TIMING SYNCHRONIZATION

Software-Defined Radio for Engineers: Chapter 6

Tuesday 23rd March, 2021

Manuel Ríos

Universidad Rafael Landívar

6.1 Matched Filter

Introduction

We will be introducing **receiver synchronization** and **signal recovery** concepts, which are necessary for wireless communications. Downstream recovery methods used in this book for coherent modulations are sensitive to **timing offset** and must be compensated for first. Since signals must travel a distance between the transmitting DAC and receiving ADC there will be a fixed but random time offset between the chains. **Timing recovery** is used to correct for this offset.

Introduction

A receiver can be designed in many different ways but the specific ordering of chapters here relates to the successive application of algorithms to be used: First **timing recovery**, then **carrier phase correction**, and finally **frame synchronization**. Then we will move on to more advanced topics including **coding** and **equalization**.

Receiver Blocks

Blocks in Figure 6.1 will be highlighted at the start of each relevant chapter to outline the progress of the overall receiver design and show how they fit with one another. In this chapter, matched filtering and timing recovery are highlighted.

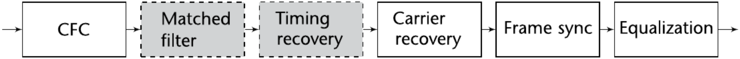


Figure 1: Receiver block diagram.

Pulse Shaping and Matched Filter

In digital communications theory:

Pulse-Shaping (transmitter) same as **Matched Filtering** (receiver)

Matched Filtering Benefits

The goal of these techniques is threefold:

1. Make the **signal suitable to be transmitted** through the communication channel by **limiting** its effective **bandwidth**.
2. **Increase** the **SNR** of the received waveform.
3. **Reduce intersymbol interference** (ISI) from multipath channels and nonlinearities.

Matched Filtering Effect

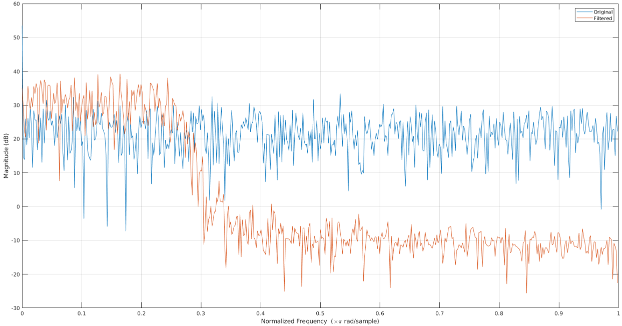


Figure 2: Frequency spectrum of PSK signal before and after pulse-shaping filter.

Source code: `MATLAB/Chapter_06/srrcFilter.m`

Matched Filtering Advantages

- When filtering a symbol, **sharp phase** and **frequency transitions** are reduced resulting in a more **compact** and **spectrally efficient signal**.
- These filter stage implementations will typically **upsample** and **downsample** signals, which **reduce their effective bandwidth**.
- Upsampling inherently increases **surface area of a symbol**, making it easier to determine, since we will have **multiple copies** of it at the receiver.

Matched Filtering Advantages

- Data will be produced at the **same rate from the transmitter** but will **not utilize the entire bandwidth available**.
- These operations of rate transitions (upsampling/downsampling) are performed during the matched filtering stages since it is **efficient to utilize a single filter to perform both operations**.

Square-Root Raised Cosine Filter

The square-root raised cosine (SRRC) is one of the most common filter used in communication systems. The SRRC has the impulse response given by the following equation:

$$h(t) = \begin{cases} \frac{1}{\sqrt{T_s}}\left(1-\beta+4\frac{\beta}{\pi}\right), & t=0 \\ \frac{\beta}{\sqrt{2T_s}}\left[\left(1+\frac{2}{\pi}\right)\sin\left(\frac{\pi}{4\beta}\right)+\left(1-\frac{2}{\pi}\right)\cos\left(\frac{\pi}{4\beta}\right)\right], & t=\pm\frac{T_s}{4\beta} \\ \frac{1}{\sqrt{T_s}}\frac{\sin\left[\pi\frac{t}{T_s}(1-\beta)\right]+4\beta\frac{t}{T_s}\cos\left[\pi\frac{t}{T_s}(1+\beta)\right]}{\pi\frac{t}{T_s}\left[1-\left(4\beta\frac{t}{T_s}\right)^2\right]}, & \text{otherwise} \end{cases}$$

where T_s is the symbol period and $\beta \in [0, 1]$ is the roll-off factor.

