# Lab 5 - Digital Modulation: Symbol Synchronization


**ECE531 – Software Defined Radio**

**Spring 2022**

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 which have different performance objectives. Moreover, MATLAB® will also be used as development tool for digital communication systems, especially the construction of a prototype software-defined radio (SDR) based simulation.

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

# 2 Pulse Shaping and Matched Filtering

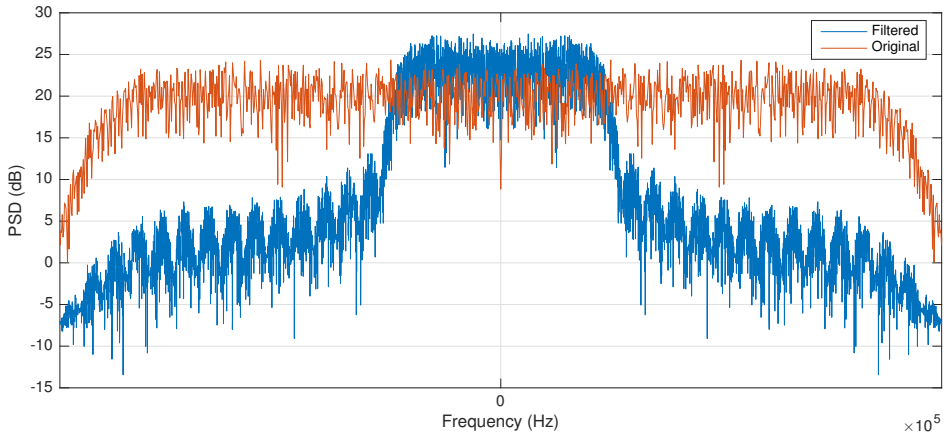


Figure 1: Frequency spectrum of PSK signal before and after pulse shaping.

The topics of matched filters and transmit filters are introduced here to discuss how they are used in practices and demonstrate effects they help alleviate. In digital communications theory when matched filtering is discussed it is typically called pulse shaping at the transmitter and matched filtering at the receiver for reference. The goal of these techniques is three fold: first to make the signal suitable to be transmitted through the communication channel mainly by limiting its effective bandwidth, increase the SNR of the received waveform, and to reduce intersymbol interference (ISI) from multi-path channels and nonlinearities.

By filtering a symbols sharp phase and frequency transitions are reduced, resulting in a more compact and spectrally efficient signal. Figure 1 provides a simple example of a DBPSK signal's frequency representation before and after filtering with a transmit filter. The filter used to generate this figure was a square-root raised cosine (SRRC) filter, which is a common filtered use in communication system. Details of the raised cosine filter are provided in Chapter 2 (§2.7.5), but the

more common 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.

The SRRC is used since it is a Nyquist type filter, which produces zero ISI when sampled correctly [1]. We can demonstrate the effect of ISI by introducing a simple nonlinearity into the channel and consult the resulting eye diagrams which were introduced in Section 2.4.1. Nonlinearities cause amplitude and phase distortions, which can happen when we clip or operate at the limits of our transmit amplifiers. For more details on the used model consult [2], but other models exists such as [3]. In Figure 2 of observe the effects of ISI as the eye becomes compressed and correct sampling becomes difficult to determine.

