Lab 5 - Digital Modulation: Symbol Synchronization

ECE531 – Software Defined Radio

Spring 2025

Daniel Gallagher, danielgallagher@arizona.edu
Department of Electrical & Computer Engineering
University of Arizona, Tucson AZ 85721

1 Overview and Objectives

This laboratory will introduce the concept of timing offset between transmitting and receiving nodes. Specifically, a simplified error model will be discussed along with three standard recovery methods, each with different performance objectives. Additionally, MATLAB® will be used as a development tool for digital communication systems, particularly for constructing a prototype software-defined radio (SDR) based simulation.

The specific learning objectives for this lab include:

- Understanding the effects of timing offset in digital communication systems
- Implementing and analyzing three standard timing recovery methods
- Developing practical SDR simulation skills using MATLAB
- Applying pulse shaping and matched filtering techniques

This lab assignment follows the background theory discussed in Chapter 6 of the textbook. MATLAB source code and implementation examples helpful for this lab are found in this chapter.

2 Pulse Shaping and Matched Filtering

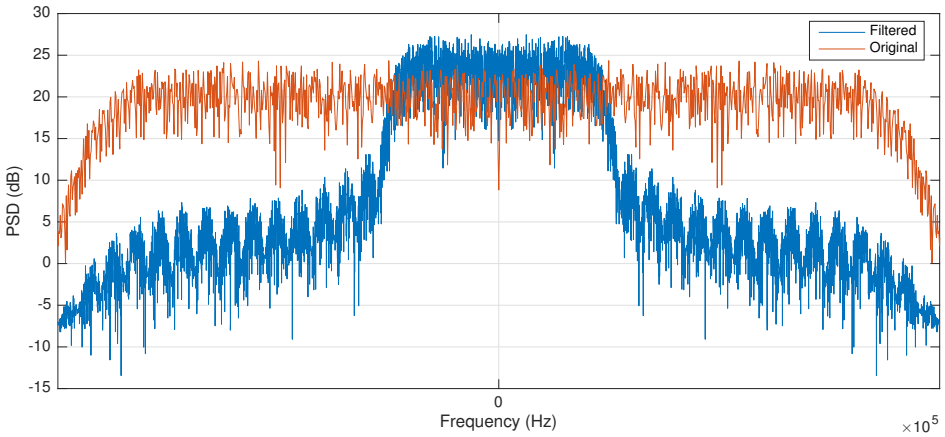

Figure 1: Frequency spectrum of PSK signal before and after pulse shaping.

The topics of matched filters and transmit filters are introduced here to establish their practical application and demonstrate the effects they help mitigate. In digital communications, these techniques are typically referred to as pulse shaping at the transmitter and matched filtering at the receiver. These processes serve three primary purposes: to limit the signal bandwidth making it suitable for transmission through the channel, to maximize the SNR of the received waveform, and to minimize intersymbol interference (ISI) caused by multi-path channels and system nonlinearities.

By filtering, a symbol's sharp phase and frequency transitions are smoothed, resulting in a more compact and spectrally efficient signal. Figure 1 demonstrates the frequency representation of a DBPSK signal before and after filtering with a transmit filter. The filter used in this example is a

square-root raised cosine (SRRC) filter, which is widely used in communication systems. While raised cosine filters were discussed in Chapter 2 (§2.7.5), the more commonly implemented SRRC has the impulse response:

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

where T_s is the symbol period and $\beta \in [0, 1]$ is the roll-off factor, which controls the bandwidth excess beyond the Nyquist frequency.

The SRRC filter is chosen because it belongs to the Nyquist class of filters, which produce zero ISI when sampled at the correct instances [1]. To illustrate the effects of ISI, we introduce a simple nonlinearity into the channel and examine the resulting eye diagrams (introduced in Section 2.4.1). Nonlinearities produce amplitude and phase distortions, which commonly occur when operating near the saturation region of transmit amplifiers. For details on the nonlinearity model used, see [2], though alternative models such as [3] also exist. In Figure 2, we observe how ISI causes the eye to compress, making accurate symbol timing determination more challenging, a critical consideration for the timing synchronization techniques discussed in subsequent sections.

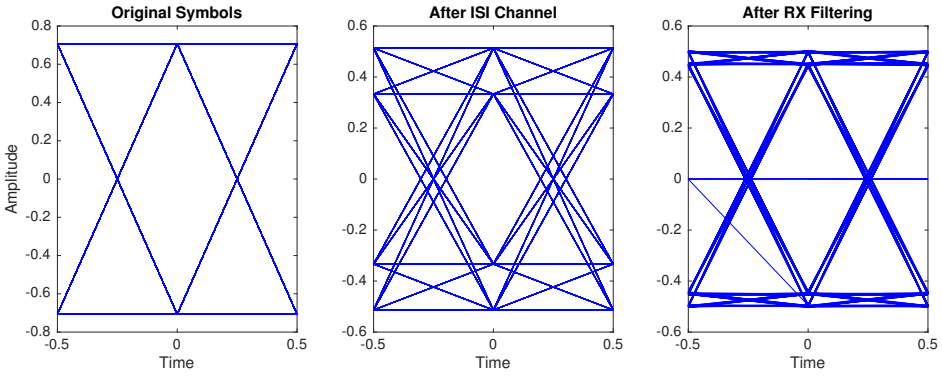

Figure 2: Eye diagrams of QPSK signal affected by nonlinearity causing ISI, which is reduced by SRRC matched filtering.