Transmitter-Receiver Arrangements

Filters can be arranged in two ways

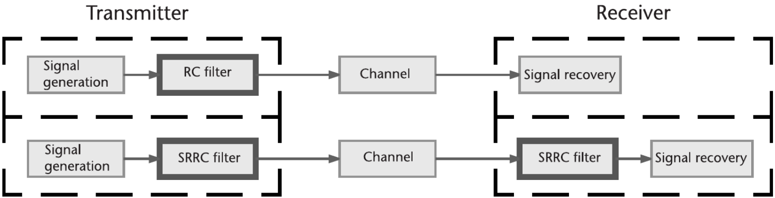


Figure 3: Arrangements of transmit filters with respect to the transmitter and receiver nodes for raised cosine and root-raised cosine filters.

Eye Diagram for Matched Filter

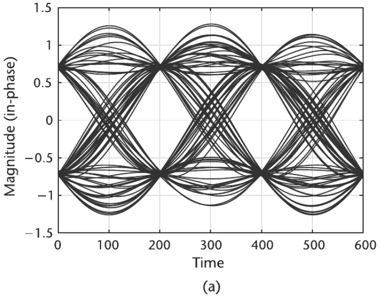


Figure 4: Eye diagrams of in-phase portion of QPSK signal after being passed through SRRC filters with different β values. (a) $\beta = 0.3$, and (b) $\beta = 0.99$ for an SNR of $27dB$.

Source code: `MATLAB/Chapter_06/srrcFilterDataEye.m`

Impulse Response for Matched Filter

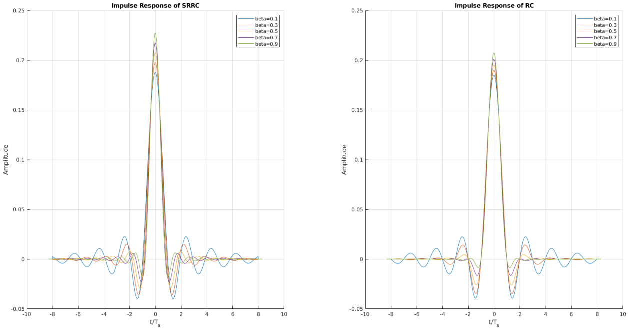


Figure 5: Impulse response comparison between square-root raised-cosine and raised-cosine filters. (a) SRRC impulse response, and (b) RC impulse response.

Source code: `MATLAB/Chapter_06/srrc_impulse_response.m`

Frequency Response for Matched Filter

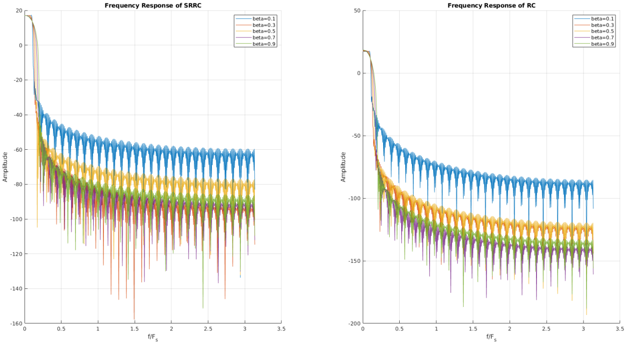


Figure 6: Frequency response comparison between square-root raised-cosine and raised-cosine filters. (a) SRRC frequency response, and (b) RC frequency response.

Source code: `MATLAB/Chapter_06/srrc_freq_response.m`

Effect of Nonlinearities

Nonlinearities cause **amplitude and phase distortions**, which can happen when we clip or operate at the limits of our transmit amplifiers.

