Lab 6 - Digital Modulation: Carrier Synchronization

ECE531 – Software Defined Radio

Spring 2024

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.

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 outline in Figure 1, we are now considering the Carrier Recovery and carrier frequency offset (CFO) blocks.

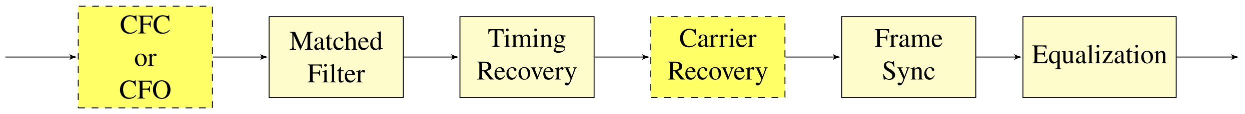

Figure 1: Receiver Block Diagram.

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.

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

The determination of f_Δ is important because it provides our carrier recovery design criteria. There is no point wasting resources on a capability to correct for a 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 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.

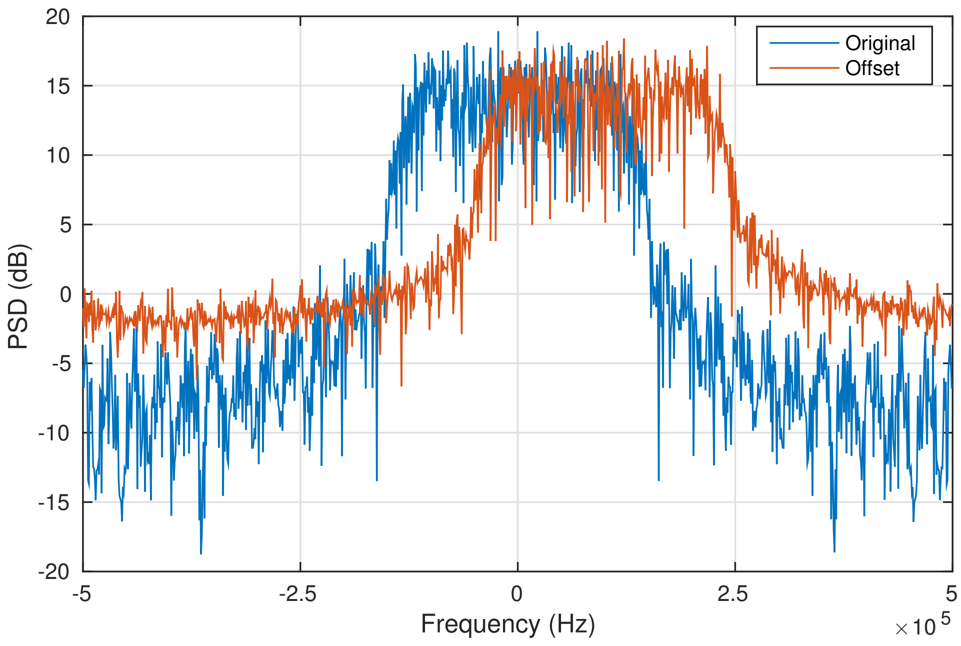

Figure 2: PSDs of offset and original transmitted signals.

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 120 kHz between the radios. Since this is quite a large range we will develop a two stage frequency compensation technique separated in coarse and fine frequency correction. This design is favorable, since it can reduce converge 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 type 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 there performance is generally limited by the length of the preambles [2], and as the preamble length is increase this decreases system throughput.