Beyond bandwidth limitation and ISI reduction, matched filtering maximizes the received signal's SNR. This benefit derives from correlation theory: since the received signal is correlated with the known pulse shape but the noise is not, matched filtering effectively enhances the signal peaks while spreading noise energy. Figure 3 demonstrates this effect, comparing signals transmitted with and without proper pulse shaping under AWGN conditions. The middle plot shows a signal

that closely resembles the transmitted sequence despite significant noise. Without pulse shaping (bottom plot), symbol detection becomes difficult, and demodulation errors occur even without timing offsets.

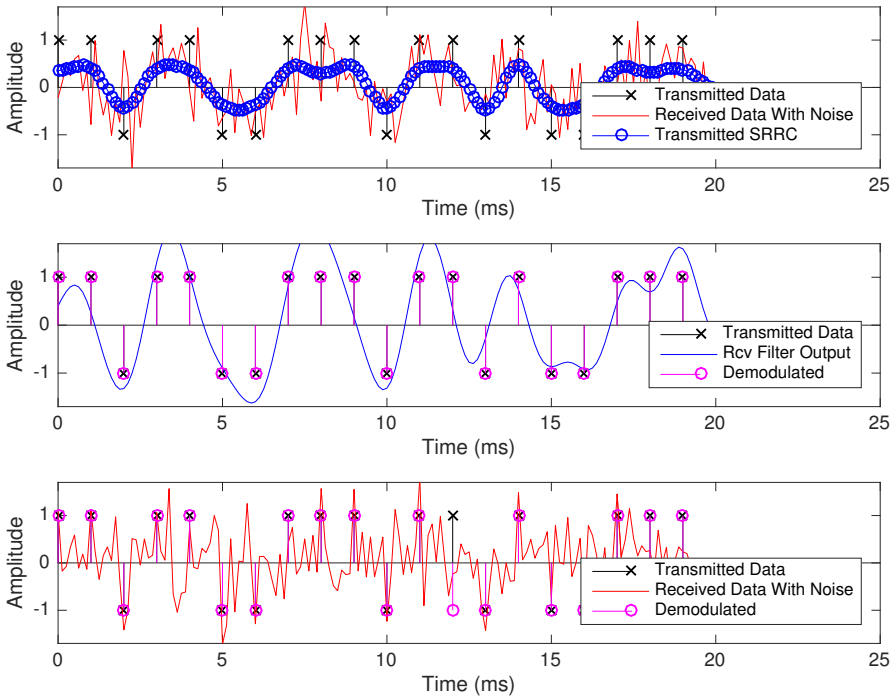

Figure 3: Comparison of pulse shaping and non-pulse shaping received signals under AWGN. (a) Transmitted SRRC filtered Signal (b) received SRRC filtered signal, (c) received signal without SRRC filtering at receiver.

Proper pulse shaping is especially critical for timing synchronization, as it creates well-defined symbol transitions that timing recovery algorithms can effectively detect and track. The choice of roll-off factor impacts not only spectral efficiency but also the performance of timing recovery methods discussed in subsequent sections.

2.1 Questions

1. Explain the difference between type I and type II Nyquist filters.
2. Generate frequency response plots for SRRC filter realizations with $\beta \in [0, 0.1, 0.25, 0.5, 1]$ and briefly discuss how the roll-off factor affects both spectral efficiency and filter complexity.
3. Generate transmit and receive plots similar to Figure 3 with $\beta \in [0.1, 0.5, 0.9]$ and analyze how different roll-off factors affect timing recovery performance.

3 Timing Error

In the most basic sense, the purpose of symbol timing synchronization is to align the sampling instances between a transmitter and receiver in a digital communication system. Without proper synchronization, the receiver may sample the incoming signal at non-optimal points, leading to increased error rates or complete communication failure. In Figure 4, we illustrate an idealized scenario where the receiver's sampling clock (shown as rising edges) perfectly aligns with the optimal sampling points of the received waveform. However, in practical systems, various factors including oscillator drift, propagation delay variations, and Doppler effects introduce an unknown timing offset.