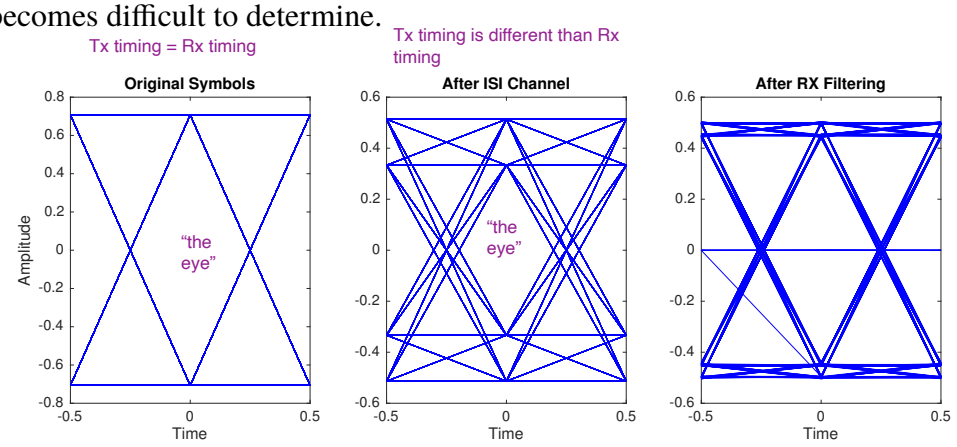


Figure 2: Eye diagrams of QPSK signal effected by nonlinearity causing ISI, which is reduced by SRRC matched filtering.

The final aspect of the matched filters we want to discusses or provide insight into is SNR maximization. This argument logically comes out of the concept of correlation. 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. We demonstrate this effect in Figure 3, where we present data transmitted with and without pulse shaping under AWGN. In the middle plot we observe a signal closely related to the originally transmitted sequence, even under high noise. However, without pulse shaping even visually the evaluation of the transmitted pulse becomes difficult. We even observe demodulation errors in this third plot without anytime timing offset introduced.

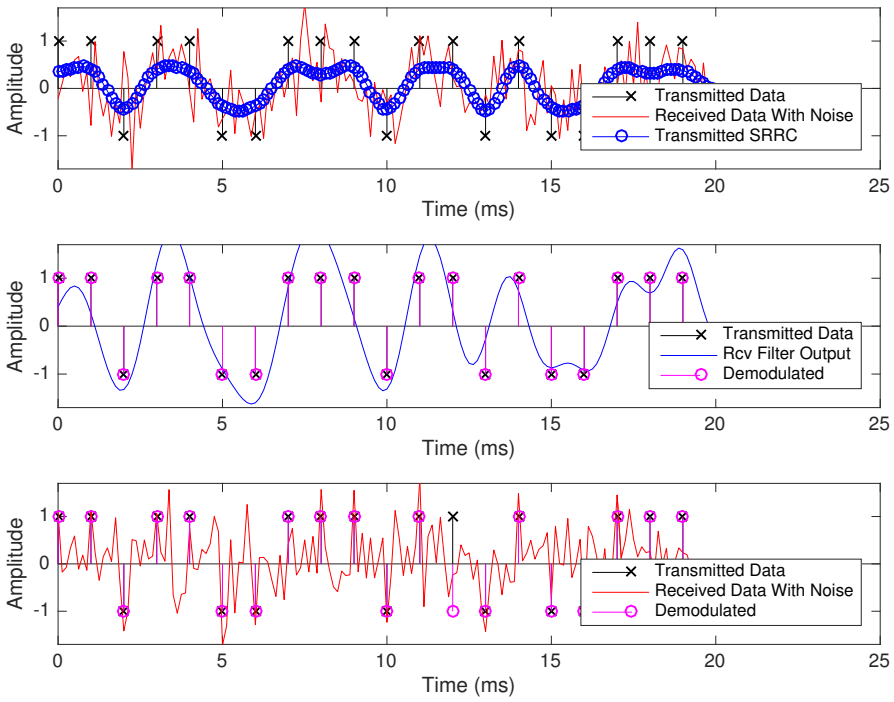


Figure 3: Comparison of pulse shaping and non-pulse shaping received signals under AWGN. (a) Transmitted SRRC filtered Signal (b) received SRRC filtered signal, (c) received signal without SRRC filtering at receiver.

## 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 $\beta \in \left[0, 0.1, 0.25, 0.5, 1\right]$.
3. Generate transmit and receive plots similar to Figure 3 with $\beta \in \left[0.1, 0.5, 0.9\right]$ and discuss the time domain effects.
4. Compare and contrast the effects up-sample factor (samples per symbol) on the design of a communication system.

