. θ_e[1]. The loop filter, characterized by the transfer function $F(s)$, filters the phase detector output and controls the nature of the loop response. The most commonly used loop filter is the "proportional-plus-integrator" filter with transfer function

$$F(s) = k_1 + \frac{k_2}{s}. \tag{3.1}$$

The proportional-plus-integrator filter has a pole at the origin of the s-plane. This pole is required for the loop to track out any frequency offset with zero steady state phase error. The final element in the loop is the voltage controlled oscillator or VCO. The instantaneous VCO frequency is proportional to the input voltage $v(t)$ so that the instantaneous phase is

$$\hat{\theta}(t) = k_0 \int_{-\infty}^{t} v(x)dx \tag{3.2}$$

where k_0 is the constant of proportionality with units radians/volt. This constant is often called the *VCO gain* or *VCO sensitivity*. When placed in the feedback portion of the loop, the instantaneous frequency of the VCO is adjusted to align the phase of the VCO output with the phase of the PLL input.

The block diagram shown in Figure 3.1 (b) is the "phase equivalent" PLL. The phase equivalent PLL is derived from the PLL by replacing the sinusoids in Figure 3.1 (a) by their phases and characterizing each block in terms of its operation on the phase. The phase equivalent PLL is usually what is analyzed when characterizing loop performance. When the phase detector characteristic is a non-linear function of θ_e, the resulting phase equivalent PLL is a non-linear feedback control system. Most non-linear phase detector characteristics are well approximated by $g(\theta_e) \approx k_p\theta_e$

[1]The plot of the phase detector characteristic is often called an "S-curve" since the phase detector characteristic of many commonly used phase detectors resembles an "S" rotated clockwise by 90 degrees.

for $\theta_e \approx 0$ where the constant of proportionality is the slope of the S-curve about the origin. Linearizing the non-linear phase equivalent PLL about the desired operating point $\theta_e \approx 0$ produces the linear feedback control system shown in Figure 3.2. Since this system is a linear system, frequency domain techniques can be used to analyze the loop responses. The most important loop responses are the phase error response $\theta_e(t)$ and the phase estimate response $\hat{\theta}(t)$. The frequency domain transfer functions are

$$G_a(s) = \frac{\Theta_e(s)}{\Theta(s)} = \frac{s^2}{s^2 + k_p k_0 k_1 s + k_p k_0 k_2} \tag{3.3}$$

$$H_a(s) = \frac{\hat{\Theta}(s)}{\Theta(s)} = \frac{k_p k_0 k_1 s + k_p k_0 k_2}{s^2 + k_p k_0 k_1 s + k_p k_0 k_2}. \tag{3.4}$$

The loop transfer function (3.4) is that of a second-order system and is of the form

$$H_a(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.5}$$

where ω_n is the natural frequency and ζ is the damping factor[2]. Equating the denominators of (3.4) and (3.5) gives the following relationships for the loop constants:

$$\begin{aligned} k_p k_0 k_1 &= 2\zeta\omega_n \\ k_p k_0 k_2 &= \omega_n^2. \end{aligned} \tag{3.6}$$

Given a desired loop response characterized by ζ and ω_n, the loop constants k_p, k_0, k_1, and k_2 are selected to satisfy the relationships (3.6). In practice, PLL responses are characterized by ζ and the equivalent noise bandwidth B_n. The equivalent noise bandwidth of a linear system is defined as the bandwidth of an ideal low-pass filter whose output power due to a white noise input is equal to the output power of the linear system due to the same white noise input. Expressed mathematically, this relationship is

$$B_n = \frac{|H_a(0)|^2}{2} \int_{-\infty}^{\infty} |H_a(j2\pi f)|^2 \, df. \tag{3.7}$$

Using the transfer function (3.4) based on the proportional-plus-integrator loop filter (3.1), the equivalent noise bandwidth (3.7) evaluates to

$$B_n = \frac{\omega_n}{2}\left(\zeta + \frac{1}{4\zeta}\right). \tag{3.8}$$

[2]The damping factor ζ controls the nature of the loop response. When $\zeta < 1$, the loop response is *underdamped*: the poles are complex conjugates the the time-domain response is an exponentially damped sinusoid. When $\zeta = 1$, the loop response in *critically damped*: the poles are real and repeated. When $\zeta > 1$, the loop is *overdamped*: the poles are real and distinct and the loop response is the sum of decaying exponentials.

The relationships (3.6) may be expressed in terms of B_n and the damping factor ζ as

$$\begin{aligned} k_p k_0 k_1 &= \frac{4\zeta B_n}{\zeta + \dfrac{1}{4\zeta}} \\ k_p k_0 k_2 &= \frac{4B_n^2}{\left(\zeta + \dfrac{1}{4\zeta}\right)^2} \end{aligned} \tag{3.9}$$

PLL performance is often characterized by the *acquisition time* and *tracking performance*. The acquisition time is the time required for the PLL to go from an initial frequency and/or phase offset to phase lock. A PLL requires a non-zero period of time for to reduce the frequency error to zero. Once frequency lock is achieved, an additional period is required to reduce the loop phase error to an acceptable level. Thus the acquisition time T_{LOCK} is the time to achieve frequency lock T_{FL} plus the time to achieve phase lock T_{PL}. For a second order PLL, these lock times are well approximated by

$$T_{\text{FL}} \approx 4\frac{(\Delta f)^2}{B_n^3} \tag{3.10}$$

$$T_{\text{PL}} \approx \frac{1.3}{B_n} \tag{3.11}$$

where Δf is the frequency offset. The frequency offset cannot be arbitrarily large. If Δf is too big, then the PLL will not be able to lock. As long as the frequency offset satisfies

$$\Delta f \leq \left(2\pi\sqrt{2}\zeta\right) B_n \approx 6B_n \tag{3.12}$$

the PLL will eventually lock. This characteristic places an upper limit on the frequency offset the PLL is able to handle. This upper limit is called the *pull-in range*.

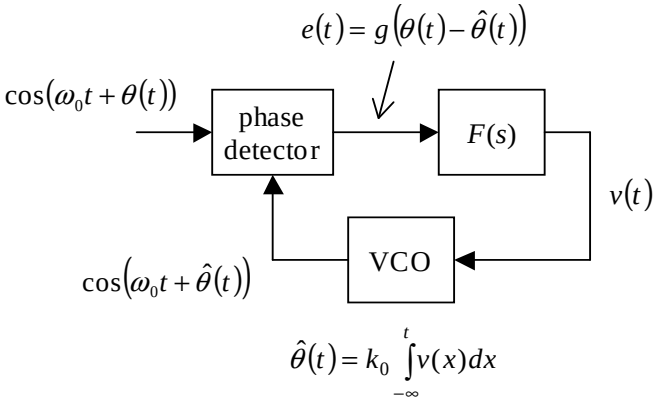
Tracking performance is quantified by the variance of the phase error. Conceptually, the phase error variance, $\sigma_{\theta_e}^2$, is

$$\sigma_{\theta_e}^2 = \text{E}\left\{\left|\theta - \hat{\theta}\right|\right\}. \tag{3.13}$$

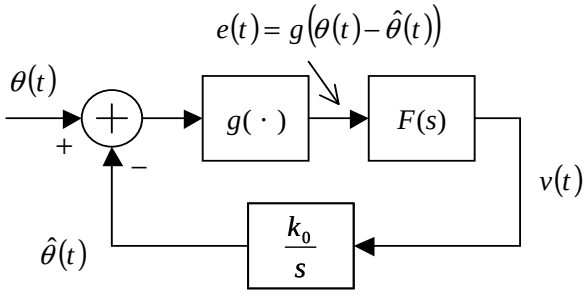
A linear PLL which has a sinusoidal input with power P_{in} W together with additive white Gaussian noise with power spectral density $N_0/2$ W/Hz, the phase error variance is

$$\sigma_{\theta_e}^2 = \frac{N_0 B_n}{P_{\text{in}}}. \tag{3.14}$$

Since the noise power at the PLL input (within the frequency band of interest to the PLL) is $N_0 B_n$, the ratio $P_{\text{in}}/N_0 B_n$ often called the loop signal to noise ratio. Thus, for a linear PLL with additive



(a)



(b)

Figure 3.1: Basic PLL configuration: (a) The three basic components of a PLL. (b) The corresponding phase equivalent PLL

white Gaussian noise, the phase error variance is inversely proportional to the loop signal to noise ratio.

Equations (3.10) and (3.11) indicate that acquisition time is inversely proportional to a power of B_n. This suggests that the larger equivalent loop bandwidth, the faster the acquisition. Equation (3.14) shows that the tracking error is proportional to B_n. This suggests that the smaller the equivalent loop bandwidth, the smaller the tracking error. Thus fast acquisition and good tracking place competing demands on PLL design. Acquisition time can be decreased at the expense of increased tracking error. Tracking error can be decreased at the expense of increased acquisition time. A good design balances the two performance criteria. Where that balance is depends on the application, the signal to noise level, and system-level performance specifications.

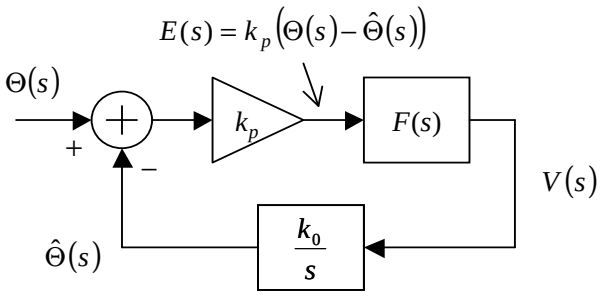


Figure 3.2: Linearized phase equivalent PLL corresponding to the PLL in Figure 3.1 (b).

3.2.2 Discrete-Time Phase-Locked Loops

A discrete-time phase-locked loop is illustrated in Figure 3.3 (a). Just like the continuous-time PLL of Figure 3.1 (a), the discrete-time PLL consists of three elements: a discrete-time phase detector, a discrete-time loop filter, and a direct-digital synthesizer[3], or DDS. The DDS plays the same role in the discrete-time PLL as the VCO did in the continuous-time PLL. The input to the discrete-time PLL are T-spaced samples of a sinusoid with frequency $\Omega_0 = \omega_0 T$ radians/sample and with time-varying phase $\theta(nT)$ where T is the sample time. The output of the DDS is a sinusoid with frequency Ω_0 radians/sample and time-varying phase $\hat{\theta}(nT)$. The phase of the DDS output is the PLL estimate of the phase of the input sinusoid. The phase detector output is $g\left(\theta_e(nT)\right)$ where $\theta_e(nT) = \theta(nT) - \hat{\theta}(nT)$. The loop filter, characterized by the z-domain transfer function $F(z)$, filters the sequence of phase detector outputs and controls the nature of the loop response. A commonly used loop filter is

$$F(z) = K_1 + \frac{K_2}{1 - z^{-1}} \tag{3.15}$$

where upper case filter constants have been used to distinguish them from their counterparts in the continuous-time PLL. The motivation for this filter structure is that it mimics the proportional-plus-integrator loop filter used in continuous-time PLLs. The instantaneous frequency of the DDS is proportional to the DDS input $v(nT)$. As such the instantaneous phase of the DDS output is given by

$$\hat{\theta}(nT) = K_0 \sum_{k=-\infty}^{n-1} v(kT) \tag{3.16}$$

where K_0 is the constant of proportionality. The phase equivalent discrete-time PLL is shown in Figure 3.3 (b). Again, if the phase-detector characteristic is non-linear, then the resulting feedback control system is non-linear. Linearizing about the desired operating point $\theta_e \approx 0$, the phase detector characteristic is replaced by the linear approximation $g(\theta_e) \approx K_p \theta_e$ and the resulting linear phase equivalent discrete-time PLL shown in Figure 3.4 is obtained. Since the feedback control system of Figure 3.4 is linear, frequency domain techniques can be used to analyze the performance. The z-domain transfer function for the phase error and phase estimate are

$$G_d(z) = \frac{\Theta_e(z)}{\Theta(z)} \tag{3.17}$$

$$H_d(z) = \frac{\hat{\Theta}(z)}{\Theta(z)} = \frac{K_0 K_p \left(K_1 + K_2\right) z^{-1} - K_0 K_p K_1 z^{-2}}{1 - 2\left(1 - \frac{1}{2} K_0 K_p \left(K_1 + K_2\right)\right) z^{-1} + \left(1 - K_0 K_p K_1\right) z^{-2}}. \tag{3.18}$$

[3]The DDS is sometimes called a numerically controlled oscillator or NCO.

The loop responses are determined by the loop filter constants K_1 and K_2. Of the many ways the constants could be chosen, one of the most common is to chose the constants to impart on the discrete-time loop, the operating characteristics of the corresponding continuous-time loop. One way to accomplish this to to apply Tustin's Equation (or bilinear transform)

$$\frac{1}{s} = \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}} \tag{3.19}$$

to the transfer function $H_a(s)$ of the continuous-time PLL. The result is

$$H_a\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\right) = \frac{\dfrac{(2\zeta+\theta_n)\,\theta_n}{1+2\zeta\theta_n+\theta_n^2} + 2\dfrac{(\theta_n-\zeta)\,\theta_n}{1+2\zeta\theta_n+\theta_n^2}z^{-1} + \dfrac{\theta_n^2}{1+2\zeta\theta_n+\theta_n^2}z^{-2}}{1+2\dfrac{1-\theta_n^2}{1+2\zeta\theta_n+\theta_n^2}z^{-1} + \dfrac{1-2\zeta\theta_n+\theta_n^2}{1+2\zeta\theta_n+\theta_n^2}z^{-2}} \tag{3.20}$$

where

$$\theta_n = \frac{\omega_n T}{2}. \tag{3.21}$$

Equating the coefficients of z^{-1} and z^{-2} in the denominators of $H_d(z)$ — given by (3.18) — and $H_a\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\right)$ — given by (3.20) — gives the relationship between the filter constants K_1 and K_2 of the discrete-time PLL and the damping factor and natural frequency of the corresponding continuous-time PLL:

$$K_0 K_p K_1 = \frac{4\zeta\theta_n}{1+2\zeta\theta_n+\theta_n^2} \tag{3.22}$$

$$K_0 K_p K_2 = \frac{4\theta_n^2}{1+2\zeta\theta_n+\theta_n^2}. \tag{3.23}$$

Equations (3.22) and (3.23) express the loop filter constants K_1 and K_2 in terms of the desired loop damping factor and natural frequency. Solving (3.8) for ω_n and substituting produces the following

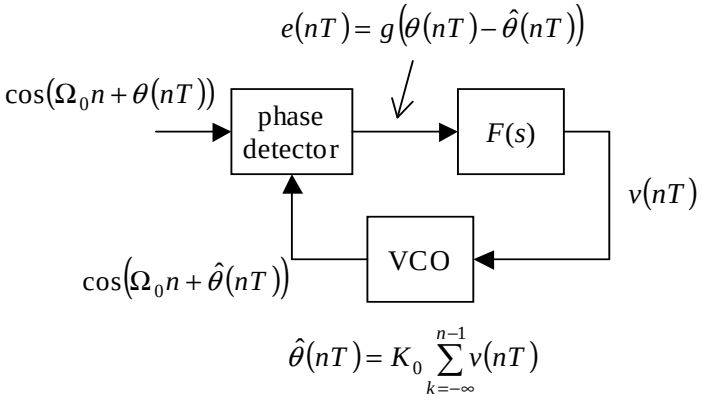
expressions for K_1 and K_2 in terms of the damping factor and loop bandwidth:

$$\begin{aligned} K_0K_pK_1 &= \frac{4\zeta\left(\dfrac{B_nT}{\zeta+\dfrac{1}{4\zeta}}\right)}{1+2\zeta\left(\dfrac{B_nT}{\zeta+\dfrac{1}{4\zeta}}\right)+\left(\dfrac{B_nT}{\zeta+\dfrac{1}{4\zeta}}\right)^2} \\ K_0K_pK_2 &= \frac{4\left(\dfrac{B_nT}{\zeta+\dfrac{1}{4\zeta}}\right)^2}{1+2\zeta\left(\dfrac{B_nT}{\zeta+\dfrac{1}{4\zeta}}\right)+\left(\dfrac{B_nT}{\zeta+\dfrac{1}{4\zeta}}\right)^2} \end{aligned} \tag{3.24}$$

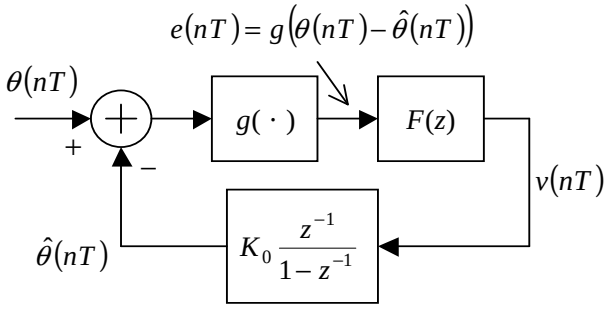
Note that when the equivalent loop bandwidth is small relative to the sample rate, $B_nT << 1$ so that equations (3.24) are well approximated by

$$\begin{aligned} K_0K_pK_1 &\approx \frac{4\zeta}{\zeta+\dfrac{1}{4\zeta}}\left(B_nT\right) \\ K_0K_pK_2 &\approx \frac{4}{\left(\zeta+\dfrac{1}{4\zeta}\right)^2}\left(B_nT\right)^2. \end{aligned} \tag{3.25}$$

Comparing equations (3.25) with equations (3.9) shows that for the case where the sample rate is large relative to the loop equivalent bandwidth, the expressions for the loop filter constants for the discrete-time loop are the same as those for the continuous-time loop except that the loop bandwidth is normalized by the sample rate.



(a)



(b)

Figure 3.3: Basic discrete-time PLL configuration: (a) the three basic discrete-time components of the discrete-time PLL. (b) The corresponding phase equivalent PLL.

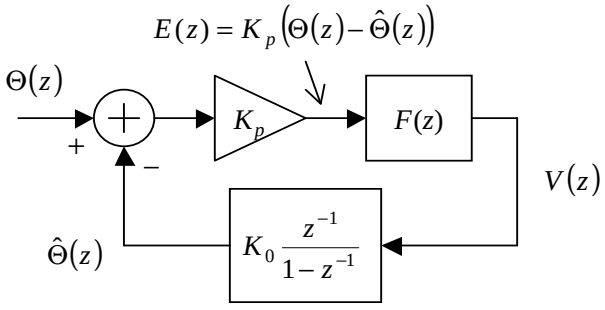


Figure 3.4: Linearized phase equivalent discrete-time PLL corresponding to the PLL in Figure 3.3 (b).

3.2.3 Summary

The preceding sections have summarized the key points from the Appendix that will be needed for the subsequent treatment of carrier phase synchronization and symbol timing synchronization. Both the continuous-time PLLs and the discrete-time PLLs analyzed above were designed to track the phase of a sinusoid at the loop input. Unfortunately, for communications synchronization applications, a sinusoid at the desired phase and frequency is rarely available. For example, in carrier phase synchronization for QPSK, the received waveform possesses 90-degree phase shifts due to the data phase shift keying the carrier. These phase shifts are in addition to the unknown carrier phase. If a QPSK waveform were input directly into a PLL designed to track the phase of a simple sinusoid, the PLL would try to track the phase shifts due to the data and probably never lock. Thus the carrier phase synchronization PLL must *remove* the phase shifts due to the data and track the remaining phase. This task can be accomplished by proper design of the phase detector.

The same idea applies to symbol timing synchronization. Most wireless applications do have have the luxury of embedding a reference clock signal in the modulated waveform. As a consequence, the symbol timing synchronization PLL must extract the data clock from the modulated waveform itself. Again, the data must be *removed* from the modulated waveform thus allowing the PLL to track the underlying data clock. As is the case with carrier phase synchronization, this task can be accomplished by proper design of the phase detector.

Since the phase detector will be responsible for removing the effects of the modulation on the underlying unmodulated carrier and data clock, the focus of the following sections is on the design and analysis of the phase detector. It will be important to keep in mind that the overall PLL still has the structure illustrated in Figure 3.1 (a) for continuous-time PLLs or Figure 3.3 (a) for discrete-time PLLs.

3.3 Carrier Phase Synchronization

In this section, the traditional order of presenting continuous-time systems followed by the discrete-time counterpart is reversed. This is done since it is easier to understand the operation of carrier phase synchronization in terms of a rotation in two-space using discrete-time signal processing.

Referring to Figure 3.5, the received MQASK waveform at IF may be represented as

$$r(t) = \sum_k a_1(k)p(t - kT_s)\cos(\omega_0 t + \theta) - a_2(k)p(t - kT_s)\sin(\omega_0 t + \theta) + w(t) \tag{3.26}$$

where $a_1(k)$ and $a_2(k)$ are the inphase and quadrature components of the k-th symbol, $p(t)$ is the unit energy pulse shape with support on $-L_pT_s \le t \le L_pT_s$, T_s is the symbol time, ω_0 is the radian IF frequency, θ is the unknown carrier phase offset, and $w(t)$ is the additive white Gaussian noise. The IF signal is sampled at a rate $F_s = 1/T$ samples/sec. The n-th sample of the received signal is

$$r(nT) = \sum_k a_1(k)p(nT - kT_s)\cos(\Omega_0 n + \theta) - a_2(k)p(nT - kT_s)\sin(\Omega_0 n + \theta) + w(nT) \tag{3.27}$$

where $\Omega_0 = \omega_0 T$ radians/sample. The received signal is downconverted using quadrature sinusoids $\cos(\Omega_0 n + \hat{\theta})$ and $-\sin(\Omega_0 n + \hat{\theta})$ to produce the inphase and quadrature components of the received signal which, neglecting the double frequency terms, may be expressed as

$$\begin{aligned} I(nT) &= \sum_k a_1(k)p(nT - kT_s)\cos(\theta - \hat{\theta}) - a_2(k)p(nT - kT_s)\sin(\theta - \hat{\theta}) + w_I(nT) \\ Q(nT) &= \sum_k a_1(k)p(nT - kT_s)\sin(\theta - \hat{\theta}) + a_2(k)p(nT - kT_s)\cos(\theta - \hat{\theta}) + w_Q(nT). \end{aligned} \tag{3.28}$$

The inphase and quadrature components are filtered by the matched filter whose impulse response is $p(-nT)$ to produce the inphase and quadrature matched filter outputs

$$\begin{aligned} x(nT) &= \sum_k a_1(k)R_p(nT - kT_s)\cos(\theta - \hat{\theta}) - a_2(k)R_p(nT - kT_s)\sin(\theta - \hat{\theta}) + v_I(nT) \\ y(nT) &= \sum_k a_1(k)R_p(nT - kT_s)\sin(\theta - \hat{\theta}) + a_2(k)R_p(nT - kT_s)\cos(\theta - \hat{\theta}) + v_Q(nT). \end{aligned} \tag{3.29}$$

where $R_p(u)$ is the autocorrelation function of the pulse shape given by

$$R_p(u) = \int_{-L_pT_s}^{L_pT_s} p(t)p(t - u)dt. \tag{3.30}$$

Assuming perfect timing synchronization, $x(nT)$ and $y(nT)$ are sampled at $n = kT_s/T$ to produce the inphase and quadrature matched filter outputs corresponding to the k-th symbol. Assuming

$R_p(0) = 1$, $R_p(mT_s) = 0$ for $m \neq 0$ these outputs may be expressed as

$$\begin{aligned} x(kT_s) &= a_1(k)\cos(\theta - \hat{\theta}) - a_2(k)\sin(\theta - \hat{\theta}) + v_I(kT_s) \\ y(kT_s) &= a_1(k)\sin(\theta - \hat{\theta}) + a_2(k)\cos(\theta - \hat{\theta}) + v_Q(kT_s). \end{aligned} \tag{3.31}$$

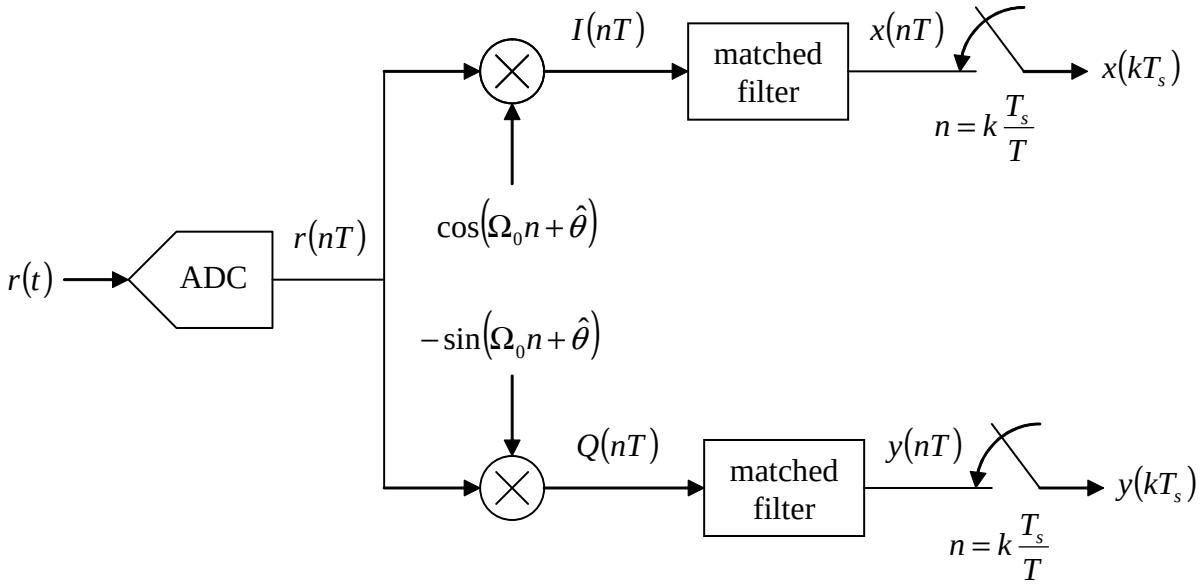
This shows that, for MQASK waveforms, the the effect of uncompensated carrier phase offset is a rotation in the signal space projections. This is illustrated for the case of QPSK shown in Figure 3.6. In the absence of noise and assuming perfect timing synchronization, $x(kT_s)$ and $y(kT_s)$ form the Cartesian coordinates of a rotated version of the true symbol point $(a_1(k), a_2(k))$. The angle of rotation is the uncompensated phase error $\theta - \hat{\theta}$.

Two approaches to carrier phase synchronization can be envisioned. In the first approach, phase compensation is performed at the output of the matched filter as illustrated in Figure 3.7. The quadrature sinusoids used for downconversion are $\cos \Omega_0 n$ and $-\sin \Omega_0 n$ so that the downsampled matched filter outputs are special cases of (3.31) with $\hat{\theta} = 0$:

$$\begin{aligned} x(kT_s) &= a_1(k)\cos(\theta) - a_2(k)\sin(\theta) + v_I(kT_s) \\ y(kT_s) &= a_1(k)\sin(\theta) + a_2(k)\cos(\theta) + v_Q(kT_s). \end{aligned} \tag{3.32}$$

The sampled matched filter outputs $(x(kT_s), y(kT_s))$ are de-rotated by the estimated carrier phase offset using a rotation function following the matched filters. Since the sampled matched filter outputs form a discrete-time sequence, this approach is a purely discrete-time approach. The second approach modifies the phases of the quadrature sinusoids used to mix the IF signal to baseband. This approach is illustrated in Figure 3.8 and is a commonly used architecture for both continuous-time and discrete-time implementations.

In both figures, the dashed line represents an optional connection between the symbol estimates $\hat{a}_1(k)$ and $\hat{a}_2(k)$ and the phase error detector represented by the "Compute Phase Error" block. When the phase error detector uses the symbol estimates to compute the phase error, the resulting PLL is called a *decision-directed* loop. Alternatively, the phase error may be computed using knowledge of the transmitted data symbols. Usually, the known data takes the form of a predefined data sequence, known as a *training sequence*, that is inserted at the beginning of the transmission for the purposes of phase acquisition. This approach is commonly used for packetized data links where the training sequence forms part of the packet header or preamble. A carrier phase PLL that uses known data is often called a *data aided* PLL.



$$r(t) = \sum_k a_1(k)p(t - kT_s)\cos(\omega_0 t + \theta) - a_2(k)p(t - kT_s)\sin(\omega_0 t + \theta)$$

Figure 3.5: Block diagram of a discrete-time MQASK receiver using IF sampling. The phase offset between the received signal and the local oscillators is shown.

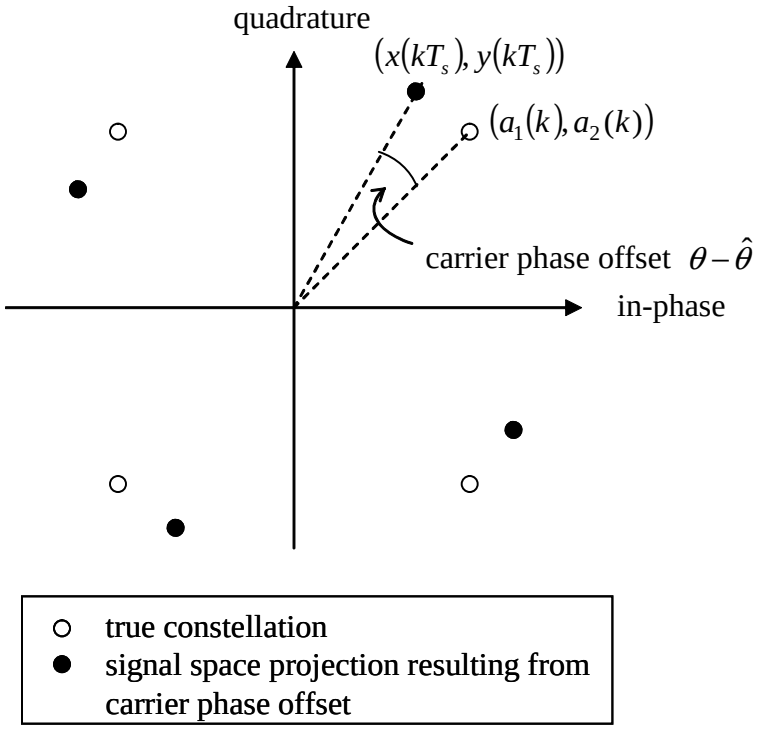


Figure 3.6: The rotation effect of carrier phase offset on the QPSK constellation.

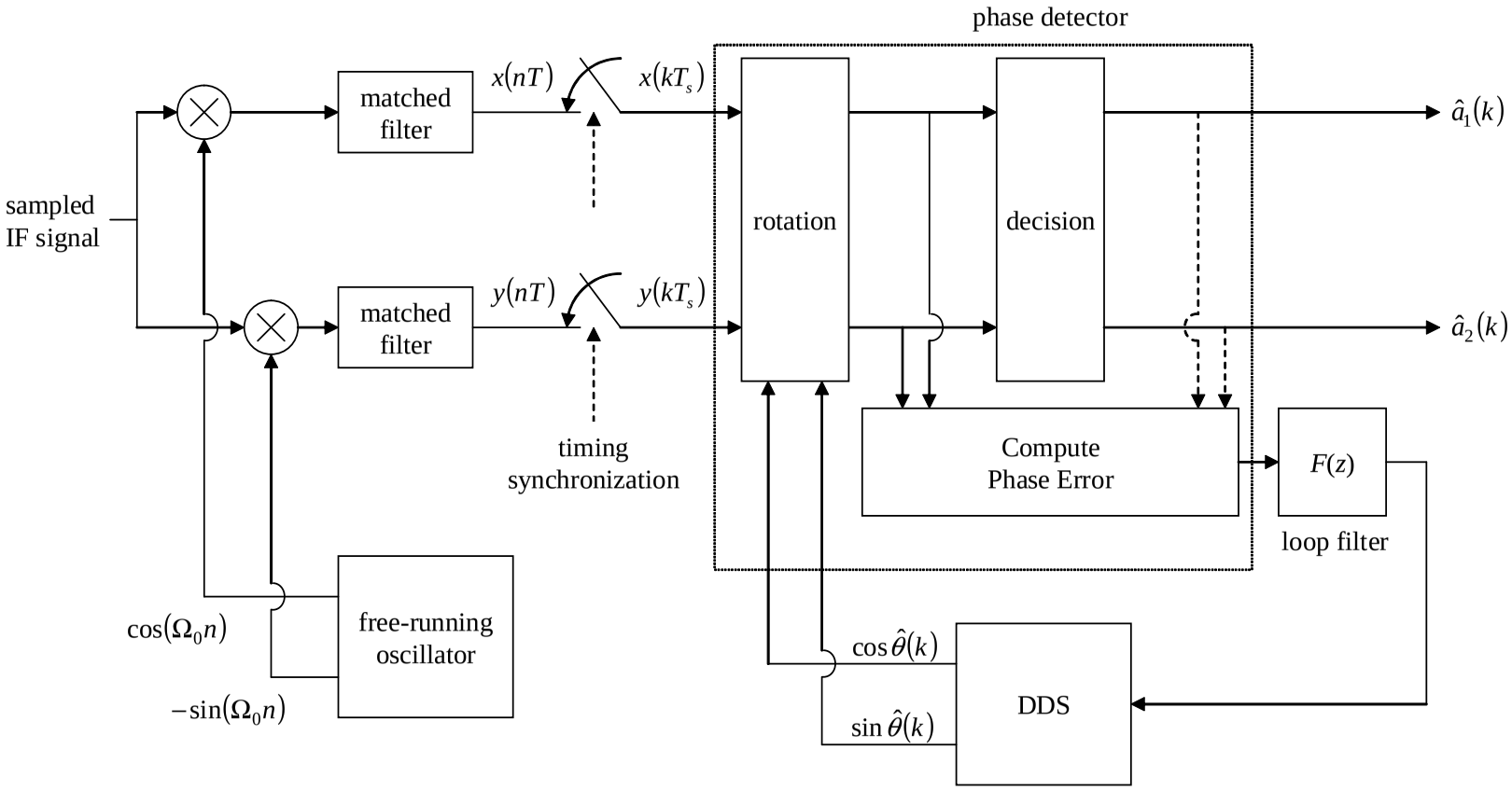


Figure 3.7: Carrier phase synchronization using a post-matched filter derotation operation.

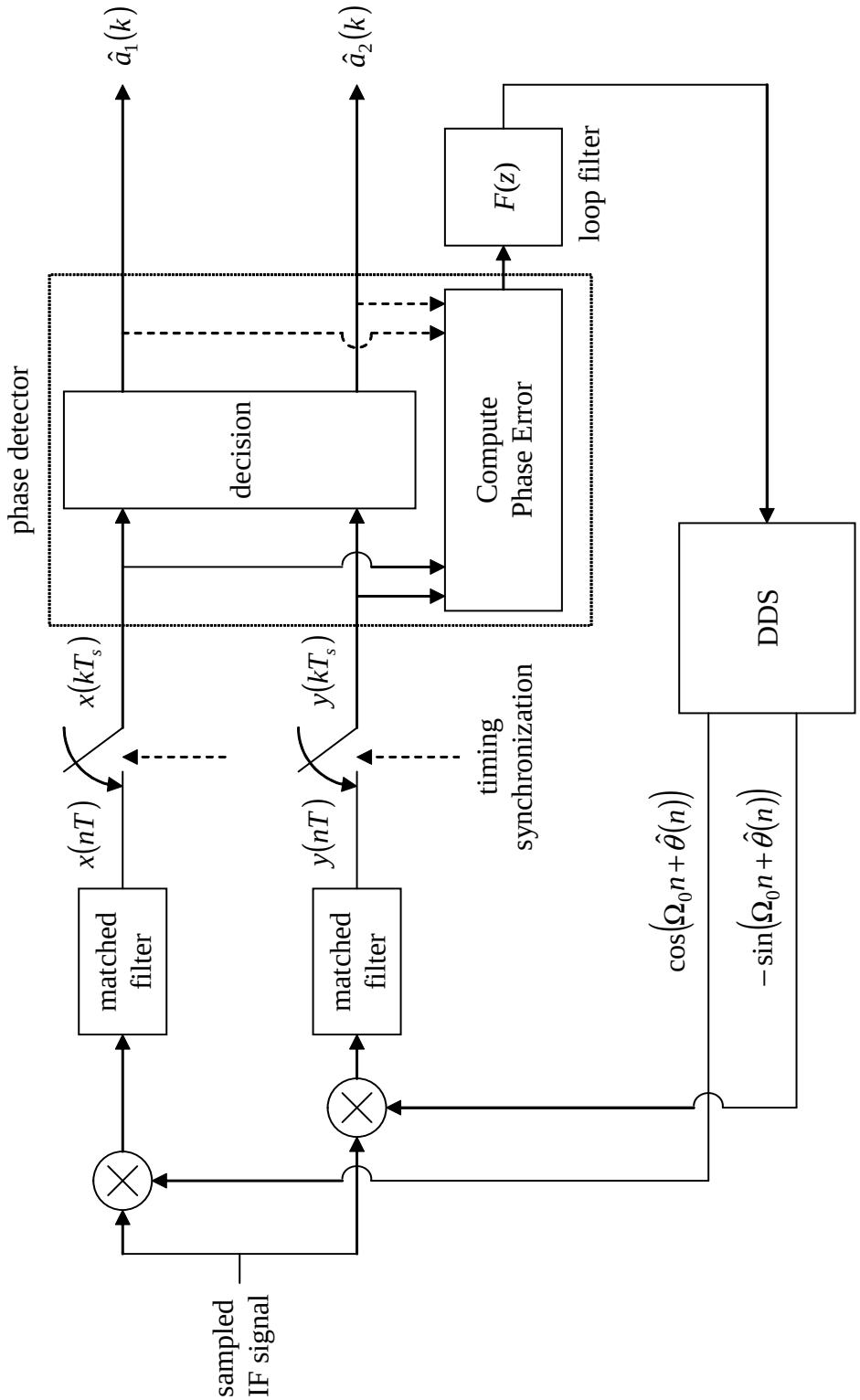


Figure 3.8: Carrier phase synchronization using phase adjusted quadrature sinusoids.

3.3.1 Carrier Phase Synchronization for QPSK

As an example of the carrier phase architecture outlined in Figure 3.7, consider the QPSK carrier phase synchronizer illustrated in Figure 3.9. The inphase and quadrature matched filter outputs, $x(kT_s)$ and $y(kT_s)$, are rotated by $-\hat{\theta}(k)$, to align the signal space projection $(x'(kT_s), y'(kT_s))$ with the constellation points. The relationship between the inphase and quadrature matched filter outputs and the rotated points is given by the matrix equation

$$\begin{bmatrix} x'(kT_s) \\ y'(kT_s) \end{bmatrix} = \begin{bmatrix} \cos\hat{\theta}(k) & \sin\hat{\theta}(k) \\ -\sin\hat{\theta}(k) & \cos\hat{\theta}(k) \end{bmatrix} \begin{bmatrix} x(kT_s) \\ y(kT_s) \end{bmatrix}. \tag{3.33}$$

With the switch in the lower position, the phase error is extracted from the point $(x'(kT_s), y'(kT_s))$ by computing the residual phase difference between $(x'(kT_s), y'(kT_s))$ and the transmitted constellation point $(a_1(k), a_2(k))$.

Computation of the phase error is easily understood in geometric terms. Consider the scenario shown in Figure 3.11. The phase angle of the de-rotated matched filter outputs is

$$\theta_r(k) = \tan^{-1}\left\{\frac{y'(kT_s)}{x'(kT_s)}\right\} \tag{3.34}$$

and the phase angle of the transmitted constellation point is

$$\theta_d(k) = \tan^{-1}\left\{\frac{a_2(kT_s)}{a_1(kT_s)}\right\}. \tag{3.35}$$

The phase error for the k-th symbol is thus

$$\begin{aligned} e(k) &= \theta_r(k) - \theta_d(k) \\ &= \tan^{-1}\left\{\frac{y'(kT_s)}{x'(kT_s)}\right\} - \tan^{-1}\left\{\frac{a_2(kT_s)}{a_1(kT_s)}\right\}. \end{aligned} \tag{3.36}$$

The S-curve is obtained by writing

$$\begin{bmatrix} x'(kT_s) \\ y'(kT_s) \end{bmatrix} = \begin{bmatrix} \cos\theta_e & -\sin\theta_e \\ \sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} a_1(k) \\ a_2(k) \end{bmatrix},$$

and computing $e(k)$ in terms of $a_1(k)$, $a_2(k)$ and θ_e. The average S-curve, denoted $\overline{g}(\theta_e)$, is obtained by averaging over the four possible symbols $(a_1(k), a_2(k)) \in \{\pm 1, \pm 1\}$. After a little algebra the result is

$$\overline{g}(\theta_e) = \theta_e. \tag{3.37}$$

The average S-curve (3.37) is plotted in Figure 3.14 (a) where it is seen that this phase detector is an ideal linear phase detector with $K_p = 1$.

When the actual transmitted data symbols are unknown (either the training sequence has passed or there was no training sequence provided) the carrier phase synchronizer can use the data decisions to compute the phase error. This approach is illustrated in Figure 3.10. The phase error is extracted from the point $(x'(kT_s), y'(kT_s))$ by computing the residual phase difference between $(x'(kT_s), y'(kT_s))$ and the nearest constellation point $(\hat{a}_1(k), \hat{a}_2(k))$ where, for QPSK, $\hat{a}_1(k) = \text{sgn}\{x'(kT_s)\}$ and $\hat{a}_2(k) = \text{sgn}\{y'(kT_s)\}$. Thus, the *decision directed* carrier phase synchronizer replaces $a_1(k)$ and $a_2(k)$ in (3.36) with the decisions $\hat{a}_1(k)$ and $\hat{a}_2(k)$:

$$e(k) = \tan^{-1}\left\{\frac{y'(kT_s)}{x'(kT_s)}\right\} - \tan^{-1}\left\{\frac{\hat{a}_2(k)}{\hat{a}_1(k)}\right\} \tag{3.38}$$

$$= \tan^{-1}\left\{\frac{y'(kT_s)}{x'(kT_s)}\right\} - \tan^{-1}\left\{\frac{\text{sgn}\{y'(k)\}}{\text{sgn}\{x'(k)\}}\right\}. \tag{3.39}$$

The average S-curve for the phase detector based on (3.39) is computed using the same procedure used to compute (3.36). The average S-curve for the decision-directed phase detector is

$$\overline{g}(\theta_e) = \begin{cases} \theta + \pi & -\pi \le \theta_e < -\dfrac{3\pi}{4} \\ \theta + \dfrac{\pi}{2} & -\dfrac{3\pi}{4} < \theta_e < -\dfrac{\pi}{4} \\ \theta & -\dfrac{\pi}{4} < \theta_e < \dfrac{\pi}{4} \\ \theta - \dfrac{\pi}{2} & \dfrac{\pi}{4} < \theta_e < \dfrac{3\pi}{4} \\ \theta - \pi & \dfrac{3\pi}{4} < \theta_e \le \pi \end{cases} \tag{3.40}$$

and is plotted in Figure 3.14 (b). Note that the slope of the S-curve at $\theta_e = 0$ is 1 so that $K_p = 1$ for this phase detector.

Comparing the S-curves for the data-aided and decision-directed phase detectors in Figure 3.14 reveals some interesting differences. The S-curve for the decision-directed loop crosses zero at

$$\theta_e = -\frac{3\pi}{4}, -\frac{\pi}{2}, -\frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi.$$

Since a phase-locked loop locks at $\theta_e = 0$, the question arises: which of these zero crossing represents a stable lock point? As it turns out, only those values of θ_e where $g(\theta_e)$ passes through zero with a positive slope are stable lock points. Thus the stable lock points are

$$\theta_e = -\frac{\pi}{2}, 0, \frac{\pi}{2}, \pi.$$

As a consequence, the QPSK carrier phase PLL could lock in phase with true carrier phase, $\pm 90^\circ$ out of phase with the true carrier phase, or 180° out of phase with the true carrier phase. This PLL

possesses what is called a $\pi/2$ *phase ambiguity*[4]. The phase ambiguity is a byproduct of removing the data-induced phase shifts from the received signal. Since the QPSK constellation has a $\pi/2$ rotational symmetry, a $\pi/2$ phase ambiguity is to be expected.

The phase detectors based on (3.36) and (3.39) require two four-quadrant arctangent operations and a subtraction. A reduced complexity phase detector can be obtained by using the sine of the phase error in place of the phase error. Taking the sine of (3.36) and applying the identity $\sin(A-B) = \sin A \cos B - \cos A \sin B$ produces

$$\sin\left(\theta_r(k) - \theta_d(k)\right) = \sin\left(\theta_r(k)\right)\cos\left(\theta_d(k)\right) - \cos\left(\theta_r(k)\right)\sin\left(\theta_d(k)\right) \tag{3.41}$$

$$= \frac{y'(kT_s)a_1(k) - x'(kT_s)a_2(k)}{\sqrt{x'^2(kT_s) + y'^2(kT_s)}\sqrt{a_1^2(k) + a_2^2(k)}} \tag{3.42}$$

To avoid the division suggested by (3.42), the numerator alone can be used as the error signal while the denominator terms are absorbed into the phase detector gain K_p. Thus

$$e(k) = y'(kT_s)a_1(k) - x'(kT_s)a_2(k). \tag{3.43}$$

A block diagram of this approach is illustrated in Figure 3.12. Again, the average S-curve is obtained by writing

$$\begin{bmatrix} x'(kT_s) \\ y'(kT_s) \end{bmatrix} = \begin{bmatrix} \cos\theta_e & -\sin\theta_e \\ \sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} a_1(k) \\ a_2(k) \end{bmatrix},$$

computing $g\left(\theta_e\right)$ in terms of $a_1(k)$ and $a_2(k)$, and averaging over the four possible symbols $(a_1(k), a_2(k)) \in \{\pm 1, \pm 1\}$. After a little algebra the result is

$$\overline{g}(\theta_e) = 2\sin\theta_e. \tag{3.44}$$

Note that for $\theta_e \approx 0$, $\overline{g}(\theta_e) \approx 2\theta_e$ from which $K_p = 2$.

The decision-directed version of the simplified error detector is obtained by replacing $a_1(k)$ and $a_2(k)$ in (3.43) with $\hat{a}_1(k)$ and $\hat{a}_2(k)$. The error signal for the simplified decision-directed phase detector is

$$e(k) = y'(kT_s)\text{sgn}\left\{x'(kT_s)\right\} - x'(kT_s)\text{sgn}\left\{y'(kT_s)\right\}. \tag{3.45}$$

A block diagram of a QPSK carrier phase PLL using the simplified error detector (3.45) is shown

[4]An S-curve with L stable lock points produces a phase-locked loop with a $2\pi/L$ phase ambiguity.

in Figure 3.13. The average S-curve for (3.45) is

$$\overline{g}\left(\theta_e\right) = \begin{cases} -2\sin\theta_e & -\pi \le \theta_e < -\dfrac{3\pi}{4} \\ 2\cos\theta_e & -\dfrac{3\pi}{4} < \theta_e < -\dfrac{\pi}{4} \\ 2\sin\theta_e & -\dfrac{\pi}{4} < \theta_e < \dfrac{\pi}{4} \\ -2\cos\theta_e & \dfrac{\pi}{4} < \theta_e < \dfrac{2\pi}{4} \\ -2\sin\theta_e & \dfrac{3\pi}{4} < \theta_e \le \pi \end{cases} . \tag{3.46}$$

Again, note that for $\theta_e \approx 0$, $\overline{g}(\theta_e) \approx 2\theta_e$ from which $K_p = 2$.

The average S-curves for the simplified data-aided phase detector (3.44) and the simplified decision-directed phase detector (3.46) are plotted in Figure 3.15 (a) and (b), respectively. The two S-curves for the simplified error detector differ from each other in the same way the two S-curves for the arctangent-based error detector differed. The S-curve for the data-aided error detector is non-linear, but possesses only one stable lock point. The S-curve for the decision-directed error detector crosses zero with a positive slope at

$$\theta_e = -\frac{\pi}{2}, 0, \frac{\pi}{2}, \pi$$

and thus has four stable lock points resulting in a $\pi/2$ phase ambiguity.

As an example of loop design using the simplified decision-directed error detector, suppose system requirements call for a critically damped QPSK carrier phase synchronizer PLL with an equivalent loop bandwidth of 2% of the symbol rate. Using a discrete-time proportional-plus-integrator loop filter, $\zeta = 1/\sqrt{2}$ and $B_nT_s = B_nT = 0.02$ together with $K_p = 2$ and $K_0 = 1$ in (3.24), the loop filter constants are

$$K_1 = 2.6 \times 10^{-2} \tag{3.47}$$

$$K_2 = 6.9 \times 10^{-4} \tag{3.48}$$

The phase estimate $\hat{\theta}(k)$ and phase error $e(k)$ for a sequence of 250 randomly generated QPSK symbols is illustrated in Figure 3.16. The carrier phase offset is a $\pi/4$ step. Observe that $\hat{\theta}(k)$ settles to $\pi/4$ after about 200 symbols and that the phase error settles to zero at the same time. The nature of the transient response is controlled by the loop filter constants which are determined by the damping factor and loop bandwidth. This loop is slightly underdamped and exhibits an overshoot in response to the phase step input in the carrier.

Another popular architecture for QPSK carrier phase synchronization is based on the general architecture illustrated in Figure 3.8. In this system, the DDS is designed to operate at the IF frequency Ω_0 radians/sample. Carrier phase compensation is incorporated into DDS used to generate

the quadrature sinusoids used for the down-conversion from IF. As such, there is no need for the phase rotation block. Any of the four phase error detectors (3.36), (3.39), (3.43), and (3.45) described above can be used in the "Compute Phase Error" block where the sampled matched filter outputs $x(kT_s)$ and $y(kT_s)$ are used in place of $x'(kT_s)$ and $y'(kT_s)$. Note that the DDS operates at N samples/symbol while the phase error estimate is updated once per symbol. As a consequence, a rate conversion is required. As an example, consider the block diagram in Figure 3.17. In this example, the upsample block is placed in between the phase detector and the loop filter. As a consequence, the phase detector and loop filter operate at one sample/symbol while the rest of the loop (the DDS, and matched filters) operate at N samples/symbol. The filter constants should be computed assuming operation at one sample/symbol with a DDS constant $K_0 = N$. This is because from the loop filter point of view, the DDS increments N times for each step in the loop filter. Since the matched filter is included in the closed-loop path, a small equivalent loop bandwidth is required for stable operation when the loop filter constants are based on a second-order system.

Returning to the design example requiring a QPSK carrier phase PLL with an equivalent loop bandwidth 2% of the symbol rate, assume the IF sample rate is $N = 16$ samples/symbol. For the QPSK carrier phase PLL illustrated in Figure 3.17, $K_p = 2$ and $K_0 = 16$. Using $\zeta = 1/\sqrt{2}$ and $B_nT_s = 16B_nT = 0.02$ in (3.24), the loop filter constants are

$$K_1 = 1.7 \times 10^{-3} \tag{3.49}$$

$$K_2 = 2.8 \times 10^{-6} \tag{3.50}$$

The phase error for a sequence of 250 randomly generated QPSK symbols is illustrated in Figure 3.18. As before, the carrier frequency phase is $\pi/4$ step. Two important observations should be noted. First, the phase error exhibits discontinuities that are a consequence of the fact that phase adjustments are being made at the IF frequency rather than at baseband. The interaction between small phase adjustments and the carrier frequency is more likely to cause 180° phase jumps in the error signal $e(k)$. Second, the error signal takes a long time to settle to zero. This is a consequence of the inclusion of the matched filters in the closed loop path.