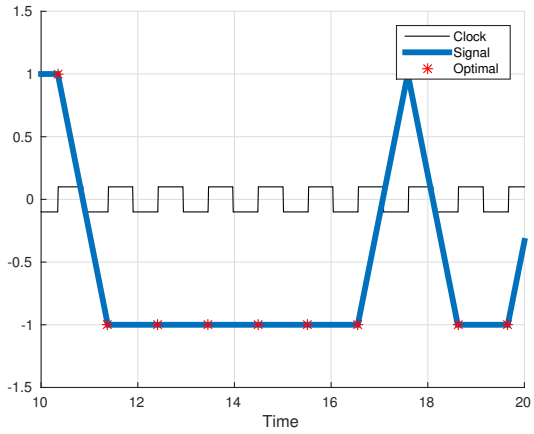

Figure 4: Example received symbols with associated optimal clocking signal. Sampling occurs at the rising edges of the clock.

When examining timing error in the constellation domain, we observe a characteristic "smearing" or clustering of the received symbols. Figure 5 demonstrates this effect in simulation for timing offsets of $0.2N$ and $0.5N$, where N represents the samples per symbol. The $0.5N$ offset represents the worst-case scenario, as the receiver is sampling exactly halfway between optimal sampling points. Figure 6 shows an actual QPSK signal transmitted through a Pluto SDR loopback configuration, exhibiting similar clustering patterns along with phase rotation due to carrier offset.

Mathematically, the received signal after matched filtering can be modeled as:

$$x(kT_s) = \sum_n x(n)p((k-n)T_s - \tau) + v(kT_s), \tag{2}$$

where p represents the autocorrelation of the pulse shape applied to the source data x, v is the filtered noise, and τ is the unknown timing offset that must be estimated for proper symbol recovery. This timing offset is typically a fraction of the symbol period T_s and varies slowly over time due to clock drift between transmitter and receiver oscillators.

The timing recovery methods discussed in Section 4 aim to estimate this offset τ and compensate for it through interpolation techniques, allowing the receiver to sample at the optimal points despite the presence of timing uncertainty.

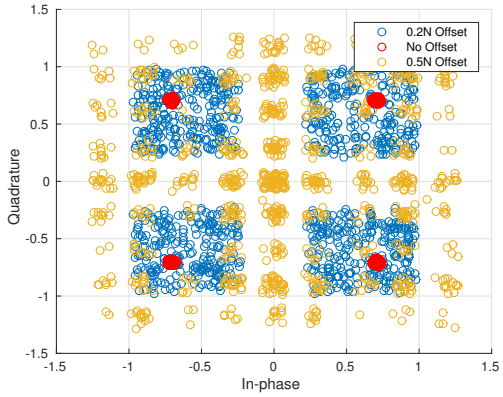

Figure 5: Simulation example showing timing offset effects on QPSK constellation at offsets of $0.2N$ and $0.5N$.

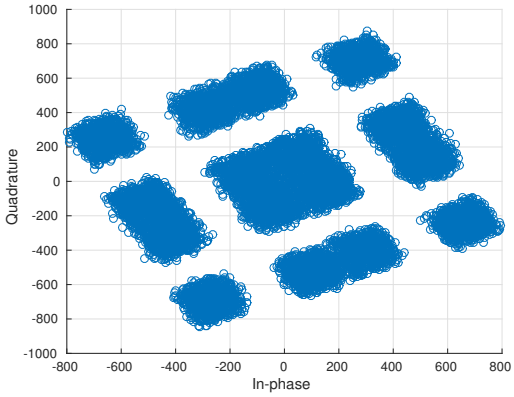

Figure 6: QPSK constellation with actual data sent through Pluto SDR using a loopback cable, showing both timing and phase offset effects.

3.1 Questions

1. Regenerate Figure 6 with data from your Pluto SDR and explain why the constellation appears rotated. It may be helpful to start with `plutoLoopback.m`.

4 Symbol Timing Compensation

In digital communications, symbol timing compensation is essential to correct mismatches between transmitter and receiver clocks. This section introduces digital phase-locked loop (PLL) methods for timing recovery with three different timing error detectors: Zero-Crossing (ZC), Gardner, and Müller and Mueller (MM). We selected PLL-based approaches because they integrate well with other carrier recovery solutions. First, we'll provide a conceptual overview of timing synchronization, then examine each component in detail.

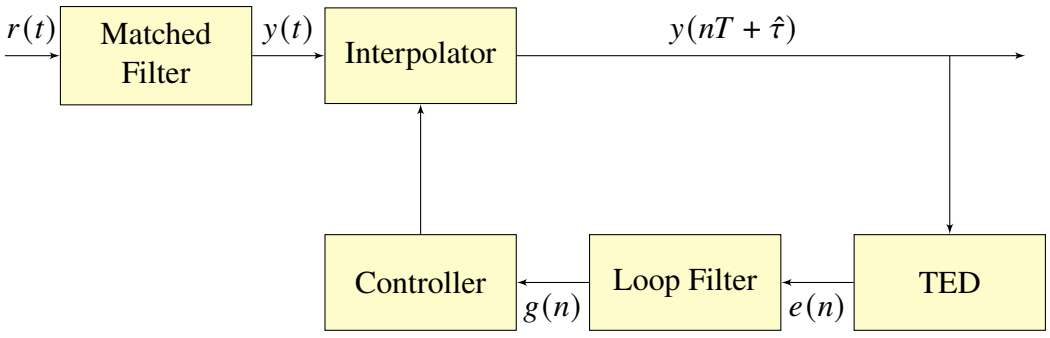

Figure 7: Basic structure of PLL for timing recovery.