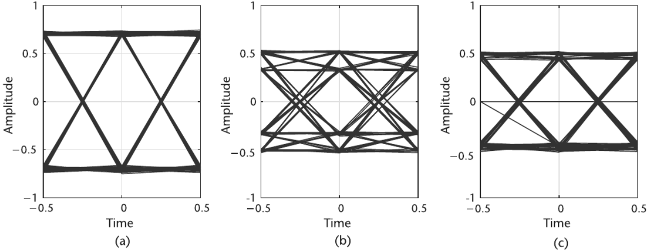


Figure 7: Eye diagrams of QPSK signal affected by nonlinearity causing ISI, which is reduced by SRRC matched filtering. (a) Original signal at transmitter, (b) passed through nonlinearity without pulse-shaping, and (c) SRRC filters used at transmitter and receiver with nonlinearity.

Rate Conversion with Polyphase Filters

Rate conversion will typically occur in these transmit or receive filters. Therefore, a polyphase filter can be used where the taps of the SRRC filter are used within the individual arms of the polyphase filter.

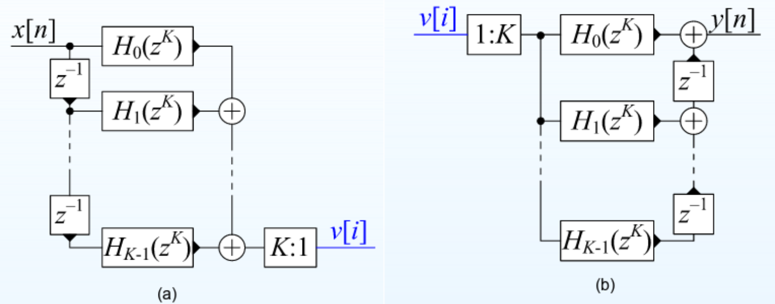


Figure 8: Polyphase filter structure. (a) Downsampler, and (b) Upsampler.

A Note on Matched Filter

Universidad Rafael Landívar
Tradición Jesuita en Guatemala

Matched Filter Maximizes the Signal

Since the pulsed-shaped/filtered signal is correlated with the pulse-shaped filter and not the noise, matched filtering will have the effect of **SNR maximizing the signal**, creating peaks at central positions of receive pulses.

6.2 Timing Error

Purpose of Symbol Timing

Purpose of Symbol Timing

In the most basic sense the **purpose of symbol timing synchronization is to align the clocking signals**. or sampling instances of two disjointed communicating devices.

Fractional Delay

- A fractional delay τ is introduced at the reciever. (Less than a sample)
- Signal is sampled at the wrong positions and the eventual demodulated signal is incorrect.

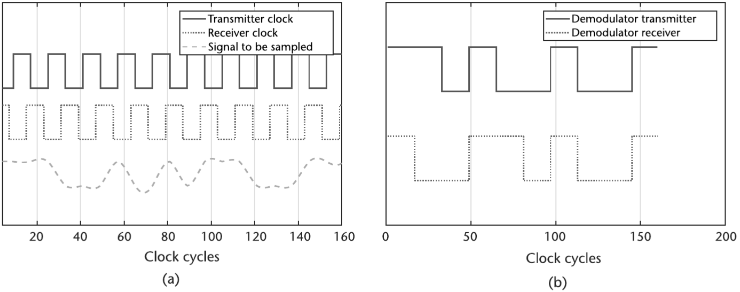


Figure 9: (a) Transmitter and receiver clocking signals with analog waveform to be sampled, and (b) demodulator outputs of receiver and transmitter according to their sampling times.