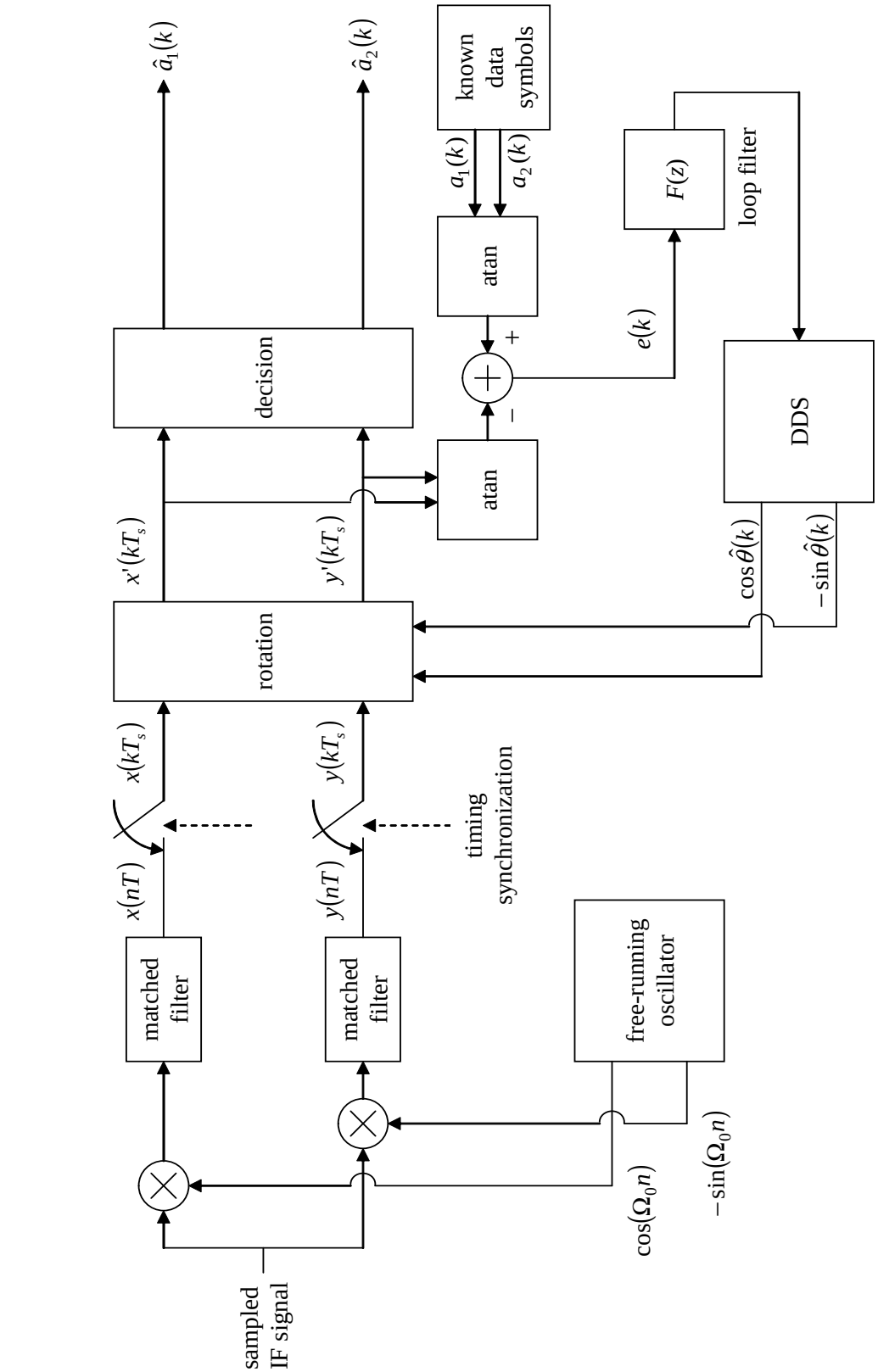


Figure 3.9: Block diagram of the QPSK carrier phase PLL using an error signal based on (3.36).

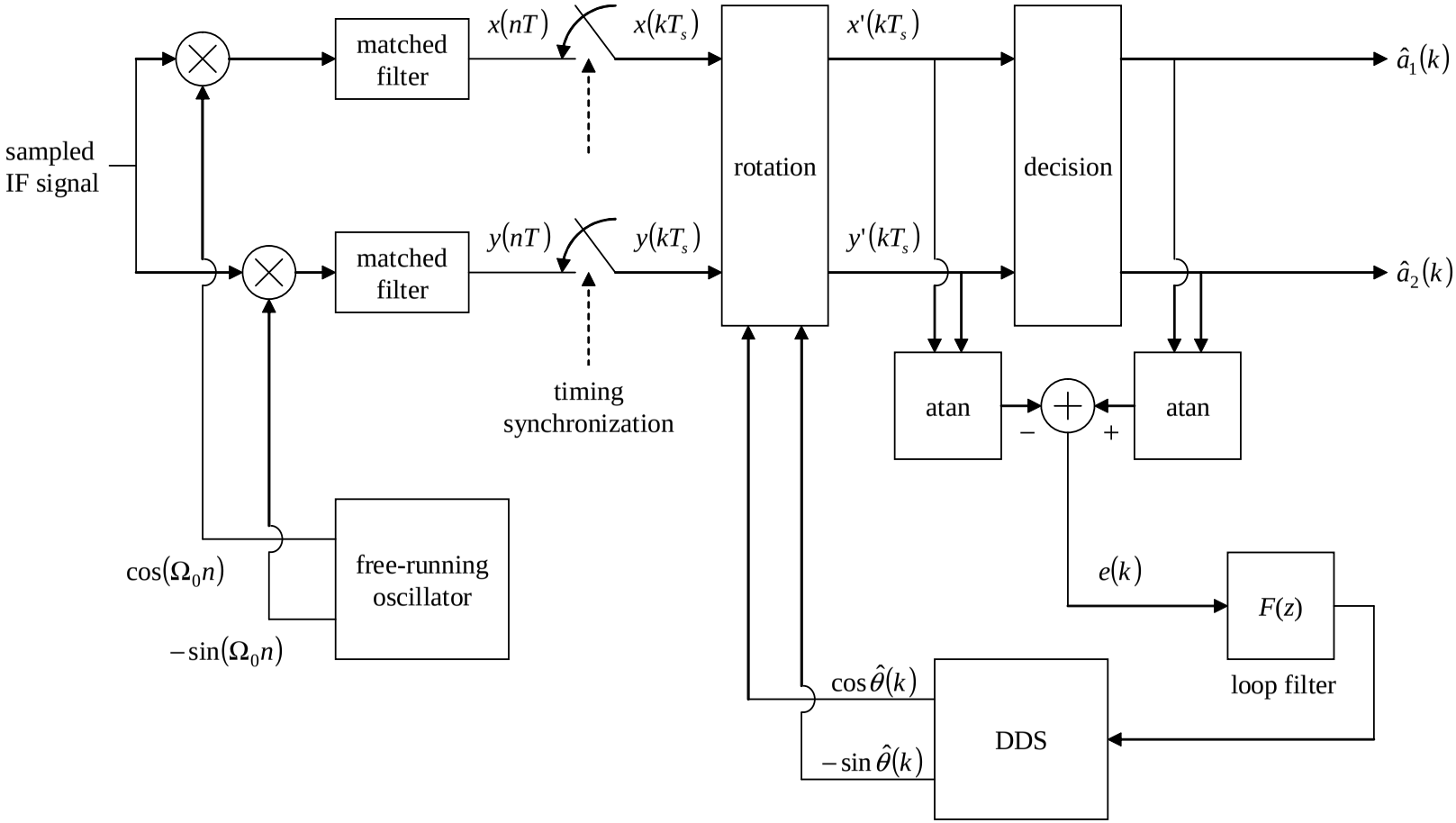


Figure 3.10: Block diagram of the QPSK carrier phase PLL using an error signal based on (3.39).

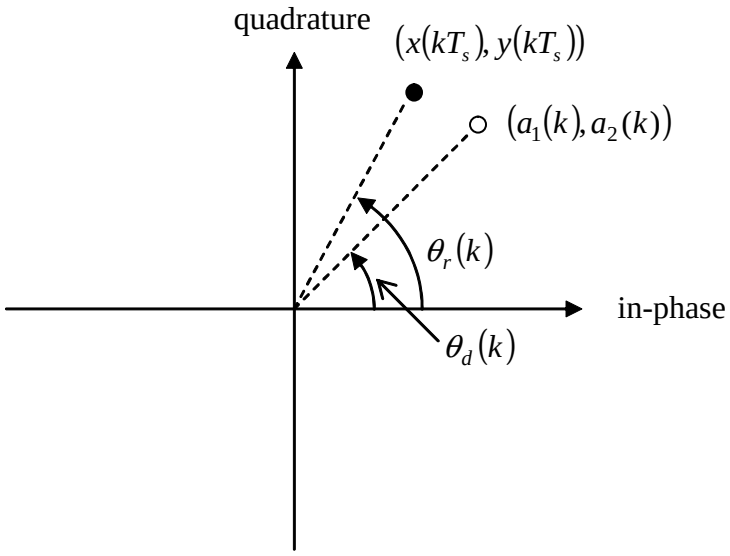


Figure 3.11: Geometric representation of the phase error computation in a QPSK carrier phase PLL.

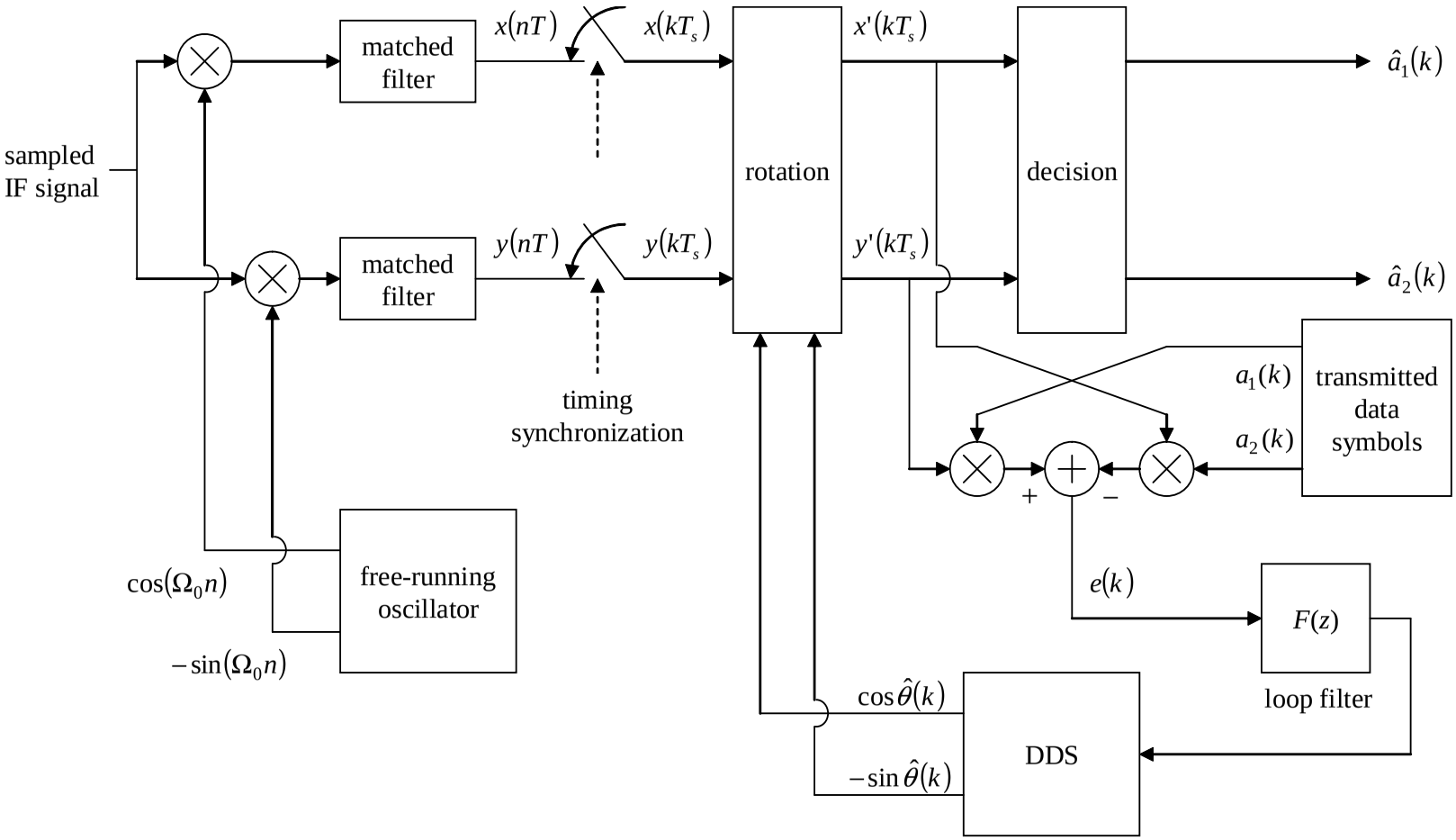


Figure 3.12: Block diagram of the QPSK carrier phase PLL using an error signal based on (3.43).

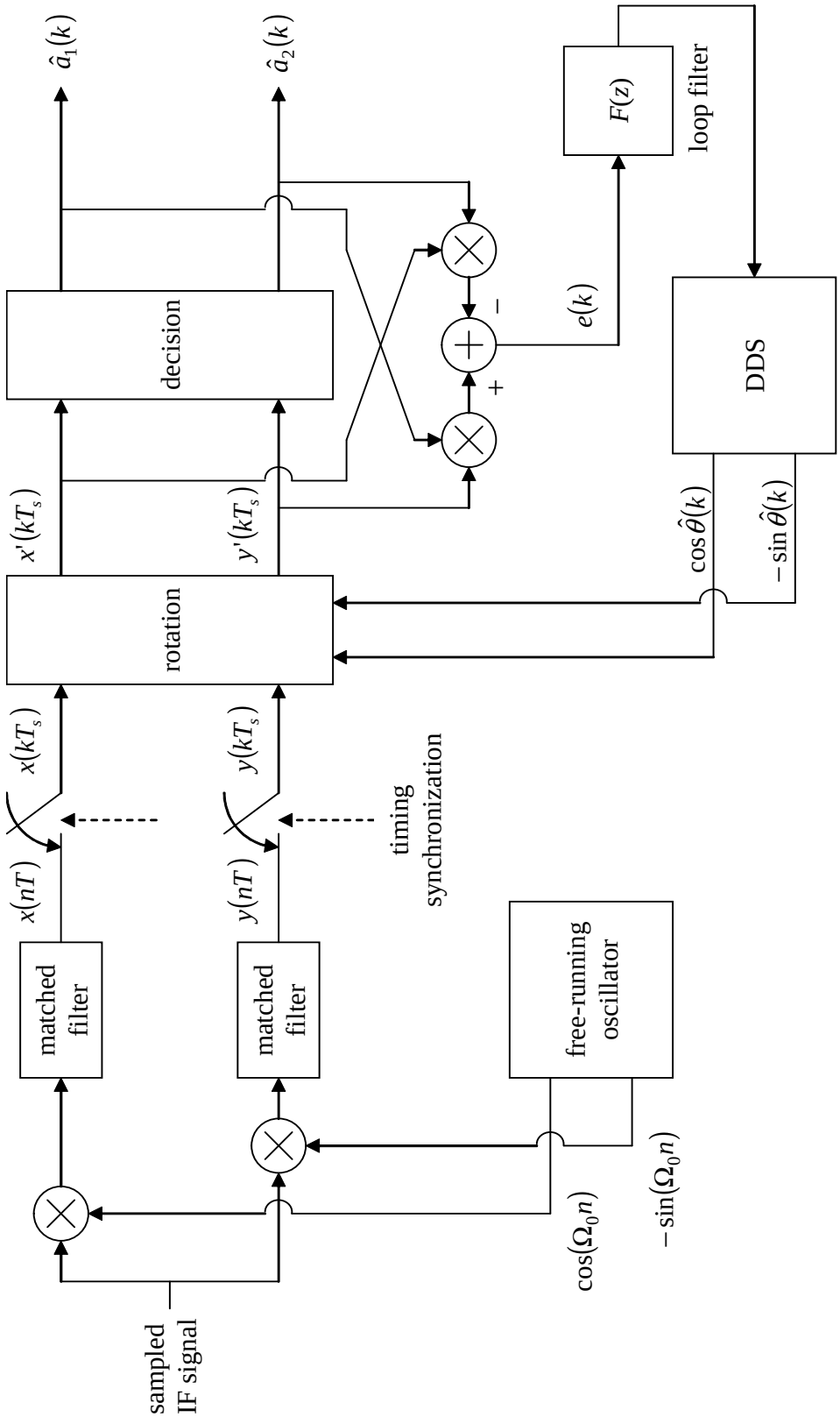


Figure 3.13: Block diagram of the QPSK carrier phase PLL using an error signal based on (3.45).

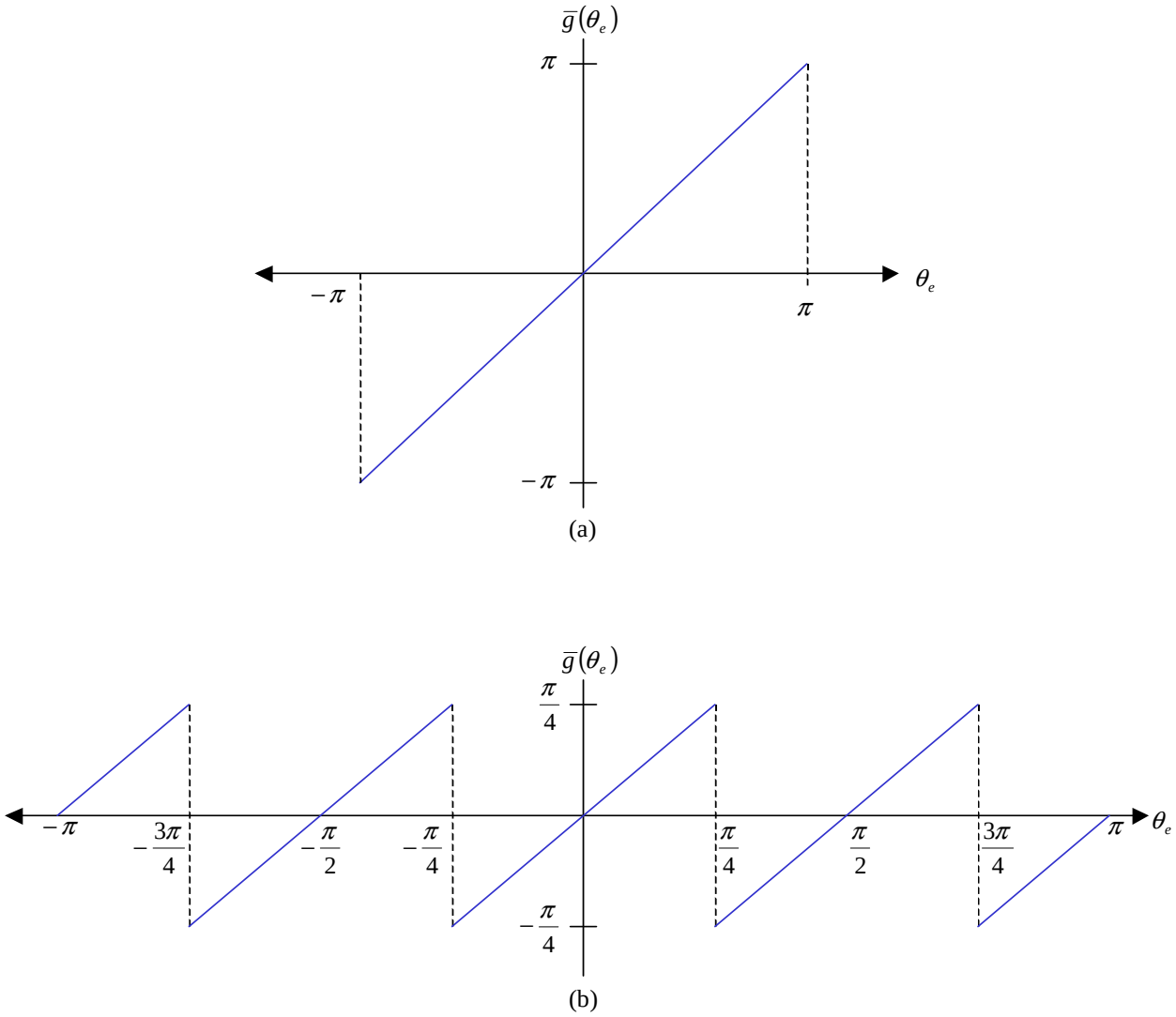


Figure 3.14: S-curve for the data-aided phase detector (3.37) (a) and the decision-directed phase detector (3.40) (b).

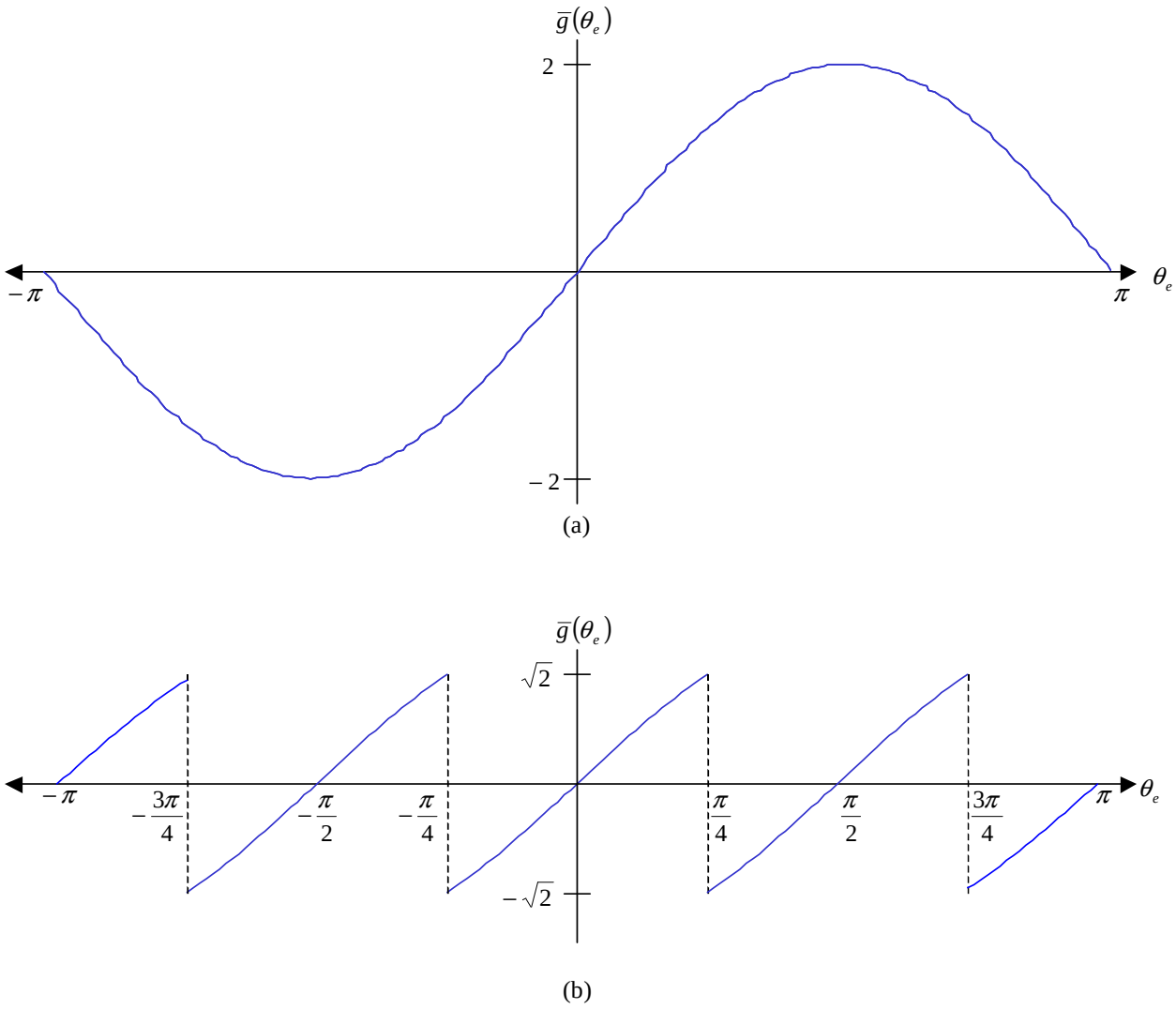


Figure 3.15: S-curve for the data-aided phase detector (3.44) (a) and the decision-directed phase detector (3.46) (b).

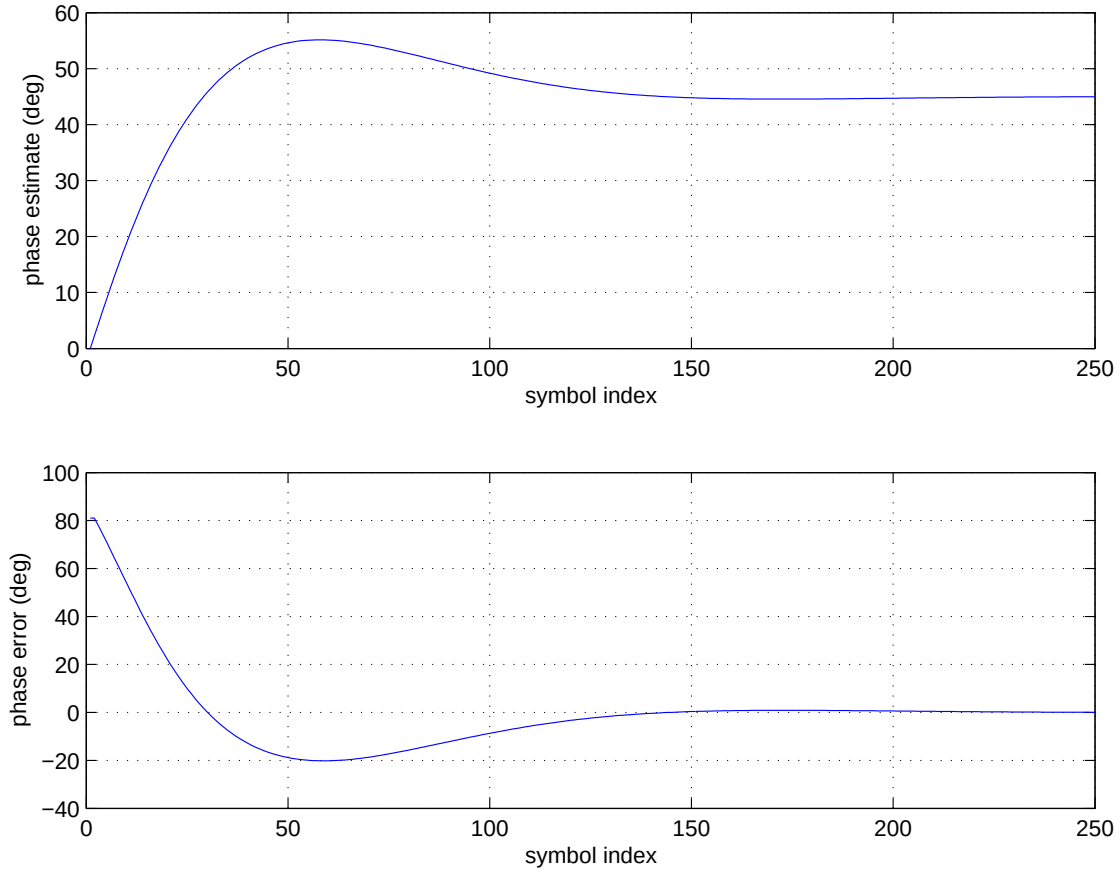


Figure 3.16: Phase estimate $\hat{\theta}(k)$ (top) and corresponding phase error $e(k)$ (bottom) for the first QPSK carrier phase PLL example.

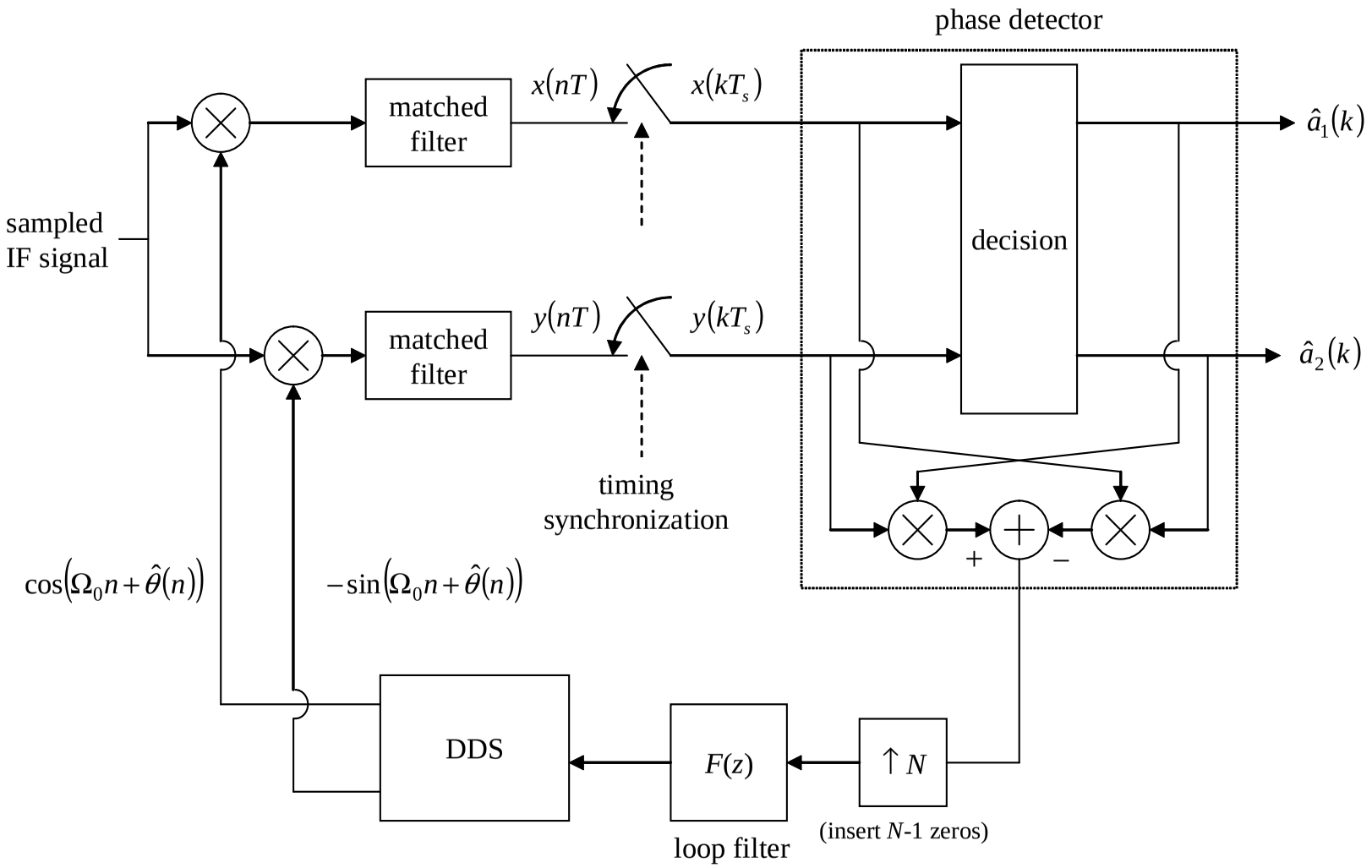


Figure 3.17: QPSK carrier phase synchronization based on the general architecture shown in Figure 3.8 and using the phase error (3.45).

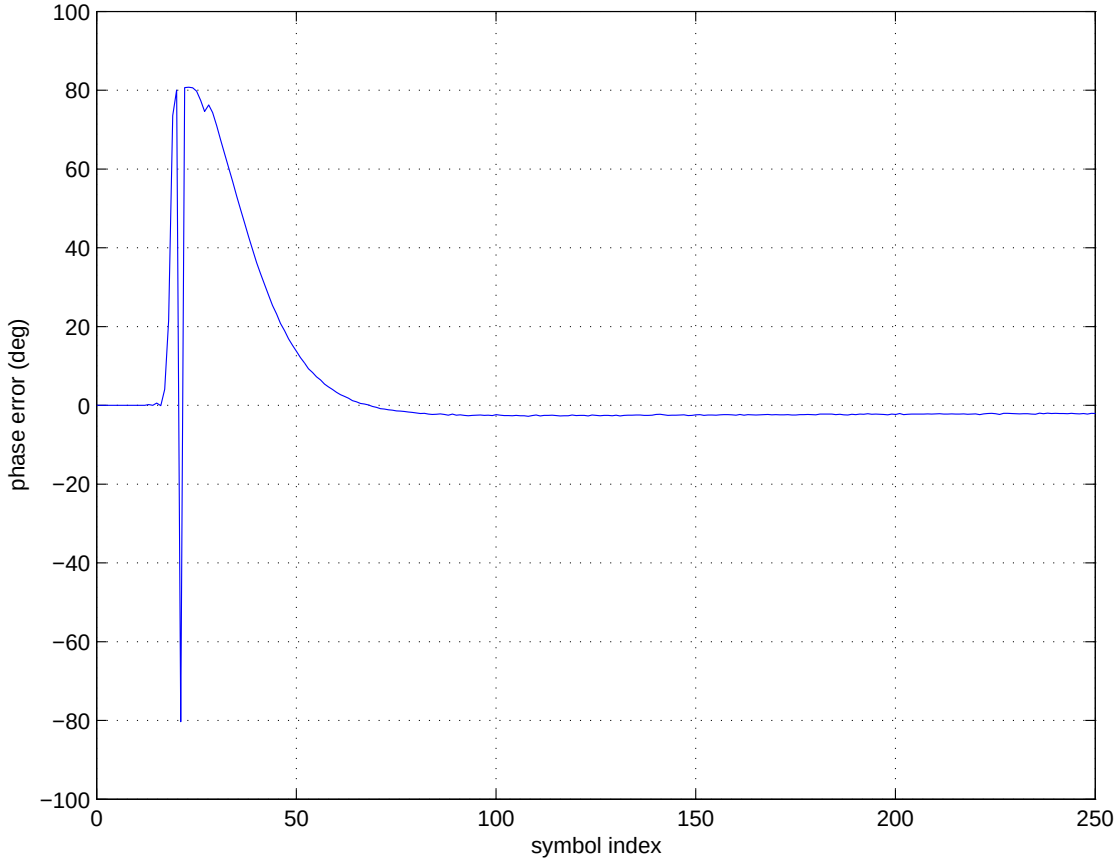


Figure 3.18: Phase error $e(k)$ for the second QPSK carrier phase PLL example.

3.3.2 Carrier Phase Synchronization for BPSK

Either of the two general architectures illustrated in Figures 3.7 and 3.8 can be used for BPSK carrier phase synchronization. Since either of these approaches use the same phase detector, the focus of this section is on the phase detector. The difference between carrier phase synchronization for BPSK and QPSK lies in the phase detector. The BPSK phase detector based on the arctangent uses the error signal

$$e(k) = \tan^{-1}\left\{\frac{y'(kT_s)}{x'(kT_s)}\right\} - \tan^{-1}\left\{\frac{0}{a_1(k)}\right\} \tag{3.51}$$

for the data-aided phase detector, and

$$e(k) = \tan^{-1}\left\{\frac{y'(kT_s)}{x'(kT_s)}\right\} - \tan^{-1}\left\{\frac{0}{\text{sgn}\left\{x'(kT_s)\right\}}\right\} \tag{3.52}$$

for the decision-directed phase detector. The numerator of the second term in (3.51) and (3.52) is zero since the BPSK constellation is a one-dimensional constellation: there is no data on the quadrature carrier component when the receiver is operating in phase coherence with the transmitter. Note that the second term on the right hand side of (3.51) and (3.52) is 0 when $x'(kT_s) > 0$ and π when $x'(kT_s) < 0$. It should also be observed that (3.51) follows from (3.36) and (3.52) follows from (3.39) when $a_2(k) = 0$. When the architecture of Figure 3.8 is used, $x(kT_s)$ and $y(kT_s)$ are used in place of $x'(kT_s)$ and $y'(kT_s)$ in (3.51) and (3.52).

The S-curve for the phase detector (3.51) is

$$\overline{g}\left(\theta_e\right) = \theta_e \tag{3.53}$$

while the S-curve for the phase detector (3.52) is

$$\overline{g}\left(\theta_e\right) = \begin{cases} \theta_e + \pi & -\pi < \theta_e < -\dfrac{\pi}{2} \\ \theta_e & -\dfrac{\pi}{2} < \theta_e < \dfrac{\pi}{2} \\ \theta_e - \pi & \dfrac{\pi}{2} < \theta_e < \pi \end{cases}. \tag{3.54}$$

The S-curves (3.53) and (3.54) are plotted in Figure 3.19 (a) and (b), respectively. Note that the S-curve for the decision directed phase detector possesses two stable lock points at $\theta_e = 0$ and $\theta_e = \pi$ and therefor has a π-phase ambiguity which is equal to the rotational symmetry of the BPSK constellation. Both phase detectors have unity slope at $\theta_e = 0$. Thus $K_p = 1$.

The simplified phase detectors based on $\sin\theta_e$ are

$$e(k) = y'(kT_s)a_1(k) \tag{3.55}$$

for the data-aided phase detector and

$$g\left(\theta_e\right) = y'(kT_s)\text{sgn}\left\{x'(kT_s)\right\} \tag{3.56}$$

for the decision-directed phase detector. Note that (3.55) follows from (3.43) and (3.56) follows from (3.45) when $a_2(k) = 0$. As before, if the architecture of Figure 3.8 is used, $x(kT_s)$ and $y(kT_s)$ are used in place of $x'(kT_s)$ and $y'(kT_s)$ in (3.55) and (3.56).

The S-curve for the data-aided phase detector (3.55) is

$$\overline{g}\left(\theta_e\right) = \sin\theta_e \tag{3.57}$$

while the S-curve for the phase detector (3.56) is

$$\overline{g}\left(\theta_e\right) = \begin{cases} -\sin\theta_e & -\pi < \theta_e < -\dfrac{\pi}{2} \\ \sin\theta_e & -\dfrac{\pi}{2} < \theta_e < \dfrac{\pi}{2} \\ -\sin\theta_e & \dfrac{\pi}{2} < \theta_e < \pi \end{cases}. \tag{3.58}$$

The S-curves (3.57) and (3.58) are plotted in Figure 3.20 (a) and (b), respectively. Observe that the S-curve for the decision-directed phase detector possesses two stable lock points at $\theta_e = 0$ and $\theta_e = \pi$ and therefor has a π-phase ambiguity. Both S-curves are approximated by $\sin\theta_e \approx \theta_e$ for $\theta_e \approx 0$ and therefor have $K_p = 1$.

At first, it may seem odd that a carrier phase synchronizer for BPSK requires the quadrature matched filter outputs. The quadrature component is needed to compute the phase rotation at the matched filter outputs. Since there is no information on the quadrature component of the carrier for BPSK, the BPSK carrier phase PLL locks when the residual quadrature component goes to zero. Observe that all of the BPSK phase detectors go to zero when $y'(kT_s)$ is zero.

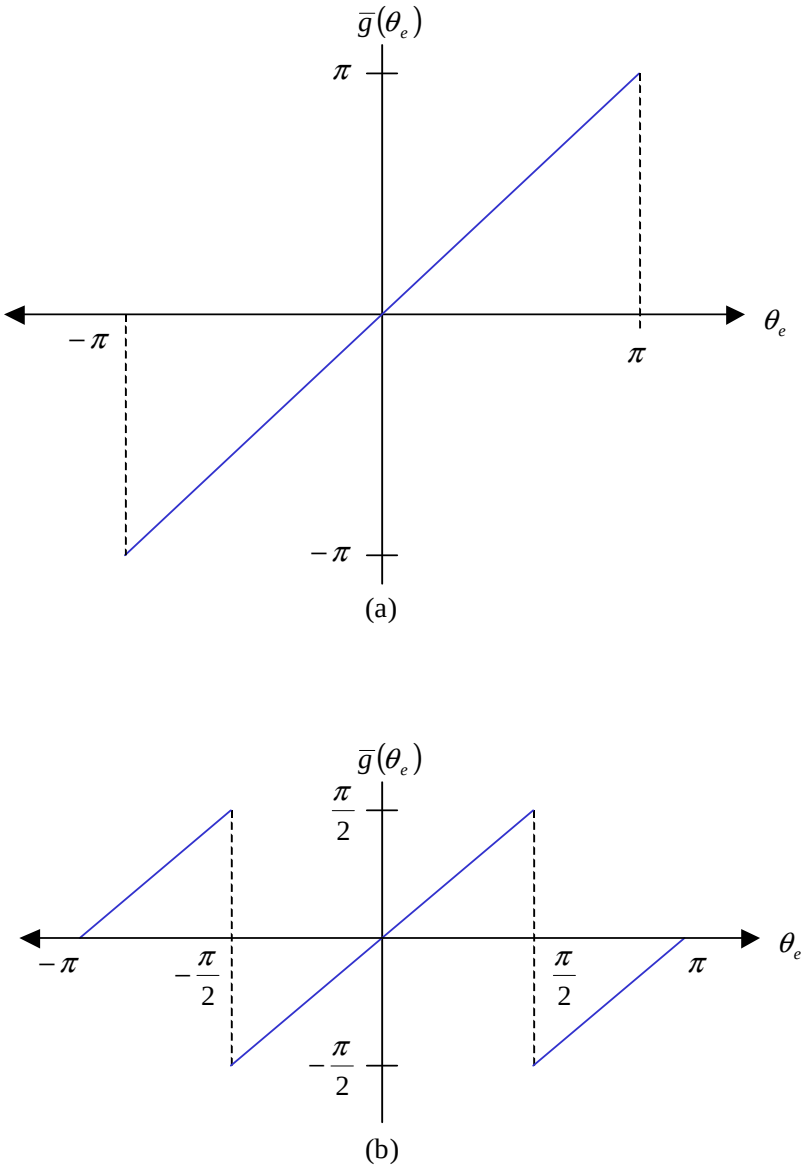


Figure 3.19: S-curves for the data-aided BPSK phase detector (3.53) (a) and the decision-directed BPSK phase detector (3.54).

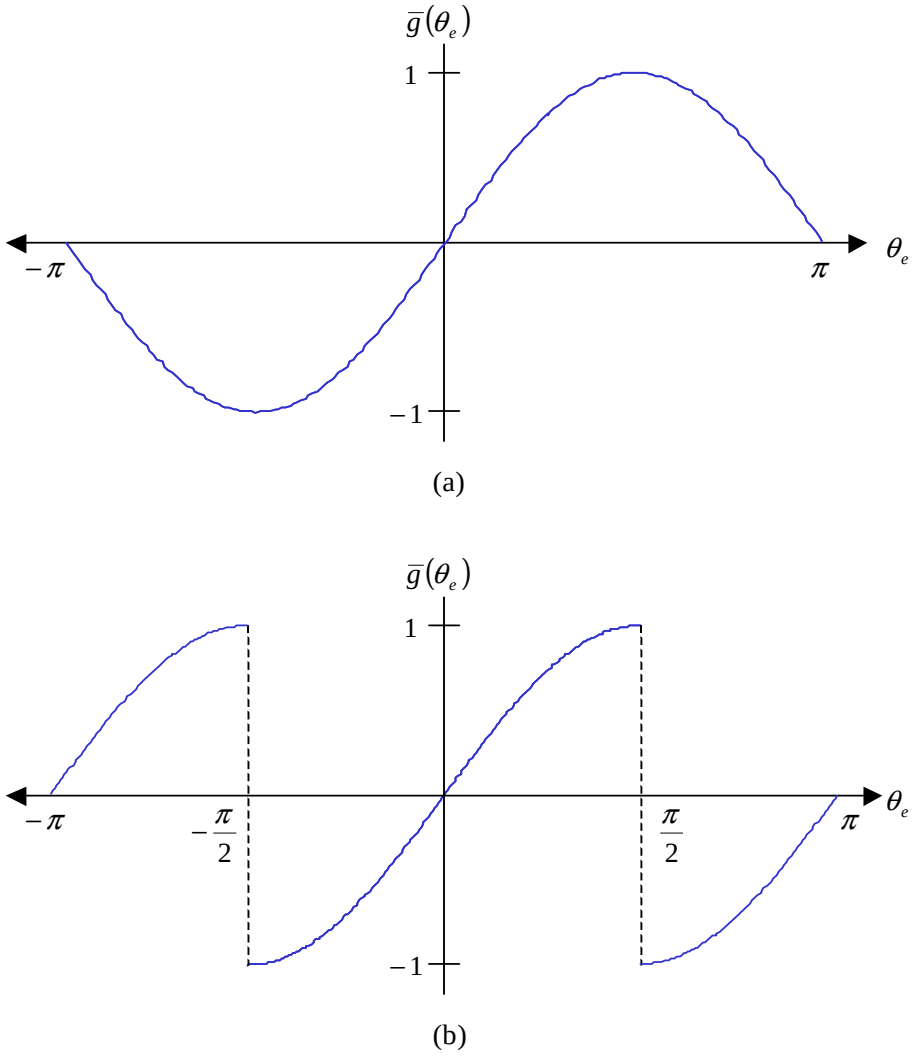


Figure 3.20: S-curves for the data-aided BPSK phase detector (3.57) (a) and the decision-directed BPSK phase detector (3.58).

3.3.3 Carrier Phase Synchronization for MQASK

The carrier phase synchronization for the general case of M-ary QASK can be based on either of the architectures illustrated in Figures 3.7 and 3.8. Following the same line of reasoning associated with Equations (3.34) through (3.36), the ideal linear phase detector for the general MQASK case is (3.36) except that $a_1(k)$ and $a_2(k)$ are no longer restricted to the set $\{-1, +1\}$. In fact, either of the the carrier phase PLLs illustrated in Figures 3.9 and 3.12 for QPSK can be used for MQASK in general. The decision block changes as a function of the constellation. The decision-directed phase detector takes the form (3.38) instead of (3.39) since the data symbols are confined to $a_1(k) \in \{-1, +1\}$ and $a_2(k) \in \{-1, +1\}$ for the general case.

The reduced complexity phase detector follows from the same line of reasoning applied to (3.41) – (3.43). The data-aided phase detector is

$$e(k) = y'(kT_s)a_1(k) - x'(kT_s)a_2(k). \tag{3.59}$$

The decision directed phase detector is obtained by replacing $a_1(k)$ and $a_2(k)$ in (3.59) with the decisions $\hat{a}_1(k)$ and $\hat{a}_2(k)$:

$$e(k) = y'(kT_s)\hat{a}_1(k) - x'(kT_s)\hat{a}_2(k). \tag{3.60}$$

Note that in all cases, $x'(kT_s)$ and $y'(kT_s)$ are used to compute the error signal when the architecture of Figure 3.7 is used while $x(kT_s)$ and $y(kT_s)$ are used in place of $x'(kT_s)$ and $y'(kT_s)$ when the architecture of Figure 3.8 is used.

Even though the block diagrams of the MQASK carrier phase PLLs are identical to the QPSK carrier phase PLLs, the properties of S-curves are strongly dependent on the constellation. For example, consider the S-curves for the 8-PSK, square 16-QASK, and CCITT V.29 16-QASK constellations using the phase detector (3.60) plotted in Figures 3.21, 3.22, and 3.23, respectively. Note that while the S-curves are different, they do have a few features in common. First, each S-curve crosses zero at $\theta_e = 0$ with a positive slope thereby indicating that $\theta_e = 0$ is a stable lock point for the PLL. Second, each S-curve is approximately linear for $\theta_e \approx 0$. Finally, each S-curve possess multiple stable lock points. The number of stable lock points and the values of θ_e where they occur is determined by the rotational symmetry of the constellation.

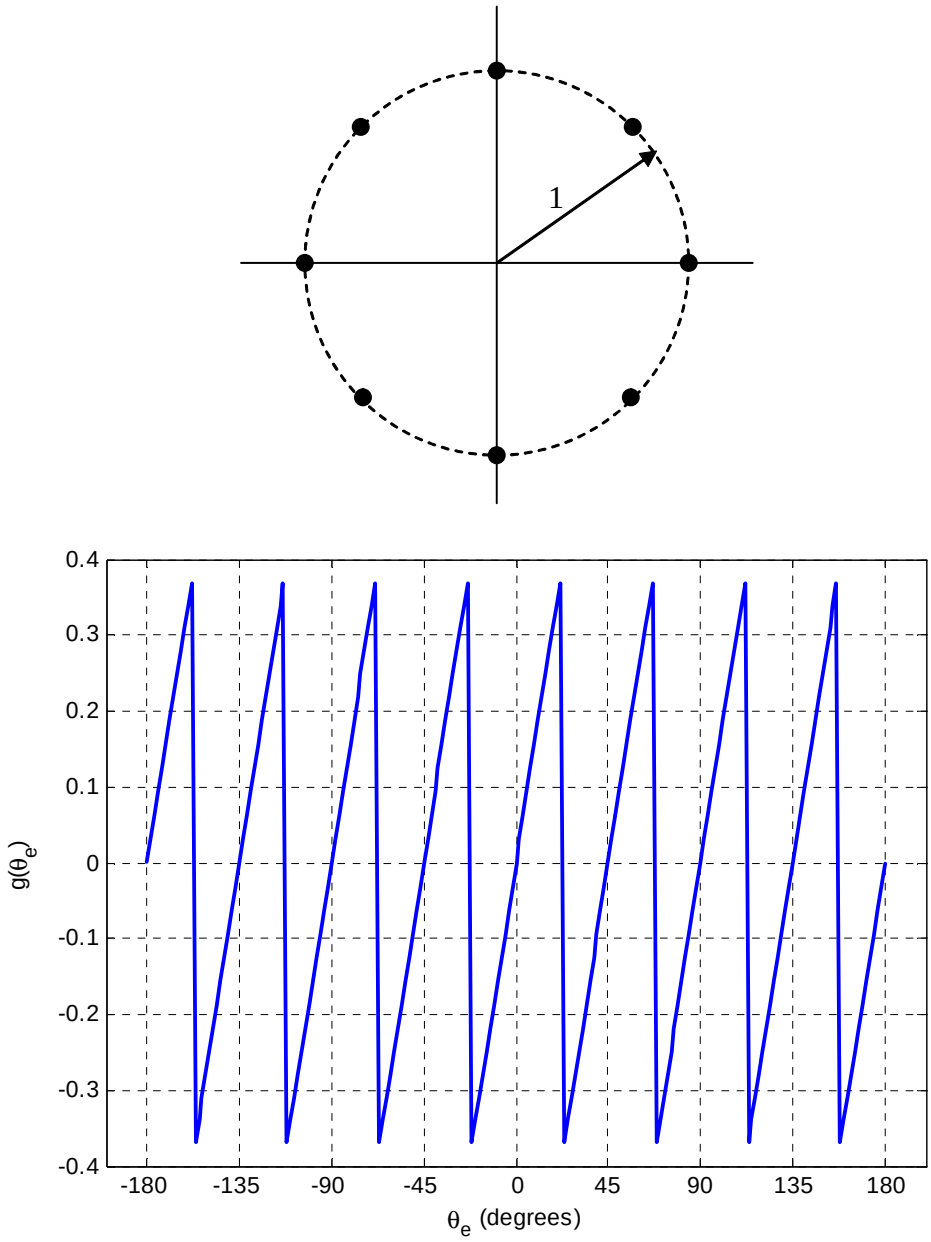


Figure 3.21: S-curve for the square 8-PSK constellation sing the phase detector (3.60).