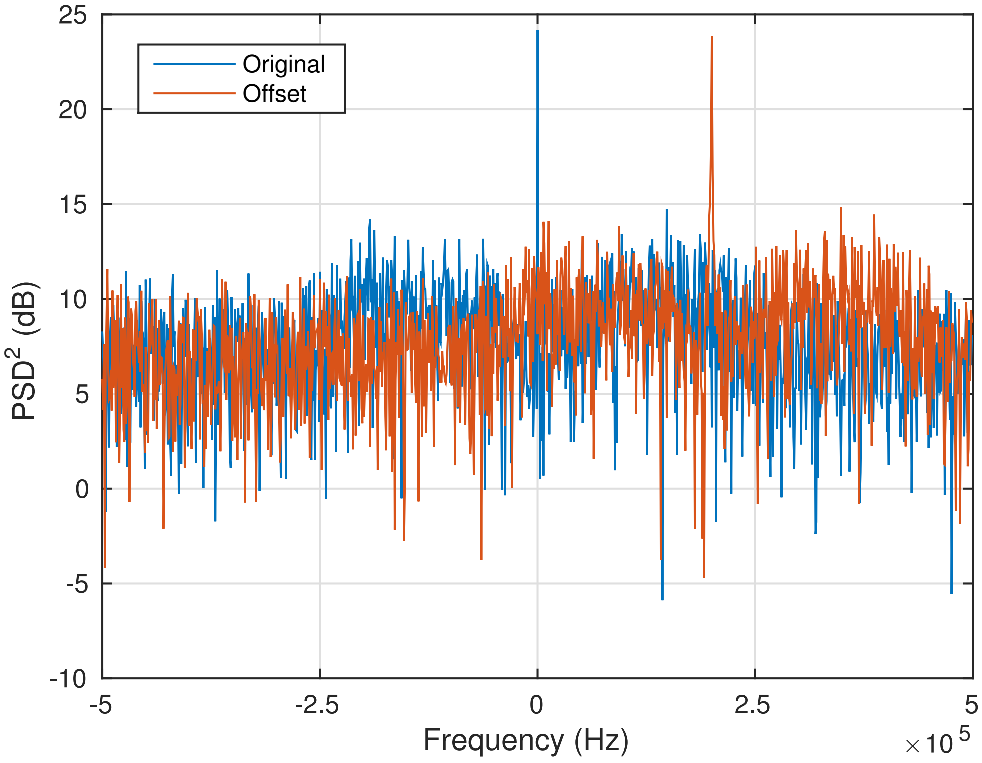

Figure 3: PSDs squared of offset and original transmitted signals.

Alternatively, blind or non-data-aided (NDA) methods can operate over the entire duration of the signal. Therefore, it can be argued in a realistic system NDA will outperform DA algorithms. In this lab we will implement a NDA FFT-based technique for coarse compensation. The concept applied here is straightforward, based on our initial inspection provided in Figure 2 we can provide a rough estimate on the symbols offsets. To make this more accurate for BPSK/DBPSK we can simply square received signal to create Figure 3. This provides 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:

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

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 increasing the FFT size or decrease 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.
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. Evaluate the effective range of this FFT method for DBPSK and QPSK. How would you parameterize this estimator for two talking Plutos with these modulation schemes?
4. At what angular frequency offset would we still recovery DBPSK without additional frequency correction and for how many symbols?

3.3 Fine Frequency Correction

After coarse frequency correction (CFC) there will still be offset based upon the configured resolution chosen. 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, due to how fine frequency offset effects are typically examined with a constellation diagram. If a discrete digitally modulated signal exhibits frequency offset, this will cause rotation overtime with a constellation diagram. In Figure 4 we demonstrate this effect which was generated from the provided script `lab6part2.m`. Each number relates a sample's relative occurrence in time, and note that if a positive frequency offset is applied the rotation will be counterclockwise and clockwise with a negative offset. When the script `lab6part2.m` is run this effect should be animated by the `comm.ConstellationDiagram` scope. The rate of the rotation is equal to the frequency offset, which is another reason it is sometimes defined as ω (angular frequency).