# 3 Timing Error

In the most basic sense the purpose of symbol timing synchronization is the align the clocking signals or sampling instances of two disjoint communicating devices. In Figure 4 we consider a simple example where we overlap the clocking signal and input signal to be sampled. The sampling occurs at the rising clock edges, and the optimal timing is provided. However, in a real-world environment there will some unknown non-zero delay or shift of the clocking signal which can result non-optimal sampling or even wrong decisions.

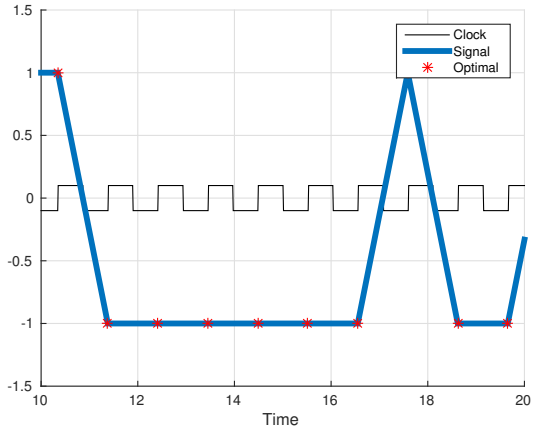


Figure 4: Example received symbols and associated optimal clocking.

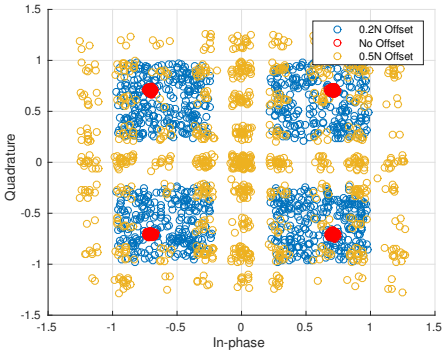


Figure 5: Simulation example of timing offset effects on QPSK constellation.

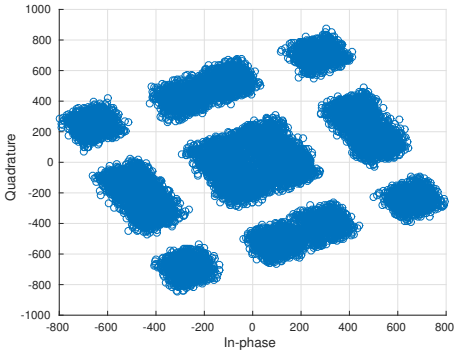


Figure 6: Example with data sent through Pluto using a loopback cable.

In the case of our wireless transmissions, if we consider this error from the perspective of the constellation diagram we will observe clustering or scattering of the symbols. In Figure 5, we provide a simulated timing offsets of $0.2N$ and $0.5N$, where $N$ is the samples per symbol. An offset of $0.5N$ is the worse case because we are exactly between two symbols. In Figure 6 we provide a QPSK transmitted through loopback of a single Pluto SDR. We can clearly observe a similar clustering.

Mathematically we can model this offset input signal which has passed through the receive matched filter as:

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

where $p$ is the autocorrelation of the pulse shape use on the source data $x$, and $v$ is the shaped noise. The unknown delay $\tau$ must be estimated to provide correct demodulation downstream.

### 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

There are many ways to perform correction for symbol timing mismatches between transmitters and receivers. However, here we will consider three digital phase lock loop (PLL) strategies for our carrier recovery implementations. This type of timing recovery was chosen because it can be integrated with our existing recovery solutions. For the PLL outlined in Figure 7, we will first provide an overview conceptually how timing is synchronized, and then move into each individual block, explaining their design. The detectors discussed will be: zero-crossing, Müller and Mueller, and Gardner.