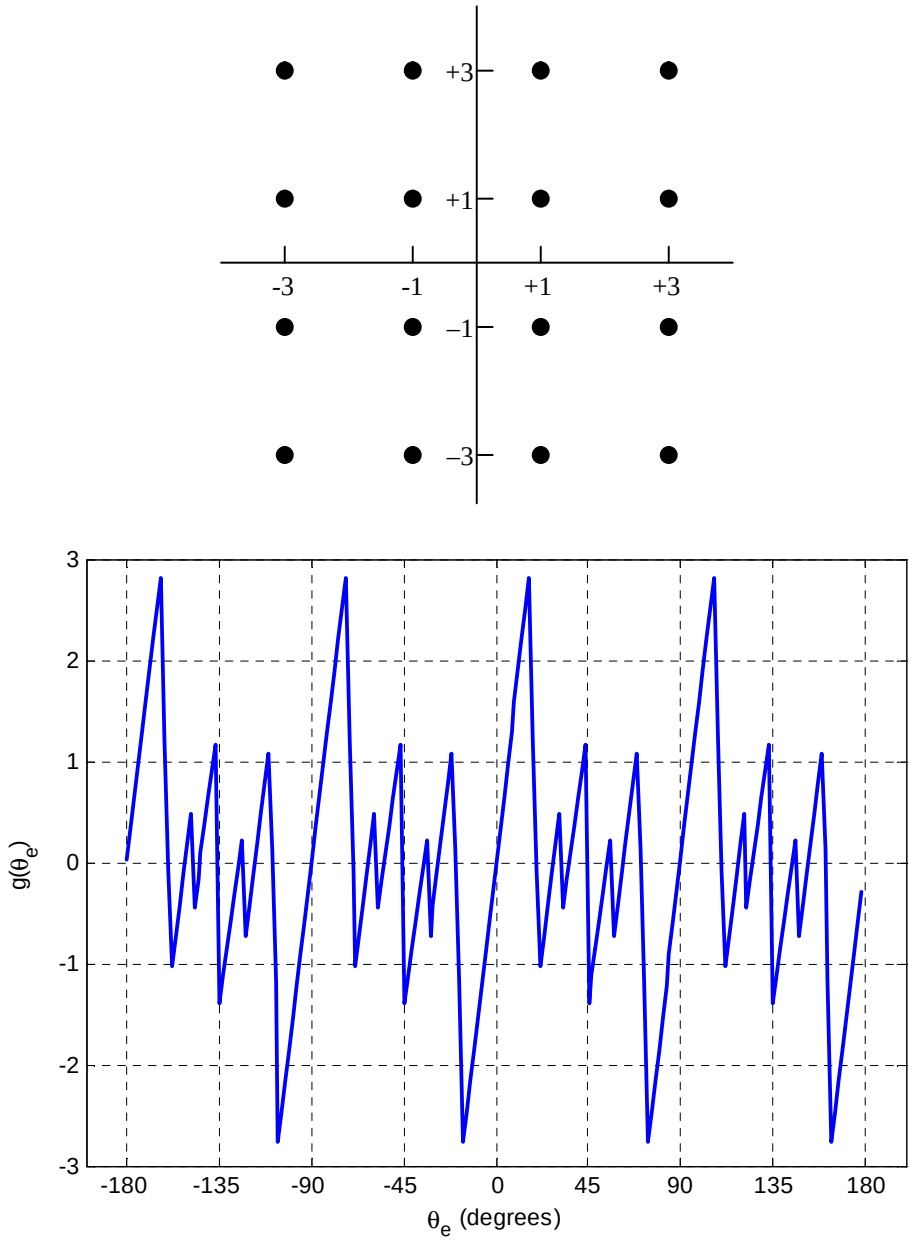


Figure 3.22: S-curve for the square 16-QASK constellation sing the phase detector (3.60).

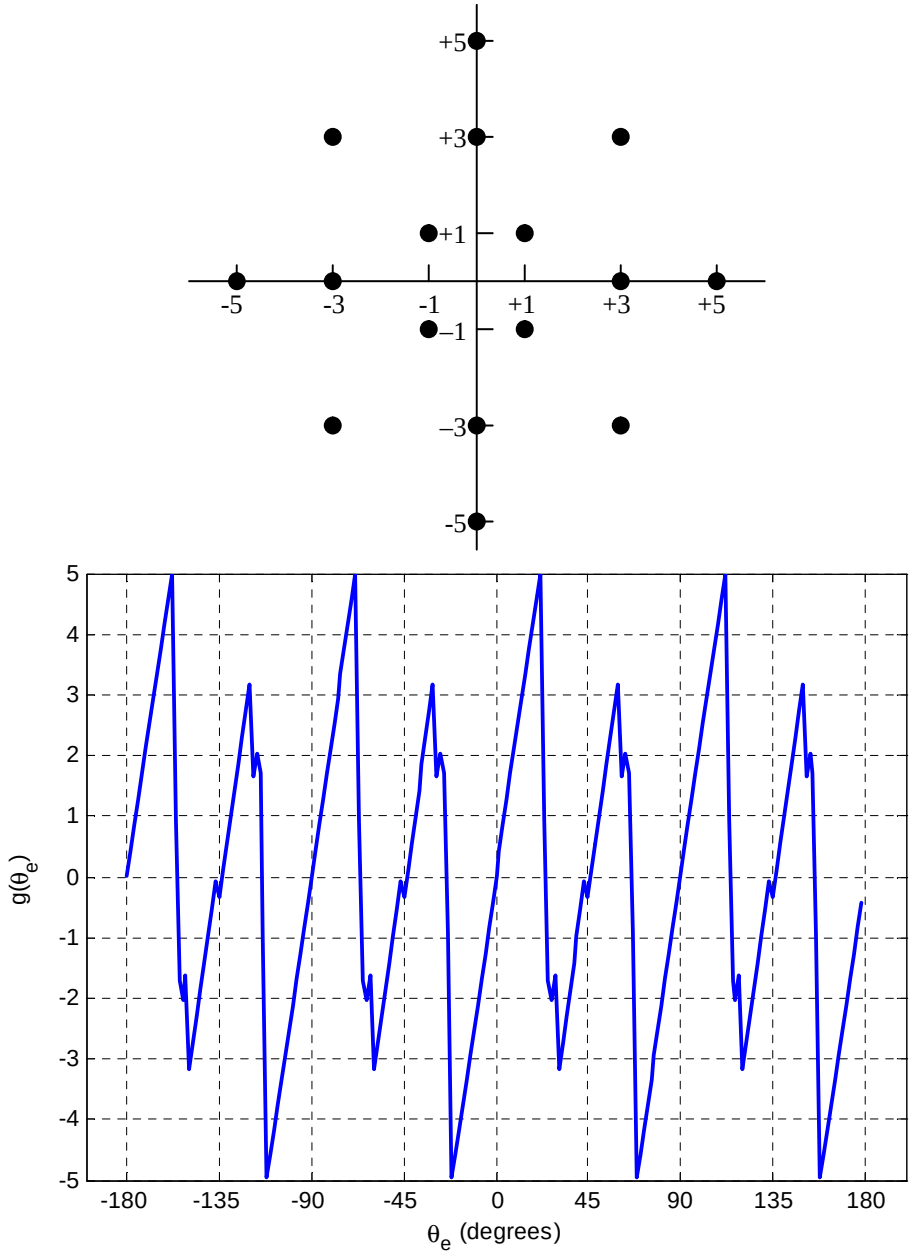


Figure 3.23: S-curve for the CCITT V.29 16-QASK constellation using the phase detector (3.60).

3.3.4 Carrier Phase Synchronization for Offset QPSK

Either of the carrier phase PLL basic structures introduced Figures 3.7 and 3.8 may be applied to to carrier phase synchronization using offset QPSK. The application requires one important modification: the matched filters are sampled at two samples/symbol since the quadrature component of the transmitted signal is delayed by half a symbol period.

The offset QPSK carrier phase PLL counterpart to Figure 3.7 is illustrated in Figure 3.24. Samples of the bandlimited IF signal are downconverted using a free running oscillator. The resulting inphase and quadrature signals, $I(nT)$ and $Q(nT)$, respectively, are filtered by matched filters. The inphase and quadrature matched filter outputs, $x(nT)$ and $y(nT)$, respectively, are sampled at 2 samples/symbol with perfect timing synchronization. For convenience, the samples are indexed by the symbol index k. The optimum sampling instants for the inphase component are kT_s while the optimum sampling instants for the quadrature component are $kT_s + T_s/2$ for $k = 0, 1, \ldots$. The signals $\ldots, x(kT_s), x(kT_s + T_s/2), \ldots$ and $\ldots, y(kT_s), y(kT_s + T_s/2), \ldots$, are rotated by an angle $-\hat{\theta}(k)$ to produce $\ldots, x'(kT_s), x'(kT_s + T_s/2), \ldots$ and $\ldots, y'(kT_s), y'(kT_s + T_s/2), \ldots,$. Using the index notation just described, $x'(kT_s)$ and $y'(kT_s + T_s/2)$ are used for detection. As shown below, $x'(kT_s + T_s/2)$ and $y'(kT_s)$ are used to compute the carrier phase error signal.

Let the samples of the IF signal be given by

$$r(nT) = \sum_m a_1(m)p(nT - mT_s)\cos(\Omega_0 n + \theta) - \sum_m a_2(m)p(nT - mT_s)\sin(\Omega_0 n + \theta) \quad (3.61)$$

where $1/T$ is the sample rate, $a_1(m) \in \{-1, +1\}$ and $a_2(m) \in \{-1, +1\}$ are the information symbols, $p(nT)$ is a unit energy pulse shape with support on the interval $-L_pT_s/T < n < L_pT_s/T$, Ω_0 is the IF frequency in radians/sample, and θ is the unknown carrier phase offset. The matched filter outputs may be expressed as

$$x(nT) = \sum_m a_1(m)R_p\left(nT - mT_s\right)\cos\theta - \sum_m a_2(m)R_p\left(nT - mT_s - T_s/2\right)\sin\theta \quad (3.62)$$

$$y(nT) = \sum_m a_1(m)R_p\left(nT - mT_s\right)\sin\theta + \sum_m a_2(m)R_p\left(nT - mT_s - T_s/2\right)\cos\theta \quad (3.63)$$

where $R_p\left(u\right)$ is the autocorrelation function of the pulse shape given by (3.82). After some algebra and the application of basic trigonometric identities, the rotated matched filter outputs can be

shown to be

$$x'(kT_s) = a_1(k)\cos(\theta - \hat{\theta}(k)) - \sum_m a_2(m)R_p\left((k-m)T_s - T_s/2\right)\sin(\theta - \hat{\theta}(k)) \tag{3.64}$$

$$y'(kT_s) = a_1(k)\sin(\theta - \hat{\theta}(k)) + \sum_m a_2(m)R_p\left((k-m)T_s - T_s/2\right)\cos(\theta - \hat{\theta}(k)) \tag{3.65}$$

$$x'(kT_s + T_s/2) = \sum_m a_2(m)R_p\left((k-m)T_s + T_s/2\right)\cos(\theta - \hat{\theta}(k)) - a_2(k)\sin(\theta - \hat{\theta}(k)) \tag{3.66}$$

$$y'(kT_s + T_s/2) = \sum_m a_2(m)R_p\left((k-m)T_s + T_s/2\right)\sin(\theta - \hat{\theta}(k)) - a_2(k)\cos(\theta - \hat{\theta}(k)) \tag{3.67}$$

where $R_p\left((k-m)T_s\right) = 0$ for $m \neq k$ is assumed. The terms $y'(kT_s)$ and $x'(kT_s + T_s/2)$ contain the product of a single symbol and the sine of the phase error. Knowledge of the symbol or the symbol estimate can by used to provide the correct sign to the sine of the phase error. The data-aided phase error is thus

$$e(k) = a_1(k)y'(kT_s) - a_2(k)x'(kT_s + T_s/2). \tag{3.68}$$

The decision-directed phase error may be expressed in one of two forms

$$e(k) = \hat{a}_1(k)y'(kT_s) - \hat{a}_2(k)x'(kT_s + T_s/2) \tag{3.69}$$

$$= \text{sgn}\left\{x'(kT_s)\right\}y'(kT_s) - \text{sgn}\left\{y'(kT_s + T_s/2)\right\}x'(kT_s + T_s/2). \tag{3.70}$$

The S-curve for the data-aided phase error detector may be computed by substituting (3.65) and (3.66) for $y'(kT_s)$ and $x'(kT_s + T_s/2)$, respectively, in (3.68). Using $\theta_e = \theta - \hat{\theta}$, $a_1^2(k) = 1$ and $a_2^2(k) = 1$, the S-curve is

$$g(\theta_e) = 2\sin\theta_e + \sum_m \left[a_1(k)a_2(m) - a_2(k)a_1(m)\right]R_p\left((k-m)T_s + T_s/2\right)\cos\theta_e. \tag{3.71}$$

The S-curve is thus the familiar sine of the phase error plus a second term which represents the self noise of this phase error detector. The self noise has an average value of zero if $a_1(m)$ and $a_2(m)$ are uncorrelated. The S-curve for both the data-aided error signal (3.68) and the decision-directed error signal (3.70) are plotted in Figure 3.25. Note that the decision-directed error detector does not have a stable lock point at $\theta_e = \pm\pi/2$ as its non-offset counterpart did. Since stable lock points

are those where the S-curve crosses zero with positive slope, both the data-aided and data-directed phase error detectors have two stable lock points and hence a π-phase ambiguity.

Carrier phase synchronization for offset QPSK can also be accomplished using a tunable DDS at IF as illustrated in Figure 3.26. Using the same notation as before, samples of the inphase and quadrature matched filter outputs may be expressed as

$$x(nT) = \sum_m a_1(m) R_p\left(nT - mT_s\right)\cos(\theta - \hat{\theta}) - \sum_m a_2(m) R_p\left(nT - mT_s - T_s/2\right)\sin(\theta - \hat{\theta}) \tag{3.72}$$

$$y(nT) = \sum_m a_1(m) R_p\left(nT - mT_s\right)\sin(\theta - \hat{\theta}) + \sum_m a_2(m) R_p\left(nT - mT_s - T_s/2\right)\cos(\theta - \hat{\theta}) \tag{3.73}$$

Sampling the matched filter outputs at $n = \frac{kT_s}{T}$ and $n = \frac{kT_s}{T} + \frac{T_s}{2T}$, produces

$$x(kT_s) = a_1(k)\cos(\theta - \hat{\theta}(k)) - \sum_m a_2(m) R_p\left((k-m)T_s - T_s/2\right)\sin(\theta - \hat{\theta}(k)) \tag{3.74}$$

$$y(kT_s) = a_1(k)\sin(\theta - \hat{\theta}(k)) + \sum_m a_2(m) R_p\left((k-m)T_s - T_s/2\right)\cos(\theta - \hat{\theta}(k)) \tag{3.75}$$

$$x(kT_s + T_s/2) = \sum_m a_2(m) R_p\left((k-m)T_s + T_s/2\right)\cos(\theta - \hat{\theta}(k)) - a_2(k)\sin(\theta - \hat{\theta}(k)) \tag{3.76}$$

$$y(kT_s + T_s/2) = \sum_m a_2(m) R_p\left((k-m)T_s + T_s/2\right)\sin(\theta - \hat{\theta}(k)) - a_2(k)\cos(\theta - \hat{\theta}(k)) \tag{3.77}$$

which are identical to (3.64) – (3.67) except that phase compensation is performed at IF through the DDS instead of after the matched filters. Using the same line of reasoning as before, the data-aided error signal is

$$e(k) = a_1(k) y(kT_s) - a_2(k) x(kT_s + T_s/2) \tag{3.78}$$

and the decision-directed phase error is

$$e(k) = \text{sgn}\left\{x(kT_s)\right\} y(kT_s) - \text{sgn}\left\{y(kT_s + T_s/2)\right\} x(kT_s + T_s/2). \tag{3.79}$$

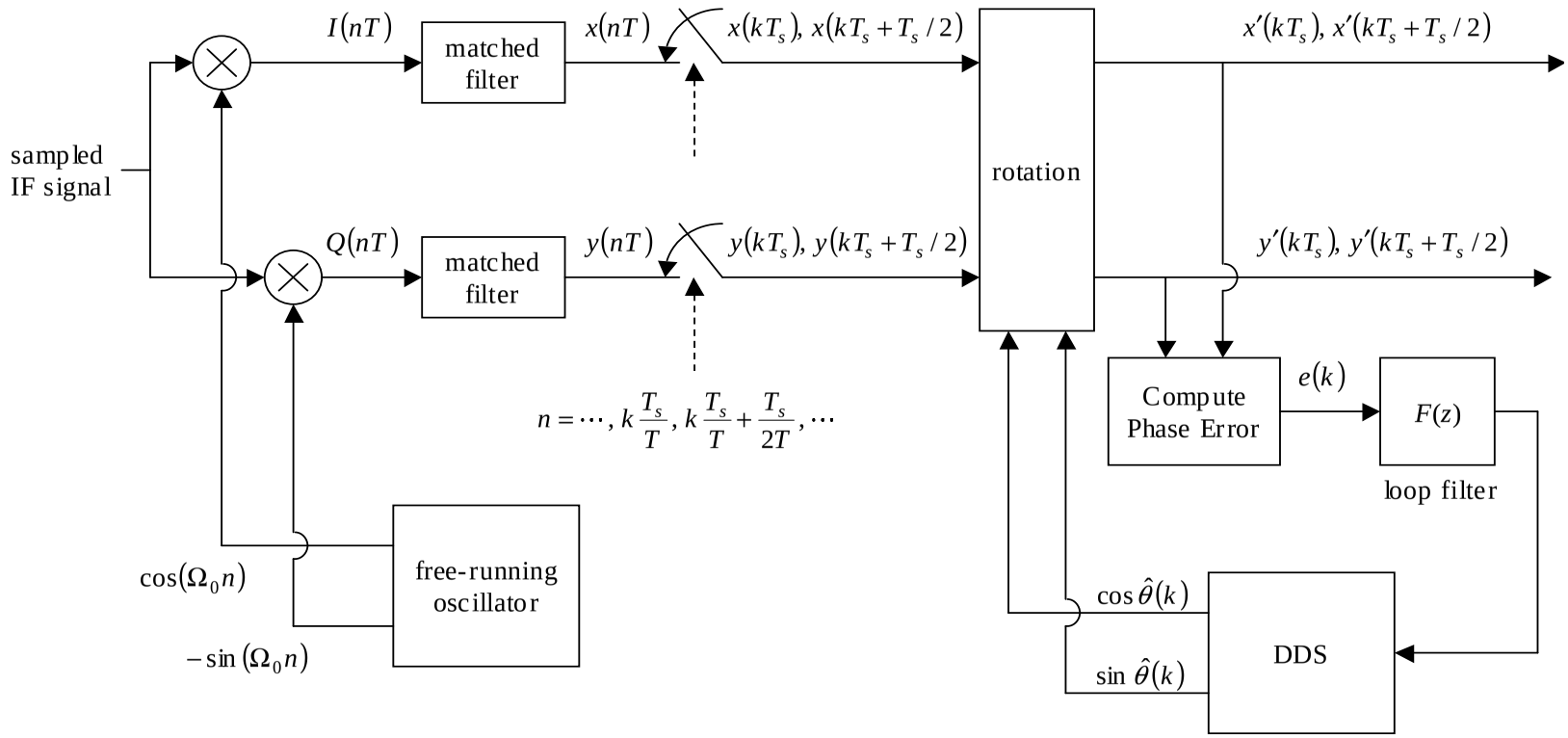


Figure 3.24: Carrier phase synchronization PLL for offset QPSK based on a free-running quadrature LO and a phase rotation. Compare with the carrier phase synchronization system for non-offset MQASK illustrated in Figure 3.7.

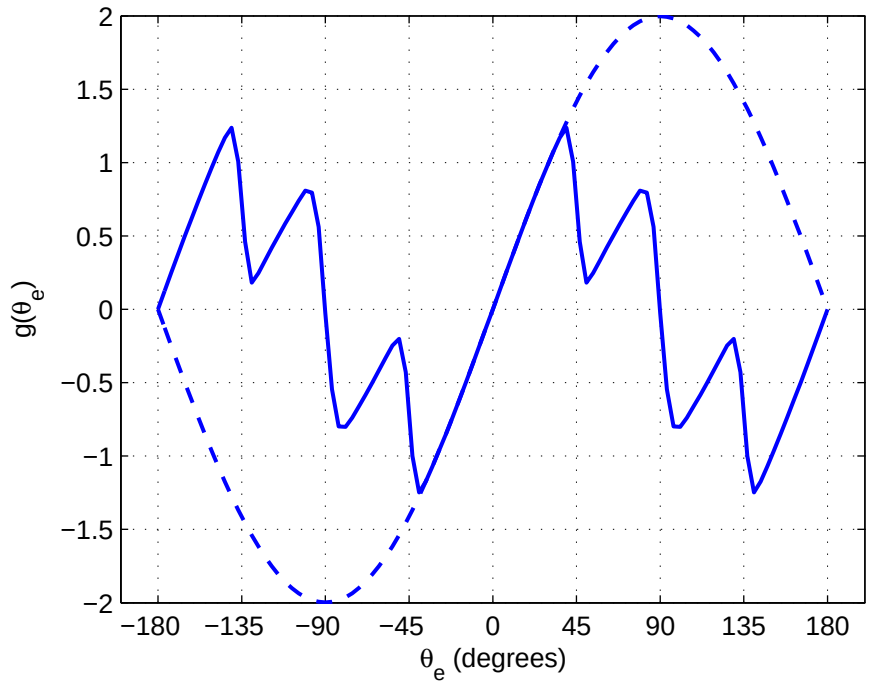


Figure 3.25: S-curves for the data-aided OQPSK phase error detector (dashed line) and the decision-directed OQPSK phase error detector (solid line) for $E_b/N_0 = 20$ dB.

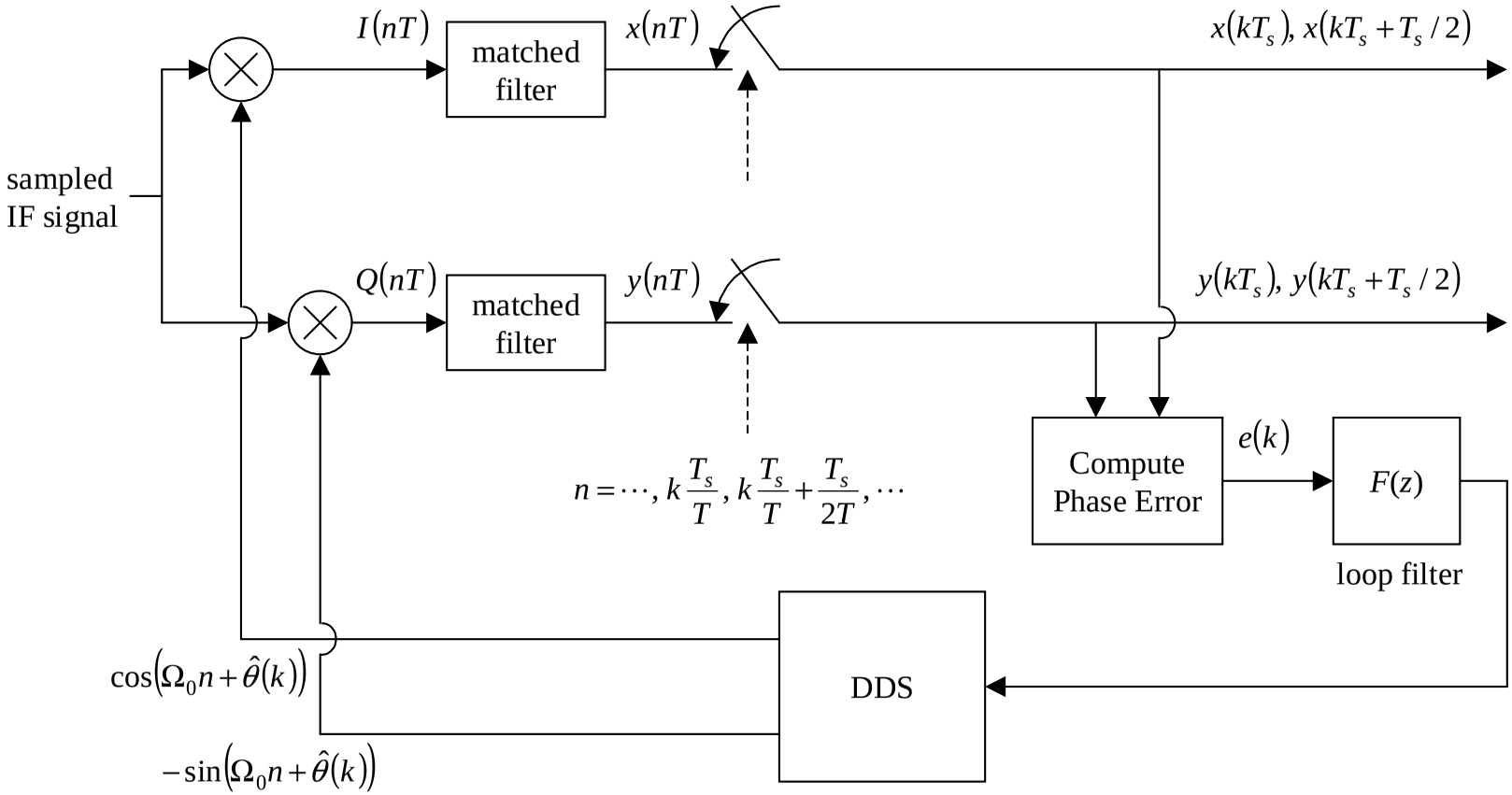


Figure 3.26: Carrier phase synchronization PLL for offset QPSK based on an IF DDS. Compare with the carrier phase synchronization system for non-offset MQASK illustrated in Figure 3.8.

3.3.5 Carrier Phase Synchronization for BPSK and QPSK Using Continuous-Time Techniques

Carrier phase PLLs using continuous-time processing almost always use the architecture illustrated in Figure 3.8 due to the difficulty of constructing a baseband VCO required by the architecture of Figure 3.7. Figure 3.27 illustrates a hybrid architecture using both continuous-time and discrete-time processing for QPSK carrier phase synchronization. The phase detector is a discrete-time processor that updates the carrier phase offset once per symbol using the sequence of matched filter outputs. Any of the data-aided or decision-directed phase error signals for QPSK or BPSK introduced above can be used here. The sequence of phase errors is converted to a continuous-time signal by the digital-to-analog converter. The converted signal forms the input to the continuous-time loop filter $F(s)$ which drives a continuous-time VCO. The sinusoidal output of the VCO is split into quadrature sinusoids using a phase shifter (usually a delay element). The quadrature sinusoids are used to downconvert the IF signal and to separate the IF signal into its quadrature components. The quadrature components are matched filtered and sampled to produce the signal space projection used by the phase detector to update the phase error.

A purely "analog" solution to QPSK carrier phase synchronization is illustrated in Figure 3.28. This structure is called a *Costas loop*. The phase error computation involves the difference between the cross products of the baseband inphase and quadrature signals and their signs. This structure — which results from a recursive solution to maximum likelihood phase estimation as shown in Section 3.6 — is reminiscent of the simplified decision directed phase error given by (3.45) and illustrated in Figure 3.13. (Note that the sign on the error signal is switched in Figures 3.13 and 3.28. This is due to the use of $-\sin(\cdot)$ for the quadrature component in Figure 3.13 and $\sin(\cdot)$ for the quadrature component in Figure 3.28.) To see how this works, consider the plot of the baseband inphase and quadrature components plotted in Figure 3.29. During each symbol interval, the signs of the baseband inphase and quadrature components could be used as decisions from which the sine of the phase error can be continuously updated.

The Costas loop for BPSK carrier phase synchronization is illustrated in Figure 3.30. Note that this structure is reminiscent of the discrete-time BPSK carrier phase PLL based on the phase error given by (3.56) where the sign of the baseband inphase component plays the role of the decision. As with QPSK, a sign reversal (not shown) is present due to the use of $-\sin(\cdot)$ in discrete-time carrier phase PLLs using a quadrature DDS and $\sin(\cdot)$ in Figure 3.30.

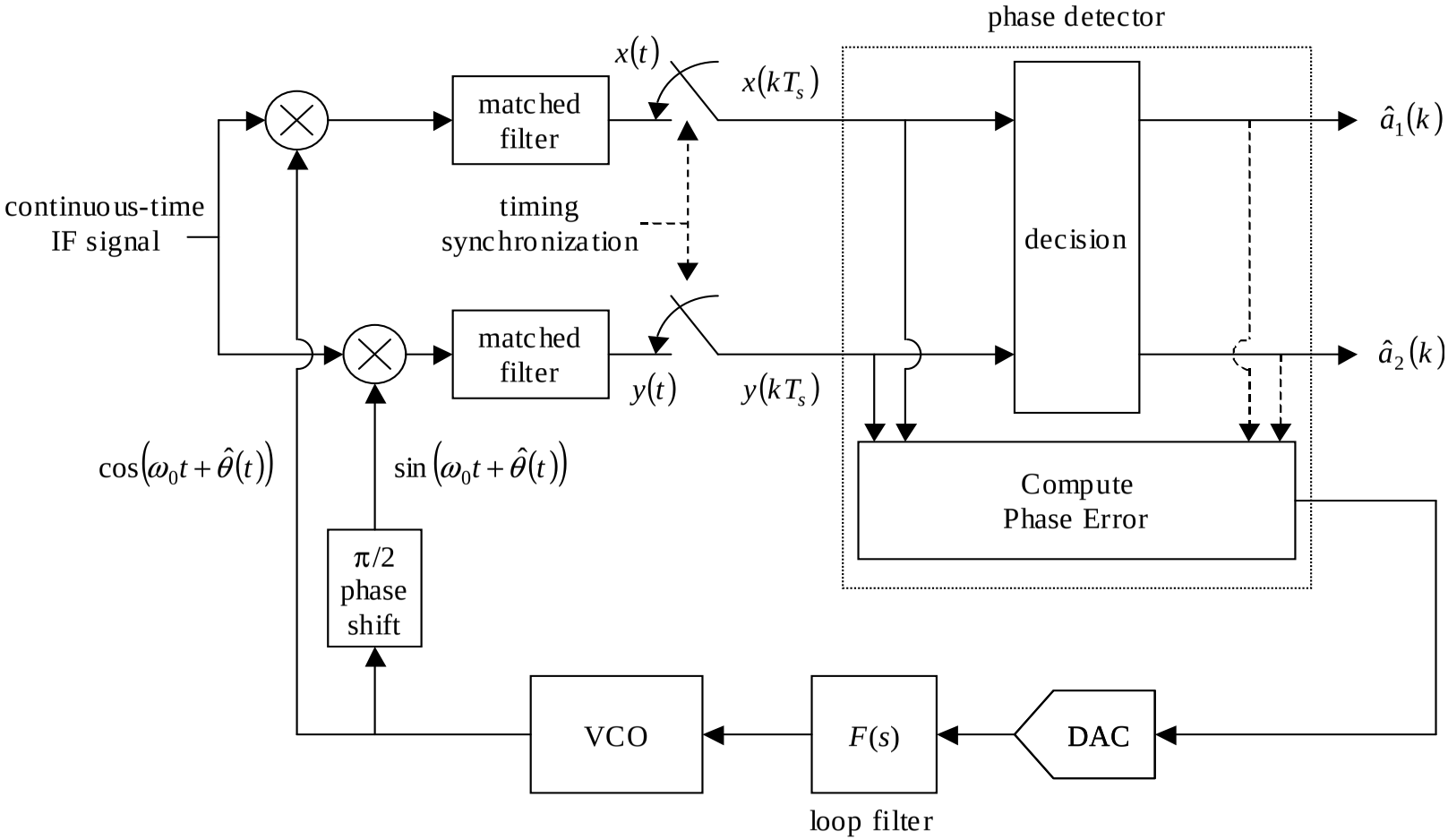


Figure 3.27: A hybrid continuous-time/discrete-time carrier phase PLL for QPSK.

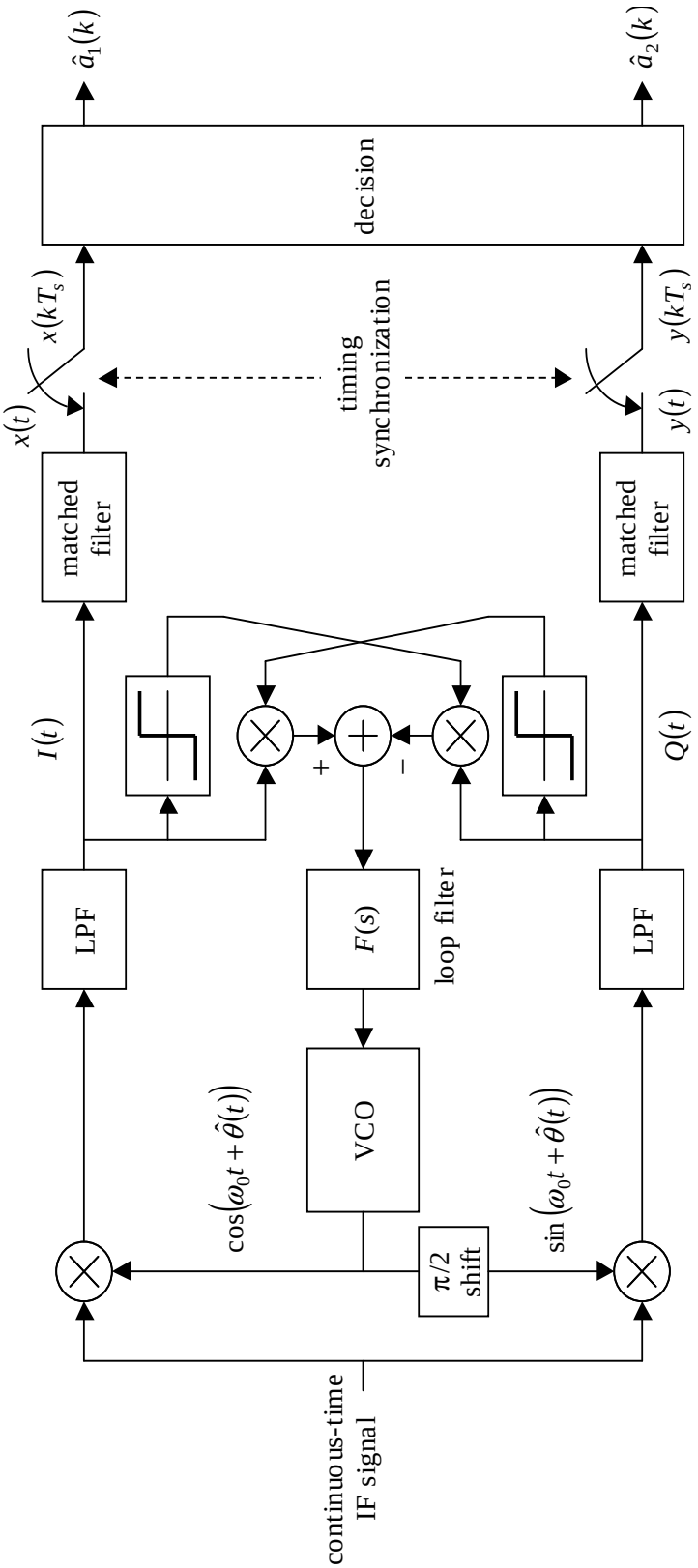


Figure 3.28: A QPSK carrier phase PLL based on the Costas loop.

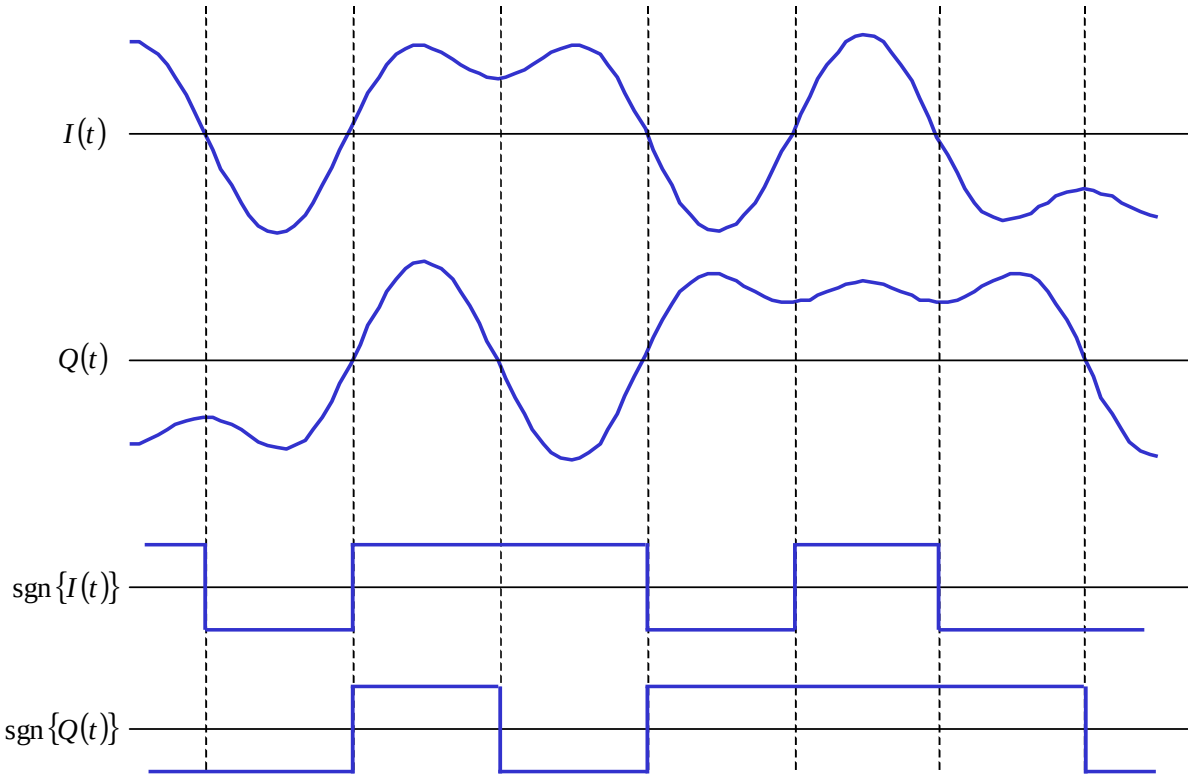


Figure 3.29: Example of baseband inphase and quadrature signal components, $I(t)$ and $Q(t)$, respectively, and their signs used by the QPSK Costas loop shown in 3.28.

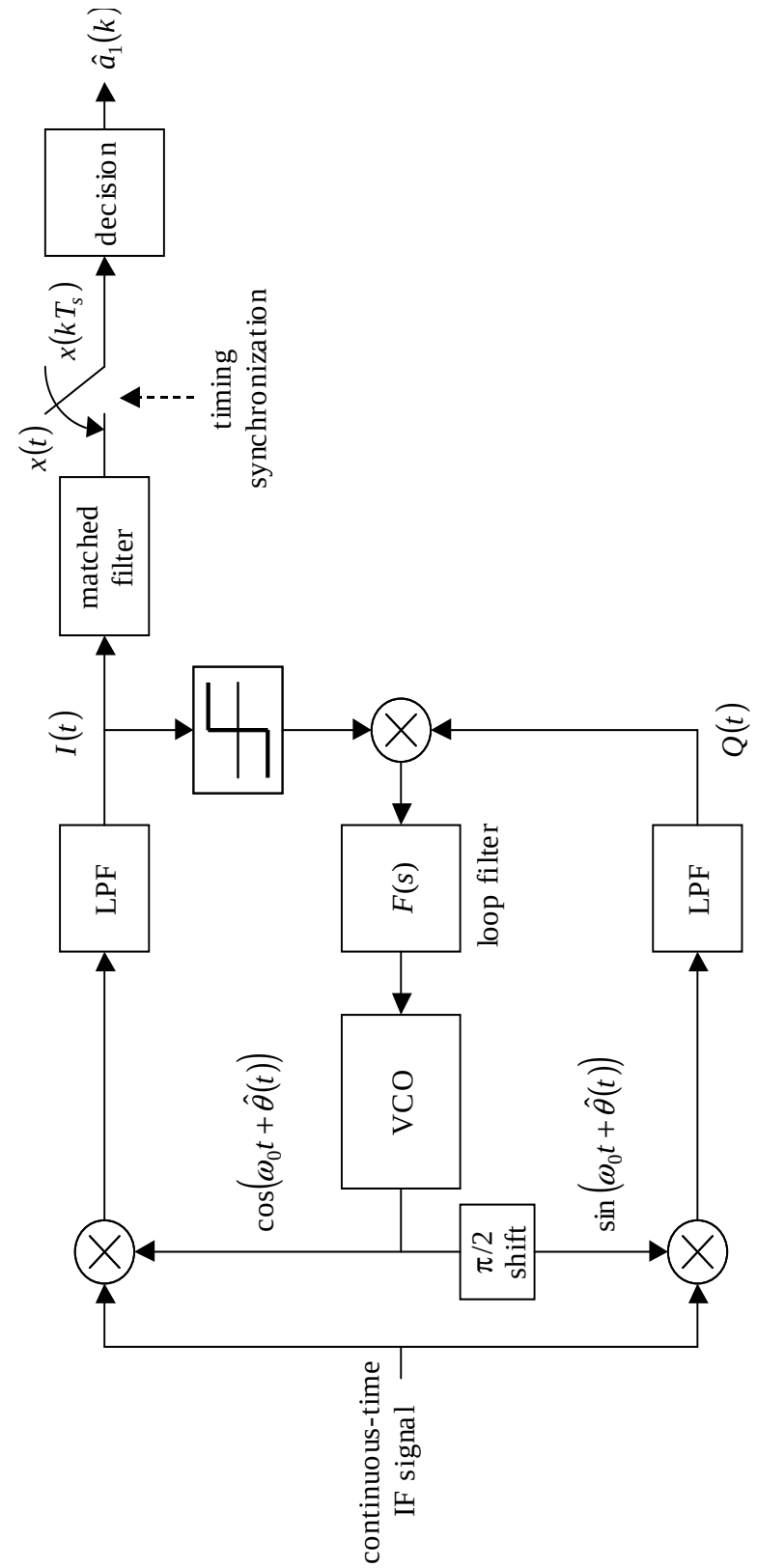


Figure 3.30: A BPSK carrier phase PLL based on the Costas loop.

3.4 Symbol Timing Synchronization

Symbol timing synchronization is the process of estimating a clock signal that is aligned in phase and frequency with the clock used to generate the data at the transmitter. Since it is not efficient to allocate spectrum to transmit a separate clock signal from the transmitter to the receiver for the purposes of timing synchronization, the data clock must be extracted from the noisy received waveforms that carry the data. For matched filter detectors, the clock signal is used to identify the instants when the matched filter should be sampled. The effect of timing errors on matched filter receivers for MQASK were introduced earlier.

The form of the symbol timing synchronizer is quite different for continuous-time and discrete-time systems and is perhaps one of the biggest differences between the two implementations. Continuous-time techniques are reviewed in Section 3.4.1. Discrete-time techniques for symbol timing synchronization are covered in detail in Section 3.4.3. In both cases, symbol timing synchronization for M-ary PAM are developed. Extensions to MQASK are described in Sections 3.4.2 and 3.4.4

3.4.1 Continuous-Time Techniques for M-ary PAM

Figure 3.31 shows a block diagram of the basic architecture for symbol timing synchronization for M-ary PAM using continuous-time techniques. Let the received M-ary PAM signal be

$$r(t) = \sum_k a(k)p(t - kT_s - \tau) + w(t) \tag{3.80}$$

where $a(k) \in \{-(M-1), -(M-3), \ldots, -1, 1, \ldots, M-3, M-1\}$ is the k-th PAM symbol, T_s is the symbol time, τ is the unknown timing delay, $p(t)$ is a unit energy pulse shape with support on the interval $-L_pT_s \leq t \leq L_pT_s$, and $w(t)$ is the additive white Gaussian noise. The received signal is passed through a matched filter whose impulse response is $p(-t)$. The output of the matched filter $x(t)$ may be expressed as

$$x(t) = \sum_k a(k)R_p(t - kT_s - \tau) + v(t) \tag{3.81}$$

where $R_p(u)$ is the autocorrelation function of the pulse shape defined by

$$R_p(u) = \int_{-L_pT_s}^{L_pT_s} p(t)p(t-u)dt \tag{3.82}$$

and $v(t) = w(t) * p(-t)$ represents the noise at the output of the matched filter. Ideally, the matched filter output should be sampled at $t = kT_s + \tau$ for detection. This is easy if τ is known. When τ is not known, it must be estimated. This is the role of the symbol timing synchronizer.

Using the estimate $\hat{\tau}$ provided by symbol timing synchronizer, the matched filter output at $t = kT_s + \hat{\tau}$ is

$$x(kT_s + \hat{\tau}) = a(k)R_p(\hat{\tau} - \tau) + \sum_{m \neq k} a(m)R_p\left((k-m)T_s + \hat{\tau} - \tau\right). \tag{3.83}$$

In the following, it will be convenient to express this output in terms of the timing error $\tau_e = \tau - \hat{\tau}$:

$$x(kT_s + \hat{\tau}) = a(k)R_p(-\tau_e) + \sum_{m \neq k} a(m)R_p\left((k-m)T_s - \tau_e\right). \tag{3.84}$$

Note that when the pulse shape satisfies the Nyquist condition for no ISI, the second term is zero for $\tau_e = 0$.

In a continuous-time detector, the goal of the symbol timing is to produce a clock signal aligned with the data transitions as illustrated in Figure 3.31 for binary PAM. In this example, the rising edge of the clock is aligned with the symbol transitions and is used to trigger the sample-and-hold operation at the matched filter output. The symbol timing PLL in Figure 3.31 consists of a timing error detector, loop filter, and voltage controlled clock (VCC) arranged as shown in Figure 3.32. The timing error detector computes the phase error between the VCC output and the clock signal embedded in the matched filter outputs. The loop filter controls the nature of the loop response and the VCC plays the role of the VCO (i.e. the VCC output is a clock signal whose instantaneous frequency is proportional to the VCC input).

The operation of the timing error detector is best understood using the eye diagram. Figure 3.33 illustrates this concept for binary PAM. Observe that the optimum sampling instant coincides with the time instant of maximum average eye opening. The time instant of maximum eye opening occurs at the time instant where the average slope of the eye diagram is zero. The non-zero slope at $t = \tau$ are points in trajectories corresponding to no data sign transition followed by a data sign transition or a data sign transition followed by no data sign transition. This feature reinforces the fact that symbol timing synchronizers rely on data sign transitions to produce a proper timing error signal.

The forgoing demonstrates that the slope of the eye diagram can be used to generate a timing error. Figure 3.34 demonstrates that the sign of the slope of the eye must be qualified to provide the correct timing error. Figure 3.34 (a) illustrates the case where the current sampling instant is early (i.e., $\hat{\tau}(k) < \tau$ which means $\tau_e(k) > 0$) and the data symbol $a(k) = +1$. The next sampling instant $\hat{\tau}(k+1)$ should be greater than $\hat{\tau}(k)$. This is accomplished by increasing the period of the VCC output. The slope of the matched filter output at $t = \hat{\tau}(k)$ is positive and can be used as the error signal. Note that this applies only to the part of the eye corresponding to a transition from $a(k-1) = -1$ to $a(k) = +1$. The approximately horizontal portion of the eye diagram just above

the end of the arrow corresponds to the case where there is no data transition (i.e. $a(k-1) = +1$). The slope of the eye diagram is very small along this trajectory. As a consequence, little timing error information is provided in the absence of a data transition. This property will be more obvious in the examples to come.

Figure 3.34 (b) demonstrates the case where the current sampling instant is late (i.e., $\hat{\tau}(k) > \tau$ or $\tau_e < 0$) and the data symbol $a(k) = +1$. In this case, the period of the VCC should be decreased to force $\hat{\tau}(k+1) < \hat{\tau}(k)$. The slope of the eye at $t = \hat{\tau}(k)$ along the trajectory from $a(k) = +1$ to $a(k+1) = -1$ is negative and thus indicates the proper adjustment to the VCC period. As before, the trajectory from $a(k) = +1$ to $a(k+1) = +1$ has a very small slope and thus provides little or no timing error information.

In the preceding two cases, the data symbol $a(k) = +1$. Figures 3.34 (c) and (d) demonstrate what happens when $a(k) = -1$. In Figure 3.34 (c), the current sampling instant is early (i.e., $\hat{\tau}(k) < \tau$ or $\tau_e > 0$) and the period of the VCC should be increased to force $\hat{\tau}(k+1) > \hat{\tau}(k)$. Unfortunately, the slope of the eye corresponding to the transition from $a(k-1) = +1$ to $a(k) = -1$ is negative at $t = \hat{\tau}(k)$ and therefor does not indicate the proper adjustment to the VCC period. This is because the data symbol $a(k)$ is negative. If the slope of the eye $t = \hat{\tau}(k)$ is altered by the sign of the data symbol, then a signal that provides the proper adjustment to the VCC period is obtained. Similarly, Figure 3.34 (d) shows the case where the current sampling instant is late (i.e., $\hat{\tau}(k) > \tau$ or $\tau_e < 0$) which requires the $\hat{\tau}(k+1) < \hat{\tau}(k)$ and a decrease in the VCC period. Since the slope of the eye corresponding to the transition from $a(k) = -1$ to $a(k+1) = +1$ at $t = \hat{\tau}(k)$ is positive, the product of the slope and the sign of $a(k)$ provides the proper signal for adjusting the VCC period.

The forgoing observations suggest a timing error signal of the form

$$e(k) = a(k)\dot{x}\left(kT_s + \hat{\tau}(k)\right) \tag{3.85}$$

for the data-aided case, and

$$e(k) = \hat{a}(k)\dot{x}\left(kT_s + \hat{\tau}(k)\right) \tag{3.86}$$

for the decision-directed case where $\dot{x}(t)$ is the time derivative of the matched filter output. As is turns out, this error signal follows from the maximum likelihood estimate for timing offset as outlined in Section 3.6. A block diagram of a symbol timing PLL based on (3.86) is illustrated in Figure 3.35 for binary PAM. Generation of the error signal requires a differentiator connected to the matched filter output. The output of the differentiator is sampled at $t = kT_s + \hat{\tau}(k)$ to provide the slope of the eye at $\hat{\tau}(k)$. The sign of this slope is qualified by the sign of the data by multiplying by the sign of the corresponding sampled matched filter output to form the error signal (3.86).