Mathematical Model of Delay

We can model this offset received signal r as:

$$r(t) = \sum_{n} x(n)h\left(t - \tau(t) - nT_s\right) + v(t)$$

where:

- x is the transmitted symbol
- h is the pulse shape of the transmit filter
- τ is the fractional offset
- T_s is the sampling period
- n is the sample index
- v is the additive channel noise

Mathematical Model of Delay

Universidad Rafael Landívar
Tradición Jesuita en Guatemala

After reception the r is passed through the receive matched filter and the relation of the source symbols will be:

$$y(t) = \sum_n x(n)\bar{h}_A\left(t - \tau(t) - nT_s\right) + v_h(t)$$

where:

- $h_A = h(t) * \bar{h}(-t)$ is the autocorrelation of the transmit filter and its conjugate used on the source data x
- v_h is the shaped noise
- y is the output symbols

Interpolation and Pulse Shaping at the Transmitter

- **Interpolate** the signal to be transmitted at the transmit filter stage **before** actually sending the signal.
- Reduces the throughput of our system since it provides the receiver **more data** to perform decisions **without** having to **oversample** the signal itself.

MATLAB

`comm.RaisedCosineTransmitFilter` uses a polyphase interpolator to upsample the signal and applies the necessary RC or SRRC taps.

Upsampling Factor N

- The upsampling factor N, also known as **sample per symbol**, will be chosen **based on the recovery algorithms** used and the **desired data rate of the system**.
- N can improve the recovery process at the receiver to a point, but will **reduce our useful bandwidth**, forcing hardware to run at higher rates to achieve the same throughput.

Variable Delay τ

- The unknown delay τ **must be estimated** to provide correct demodulation downstream.
- A crude, but simple way, can be to **fractionally resample** the signal with use of a polyphase filter.

MATLAB

`dsp.VariableFractionalDelay` delays the input signal by a specified number of fractional samples.

Different Effect of Variable Delay τ

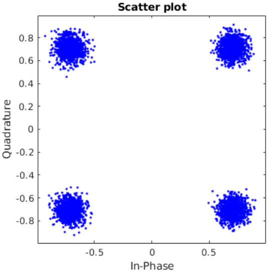


Figure 10: Offset of $\tau = 0.1N$

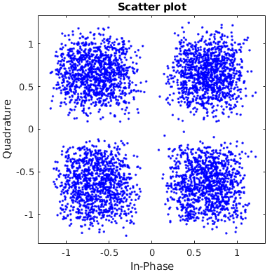


Figure 11: Offset of $\tau = 0.2N$

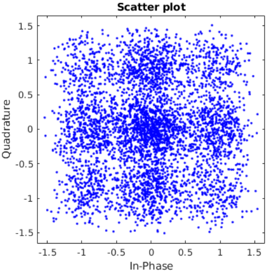


Figure 12: Offset of $\tau = 0.5N$

Source code: `MATLAB/Chapter_06/qpskTimingError.m`

Estimate $\hat{\tau}$

Correct sampling can be obtained if we find a value $\hat{\tau}$ that satisfies $\hat{\tau} + \tau = kT_s$ and $k = \mathbb{Z}_{\geq 0}$

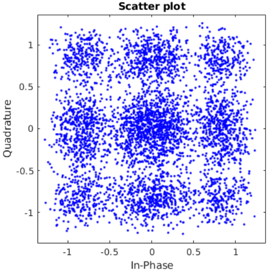


Figure 13: $\tau = 0.2N$ and $\hat{\tau} = 0.1N$

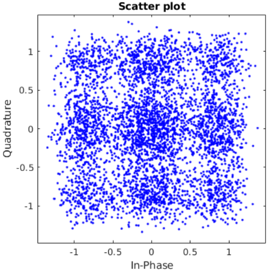


Figure 14: $\tau = 0.2N$ and $\hat{\tau} = 0.2N$

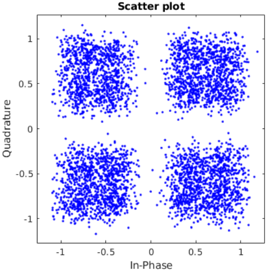


Figure 15: $\tau = 0.2N$ and $\hat{\tau} = 0.3N$

Source code: `MATLAB/Chapter_06/qpskTimingErrorTauHat.m`

6.3 Symbol Timing Compensation

Timing Compensation

We will use a PLL because:

- Share same methodology as in carrier recovery implementations (Chapter 7)
- Can be integrated with future recovery solutions
- Can be robust
- Is not overly algorithmicly complex

