WPI

ECE3311 – Principles of Communication Systems

Project 04

Due: 9:00 AM, Tuesday 1 December 2020

1 Project Objective & Learning Outcomes

The objective of this project is to introduce the concept of the phase locked loop (PLL).

From this project, it is expected that the following learning outcomes are achieved:

- Obtain an understanding of how the general concept of the phase locked loop enables a receiver to lock onto the carrier frequency of an intercepted waveform.
- Investigate the operations of a Costas Loop when applied as a fine frequency compensator to several scenarios involving mismatched carrier frequencies between the transmitter and receiver due to local oscillator imperfections.
- Study how a Costas Loop performs when its operating parameters are well-chosen with respect to time-varying phase values of intercepted passband signals.

For this project, you will need to download the Jupyter Notebook "*Project 4 ECE3311.ipynb*" from the ECE3311 Canvas website and open it in VSCode, which will provide you with all the example source code and outputs of subsequent topics discussed in this project. Additionally, three separate NumPy mystery signal files, *mysterysignal1.npy*, *mysterysignal2.npy*, and *mysterysignal3.npy*, are provided in support of Section 6.

2 Preparations

One of the primary reasons for using the Python programming language in this course is that it is extensively used in real-world radio hardware prototyping via *software-defined radio* (SDR). Given the complexity of implementing an actual phase locked loop (PLL) algorithm, we will be using a PLL algorithm *prêt-à-porter* from a major SDR open source development environment called *GNU Radio* [1].

Since GNU Radio was not originally installed with the ECE3311 Ubuntu disk image, you will need to install this before continuing with the rest of the project. To install Gnu Radio, open up a terminal and enter the following command:

```
sudo apt install gnuradio
```

Running this command will take several minutes (good opportunity to refill on coffee), but once fully executed all the files needed for using the associated PLL algorithms should be available in Python.

3 Phase Locked Loops

Phase locked loops (PLLs) are closed-loop systems employed in communication systems that can converge and "lock" onto the carrier frequency of an intercepted signal. With the carrier frequency identified, the intercepted signal can be demodulated down to baseband and the signal can be decoded in order to extract its message signal $m(t)$. However, PLLs are not trivial to design and implement, especially given their complex behavior and feedback path that can influence whether the system converges to a stable solution.

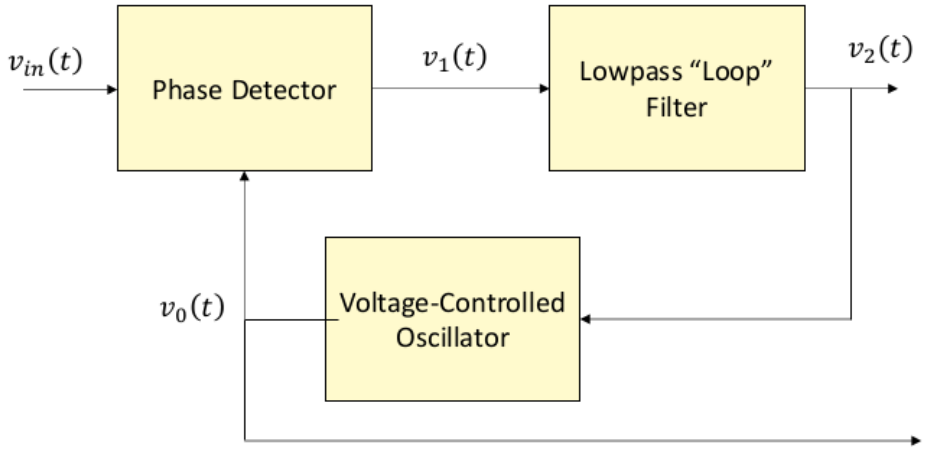

Figure 1: Illustration of a generic PLL, consisting of phase detector, lowpass "loop" filter, and voltage-controlled oscillator blocks.

Reproducing Figure 4–21 from the course textbook, an illustration of a generic PLL is shown in Figure 1 that consists of the following basic functional components: a phase detector, a lowpass "loop" filter, and a voltage-controlled oscillator. The phase detector is designed to extract out the phase difference between the input signal $v_{\text{in}}(t)$ and the output of the VCO $v_0(t)$, where the latter serves as our reference signal and is used by the receiver to maintain the "lock" on $v_{\text{in}}(t)$. Since the result of the phase detector, $v_1(t)$, produces high frequency elements that could disrupt the locking process by the PLL on $v_{\text{in}}(t)$, a lowpass filter (*i.e.*, loop filter) is used to remove those high frequency components such that only the phase differences (contained within $v_2(t)$) are used by the PLL and the rest of the receiver. Finally, the phase difference values contained within $v_2(t)$ are used to determine the level of oscillation at the output of the VCO, $v_0(t)$.