Figure 3.35 also plots an example of the received waveform $r(t)$ (neglecting noise), the corresponding matched filter output $x(t)$, the derivative of the matched filter output $\dot{x}(t)$, and the product $\text{sgn}\{x(t)\}\,\dot{x}(t)$ whose samples are used as the timing error signal. Observe that the timing error signal $\text{sgn}\{x(t)\}\,\dot{x}(t)$ is zero at the optimum sampling time[5]. In the absence of data transitions the timing error signal is zero throughout the entire symbol interval since the derivative of the matched filter output is zero. This demonstrates that data transitions are necessary to provide sufficient timing error information to obtain symbol timing synchronization. If too many consecutive symbols are the same, the symbol timing PLL could drift out of lock since the timing error is zero.

Often, it is desirable reduce the complexity of the timing error detector by approximating the derivative operation with a difference as illustrated in Figure 3.36 for binary PAM and the case $\hat{\tau}(k) < \tau$ and $a(k) = 1$. Since

$$\dot{x}\left(t_0\right) = \lim_{\Delta \to 0} \frac{x(t_0 + \Delta) - x(t_0 - \Delta)}{2\Delta}, \tag{3.87}$$

the derivative of the matched filter output may by approximated using the difference as shown. Qualifying the sign of the difference by the data produces a timing error detector known as an *early-late gate detector*. The data-aided version of the early-late gate timing error is

$$e(k) = a(k)\left[x\left(kT_s + \hat{\tau}(k) + \Delta T_s\right) - x\left(kT_s + \hat{\tau}(k) - \Delta T_s\right)\right] \tag{3.88}$$

and the decision-directed version is

$$e(k) = \hat{a}(k)\left[x\left(kT_s + \hat{\tau}(k) + \Delta T_s\right) - x\left(kT_s + \hat{\tau}(k) - \Delta T_s\right)\right]. \tag{3.89}$$

For binary PAM, a popular form of the early-late gate error signal that does not rely directly on the decisions is

$$e(k) = \left|x\left(kT_s + \hat{\tau}(k) + \Delta T_s\right)\right| - \left|x\left(kT_s + \hat{\tau}(k) - \Delta T_s\right)\right|. \tag{3.90}$$

A block diagram of a binary PAM timing PLL based on the early-late gate (3.90) is illustrated in Figure 3.37. Also shown are the baseband received signal $r(t)$ (neglecting noise), the corresponding matched filter output $x(t)$, and the signal $\left|x\left(t + \Delta T_s\right)\right| - \left|x\left(t - \Delta T_s\right)\right|$ for $\Delta = 1/4$. Observe that the timing error signal $\left|x\left(t + \Delta T_s\right)\right| - \left|x\left(t - \Delta T_s\right)\right|$ is zero at the optimum sampling instants when there is a data transition. When there is no data transition, the timing error signal is not zero at the optimum sampling instant. This non-zero value is called *self noise* and can limit the accuracy of the timing PLL if the density of data transitions is not sufficiently high.

[5]The timing error signal $\text{sgn}\{x(t)\}\,\dot{x}(t)$ is also zero half-way between the optimum sampling times. This zero crossing is not a stable lock point.

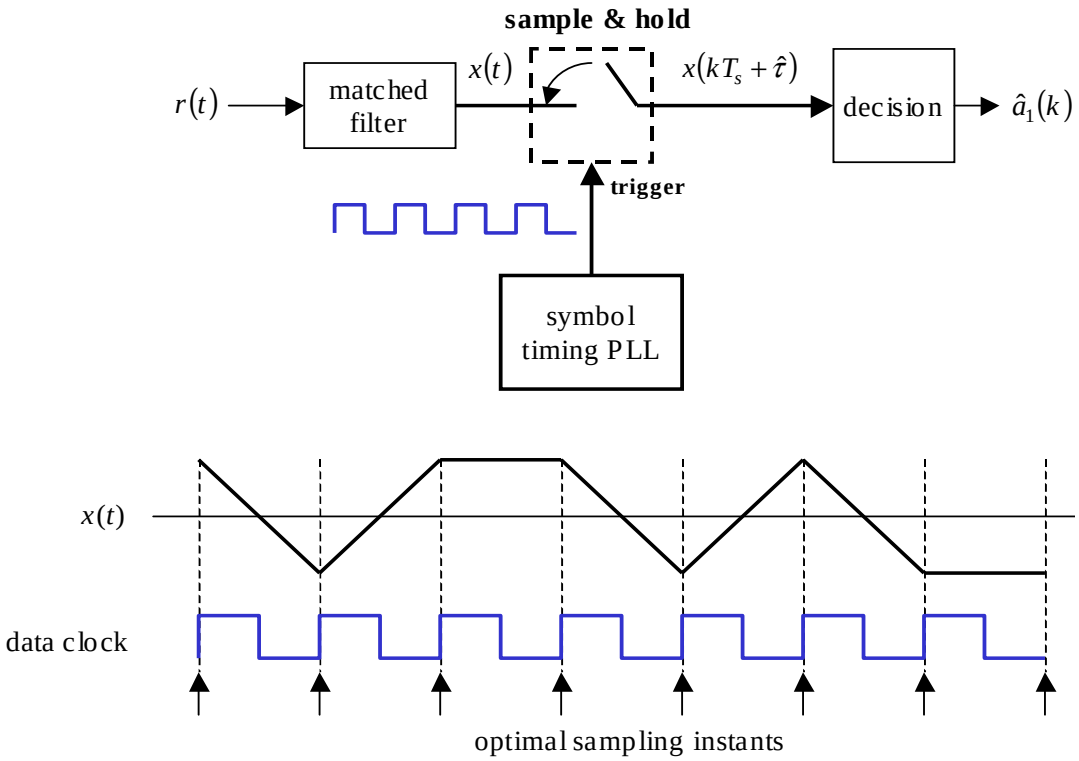


Figure 3.31: Block diagram for a binary PAM detector showing the role of the symbol timing PLL and the relationship between the matched filter output and the symbol timing PLL output.

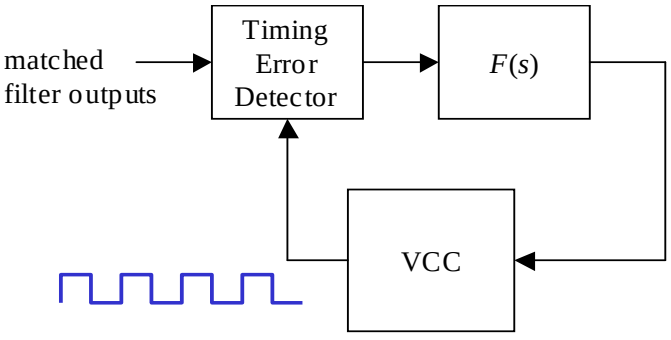


Figure 3.32: The three basic components of an continuous-time symbol timing PLL.

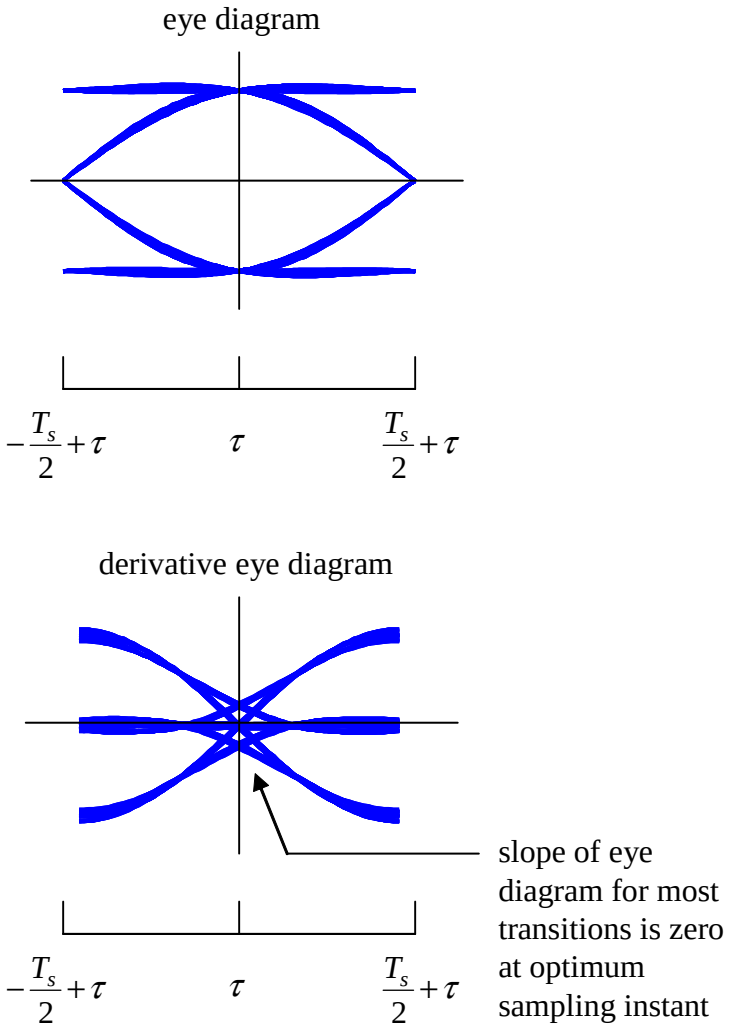


Figure 3.33: Eye diagram demonstrating that the optimum sampling instant and the instant of maximum average eye opening coincide with the instant where the slope of the average eye is 0.

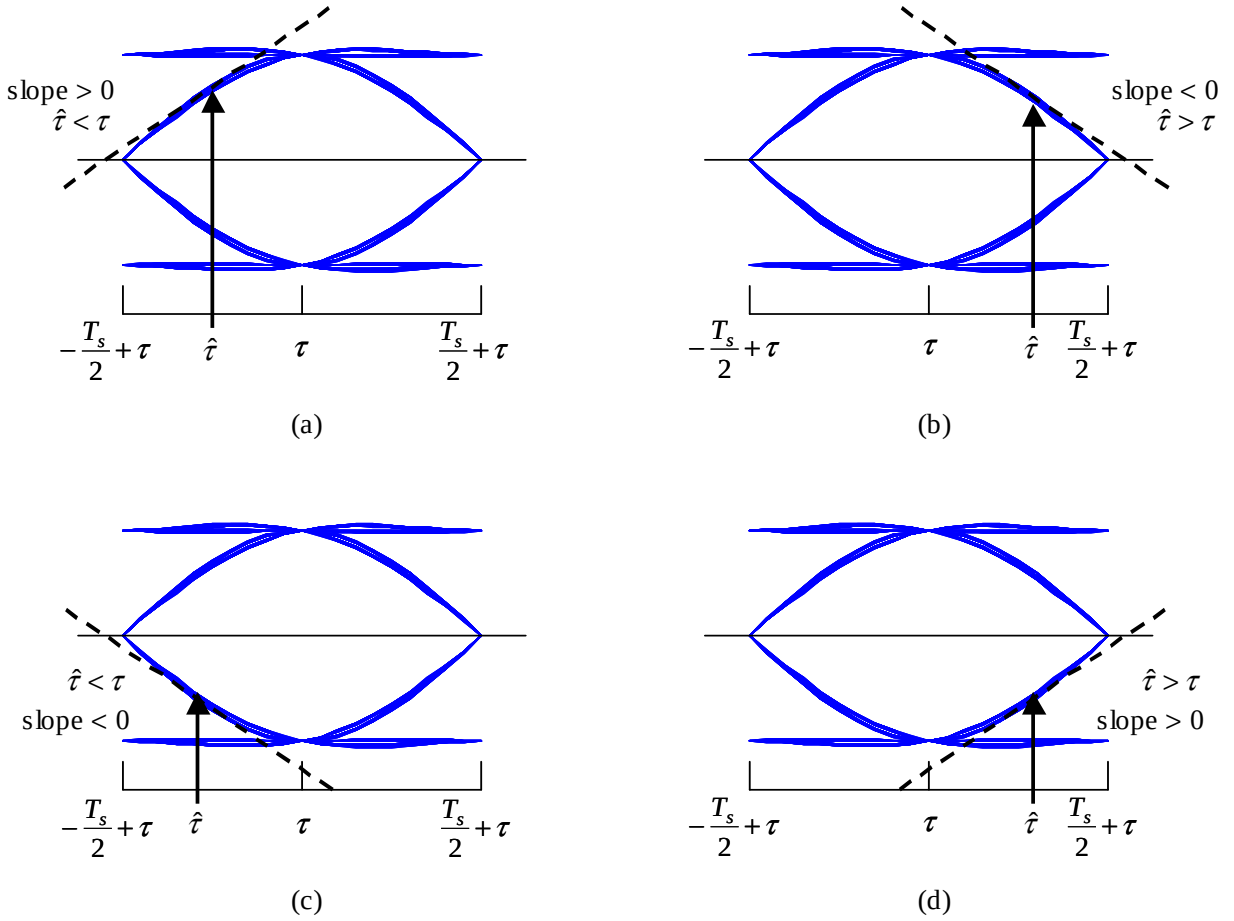


Figure 3.34: The use of the slope of the eye as a timing error signal. (a) The timing estimate is early and the corresponding symbol is positive. (b) The timing estimate is late and the corresponding symbol is positive. (c) The timing estimate is early and the corresponding symbol is negative. (d) The timing estimate is late and the corresponding symbol is negative.

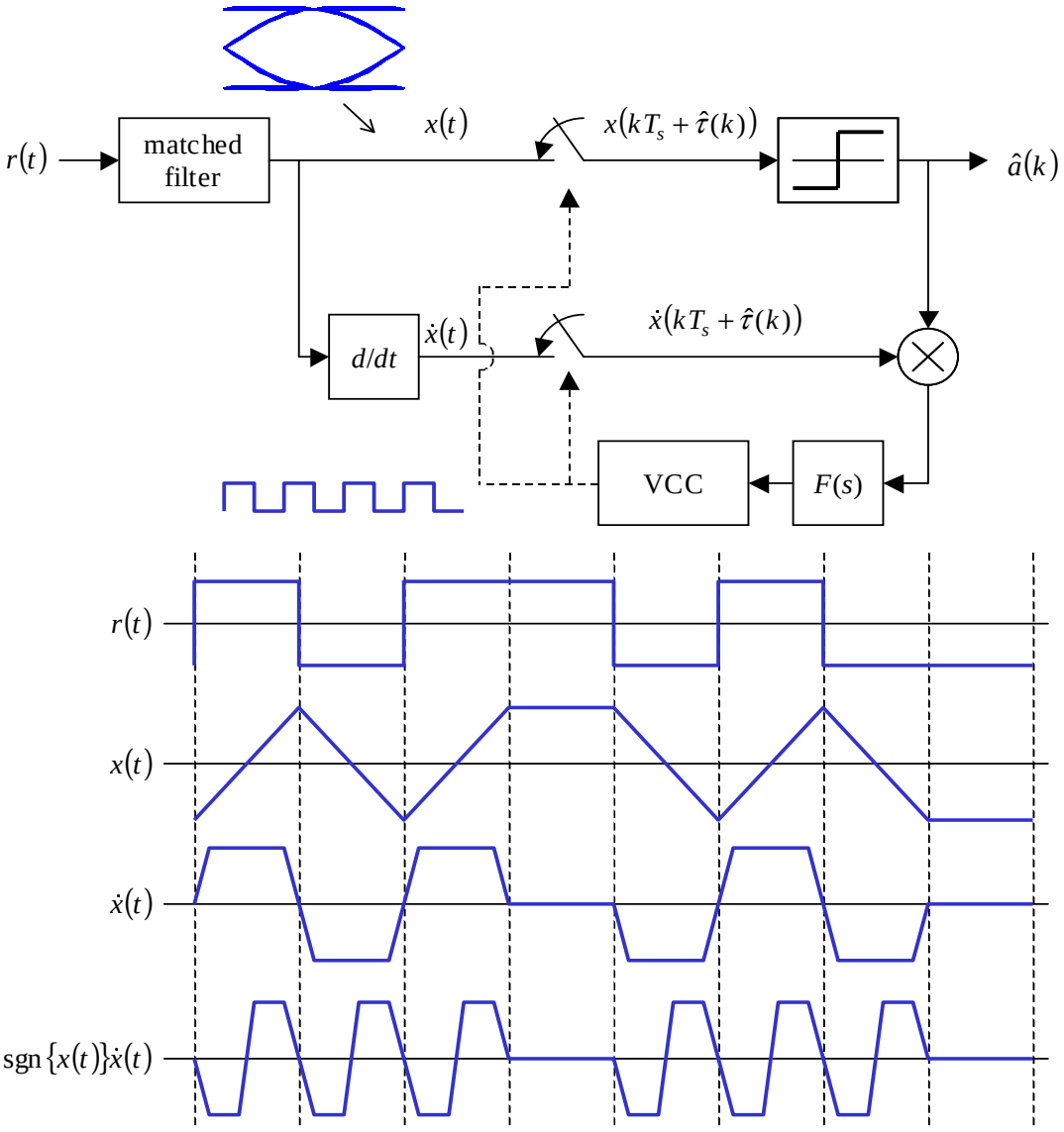


Figure 3.35: Timing PLL for binary PAM showing the detail of the phase detector based on (3.86).

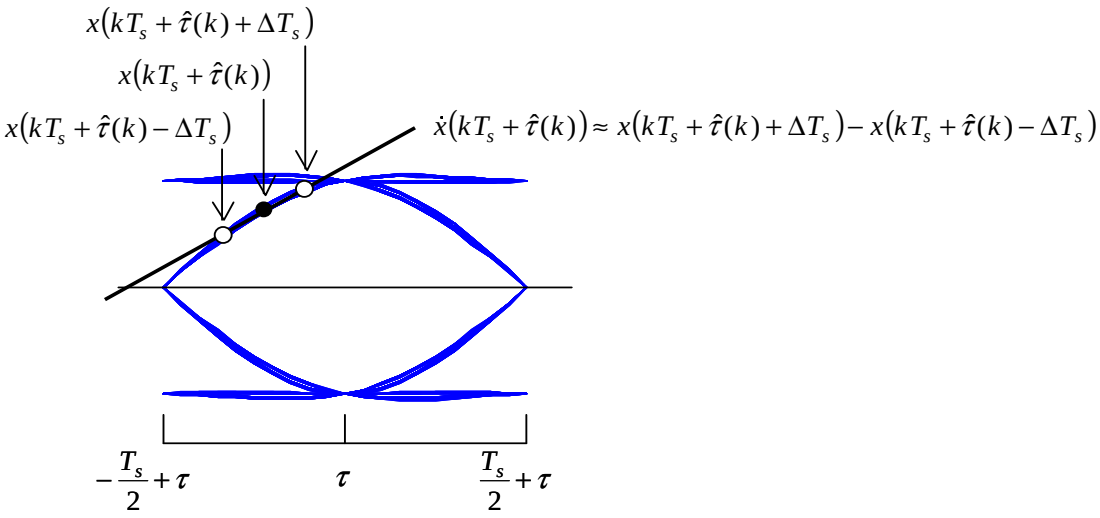


Figure 3.36: Illustration of using a difference to approximate the derivative of the eye diagram.

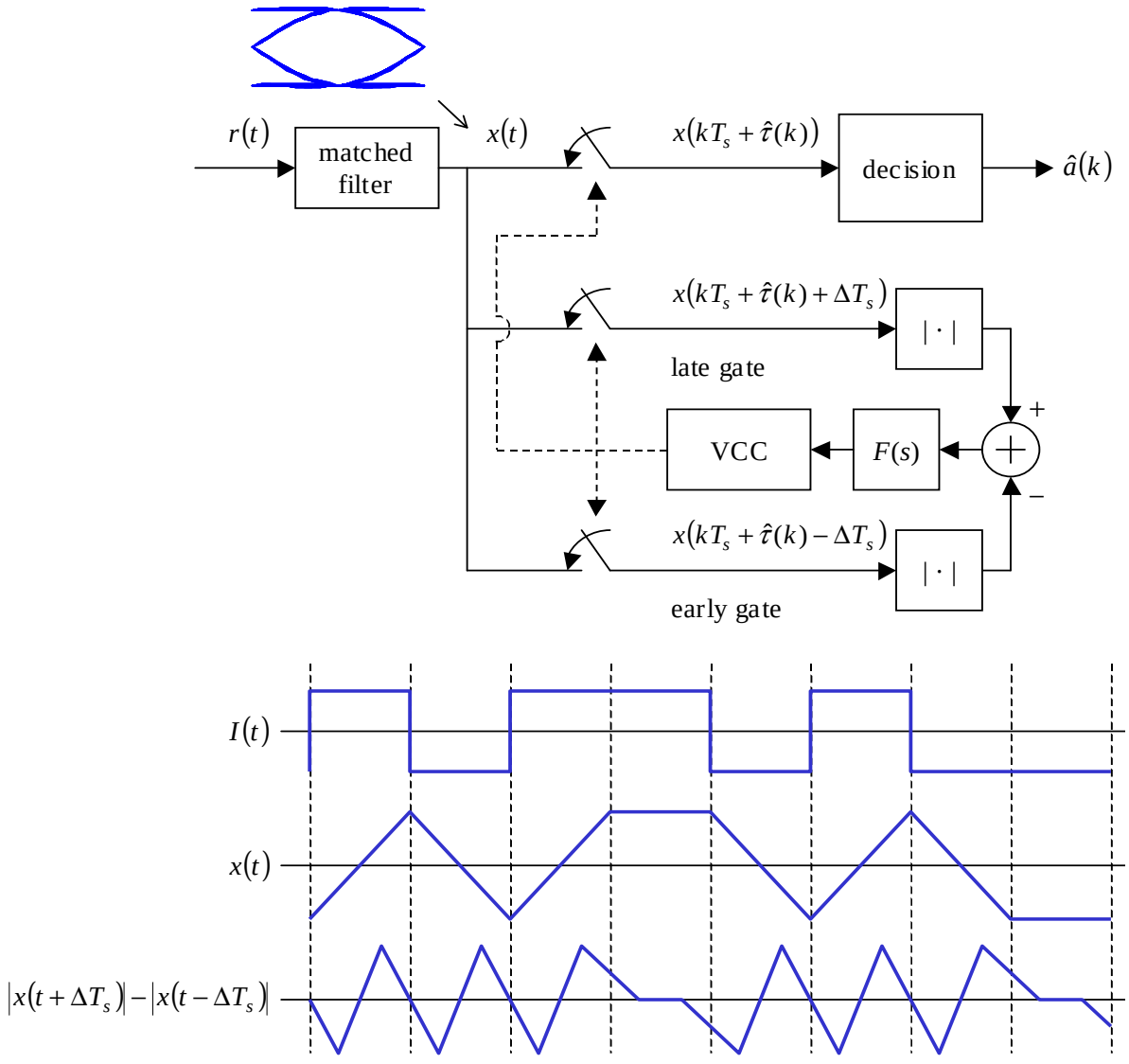


Figure 3.37: Timing PLL for binary PAM using the early-late gate detector (3.90).

3.4.2 Continuous-Time Techniques for MQASK

The general form for the non-offset MQASK symbol timing PLL is shown in 3.38 for the case of QPSK. Assuming perfect carrier phase synchronization and neglecting the double frequency terms and noise, the inphase and quadrature components are

$$I(t) = \sum_m a_1(m)p\left(t - mT_s - \tau\right) \tag{3.91}$$

$$Q(t) = \sum_m a_2(m)p\left(t - mT_s - \tau\right) \tag{3.92}$$

where τ is the unknown delay to be estimated by the symbol timing synchronizer. The matched filter outputs are

$$x(t) = \sum_m a_1(m)R_p\left(t - mT_s - \tau\right) \tag{3.93}$$

$$y(t) = \sum_m a_2(m)R_p\left(t - mT_s - \tau\right) \tag{3.94}$$

where $R_p(u)$ is the autocorrelation function of the pulse shape defined by (3.82). Both of these equations are of the same form as (3.81), the sampled matched filter output for M-ary PAM. Thus, timing error information can be derived from both the inphase and quadrature components in parallel. Extending the notions developed in Section 3.4.1, the data-aided timing error signal is

$$e(k) = a_1(k)\dot{x}\left(kT_s\right) + a_2(k)\dot{y}\left(kT_s\right) \tag{3.95}$$

and the decision directed timing error signal is

$$e(k) = \hat{a}_1(k)\dot{x}\left(kT_s\right) + \hat{a}_2(k)\dot{y}\left(kT_s\right) \tag{3.96}$$

where $\dot{x}(t)$ and $\dot{y}(t)$ are the time derivatives of the inphase and quadrature matched filter outputs, respectively. A popular form for the QPSK decision directed error signal results from a straight forward extension of (3.86):

$$e(k) = \text{sgn}\left\{a_1(k)\right\}\dot{x}\left(kT_s\right) + \text{sgn}\left\{a_2(k)\right\}\dot{y}\left(kT_s\right). \tag{3.97}$$

The derivative operation can be replaced by an early-late gate structure on both the inphase and quadrature components

$$\begin{aligned} e(k) = &\left|x\left(kT_s + \hat{\tau}(k) + \Delta T_s\right)\right| - \left|x\left(kT_s + \hat{\tau}(k) + \Delta T_s\right)\right| \\ &+ \left|y\left(kT_s + \hat{\tau}(k) + \Delta T_s\right)\right| - \left|y\left(kT_s + \hat{\tau}(k) + \Delta T_s\right)\right|. \end{aligned} \tag{3.98}$$

As an example of the extension of the binary PAM results to QPSK, a QPSK symbol timing PLL based on (3.97) is shown in Figure 3.39.

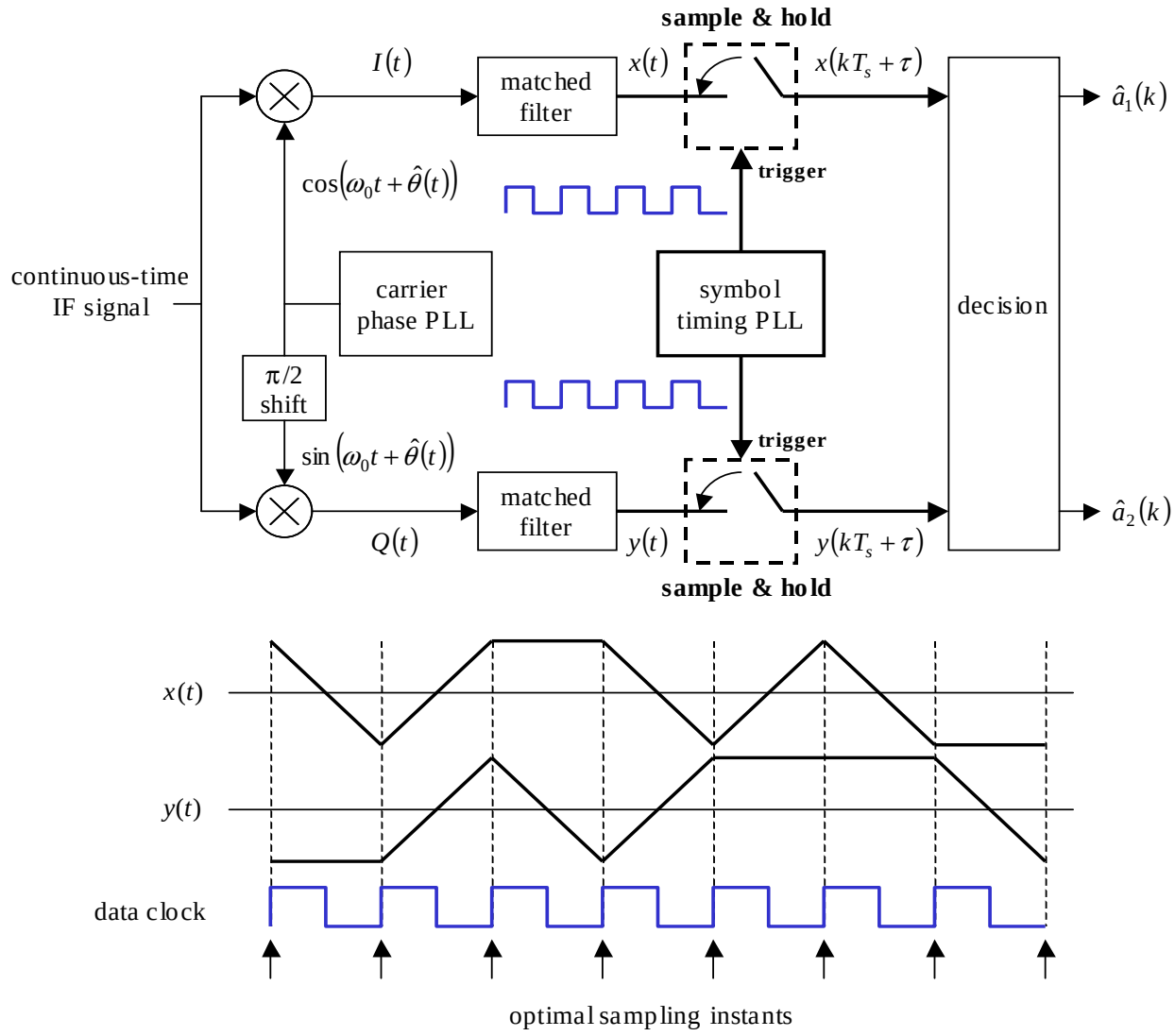


Figure 3.38: Block diagram for a QPSK detector showing the role of the symbol timing PLL and the relationship between the matched filter output and the symbol timing PLL output.

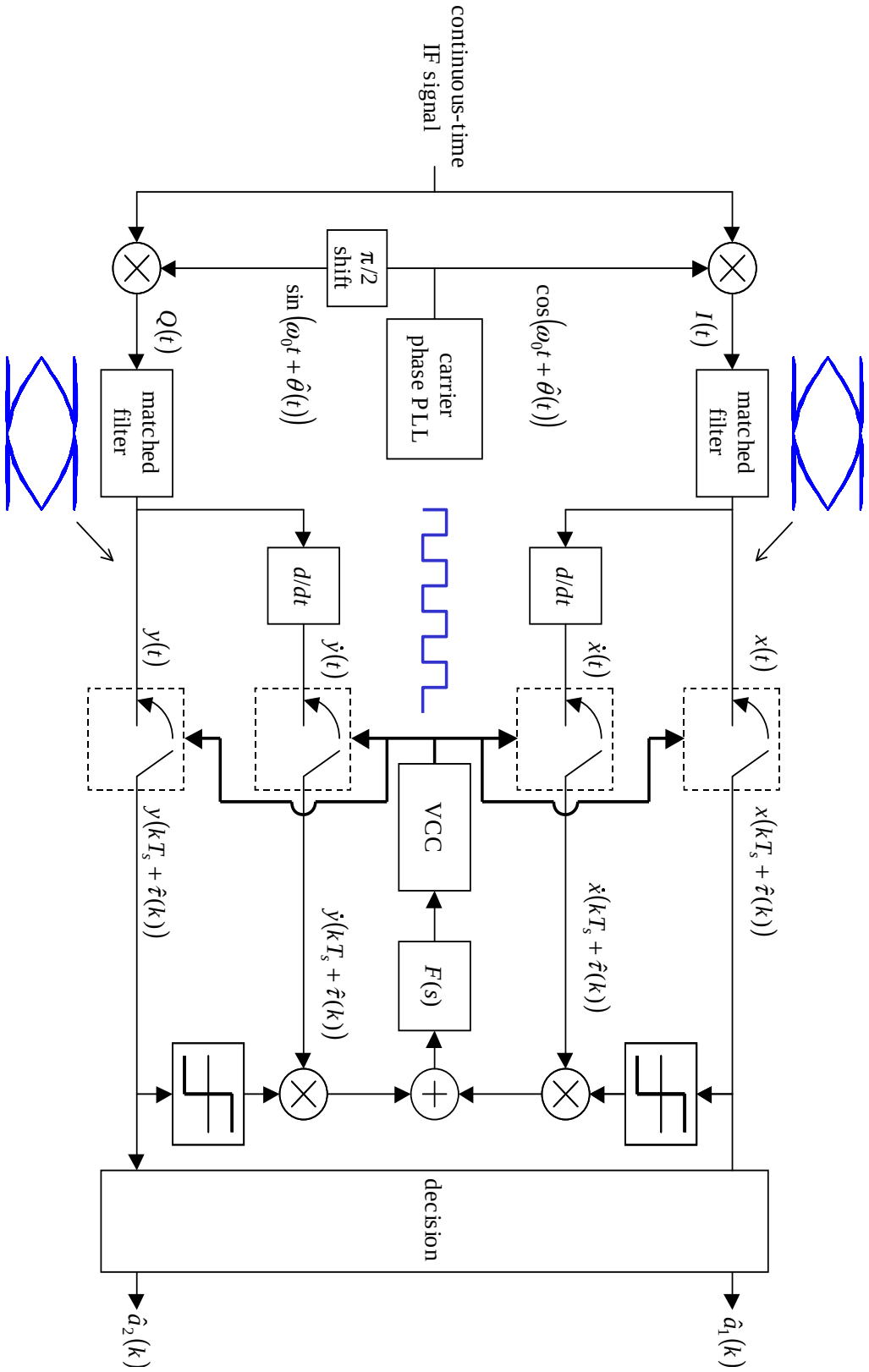


Figure 3.39: QPSK symbol timing PLL showing the detail of the phase detector based on (3.97).

3.4.3 Discrete-Time Techniques for M-ary PAM

When the matched filter is implemented as a discrete-time filter, an analog-to-digital converter (ADC) preceding the matched filter is required. An analog-to-digital converter (ADC) produces T-spaced samples of (3.80) at a rate N samples/symbol. The n-th sample of this waveform may be represented by

$$r(nT) = \sum_m a(m)p(nT - mT_s - \tau) + w(nT) \tag{3.99}$$

where $a(k) \in \{-(M-1), -(M-3), \ldots, -1, 1, \ldots, M-3, M-1\}$ is the k-th symbol; T_s is the symbol time; τ is the unknown timing delay; $p(nT)$ are samples of $p(t)$, the band-limited unit energy pulse shape with support on the interval $-L_pT_s \leq t \leq L_pT_s$; and $w(nT)$ are samples of the bandlimited thermal noise. It is assumed the data symbols are uncorrelated:

$$\mathrm{E}\left\{a(k)a(m)\right\} = \sigma_a^2\delta(m-k) \tag{3.100}$$

where

$$\sigma_a^2 = \mathrm{E}\left\{a^2(k)\right\}. \tag{3.101}$$

The received signal is processed by a discrete-time matched filter whose impulse response consists of samples of the time reversed pulse shape waveform. The matched filter output is

$$x(nT) = \sum_m a(m)R_p\left(nT - mT_s - \tau\right) + v(nT) \tag{3.102}$$

where $R_p(u)$ is the autocorrelation function of the pulse shape given by (3.82) and $v(nT) = p(-nT) * w(nT)$ is the component of the matched filter output due to the noise.

The goal of symbol-timing synchronization is to produce N samples at the matched filter outputs during each symbol interval such that one of the samples is aligned with the maximum eye opening. There are two basic approaches to the problem. The first approach, illustrated in Figure 3.40, uses timing error to adjust the phase of the voltage controlled clock (VCC) that triggers the ADC. As a result, the samples of $r(t)$ are aligned with the symbol boundaries and the optimum eye opening as shown. This approach has the advantage that it produces samples that are aligned in both phase and frequency with the data clock (i.e., T and T_s are commensurate). There are four disadvantages to this approach.

1. First, a feedback path to the continuous-time part of the system is required. The hardware overhead of transferring from the digital to analog domains via a multi-bit output bus, a data control line, a multi-bit DAC, and an analog filter to supply the control voltage to the VCC has the potential to complicate the analog front-end design.

2. Second, the transport delay of the matched filter now resides in the feedback path of the timing control loop. This significantly reduces the response time of the timing recovery loop.

3. Third, higher levels of phase noise (and hence, timing jitter) are contributed by the VCC relative to the phase noise contributed by fixed-frequency sampling clocks.

4. Fourth, this technique does not allow the ADC to be placed at the IF if the IF signal contains multiplexed signals whose symbol clocks are derived from independent sources. In software defined radios, the goal is to "push the ADC to the antenna." To meet this goal, demultiplexing and channel selection must be performed using digital signal processing on asynchronous samples of $r(t)$.

The second approach, illustrated in Figure 3.41 addresses these issues by sampling the received signal $r(t)$ at a fixed rate $1/T$ that is asynchronous with the symbol rate $1/T_s$. The time delay τ is estimated solely from the samples $x(nT)$, the asynchronous samples at the output of the matched filter. This approach produces samples that are not aligned with the symbol boundaries as shown by the eye diagram at the output of the matched filter. The role of symbol timing synchronization is to "move" the samples to the desired time instants. Another name for "moving" samples in time is *interpolation*. Since the timing synchronizer has to adapt to an unknown time delay, the interpolator must be adaptive. When working properly, the interpolator produces matched filter outputs that are aligned with the symbol boundaries and the optimum sampling instant as illustrated by the eye diagram at the output of the interpolator in Figure 3.41.

The major disadvantage to this approach is *interpolation jitter* which occurs when $T_i \neq NT$. In this case, an interpolant is output every N samples, on average. But, due to the condition $T_i \neq NT$, the fractional timing error accumulates and eventually becomes unity. When this occurs, an interpolant is output $N-1$ samples or $N+1$ after the previous interpolant samples to make up the difference. (Which it is depends on the sign of the accumulating fractional timing error.) This interpolation jitter is especially problematic if the data bits must be retransmitted over a synchronous link to some other destination. A more detailed discussion of interpolation jitter and ways to overcome it are discussed in [1, 2].

The asynchronous sampling approach is the more common approach used for timing synchronization in sampled-data detectors. The three basic components of the PLL, the phase detector, loop filter, and DDS are present in the timing loop. The interpolator and timing error detector (TED) combination plays the role of the phase detector while the interpolator control plays the role of oscillator. Timing error detectors are described in Section 3.4.3, interpolation in Section 3.4.3, and interpolation control in Section 3.4.3.

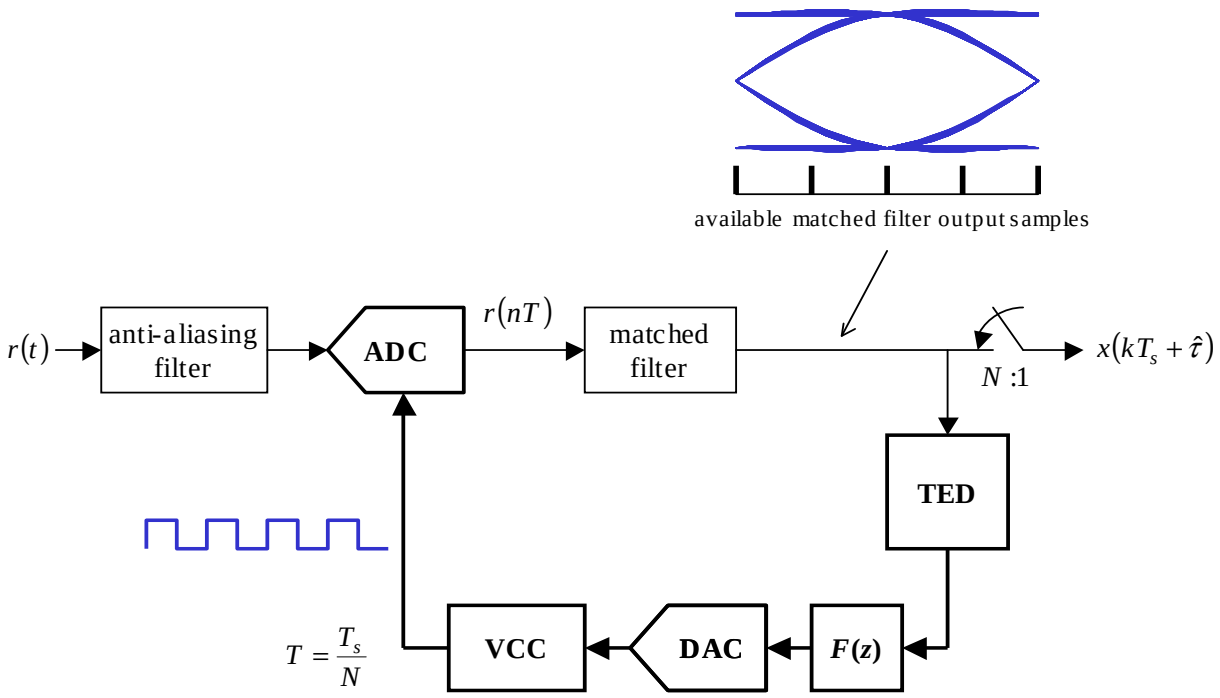


Figure 3.40: A hybrid continuous-time/discrete-time approach to symbol timing synchronization for sampled data detectors.

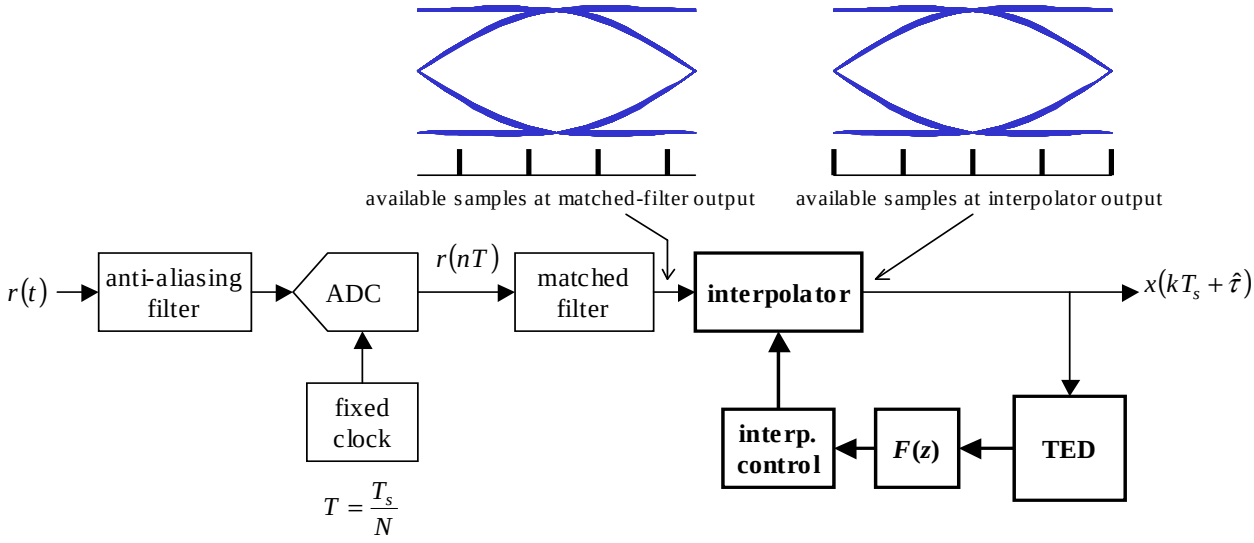


Figure 3.41: A discrete-time approach to symbol timing synchronization for sampled data detectors.

Timing Error Detectors

In general, the timing error detectors produce an error signal once every symbol based on the current timing estimate and using matched filter input, $r(nT)$, and the matched filter output $x(nT)$. In other words, the discrete-time error signal is updated at the symbol rate.

Assume an ideal interpolator is available that computes the interpolant $x(kT_s + \hat{\tau})$ using a timing delay estimate $\hat{\tau}$ and the outputs of the matched filter. The interpolant may be expressed as

$$x(kT_s + \hat{\tau}) = \sum_m a(m) R_p\left((k-m)T_s + \hat{\tau} - \tau\right) + v(kT_s + \hat{\tau}) \tag{3.103}$$

$$= \sum_m a(m) R_p\left((k-m)T_s - \tau_e\right) + v(kT_s + \hat{\tau}) \tag{3.104}$$

where $\tau_e = \tau - \hat{\tau}$ is the timing error. The timing error detector produces a signal that is a function of the timing error τ_e in the same way the phase detector in the carrier phase PLL produced a signal that was a function of the phase error. The output of the timing error detector, $e(kT_s)$, is a function of the interpolated matched filter outputs and the data symbols (or their estimates). The characteristics of the timing error detector are described by the S-curve for the timing error detector denoted $g(\tau_e)$.

Maximum Likelihood Timing Error Detector (MLTED) The maximum likelihood timing error detector is derived in Section 3.6 and uses the sign-corrected slope of the eye digram for the error signal as described in Section 3.4.1 for continuous-time timing error detectors and illustrated in Figure 3.34. The error signal for the data-aided timing error detector is

$$e(k) = a(k)\dot{x}(kT_s + \hat{\tau}) \tag{3.105}$$

while the error signal for the decision-directed timing error detector is

$$e(k) = \hat{a}(k)\dot{x}(kT_s + \hat{\tau}) \tag{3.106}$$

where $\dot{x}(kT_s + \hat{\tau})$ is the time derivative of the matched filter output at $t = kT_s + \hat{\tau}$.

The S-curve for the MLTED is obtained by computing the expected value of the error signal using (3.104) for $x(kT_s + \hat{\tau})$ and the property (3.100). The S-curve for the data-aided MLTED is

$$\begin{aligned} g(\tau_e) &= \mathrm{E}\left\{ a(k)\frac{d}{dt}x(kT_s + \tau) \right\} \\ &= \mathrm{E}\left\{ a(k)\frac{d}{dt}\sum_m a(m) R_p\left((k-m)T_s - \tau_e\right) \right\} \\ &= \sigma_a^2 \dot{R}_p\left(-\tau_e\right) \end{aligned} \tag{3.107}$$

where the last line follows from (3.100) and (3.101) and $\dot{R}_p(-\tau_e)$ is the time derivative of the pulse shape autocorrelation function evaluated at $-\tau_e$. The S-curve for the decision-directed MLTED is obtained by assuming $\hat{a}(k) = a(k)$ and proceeding as outlined above. As long as the decisions are correct, the S-curve for the decision-directed MLTED is identical to the S-curve for the data-aided MLTED. This is illustrated in Figure 3.42 which is a plot of the S-curve for the square-root raised-cosine pulse shape with 50% excess bandwidth. The S-curves for both the data-aided detector and the decision-directed detector are identical for $|\tau_e| < 0.35$. This is because $\hat{a}(k) = a(k)$. When $|\tau_e| > 0.35$ the S-curve for the decision-directed detector departs from the S-curve for the data-aided detector due to decision errors.

The detector gain K_p is the slope of $g(\tau_e) = \dot{R}_p(-\tau_e)$ at $\tau_e = 0$. K_p is a function of the excess bandwidth when $p(t)$ is the root-raised cosine pulse shape. This dependence is plotted in Figure 3.43.

Samples of the derivative of the matched filter output may be obtained from samples of the matched filter output using a filter as outlined in Chapter **??**. Denoting the impulse response of the "derivative filter" as $d(nT)$, samples of the derivative of $x(nT)$ may be expressed in one of two forms as

$$\begin{aligned} \dot{x}(nT) &= x(nT) * d(nT) \qquad (3.108) \\ &= (r(nT) * p(-nT)) * d(nT) \\ &= r(nT) * (p(-nT) * d(nT)) \\ &= r(nT) * \dot{p}(-nT) \qquad (3.109) \end{aligned}$$

where the third line follows from the second line by the associative property of convolution. The two expressions (3.108) and (3.109) suggest two alternate discrete-time systems for producing the desired samples. The system in Figure 3.44 (a) illustrates the discrete-time processing defined by (3.108). The system in Figure 3.44 (b) illustrated the discrete-time processing defined by (3.109) which uses a filter whose impulse response consists of samples of the time derivative of $p(-t)$.

A complete detector requires both matched filter outputs and derivative matched filter outputs. Thus, the use of either approach to compute the derivative matched filter requires two filters. However, the approach illustrated in Figure 3.44 (a) uses the two filters in series whereas the approach illustrated in Figure 3.44 (b) uses two filters operating on the same input samples in parallel. This second approach can be important in a delay-sensitive application such as a phase locked loop.

In either case, the sample rate used to produce the samples of $r(t)$ or $x(t)$ must satisfy the Nyquist sampling theorem. Samples of the time derivative of the $x(t)$ cannot be obtained from undersampled signals that have been distorted by aliasing. Thus the MLTED is, in general, a multi-rate discrete-time system. The input sample rate is N samples/symbol while the output

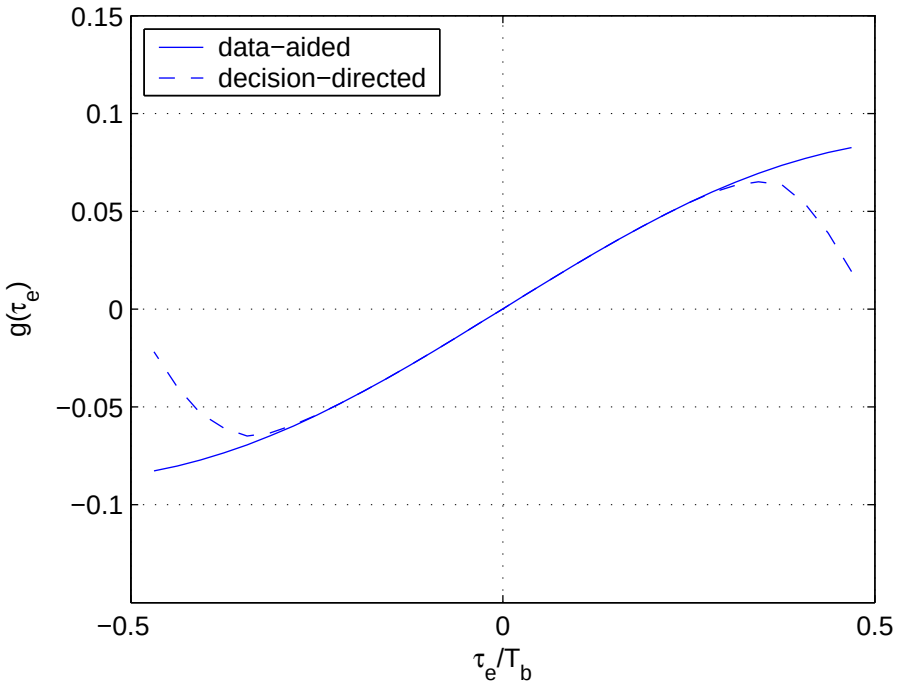


Figure 3.42: S-curves for the data-aided zero crossing detector (solid line) and the decision-directed zero crossing detector (dashed line). These are simulation results for binary PAM using a square-root raised cosine pulse shape with 50% excess bandwidth and $\sigma_a^2 = 1$. The signal-to-noise ratio is $E_b/N_0 = 20$ dB. The derivative matched filter was obtained by computing the first central difference of a unit energy matched filter at $N = 32$ samples/symbol.

sample rate is 1 sample/symbol.