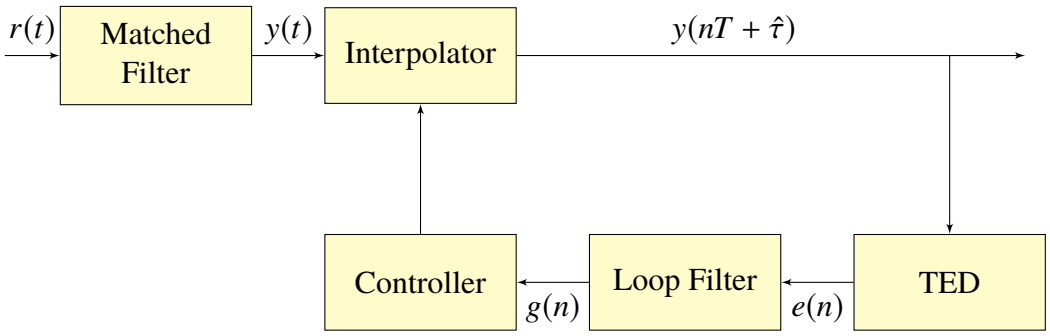


Figure 7: Basic structure of PLL for timing recovery.

The timing synchronization PLL used in all three algorithms consists are four main blocks: interpolator, timing error detector (TED), loop filter, and an interpolator controller. This all-digital PLL-based algorithm shown here works by first measuring an unknown offset error, scale the error proportionally, and apply an update for future symbols to be corrected. To provide necessary perspective on the timing error, let us considered the eye diagram presented in Figure 8. This eye diagram has been up-sampled by twenty times so we can examine the transitions more closely, which we notice are smooth unlike Figure 2. In the optimal case, we chose to "sample" our input signal at the red instances at the widest openings of the eye. However, there will be an unknown fractional delay $\tau$ which shifts this sampling period. This shifted sampling position is presented by the green selections. To help work around this issue, our receive signal is typically not decimated fully, providing the receiver with multiple samples per symbol (This is not always the case). Therefore, if we are straddling the optimal sampling position instead as in the black markers. We can simply interpolate across these points to get our desired period. This interpolation has the effect of causing a fractional delay to our sampling, essentially shifting to a new position in our eye diagram. Since $\tau$ is unknown we must weight this interpolation correctly so we do not overshoot or undershoot the desired correction.

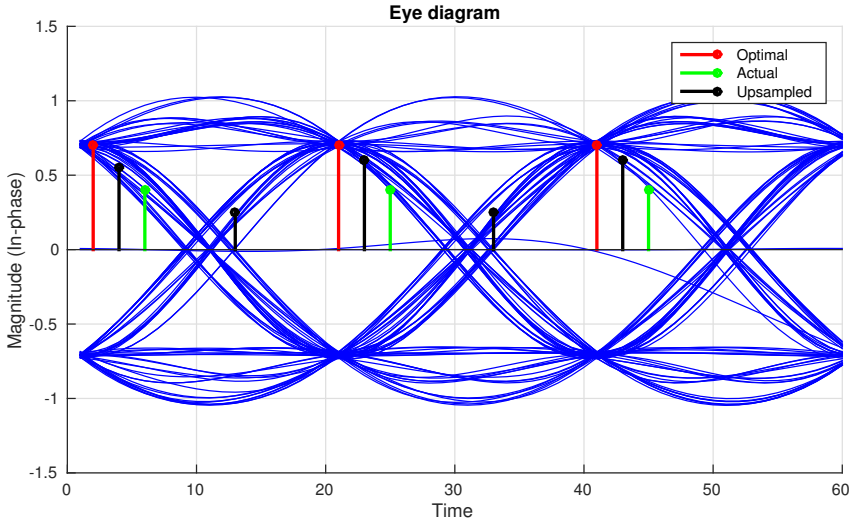


Figure 8

We will initially discuss the specifics of the blocks in Figure 7 through the perspective of the zero-crossing (ZC) method, since it is the most straight forward to understand. Then we will provide extensions to the alternative methods. In ZC as the name suggests, will produce an error signal of zero when one of the sampling positions is at the zero intersection. ZC requires two samples per symbol, resulting in the other sampling position occurring at or near the optimal position. The TED for ZC [4] is evaluated as:

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

Note that these indexes are with respect to symbols, not samples. Equation (3) first provides a direction for the error with respect to the $sgn$ operation, and the shift required to compensate is determined by the midpoints. The in-phase and quadrature portions operate independently, which is desirable. Once the error is calculated it is passed to the loop filter:

$$\theta = \frac{B_{Loop}}{M(\zeta + 0.25/\zeta)} \qquad \Delta = 1 + 2\zeta\theta + \theta^2 \tag{4}$$

$$G_1 = \frac{-4\zeta\theta}{G_D N \Delta} \qquad G_2 = \frac{-4\theta^2}{G_D N \Delta} \tag{5}$$

Here $B_{Loop}$ is the normalized loop bandwidth, $\zeta$ is our damping factor, $N$ is our samples per symbol, and $G_D$ is our detector gain. The new variable $G_D$ provide an additional step size scaling to our correction. Again the loop filter's purpose is maintain stability of the correction rate. This filter can be implemented with a simple linear equation:

$$y(n) = G_1 x(n) + G_2 \sum_{k=0}^{n} x(k), \tag{6}$$

or with a Biquad filter.

The next block to consider is the interpolation controller, which is responsible to providing the necessary signaling to the interpolator. Since the interpolator is responsible for fractionally delaying the signal, this controller must provide this information and generally the starting interpolant sample. By starting interpolant sample we are referring to the sample on the left side of the straddle, as shown by the second black sampling position from the left in Figure 8. The interpolation controller implemented here will utilize a counter-based mechanism to effectively trigger or *strobe* at the appropriate symbol positions. At these strobe positions is when the interpolator is signaled and updated.

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. If we consider the case when the timing is optimal and the output of the loop filter $v(n)$ is zero, we would want to produce a trigger every $N$ samples. Therefore, it is logical that the weighting or decrement for the counter would be:

$$W(n) = v(n) + \frac{1}{N}. \tag{7}$$

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. This modulus counter update is defined as:

$$c(n+1) = (c(n) - W(n)) \mod 1. \tag{8}$$

We determine a strobe condition, which is checked before the counter is updated, based on when these modulus wraps occur. We can easily check for this condition before the update such as:

$$Strobe = \begin{cases} c(n) < W(n) & True \\ \text{Otherwise} & False \end{cases}. \tag{9}$$

This strobing signal is the method used to define the start of a new symbol, therefore it can also be used to make sure we are estimating error over the correct samples. When the strobe occurs we will updated $\mu(n)$ our estimated gap between the interpolant point and the optimal sampling position. This update is a function of the new counter step $W(n)$ and our current count $c(n)$:

$$\mu(k) = c(n)/W(n). \tag{10}$$

This $\mu$ will be passed to our interpolator to update the delay it applies.

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.