Three types of detectors will be discussed:

- Zero-Crossing (ZC)
- Müller/Mueller
- Gardner

Structure of a PLL

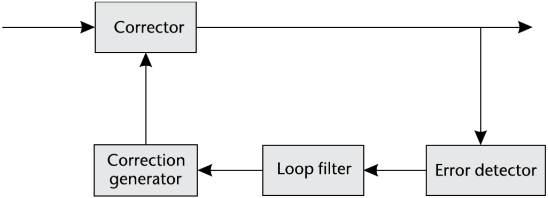


Figure 16: Basic PLL structure with four main component blocks.

- **Error Detector**: Timing or phase error of the received sample. Designed based on the structure of the desired receive constellation/symbols or the nature of the sequence itself.

Structure of a PLL

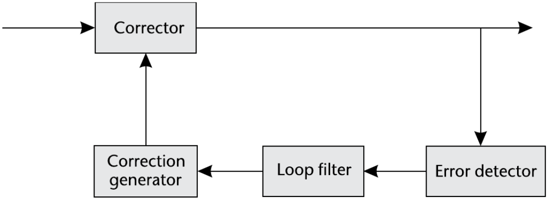


Figure 17: Basic PLL structure with four main component blocks.

- **Loop Filter**: Governs the dynamics of the PLL. Determines operational ranges, lock time, and dampness/responsiveness of the PLL.

Structure of a PLL

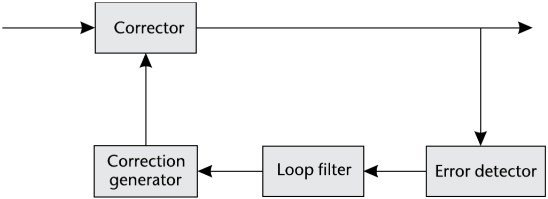


Figure 18: Basic PLL structure with four main component blocks.

- **Correction Generator**: Responsible for generation of the correction signal for the input, which again will be fed back into the system.

Structure of a PLL

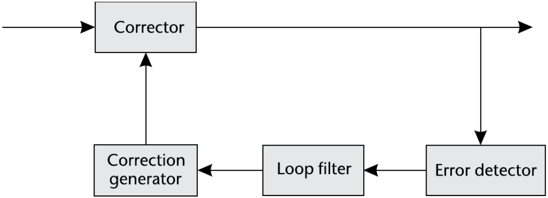


Figure 19: Basic PLL structure with four main component blocks.

- **Corrector**: Modifies the input signal based on input from the correction generator.

Structure of a PLL

- **Correction generator**, **error detector**, and **corrector** are specific to the purpose of the PLL structure, such as timing recovery or carrier recovery.
- The **loop filter** can be shared among the designs with modification to its numerical configuration.

Loop Filter

Universidad Rafael Landívar
Tradición Jesuita en Guatemala

- **Can be shared** among the designs with modification to its **numerical configuration**.
- Most challenging aspect in PLL design, but **provides the most control** over the adaption of the system.

Loop Filter -PI Filter

Universidad Rafael Landívar
Tradición Jesuita en Guatemala

Use a proportional-plus-integrator (PI) filter as our loop filter

$$F(s) = g_1 + \frac{g_2}{s}$$

where g_1 and g_2 are selectable gains.

With discrete time signals a z-domain representation is preferable:

$$F(z) = G_1 + \frac{G_2}{1 - z^-1}$$

where $G_1 \neq g1$ and $G2 \neq g2$. (Proof use bilinear transform)

Loop Filter -Gain Values

Universidad Rafael Landívar
Tradición Jesuita en Guatemala

The gain values utilize the following equations based on a preferred damping factor ζ and loop bandwidth B_Loop:

$$\theta = \frac{B_{\text{Loop}}}{M(\zeta + 0.25/\zeta)} \quad \Delta = 1 + 2\zeta\theta + \theta^2$$

$$G_1 = \frac{4\zeta\theta/\Delta}{M} \qquad G_2 = \frac{4\theta^2/\Delta}{M}$$

- M is the samples per symbol associated with the input signal.
- B_Loop is a normalized frequency and can range $B_Loop \in [0, 1]$

Loop Filter -Damping Factor ζ

For the selection of ζ:

$$\zeta = \begin{cases} < 1, & \text{Underdamp} \\ = 1, & \text{Critically Damped} \\ > 1, & \text{Overdamped} \end{cases}$$

which will determine the responsiveness and stability of the PLL.