The timing synchronization PLL shown in Figure 7 consists of four key components: interpolator, timing error detector (TED), loop filter, and an interpolator controller.

Where:

- Interpolator - applies fractional delay to input samples
- Timing Error Detector (TED) - measures timing offset
- Loop Filter - stabilizes error correction
- Interpolator Controller - manages the interpolation process

This all-digital PLL works by measuring an unknown timing offset, scaling the error appropriately, and applying corrections to subsequent symbols.

4.1 Timing Error Detection (TED)

To better understand timing error, consider the eye diagram in Figure 8, which has been up-sampled by a factor of twenty to clearly show transitions between symbols. This eye diagram is also noticiably smooth when compared to the ISI case in Figure 2. Ideally, we would sample the input signal at the red markers, where the eye is widest open. However, an unknown fractional delay τ shifts the optimal sampling points, resulting in the green sampling positions.

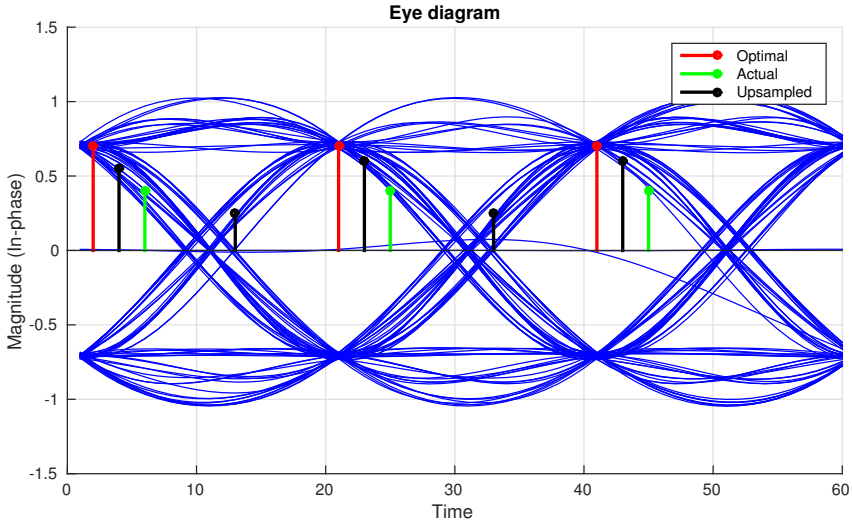

Figure 8: Eye diagram showing optimal sampling positions (red), shifted sampling due to timing offset (green), and actual sampling positions with multiple samples per symbol (black).

To address this issue, we typically oversample the received signal (providing multiple samples per symbol). When our actual sampling positions (black markers) straddle the optimal sampling point, we can interpolate between them to estimate the symbol value at the optimal time. This interpolation effectively introduces a fractional delay to our sampling, shifting to the desired position in the eye diagram. Since τ is unknown, our PLL must adaptively determine the correct interpolation weight to not overshoot or overshoot the desired correction.

4.1.1 Zero-Crossing (ZC) Method

We'll first examine the Zero-Crossing (ZC) method as it provides a clear foundation for understanding timing recovery. As the name suggests, this method produces a zero error signal when one sampling position occurs at a zero-crossing between symbols. ZC requires two samples per symbol, with one sample ideally at the zero-crossing and the other at or near the optimal symbol position.

The Timing Error Detector (TED) for ZC [4] is given by:

$$\begin{aligned} e(k) = &\operatorname{Re}(x((k-1/2)T_s+\tau))\big[\operatorname{sgn}\{\operatorname{Re}(x((k-1)T_s+\tau))\} - \operatorname{sgn}\{\operatorname{Re}(x(kT_s+\tau))\}\big] + \\ &\operatorname{Im}(x((k-1/2)T_s+\tau))\big[\operatorname{sgn}\{\operatorname{Im}(x((k-1)T_s+\tau))\} - \operatorname{sgn}\{\operatorname{Im}(x(kT_s+\tau))\}\big]. \end{aligned} \tag{3}$$

Note that these indices reference symbols, not samples. Equation (3) determines the error direction using the sgn operation, while the shift magnitude required to compensate is determined by the midpoint sample values. The in-phase and quadrature components operate independently, which is desirable for robust performance.

4.1.2 Gardner Method

The Gardner method [5] is similar to ZC but uses a different error calculation:

$$\begin{aligned} e(k) = &\operatorname{Re}(x((k-1/2)T_s+\tau))\big[\operatorname{Re}(x((k-1)T_s+\tau)) - \operatorname{Re}(x(kT_s+\tau))\big] + \\ &\operatorname{Im}(x((k-1/2)T_s+\tau))\big[\operatorname{Im}(x((k-1)T_s+\tau)) - \operatorname{Im}(x(kT_s+\tau))\big]. \end{aligned} \tag{4}$$

Like ZC, Gardner requires two samples per symbol. Its key advantage is that it doesn't require carrier phase correction, making it particularly effective for BPSK and QPSK signals. However, since Gardner is not decision-directed, it performs best when the transmit filter's excess bandwidth is in the range $\beta \in (0.4, 1)$.

