Lab 6 - Digital Modulation: Carrier 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 carrier frequency offset between transmitting and receiving nodes. Specifically, a simplified error model will be discussed along with two recovery methods which can operate jointly or independently, based on their implementation. Carrier recovery complements timing recovery, which was implemented in Lab 5, and is necessary for maintaining wireless links between radios with independent oscillators.

Note: The MATLAB Communication Systems Toolbox implements the coarse frequency correction and fine frequency correction algorithms developed in this lab and Chapter 7 with `comm.CoarseFrequencyCompensator` and `comm.CarrierSynchronizer` respectively. FFT-based and correlation-based coarse frequency compensation methods are available. Fine frequency compensation is compatible with BPSK, QPSK, OQPSK, 8-PSK, PAM, and rectangular QAM modulation schemes. *Hint: Linked Matlab documentation can be very helpful for solving the lab.*

2 System and Error Models

Throughout this Lab we will assume that timing mismatches between the transmitting and receiving radios have already been corrected. However, this is not a requirement in all cases, specifically in the initial implementation provided here, but will become a necessary condition for optimal performance of the final implementation provided. For the sake of simplicity we will also ignore timing effects in our simulations except when discussing PlutoSDR itself, since timing correction cannot be overlooked in that case. With regards to our full receiver diagram outlined in Figure 1, we are now considering the Carrier Recovery and carrier frequency offset (CFO) blocks.

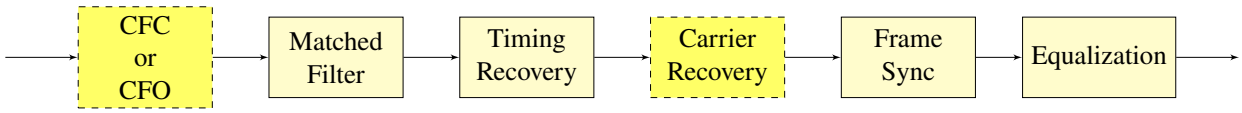

Figure 1: Receiver Block Diagram.

The receiving and transmitting nodes are generally two distinct and geographically separate units. Therefore, there will exist relative frequency offsets between their local oscillators (LOs) due to natural effects. This offset can contain random phase noise, frequency offset, frequency drift, and initial phase mismatches. However, for simplicity we will only model this offset as a fixed value.

When considering commercial oscillators, the frequency offset is provided in parts per million (PPM), which we can translate into a maximum carrier offset. In the case of the Pluto, the internal LO is rated at ±25 PPM [1] and we can use Equation (1) to relate maximum carrier offset Δf to our operating carrier frequency f_c.

$$f_\Delta = \frac{f_c \times \text{PPM}}{10^6} \tag{1}$$

The determination of f_Δ is important because it provides our carrier recovery design criteria and avoids wasting resources on a capability to correct for frequencies beyond our operational range.

Mathematically we can model a corrupted source signal s_k with a carrier frequency offset of f_o (or ω_o) as:

$$r(k) = s(k)e^{j2\pi f_o kT+\theta} + n(k) = s(k)e^{j\omega_o kT+\theta} + n(k). \tag{2}$$

where $n(k)$ is a zero-mean Gaussian random process, T is the symbol period, and θ is the carrier phase.

In MATLAB, open the provided script `lab6part1.m`, which provides the transmit side of Figure 1 along with a fixed carrier offset model. This script will provide a scope of the Power Spectral Densities (PSDs) of our signals of interest, which will be similar to Figure 2. For large offsets, it can be useful to examine the signal through a frequency plot to provide perspective on the relative offset.