- M is the samples per symbol associated with the input signal.
- B_Loop is a normalized frequency and can range $B_Loop \in [0, 1]$

6.3 Symbol Timing Compensation

6.3.2 Feedback Timing Correction

Structure of a PLL for Timing Recovery

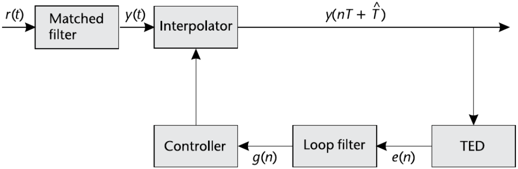


Figure 20: Basic structure of PLL for timing recovery for both decision direction and blind timing recovery.

1. Estimate unknown offset error.
2. Scale the error proportionally.
3. Apply an update for future symbols to be corrected.

Timing Error

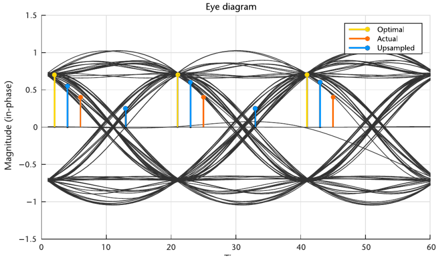


Figure 21: Eye diagram of received signal marking positions where received samples may exist. This figure is highly oversampled to show many positions, but a received sample could lie anywhere on the eye.

If we are trying to find the optimal sampling position we can interpolate across the upsampled points to get our desired period. *Think of a curve that is fitted between two points and we find where it crosses zero.*

Timing Error

- This interpolation has the effect of **causing a fractional delay** to our sampling, essentially **shifting to a new position in our eye diagram**.
- Since τ is unknown we must **weight this interpolation correctly** so we do **not overshoot or undershoot** the desired correction.
- **Interpolator**, which works at symbol rate, block **responsible for this task**.
- If all blocks work correctly, **eye diagram will open**.

Zero Crossing Method

- Produces an error signal $e(n)$ of zero when one of the sampling positions is at the zero intersection.
- Requires two samples per symbol or more.

$$
\begin{aligned}
e(n) = &\operatorname{Re}\left(y\left((n-1/2)T_{\mathrm{s}}+\tau\right)\right)\left[\operatorname{sgn}\left\{\operatorname{Re}\left(y\left((n-1)T_{\mathrm{s}}+\tau\right)\right)\right\}-\operatorname{sgn}\left\{\operatorname{Re}\left(y\left(nT_{s}+\tau\right)\right)\right\}\right] \\
&+\operatorname{Im}\left(y\left((n-1/2)T_{\mathrm{s}}+\tau\right)\right)\left[\operatorname{sgn}\left[\operatorname{Im}\left(y\left((n-1)T_{s}+\tau\right)\right)\right]-\operatorname{sgn}\left[\operatorname{Im}\left(y\left(nT_{s}+\tau\right)\right)\right]\right]
\end{aligned}
$$

- Indexes are with respect to samples, not symbols.