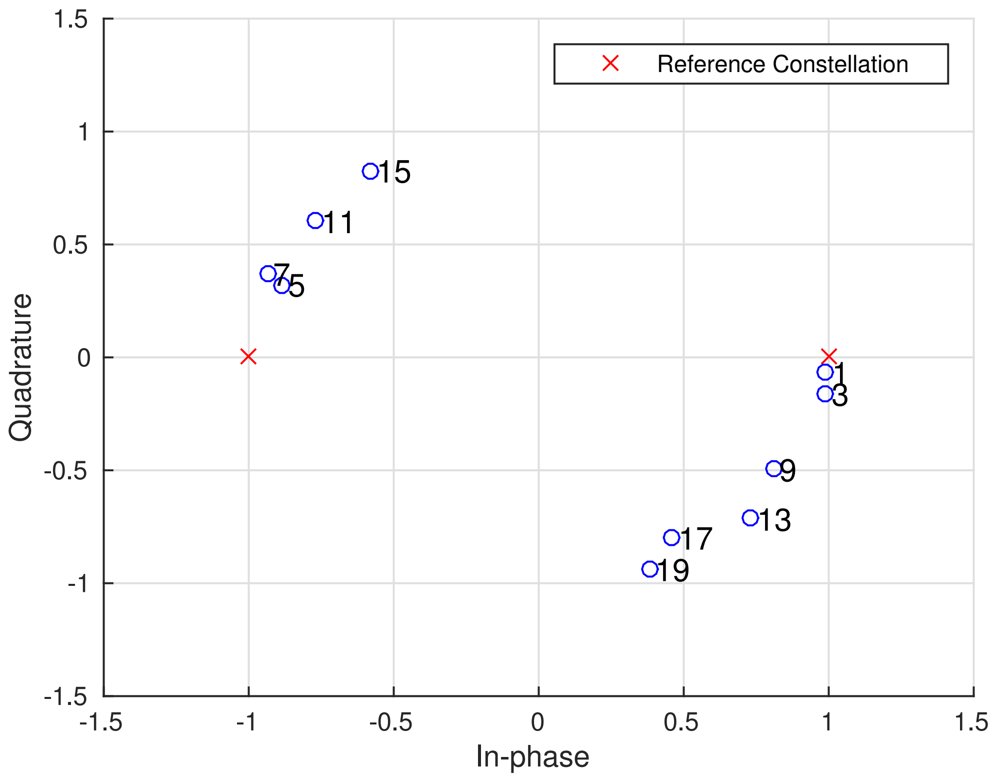

Figure 4: Constellation of source signal with frequency offset.

Unlike CFO which uses a feed-forward technique, for FFC we will be using a feed-back or closed-loop method based on phase-locked loop (PLL) theory. The structure of this algorithm is 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. The Loop Filter can determine operational frequency (sometimes called pull-in range), lock time, and dampness/responsiveness of the PLL. Finally, we have the Direct Digital Synthesizer (DDS), whose name is a remnant of analog PLL designs with voltage controlled oscillators (VCO). The DDS is responsible for generation of the correction signal for the input, which again will be fed back into the system. In the case of the FFC design, this PLL should eventually produce an output signal with desired offset equal to zero.

When considering our DBPSK system we must design all the 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 simply by:

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

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

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

where g_1 and g_2 are selectable gains.

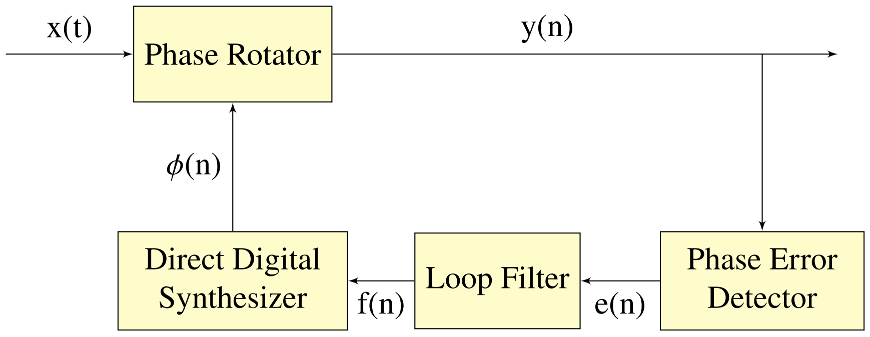

Figure 5: FFC structure based on PLL design for feed-back corrections.