4.1.3 Müller and Mueller Method

Müller and Mueller (MM) method is named after Kurt Mueller and Markus Müller [6]. This is considered a sample-efficient approach since it does not require upsampling of the source data, operating at one sample per symbol. Its error signal is given by [7]:

$$\begin{aligned} e(k) = &\operatorname{Re}(x(kT_s+\tau))\ \operatorname{sgn}\{\operatorname{Re}(x((k-1)T_s+\tau))\} \\ &- \operatorname{Re}(x((k-1)T_s+\tau))\ \operatorname{sgn}\{\operatorname{Re}(x(kT_s+\tau))\} \\ &+ \operatorname{Im}(x(kT_s+\tau))\ \operatorname{sgn}\{\operatorname{Im}(x((k-1)T_s+\tau))\} \\ &- \operatorname{Im}(x((k-1)T_s+\tau))\ \operatorname{sgn}\{\operatorname{Im}(x(kT_s+\tau))\}. \end{aligned} \tag{5}$$

MM performs best with matched filters that minimize excess bandwidth. With high excess bandwidth, the decisions made by the sgn operation may be unreliable.

4.2 Loop Filter Design

After calculating the timing error, it passes to the loop filter, which stabilizes the correction process. The filter uses these equations:

$$\theta = \frac{B_{\text{Loop}}}{M(\zeta + 0.25/\zeta)} \qquad \Delta = 1 + 2\zeta\theta + \theta^2 \tag{6}$$

$$G_1 = \frac{-4\zeta\theta}{G_D N\Delta} \qquad G_2 = \frac{-4\theta^2}{G_D N\Delta} \tag{7}$$

Where:

- B_{Loop} is the normalized loop bandwidth
- ζ is the damping factor
- N is the samples per symbol
- G_D is the detector gain (scaling factor for correction)

The filter can be implemented with a simple linear equation:

$$y(n) = G_1 x(n) + G_2 \sum_{k=0}^{n} x(k), \tag{8}$$

or alternatively with a Biquad filter.

4.3 Interpolation Controller

The interpolation controller coordinates when and how the interpolator applies fractional delays. It uses a counter-based mechanism to trigger or *strobe* at appropriate symbol positions. The controller must maintain a specific triggering interval that averages to the symbol rate. At these strobe positions is when the interpolator is signaled and updated.

When timing is optimal and the loop filter output $v(n)$ is zero, the controller should trigger every N samples. The counter decrement value is:

$$W(n) = v(n) + \frac{1}{N}. \tag{9}$$

Resulting in a maximum value of 1 under modulo-1 subtraction of the counter $c(n)$, where wraps of the modulus occur every N subtractions.

The counter $c(n)$ is updated using modulo-1 arithmetic:

$$c(n+1) = (c(n) - W(n)) \bmod 1. \tag{10}$$

A strobe condition occurs when the counter would wrap around:

$$\text{Strobe} = \begin{cases} \text{True}, & \text{if } c(n) < W(n) \\ \text{False}, & \text{otherwise} \end{cases} \tag{11}$$

When a strobe occurs, we update the fractional delay parameter $\mu(n)$:

$$\mu(k) = \frac{c(n)}{W(n)}. \tag{12}$$

This μ value is passed to the interpolator to apply the appropriate fractional delay.

We want to avoid performing timing estimates that span over multiple symbols, which would provide incorrect error signals and incorrect updates for our system. We can avoid this by adding conditions into the TED block. We provide additional structure to the TED block in Figure 9, along with additional logic to help identify how we can effectively utilize our strobe signals. Based on this TED structure, only when a strobe occurs the output error e can be non-zero. Looking downstream, since we are using a PI loop filter only non-zero inputs can update the output, and as a result modify the period of the strobe W. When the system enters steady state, where the PLL has locked, the TED output can be non-zero every N samples.