Loop Filter

Once the error is calculated it is passed to the loop filter.

$$G_1 = \frac{-4\zeta\theta}{G_D N\Delta} \quad G_2 = \frac{-4\theta^2}{G_D N\Delta}$$

- B_Loop is the normalized loop bandwidth
- ζ is the damping factor
- N is samples per symbol
- G_D is the detector gain, which provides extra scaling

This filter can be implemented with a simple linear equation:

$$y(t) = G_1 x(t) + G_2 \sum_{n=0} y(n)$$

Interpolation Controller

- Responsible to providing the necessary **signaling** to the **interpolator**
- Takes place of the **correction generator**
- Provides **information of the starting interpolant sample**
- Utilizes a **counter-based** mechanism to trigger at the appropriate symbol positions
- At these **trigger positions the interpolator is signaled** and updated, as well as an output symbol is produced from the system

Interpolation Controller -Counter Based Controller

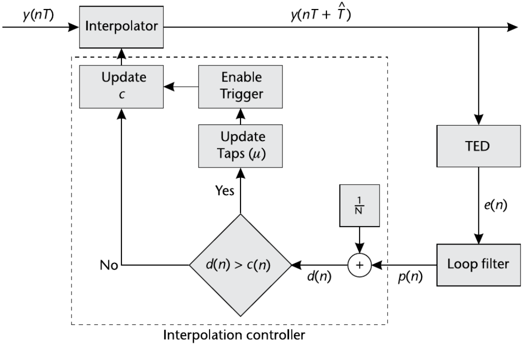
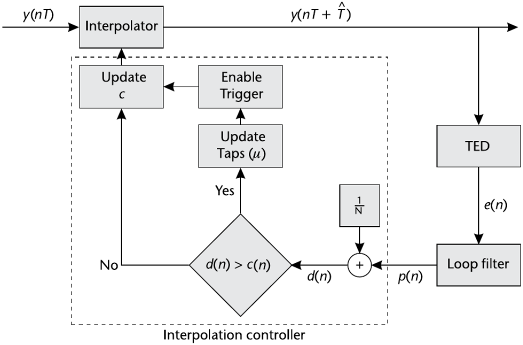


Figure 22: Timing recovery triggering logic used to maintain accurate interpolation of input signal.

The main idea behind a counter-based controller is to **maintain a specific triggering gap** between updates to the interpolator, with an **update period** on average **equal to symbol rate N** of the input stream.

Interpolation Controller -Counter Based Controller

Universidad Rafael Landívar
Tradición Jesuita en Guatemala




```
% Interpolation Controller with modulo-1 counter
d = g + 1/N;
Trigger = (Counter < d); % Check if a trigger condition
if Trigger % Update mu if a trigger
    mu = Counter / d;
end
Counter = mod(Counter - d, 1); % Update counter
```

Interpolation

- Takes the place of the corrector.
- Simply a **linear combination** of the current and past inputs y.
- **FIR filter** with any arbitrary delay $\tau \in [0, T_s]$ **cannot be realized**
- **IIR filters** do exist, but the computation of their taps are **impractical in real systems**
- Low pass FIR **Piecewise polynomial filter** (PPF) can only provide estimations of offsets to a polynomial degree
- **Alternative** implementations exists such as **polyphase-filterbank** designs

Interpolation -Piecewise Polynomial Filter

- We can easily control the form of interpolations by determining the order of the filter, which at most is equivalent to the order of the polynomial used to estimate the underlying received signal.

A second order, or quadratic, interpolation requiring a four-tap filter is given by:

$$y\left(kT_s + \mu(k)T_s\right) = \sum_{n=1}^{2} h(n)y\left((k-n)T_s\right)$$

Interpolation -Piecewise Polynomial Filter

where h_k are the filter coefficients at time instance k determined by:

$$\begin{aligned} h = [&\alpha\mu(k)(\mu(k)-1), \\ &-\alpha\mu(k)^2 - (1-\alpha)\mu(k) + 1, \\ &-\alpha\mu(k)^2 + (1+\alpha)\mu(k), \\ &\alpha\mu(k)(\mu(k)-1)] \end{aligned}$$

- where $\alpha = 0.5$
- $\mu(k)$ is related to the fractional delay

We can estimate the true delay τ as:

$$\hat{\tau} \sim \mu(k) T_s$$

Timing Synchronization Blocks

Table 1 outlines the rates for the timing recovery blocks:

Block	Operational Rate	Matlab Script
Interpolator	Sample Rate	`interpFilter`
TED	Symbol Rate	`zcTED`
Loop Filter	Symbol Rate	`loopFilter`
Interpolator Controller	Sample Rate	`interpControl`