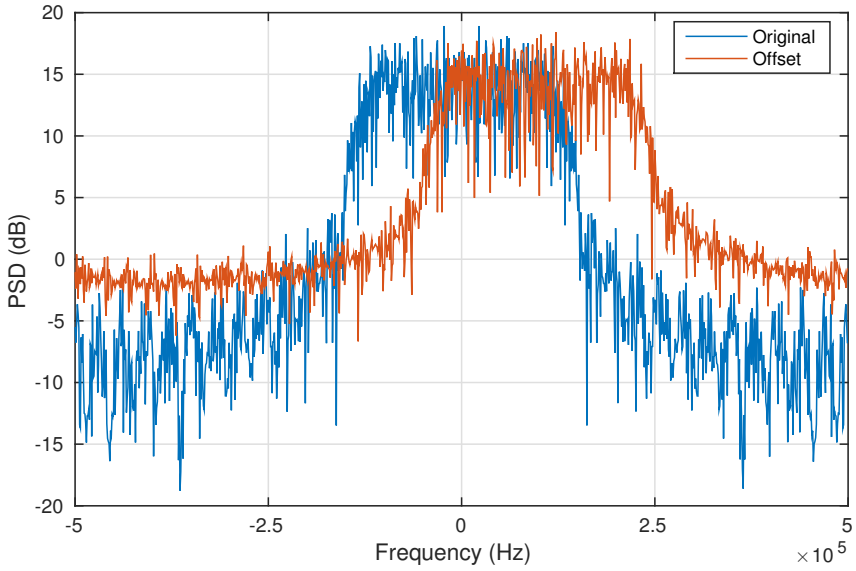

Figure 2: PSDs of offset and original transmitted signals.

2.1 Questions

1. Change 'filterUpsample' in `lab6part1.m` to 1 and observe the spectrum. Explain what you observe.
2. With the original `lab6part1.m` script increase the frequency offset in unit steps of $0.1F_s$, where F_s is the sample rate, from $0.1F_s$ to $1.0F_s$. Explain the observed effect.
3. When applying the frequency offset in `lab6part1.m` what is the reasoning behind incrementing the time vector?
4. Besides LO mismatches between transmitter and receiver, what are other possible sources of frequency offset?

3 Frequency Offset Compensation

There are many different ways to design a wireless receiver, using many different recovery techniques and arrangement of algorithms. In this lab we will consider frequency offset first and then proceed to manage the remaining synchronization tasks. As discussed in Section 2, the Pluto's oscillator is rated at ±25 PPM. Transmitting signals in an unlicensed band, such as 2.4 GHz, can produce a maximum offset of approximately 120 kHz between the radios. Since this is quite a large range, we will develop a two-stage frequency compensation technique separated into coarse and fine frequency correction. This design is favorable because it can reduce convergence or locking time for estimation of the relative carrier.

3.1 Coarse Frequency Correction

There are two primary categories of coarse frequency correction in the literature: data-aided (DA) and blind correction. DA techniques utilize correlation structures that use knowledge of the received signal, usually in the form of a preamble, to estimate the carrier offset f_o. Although DA methods can provide accurate estimates, their performance is generally limited by the length of the preambles [2], and as the preamble length is increased, this decreases system throughput.

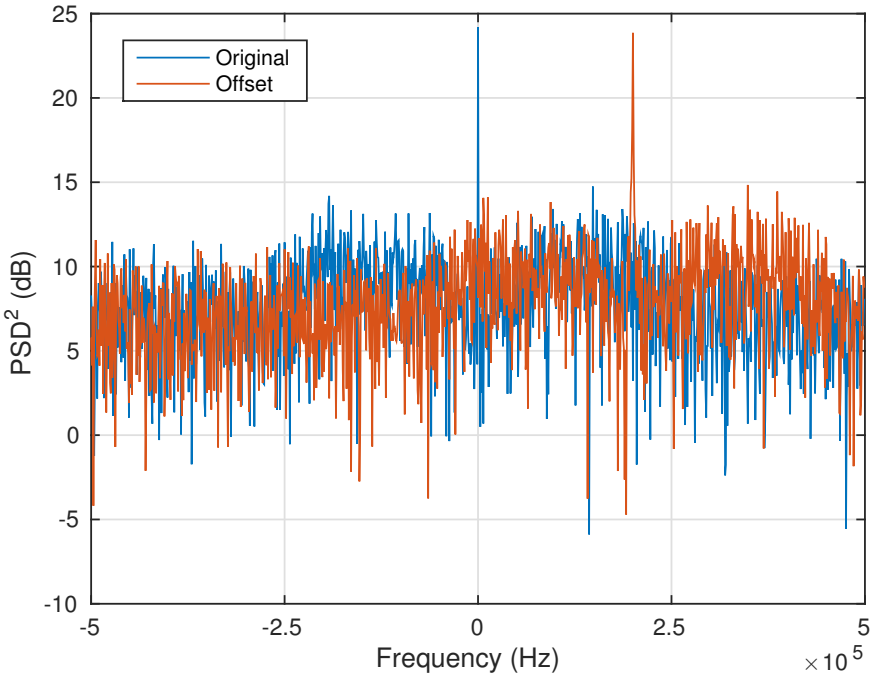

Figure 3: PSDs squared of offset and original transmitted signals.