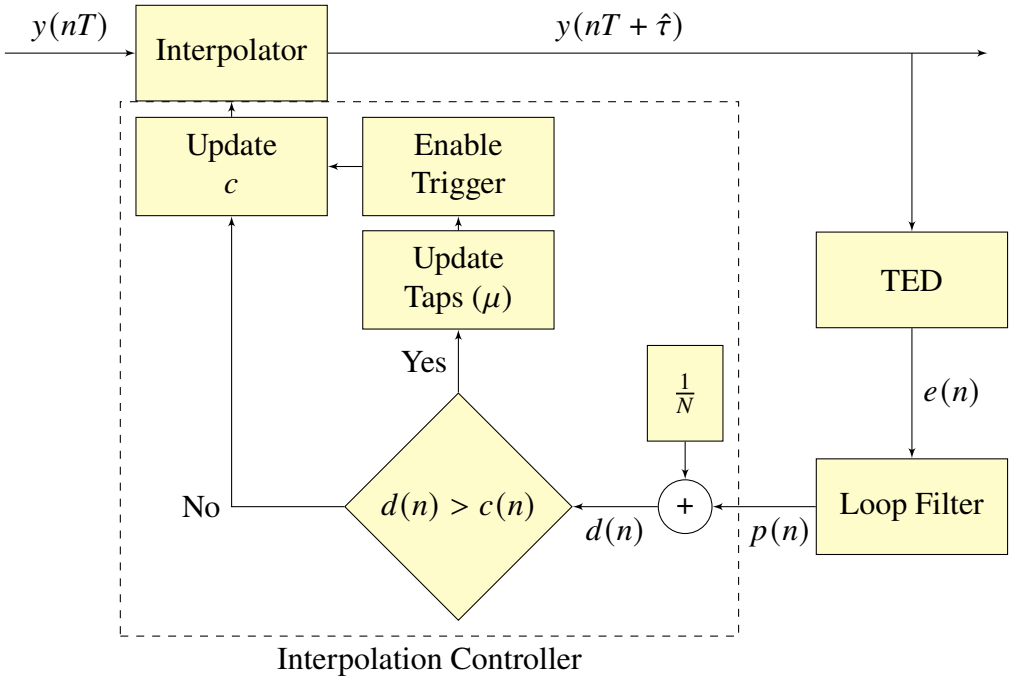

Figure 9: Timing recovery triggering logic used to maintain accurate interpolation of input signal.

4.4 Interpolator Design

The final component is the interpolator, which applies a fractional delay to the input signal. Mathematically, interpolation is a linear combination of current and past input samples. While an ideal interpolator would require an infinite impulse response (IIR) filter [8], practical implementations use finite impulse response (FIR) approximations. Alternative implementations, such as polyphase-filterbank designs, are possible.

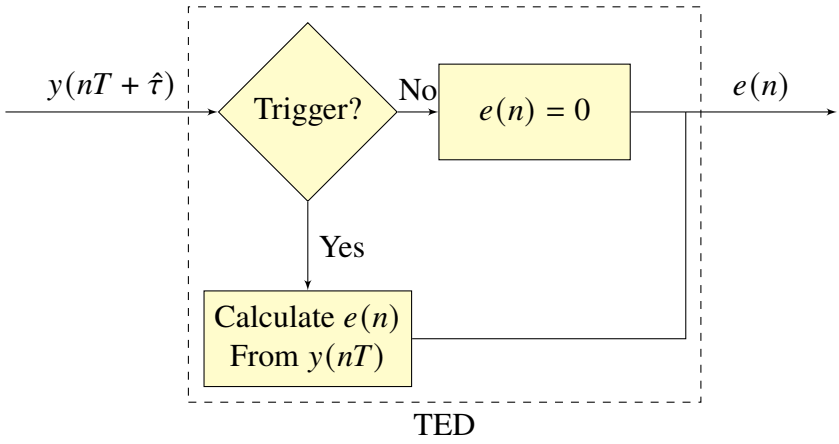

Figure 10: Internal structure of the Timing Error Detector showing how strobes control error and triggering control signals relation to operations of other blocks in Figure 9.

Here we'll use a piecewise polynomial filter (PPF) [7], which efficiently approximates the needed fractional delays. For our implementation, we'll use a quadratic (second-order) interpolation requiring a four-tap filter. The interpolator output is given by:

$$x(kT_s + \mu(k)T_s) = \sum_{n=-2}^{1} h(n)x((k-n)T_s), \tag{13}$$

where the filter coefficients $h(n)$ are determined by [9]:

$$\begin{aligned} h = [&\alpha\mu(k)(\mu(k)-1), \\ &-\alpha\mu(k)^2 - (1-\alpha)\mu(k) + 1, \\ &-\alpha\mu(k)^2 + (1+\alpha)\mu(k), \\ &\alpha\mu(k)(\mu(k)-1)], \end{aligned} \tag{14}$$

with $\alpha = 0.5$ and $\mu(k)$ representing the fractional delay from the interpolator controller. The estimated timing offset $\hat{\tau}$ relates to $\mu(k)$ as:

$$\hat{\tau} = \mu(k)T_s. \tag{15}$$

When $\mu = 0$ (no offset), the interpolator acts as a two-sample delay (one symbol delay for the ZC implementation).

The quadratic interpolator uses four samples rather than three, which might seem counterintuitive. However, odd-length filters are suboptimal for finding values between samples, and a two-sample implementation would not adequately capture the curvature visible in the eye diagram (Figure 8).

4.5 Implementation Guidelines

The timing recovery components operate at different relative rates. Table 1 provides recommended operational rates that align with the strobe-based implementation in Figure 9. When properly implemented, this system produces one output sample per symbol when the interpolator outputs align with strobe signals.