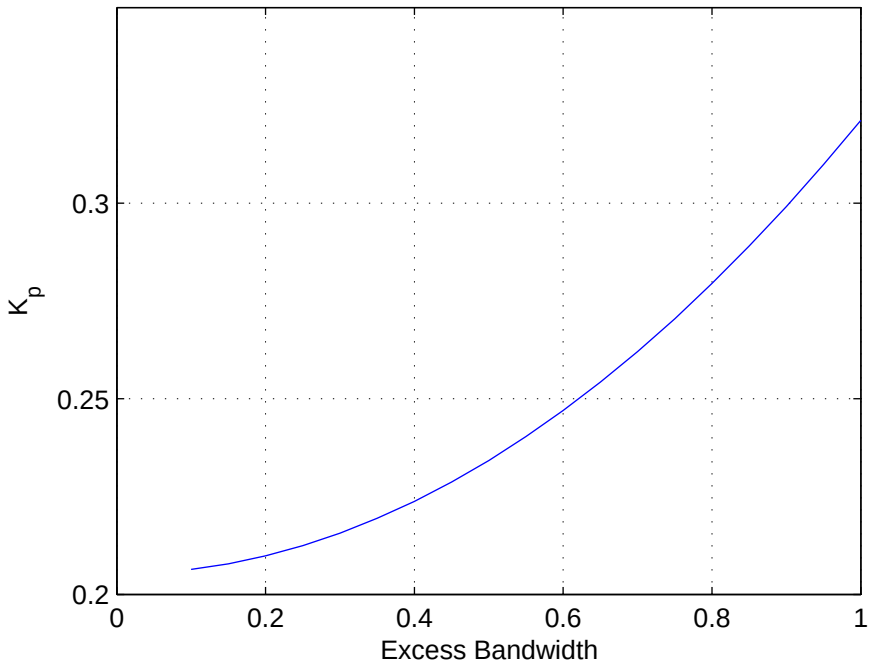


Figure 3.43: Phase detector gain, K_p, of the maximum likelihood timing error detector as a function of excess bandwidth for the square-root raised cosine pulse shape and binary PAM with $\sigma_a^2 = 1$. The derivative of $R_p(\cdot)$ was obtained by computing the first central difference of a unit amplitude raised cosine response sampled at $N = 32$ samples/symbol.

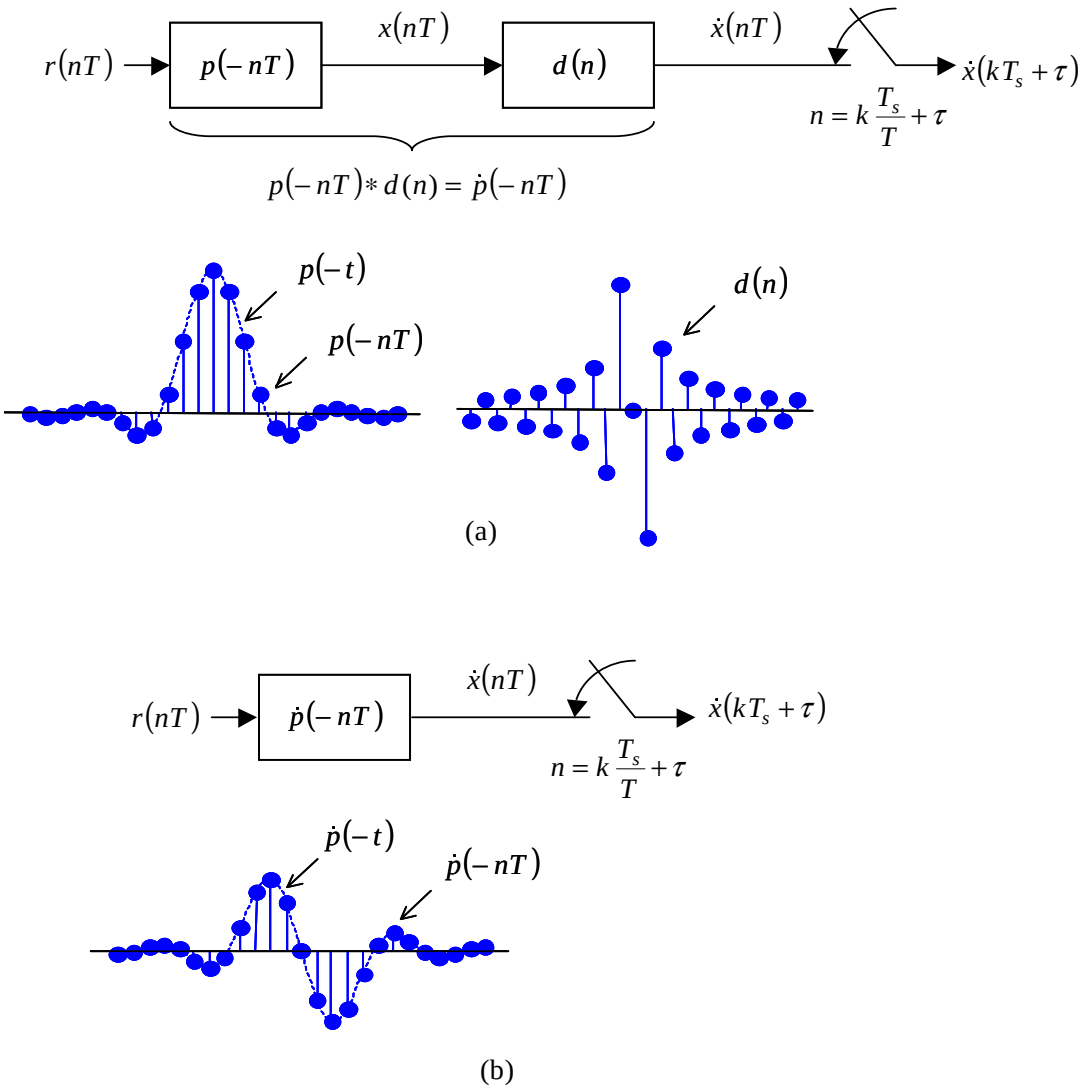


Figure 3.44: Two approaches for computing samples of the derivative of the matched filter output.

Early-Late Timing Error Detector (ELTED) The early-late timing error detector (ELTED) uses time differences, as described in Section 3.4.1, to approximate the derivative required by the MLTED. The data-aided early-late error signal is of the form

$$e(k) = a(k)\left[x\left(kT_s + \hat{\tau} + \Delta T_s\right) - x\left(kT_s + \hat{\tau} - \Delta T_s\right)\right]$$

where ΔT_s is usually selected to be a value conveniently supplied by the sample rate. Since a sample rate of 2 samples/symbol is commonly used, $\Delta = 1/2$ is a popular choice. When sampled at two samples/symbol, the matched filter outputs may be indexed using the symbol index k as

$$\ldots, x((k-1)T_s - \tau), x((k-1/2)T_s - \tau), x(kT_s - \tau), x((k+1/2)T_s - \tau), x((k+1)T_s - \tau), \ldots$$

The early-late timing error for a sample rate of two samples/symbol is

$$e(k) = a(k)\left[x\left((k+1/2)T_s + \hat{\tau}\right) - x\left((k-1/2)T_s + \hat{\tau}\right)\right] \tag{3.110}$$

for the data-aided detector and

$$e(k) = \hat{a}(k)\left[x\left((k+1/2)T_s + \hat{\tau}\right) - x\left((k-1/2)T_s + \hat{\tau}\right)\right] \tag{3.111}$$

for the decision-directed detector.

The S-curve for the ELTED is obtained by computing the expected value of the error signal using (3.104) for $x((k+1/2)T_s + \hat{\tau})$ and $x((k-1/2)T_s + \hat{\tau})$ along with the property (3.100). The S-curve for the data-aided ELTED is

$$\begin{aligned} g(\tau_e) &= \mathrm{E}\left\{a(k)\left[x((k+1/2)T_s + \tau) - x((k-1/2)T_s + \tau)\right]\right\} \\ &= \mathrm{E}\left\{a(k)\sum_m a(m)R_p\left((k-m+1/2)T_s - \tau_e\right) - R_p\left((k-m-1/2)T_s - \tau_e\right)\right\} \\ &= \sigma_a^2\left[R_p\left(T_s/2 - \tau_e\right) - R_p\left(-T_s/2 - \tau_e\right)\right] \end{aligned} \tag{3.112}$$

where the last line follows from (3.100) and (3.101). The S-curve is thus an approximation to the derivative of $R_p(t)$ at $t = -\tau_e$ using values of $R_p(t)$ half a symbol time before and after $-\tau_e$. (Compare the S-curve for the ELTED with the S-curve for the MLTED.) Since $R_p(-\tau_e)$ is an autocorrelation function, it is symmetric and therefor zero at $\tau_e = 0$. The S-curve for the decision-directed MLTED is obtained by assuming $\hat{a}(k) = a(k)$ and proceeding as outlined above. As long as the decisions are correct, the S-curve for the decision-directed MLTED is identical to the S-curve for the data-aided MLTED as illustrated in Figure 3.45 for the square-root raised cosine pulse shape with 50% excess bandwidth. Again, the S-curves for both the data-aided detector

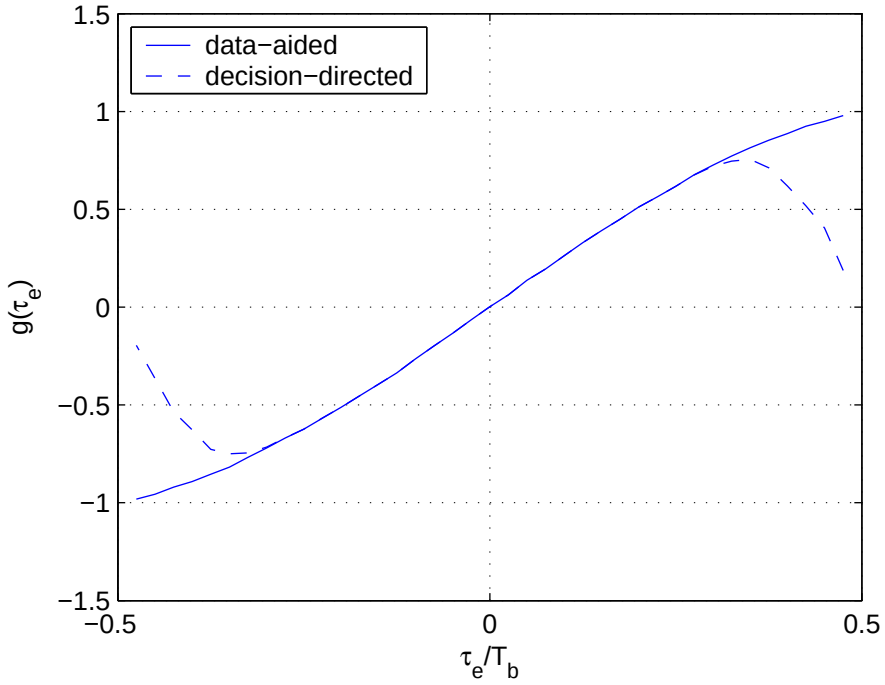


Figure 3.45: S-curves for the data-aided early-late timing error detector (solid line) and the decision-directed early-late timing error detector (dashed line). These are simulation results for binary PAM using a square-root raised cosine pulse shape with 50% excess bandwidth and $\sigma_a^2 = 1$. The signal-to-noise ratio is $E_b/N_0 = 20$ dB.

and the decision-directed detector are identical for $|\tau_e| < 0.35$. This is because $\hat{a}(k) = a(k)$. When $|\tau_e| > 0.35$ the S-curve for the decision-directed detector departs from the S-curve for the data-aided detector due to decision errors.

The ELTED gain, K_p, is the slope of $g(0)$ which is a function of the excess bandwidth when $p(t)$ is the root-raised cosine pulse shape. This dependence is plotted in Figure 3.46.

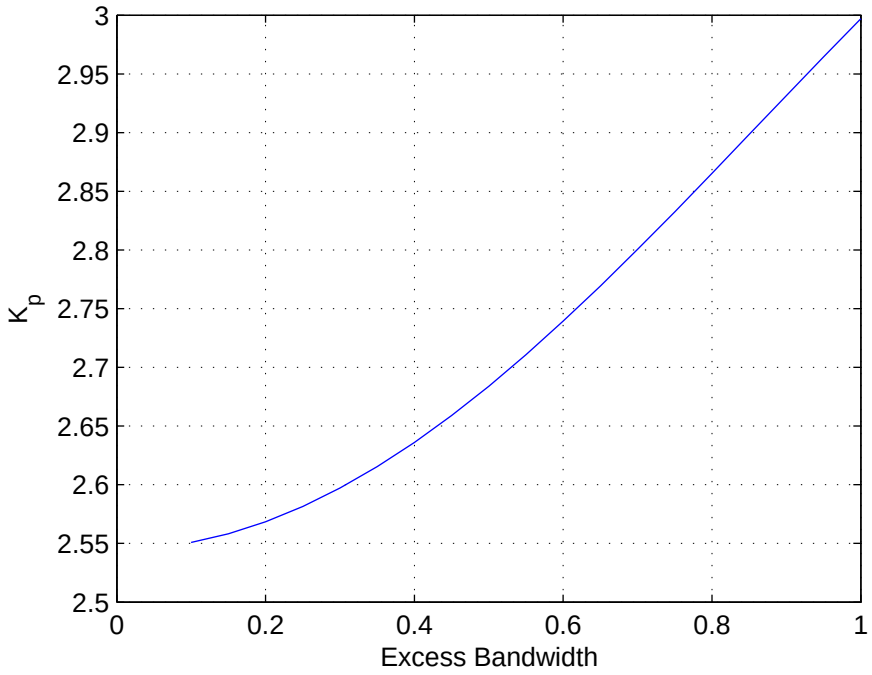


Figure 3.46: Phase detector gain, K_p, of the early-late timing error detector as a function of excess bandwidth for the for the square-root raised cosine pulse shape and binary PAM with $\sigma_a^2 = 1$.

Zero Crossing Detector (ZCTED) The zero crossing detector is intended for use with binary baseband PAM, BPSK or QPSK and operates at two samples/symbol. For convenience, assume that the matched filter outputs are available at a rate of two samples per symbol and may be indexed using the symbol index k as

$$\ldots, x((k-1)T_s - \tau), x((k-1/2)T_s - \tau), x(kT_s - \tau), x((k+1/2)T_s - \tau), x((k+1)T_s - \tau), \ldots$$

The timing error signal, using a timing offset estimate $\hat{\tau}$ to perform the interpolations is

$$e(k) = x((k-1/2)T_s - \tau_e)\,[a(k-1) - a(k)] \tag{3.113}$$

for the case of data-aided symbol timing synchronization and

$$\begin{aligned} e(k) &= x((k-1/2)T_s - \tau_e)\,[\hat{a}(k-1) - \hat{a}(k)] && (3.114)\\ &= x((k-1/2)T_s - \tau_e)\,[\text{sgn}\,\{x((k-1)T_k - \tau_e)\} - \text{sgn}\,\{x(kT_s - \tau_e)\}] && (3.115) \end{aligned}$$

for decision directed symbol timing synchronization.

The error signal is based on finding the zero crossings in the eye diagram as illustrated in Figure 3.47. The sign of the error is controlled by the difference $a(k-1)-a(k)$ or $\hat{a}(k-1)-\hat{a}(k)$. Figure 3.47 (a) illustrates the case where the timing error τ_e is positive and the data transition is positive-to-negative. The timing error is $e(k) = x((k-1/2)T_s - \tau_e)[a(k-1) - a(k)] = 2x((k-1/2)T_s - \tau_e) > 0$ and provides an error signal with the correct sign. In Figure 3.47 (b), the timing error τ_e is negative. The error signal is $e(k) = x((k-1/2)T_s - \tau_e)[a(k-1) - a(k)] = 2x((k-1/2)T_s - \tau_e) < 0$ and provides an error signal with the correct sign. When the data transition is negative-to-positive, the sign of $x((k-1/2)T_s - \tau_e)$ is wrong but is corrected by $a(k-1) - a(k) = -2$. Note that when there is no data transition, $a(k-1) - a(k) = 0$ and no timing error information is provided.

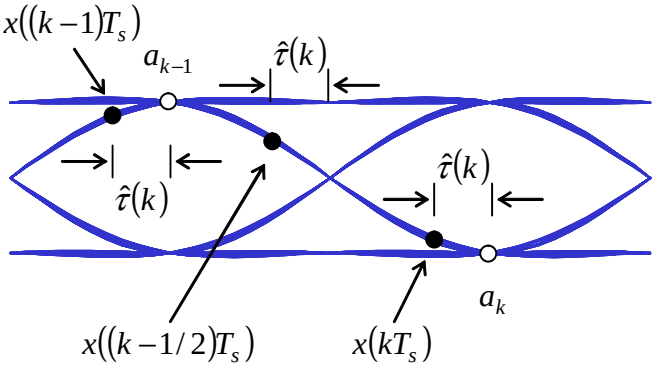
The S-curve for the ZCTED may be obtained by computing the expected value of $e(k)$ and using an expression of the form (3.104) for $x((k-1/2)T_s)$. The S-curve for the data-aided ZCD is

$$\begin{aligned} g(\tau_e) &= \text{E}\,\{x((k-1/2)T_s - \tau_e)[a(k-1) - a(k)]\} && (3.116)\\ &= \text{E}\left\{\sum_m a(m) R_p\left((k-m)T_s - T_s/2 - \tau_e\right)\left[a(k-1) - a(k)\right]\right\} && (3.117)\\ &= \sigma_a^2\left[R_p\left(T_s/2 - \tau_e\right) - R_p\left(-T_s/2 - \tau_e\right)\right] && (3.118) \end{aligned}$$

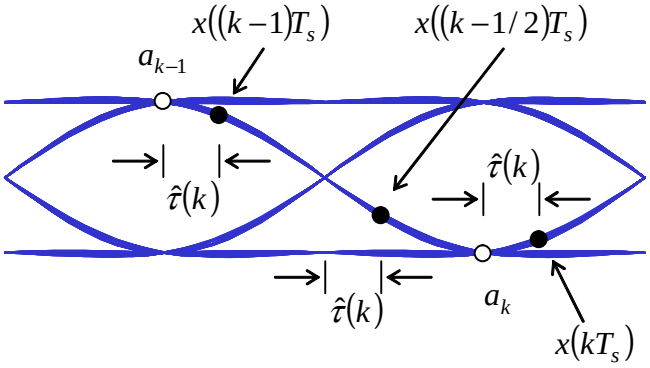
where the last line follows from (3.100) and (3.101). The S-curve is thus an estimate of the slope of $R_p(-\tau_e)$ using values of $R_p(t)$ half a symbol time before and after $-\tau_e$. Since autorcorrelation

functions are symmetric, the S-curve is zero at $\tau_e = 0$. The S-curve for the decision-directed ZCTED is identical when $\hat{a}(k-1) = a(k-1)$ and $\hat{a}(k) = a(k)$ as illustrated in Figure 3.48. Note that the S-curve for the ZCTED given by (3.118) is identical to the S-curve for the ELTED given by (3.112). The ZCTED performance, however, is superior to that of the ELTED since the ELTED suffers from a higher degree of self noise than the the ZCTED.

The TED detector gain, K_p is a function of the pulse shape which, for the square-root raised cosine pulse shape, is a function of the excess bandwidth as shown in Figure 3.49.



(a)



(b)

Figure 3.47: Example showing the operation of the zero crossing detector for positive-to-negative data transitions. (a) The timing estimate is early. (b) The timing estimate is late.

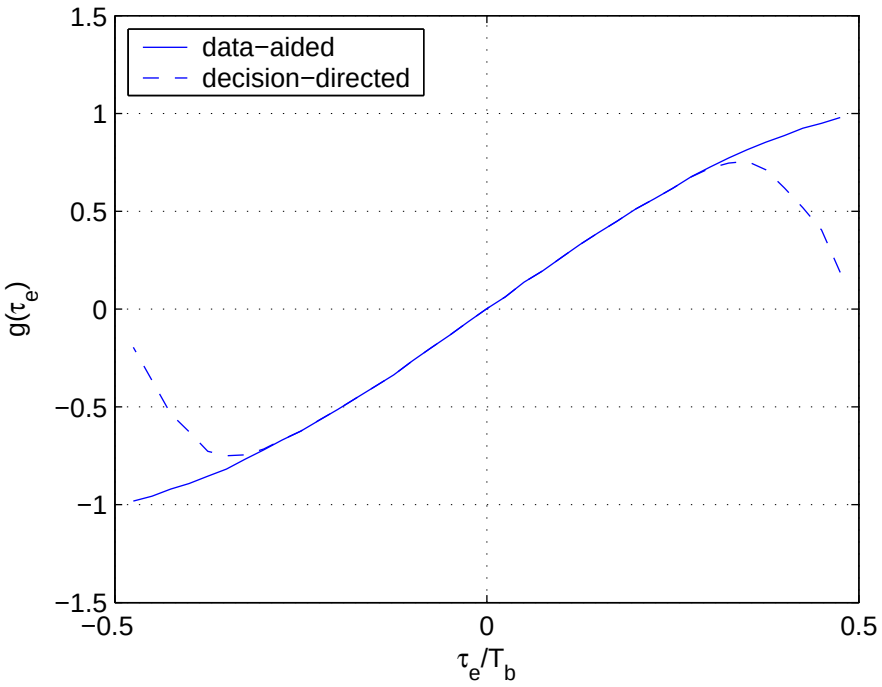


Figure 3.48: S-curves for the data-aided zero crossing detector (solid line) and the decision-directed zero crossing detector (dashed line). These are simulation results for binary PAM using a square-root raised cosine pulse shape with 50% excess bandwidth and $\sigma_a^2 = 1$. The signal-to-noise ratio is $E_b/N_0 = 20$ dB.

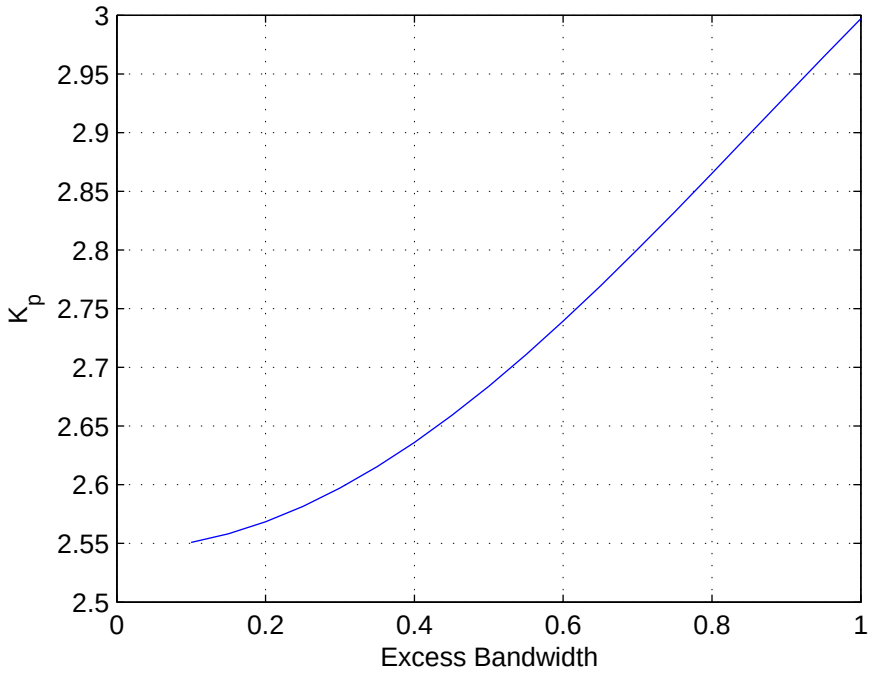


Figure 3.49: Phase detector gain, K_p, of the zero crossing detector as a function of excess bandwidth for the for the square-root raised cosine pulse shape and binary PAM with $\sigma_a^2 = 1$.

Mueller and Müller Detector (MMD) The Mueller and Müller detector (MMD) operates on the matched filter outputs sampled at one sample/symbol. The symbol timing error signal is

$$e(k) = a(k-1)x(kT_s - \tau_e) - a(k)x((k-1)T_s - \tau_e) \tag{3.119}$$

for data-aided symbol timing synchronization and

$$e(k) = \hat{a}(k-1)x(kT_s - \tau_e) - \hat{a}(k)x((k-1)T_s - \tau_e) \tag{3.120}$$

for decision-directed symbol timing synchronization. An interpretation of MMD operation may be obtained through the expression for the S-curve. As before, the S-curve is obtained by computing the expected value of $e(k)$ using (3.104) for $x(kT_s - \tau_e)$ and $x((k-1)T_s - \tau_e)$. The S-curve for the data-aided timing error detector is

$$\begin{aligned} g(\tau_e) &= \mathrm{E}\left\{a(k-1)x(kT_s - \tau_e) - a(k)x((k-1)T_s - \tau_e)\right\} \\ &= \mathrm{E}\left\{a(k-1)\sum_m a(m)R_p\left((k-m)T_s - \tau_e\right) - a(k)\sum_m a(m)R_p\left((k-1-m)T_s - \tau_e\right)\right\} \\ &= \sigma_a^2\left[R_p\left(T_s - \tau_e\right) - R_p\left(-T_s - \tau_e\right)\right]. \end{aligned} \tag{3.121}$$

The S-curve is thus an estimate of the slope of $R_p(\tau_e)$ using values of values a symbol time before and after $-\tau_e$. Since autorcorrelation functions are symmetric, the S-curve is zero at $\tau_e = 0$. The S-curve for the decision-directed MMD is identical when $\hat{a}(k-1) = a(k-1)$ and $\hat{a}(k) = a(k)$ as shown in Figure 3.50 for the square-root raised-cosine pulse shape with 50% excess bandwidth. When $|\tau_e| < 0.35$, the symbol decision are correct and the two S-curve are identical. When $|\tau_e| > 0.35$, some of the symbol decisions are incorrect and reduce the MMD gain as indicated by the departure of the S-curve for the decision-directed MMD from the S-curve for the data-aided MMD. The phase detector gain, K_p is a function of the pulse shape which, for the square-root raised cosine pulse shape, is a function of the excess bandwidth as shown in Figure 3.51.

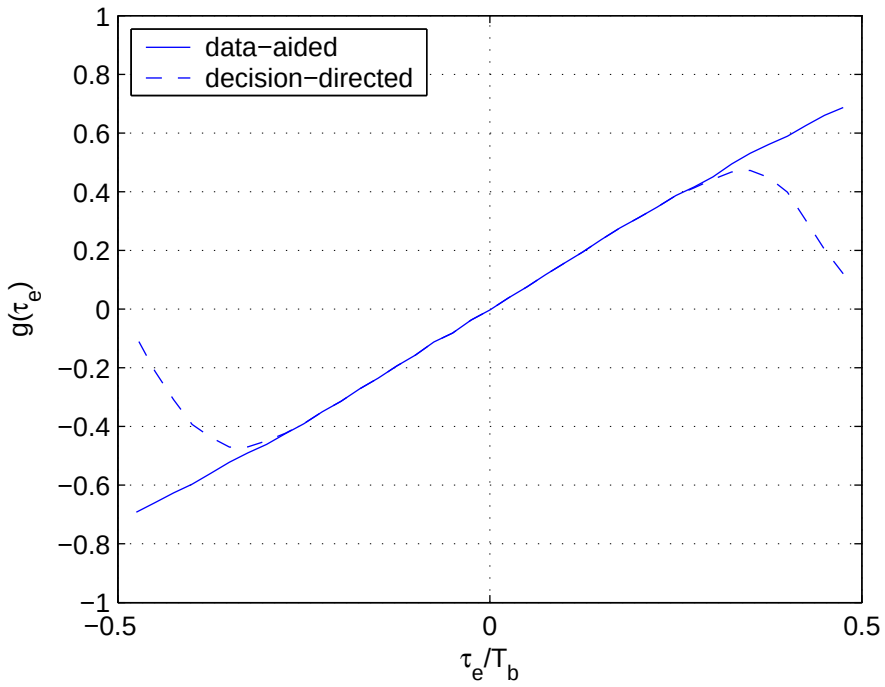


Figure 3.50: S-curves for the data-aided Mueller and Müller detector (solid line) and the decision-directed Mueller and Müller detector (dashed line). These are simulation results for binary PAM using a square-root raised cosine pulse shape with 50% excess bandwidth and $\sigma_a^2 = 1$. The signal-to-noise ratio is $E_b/N_0 = 20$ dB.

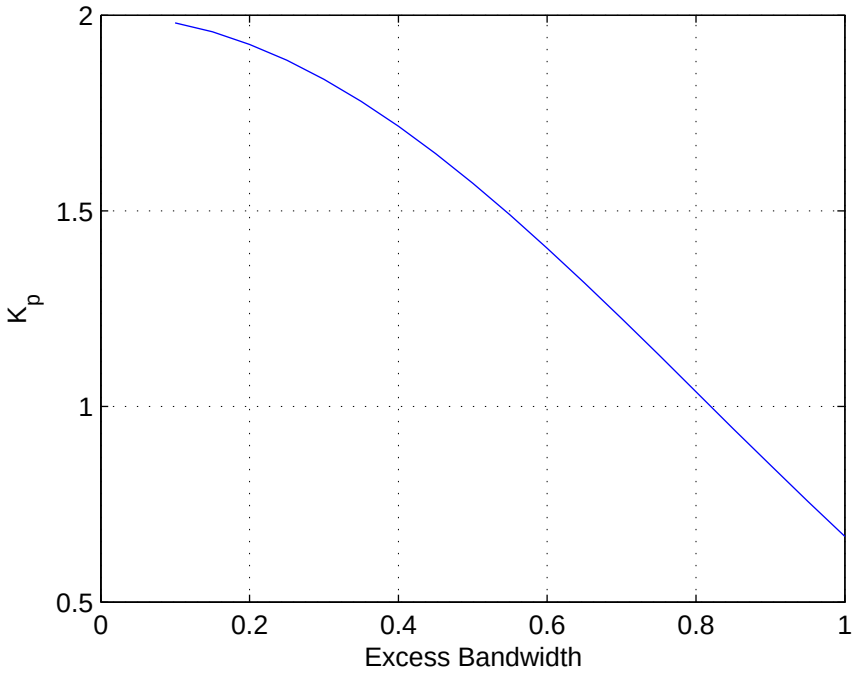


Figure 3.51: Phase detector gain, K_p, of the Mueller and Müller detector as a function of excess bandwidth for the for the square-root raised cosine pulse shape and binary PAM with $\sigma_a^2 = 1$.

Interpolation

The commonly used terms to describe interpolation are illustrated by the diagram in Figure 3.52. T-spaced samples of the bandlimited continuous time signal $x(t)$ are available and denoted

$$\ldots, x((n-1)T), x(nT), x((n+1)T), x((n+2)T), \ldots.$$

The desired sample is a sample of $x(t)$ at $t = kT_i$ and is called the k-th *interpolant*. The process used to compute $x(kT_i)$ from the available samples is called *interpolation*. When the k-th interpolant is between samples $x(nT)$ and $(x(n+1)T)$, the sample index n is called the k-th *basepoint index* and is denoted $m(k)$. The time instant kT_i is some fraction of a sample time greater than $m(k)T$. This fraction is called the k-th fractional interval and is denoted $\mu(k)$. The k-th fractional interval satisfies $0 \leq \mu(k) < 1$ and is defined by $\mu(k)T = kT_i - m(k)T$.

The fundamental equation for interpolation may be derived by considering a fictitious system involving continuous-time processing illustrated in Figure 3.53. The samples $x(nT)$ ($n = 0, 1, \ldots$) are converted to a weighted impulse train

$$x_a(t) = \sum_n x(nT)\delta(t - nT) \tag{3.122}$$

by the digital-to-analog converter (DAC). The impulse train is filtered by an interpolating filter with impulse response $h_I(t)$ to produce the continuous-time output $x(t)$. The continuous-time signal $x(t)$ may be expressed as

$$x(t) = \sum_n x(nT)h_I(t - nT). \tag{3.123}$$

To produce the desired interpolants, $x(t)$ is resampled at intervals kT_i ($k = 0, 1, \ldots$)[6]. The k-th interpolant is (3.123) evaluated at $t = kT_i$ and may be expressed as

$$x(kT_i) = \sum_n x(nT)h_I(kT_i - nT). \tag{3.124}$$

The index n indexes the signal samples. The convolution sum (3.124) may be re-expressed using a filter index i. Using $m(k) = \lfloor kT_i/T \rfloor$ and $\mu(k) = kT_i/T - m(k)$, the filter index is $i = m(k) - n$. Using the filter index, equation (3.124) may be expressed as

$$x(kT_i) = \sum_i x\left(\left(m(k) - i\right)T\right) h_I\left(\left(i + \mu(k)\right)T\right). \tag{3.125}$$

[6]If $T_i = T$ then the process produces one interpolant for each sample. This is the strict definition of *interpolation*. When $T_i \neq T$, then the sample rate of the output is different than the sample rate of the input. This process is known as *resampling* or *rate conversion*. In digital communication applications, $T_i > T$ is the case typically encountered since T is the reciprocal of the sample rate at the input to the matched filter and T_i is the reciprocal of the symbol rate.

Equation (3.125) will serve as the fundamental equation for interpolation and shows that the desired interpolant can be obtained by computing a weighted sum of the available samples. The optimum interpolation filter is an ideal low-pass filter whose impulse response is

$$h_I(t) = \frac{\sin(\pi t/T)}{\pi t/T}. \tag{3.126}$$

Given a fractional interval μ, the ideal impulse response is sampled at $t = iT - \mu T$ to produce the filter coefficients required by (3.125).

The role of the interpolation control block in Figure 3.41 is to provide the interpolator with the basepoint index and fractional interval for each desired interpolant.

For asynchronous sampling, the sample clock is independent of data clock used by the transmitter. As a consequence, the sampling instants are not synchronized to the symbol periods. The sample rate and symbol rate are *incommensurate* and the sample times never coincide exactly with the desired interpolant times. When the symbol timing PLL is in lock and the interpolants are desired once per symbol, $T_i = T_s$. The behavior of the fractional interval $\mu(k)$ as a function of k depends on the relationship between the sample clock period T and the symbol period T_s as follows:

- When T_s is incommensurate with NT, $\mu(k)$ is irrational and changes for each k for infinite precision or progresses through a finite set of values, never repeating exactly for finite precision.
- When $T_s \approx NT$, $\mu(k)$ changes very slowly for infinite precision or remains constant for many k, for finite precision.
- When T_s is commensurate with NT, but not equal: $\mu(k)$ cyclically progresses through a finite set of values.

Since the ideal interpolation filter is IIR, its use poses an often unacceptable computational burden — especially when the fractional interval changes. For this reason, FIR filters that approximate the ideal interpolation filter are preferred in digital communication applications. A popular class of FIR interpolating filters are piece-wise polynomial filters discussed below. Another alternative is to massively upsample the matched filter input, match filter at the high sample rate, then downsample the matched filter output with the appropriately chosen sample offset to obtain the desired interpolant. This approach leads to a polyphase-filterbank interpolator.

Piecewise Polynomial Interpolation The underlying continuous-time waveform $x(t)$ is approximated by a polynomial in t of the form

$$x(t) \approx c_p t^p + c_{p-1} t^{p-1} + \cdots + c_1 t + c_0. \tag{3.127}$$

The polynomial coefficients are determined by the $p+1$ sample values surrounding the basepoint index. Once the coefficient values are known, the interpolant at $t = kT_i = (m(k) + \mu(k))T$ is obtained using

$$x(kT_i) \approx c_p(kT_i)^p + c_{p-1}(kT_i)^{p-1} + \cdots + c_1(kT_i) + c_0. \tag{3.128}$$

Three special cases, $p = 1, 2$, and 3 are of interest and are illustrated in Figure 3.54. When $p = 1$, the first degree polynomial

$$x(t) \approx c_1 t + c_0 \tag{3.129}$$

is used to approximate the underlying continuous-time waveform. The desired interpolants are computed from

$$x((m(k) + \mu(k))T) = c_1((m(k) + \mu(k))T) + c_0. \tag{3.130}$$

The coefficients c_1 and c_0 are determined by the available samples and satisfy the equation

$$\begin{bmatrix} x(m(k)T) \\ x((m(k)+1)T) \end{bmatrix} = \begin{bmatrix} m(k)T & 1 \\ (m(k)+1)T & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_0 \end{bmatrix}. \tag{3.131}$$

Solving the above for c_1 and c_0 and substituting into (3.130) produces

$$x((m(k) + \mu(k))T) = \mu(k)x((m(k)+1)T) + (1 - \mu(k))x(m(k)T) \tag{3.132}$$

which is the familiar linear interpolator.

Four observations are important. The first is that the interpolant is a linear combination of the available samples. As a consequence, the interpolant can be thought of as the output of a filter with coefficients suggested by (3.132):

$$x((m(k) + \mu(k))T) = \sum_{i=-1}^{0} h_1(i)x((m(k) - i)T) \tag{3.133}$$

where

$$\begin{aligned} h_1(-1) &= \mu(k) \\ h_1(0) &= 1 - \mu(k). \end{aligned} \tag{3.134}$$

The second important observation is that the equivalent filter coefficients are a function only of the fractional interval and not a function of the basepoint index. The basepoint index defines *which* set of samples should be used to compute the interpolant. The third observation is that the interpolating filter is linear phase FIR filter which is an extremely important property for digital communications. To see that this filter is linear phase, note that the coefficients are symmetric about the center point of the filter which is defined by $\mu(k) = 1/2$. In other words, $h((m + 1/2)T) = h((-m + 1/2)T)$

for $m = 0, 1, 2, \ldots$. This is a result of using an even number of samples to compute an interpolant that is between the middle two. The final observation is that the sum of the coefficients is unity and is therefor independent of $\mu(k)$. As a consequence, the interpolating filter does not alter the amplitude of the underlying continuous-time waveform in the process of producing the interpolant.

The second observation is an attractive feature since any finite precision computing device would eventually overflow as $m(k)$ increased. The third property requires the use of an even number of samples by the interpolator. Since an even number of samples is needed to define an odd-degree approximating polynomial, odd degree approximating polynomials are popular. The next highest odd-degree polynomial is $p = 3$. In this case

$$x(t) \approx c_3 t^3 + c_2 t^2 + c_1 t + c_0 \tag{3.135}$$

is used to approximate the underlying continuous-time waveform. The desired interpolants are computed from

$$x((m(k) + \mu(k))T) = c_3((m(k) + \mu(k))T)^3 + c_2((m(k) + \mu(k))T)^2 c_1((m(k) + \mu(k))T) + c_0. \tag{3.136}$$

The coefficients c_3, c_2, c_1 and c_0 are defined by

$$\begin{bmatrix} x((m(k)-1)T) \\ x(m(k)T) \\ x((m(k)+1)T) \\ x((m(k)+2)T) \end{bmatrix} = \begin{bmatrix} ((m(k)-1)T)^3 & ((m(k)-1)T)^2 & (m(k)-1)T & 1 \\ (m(k)T)^3 & (m(k)T)^2 & m(k)T & 1 \\ ((m(k)+1)T)^3 & ((m(k)+1)T)^2 & (m(k)+1)T & 1 \\ ((m(k)+2)T)^3 & ((m(k)+2)T)^2 & (m(k)+2)T & 1 \end{bmatrix} \begin{bmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix}. \tag{3.137}$$

Solving the above for c_3, c_2, c_1 and c_0 and substituting into (3.136) produces

$$\begin{aligned} x((m(k) + \mu(k))T) = &\left(\frac{\mu(k)^3}{6} - \frac{\mu(k)}{6}\right) x((m(k)+2)T) \\ &- \left(\frac{\mu(k)^3}{2} - \frac{\mu(k)^2}{2} - \mu(k)\right) x((m(k)+1)T) \\ &+ \left(\frac{\mu(k)^3}{2} - \mu(k)^2 - \frac{\mu(k)}{2} + 1\right) x(m(k)T) \\ &- \left(\frac{\mu(k)^3}{6} - \frac{\mu(k)^2}{2} + \frac{\mu(k)}{3}\right) x((m(k)-1)T) \end{aligned} \tag{3.138}$$

which is called a cubic interpolator. When interpreted as a filter, the cubic interpolator output is of the form

$$x((m(k) + \mu(k))T) = \sum_{i=-2}^{1} h_3(i) x((m(k) - i)T) \tag{3.139}$$

where the filter coefficients are

$$\begin{aligned} h_3(-2) &= \frac{\mu(k)^3}{6} - \frac{\mu(k)}{6} \\ h_3(-1) &= -\frac{\mu(k)^3}{2} + \frac{\mu(k)^2}{2} + \mu(k) \\ h_3(0) &= \frac{\mu(k)^3}{2} - \mu(k)^2 - \frac{\mu(k)}{2} + 1 \\ h_3(1) &= -\frac{\mu(k)^3}{6} + \frac{\mu(k)^2}{2} - \frac{\mu(k)}{3}. \end{aligned} \tag{3.140}$$

Finally, for the case $p = 2$, using the approximation

$$x(t) \approx c_2 t^2 + c_1 t + c_0 \tag{3.141}$$

to approximate the underlying continuous-time waveform and

$$x((m(k) + \mu(k))T) = c_2((m(k) + \mu(k))T)^2 + c_1((m(k) + \mu(k))T) + c_0 \tag{3.142}$$

to compute the desired interpolant requires the use of 3 samples. Since the number of samples is odd, the desired interpolant is not in between the middle two and the resulting filter will not be symmetric with respect to $\mu(k) = 1/2$. The desire to use four points introduces a wrinkle that is explored in the homework where it is shown that the desired interpolant can be thought of as the output of a filter of the form

$$x((m(k) + \mu(k))T) = \sum_{i=-2}^{1} h_2(i) x((m(k) - i)T) \tag{3.143}$$

where the filter coefficients are

$$\begin{aligned} h_2(-2) &= \alpha\mu(k)^2 - \alpha\mu(k) \\ h_2(-1) &= -\alpha\mu(k)^2 + (1 + \alpha)\mu(k) \\ h_2(0) &= -\alpha\mu(k)^2 - (1 - \alpha)\mu(k) + 1 \\ h_2(1) &= \alpha\mu(k)^2 - \alpha\mu(k) \end{aligned} \tag{3.144}$$

and α is a free parameter required to account for the additional degree of freedom introduced by using four points. Simulation results have shown that $\alpha = 0.43$ is the optimal value for BPSK using the root raised cosine pulse shape with 100% excess bandwidth. Using $\alpha = 0.5$ reduces the complexity of the hardware somewhat and results in a performance loss less than 0.1 dB [3].

Using a piece-wise polynomial interpolator to produce the desired interpolant results in a computation of the form

$$x((m(k) + \mu(k))T) = \sum_{i=-I_1}^{I_2} h_p(i; \mu(k)) x((m(k) - i)T) \tag{3.145}$$

where the filter coefficients are given by (3.134), (3.144), (3.140) for $p = 1, 2$, and 3, respectively. Comparing (3.145) with the fundamental interpolation equation (3.125) shows that the filter coefficients $h_p(i; \mu(k))$ play the role of approximating the samples of the ideal interpolation filter $h_I((i - \mu(k))T)$. Plots of $h_1(i; \mu(k))$, $h_2(i; \mu(k))$, and $h_3(i; \mu(k))$ are shown in Figure 3.55. Observe that as p increases, $h_p(i; \mu(k))$ approximates (3.126) with greater and greater accuracy. In fact, in the limit $p \to \infty$, $h_p(i; \mu(k))$ approaches (3.126).

Since the filter coefficients suggested by the filter structure defined by (3.134), (3.140), and (3.144) are a function of the variable $\mu(k)$, a hardware implementation requires two-input multipliers with two variable quantities. The complexity can be reduced by formulating the problem in terms of two-input multipliers where one of the inputs is fixed. Each filter coefficient $h_p(i; \mu(k))$ in (3.145) is a polynomial in $\mu(k)$. Let

$$h_p(i; \mu(k)) = \sum_{l=0}^{p} b_l(i)\mu(k)^l \tag{3.146}$$

represent the polynomial. Substituting (3.146) into (3.145) and rearranging produces

$$x((m(k) + \mu(k))T) = \sum_{l=0}^{p} \mu(k)^l \underbrace{\sum_{i=I_1}^{I_2} b_l(i)x((m(k) - i)T)}_{v(l)} . \tag{3.147}$$

The inner sum looks like a filter equation where the input data samples $x((m(k)-i)T)$ pass through a filter with impulse response $b_l(i)$. Since the $b_l(i)$ are independent of $\mu(k)$, this filter has fixed coefficients and an efficient implementation. Computing (3.147) by nested evaluation produces in an expression of the form

$$x((m(k) + \mu(k))T) = (v(2)\mu(k) + v(1))\,\mu(k) + v(0) \tag{3.148}$$

for piece-wise parabolic interpolation and

$$x((m(k) + \mu(k))T) = ((v(3)\mu(k) + v(2))\,\mu(k) + v(1))\,\mu(k) + v(0) \tag{3.149}$$

for cubic interpolation. Mapping these expressions to hardware results in an efficient filter structure called the *Farrow Structure* illustrated in Figure 3.56. The *Farrow coefficients* for the Farrow structure are listed in Tables 3.1 and 3.2. Note that when $\alpha = 1/2$ for the piece-wise parabolic interpolator, all of the filter coefficients but one become 0, 1, or $\pm 1/2$. The resulting filter structure is elegantly simple.

Table 3.1: Farrow coefficients $b_l(i)$ for the piece-wise parabolic interpolator.

i	$b_2(i)$	$b_1(i)$	$b_0(i)$
-2	α	$-\alpha$	0
-1	$-\alpha$	$1+\alpha$	0
0	$-\alpha$	$\alpha-1$	1
1	α	$-\alpha$	0

Table 3.2: Farrow coefficients $b_l(i)$ for the cubic interpolator.

i	$b_3(i)$	$b_2(i)$	$b_1(i)$	$b_0(i)$
-2	$\frac{1}{6}$	0	$-\frac{1}{6}$	0
-1	$-\frac{1}{2}$	$\frac{1}{2}$	1	0
0	$\frac{1}{2}$	-1	$-\frac{1}{2}$	1
1	$-\frac{1}{6}$	$\frac{1}{2}$	$-\frac{1}{3}$	0

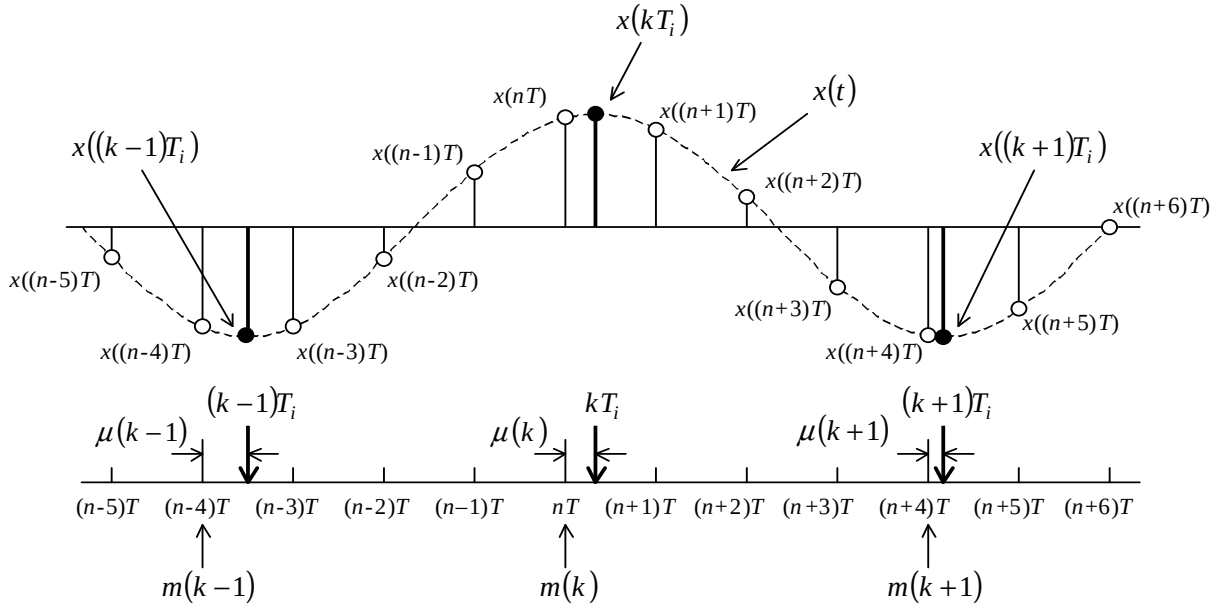


Figure 3.52: Illustration of the relationships between the interpolation interval T_i, the sample time T, the basepoint indexes, and fractional intervals.

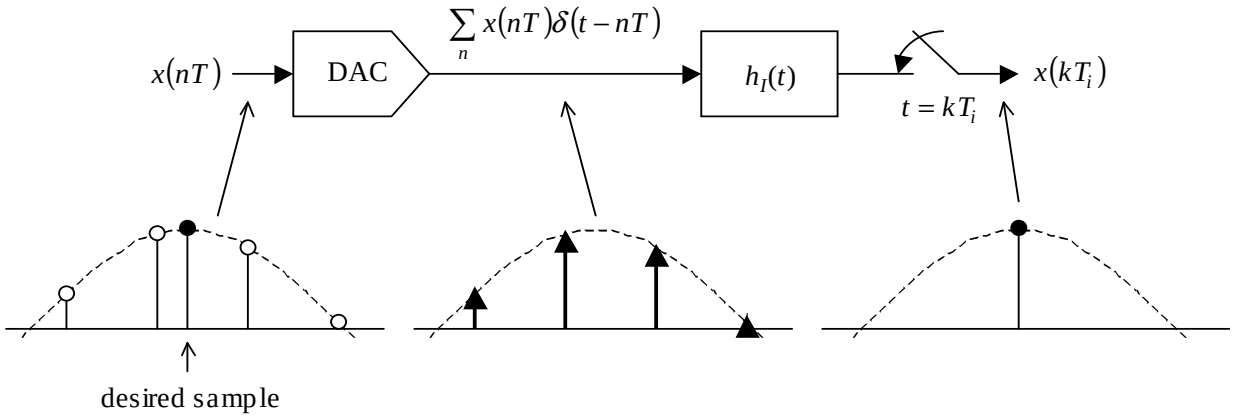
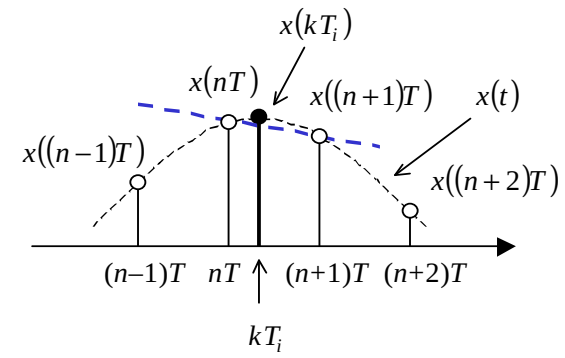
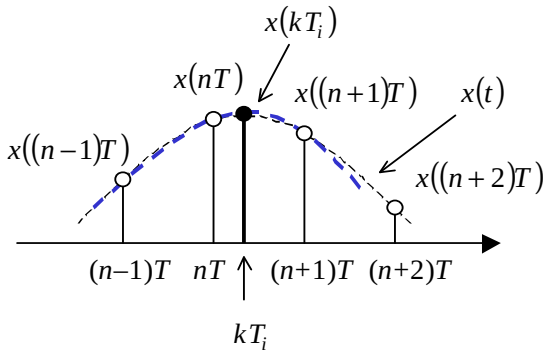


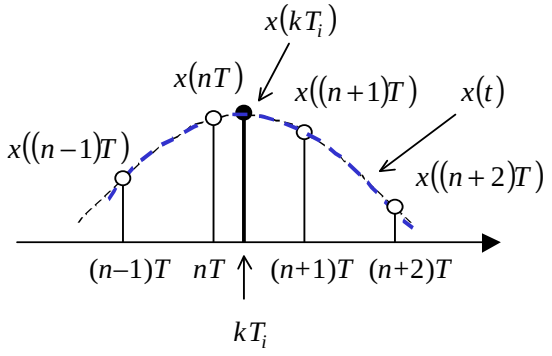
Figure 3.53: Fictitious system using continuous-time processing for performing interpolation.



$$x(t) \approx c_1 t + c_0$$
$$x(kT_i) \approx c_1(kT_i) + c_0$$



$$x(t) \approx c_2 t^2 + c_1 t + c_0$$
$$x(kT_i) \approx c_2(kT_i)^2 + c_1(kT_i) + c_0$$



$$x(t) \approx c_3 t^3 + c_2 t^2 + c_1 t + c_0$$
$$x(kT_i) \approx c_3(kT_i)^3 + c_2(kT_i)^2 + c_1(kT_i) + c_0$$

Figure 3.54: Three special cases of polynomial interpolation: linear interpolation (top), quadratic interpolation (middle), cubic interpolation (bottom).