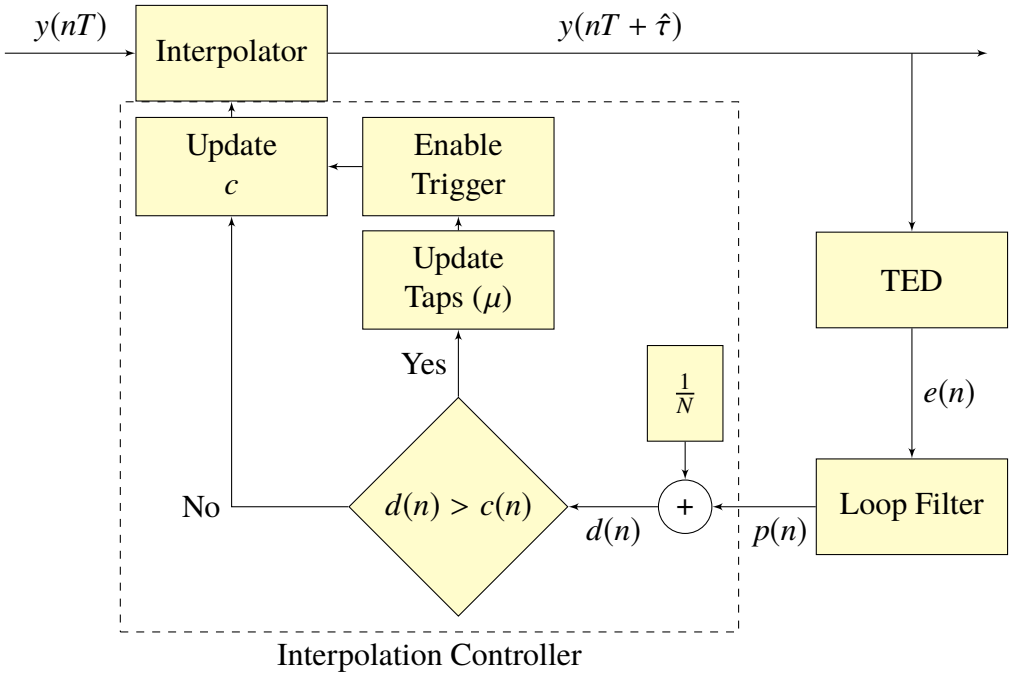


Figure 9: Timing recovery triggering logic used to maintain accurate interpolation of input signal.

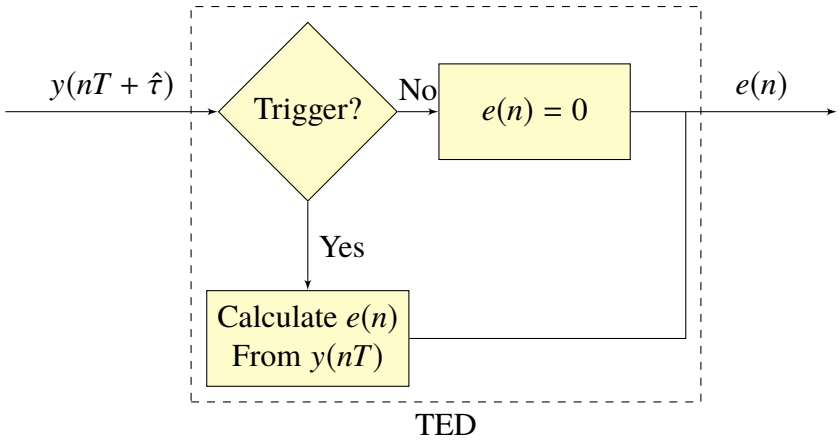


Figure 10: An internal view of the timing error detector to outline the error and triggering control signals relation to operations of other blocks in Figure 9

The final piece of the timing recovery we have not yet discussed is the interpolator itself. Interpolation is simply a linear combination of the current and past inputs $x$, which in essence can be though of as a filter. However, to create a FIR filter with any arbitrary delay $\tau \in [0, ..., T_s]$ cannot be realized [5]. Realizations for ideal interpolation IIR filters do exist, but the computation of their taps are impractical in real systems [6]. Therefore, we will use an adaptive implementation of a FIR lowpass filter called a piecewise polynomial filter (PPF) [7]. The PPF can only provide estimations of offsets to a polynomial degree. Alternative implementations exists such as polyphase-filterbank

designs, but depending on the required resolution the necessary phases become large.