PI filters produce second-order PLLs and only consist of 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 nicely by a biquad filter[2].

The last piece to this second-order PLL is the DDS, which is just an integrator. Since the Filter Loop produces a control signal which is equivalent to the frequency of the input signal, it becomes necessary to extract the phase of this signal instead. The transfer functions used for the integrator here 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 an additional delay of a single sample in the discrete domain, and since we are producing a correction signal $G_3 = -1$. Again this integrator can be implemented with a biquad filter.

[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.

In this arrangement of the PLL, filling in the desired responses in Figure 5, this system should produce an output $y(k)$ which has minimal phase and frequency offsets. Going around the loop again in Figure 5, the PED will first produce an error equal to the phase offset associated with the observed corrected[3] symbol $y(k)$, then the Loop Filter will relate this observed error and weight it against all previous errors. Finally, the DDS will convert the weighted error/control signal $f(k)$ to a phase $d(k)$ which we use to correct the next input sample $x(k+1)$. In the case of frequency offsets, d will continuously change since is it a phase value not a frequency value. However, if the input signal is too dynamic or the gains of the filters are not designed appropriately, the PLL will not be able to keep up with the changing phase (frequency) of x.

For the calculation of the gain values (G_1, G_2) utilize the following equations based on a preferred damping factor ζ and loop bandwidth B_{Loop}:

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

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

where M is the constellation order. Note that B_{Loop} is a normalized frequency. If you are interested in how these are derived, see [3, Appendix C]. For the selection of ζ:

$$\zeta = \begin{cases} \leq 1, \text{Underdamp} \\ = 1, \text{Damped} \\ \geq 1, \text{Overdamped,} \end{cases} \tag{11}$$

which will determine the responsiveness and stability of the PLL. The selection of B_{Loop} should be related to the maximum estimated frequency locking range $\Delta_{f,lock}$ range desired:

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

Note that this value is an estimate based off a linearized model of the PLL; therefore inconsistencies may exists in the simulated versions. However, this PLL design should perform well even under strong noise conditions when configured correctly. Unlike the CFO correction this FFC will generally not have the same operational range. In your designs is may be useful to start with a damping factor of $\zeta = \frac{1}{\sqrt{2}}$ and a loop bandwidth of $B_{Loop} = 0.01$.

With this FFC implementation we can implement a system that provides a stable constellation with minimal rotation or frequency offset. Figure 6 provides an initial constellation and the resulting constellation after FFC, and Figure 7 provides an example of a converging PLL over time.

When determining performance of the FFC there are several common methods. First is to evaluate the mean squared error (MSE) of the estimated offset, and second is to take an error vector