Alternatively, blind or non-data-aided (NDA) methods can operate over the entire duration of the signal. Therefore, it can be argued that in a realistic system, NDA will outperform DA algorithms.

In this lab, we will implement an NDA FFT-based technique for coarse compensation. The concept is straightforward: based on our initial inspection provided in Figure 2, we can provide a rough estimate of the symbol offsets. To make this more accurate for BPSK/DBPSK, we can simply square the received signal, which produces Figure 3. This squaring operation creates clearly visible peaks at twice the original offset. The exact algorithm is taken from [4], and f_o can be estimated directly for our BPSK/DBPSK system as follows:

$$\hat{f}_o = \frac{1}{2TK} \underset{f}{\text{argmax}} \left| \sum_{k=0}^{K-1} r^2(k) e^{-j2\pi kT/K} \right| \tag{3}$$

where K is the FFT length. The estimation in Equation (3) is defined as coarse since the resulting $\hat{f}_o$ can only be one of K values produced by the FFT. However, we can extend this accuracy by interpolating across a fixed set of FFT bins if desired. The frequency resolution of each FFT bin for the DBPSK system is simply:

$$f_r = \frac{1}{2TK}. \tag{4}$$

Therefore, we can increase the performance of our estimator by either increasing the FFT size or decreasing the sample rate of the signal.

3.2 Questions

1. Based on Equation (3), implement a coarse frequency correction function in MATLAB using the `fft` function. Utilize `lab6part1.m` to help generate the necessary source signals. Note: you may use Matlab's comm.CoarseFrequencyCompensator system object.
2. Modify Equation (3) to handle a Quadrature Phase Shift Keying (QPSK) signal instead of DBPSK, and provide a MATLAB function for coarse frequency correction of such a signal.

3.3 Fine Frequency Correction

After coarse frequency correction (CFC), there will still be residual offset based upon the configured resolution. Fine frequency correction (FFC), also called carrier phase correction, should produce a stable constellation for eventual timing recovery and demodulation. We describe this correction as producing a stable offset because fine frequency offset effects are typically examined with a constellation diagram. If a discrete digitally modulated signal exhibits frequency offset, this will cause rotation over time in the constellation diagram. Figure 4 demonstrates this effect, generated from the provided script `lab6part2.m`. Each number indicates a sample's relative occurrence in time. Note that with a positive frequency offset, the rotation will be counterclockwise, and with a negative offset, it will be clockwise. When running the script `lab6part2.m`, this effect should be animated by the `comm.ConstellationDiagram` scope. The rotation rate equals the frequency offset, which is another reason it is sometimes defined as ω (angular frequency).

Unlike CFC which uses a feed-forward technique, for FFC we will implement a feed-back or closed-loop method based on phase-locked loop (PLL) theory. The structure of this algorithm is

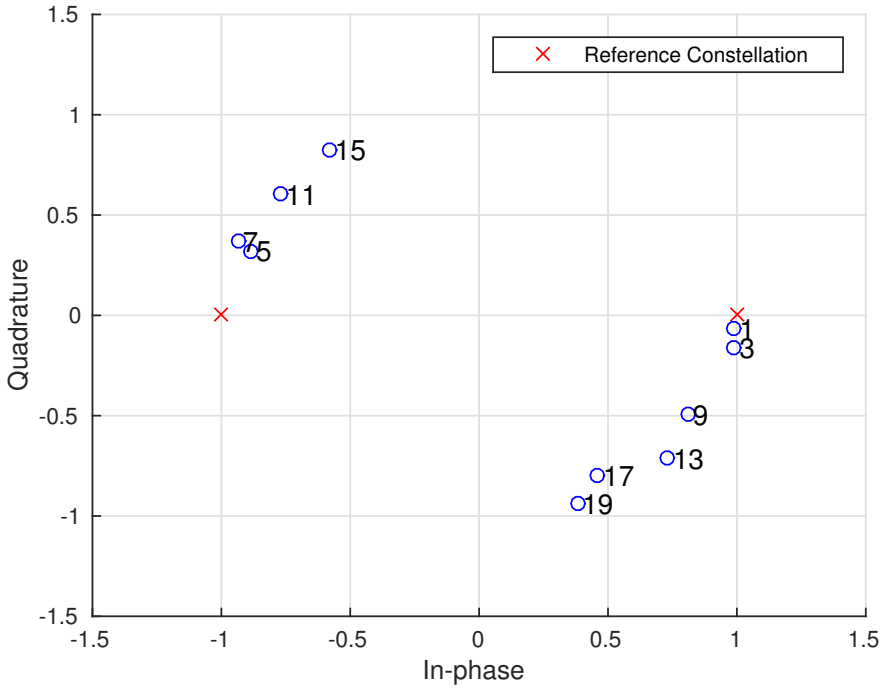

Figure 4: Constellation of source signal with frequency offset.