Table 1: Operational Rates of Timing Recovery Blocks

Block	Operational Rate
Interpolator	Sample Rate
TED	Symbol Rate
Loop Filter	Symbol Rate
Interpolator Controller	Sample Rate

4.6 Questions

1. Starting with `TimingError.m`, which demonstrates a slowly changing timing offset, implement in MATLAB one of the timing correction algorithms (ZC, MM, or Gardner).
2. Provide error vector magnitude (EVM) measurements of the signal before and after your timing correction at different SNR levels (e.g., 5dB, 10dB, 15dB, 20dB). Use the `comm.EVM` object in MATLAB to calculate EVM with a reference constellation.
3. Introduce a fixed phase offset of $\pi/8$ radians and repeat your EVM measurements. Explain how the phase offset affects timing recovery performance.

Implementation Guidance

- It is suggested to use the `comm.SymbolSynchronizer` object available in MATLAB's Communications Toolbox. Alternately, MATLAB code from the textbook is also included in the supporting materials to help implement your own loop filter.
- If implementing your own loop filter, a recommended starting configuration for your own loop filter should be:
  - $\left[N, \zeta, B_{Loop}, G_D\right] = \left[2, 1, 0.1, 2.7\right]$.
- To help with troubleshooting:
  - Start with noise-free signals to confirm basic functionality
  - Gradually introduce noise sources once your implementation is working
  - Visualize the error signal $e(n)$ over time to verify convergence
  - Monitor the interpolation parameter $\mu(k)$ to ensure it stabilizes
- For interpolation, the `interp1` function with cubic interpolation provides a good starting point before implementing the PPF approach described in Section 4.

5 Adding Pieces Together

Throughout this laboratory, we have outlined the structure and logic behind a PLL-based timing recovery algorithm and the associated MATLAB code. In the remaining laboratories, we will discuss putting the algorithmic components together and provide some intuition on what happens during evaluation. Here we will address parameterization and the relation to system dynamics. The system-level scripts have shown a constant theme throughout where data is: modulated, transmit filtered, passed through a channel with timing offset, recieve filtered, then is timing recovered.

Many rate changes can occur in this series of steps. To help understand these relations better, we can map things out as in Figure 11, which takes into account these stages. Here the modulator produces symbols equal to the sample rate, T_s. Once passing through the transmit filter, we acquire our upsampling factor N, which increases our samples per symbol to N. At the receiver, the receive filter performs decimation by a factor N_F where $N_F \leq N$. Finally, the timing recovery block compensates for the the fractional offset τ and returns to the original rate of one sample per symbol.

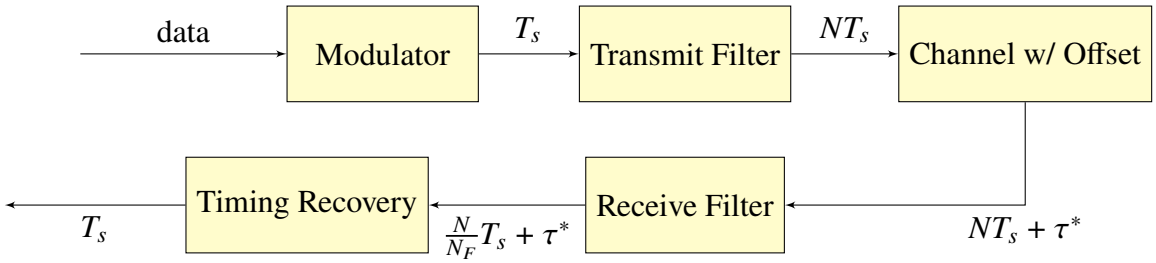

Figure 11: Relative rates of transmit and receive chains with respect to the sample rate at different stages. Here τ^* represents a timing shift, not an increase in the data rate.

5.1 Questions

1. Provide your chosen arrangement of recovery blocks and reasoning behind their selected placement. Include a discussion of any decimation or interpolation factors used and the respective samples-per-symbol.
2. With your new receive chain, generate a received signal (before the receive filter in Figure 3) under the following conditions and provide EVM measurements after timing recovery:
   (a) No offsets (ideal baseline case)
   (b) A fixed timing offset of $0.25T_s$
   (c) A fixed phase offset of $\pi/4$ radians

   Provide results over a range of SNR values.

6 Automatic Timing Compensation With Pluto

6.1 Objective

The objective of this problem is to design and implement a software-defined radio (SDR) communication system capable of automatically calculating and correcting the timing offsets between two Pluto SDRs.

In Section 4, you implemented and simulated timing correction. Now we will introduce the PlutoSDR to deal with realistic timing behavior. To simplify this process, a set of required tasks are staged to gradually increase the difficulty of the overall implementation. We have presented three methods for timing correction, but you are free to use these in any arrangement you want, or use your own algorithms. However, you must provide an evaluation of the overall system performance.