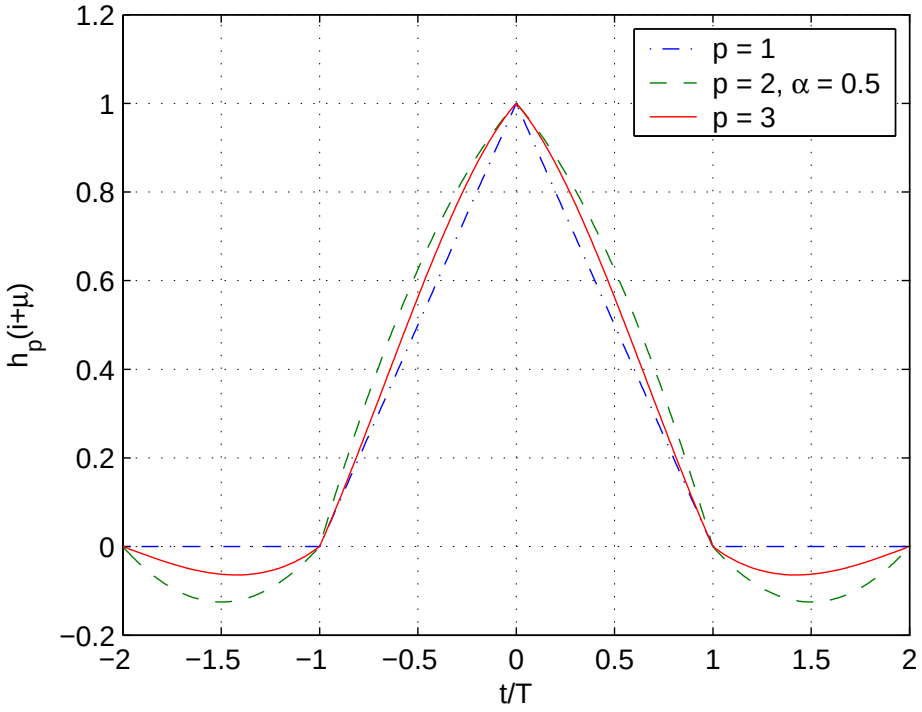


Figure 3.55: Plot of the filter impulse responses resulting from piece-wise polynomial interpolation.

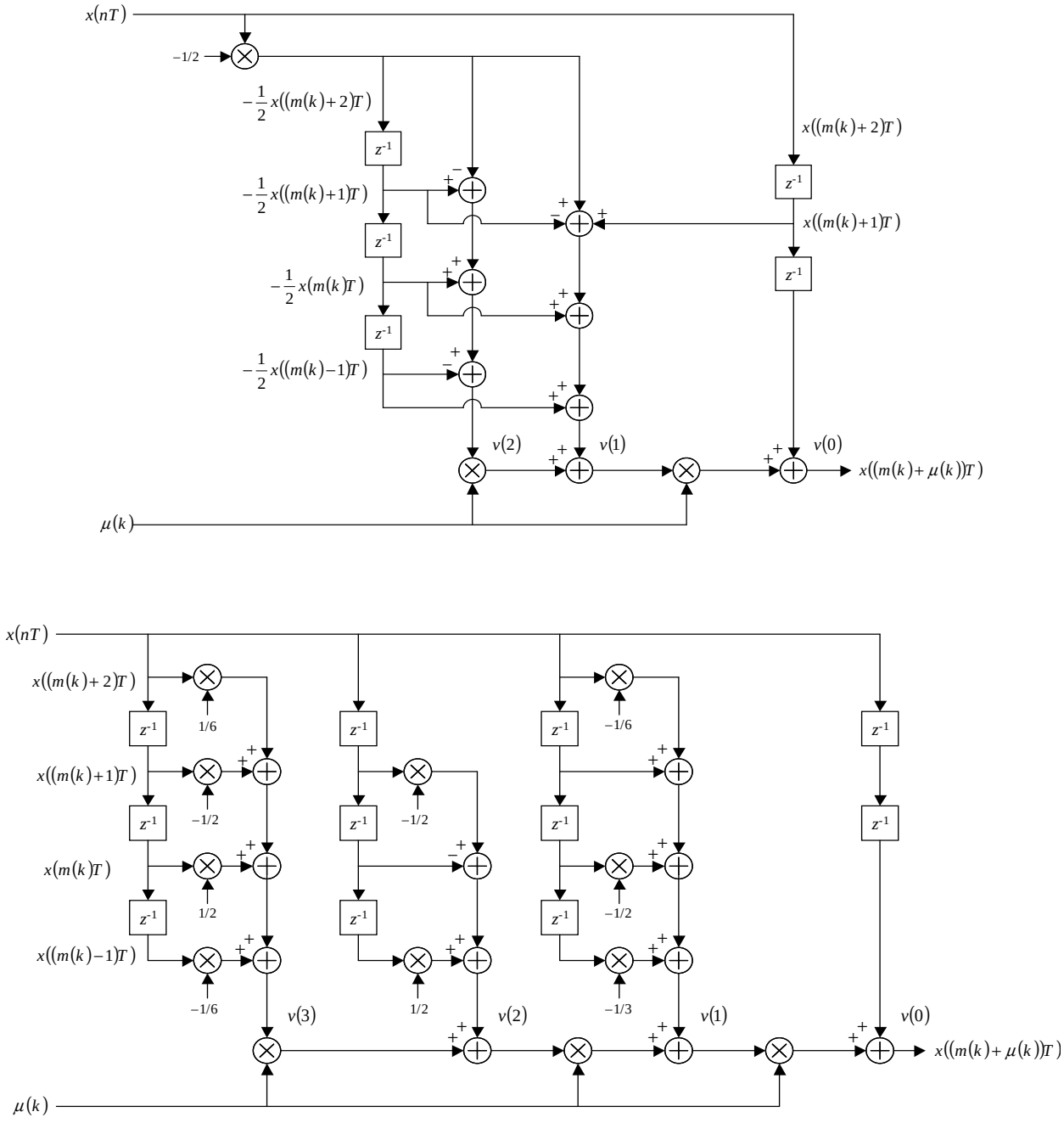


Figure 3.56: Farrow interpolator structures for the piece-wise parabolic with $\alpha = 1/2$ (top) and cubic (bottom) interpolators.

Polyphase Filterbank Interpolation An alternate approach to interpolation is to upsample the matched filter output by a factor Q then down sample with the appropriate offset to produce a sample close to the desired interpolant. How close the sample is to the desired interpolant is controlled by the upsample factor Q. A conceptual block diagram of this process is shown in Figure 3.57 (a) for the case of binary PAM. (Generalizations to M-ary PAM are straight forward.) The input to the matched filter consists of samples the received signal $r(nT)$ sampled at N samples/symbol (i.e., $T_s = NT$). The impulse response $h(nT)$ of the matched filter consists of T-spaced samples of a time-reversed version of the pulse shape $p(t)$: $h(nT) = p(-nT)$. The matched filter output, $x(nT)$ is upsampled by inserting $Q-1$ zeros between each sample. An interpolating low-pass filter is used to produce samples of the matched filter output at a rate of NQ samples/symbol. This signal is denoted $x(nT/Q)$. The matched filter output with the desired delay is obtained by downsampling $x(nT/Q)$ with the proper offset.

The upsample-and-interpolate operation can be applied to the matched filter input instead of the output as illustrated in Figure 3.57 (b). The inphase component of the received signal is upsampled by inserting $Q-1$ zeros between each sample. The upsampled signal is low-pass filtered to produce $r(nT/Q)$ which consists of samples of the inphase component at the high sample rate. In this case, the impulse response of the matched filter consists of T/Q-spaced samples of $p(-t)$. The desired matched filter output is obtained by downsampling the matched filter outputs at the high sample rate, $x(nT/Q)$, with the proper offset.

Since both the interpolating filter and matched filter are low-pass filters, it is not necessary to filter twice. The low pass interpolating filter may be removed as shown in Figure 3.57 (c). The key difference here is that the matched filter is performing two functions: interpolation and shaping. In other words, the matched filter outputs at the high sample rate, $x(nT/Q)$ are not identical to an upsampled version of the input $r(nT/Q)$.

The matched filter outputs at the high sample may be expressed as

$$x\left(n\frac{T}{Q}\right) = \sum_{l=-QNL}^{QNL} r\left((n-l)\frac{T}{Q}\right) h\left(l\frac{T}{Q}\right). \tag{3.150}$$

The sequence $x(nT/Q)$ may be downsampled by Q to produce a sequence at N samples/symbol where every N-th sample is as close to $x(kT_s+\tau)$ as the resolution allows. The polyphase decomposition is due to the fact that not all of the multiplies defined by (3.150) are required. Since

$$r\left(n\frac{T}{Q}\right) = \begin{cases} r(nT) & n = 0, \pm Q, \pm 2Q, \ldots \\ 0 & \text{otherwise,} \end{cases} \tag{3.151}$$

only every Q-th value of $r(nT/M)$ in the FIR matched filter is non-zero. At a time instant at the

high sample rate, these non-zero values coincide with the filter coefficients

$$\ldots, h(-2QT), h(-QT), h(0), h(QT), h(2QT), \ldots$$

and the filter output may be expressed as

$$\sum_{i=-NL}^{NL} r((n-i)T)h(iT) = x(nT). \tag{3.152}$$

At the next time instant the non-zero values of $r(nT/Q)$ coincide with the filter coefficients

$$\ldots, h(-2QT+1), h(-QT+1), h(1), h(QT+1), h(2QT+1), \ldots$$

so that the filter output may be expressed as

$$\sum_{i=-NL}^{NL} r((n-i)T)h\left(\left(i+\frac{1}{Q}\right)T\right) = x\left(\left(n-\frac{1}{Q}\right)T\right). \tag{3.153}$$

At the q-th time instant, the non-zero values of $r(nT/Q)$ coincide with the filter coefficients

$$\ldots, h(-2QT+q), h(-QT+q), h(q), h(QT+q), h(2QT+q), \ldots$$

so that the filter output may be expressed as

$$\sum_{i=-NL}^{NL} r((n-i)T)h\left(\left(i+\frac{q}{Q}\right)T\right) = x\left(\left(n-\frac{q}{Q}\right)T\right). \tag{3.154}$$

This characteristic is illustrated in Figure 3.58 where a parallel bank of Q filters, operating at the low sample rate $1/T$ is shown. Each filter in the filterbank is a downsampled version of the matched filter, except with a different index offset. The impulse response for $h_q(nT)$ is

$$h_q(nT) = h\left(nT + \frac{q}{Q}T\right) \qquad \text{for } q = 0, 1, \ldots, Q-1. \tag{3.155}$$

The data samples $r(nT)$ form the input to all the filters in the filterbank simultaneously. The desired phase shift of the output is selected by connecting the output to the appropriate filter in the filterbank.

To see that the output of the q-th filter in the polyphase filter bank given by (3.154) does indeed produce the desired result given by (3.124), assume for the moment that T_i/T in (3.124) is sufficiently close to one so that $m(k) = n$. Then (3.125) becomes

$$x(kT_i) = \sum_i x\left((k-i)T\right) h_I\left((i+\mu(k))T\right). \tag{3.156}$$

Since the polyphase filterbank implementation uses the matched filter as the interpolation filter, the input data sequence $r(nT_s)$ in (3.154) plays the role of the matched filter output $x(nT)$ in (3.125) and the matched filter $h(nT)$ in (3.154) plays the role of the interpolation filter in (3.125). The comparison shows that the ratio of the polyphase filter stage index q to the number of filterbank stages Q plays the same role as the fractional interval $\mu(k)$ in the interpolation filter. In this way, the polyphase filterbank implements the interpolation defined by (3.125) with a quantized fractional interval. The degree of quantization is controlled by the number of polyphase filter stages in the filterbank. The observations regarding the behavior of $\mu(k)$ above apply to the filter stage index q for the cases where T and T_s are not commensurate.

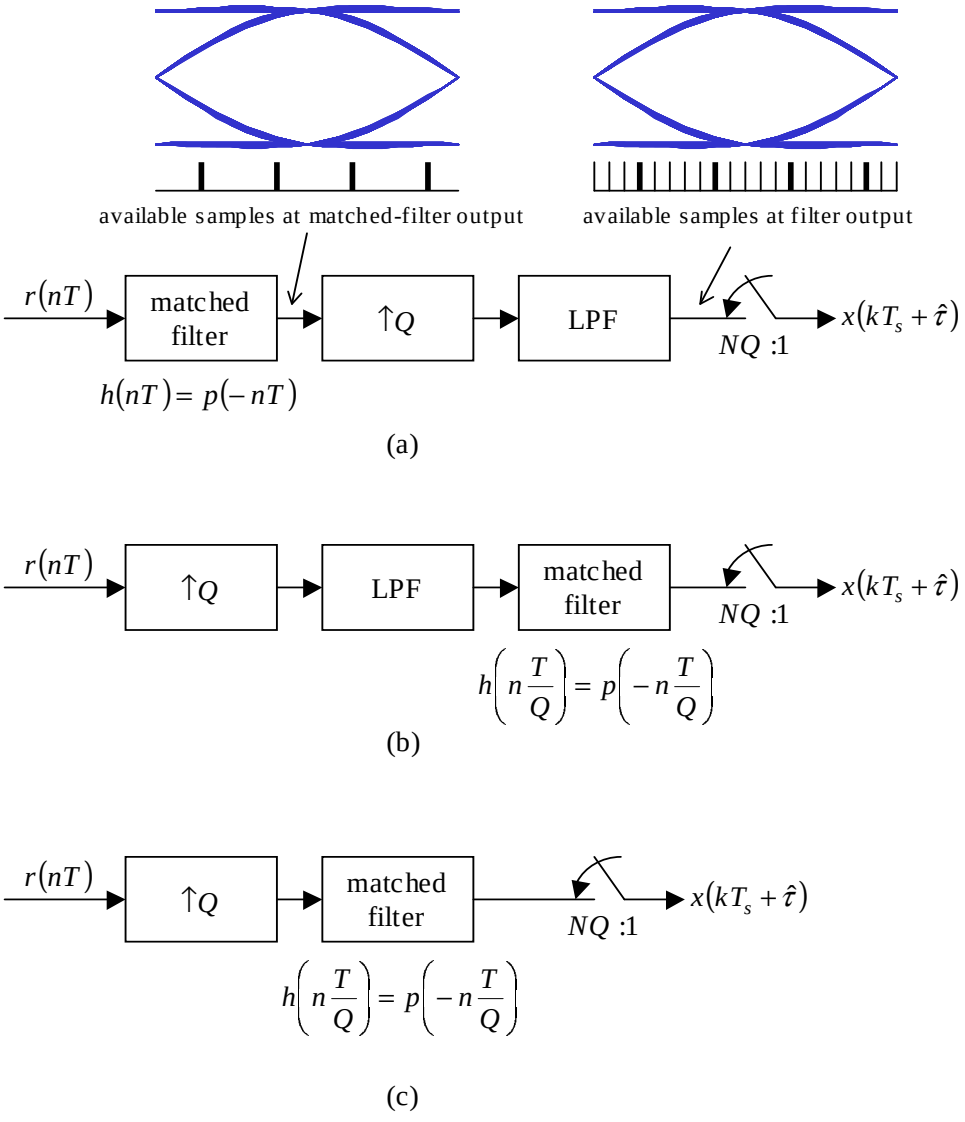


Figure 3.57: An upsample approach to interpolation. (a) Upsample and interpolation applied to the matched filter output. (b) Upsample and interpolation applied the matched filter input. (c) Using the matched filter for both interpolation and shaping.

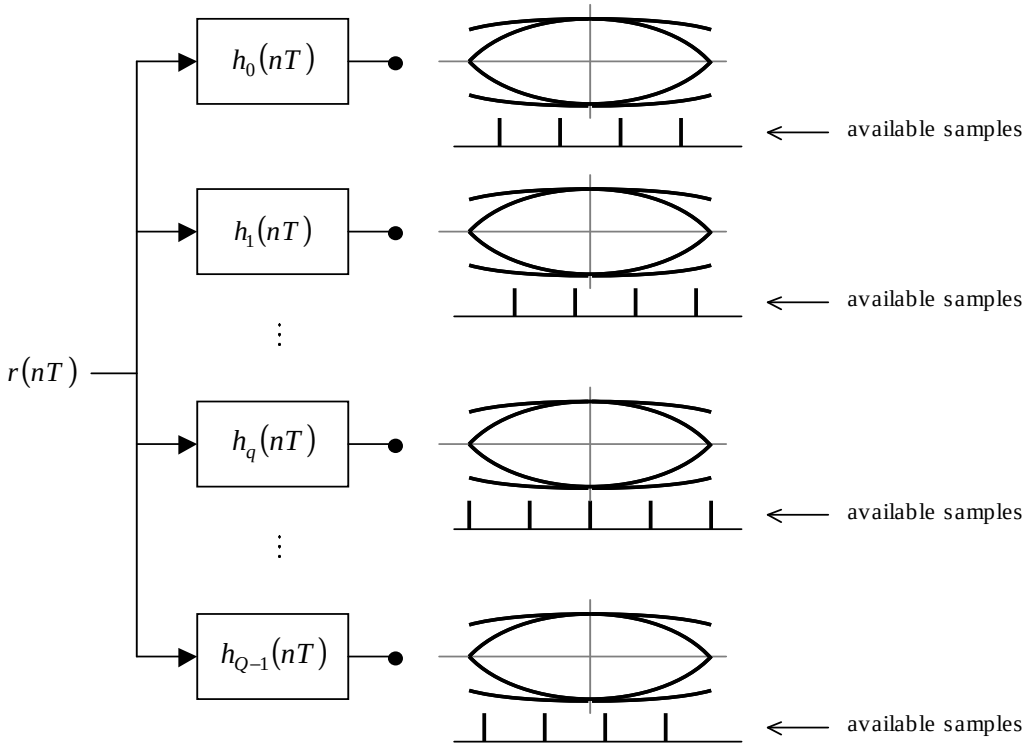


Figure 3.58: Polyphase matched-filter filterbank outputs illustrating how each filter in the filterbank produces an output sequence with a different delay.

Interpolation Control

The purpose of the interpolator control block in Figure 3.41 to provide the interpolator with the k-th basepoint index $m(k)$ and the k-th fractional interval $\mu(k)$. The basepoint index is usually not computed explicitly but rather identified by a signal often called a *strobe*. Two commonly used methods for interpolation control are a counter-based method and a recursive method.

Modulo-1 Counter Interpolation Control For the case where interpolants are required every N samples, interpolation control can be accomplished using a modulo-1 counter designed to underflow every N samples where where the underflows are aligned with the basepoint indexes. A block diagram of this approach is shown in Figure 3.59. The T-spaced samples of the matched filter input are clocked into the matched filter with the same clock used to update the counter. A decrementing modulo-1 counter is shown here as it simplifies the computation of the fractional interval. An incrementing modulo-1 counter could also be used and is explored in a homework problem.

The counter decrements by $1/N$ on average so that underflows occur every N samples on average. The loop filter output $v(n)$ adjusts the amount by which the counter decrements. This is done to align the underflows with the sample times of the desired interpolant. When operating properly, the modulo-1 counter underflows occur a clock period after the desired interpolant as illustrated in Figure 3.60. The underflow condition is indicated by a strobe and is used to identify to the interpolator that the previous sample was the basepoint index for the desired interpolant.

The fractional interval may be computed directly from the contents of the modulo-1 counter on underflow. The counter value $\eta(n)$ satisfies the recursion

$$\eta(n) = (\eta(n-1) - W(n-1)) \mod 1 \tag{3.157}$$

where $W(n) = 1/N + v(n)$ is the counter input and is the current estimate of the ratio T_i/T. The counter value immediately preceding kT_i (the desired interpolant time) is $\eta(m(k))$ and the counter value immediately following the kT_i is $1-\eta(m(k)+1)$. Using similar triangles, the counter values and fractional interval satisfy the relationship

$$\frac{\mu(k)T}{\eta(m(k))} = \frac{(1-\mu(k))T}{1-\eta(m(k)+1)} \tag{3.158}$$

which can be solved for $\mu(k)$:

$$\mu(k) = \frac{\eta(m(k))}{1-\eta(m(k)+1)+\eta(m(k))} = \frac{\eta(m(k))}{W(m(k))}. \tag{3.159}$$

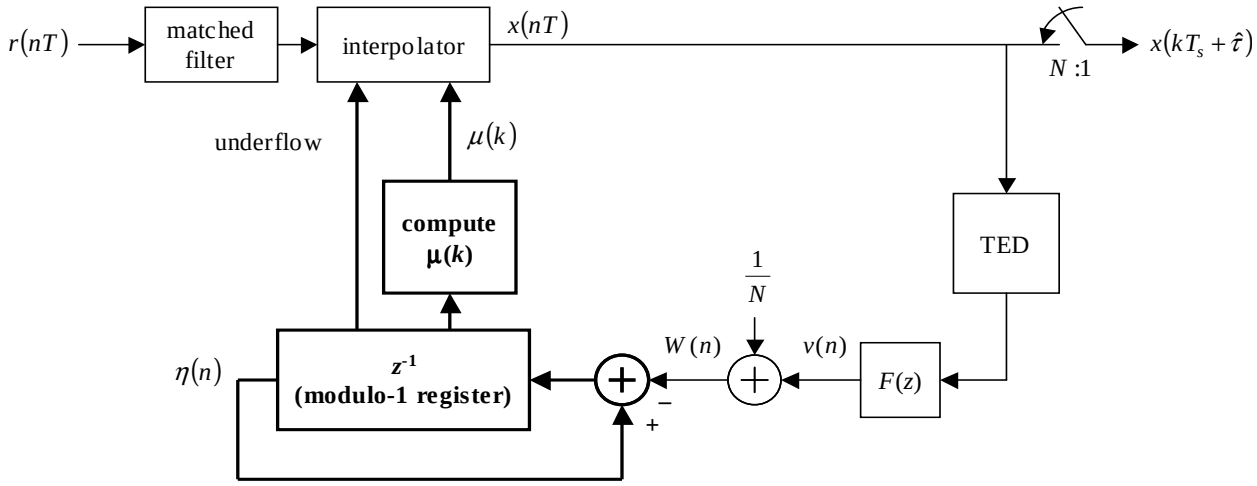


Figure 3.59: Modulo-1 counter for interpolation control in a baseband PAM system. The basepoint index is identified by the underflow strobe and the fractional interval updated using the counter contents on underflow.

The underflow period (in samples) of the NCO is

$$\frac{1}{W(n)} = \frac{1}{\frac{1}{N} + v(n)} \tag{3.160}$$

$$= \frac{N}{1 + Nv(n)}. \tag{3.161}$$

When in lock, $v(n)$ is zero on average an the NCO underflow period is N samples on average. During acquisition, $v(n)$ adjusts the underflow period to align the underflow events with the symbol boundaries as described above. An important caveat is lurking in the details when using NCO interpolation control. A positive phase error $(\tau - \hat{\tau}(k) > 0)$ means $\hat{\tau}(k+1)$ must be greater than $\hat{\tau}(k)$. This is accomplished by increasing the period of NCO underflows. The underflow period is increased by forcing $1 + Nv(n) < 1$ which, in turn, requires $v(n) < 0$. In the same way, a negative phase error $(\tau - \hat{\tau}(k) < 0)$ means $\hat{\tau}(k+1)$ must be less than $\hat{\tau}(k)$. This is accomplished by decreasing the period of NCO underflows. The underflow period is decreased by forcing $1 + Nv(n) > 1$ which, in turn, requires $v(n) > 0$. Thus the sign of the phase error the opposite the sign of what is required by the NCO controller for proper operation. This characteristic can be easily accommodated by changing the sign on the TED gain: i.e., using $-K_p$ in stead of K_p.

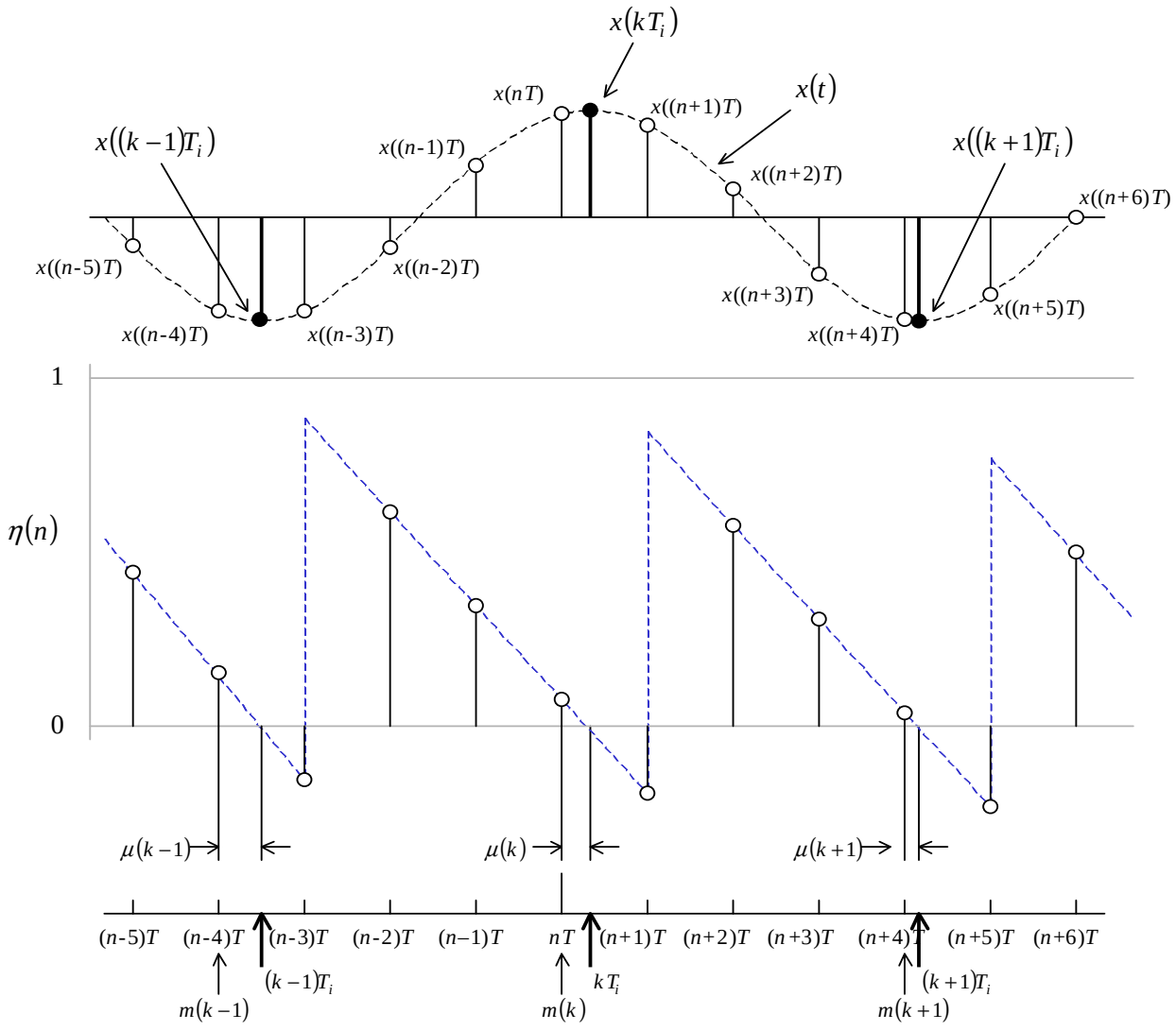


Figure 3.60: Illustration of the relationship between the available samples, the desired interpolants, and the modulo-1 counter contents.

Recursive Interpolation Control The relationship for recursive interpolation control can be obtained by writing the expressions for two successive interpolation instants as

$$\begin{aligned} kT_i &= (m(k) + \mu(k))T \\ (k+1)T_i &= (m_{k+1} + \mu_{k+1})T \end{aligned} \tag{3.162}$$

and subtracting the two to obtain the recursion

$$m_{k+1} = m(k) + \frac{T_i}{T} + \mu(k) - \mu_{k+1}. \tag{3.163}$$

Since m_k and m_{k+1} are integers, the fractional part of the right-hand side of (3.163) must be zero from which the recursion for the fractional interval is obtained:

$$\mu_{k+1} = \left(\mu(k) + \frac{T_i}{T}\right) \mod 1. \tag{3.164}$$

Since $0 \leq \mu_{k+1} < 1$, the relationship

$$m_{k+1} + \mu_{k+1} = m_k + \frac{T_i}{T} + \mu(k) < m_{k+2} \tag{3.165}$$

must hold from which the recursion on the sample count increment is

$$m_{k+1} - m(k) = \left\lfloor \frac{T_i}{T} + \mu(k) \right\rfloor. \tag{3.166}$$

The sample count increment is a more useful quantity than the actual basepoint index because any finite-precision counter used to compute and/or store $m(k)$ would eventually overflow. As was the case with the counter-based control, the ratio T_i/T required by (3.164) and (3.166) estimated by $W(n) = 1/N + v(n)$ where $v(n)$ is the output of the loop filter.

Examples

Two examples are provided to put all the pieces together. Both examples use binary PAM as the modulation. The first uses the maximum likelihood timing error detector and operates at 16 samples/symbol. The second uses the zero crossing detector and operates at 2 samples/symbol.

Binary PAM with MLTED This example illustrates the use of the MLTED error detector and the NCO interpolator control for binary PAM. A block diagram is illustrated in Figure 3.61. The pulse shape is the square-root raised cosine with 50% excess bandwidth. The received signal is sampled at a rate equivalent to $N = 16$ samples/sec. Since $r(t)$ is 8 times oversampled, a linear interpolator is adequate. Note that this system is different from the one suggested by the system in Figures 3.41 and 3.59 in that the interpolator precedes the matched filter. This was done to illustrate that the interpolator may be placed at either location in the processing chain.

Samples of the received signal are filtered by a discrete-time matched filter and derivative matched filter in parallel. The outputs are downsampled to 1 sample/symbol as directed by the NCO controller. The timing error signal is formed as prescribed by the decision directed MLTED (3.106). In this implementation, the loop filter and NCO operate at the high sample rate of 16 samples/symbol. As a consequence, the error signal, which is updated at 1 sample/symbol, must be upsampled. The upsampling is performed by inserting zeros in between the error signal updates. The error signal is filtered by a discrete-time proportional-plus-integrator loop filter. The loop filter output forms the input to a decrementing modulo-1 register or NCO. The NCO controls the interpolation process as described in Section 3.4.3. Since the interpolator is not performing a sample rate change, there is no need to provide basepoint index information. The interpolator produces one interpolant for each input sample.

The timing synchronization system can also be described as a computer program. The challenge with this approach is that the timing synchronization system is a parallel system while a computer program is a sequential representation. This is a common problem in system modeling: simulating an inherently parallel system on a sequential processor. A common method for generating the sequential representation is to write a program loop where each pass through the loop represents a clock cycle in the digital system. Within the loop, the parallel arithmetic (combinatorial) expressions are evaluated in topological order. Next the registered values (memory) are updated.

The code segment listed below is written using a Matlab-style syntax and consists of a `for` loop iterating on the samples of the received signal. The structure of the `for` loop follows the convention of updating the arithmetic (or combinatorial) quantities first and the registered values (or memory) last. The variable names used in the code segment are the same as those used in

Figure 3.61 with the following additions:

`r_prev`	a scaler holding the previous input value. This value is needed by the linear interpolator to compute the desired interpolants
`rI`	a scalar representing the interpolant $r\left(nT + \hat{\tau}\right)$.
`mf`	a row vector consisting of samples of the matched filter impulse response
`dmf`	a row vector consisting of samples of the derivative matched filter impulse response
`rIBuff`	a column vector of interpolator outputs used by the matched filter and derivative matched filter
`xx`	a vector holding the matched filter outputs $x\left(kT_s + \hat{\tau}\right)$ for $k = 0, 1, \ldots$.

The code segment is not written in the most efficient manner, but rather to explain the sequence of operations for proper PLL operation.

```
initialize
for n=1:length(r)

    % evaluate arithmetic expressions in topological order

    if NCO < 0
        underflow = 1;
    else
        underflow = 0;
    end
    if underflow == 1
        mu = mu_temp;
    end
    rI = mu*r(n) + (1 - mu)*r_prev;
    x = mf*[rI; rIBuff];
    xdot = dmf*[rI; rIBuff];
    if underflow == 1
        e = sign(x)*xdot;
    else
        e = 0;
    end
```

```
    vp = K1*e;              % proportional component of loop filter
    vi = vi + K2*e;         % integrator component of loop filter
    v = vp + vi;            % loop filter output
    W = 1/N + v;            % NCO control word

    % update registers

    mu_temp = NCO/W;
    if underflow == 1
        NCO = NCO + 1 - W;
    else
        NCO = NCO - W;
    end
    IBuff = r(n);
    rIBuff = [rI; rIBuff(1:end-1)];

    % update output

    if underflow == 1
        xx(k) = x;
        k = k + 1;
    end
end
```

As an example, consider a symbol timing PLL with performance requirements $B_nT_s = 0.005$ and $\zeta = 1/\sqrt{2}$. Figure 3.43 gives the phase detector gain $K_p = 0.235$. As explained in Section 3.4.3, $K_p = -0.235$ should be used when interpolation control is based on a decrementing NCO. The phase detector gain also needs to be adjusted to account for the fact that the phase detector operates at 1 sample/symbol while the loop filter and NCO operate at 16 samples/symbol. Since zeros are inserted between the updates of the timing error, the timing error seen by the loop filter is $1/N$ what it would be otherwise. Hence $K_p = -0.235/16 = -0.0147$. Using $N = 16$, the loop constants given by (3.24) are

$$K_1K_pK_0 = 9.9950 \times 10^{-4}$$
$$K_2K_pK_0 = 4.9976 \times 10^{-7}.$$

Finally, solving for K_1 and K_2 using $K_p = -0.0147$ and $K_0 = 1$ gives

$$K_1 = -6.8051 \times 10^{-2}$$
$$K_2 = -3.4026 \times 10^{-5}.$$

A plot of the timing error signal $e(k)$ and the fractional interval $\mu(k)$ are illustrated in Figure 3.62 for 600 random symbols. The plot of $\mu(k)$ shows that the loop locks after about 500 symbols at the steady state value $\mu = 0.5$. The plot of $\mu(k)$ looks "noisy." This is due to the self noise produced by the timing error detector.

While the interpolator does not require basepoint index information from the NCO controller, the rate change at the matched filter and derivative matched filter outputs does require basepoint index information. During acquisition, the PLL has to find the right basepoint index for the desired matched filter output. This search is indicated by the "ramping" effect observed in the plot of μ during the first 200 symbols. Each time μ touches zero, it wraps to $\mu = 1$ and reduces the interval between the current basepoint index and the next basepoint index by 1.

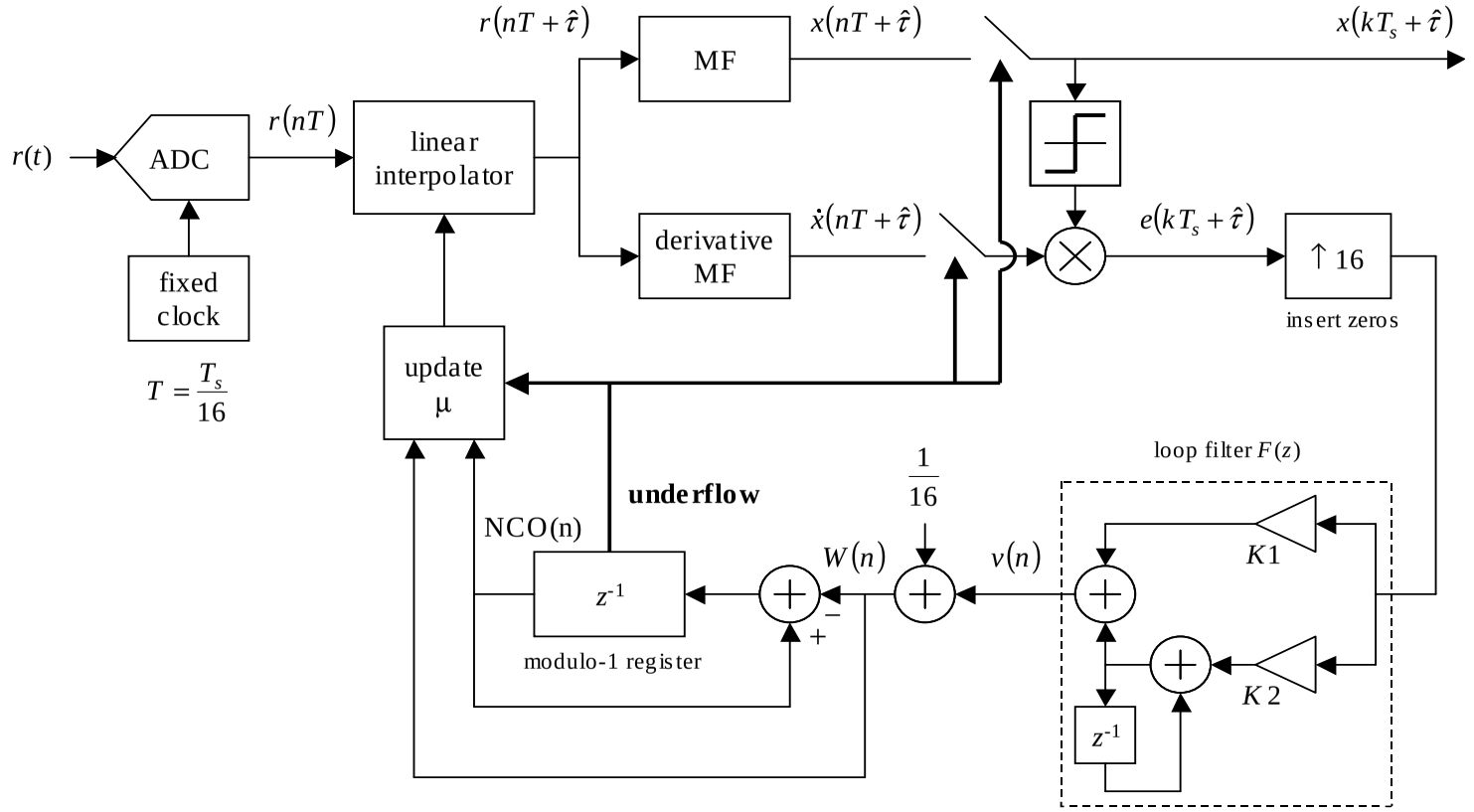


Figure 3.61: Binary PAM symbol timing synchronization system based on the MLTED using a linear interpolator and an proportional-plus-integator loop filter.

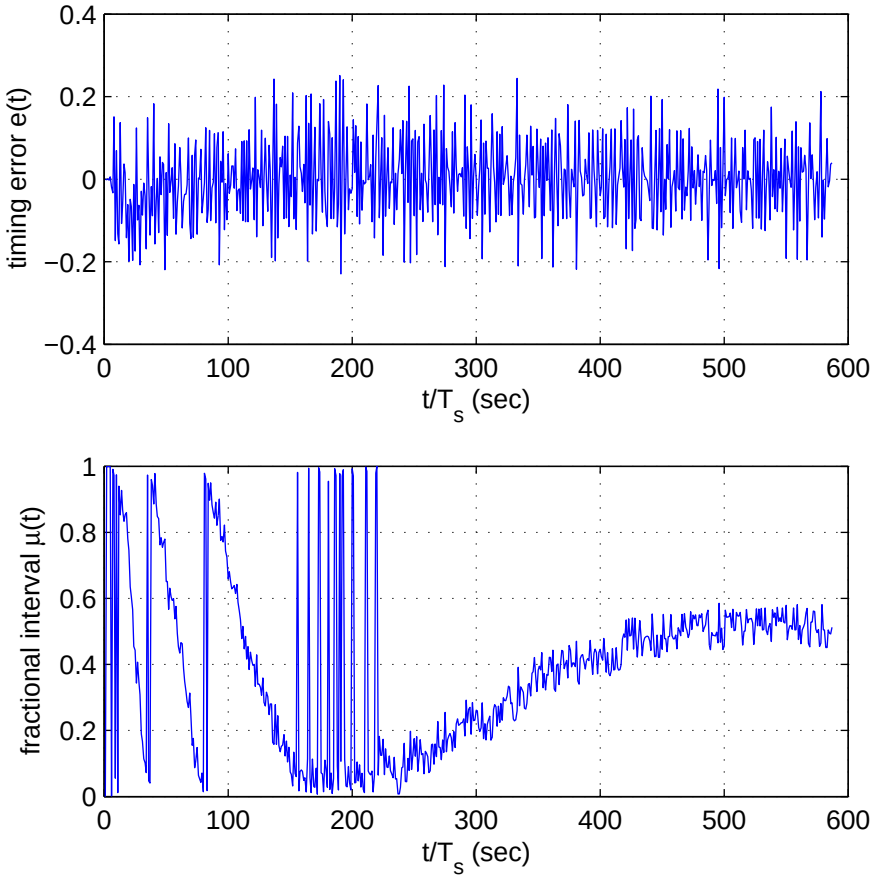


Figure 3.62: Timing error signal and fractional interpolation interval for the symbol timing synchronization system illustrated in Figure 3.61.

A practical variation on this design is illustrated in Figure 3.63. In this example, interpolation is moved to the output side of the matched filter and derivative matched filter. This placement requires two interpolators operating in parallel as shown. In this architecture, the two interpolators are required to perform a sample rate conversion. Hence the underflow strobe from the NCO controller is required to provide basepoint index information to the interpolators. Relative to the architecture illustrated in Figure 3.61, this architecture has the disadvantage that two interpolators are required. But, it has the advantage that the matched filter and derivative matched filters are not in the closed loop path.

As before, the received signal is sampled at a rate equivalent to 16 samples/symbol to produce the samples $r(nT)$. These samples are filtered by a matched filter and derivative matched filter operating at 16 samples/symbol to produce the outputs $x(nT)$ and $\dot{x}(nT)$. These outputs form the inputs to two linear interpolators also operating in parallel. The interpolators produce one interpolant per symbol as directed by the NCO controller. The NCO controller provides both the basepoint index (via the underflow strobe) and the fractional interval. The two interpolator outputs $x(kT_s + \tau)$ and $\dot{x}(kT_s + \tau)$ are used to compute the timing error signal $e(k)$ given by (3.106). The error signal is upsampled by 16 to match the operating rate of the loop filter and NCO controller.

An equivalent description using a Matlab style code segment is shown below. The code segment uses the same variable names as Figure 3.63 with the following additions:

`x_prev`	a scaler holding the previous matched filter output. This value is required by the linear interpolator operating on the matched filter outputs.
`xdot_prev`	a scaler holding the previous derivative matched filter output. This value is required by the linear interpolator operating on the derivative matched filter outputs.
`xI`	a scalar representing the interpolant $x\left(kT_s + \hat{\tau}\right)$.
`xdotI`	a scalar representing the interpolant $\dot{x}\left(kT_s + \hat{\tau}\right)$.
`xx`	a vector holding the matched filter outputs $x\left(kT_s + \hat{\tau}\right)$ for $k = 0, 1, \ldots$.

The code segment consists of a `for` loop that iterates on the matched filter and derivative matched filter output samples. The code segment is not written in the most efficient manner, but rather to explain the sequence of operations for proper PLL operation.

```
initialize
for n=1:length(x)
```

```
% evaluate arithmetic expressions in topological order

if NCO < 0
    underflow = 1;
else
    underflow = 0;
end
if underflow == 1
    mu = mu_temp;
end
if underflow == 1
    xI = mu*x(n) + (1 - mu)*x_prev;
    xdotI = mu*xdot(n) + (1 - mu)*xdot_prev;
    e = sign(xI)*xdotI;
else
    e = 0;
end
vp = K1*e;              % proportional component of loop filter
vi = vi + K2*e;         % integrator component of loop filter
v = vp + vi;            % loop filter output
W = 1/N + v;            % NCO control word

% update registers

mu_temp = NCO/W;
if underflow == 1
    NCO = NCO + 1 - W;
else
    NCO = NCO - W;
end
x_prev = x(n);
xdot_prev = xdot(n);

% update output

if underflow == 1
    xx(k) = xI;
```

```
        k = k + 1;
    end
end
```

An example of the phase error and fractional interval are plotted in Figure 3.64 for 600 random symbols. The loop filter constants are identical to those used previously. As before, the timing PLL locks after about 500 symbols. The shape of the fractional interval plot is quite similar to the fractional interval plot in Figure 3.62. Differences are due to the placement of the matched filter and derivative matched filter. In Figure 3.62, the matched filter and derivative matched filters are in the closed loop path while in Figure 3.64 they are not.

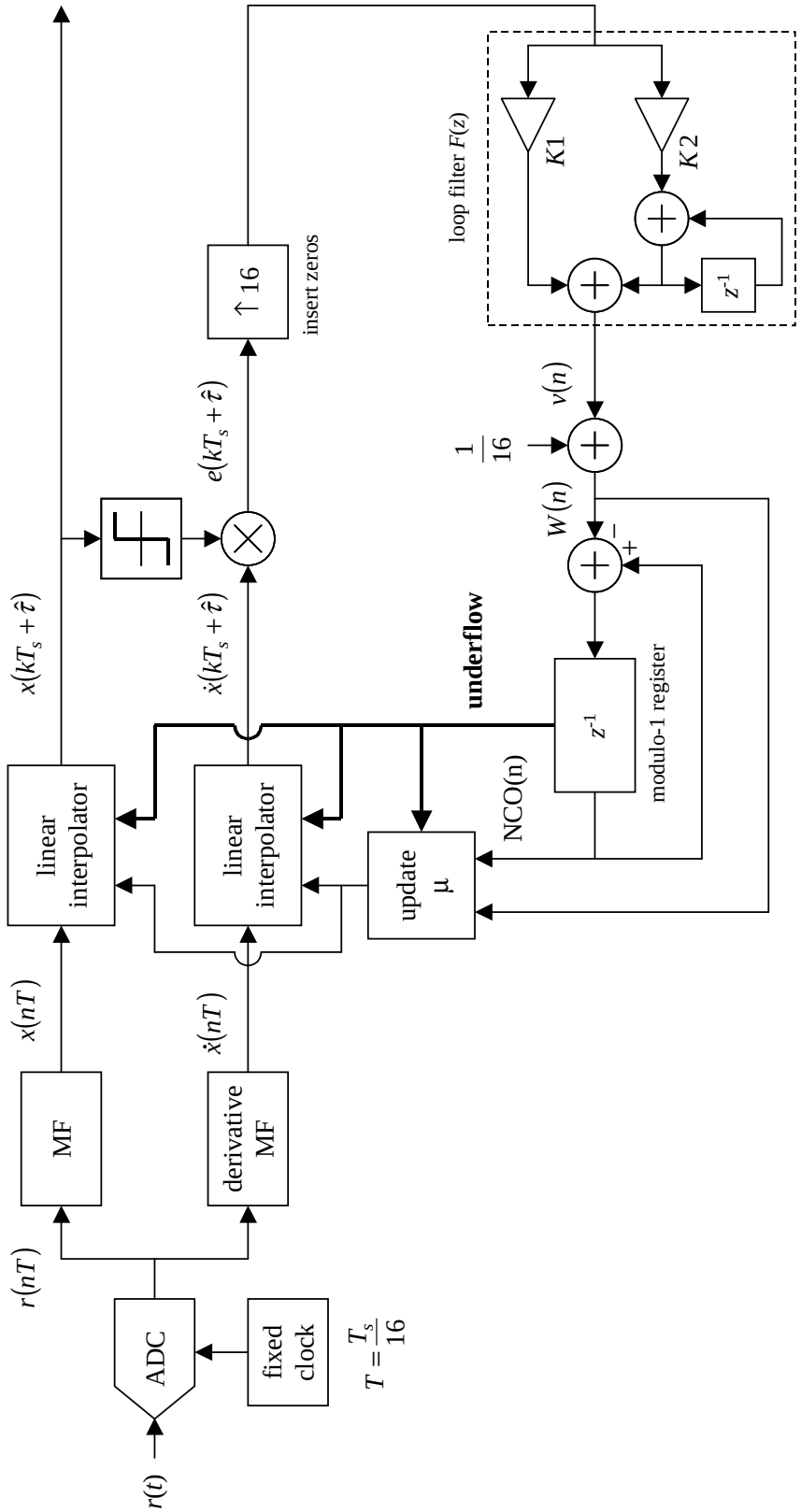


Figure 3.63: Binary PAM symbol timing synchronization system based on the MLTED using a linear interpolator and an proportional-plus-integator loop filter.

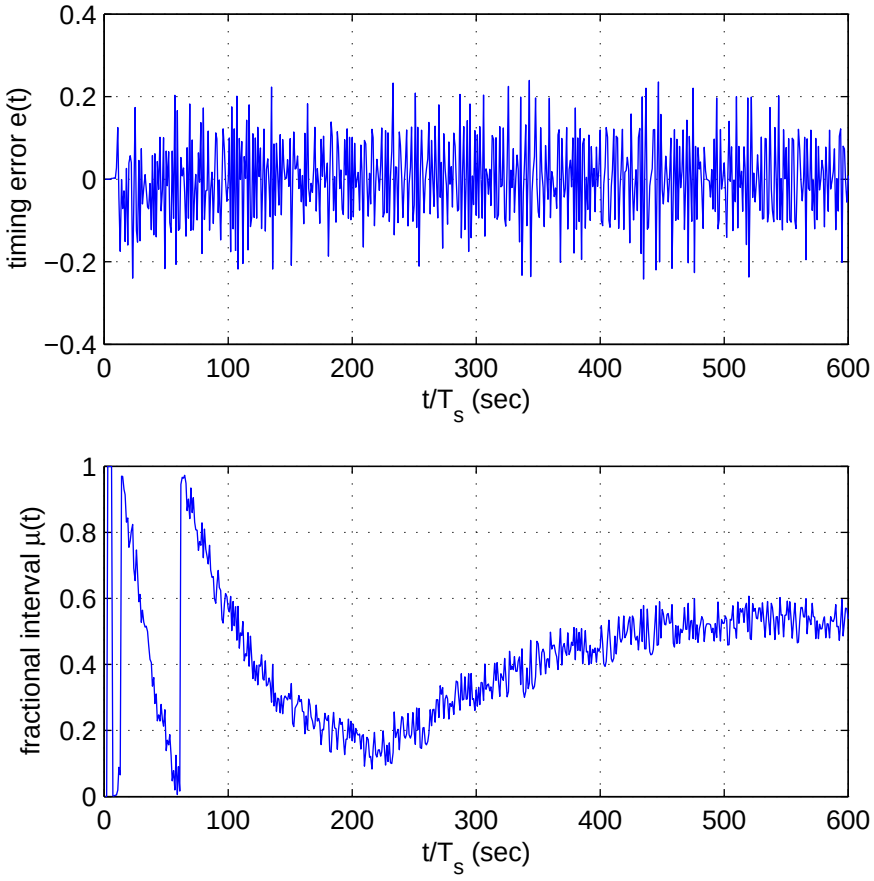


Figure 3.64: Timing error signal and fractional interpolation interval for the symbol timing synchronization system illustrated in Figure 3.63.