provided in Figure 5, derived from [3, Chapter 7], which essentially "locks" when the error signal $e(k) = \theta - \hat{\theta}(k)$ is driven to zero. This all-digital PLL-based algorithm works by first measuring the phase offset of the received sample in the Phase Error Detector (PED), which we call the error signal $e(k)$. The PED is designed based on the structure of the desired receive constellation/symbols. Next, the Loop Filter helps govern the dynamics of the overall PLL. It determines operational frequency range (sometimes called pull-in range), lock time, and dampness/responsiveness of the PLL. Finally, we have the Direct Digital Synthesizer (DDS), whose name derives from analog PLL designs with voltage controlled oscillators (VCO). The DDS generates the correction signal for the input, which is fed back into the system. In the FFC design, this PLL should eventually produce an output signal with frequency offset equal to zero.

When considering our DBPSK system, we must design all components in Figure 5 specifically. However, we can generalize this design for some modulation schemes. The instantaneous phase error for a rotated DBPSK input signal $y(k)$ can be determined by:

$$e(k) = \text{sign}(\text{Re}(y(k))) \times \text{Im}(y(k)) \tag{5}$$

where Re() and Im() represent the real and imaginary components respectively. The reasoning behind Equation (5) is straightforward. On the other hand, the Loop Filter in all PLL designs is the most challenging aspect, but provides the most control over the adaptation of the system. Here we will use a proportional-plus-integrator (PI) filter as our Loop Filter, with a transfer function:

$$F(s) = g_1 + \frac{g_2}{s}, \tag{6}$$

where g_1 and g_2 are selectable gains.

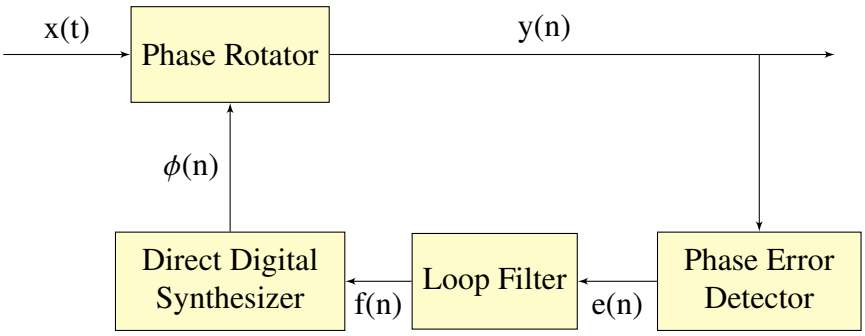

Figure 5: FFC structure based on PLL design for feed-back corrections.

PI filters produce second-order PLLs with a single pole in their transfer function. Since we are dealing with discrete time signals, a z-domain representation is preferable:

$$F(z) = G_1 + \frac{G_2}{1 - z^{-1}}, \tag{7}$$

where $G_1 \neq g_1$ and $G_2 \neq g_2$[1]. The fractional portion of Equation (7) can be represented by a biquad filter[2].

The last piece of this second-order PLL is the DDS, which is just an integrator. Since the Loop Filter produces a control signal equivalent to the frequency of the input signal, we need to extract the phase of this signal instead. The transfer functions used for the integrator are:

$$D(s) = G_3 \frac{1}{s} \quad \rightarrow \quad D(z) = G_3 \frac{z^{-1}}{1 - z^{-1}}. \tag{8}$$

Note that we have added a single sample delay in the discrete domain, and since we are producing a correction signal, $G_3 = -1$. This integrator can also be implemented with a biquad filter.