In general, the PLL is a complicated closed loop feedback control system that attempts to maintain a constant phase error θ_e over time. Note there is an initial transient response by the PLL as it attempts to achieve lock; plots of θ_e as the PLL tries to lock onto $v_{\text{in}}(t)$ is very useful in characterizing how well or how poorly the PLL is performing with respect to achieving a constant θ_e.

3.1 Costas Loop

In many communication systems, Costas Loops [2, 3, 4] are frequently used in order to perform fine frequency correction. The block diagram for a Costas Loop is shown in Figure 2. As opposed to the generic PLL discussed earlier in this project handout, the Costas Loop decomposes the input signal $v_{\text{in}}(t) = A_c m(t)\cos(\omega_c t)$ into inphase (I) and quadrature (Q) branches by multiplying the signal by $A_0\cos(\omega_c t + \theta_e)$ and $A_0\sin(\omega_c t + \theta_e)$, respectively. After lowpass filtering the I and Q results in

order to remove the double frequency terms that will occur from the multiplication process, we get $v_1(t) = 0.5A_0A_c\cos(\theta_e)m(t)$ and $v_2(t) = 0.5A_0A_c\sin(\theta_e)m(t)$. Then, $v_1(t)$ and $v_2(t)$ are multiplied together to produce $v_3(t)$ which is subsequently filtered by the loop filter to produce the input to the VCO, $v_4(t)$. The loop filter smooths out the rapid variations in $v_3(t)$ resulting from $m^2(t)$, thus yielding $v_4(t) = K\sin(2\theta_e)$, where $K = 0.125A_0^2A_c^2\langle m^2(t)\rangle$ and $\langle m^2(t)\rangle$ is the DC level of $m^2(t)$. This enables the Costas Loop to lock onto $v_{\text{in}}(t)$ with very small values for θ_e.

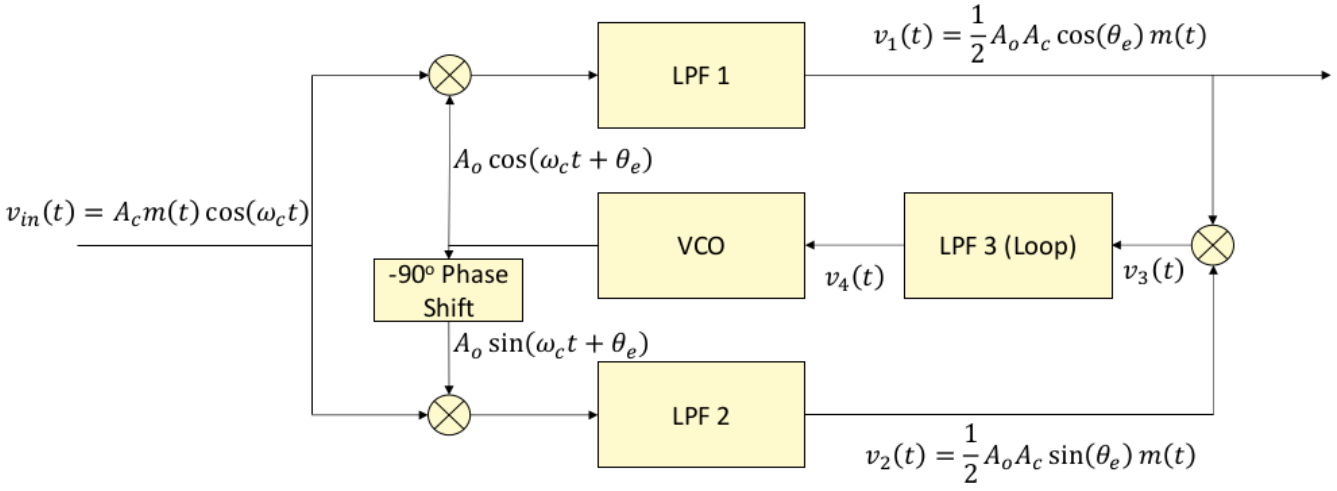

Figure 2: Illustration of a Costas Loop and all its respective components. Notice how both inphase (I) and quadrature (Q) signal components are retrieved from input signal $v_{\text{in}}(t)$ and used later on to extract the phase rotation term for the signal.

Although the Costas Loop might appear straightforward, it is actually a very challenging control problem that requires extensive analysis (refer to [3] for a detailed analysis of Costas Loops from a control theory perspective). In particular, it is very challenging to select parameters that would yield a Costas Loop that will converge to a stable and constant phase error value θ_e. Two parameters that you will be exploring as part of this project are the *loop bandwidth* W_L and the *damping factor* ζ.