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

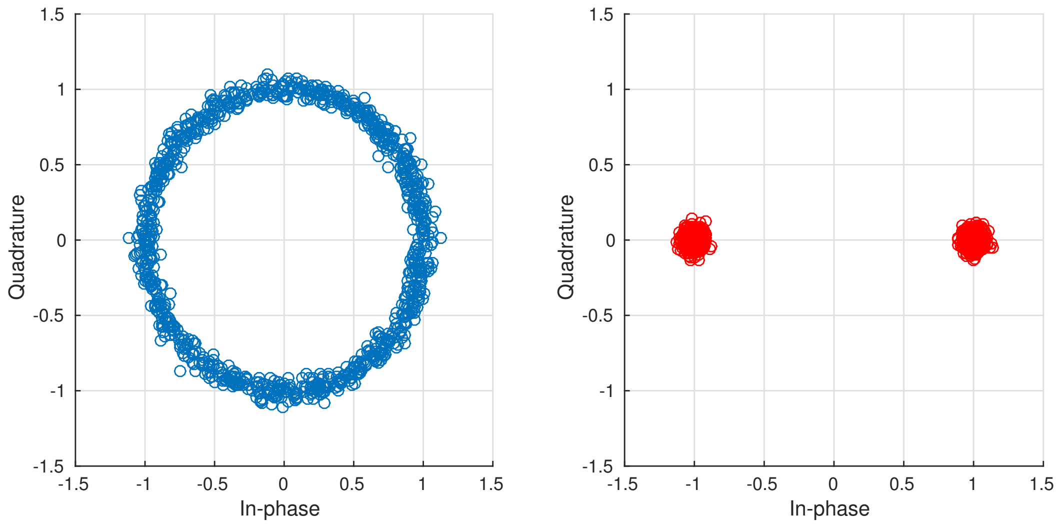

Figure 6: Constellation of source signal with frequency offset (left) and corrected constellation using FFC PLL design (right).

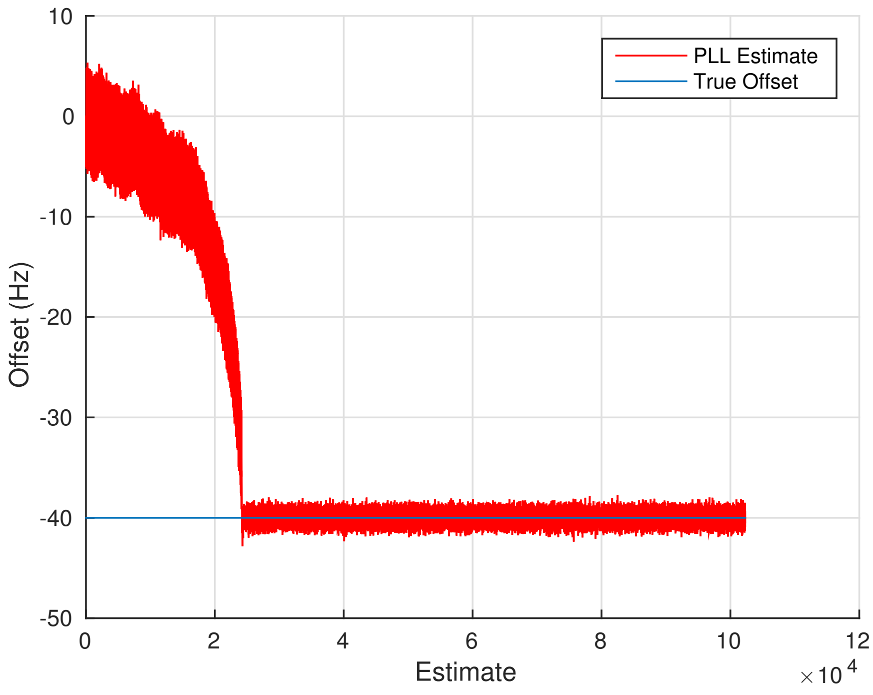

Figure 7: Estimations over time and converge of implemented PLL.

magnitude (EVM) measurement compared to a reference constellation. EVM measurements can be useful because they provide information on the correctness of the signal orientation, which will provide insight into possible downstream errors. To calculate EVM in percent RMS use the following equation:

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

where

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

and $\bar{y}(k)$ is the reference symbol for $y(k)$. EVM is a measure on the dispersiveness of the received signal. Therefore, the lower the EVM values the better.

3.4 Questions

1. Based on Section 3.3 implement a FFC in MATLAB. Utilize `lab6part1.m` to help generate the necessary source signals with frequency offsets.
   You may also use `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 MSE and/or EVM evaluations.
4. Ignoring timing correction what is the maximum frequency offset your FFC can handle? Describe how you determined this conceptually.
5. What would be an appropriate PED for QPSK?

4 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.
- Conclusions to the overall lab that discuss meaningful lessons-learned and other take-aways from the assignment.
- 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] 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.