In this arrangement, the PLL should produce an output $y(k)$ with minimal phase and frequency offsets. Following the loop in Figure 5, the PED first produces an error equal to the phase offset of the corrected[3] symbol $y(k)$, then the Loop Filter relates this observed error and weights it against all previous errors. Finally, the DDS converts the weighted error/control signal $f(k)$ to a phase $d(k)$ which corrects the next input sample $x(k+1)$. For frequency offsets, d will continuously change since it is a phase value, not a frequency value. However, if the input signal is too dynamic or the filter gains are not properly designed, the PLL cannot keep up with the changing phase (frequency) of x.

[1] A simple way to translate between Equations (6) and (7) is to utilize a bilinear transform.

[2] See `dsp.BiquadFilter` for a simple realization of a biquad filter.

[3] We define this as a corrected symbol since it has passed through the rotator and we will not apply additional phase shifts to this sample. This is also the output of the PLL.

For calculating the gain values (G_1, G_2), use the following equations based on a desired damping factor ζ and loop bandwidth B_{Loop}:

$$\theta = \frac{B_{\text{Loop}}}{M(\zeta + 0.25/\zeta)} \qquad \Delta = 1 + 2\zeta\theta + \theta^2 \tag{9}$$

$$G_1 = \frac{4\zeta\theta/\Delta}{M} \qquad G_2 = \frac{(4/M)\theta^2/\Delta}{M} \tag{10}$$

where M is the constellation order. Note that B_{Loop} is a normalized frequency. For the derivation of these equations, see [3, Appendix C]. The selection of ζ affects the system behavior:

$$\zeta = \begin{cases} < 1, & \text{Underdamped} \\ = 1, & \text{Critically damped} \\ > 1, & \text{Overdamped} \end{cases} \tag{11}$$

which determines the responsiveness and stability of the PLL. The selection of B_{Loop} should relate to the maximum desired frequency locking range $\Delta_{f,\text{lock}}$:

$$\Delta_{f,\text{pull}} = \frac{4(2\pi\sqrt{2}\zeta B_{\text{Loop}})^2}{B_{\text{Loop}}^3}. \tag{12}$$

Note that this value is an estimate based on a linearized model of the PLL; therefore, some inconsistencies may exist in simulation. However, this PLL design should perform well even under strong noise conditions when configured correctly. Unlike the CFO correction, this FFC generally has a smaller operational range. In your designs, it may be useful to start with a damping factor of $\zeta = \frac{1}{\sqrt{2}}$ and a loop bandwidth of $B_{\text{Loop}} = 0.01$.

With this FFC implementation, we can achieve a stable constellation with minimal rotation or frequency offset. Figure 6 shows an initial constellation and the resulting constellation after FFC, while Figure 7 demonstrates a converging PLL over time.

When evaluating FFC performance, there are several common methods. First is to calculate the mean squared error (MSE) of the estimated offset, and second is to measure the error vector magnitude (EVM) compared to a reference constellation. EVM measurements are particularly useful as they provide information on signal orientation correctness, which indicates potential downstream processing issues. To calculate EVM in percent RMS, use:

$$\text{EVM}_{\text{RMS}} = 100 \times \sqrt{\frac{\frac{1}{N}\sum_{k=0}^{N-1} e_{\text{const}}(k)}{\frac{1}{N}\sum_{k=0}^{N-1}(\text{Re}(y(k))^2 + \text{Im}(y(k))^2)}}, \tag{13}$$

where

$$e_{\text{const}}(k) = (\text{Re}(y(k)) - \text{Re}(\bar{y}(k)))^2 + (\text{Im}(y(k)) - \text{Im}(\bar{y}(k)))^2 \tag{14}$$

and $\bar{y}(k)$ is the reference symbol for $y(k)$. EVM measures the dispersiveness of the received signal, with lower values indicating better performance.