The loop bandwidth refers to the bandwidth of the loop filter, which corresponds the "LPF 3" in Figure 2. Suppose the loop filter possesses a transfer function $F(f)$, then according to Eq. (2-23) in Reference [4] the loop bandwidth is equal to:

$$W_L = \int_{-\infty}^{+\infty} \left|\frac{Y(f)}{1+Y(f)}\right|^2 df \tag{1}$$

where $Y(f) = K_vK_mA^2F(f)/j2\pi f$ (see Eq. (2-17) in Reference [4]). Note that W_L corresponds to the bandwidth of a digital filter, which means it is defined across $-\pi$ to π.

As for the damping factor ζ, it is used in several places within the construction of the overall Costas Loop to determine the loop response, which can either be over-dampened ($\zeta > 1$), under-dampened($\zeta < 1$), or critically-dampened ($\zeta = 1$). In many instances, $\zeta = 0.707$ is often used when implementing PLLs.

Overall, the behavior and performance of your Costas Loop can be entirely characterize by the choice of W_L and ζ, and these parameters are defined as inputs to the Costas Loop class available from GNU Radio, as we will see in the next section.

4 Basic Experimentation

In this project, we will be working in the *complex baseband* domain, *i.e.*, all the signals have already been down converted from passband to around 0 Hz but they have not yet been mapped back from $g(m(t))$ to a binary representation. Additionally, coarse frequency correction has been performed on the intercepted signal; the purpose of the Costas Loop in this project is to provide the fine frequency correction.

For this section, we will experiment with a simple quadrature phase shift keying (QPSK) waveform possessing a constant frequency offset Δf and constant phase offset θ_0. The frequency offset is potentially due to the imperfections of the local oscillator, while the phase offset is probably the result of the channel between the transmitter and the receiver. Furthermore, $W_L = 2\pi/100$ and $\zeta = 0.707$.

Question 1 (2 points): Plot the scatter plot before and after the Costas Loop, and brief describe what you observe and explain the difference between these two results.

Question 2 (2 points): Plot the phase error θ_e for the Costas Loop. What do you observe and explain.

Question 3 (5 points): Repeat this experiment for $\zeta = 0.1$, 1, and 2. Plot both the scatter plots and phase error curves. What do you observe and explain.

5 Time-Varying Phase

Now that you have experienced a scenario where the phase offset and frequency offset values were constant, let us now explore the case when these values are time-varying. Specifically, the frequency offset Δf is still maintained as a constant, but the phase offset is time-varying and defined as a function of time; $\theta_1(t)$ is a linearly increasing function, $\theta_2(t)$ is a sine function, and $\theta_3(t)$ is a unit step function.

Question 4 (9 points): Plot both the scatter plots and phase error curves for each of these three different phase offset functions. Explain your observations, especially if there are any scenarios where the Costas Loop loses lock and attempts to reacquire the signal.

6 Mystery Files

In this last section of the project, you are provided with three mystery files containing the complex baseband signals at the receiver after down conversion. Each of these files possesses a different modulation schedule (BPSK, QPSK, and 8PSK). Additionally, each of these transmissions has experienced a phased offset defined by an arbitrary function that will not be disclosed.

Question 5 (5 points): Plot both the scatter plots for each of these three mystery files before and after the Costas Loop. Explain your observations. Also specify your choice of W_L and ζ.

Question 6 (5 points): Plot the phase error curves for each of these three different phase offset functions. Explain your observations.

7 Report Submission

For each project in this course, the report submission to be uploaded to the ECE3311 Canvas website will consist of a single comprehensive Jupyter Notebook (.ipynb) file and nothing else unless it is requested. For this project, only the Jupyter Notebook is to be submitted electronically by the due date. Failure to submit the Jupyter Notebook by the specified due date and time will result in a grade of "0%" for the project.

Several important items to keep in mind when preparing your submission:

- Include the course number, project team number, names of team members, and submission date at the top of the Jupyter Notebook.
- Make sure your source code is thoroughly documented such that it is made clear what exact the program is doing line-by-line. Adequate commenting is worth **5 points** of the total score for this project.
- Responses to all questions indicated in the project handout. Please make sure that the responses are of sufficient detail.

References

[1] GNU Radio, "Gnu radio: The free and open software radio ecosystem." https://www.gnuradio.org/".

[2] J. Feigin, "Practical costas loop design," *RF Signal Processing*, pp. 20–36, Jan. 2002.

[3] R. E. Best, N. V. Kuznetsov, G. A. Leonov, M. V. Yuldashev, and R. V. Yuldashev, "Tutorial on dynamic analysis of the costas loop," *Annual Reviews in Control*, 2016.

[4] A. M. Mehta, "Digital simulation of a costas loop demodulator in gaussian noise and cw interference," Master's thesis, Missouri University of Science and Technology, 1970.