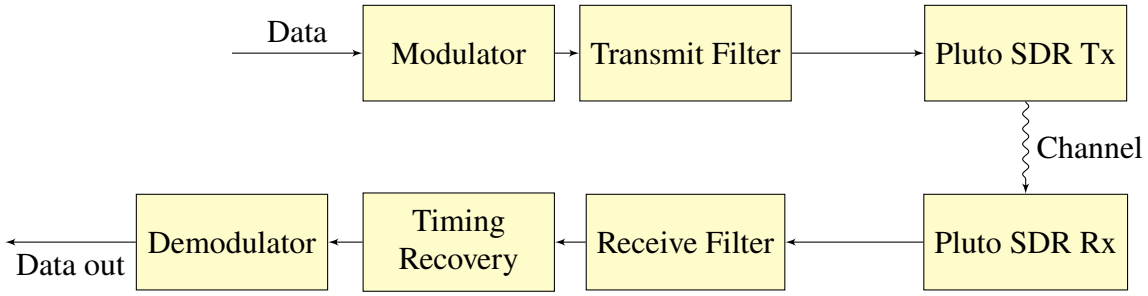

Figure 12: Complete SDR system with timing synchronization for Pluto SDR testing.

6.2 Testing on Hardware

With your implemented timing correction code, perform the following tasks:

1. First, using a single PlutoSDR in loopback configuration, transmit your DBPSK or QPSK reference signal. This configuration will provide minimal to zero frequency offset, allowing you to focus on phase and timing compensation. Graph the resulting baseline Error Vector Magnitude (EVM) and constellation diagram before and after timing correction.
2. **Optional:** Using a pair of PlutoSDRs, repeat this estimation as previously performed with a single PlutoSDR. Provide EVM measurements and constellation diagrams at different distances and/or different gain settings for the radios.
3. Compare your hardware results with the simulation results obtained in Section 5. Discuss:
    - Key differences between simulation and hardware performance
    - Unexpected challenges encountered with the hardware implementation
    - Any modifications required to your algorithms when moving from simulation to hardware

7 Lab Report Preparation & Submission Instructions

Laboratory reports should be uploaded as PDF or Word documents outside of a zip archive and formatted as follows:

- Cover page: includes course number, laboratory title, name, submission date.
- Table of Contents, List of Tables, List of Figures. (Optional)
- Lab Results:
  - Introduction to the laboratory experiment, including a brief description of the objectives and goals.
  - Detailed explanation of the laboratory experiment, including the design, implementation, and testing of the system.
  - Results and discussion of the laboratory experiment, including captured outputs, observations, and responses to laboratory questions.
  - Conclusions to the overall lab that discuss meaningful lessons learned and other takeaways from the assignment. (Important)
- Upload source files with report submission. You may additionally include select code source listings in your report to highlight techniques used.
  - Note: Python files autogenerated from GNURadio do not need to be uploaded if the .grc files are included.

Remember to write your laboratory report in a detailed and descriptive manner, explaining your experience and observations in such a way that it provides the reader with some insight as to what you have accomplished. Furthermore, please include images and outputs wherever possible in your laboratory report document.

References

[1] J. G. Proakis and M. Salehi, *Digital Communications 5ed*. McGraw-Hill, 2008.

[2] A. A. M. Saleh, "Frequency-independent and frequency-dependent nonlinear models of twt amplifiers," *IEEE Transactions on Communications*, vol. 29, no. 11, pp. 1715–1720, November 1981.

[3] S. Boumaiza, T. Liu, and F. M. Ghannouchi, "On the wireless transmitters linear and nonlionear distortions detection and pre-correction," in *2006 Canadian Conference on Electrical and Computer Engineering*, May 2006, pp. 1510–1513.

[4] U. Mengali, *Synchronization Techniques for Digital Receivers*, ser. Applications of Communications Theory. Springer US, 1997. [Online]. Available: https://books.google.com/books?id=89Gscsw7PvoC

[5] F. Gardner, "A bpsk/qpsk timing-error detector for sampled receivers," *IEEE Transactions on Communications*, vol. 34, no. 5, pp. 423–429, May 1986.

[6] K. Mueller and M. Muller, "Timing recovery in digital synchronous data receivers," *IEEE Transactions on Communications*, vol. 24, no. 5, pp. 516–531, May 1976.

[7] M. Rice, *Digital Communications: A Discrete-Time Approach*. Upper Saddle River, New Jersey: Pearson/Prentice Hall, 2009.

[8] J. P. Thiran, "Recursive digital filters with maximally flat group delay," *IEEE Transactions on Circuit Theory*, vol. 18, no. 6, pp. 659–664, Nov 1971.

[9] L. Erup, F. M. Gardner, and R. A. Harris, "Interpolation in digital modems. ii. implementation and performance," *IEEE Transactions on Communications*, vol. 41, no. 6, pp. 998–1008, Jun 1993.

Acknowledgement

This laboratory assignment is based on material provided by the *Software-Defined Radio for Engineers* book course material which was previously taught at Worcester Polytechnic Institute.