The PPF are useful since 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. Here we will use a second order, or quadratic, interpolation requiring a four tap filter. The general form of the interpolator's output is given by:

$$x(kT_s + \mu(k)T_s) = \sum_{n=-2}^{1} h(n)x((k-n)T_s), \tag{11}$$

where $h_k$ are the filter coefficients at time instance $k$ determined by [8]:

$$\begin{aligned} h = &[\alpha\mu(k)(\mu(k)-1), \\ &-\alpha\mu(k)^2 - (1-\alpha)\mu(k) + 1, \\ &-\alpha\mu(k)^2 + (1+\alpha)\mu(k), \\ &\alpha\mu(k)(\mu(k)-1)], \end{aligned} \tag{12}$$

where $\alpha = 0.5$. $\mu(k)$ is the fractional delay which is provided by the interpolator control block, which relates the symbol period $T_s$ to the estimated offset. Therefore, we can estimate the true delay $\tau$ as:

$$\hat{\tau} = \mu(k)T_s. \tag{13}$$

Without any offset ($\mu = 0$), the interpolator acts as a two sample delay or single symbol delay for the ZC implementation. We can extend the PPF to utilize more samples creating cubic and greater interpolations, but there implementations by become more complex. The underlying waveform should be considered when determining the implementation of the interpolator, and the required degrees of freedom to accurately capture the required shape.

This design using four samples in a quadratic form can be considered irregular, since the degree of taps does not reach 3. However, odd length realizations (using an odd number of samples) are not desirable since we are trying to find values in-between the provided samples. We also do not want a 2 sample implementation due to the curvature of the eye in Figure 8.

The overall design of the synchronizer can be complex and implementations do operate at different relative rated. Therefore, we have provided Table 1 on guide of a recommended implementation. These rates align with the strobe implementation outlined in Figure 9. This system will result in one sample per symbol when output samples of the interpolator are aligned with the strobes.

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 |

## 4.1 Gardner

The second TED we will considered is called Gardner [9], which is very similar to ZC. The error signal is determined by:

$$\begin{aligned} e(k) = Re(x((k-1/2)T_s+\tau))\Big[Re(x((k-1)T_s+\tau)) - Re(x(kT_s+\tau))\Big] + \\ Im(x((k-1/2)T_s+\tau))\Big[Im(x((k-1)T_s+\tau)) - Im(x(kT_s+\tau))\Big]. \end{aligned} \tag{14}$$

This method also requires two samples per symbol and differs only in the quantization of the error direction from ZC. One useful aspect of Gardner is that it does not require carrier phase correction and works specifically well with BPSK and QPSK signals. However, since Gardner is not a decision directed method for best performance the excess bandwidth of the transmit filters should be $\beta \in (0.4, 1)$.

## 4.2 Müller Mueller

Next is the Müller and Mueller (MM) method named after Kurt Mueller and Markus Müller [10]. This can be considered the most efficient method since it does not require upsampling of the source data, operating at one sample per symbol. The error signal is determined by [7]:

$$\begin{aligned} e(k) = & Re(x((k)T_s+\tau))sgn\{Re(x((k-1)T_s+\tau))\} \\ & - Re(x((k-1)T_s+\tau))sgn\{Re(x((k)T_s+\tau))\} \\ & + Im(x((k)T_s+\tau))sgn\{Im(x((k-1)T_s+\tau))\} \\ & - Im(x((k-1)T_s+\tau))sgn\{Im(x((k)T_s+\tau))\} \end{aligned} \tag{15}$$

MM also operates best when the matched filtering used minimizes the excess bandwidth. When the excess bandwidth is high the decisions produce by the $sgn$ operation can be invalid.

## 4.3 Questions

1. Starting with `TimingError.m`, which demonstrates a slowly changing timing offset, implement in MATLAB either the ZC, MM, or Gardner timing correction algorithm.
2. Provide error vector magnitude (EVM) measurements of the signal before an after your timing correction at different noise levels. (You may use the `comm.EVM` object in MATLAB.)
3. Introduce a fixed phase offset and repeat your EVM measurements from Question 1.