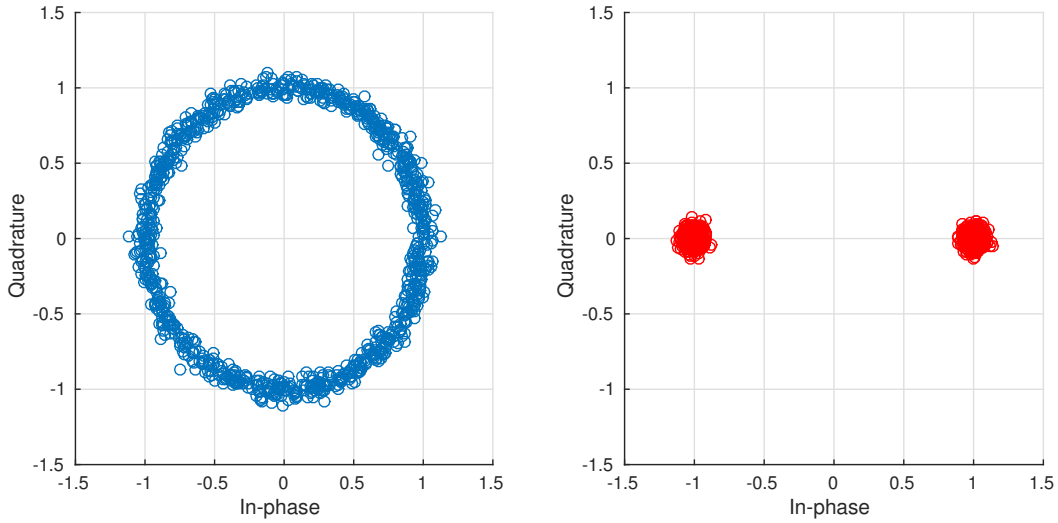

Figure 6: Constellation of source signal with frequency offset (left) and corrected constellation using FFC PLL design (right).

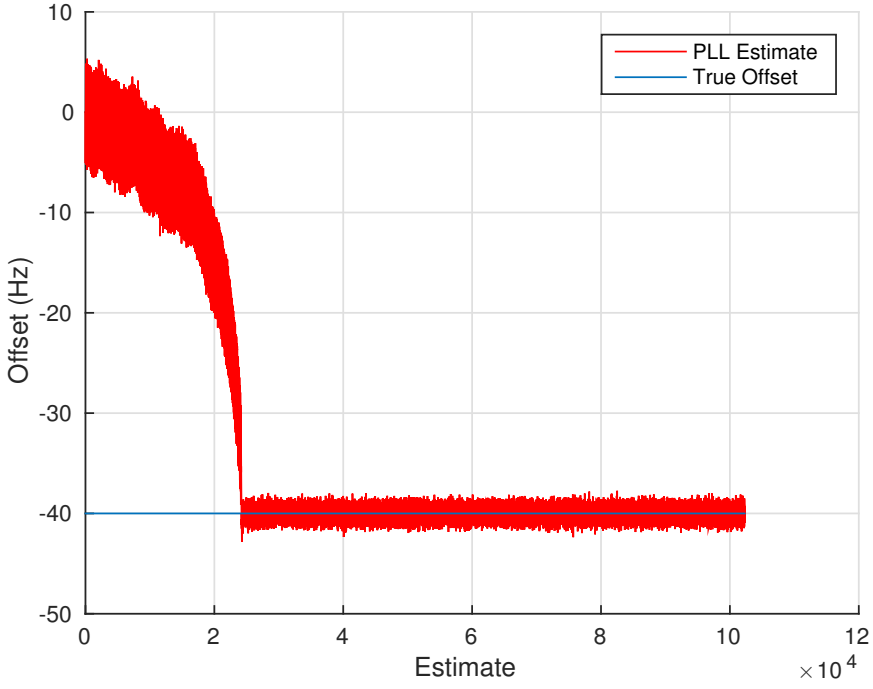

Figure 7: Estimations over time and convergence of implemented PLL.

3.4 Questions

1. Based on Section 3.3, implement an FFC in MATLAB. Utilize `lab6part1.m` to help generate the necessary source signals with frequency offsets.
   You may also use Matlab's `comm.CarrierSynchronizer` if you choose.
2. With your constructed FFC, evaluate the effect on convergence times for significant offsets with different damping factors and loop bandwidths. Illustrate scenarios of interest with plots. (Note: Phase error estimate is an available output if using MATLAB object)
3. With your constructed FFC, generate phase corrected estimates and check them against your chosen offset with EVM and/or MSE evaluations.
4. What would be an appropriate PED for QPSK?

4 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] Analog Devices. Rf agile transceiver: Ad9364. [Online]: http://www.analog.com/media/en/technical-documentation/data-sheets/AD9364.pdf.

[2] Michele Morelli and Umberto Mengali. Feedforward frequency estimation for psk: A tutorial review. *European Transactions on Telecommunications*, 9(2):103–116, 1998.

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

[4] Y. Wang, K. Shi, and E. Serpedin. Non-data-aided feedforward carrier frequency offset estimators for qam constellations: A nonlinear least-squares approach. *EURASIP Journal on Advances in Signal Processing*, 2004(13):856139, 2004.

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.