Binary PAM with ZCTED This example illustrates the use of the ZCTED error detector and the NCO interpolator control for binary PAM. A block diagram is illustrated in Figure 3.61. The pulse shape is the square-root raised cosine with 50% excess bandwidth. The received signal is sampled at a rate equivalent to $N = 2$ samples/symbol. Samples of the received signal are filtered by a discrete-time matched filter operating at 2 samples/symbol. The matched filter outputs $x(nT)$ are used by the piece-wise parabolic interpolator to compute the interpolants $x(nT + \hat{\tau})$. These interpolants form the input to the zero crossing detector described in Section 3.4.3 and given by (3.115). The timing error signal is updated at 1 sample/symbol. Since the loop filter and NCO control operate at $N = 2$ samples/symbol, the timing error signal is upsampled by inserting a zero(s) in between the updates. The upsampled timing error signal is filtered by the proportional-plus-integrator loop filter. The loop filter output forms the input to a decrementing modulo-1 register or NCO. The NCO controls the interpolation process as described in Section 3.4.3.

A code segment modeling the system is listed below. It is written using a Matlab-style syntax and consists of a `for` loop iterating on the samples of the matched filter output. The structure of the `for` loop follows the convention of updating the arithmetic (or combinatorial) quantities first and the registered values (or memory) last. The variable names used in the code segment are the same as those used in Figure 3.61 with the following additions:

`IBuff`	a 3×1 vector holding the previous matched filter values needed to compute the interpolants
`xI`	a scalar representing the interpolant $x\left(nT + \hat{\tau}\right)$
`TEDBuff`	a 2×1 column vector of interpolator outputs used by the timing error detector
`xx`	a vector holding the matched filter outputs $x\left(kT_s + \hat{\tau}\right)$ for $k = 0, 1, \ldots$.

```
for n=1:length(x)

    % evaluate arithmetic expressions in topological order

    if NCO < 0
        underflow = 1;
    else
        underflow = 0;
    end
    if underflow
        mu = mu_temp;
```

```
    end
    v2 = 1/2*[1, -1, 1, 1]*[x(n); IBuff];  % Farrow structure for the
    v1 = 1/2*[-1, 3, 1, -1]*[x(n); IBuff]; % piecewise parabolic
    v0 = [0, 0, 1, 0]*[x(n); IBuff];       % interpolator
    xI = (mu*v2 + v1)*mu + v0;             % interpolator output
    if underflow == 1
        e = TEDBuff(1) * (sign(TEDBuff(2)) - sign(xI));
    else
         e = 0;
    end
    vp = K1*e;             % proportional component of loop filter
    vi = vi + K2*e;        % integrator component of loop filter
    v = vp + vi;           % loop filter output
    W = 1/N + v;           % NCO control word

    % update registers

    mu_temp = NCO/W;
    if underflow == 1
        NCO = NCO + 1 - W;
    else
        NCO = NCO - W;
    end
    IBuff = [x(n); IBuff(1:end-1)];
    TEDBuff = [xI; TEDBuff(1)];

    % update output

    if underflow == 1
        xx(k) = xI;
        k = k + 1;
    end
end
```

As an example, consider a symbol timing PLL with performance requirements $B_nT_s = 0.01$ and $\zeta = 1/\sqrt{2}$. Figure 3.49 gives the phase detector gain $K_p = 2.7$. As explained in Section 3.4.3, $K_p = -2.7$ should be used when the interpolation control is based on a decrementing NCO. The phase detector gain also needs to be adjusted to account for the fact that the phase detector operates

at 1 sample/symbol while the loop filter and NCO operate at 2 samples/symbol. Since zeros are inserted between the updates of the timing error, the timing error seen by the loop filter is $1/N$ what it would be otherwise. Hence $K_p = -2.7/2 = -1.35$. Using $N = 2$, the loop constants given by (3.24) are

$$K_1 K_p K_0 = 1.5872 \times 10^{-2}$$
$$K_2 K_p K_0 = 1.2698 \times 10^{-4}.$$

Finally, solving for K_1 and K_2 using $K_p = -1.35$ and $K_0 = 1$ gives

$$K_1 = -1.1757 \times 10^{-2}$$
$$K_2 = -9.4061 \times 10^{-5}.$$

A plot of the timing error signal $e(k)$ and the fractional interval $\mu(k)$ are illustrated in Figure 3.66 for 600 random symbols. The plot of $\mu(k)$ shows that the loop locks after about 300 symbols at the steady state value $\mu = 0.5$. Since the ZCTED does not produce any self-noise, the plot of μ has a much "cleaner" look than the plot of μ for the MLTED in Figure 3.62.

The code listing above does not work for the case of sample clock frequency offset. That is, for the case $T \neq T_s/2$, the code must be modified to account for the cases when an interpolant is required during two consecutive clock cycles ($T > T_s/2$) or for the case when two clock cycles occur between consecutive interpolants ($T < T_s/2$).

The case $T > T_s/2$ is illustrated in Figure 3.67. The desired samples appear to "slide to the left" since the samples are spaced slightly further apart than $T_s/2$. Most of the time, a desired matched filter interpolant is produced for every two available matched filter samples. Since $T > T_s/2$, a residual timing error accumulates. As the residual timing error accumulates, the fractional interval $\mu(k)$ decreases with time as shown. Eventually the accumulated residual timing error exceeds a sample period. This coincides with $\mu(k)$ decreasing to 0 and wrapping around to 1. When this occurs desired matched filter interpolants occur one sample apart instead of the normal two samples apart. As shown, when this occurs, one of the samples needed by the ZCTED is never produced. This missing sample must be inserted or "stuffed" into the ZCTED registers to ensure proper operation after the "wrap around."

The case $T < T_s/2$ is illustrated in Figure 3.68. In this case, the desired samples appear to "slide to the right" since the samples are spaced slightly closer together than $T_s/2$. Most of the time, a desired matched filter interpolant is produced for every two available matched filter samples. Since $T < T_s/2$, a residual timing error accumulates. As the residual timing error accumulates, the fractional interval $\mu(k)$ increases with time as shown. Eventually the accumulated residual timing error exceeds a sample period. This coincides with $\mu(k)$ exceeding 1 and wrapping

around to 0. When this occurs, the desired matched filter interpolants are spaced three samples apart instead of the normal two. As a consequence, the interpolator produces an extra sample that should be ignored, or "skipped" by the ZCTED. This is accomplished by not shifted the ZCTED registers after the "wrap around."

A modified segment of code to account for this condition is shown below. A new variable `old_underflow` is introduced. This variable, together with `underflow` are used to determine whether normal operation, "stuffing," or "skipping" should occur. Again, the code is not written in the most efficient manner, but rather to provide a description of the subtleties associated with proper operation of the ZCTED.

```
for n=1:length(x)

    % evaluate arithmetic expressions in topological order

    if NCO < 0
        underflow = 1;
    else
        underflow = 0;
    end
    if underflow
        mu = mu_temp;
    end
    v2 = 1/2*[1, -1, 1, 1]*[x(n); IBuff];  % Farrow structure for the
    v1 = 1/2*[-1, 3, 1, -1]*[x(n); IBuff]; % piecewise parabolic
    v0 = [0, 0, 1, 0]*[x(n); IBuff];       % interpolator
    xI = (mu*v2 + v1)*mu + v0;             % interpolator output
    if underflow == 1 & old_underflow == 0
        e = TEDBuff(1) * (sign(TEDBuff(2)) - sign(xI));
    else
        e = 0;
    end
    vp = K1*e;             % proportional component of loop filter
    vi = vi + K2*e;        % integrator component of loop filter
    v = vp + vi;           % loop filter output
    W = 1/N + v;           % NCO control word

    % update registers
```

```
    mu_temp = NCO/W;
    if underflow == 1
        NCO = NCO + 1 - W;
    else
        NCO = NCO - W;
    end
    IBuff = [x(n); IBuff(1:end-1)];
    if underflow == 0 & old_underflow = 0
        TEDBuff = TEDBuff;                      % skip current sample
    elseif underflow == 0 & old_underflow == 1
        TEDBuff = [xI; TEDBuff(1)];             % normal operation
    elseif underflow == 1 & old_underflow == 0
        TEDBuff = [xI; TEDBuff(1)];             % normal operation
    elseif underflow == 1 & old_underflow == 1
        TEDBuff = [xI; 0; TEDBuff(1)];          % stuff missing sample
    end
    old_underflow = underflow;

    % update output

    if underflow == 1
        xx(k) = xI;
        k = k + 1;
    end
end
```

As this code segment illustrates, the "upsampled by 2" function inserted in between the timing error detector and the loop filter is only an abstraction. The upsample operation is performed by inserting zeros in between the timing error updates. Most of the time 1 zero is inserted. But sometimes no zeros are inserted; sometimes 2 zeros are inserted.

As an example of operation for the case where the sample clock frequency is slightly higher than 2 samples/symbol (i.e., $T < T_s/2$), suppose the samples $r(nT)$ were obtained where T satisfied

$$T = \frac{T_s}{2 + \frac{1}{400}}$$

or, what is equivalent

$$\text{sample rate} = \left(2 + \frac{1}{400}\right) \times \text{symbol rate}.$$

The sampling clock frequency is 1/400 of the symbol rate faster than 2 samples/symbol. The error signal and fractional interval for the same timing PLL considered previously are plotted in Figure 3.69. As expected, the fractional interval ramps from 0 to 1 and rolls over every 400 symbol times. This is because the frequency error in the sample clock is 1/400 of the symbol rate. The error signal indicates that the timing PLL locks after about 100 symbols. This case is the symbol timing PLL equivalent of a phase ramp input for the generic PLL reviewed in Section 3.2.1 and explained in Section A.2.1 in Appendix A.

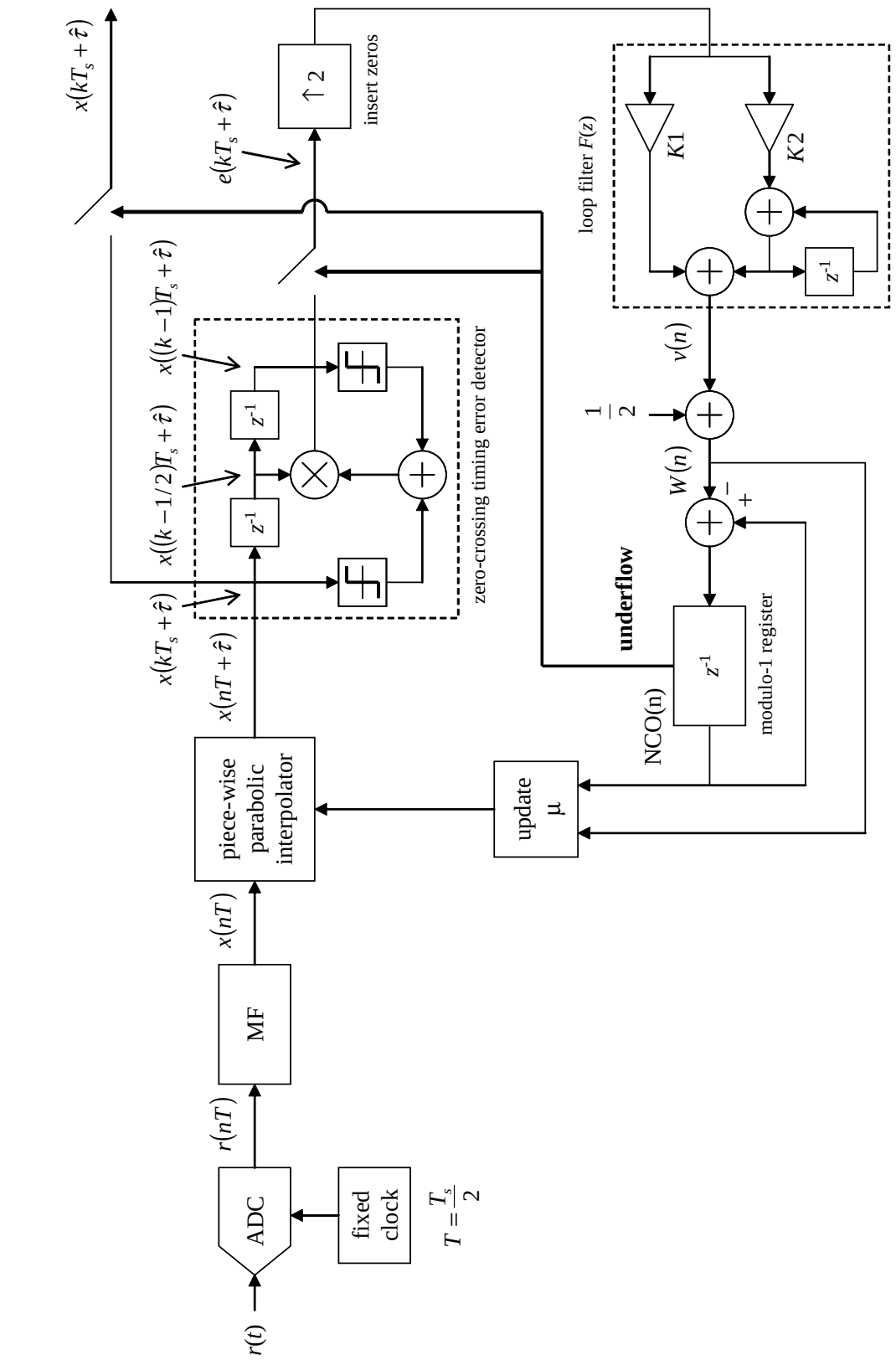


Figure 3.65: Binary PAM symbol timing synchronization system based on the ZCTED using a linear interpolator and an proportional-plus-integator loop filter.

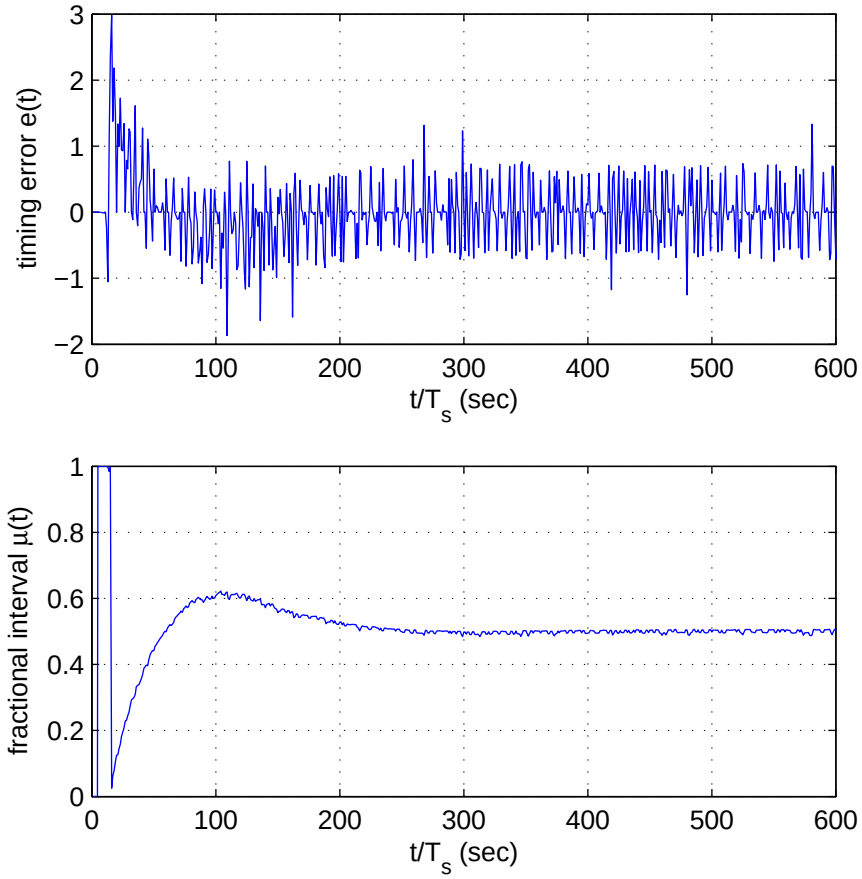


Figure 3.66: Timing error signal and fractional interpolation interval for the symbol timing synchronization system illustrated in Figure 3.65.

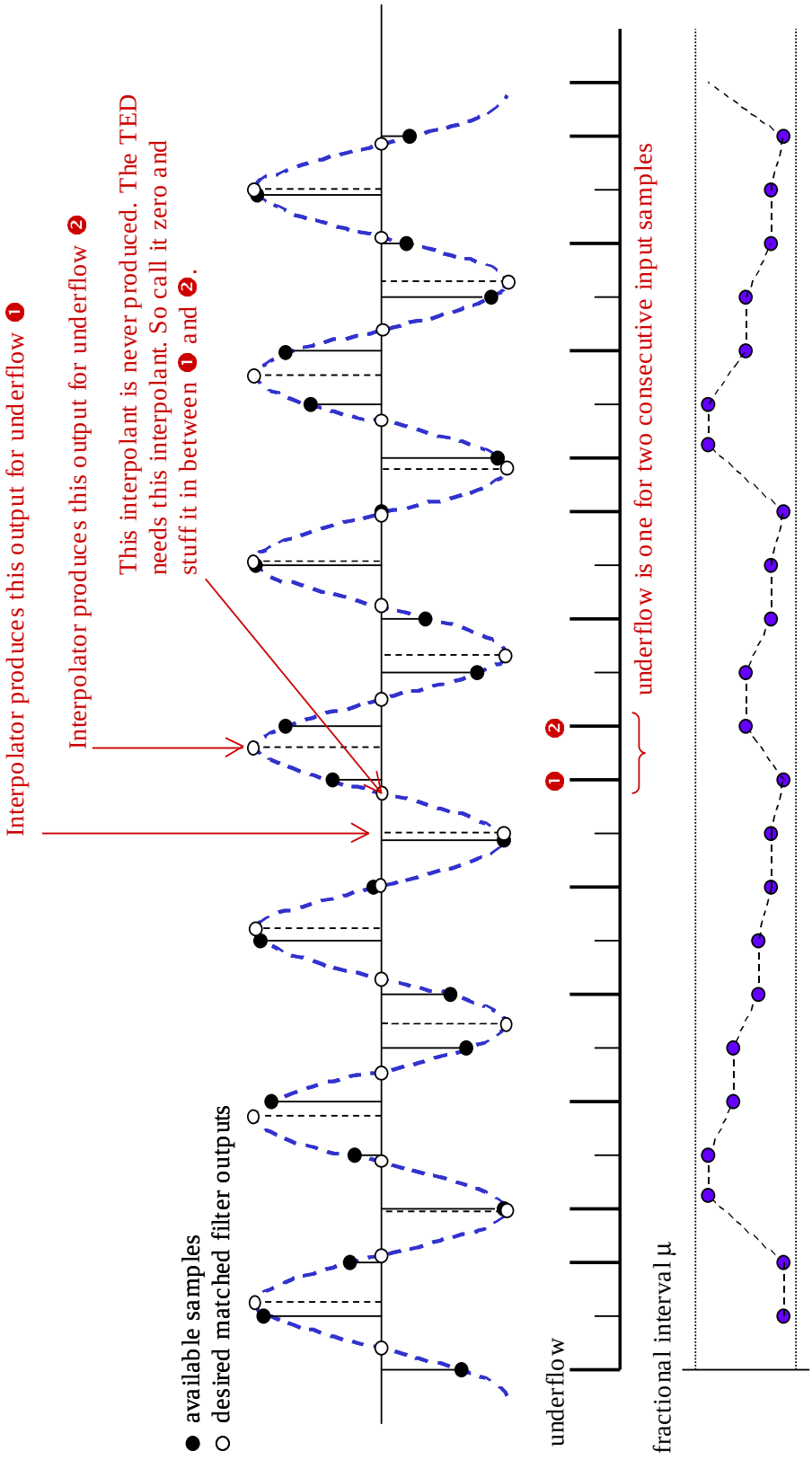


Figure 3.67: An illustration of the relationship between the available matched filter output samples, the desired interpolants, the underflow from the NCO interpolation controller, and the fractional interval for the case where the sample clock frequency is slightly slower than 2 samples/symbol (i.e., $T > T_s/2$).

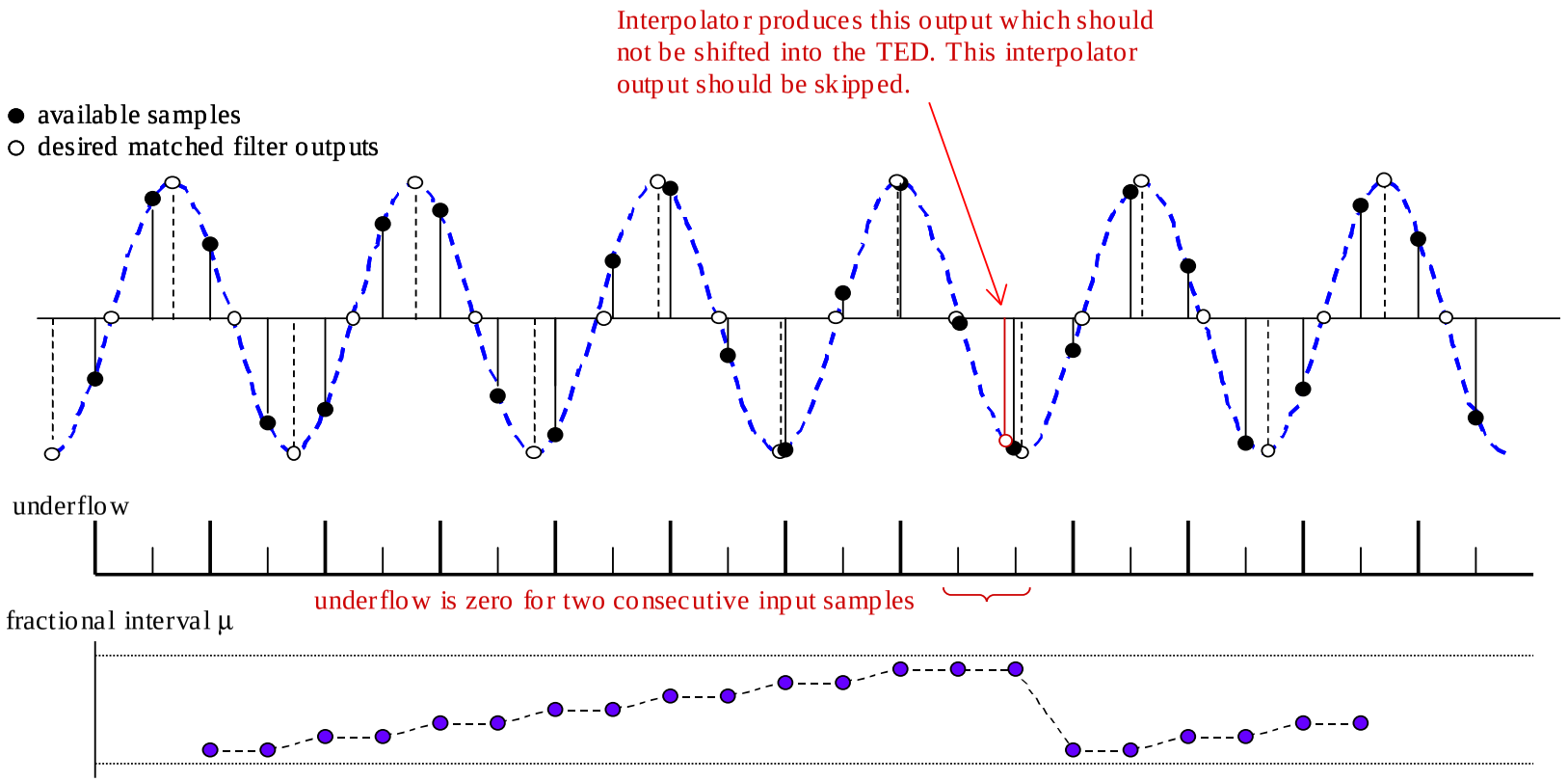


Figure 3.68: An illustration of the relationship between the available matched filter output samples, the desired interpolants, the underflow from the NCO interpolation controller, and the fractional interval for the case where the sample clock frequency is slightly faster than 2 samples/symbol (i.e., $T < T_s/2$).

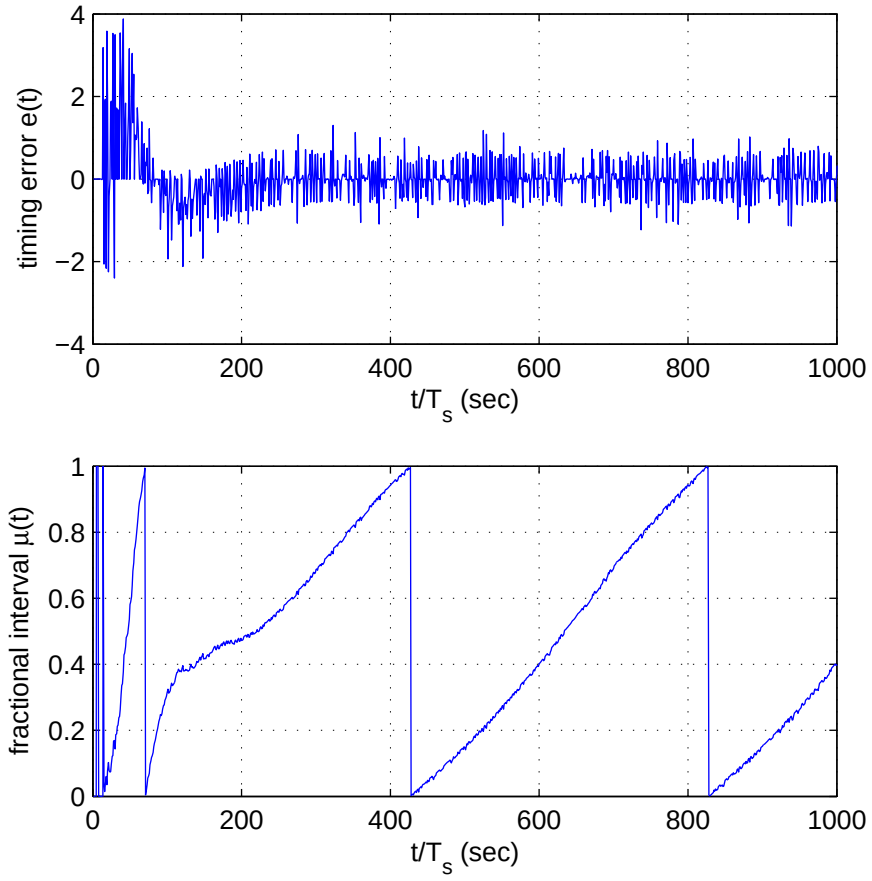


Figure 3.69: Timing error signal and fractional interpolation interval for the symbol timing synchronization system illustrated in Figure 3.65 for the case where the sample clock is slightly faster than 2 samples/symbol.

3.4.4 Discrete-Time Techniques for MQASK

Let the received IF MQASK signal be

$$r(t) = \sum_n a_1 p\left(t - nT_s - \tau\right)\cos(\omega_0 t + \theta) - a_2 p\left(t - nT_s - \tau\right)\sin(\omega_0 t + \theta) + w(t) \quad (3.167)$$

where $p(t)$ is unit energy pulse shape with support on the interval $-L_p T_s \leq tL_p T_s$, T_s is the symbol time, τ is the unknown timing delay to be estimated, and $w(t)$ is a random process representing additive white Gaussian noise. ADC placement is an important system-level consideration that requires some discussion at this point.

There are two locations where the ADC is commonly placed as illustrated in Figure 3.70. Figure 3.70 (a) shows a configuration commonly referred to as "IF sampling." The ADC samples the bandlimited signal $r(t)$ every T_{IF} seconds where the sampling rate satisfies the Nyquist rate condition for the bandpass IF signal. These samples are mixed by quadrature discrete-time sinusoids to produce samples of the baseband inphase and quadrature components $I\left(nT_{\text{IF}}\right)$ and $Q\left(nT_{\text{IF}}\right)$. $I\left(nT_{\text{IF}}\right)$ and $Q\left(nT_{\text{IF}}\right)$ are filtered by the discrete-time matched filters with impulse response $h\left(nT_{\text{IF}}\right) = p\left(-nT_{\text{IF}}\right)$. The desire is produce N_{IF} samples of the inphase and quadrature matched filter outputs during each symbol such that one of the samples on both the inphase and quadrature components are aligned with the maximum average eye opening.

The second commonly used option for ADC placement is shown in Figure 3.70 (b). The bandpass IF signal $r(t)$ is mixed to baseband using continuous-time quadrature sinusoids and lowpass filtered to produce the inphase and quadrature baseband components $I(t)$ and $Q(t)$. $I(t)$ and $Q(t)$ and sampled by a pair of ADCs (or a dual-channel ADC) to produce samples of the inphase and quadrature baseband components $I\left(nT_{\text{BB}}\right)$ and $Q\left(nT_{\text{BB}}\right)$, respectively. $I\left(nT_{\text{BB}}\right)$ and $Q\left(nT_{\text{BB}}\right)$ are filtered by the discrete-time matched filters with impulse response $h\left(nT_{\text{BB}}\right) = p\left(-nT_{\text{BB}}\right)$. As before, the desire is produce N_{BB} samples of the inphase and quadrature matched filter outputs during each symbol such that one of the samples on both the inphase and quadrature components are aligned with the maximum average eye opening.

Which of the two approaches is preferred depends on many factors including the symbol rate and IF frequency (which determine the required sample rate), cost, performance requirements, the availability of good analog IF filters for channel selection and/or adjacent channel rejection, etc. Some generalizations can be made. The two-channel baseband sampling option has the advantage that it often requires a lower sample rate than that required for IF sampling[7] (i.e., $T_{\text{IF}} < T_{\text{BB}}$). This

[7]The reason this is not *always* true is because bandpass sampling can be used for IF sampling. Care must be taken to ensure that the aliased spectra of the IF signal do not overlap. When this condition can be satisfied, it is often the case that the IF sampling rate is the same as the baseband sampling rate.

option is attractive for applications where the symbol rate is one-half to one-quarter the maximum available clock rate. The IF sampling option has the following advantages:

1. Only one ADC is required instead of two (or a single channel ADC instead of a two-channel ADC).

2. The down-conversion from IF is true quadrature conversion. The two-channel baseband sample requires this operation be done with continuous-time processing. A good analog I/Q mixer requires perfectly balanced inphase and quadrature mixers along with a phase shifter to produce the quadrature sinusoids. These requirements can be challenging especially in harsh operating environments.

3. In applications where the IF signal contains closely spaced frequency division multiplexed signals, channel selection can often be realized with better adjacent channel rejection using discrete-time processing. Placing the ADC at IF allows this to be done.

In general, the advantages of IF sampling outweigh the disadvantages of the higher clock rate requirements. For this reason, If sampling is used whenever system constraints allow it.

It is not important which of the two approaches is used for the purposes of describing symbol timing synchronization using discrete-time techniques. In either case, the matched filter inputs are the samples of $I(t)$ and $Q(t)$. These samples are denoted $I(nT)$ and $Q(nT)$, respectively; whether $T = T_s/N_{\text{IF}}$ or $T = T_s/N_{\text{BB}}$ is not important as long as it is known. $I(nT)$ and $Q(nT)$ are of the same form as $r(nT)$ in Section 3.4.3. Timing error detectors operate on both $I(nT)$ and $Q(nT)$ in the same way they operated on $r(nT)$ in Section 3.4.3. The outputs of the two timing error detectors are summed to form the error signal. The error signal is filtered by the loop filter and drives the interpolation control. The general structure for MQASK symbol timing synchronization with IF sampling is illustrated in Figure 3.71.

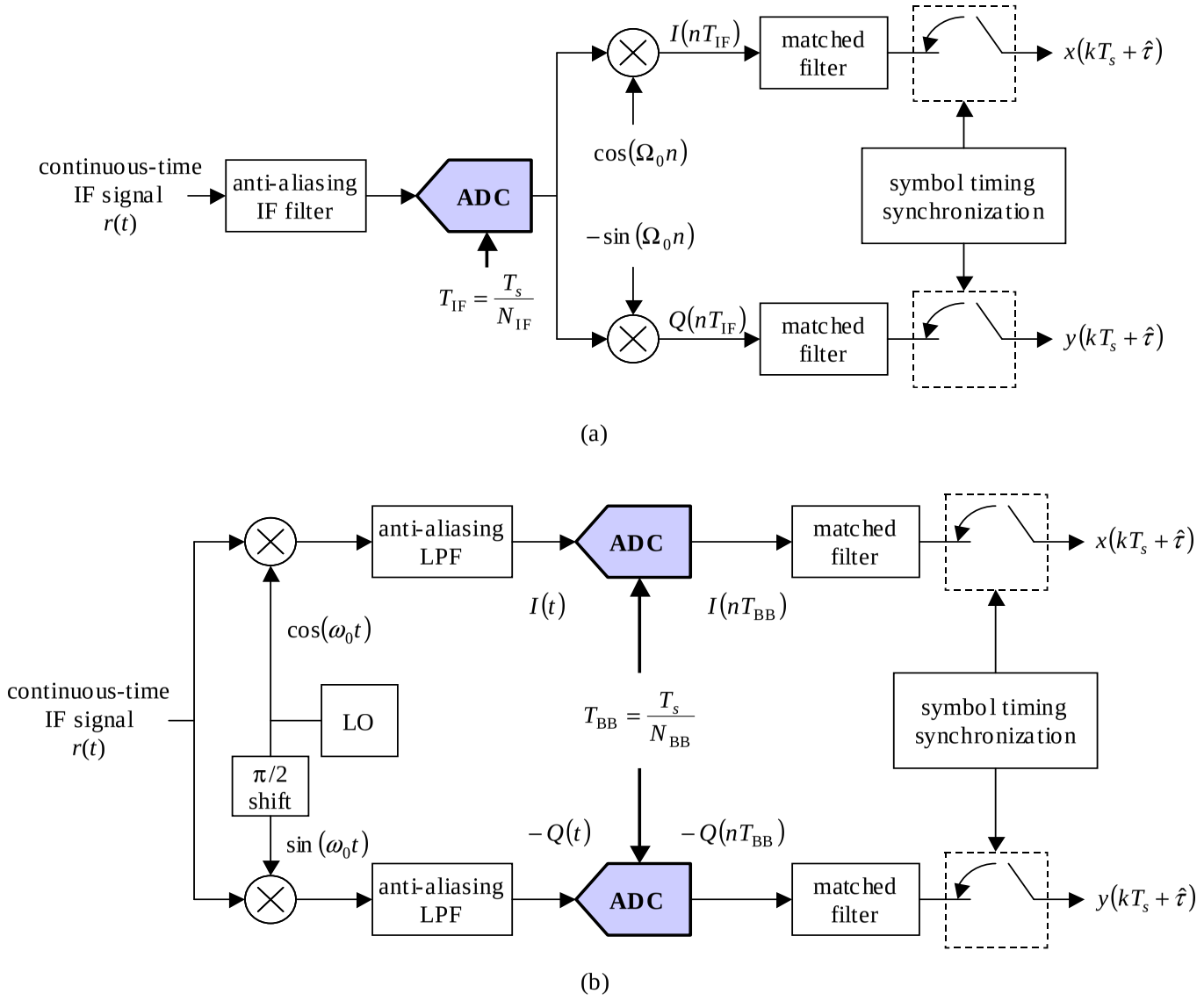


Figure 3.70: Two commonly used options for ADC placement: (a) IF sampling (b) dual-channel baseband sampling.

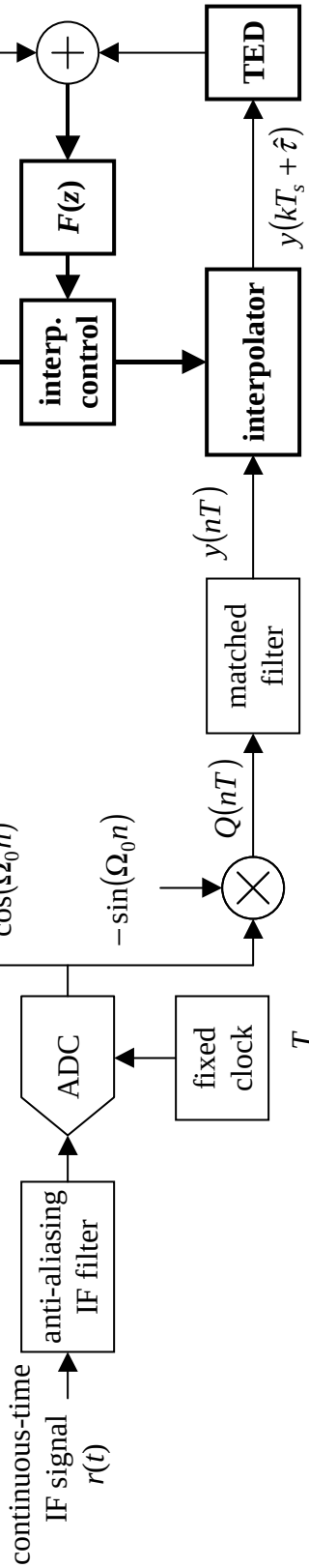


Figure 3.71: General structure for symbol timing synchronization for MQASK using IF sampling.

3.5 Discrete-Time Techniques for Offset QPSK

Assuming IF sampling and perfect phase synchronization, let the discrete-time IF signal be

$$r(nT) = \sum_m a_1(m)p(nT - mT_s - \tau)\cos(\Omega_0 n) - \sum_m a_2(m)p(nT - mT_s - \tau)\sin(\Omega_0 n) \quad (3.168)$$

where $1/T$ is the sample rate, $a_1(m) \in \{-1, +1\}$ and $a_2(m) \in \{-1, +1\}$ are the information symbols, $p(nT)$ is a unit energy pulse shape with support on the interval $-L_pT_s/T < n < L_pT_s/T$, Ω_0 is the IF frequency in radians/sample, and τ is the unknown symbol timing offset. The matched filter outputs may be expressed as

$$x(nT) = \sum_m a_1(m)R_p\left(nT - mT_s - \tau\right) \quad (3.169)$$

$$y(nT) = \sum_m a_2(m)R_p\left(nT - mT_s - T_s/2 - \tau\right) \quad (3.170)$$

where $R_p\left(u\right)$ is the autocorrelation function of the pulse shape given by (3.82).

The relationship between the two eye patterns formed by $x(nT)$ and $y(nT)$ is illustrated in Figure 3.72. The maximum average eye opening on $y(nT)$ is delayed from the maximum average eye opening on $x(nT)$ by $T_s/2$. The inphase matched filter output $x(nT)$ should be sampled at

$$n = k\frac{T_s}{T} + \tau \quad (3.171)$$

while the quadrature matched filter output $y(nT)$ should be sampled at

$$n = k\frac{T_s}{T} + \frac{T_s}{2T} + \tau \quad (3.172)$$

for $k = 0, 1, \ldots$.

Following the same line of reasoning as before, the slope of eye patterns can be used as a timing error signal. Since the eye patterns are delayed $T_s/2$ from each other, this method must be modified. The maximum-likelihood data-aided timing error detector uses the error signal

$$e(k) = a_1(k)\dot{x}(kT_s + \hat{\tau}(k)) + a_2(k)\dot{y}(kT_s + T_s/2 + \hat{\tau}(k)) \quad (3.173)$$

where $\dot{x}(kT_s + \hat{\tau}(k))$ is the time derivative of $x(t)$ evaluated at $t = kT_s + \hat{\tau}(k)$ and $\dot{y}(kT_s + T_s/2 + \hat{\tau}(k))$ is the time derivative of $y(t)$ evaluated at $t = kT_s + T_s/2 + \hat{\tau}(k)$. The slopes of the matched filter outputs at time instants offset by half a symbol period are combined to form the error signal. The decision-directed maximum likelihood timing error detector uses the error signal

$$e(k) = \text{sgn}\left\{x(kT + s + \hat{\tau}(k))\right\}\dot{x}(kT_s + \hat{\tau}(k)) + \text{sgn}\left\{y(kT_s + T_s/2 + \hat{\tau}(k))\right\}\dot{y}(kT_s + T_s/2 + \hat{\tau}(k)). \quad (3.174)$$

The time derivative may be computed using the techniques described in Section 3.4.3 and illustrated in Figure 3.44. The early-late techniques, described in Section 3.4.3 can be used to approximate the derivatives with the appropriate modifications suggested by (3.173) and (3.174). The zero-crossing detector can also be applied to $x(nT)$ and $y(nT)$ with appropriate delays.

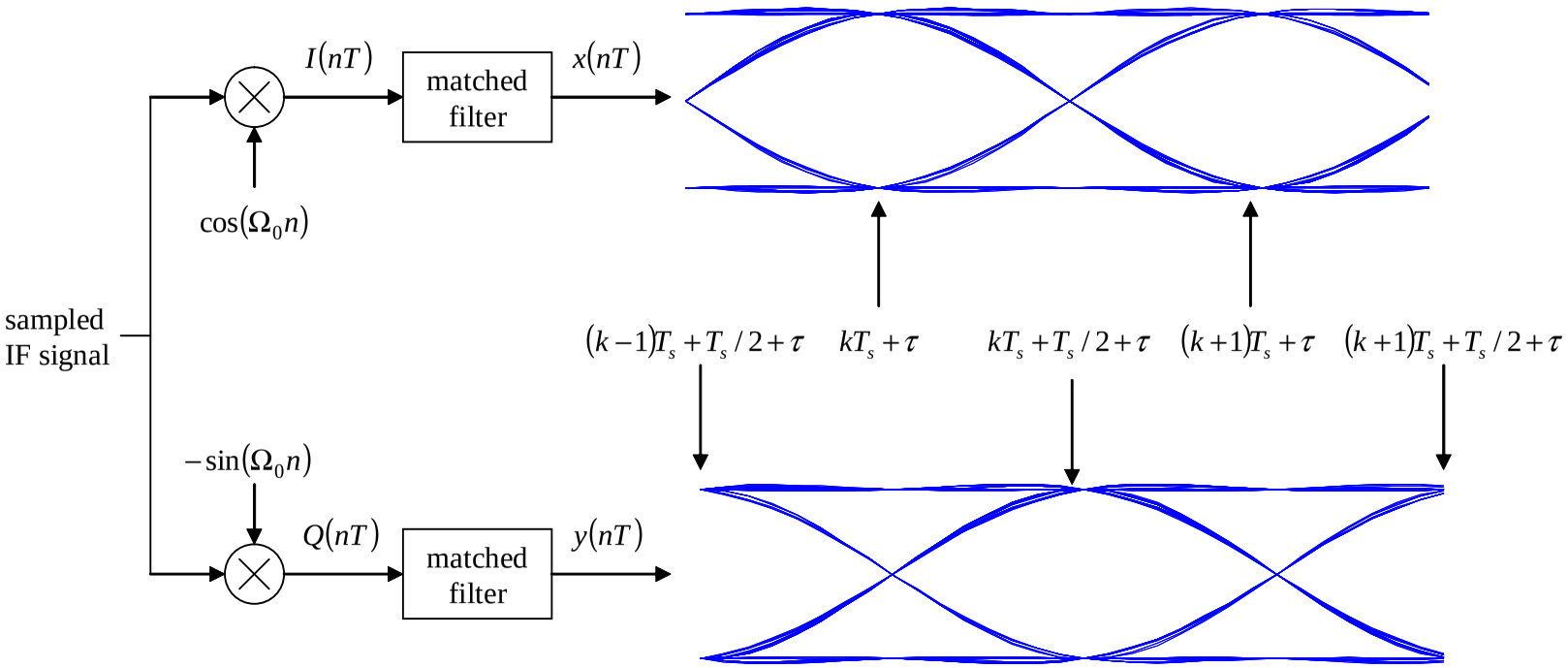


Figure 3.72: Eye diagrams of the inphase and quadrature matched filter outputs for offset QPSK showing the relationship between the maximum average eye openings.

3.6 Maximum Likelihood Estimation

Maximum likelihood estimation uses conditional probabilities as a measure of "how likely" a parameter is given noisy observations. This technique was applied in Chapter **??** to derive the optimum (in the maximum likelihood sense) structure for detectors. The problem was cast as an estimation problem where the information symbols were the unknown quantity. Maximum likelihood estimation can also be applied to synchronization. In this case, the carrier phase offset or timing delay offset (or both) are the unknowns that need to be estimated. The technique is demonstrated for QPSK. Extensions to other 2-dimensional signal sets and other D-dimensional signal sets are straightforward.

3.6.1 Preliminaries

Let the observation interval be $T_0 = L_0 T_s$ seconds and let the received IF signal be

$$r(t) = s(t) + w(t) \tag{3.175}$$

where

$$s(t) = \sum_{k=0}^{L_0} a_1(k)p(t - kT_s - \tau)\cos(\omega_0 t + \theta) - a_2(k)p(t - kT_s - \tau)\sin(\omega_0 t + \theta) \tag{3.176}$$

and $w(t)$ is a zero-mean white Gaussian random process with power spectral density $N_0/2$ W/Hz. For QPSK, $a_1(k) \in \{-1, +1\}$ and $a_2(k) \in \{-1, +1\}$ for $k = 0, 1, \ldots, L_0 - 1$. The IF signal is sampled every T seconds to produce the sequence

$$r(nT) = s(nT) + w(nT); \qquad n = 0, 1, \ldots, NL_0 - 1. \tag{3.177}$$

The sampled signal component may be expressed as

$$s(nT) = \sum_{k=0}^{L_0-1} a_1(k)p(nT - kT_s - \tau)\cos(\Omega_0 n + \theta) - a_2(k)p(nT - kT_s - \tau)\sin(\Omega_0 n + \theta) \tag{3.178}$$

for $n = 0, 1, \ldots, NL_0 - 1$. For convenience, the following vectors are defined

$$\mathbf{r} = \begin{bmatrix} r(0) \\ r(T) \\ \vdots \\ r((NL_0 - 1)T) \end{bmatrix} \quad \mathbf{s} = \begin{bmatrix} s(0) \\ s(T) \\ \vdots \\ s((NL_0 - 1)T) \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w(0) \\ w(T) \\ \vdots \\ w((NL_0 - 1)T) \end{bmatrix}. \tag{3.179}$$

The vector $\mathbf{w}$ is a sequence of independent and identically distributed Gaussian random variables with zero mean and variance

$$\sigma^2 = \frac{N_0}{2T}. \tag{3.180}$$

The probability density function of $\mathbf{w}$ is

$$p(\mathbf{w}) = \frac{1}{(2\pi\sigma^2)^{L_0 N/2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{n=0}^{NL_0-1} w^2(nT)\right\}. \tag{3.181}$$

For notational convenience, define the symbol vector $\mathbf{a}$ as

$$\mathbf{a} = \begin{bmatrix}\mathbf{a}(0) & \mathbf{a}(1) & \cdots & \mathbf{a}(L_0-1)\end{bmatrix}^T \tag{3.182}$$

where

$$\mathbf{a}(k) = \begin{bmatrix} a_1(k) \\ a_2(k) \end{bmatrix}. \tag{3.183}$$

To emphasize the fact that the $\mathbf{s}$ is a function of $\mathbf{a}$, θ, and τ, $\mathbf{s}$ will be expressed as $\mathbf{s}(\mathbf{a}, \theta, \tau)$ and samples of the signal component $s(nT)$ will be expressed as $s(nT; \mathbf{a}, \theta, \tau)$.

Carrier phase synchronization and symbol timing synchronization can be thought of as estimation problems. The goal is to estimate the parameters θ and τ from the samples $r(nT) = s(nT; \mathbf{a}, \theta, \tau) + w(nT)$. The maximum likelihood estimate is the one that maximizes the logarithm of the conditional probability $p(\mathbf{r}|\mathbf{a}, \theta, \tau)$. Using the probability density function of $\mathbf{w}$ given by (3.181), the conditional probability $p(\mathbf{r}|\mathbf{a}, \theta, \tau)$ is

$$p(\mathbf{r}|\mathbf{a}, \theta, \tau) = \frac{1}{(2\pi\sigma^2)^{L_0 N/2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{n=0}^{NL_0-1} |r(nT) - s(nT; \mathbf{a}, \theta, \tau)|^2\right\}. \tag{3.184}$$

The log-likelihood function $\Lambda(\mathbf{a}, \theta, \tau)$ is the logarithm of (3.184):

$$\Lambda(\mathbf{a}, \theta, \tau) = -\frac{L_0 N}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{n=0}^{NL_0-1} |r(nT) - s(nT; \mathbf{a}, \theta, \tau)|^2 \tag{3.185}$$

Later it will be convenient to express the cross product sum as

$$\begin{aligned}\sum_{n=0}^{NL_0-1} r(nT)s(nT; \mathbf{a}, \theta, \tau) = &\sum_{k=0}^{L_0-1} a_1(k) \sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT - kT_s - \tau)\cos(\Omega_0 n + \theta) \\ &- \sum_{k=0}^{L_0-1} a_2(k) \sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT - kT_s - \tau)\sin(\Omega_0 n + \theta).\end{aligned} \tag{3.186}$$

Two approaches will be taken to obtain the maximum likelihood estimators for θ and τ. The first approach assumes $\mathbf{a}$ is known[8]. In this case the estimators for θ and τ are functions of the data symbols.

The second approach does not assume $\mathbf{a}$ is known. In this case, the dependence on $\mathbf{a}$ is removed by assuming the symbol sequence $\mathbf{a}$ is random and using the total probability theorem to obtain the average conditional probability density function $p(\mathbf{r}|\theta,\tau)$. The maximum likelihood estimate maximizes the logarithm of $p(\mathbf{r}|\theta,\tau)$.

The average conditional probability density function $p(\mathbf{r}|\theta,\tau)$ is related to the conditional probability density function $p(\mathbf{r}|\mathbf{a},\theta,\tau)$ by the total probability theorem:

$$p(\mathbf{r}|\theta,\tau) = \int p(\mathbf{r}|\mathbf{a},\theta,\tau)p(\mathbf{a})d\mathbf{a} \tag{3.187}$$

where $p(\mathbf{a})$ is the probability density function of the symbol sequence $\mathbf{a}$. The most commonly used probability density function for the data sequence assumes the symbols are independent and equally likely. Independence implies

$$p(\mathbf{a}) = \prod_{k=0}^{L_0-1} p(\mathbf{a}(k)) \tag{3.188}$$

while equally likely implies

$$\begin{aligned} p(\mathbf{a}(k)) = \frac{1}{4}\delta(a_1(k)-1)\delta(a_2(k)-1) + \frac{1}{4}\delta(a_1(k)-1)\delta(a_2(k)+1) \\ + \frac{1}{4}\delta(a_1(k)+1)\delta(a_2(k)-1) + \frac{1}{4}\delta(a_1(k)+1)\delta(a_2(k)+1). \end{aligned} \tag{3.189}$$

Thus,

$$p(\mathbf{r}|\theta,\tau) = \int p(\mathbf{r}|\mathbf{a},\theta,\tau)p(\mathbf{a})d\mathbf{a} \tag{3.190}$$

$$= \prod_{k=0}^{L_0-1} \int p(\mathbf{r}|\mathbf{a}(k),\theta,\tau)p(\mathbf{a}(k))d\mathbf{a}(k) \tag{3.191}$$

$$\begin{aligned} = \prod_{k=0}^{L_0-1} \Big\{ \frac{1}{4}p(\mathbf{r}|\mathbf{a}(k)=[1,1],\theta,\tau) + \frac{1}{4}p(\mathbf{r}|\mathbf{a}(k)=[1,-1],\theta,\tau) \\ + \frac{1}{4}p(\mathbf{r}|\mathbf{a}(k)=[-1,1],\theta,\tau) + \frac{1}{4}p(\mathbf{r}|\mathbf{a}(k)=[-1,-1],\theta,\tau) \Big\} \end{aligned} \tag{3.192}$$

[8]For packetized burst mode communication systems with a known preamble or header, the L_0 data symbols are known and should be used for synchronization.