## Implementation Guidance

- You may elect to use the included code to help implement the loop filter and/or the `comm.SymbolSynchronizer` object in MATLAB.
- A recommended starting configuration for your loop filter should be $\left[N, \zeta, B_{Loop}, G_D\right] = \left[2, 1, 0.1, 2.7\right]$.
- Remove noise sources and then reintroduce them once your implementation becomes functional.
- Bit stuffing/removal is not discussed, but is simple to add. These are required when you receive two strobes one after the other at the sample rate, which should not occur. In this case introduce an extra zero into your TED input buffer, forcing an error calculation across at least one zero. This should not really happen in simulation since there is no jitter, but for more information consult [7, pg. 491].

# 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 also 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, filtered again, then is timing recovered. Many rate changes can happen 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. Once passing through the transmit filter we acquire our upsampling factor $N$ , which increases our samples per symbol to $N$ . At the receiver we can perform decimation in the receive filter, by a factor $N_F$ where $N_F \leq N$ . Finally, we will perform timing recovery across the remaining samples and remove the fractional offset $\tau$ , returning to the original rate of one sample per symbol.

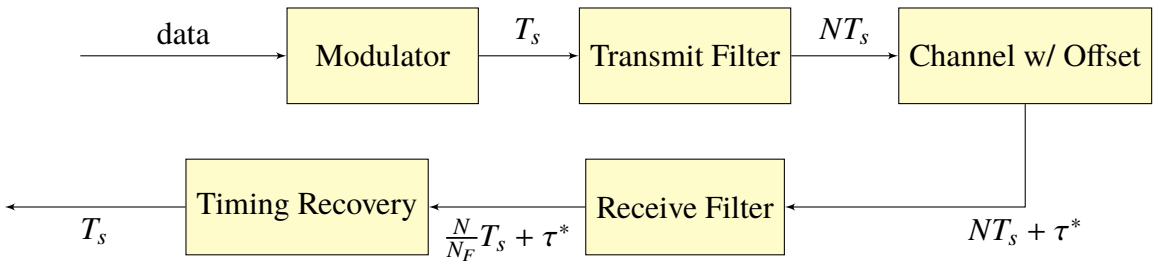


Figure 11: Relative rates of transmit and receive chains with respect to the sample rate at different stages. Here $\tau^*$ represents a timing shift not an increase in the data rate. This is a slight abuse of notation.

## 5.1 Questions

1. Provide your chosen arrangement of recovery blocks and reasoning behind their selected placement.
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
   (b) A fixed timing offset
   (c) A fixed phase offset

   Provide results over a small range of SNRs.

# 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 Sections 4 you have implemented and simulated timing correction. Now we will introduce the Pluto 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.

## 6.2 Testing on Hardware

With your built frequency correction code perform the following tasks:

1. First, using a single Pluto transmit your DBPSK or QPSK reference signal across to the same radio. Using a loopback configuration will provide you will minimal to zero frequency offset. Therefore, you will only have to compensate for phase and timing. Graph the resulting baseline Error vector magnitude (EVM).
2. Repeat this experiment by increasing the center frequency of the transmit or receive chain. Graph the resulting EVM.
3. **OPTIONAL:** Using a pair of Plutos repeat this estimation again as previously performed with a single Pluto. Provide EVM measurements at different distances and/or different gain setting for the radio.
4. Compare your results here from what you have obtained in Sections 5 in a way you determine as appropriate.

# 7 Lab Report Preparation & Submission Instructions

Include all your answers, results, and source code in a laboratory report formatted as follows:

- Cover page: includes course number, laboratory title, name, submission date.
- Suggested: Table of contents, list of tables, list of figures.
- Commentary on designed implementations, responses to laboratory questions, captured outputs, and explanation of observations.
- Meaningful conclusions to the lab.
- Upload source files with report submission. You may also list select code source in your report appendix. Note: Python files autogenerated from GNURadio do not need to be uploaded in additon to .grc files.

Remember to write your laboratory report in a descriptive approach, 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] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay [fir/all pass filters design]," *IEEE Signal Processing Magazine*, vol. 13, no. 1, pp. 30–60, Jan 1996.

[6] 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.

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

[8] 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.

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

[10] 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.

# 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.