Table 1: Operational rates and Matlab scripts of timing recovery blocks

Source code: `MATLAB/Chapter_06/timing_sync/`

6.4 Alternative Error Detectors and System Requirements

Limits of Zero Crossing Method

- Can't operate under carrier phase or frequency offsets (would require compensation first before application of ZC)
- Upsample factor N of at least two may not be possible for certain systems due to bandwidth and data rate constraints

Gardner TED

The error signal is determined by:

$$
\begin{aligned}
e(n) = &\operatorname{Re}\left(y\left((n-1/2)T_s+\tau\right)\right)\left[\operatorname{Re}\left(y\left((n-1)T_s+\tau\right)\right)-\operatorname{Re}\left(y\left(nT_s+\tau\right)\right)\right]+\\
&\operatorname{Im}\left(y\left((n-1/2)T_s+\tau\right)\right)\left[\operatorname{Im}\left(y\left((n-1)T_s+\tau\right)\right)-\operatorname{Im}\left(y\left(nT_s+\tau\right)\right)\right]
\end{aligned}
$$

- Requires two samples per symbol
- Does not require carrier phase correction
- Performs well for BPSK and QPSK signals
- The excess bandwidth of the transmit filters should be $\beta \in [0.4, 1]$

Müller and Mueller TED

The error signal is determined by:

$$\begin{aligned} e(k) = & \operatorname{Re}\left(y\left((k)T_s + \tau\right)\right) \times \operatorname{sgn}\left\{\operatorname{Re}\left(y\left((k-1)T_s + \tau\right)\right)\right\} \\ & - \operatorname{Re}\left(y\left((k-1)T_s + \tau\right)\right) \times \operatorname{sgn}\left\{\operatorname{Re}\left(y\left((k)T_s + \tau\right)\right)\right\} \\ & + \operatorname{Im}\left(y\left((k)T_s + \tau\right)\right) \times \operatorname{sgn}\left\{\operatorname{Im}\left(y\left((k-1)T_s + \tau\right)\right)\right\} \\ & - \operatorname{Im}\left(y\left((k-1)T_s + \tau\right)\right) \times \operatorname{sgn}\left\{\operatorname{Im}\left(y\left((k)T_s + \tau\right)\right)\right\} \end{aligned}$$

- Most efficient method since does not require upsampling of the source data
- Operates best when the matched filtering used minimizes the excess bandwidth, meaning β is small
- Performance can be questionable at $N = 1$ due to the lack of information available per symbol

6.5 Putting the Pieces Together

System Level -Timing Recovery

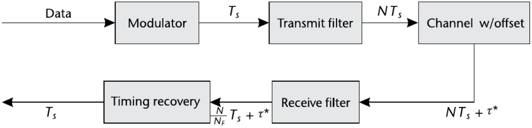


Figure 23: Relative rates of transmit and receive chains with respect to the sample rate at different stages.

- **Modulator** produces symbols equal to the sample rate.
- **Transmit Filter** increases samples by an upsampling factor N.
- **Receive Filter** decimates by a factor N_F, $N_F \leq N$.
- **Timing Recovery** across the remaining samples and remove fractional offsetτ.

System Level -Timing Recovery Matlab

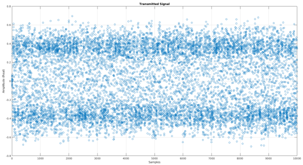


Figure 24: Transmitted Signal, Interpolation = 4

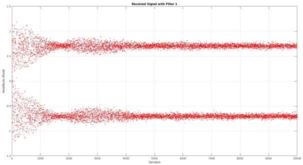


Figure 25: Received Signal, Decimation =1, Symbol Sync = 4

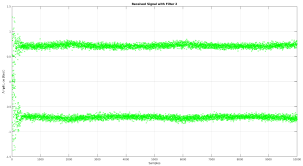


Figure 26: Received Signal, Decimation =2, Symbol Sync = 2

Source code:

`MATLAB/Chapter_06/systemExample.m`

Thank you!!!