By writing (3.184) as

$$p(\mathbf{r}|\mathbf{a},\theta,\tau)=\prod_{k=0}^{L_0-1}\frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}|r(nT)-s(nT;\mathbf{a},\theta,\tau)|^2\right\} \tag{3.193}$$

and using the substitution

$$s(nT;\mathbf{a}(k),\theta,\tau)=a_1(k)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)-a_2(k)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta) \tag{3.194}$$

each term in (3.192) may be expressed as

$$\begin{aligned}p(\mathbf{r}|\mathbf{a}(k)=[1,1],\theta,\tau)=\prod_{k=0}^{L_0-1}\frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}|r(nT)|^2+|p(nT-kT_s-\tau)|^2\right\}\\ \times\exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)\right\}\\ \times\exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta)\right\}\end{aligned} \tag{3.195}$$

$$\begin{aligned}p(\mathbf{r}|\mathbf{a}(k)=[1,-1],\theta,\tau)=\prod_{k=0}^{L_0-1}\frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}|r(nT)|^2+|p(nT-kT_s-\tau)|^2\right\}\\ \times\exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)\right\}\\ \times\exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta)\right\}\end{aligned} \tag{3.196}$$

$$\begin{aligned}p(\mathbf{r}|\mathbf{a}(k)=[-1,1],\theta,\tau)=\prod_{k=0}^{L_0-1}\frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}|r(nT)|^2+|p(nT-kT_s-\tau)|^2\right\}\\ \times\exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)\right\}\\ \times\exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta)\right\}\end{aligned} \tag{3.197}$$

$$p(\mathbf{r}|\mathbf{a}(k)=[-1,-1],\theta,\tau)=\prod_{k=0}^{L_0-1}\frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}|r(nT)|^2+|p(nT-kT_s-\tau)|^2\right\}$$
$$\times\exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)\right\}$$
$$\times\exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta)\right\} \quad (3.198)$$

Substituting (3.195) – (3.198) into (3.192) and collecting similar terms produces

$$p(\mathbf{r}|\theta,\tau)=\frac{1}{4}\prod_{k=0}^{L_0-1}\frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}|r(nT)|^2+|p(nT-kT_s-\tau)|^2\right\}$$
$$\times\left(\exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)\right\}\right.$$
$$\left.+\exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)\right\}\right)$$
$$\times\left(\exp\left\{\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta)\right\}\right.$$
$$\left.+\exp\left\{-\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta)\right\}\right) \quad (3.199)$$

Applying the identity

$$\frac{e^x+e^{-x}}{2}=\cosh(x) \quad (3.200)$$

to (3.199) produces

$$p(\mathbf{r}|\theta,\tau)=\prod_{k=0}^{L_0-1}\frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}|r(nT)|^2+|p(nT-kT_s-\tau)|^2\right\}$$
$$\times\cosh\left(\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)\right)$$
$$\times\cosh\left(\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)}r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta)\right) \quad (3.201)$$

The average log-likelihood function is

$$\overline{\Lambda}(\theta,\tau) = -\frac{NL_0}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{k=0}^{L_0-1}\sum_{N(k-L)}^{N(k+L)} |r(nT)|^2 + |p(nT-kT_s-\tau)|^2 + \sum_{k=0}^{L_0-1}\ln\cosh\left(\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)} r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)\right) + \sum_{k=0}^{L_0-1}\ln\cosh\left(\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)} r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta)\right) \quad (3.202)$$

3.6.2 Carrier Phase Estimation

Known Symbol Sequence and Known Timing

For the case where the data symbols are known, the maximum likelihood estimate $\hat{\theta}$ is the value of θ that maximizes the log-likelihood function $\Lambda(\mathbf{a},\theta,\tau)$ given by (3.185). This estimate is the value of θ that forces the partial derivative of $\Lambda(\mathbf{a},\theta,\tau)$ with respect to θ to be zero. The partial derivative of $\Lambda(\mathbf{a},\theta,\tau)$ is

$$\frac{\partial}{\partial\theta}\Lambda(\mathbf{a},\theta,\tau) = -\frac{1}{2\sigma^2}\frac{\partial}{\partial\theta}\sum_{n=0}^{NL_0-1}|r(nT)-s(nT;\mathbf{a},\theta,\tau)|^2 \quad (3.203)$$

$$= -\frac{1}{2\sigma^2}\frac{\partial}{\partial\theta}\sum_{n=0}^{NL_0-1}\left[|r(nT)|^2 - 2r(nT)s(nT;\mathbf{a},\theta,\tau) + |s(nT;\mathbf{a},\theta,\tau)|^2\right]. \quad (3.204)$$

The partial derivatives of the first and third terms are zero since the energy in the received signal and the energy in a QPSK waveform are the same for all phase rotations. All that remains is the middle term. Substituting (3.178) for $s(nT;\mathbf{a},\theta,\tau)$, interchanging the order of summations, and computing the derivative yields

$$\frac{\partial}{\partial\theta}\Lambda(\mathbf{a},\theta,\tau) = -\frac{1}{\sigma^2}\sum_{k=0}^{L_0-1}a_1(k)\sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta) - \sum_{k=0}^{L_0-1}a_2(k)\sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta). \quad (3.205)$$

Recall that the k-th matched filter outputs for the inphase and quadrature components using a phase coherent IF downconversion (see Figure 3.8) are

$$x(kT_s) = \sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT - kT_s - \tau)\cos(\Omega_0 n + \hat{\theta}) \tag{3.206}$$

$$y(kT_s) = -\sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT - kT_s - \tau)\sin(\Omega_0 n + \hat{\theta}). \tag{3.207}$$

Note that inner sum in the first term of (3.205) is the quadrature matched filter output and the inner sum in the second term of (3.205) is the inphase matched filter output. Using the notation $x(kT_s;\theta)$ and $y(kT_s;\theta)$ to emphasize that the matched filter outputs are a function of the phase estimate, (3.205) can be expressed in the more compact form

$$\frac{\partial}{\partial\theta}\Lambda(\mathbf{a},\theta,\tau) = \sum_{k=0}^{L_0-1} a_1(k)y(kT_s;\theta) - a_2(k)x(kT_s;\theta). \tag{3.208}$$

The maximum likelihood estimate $\hat{\theta}$ satisfies

$$0 = \sum_{k=0}^{L_0-1} a_1(k)y(kT_s;\hat{\theta}) - a_2(k)x(kT_s;\hat{\theta}). \tag{3.209}$$

This equation may be solved iteratively. A value for θ is chosen and used to compute the right-hand side of (3.209). The estimate for θ is increased (if the computation is negative) or decreased (if the computation is positive) until θ satisfies (3.209). A block diagram of a system which finds the maximum likelihood estimate iteratively is illustrated in Figure 3.73. Note that it is a PLL structure that uses the right-hand side of (3.209) as the error signal. The summation block plays the role of the loop filter (recall that the loop filter contains an integrator). Compare this block diagram with the QPSK carrier phase PLL shown in Figure 3.17. If the symbol decisions in Figure 3.17 are replaced by the true symbols in the error detector, then the two systems are equivalent.

Returning to (3.205) and using the identities

$$\cos(A+B) = \cos A\cos A - \sin A\sin B$$
$$\sin(A+B) = \sin A\cos B + \cos A\sin B,$$

(3.205) may be expressed as

$$\begin{aligned}\frac{\partial}{\partial\theta}\Lambda(\mathbf{a},\theta,\tau) = &-\sum_{k=0}^{L_0-1} a_1(k) \sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n)\sin\theta \\ &-\sum_{k=0}^{L_0-1} a_1(k) \sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n)\sin\theta \\ &-\sum_{k=0}^{L_0-1} a_2(k) \sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n)\cos\theta \\ &+\sum_{k=0}^{L_0-1} a_2(k) \sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n)\sin\theta. \quad (3.210)\end{aligned}$$

Recall that the k-th matched filter outputs for the inphase and quadrature components using non-coherent IF conversion (see Figure 3.7) are

$$x(kT_s) = \sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n) \quad (3.211)$$

$$y(kT_s) = -\sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n). \quad (3.212)$$

Using these definitions, (3.210) may be expressed as

$$\begin{aligned}&\frac{\partial}{\partial\theta}\Lambda(\mathbf{a},\theta,\tau) = \\ &\sum_{k=0}^{L_0-1} a_1(k)\left[y(kT_s)\cos\theta - x(kT_s)\sin\theta\right] - \sum_{k=0}^{L_0-1} a_2(k)\left[x(kT_s)\cos\theta + y(kT_s)\sin\theta\right]. \quad (3.213)\end{aligned}$$

The terms in the square brackets are the equations for the rotation of the point $(x(kT_s), y(kT_s))$ by an angle $-\theta$. Following the notation introduced in Section 3.3, let $(x'(kT_s;\theta), y'(kT_s;\theta))$ represent the rotated point (θ is included to emphasize the dependence on θ) so that

$$\begin{bmatrix} x'(kT_s;\theta) \\ y'(kT_s;\theta)\end{bmatrix} = \begin{bmatrix}\cos\theta & \sin\theta \\ -\sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix} x(kT_s) \\ y(kT_s)\end{bmatrix}. \quad (3.214)$$

Thus (3.213) may be expressed as

$$\frac{\partial}{\partial\theta}\Lambda(\mathbf{a},\theta,\tau) = \sum_{k=0}^{L_0-1} a_1(k)y'(kT_s;\theta) - a_2(k)x'(kT_s;\theta) \quad (3.215)$$

and an alternate expression for the maximum likelihood phase estimate is

$$0 = \sum_{k=0}^{L_0-1} a_1(k)y'(kT_s;\hat{\theta}) - a_2(k)x'(kT_s;\hat{\theta}). \tag{3.216}$$

Note that two forms for the maximum likelihood estimator (3.209) and (3.216) are identical. The difference is where carrier phase compensation occurs. A block diagram illustrating the iterative solution to (3.216) is shown in Figure 3.74. This is a PLL structure where the right-hand side of (3.216) is the error signal. The solution shown in Figure 3.74 is almost identical to that shown in Figure 3.13.

Setting (3.213) to zero and solving for θ results in a closed form solution for the maximum likelihood phase estimate. Grouping the terms which have the cosine in common and grouping the terms that have the sine in common and solving produces

$$\frac{\sin\hat{\theta}}{\cos\hat{\theta}} = \frac{\displaystyle\sum_{k=0}^{L_0-1} a_1(k)y(kT_s) - a_2(k)x(kT_s)}{\displaystyle\sum_{k=0}^{L_0-1} a_1(k)x(kT_s) + a_2(k)y(kT_s)} \tag{3.217}$$

from which the maximum likelihood phase estimate is

$$\hat{\theta} = \tan^{-1}\left\{\frac{\displaystyle\sum_{k=0}^{L_0-1} a_1(k)y(kT_s) - a_2(k)x(kT_s)}{\displaystyle\sum_{k=0}^{L_0-1} a_1(k)x(kT_s) + a_2(k)y(kT_s)}\right\}. \tag{3.218}$$

This solution is useful for packetized communications links where the carrier phase offset θ will remain constant over the duration of the data packet. Such detectors typically use block processing in place of iterative processing.

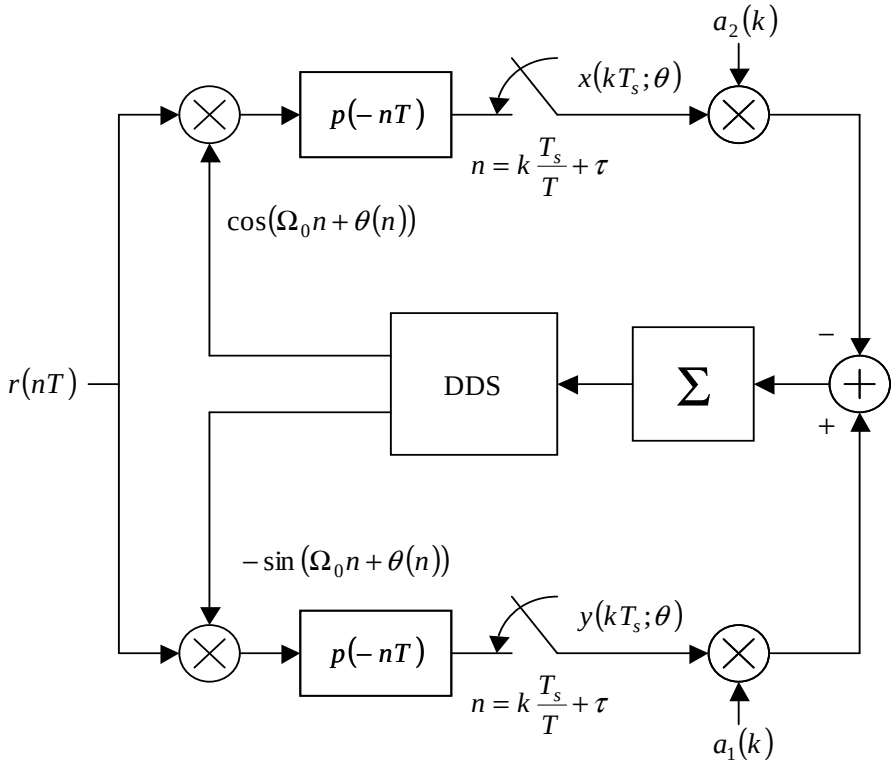


Figure 3.73: Block diagram of the maximum-likelihood QPSK phase estimator based on the form (3.209).

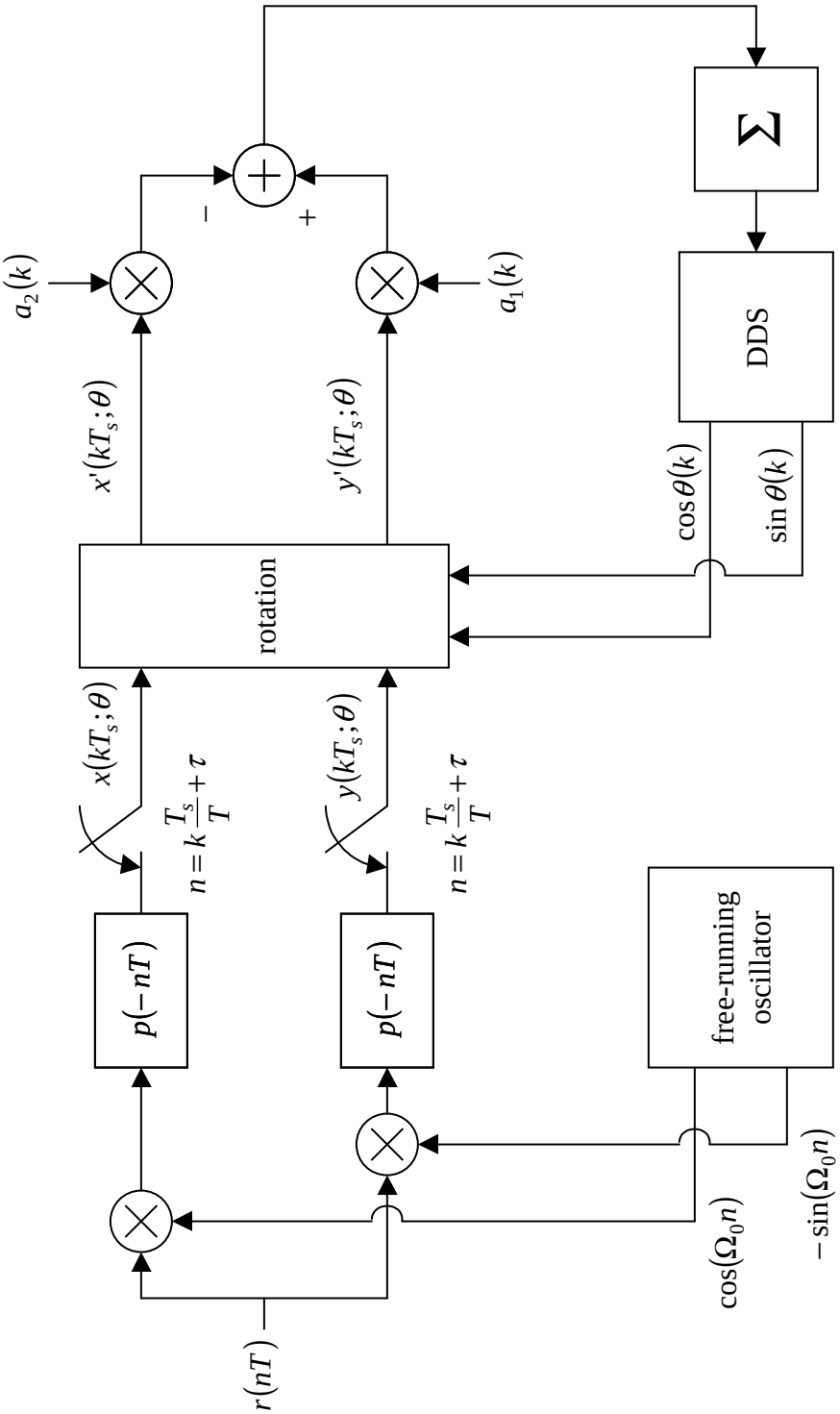


Figure 3.74: Block diagram of the maximum-likelihood QPSK phase estimator based on the form (3.216).

Unknown Symbol Sequence and Known Timing

When the timing is known, but the symbol sequence is not known, it is possible to use the symbol decisions $\hat{a}_1(k)$ and $\hat{a}_2(k)$ in place of the true symbols. Using the results from the preceding section, two forms for the decision-directed maximum-likelihood phase estimator result. The first results from replacing $a_1(k)$ and $a_2(k)$ in (3.209) with the decisions $\hat{a}_1(k)$ and $\hat{a}_2(k)$:

$$0 = \sum_{k=0}^{L_0-1} \hat{a}_1(k) y(kT_s;\hat{\theta}) - \hat{a}_2(k) x(kT_s;\hat{\theta}). \tag{3.219}$$

The second results from replacing $a_1(k)$ and $a_2(k)$ in (3.216) with the decisions $\hat{a}_1(k)$ and $\hat{a}_2(k)$:

$$0 = \sum_{k=0}^{L_0-1} \hat{a}_1(k) y'(kT_s;\hat{\theta}) - \hat{a}_2(k) x'(kT_s;\hat{\theta}). \tag{3.220}$$

Block diagrams for these two forms of the decision-directed maximum-likelihood estimator are identical to those for the two forms of the data-aided maximum likelihood estimator illustrated in Figures 3.73 and 3.74 except the symbol decisions are used in place of the true data symbols. Note that block diagrams for these two forms of the decision-directed maximum-likelihood estimator are essentially similar to those for the decision-directed QPSK carrier phase PLLs shown in Figures 3.13 (with the switch in the upper position) and 3.17, respectively.

Unknown Symbol Sequence and Unknown Timing

When both the symbol timing and the symbol sequence are unknown, the maximum likelihood phase estimate is the one that maximizes the average log-likelihood function $\overline{\Lambda}(\theta,\tau)$ given by (3.202). The partial derivative of $\overline{\Lambda}(\theta,\tau)$ is

$$\begin{aligned}\frac{\partial}{\partial\theta}\overline{\Lambda}(\theta,\tau) = &- \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)} r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)\right)\\ &\times\left[\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)} r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta)\right]\\ &+ \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)} r(nT)p(nT-kT_s-\tau)\sin(\Omega_0 n+\theta)\right)\\ &\times\left[\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)} r(nT)p(nT-kT_s-\tau)\cos(\Omega_0 n+\theta)\right]\end{aligned} \tag{3.221}$$

Using the relationships (3.206) and (3.207) for the inphase and quadrature matched filter outputs, respectively, (3.221) may be expressed in the more compact form

$$\frac{\partial}{\partial\theta}\overline{\Lambda}(\theta,\tau) = \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2}x(kT_s;\theta)\right)\frac{1}{\sigma^2}y(kT_s;\theta) - \tanh\left(\frac{1}{\sigma^2}y(kT_s;\theta)\right)\frac{1}{\sigma^2}x(kT_s;\theta). \quad (3.222)$$

The maximum-likelihood phase estimate is the value of θ that forces (3.222) to zero:

$$0 = \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2}x(kT_s;\hat{\theta})\right)\frac{1}{\sigma^2}y(kT_s;\hat{\theta}) - \tanh\left(\frac{1}{\sigma^2}y(kT_s;\hat{\theta})\right)\frac{1}{\sigma^2}x(kT_s;\hat{\theta}). \quad (3.223)$$

A block diagram outlining an iterative approach to finding $\hat{\theta}$ based on (3.223) is shown in Figure 3.75. This is a PLL structure where the right-hand side of (3.223) is the error signal. This complexity of this structure is often reduced by replacing the hyperbolic tangent with an approximation. The plot of $\tanh(X)$ vs. X shown in Figure 3.76 shows that the hyperbolic tangent is well approximated by

$$\tanh(X) \approx \begin{cases} X & |X| < 0.3 \\ \text{sgn}\{X\} & |X| > 3 \end{cases} \quad (3.224)$$

Thus, the form of the approximation is determined by the magnitude of the argument. Equation (3.223) shows that the magnitude of the argument is proportional to the reciprocal of the noise variance σ^2. The magnitude of σ^2 relative to the magnitudes of $a_1(k)$ and $a_2(k)$ is determined by the signal to noise ratio. For small signal to noise ratios, the hyperbolic tangent block can be eliminated (i.e., replaced by a wire). For large signal to noise ratios, the hyperbolic tangent block can be replaced by a sgn $\{X\}$ block. (Compare the alteration of the block diagram in Figure 3.75 using this approximation with the Costas Loop shown in Figure 3.28.)

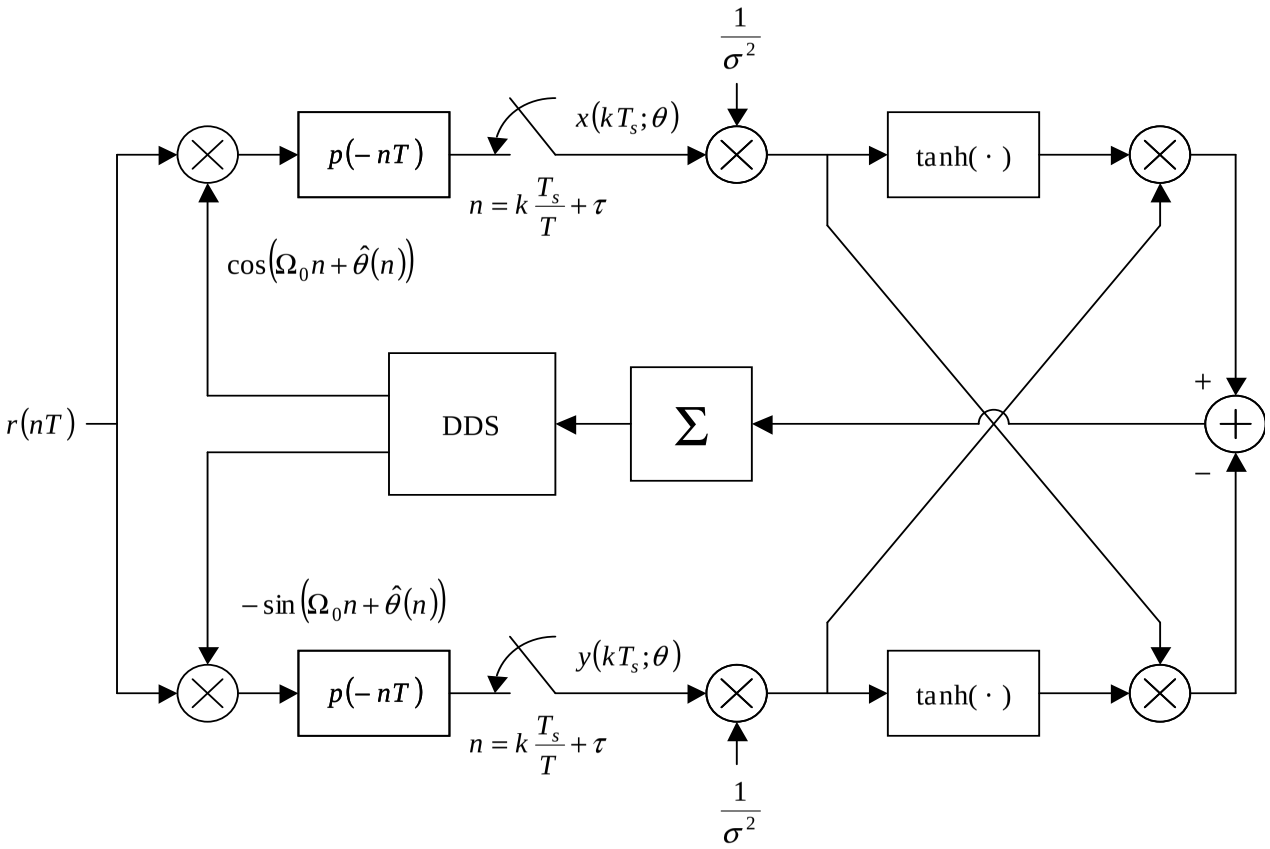


Figure 3.75: Block diagram of the maximum-likelihood QPSK phase estimator based on the form (3.223).

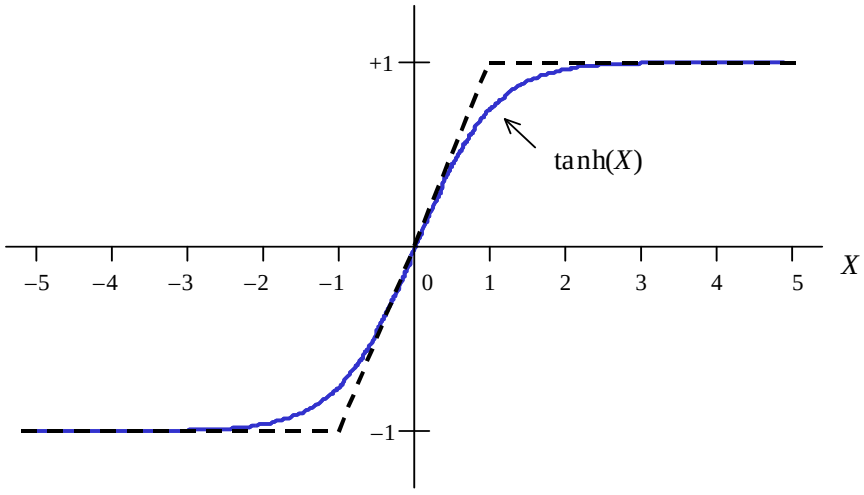


Figure 3.76: Plot of $\tanh(X)$ vs. X illustrating the accuracy of the approximation (3.224).

3.6.3 Symbol Timing Estimation

Known Symbol Sequence and Known Carrier Phase

For the case of known symbols, the maximum likelihood timing estimate is the value of τ that maximizes the log-likelihood function $\Lambda(\mathbf{a}, \theta, \tau)$ given by (3.185). The partial derivative of $\Lambda(\mathbf{a}, \theta, \tau)$ is

$$\frac{\partial}{\partial \tau}\Lambda(\mathbf{a}, \theta, \tau) = -\frac{1}{2\sigma^2}\frac{\partial}{\partial \tau}\sum_{n=0}^{NL_0-1} |r(nT) - s(nT; \mathbf{a}, \theta, \tau)|^2 \tag{3.225}$$

$$= -\frac{1}{2\sigma^2}\frac{\partial}{\partial \tau}\sum_{n=0}^{NL_0-1} \left[|r(nT)|^2 - 2r(nT)s(nT; \mathbf{a}, \theta, \tau) + |s(nT; \mathbf{a}, \theta, \tau)|^2\right]. \tag{3.226}$$

The partial derivative of the first term is zero since the energy in the received signal does not depend on the timing offset. The partial derivative of the third term is approximately zero as there is a weak dependence on τ. For QPSK, this approximation is quite good and shall be carried through with the remainder of this development. As was the case with carrier phase estimation, all that remains is the middle term. Substituting (3.178) for $s(nT; \mathbf{a}, \theta, \tau)$, interchanging the order of summations produces

$$\frac{\partial}{\partial \tau}\Lambda(\mathbf{a}, \theta, \tau) = \frac{1}{\sigma^2}\frac{\partial}{\partial \tau}\sum_{k=0}^{L_0-1} a_1(k) \sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT - kT_s - \tau)\cos(\Omega_0 n + \theta)$$

$$- \frac{1}{\sigma^2}\frac{\partial}{\partial \tau}\sum_{k=0}^{L_0-1} a_1(k) \sum_{n=(k-L)N}^{(k+L)N} r(nT)p(nT - kT_s - \tau)\sin(\Omega_0 n + \theta). \tag{3.227}$$

Recognizing the inner summations as matched filter outputs and using the identities (3.206) and (3.207), (3.227) may be expressed as

$$\frac{\partial}{\partial \tau}\Lambda(\mathbf{a}, \theta, \tau) = \frac{1}{\sigma^2}\frac{\partial}{\partial \tau}\sum_{k=0}^{L_0-1} a_1(k)x(kT_s + \tau) - a_2(k)y(kT_s + \tau) \tag{3.228}$$

$$= \frac{1}{\sigma^2}\sum_{k=0}^{L_0-1} a_1(k)\dot{x}(kT_s + \tau) - a_2(k)\dot{y}(kT_s + \tau) \tag{3.229}$$

where $\dot{x}(kT_s + \tau)$ and $\dot{y}(kT_s + \tau)$ are samples of the time derivatives of the inphase and quadrature matched filter outputs, respectively. These time derivatives may be computed from samples of the matched filter inputs using a filter whose impulse response consists of samples of the time derivative of the pulse shape as illustrated in Figure 3.44 in Section 3.4.3.

The maximum likelihood timing estimate $\hat{\tau}$ is the value of τ that forces (3.229) to zero:

$$0 = \sum_{k=0}^{L_0-1} a_1(k)\dot{x}(kT_s + \hat{\tau}) - a_2(k)\dot{y}(kT_s + \hat{\tau}). \tag{3.230}$$

Unlike the maximum likelihood carrier phase estimate, there is no closed form solution for $\hat{\tau}$. A block diagram illustrating an iterative method for finding $\hat{\tau}$ is shown in Figure 3.77. The solution is a PLL structure where the right-hand side of (3.230) is the error signal.

Unknown Symbol Sequence and Known Carrier Phase

When the carrier phase θ is known and the symbol sequence is unknown, the symbol decision $\hat{a}_1(k)$ and $\hat{a}_2(k)$ may be used in place of the true data symbols. Applying this concept to the data-aided maximum likelihood estimate (3.230) results in the condition for the decision-directed maximum likelihood timing estimate

$$0 = \sum_{k=0}^{L_0-1} \hat{a}_1(k)\dot{x}(kT_s + \hat{\tau}) - \hat{a}_2(k)\dot{y}(kT_s + \hat{\tau}). \tag{3.231}$$

The block diagram illustrating an iterative method for finding $\hat{\tau}$ is identical to the block diagram shown in Figure 3.77 where the symbol decisions replace the true data symbols.

Unknown Symbol Sequence and Unknown Carrier Phase

For the case of unknown data symbols, the maximum likelihood timing estimate is the value of τ that maximizes the average log-likelihood function $\overline{\Lambda}(\theta, \tau)$ given by (3.202). The partial derivative of $\overline{\Lambda}(\theta, \tau)$ is

$$\begin{aligned}
\frac{\partial}{\partial \tau}\overline{\Lambda}(\theta,\tau) = &\sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)} r(nT)p(nT - kT_s - \tau)\cos(\Omega_0 n + \theta)\right) \\
&\times \frac{\partial}{\partial \tau}\left[\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)} r(nT)p(nT - kT_s - \tau)\cos(\Omega_0 n + \theta)\right] \\
&+ \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)} r(nT)p(nT - kT_s - \tau)\sin(\Omega_0 n + \theta)\right) \\
&\times \frac{\partial}{\partial \tau}\left[\frac{1}{\sigma^2}\sum_{n=N(k-L)}^{N(k+L)} r(nT)p(nT - kT_s - \tau)\sin(\Omega_0 n + \theta)\right]
\end{aligned} \tag{3.232}$$

Using the relationships (3.206) and (3.207), (3.232) can be expressed in the more compact form

$$\frac{\partial}{\partial\tau}\overline{\Lambda}(\theta,\tau) = \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2}x(kT_s+\tau)\right)\frac{\partial}{\partial\tau}\left[\frac{1}{\sigma^2}x(kT_s+\tau)\right] + \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2}y(kT_s+\tau)\right)\frac{\partial}{\partial\tau}\left[\frac{1}{\sigma^2}y(kT_s+\tau)\right]. \quad (3.233)$$

Denoting the time derivatives of the inphase and quadrature matched filter outputs by $\dot{x}(kT_s+\tau)$ and $\dot{y}(kT_s+\tau)$, respectively, the maximum-likelihood timing estimate $\hat{\tau}$ satisfies

$$0 = \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2}x(kT_s+\hat{\tau})\right)\frac{1}{\sigma^2}\dot{x}(kT_s+\hat{\tau}) + \sum_{k=0}^{L_0-1} \tanh\left(\frac{1}{\sigma^2}y(kT_s+\hat{\tau})\right)\frac{1}{\sigma^2}\dot{y}(kT_s+\hat{\tau}). \quad (3.234)$$

A block diagram outlining an iterative method for finding $\hat{\tau}$ is shown in Figure 3.78. The basic structure is that of a PLL that uses the right-hand side of (3.234) as the error signal. Low signal-to-noise ratio and large signal-to-noise ratio approximations for the hyperbolic tangent based on (3.224) may be used to reduce the complexity of the system. For example, the high signal-to-noise ratio approximations replaces the hyperbolic tangent block with a sign block. Compare this block diagram with the QPSK timing PLL illustrated in Figure 3.39.

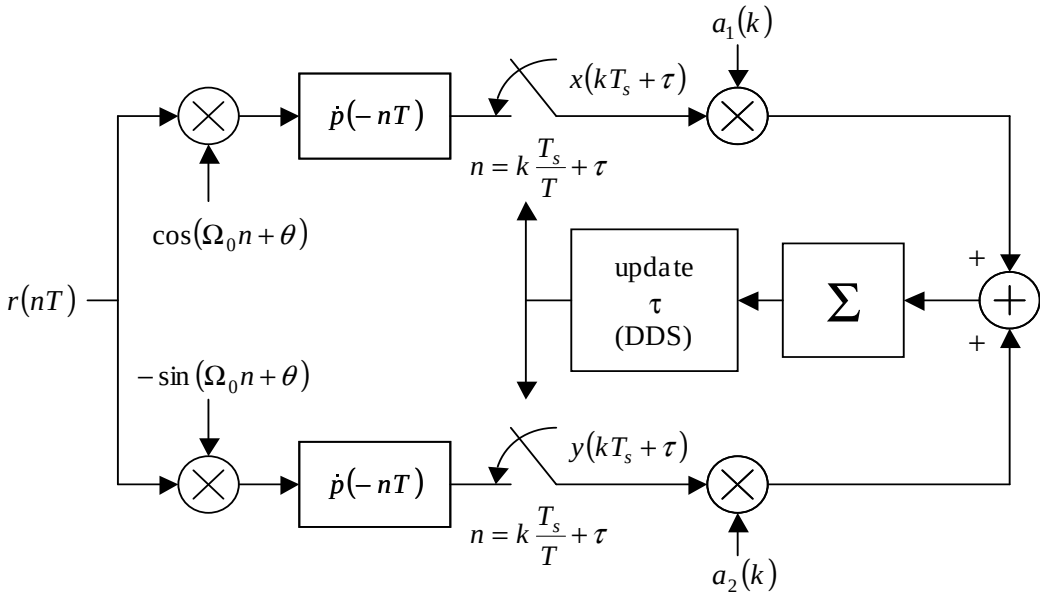


Figure 3.77: Block diagram of the maximum-likelihood QPSK timing estimator based on (3.230).

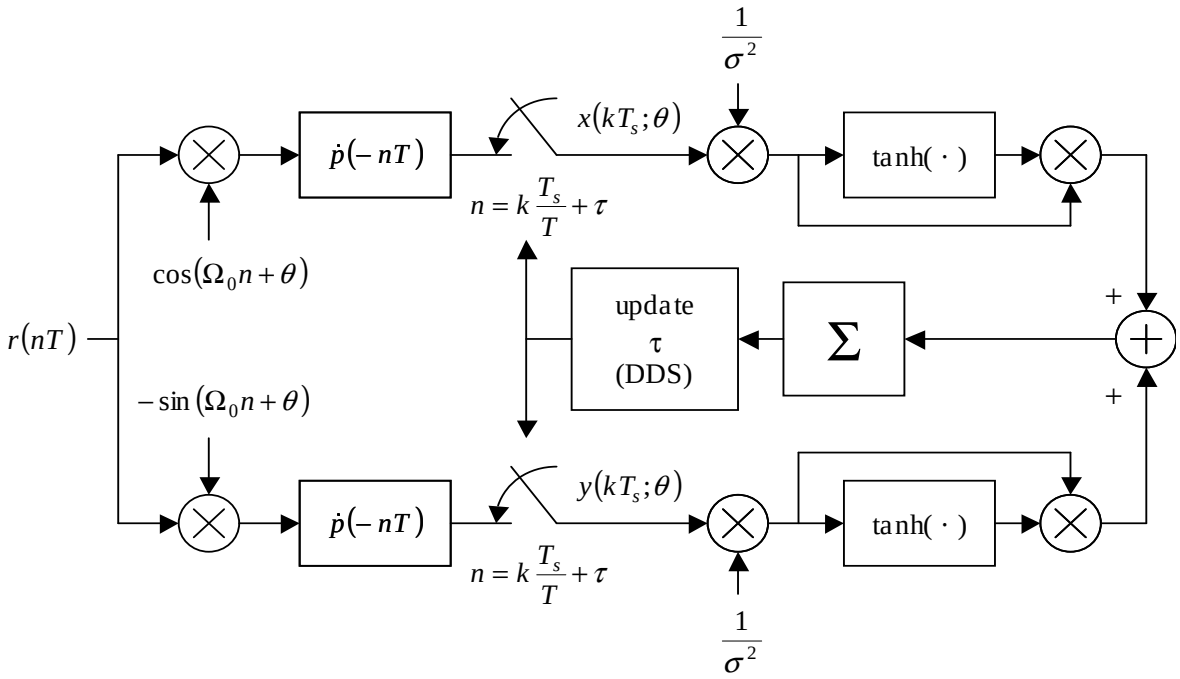


Figure 3.78: Block diagram of the maximum-likelihood QPSK timing estimator based on (3.234).

3.7 Notes and References

In the early years of digital communications, synchronization subsystems were characterized by ad-hoc techniques that later were shown to be approximations to maximum likelihood estimation. There are several aspects of synchronization that were not covered in this chapter. These include frequency synchronization, non-iterative techniques for carrier phase estimation (this is particularly useful in packetized burst communications), frame synchronization, and carrier phase and symbol timing synchronization for CPM. Many text books cover synchronization from a more theoretical point of view [4, 5, 6]. I have been strongly influenced by the wonderful text by Umberto Mengala and Aldo D'Andrea [4] which emphasizes discrete-time techniques. For symbol timing synchronization, the seminal papers by Gardner and his colleagues at the European Space Agency [1, 3]. I have tried to provide a strong link between discrete-time phase lock loops and the phase/timing error signals developed in the text as this important topic has not received a lot of attention in the published work.

Bibliography

[1] F. Gardner, "Interpolation in digital modems — part I: Fundamentals," *IEEE Transactions on Communications*, vol. 41, no. 3, pp. 501–507, March 1993.

[2] X. Qin, H. Wang, L. Zeng, and F. Xiong, "An all digital clock smoothing technique— counting -prognostication," *IEEE Transactions on Communications*, vol. 51, no. 2, pp. 166–169, February 2003.

[3] L. Erup, F. Gardner, and R. Harris, "Interplation in digital mdemes — part II: Implementation and performance," *IEEE Transactions on Communications*, vol. 41, no. 6, pp. 998–1008, June 1993.

[4] U. Mengali and A D'Andrea, *Synchronization Techniques for Digital Receivers*, Plenum Press, New Work, 1997.

[5] H. Meyr and G. Ascheid, *Synchronization in Digital Communications*, vol. 1, John Wiley & Sons, new york edition, 1990.

[6] J. Bingham, *The Thoery and Practice of Modem Design*, John Wiley & Sons, New York, 1983.

3.8 Exercises

3.1 Derive the expression given by (3.37) for the average S-curve for the linear QPSK data-aided phase error detector based on the error signal (3.36).

3.2 Derive the expression given by (3.40) for the average S-curve for the linear QPSK decision-directed phase error detector based on the error signal (3.39).

3.3 Show that the sine of the phase error for the linear QPSK phase error detector is given by (3.42).

3.4 Derive the expression given by (3.44) for the average S-curve for the simplified QPSK data-aided phase error detector based on the error signal (3.43).

3.5 Derive the expression given by (3.46) for the average S-curve for the simplified QPSK decision-directed phase error detector based on the error signal (3.45).

3.6 This problem explores the performance of carrier phase synchronization for QPSK.

(a) Compare the S-curves for the data-aided phase error detector (3.37) and the decision-directed phase error detector (3.40) for the linear phase error detector. How are they the same? How are the different?

(b) Compare the S-curves for the data-aided phase error detector (3.44) and the decision-directed phase error detector (3.46) for the simplified phase error detector. How are they the same? How are they different?

(c) Compare the S-curves for the linear data-aided phase error detector (3.37) and the simplified data-aided phase error detector (3.44). How are they the same? How are they different?

(d) Compare the S-curve for the linear decision-directed phase error detector (3.40) and the simplified decision-directed phase error detector (3.44). How are they the same? How are they different?

3.7 Derive the expression given by (3.54) for the average S-curve for the linear BPSK data-aided phase error detector based on the error signal (3.51).

3.8 Derive the expression given by (3.54) for the average S-curve for the linear BPSK decision-directed phase error detector based on the error signal (3.52).

3.9 Derive the expression given by (3.57) for the average S-curve for the simplified BPSK data-aided phase error detector based on the error signal (3.55).

3.10 Derive the expression given by (3.58) for the average S-curve for the simplified BPSK decision-directed phase error detector based on the error signal (3.56).

3.11 This problem explores the performance of carrier phase synchronization for BPSK.

(a) Compare the S-curves for the data-aided phase error detector (3.53) and the decision-directed phase error detector (3.54) for the linear phase error detector. How are they the same? How are the different?

(b) Compare the S-curves for the data-aided phase error detector (3.57) and the decision-directed phase error detector (3.58) for the simplified phase error detector. How are they the same? How are they different?

(c) Compare the S-curves for the linear data-aided phase error detector (3.53) and the simplified data-aided phase error detector (3.57). How are they the same? How are they different?

(d) Compare the S-curve for the linear decision-directed phase error detector (3.54) and the simplified decision-directed phase error detector (3.57). How are they the same? How are they different?

3.12 This problem explores S-curves for the Y constellation.

(a) Derive the average S-curve for the Y constellation for a phase error detector based on an error signal of the form (3.36).

(b) Derive the average S-curve for the Y constellation for a phase error detector based on an error signal of the form (3.39).

(c) Derive the average S-curve for the Y constellation for a phase error detector based on an error signal of the form (3.43).

(d) Derive the average S-curve for the Y constellation for a phase error detector based on an error signal of the form (3.45).

3.13 This problem explores S-curves for the 8-PSK constellation.

(a) Derive the average S-curve for the 8-PSK constellation for a phase error detector based on an error signal of the form (3.36).

(b) Derive the average S-curve for the 8-PSK constellation for a phase error detector based on an error signal of the form (3.39).

(c) Derive the average S-curve for the 8-PSK constellation for a phase error detector based on an error signal of the form (3.43).

(d) Derive the average S-curve for the 8-PSK constellation for a phase error detector based on an error signal of the form (3.45).

3.14 This problem explores S-curves for the 16-QASK constellation.

(a) Derive the average S-curve for the 16-QASK constellation for a phase error detector based on an error signal of the form (3.36).

(b) Derive the average S-curve for the 16-QASK constellation for a phase error detector based on an error signal of the form (3.39).

(c) Derive the average S-curve for the 16-QASK constellation for a phase error detector based on an error signal of the form (3.43).

(d) Derive the average S-curve for the 16-QASK constellation for a phase error detector based on an error signal of the form (3.45).

3.15 Derive the S-curve for the data-aided MLTED given by (3.107) based on the error signal (3.105).

3.16 Derive the S-curve for the data-aided ELTED given by (3.112) based on the error signal (3.110).

3.17 Derive the S-curve for the data-aided ZCTED given by (3.118) based on the error signal (3.113).

3.18 Derive the S-curve for the data-aided MMTED given by (3.121) based on the error signal (3.119).

3.19 Derive the linear interpolator filter (3.132) from (3.130) and (3.131).

3.20 Derive the cubic interpolator filter (3.138) from (3.136) and (3.137).

3.21 This problem steps through the derivation of the piece-wise parabolic interpolator (3.143).

(a) Using the second order polynomial approximation

$$x(t) = c_2 t^2 + c_1 t + c_0$$

express $x\left((m+\mu)T\right)$ as a polynomial in μ. The answer should be of the form

$$x((m+\mu)T) = b_2\mu^2 + b_1\mu + b_0$$

where the b's are functions of the c's, m, and T.

(b) Using the boundary conditions $x(mT)$ and $x((m+1)T)$, solve for b_0 and b_1 and show that $x((m+\mu)T)$ may be expressed as

$$x((m+\mu)T) = c_2T^2\left(\mu^2-\mu\right) + \mu x((m+1)T) + (1-\mu)x(mT)$$

This result shows that $x((m+\mu)T)$ is a linear combination of $x((m+1)T)$ and $x(mT)$ plus another term. If c_2 is also a linear combination of $x((m+1)T)$ and $x(mT)$, then $x((m+\mu)T)$ can be regarded as the output of a filter with inputs $x(mT)$ and $x((m+1)T)$. Part (c) shows that c_2 must be a function of more than $x(mT)$ and $x((m+1)T)$ in order to produce a piece-wise parabolic interpolator. In part (d), a piece-wise parabolic interpolator of the form given by (3.143) is derived.

(c) Suppose c_2 is a linear combinatation of $x((m+1)T)$ and $x(mT)$. That is

$$c_2 = A_{-1}x((m+1)T) + A_0x(mT).$$

Substitute the above relationship into the expression in part (b) and express $x((m+\mu)T)$ as a linear combination of $x(mT)$ and $x((m+1)T)$:

$$x((m+\mu)T) = B_{-1}x((m+1)T) + B_0x(mT).$$

There are two unknowns in the resulting equation: A_{-1} and A_0. The linear phase and unity gain constraints provide two conditions that can be used to solve for the unknowns. The linear phase constraint means the coefficients are symmetric about the center of the filter. Since the center of the filter corresponds to $\mu = 1/2$, this constraint imposes the relationship $B_{-1} = B_0$ when $\mu = 1/2$. The unity gain constraint means $B_{-1} + B_0 = 1$. Show that the application of these two constraints requires $A_{-1} = A_0 = 0$ so that a linear interpolator is the only interpolator that satisfies all the constraints for this case.

(d) Since a even number of filter taps are required, suppose c_2 is a linear combination of $x((m+2)T)$, $x((m+1)T)$, $x(mT)$ and $x((m-1)T)$. That is

$$c_2 = A_{-2}x((m+2)T) + A_{-1}x((m+1)T) + A_0x(mT) + A_1x((m-1)T).$$

Substitute the above relationship into the expression in part (b) and express $x((m+\mu)T)$ as a linear combination of $x((m+2)T)$, $x((m+1)T)$, $x(mT)$ and $x((m-1)T)$ of the form

$$x((m+\mu)T) = B_{-2}x((m+2)T) + B_{-1}x((m+1)T) + B_0x(mT) + B_1x((m-1)T).$$

There are four unknowns in the resulting expression: A_{-2}, A_{-1}, A_0, and A_1. The linear phase and unity gain constraints provide three equations the four unknowns must satisfy. The linear phase constraint imposes the condition $B_{-1} = B_0$ and $B_{-2} = B_1$ when $\mu = 1/2$. The unity gain constraint imposes the condition $B_{-2} + B_{-1} + B_0 + B_1 = 1$. One more equation is needed to solve for the four unknowns. This remaining condition is provided by setting $A_2 = \alpha$ where α is a free parameter. Show that using these conditions to solve for A_{-2}, A_{-1}, A_0, and A_1, $x((m+\mu)T)$ may be expressed as

$$\begin{aligned} x((m+\mu)T) = {} & \left[\alpha\mu^2 - \alpha\mu\right] x((m+2)T) + \left[-\alpha\mu^2 + (\alpha+1)\mu\right] x((m+1)T) \\ & + \left[-\alpha\mu^2 + (\alpha-1)\mu + 1\right] x(mT) + \left[\alpha\mu^2 - \alpha\mu\right] x((m-1)T). \end{aligned}$$

3.22 Derive the Farrow filter structure for the linear interpolator.

(a) Produce a table similar to Table 3.1.

(b) Sketch a block diagram of the resulting Farrow filter similar to those shown in Figure 3.56.

3.23 Do the following for the piece-wise parabolic interpolator:

(a) Derive the Farrow coefficients for the piece-wise parabolic interpolator listed in Table 3.1.

(b) Sketch a block diagram of the Farrow filter similar to that shown in Figure 3.56 for the general piece-wise parabolic interpolator.

(c) Show that when $\alpha = 1/2$, the answer in part (b) reduces to the structure shown in Figure 3.56.

3.24 Derive the Farrow coefficients for the cubic interpolator listed in Table 3.2.

3.25 Derive the maximum likelihood carrier phase estimator for BPSK assuming a known bit sequence and known timing.

3.26 Derive the maximum likelihood carrier phase estimator for BPSK assuming an unknown bit sequence and known timing.

3.27 Derive the maximum likelihood carrier phase estimator for BPSK assuming and unknown bit sequence and unknown timing.

3.28 Show that the data-aided carrier phase error signal (3.78) follows from the maximum likelihood carrier phase estimator for offset QPSK assuming a known symbol sequence and known symbol timing.

3.29 Derive the maximum likelihood bit timing estimator for BPSK assuming a known bit sequence and known carrier phase.

3.30 Derive the maximum likelihood bit timing estimator for BPSK assuming an unknown bit sequence and known carrier phase.

3.31 Derive the maximum likelihood bit timing estimator for BPSK assuming an unknown bit sequence and unknown carrier phase.

3.32 Derive the maximum likelihood symbol timing estimator for offset QPSK assuming a known symbol sequence and known carrier phase.

3.33 Derive the maximum likelihood symbol timing estimator for offset QPSK assuming an unknown symbol sequence and known carrier phase.