The Costas Loop – Closing the Loop

by Eric Hagemann

My previous column (check for it in the online archives) introduced a structure suitable for implementing a phase-locked loop in software. It derived the structure from work done by J. P. Costas, work he originally described in his patent (http://www.eepatents.com/feature/?m07d01). Left unexamined in my introductory column were the feedback mechanism as well as how to close the loop and bring it into lock. Thus, this column focuses on methods of closing the loop and finishing the structure; the next installment will focus on setting loop behavior.

The Costas loop

To refresh your memory, examine Fig 1, which depicts the basic structure of the Costas loop. It performs a quadrature mix between a reference waveform and a received waveform to form two error signals, which when multiplied together creates a suitable signal for adjusting the oscillator. With care you can adjust the oscillator into lock -- a condition where the oscillator and reference waveform are matched in phase and frequency.

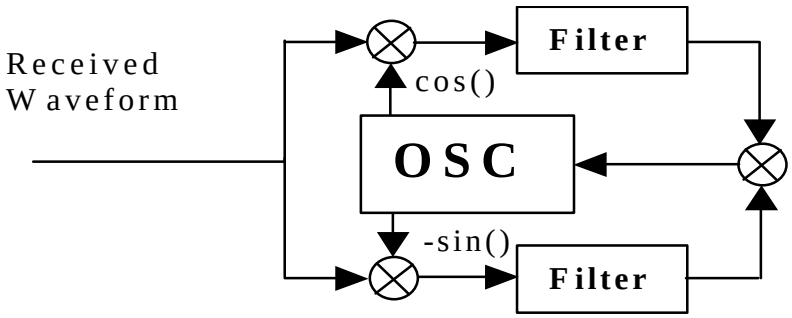

Fig 1 -- A Costas loop uses a quadrature mix and lowpass filters to generate an error signal suitable for adjusting the oscillator into lock

The software oscillator this design employs is a numerically controlled oscillator (NCO), which generates the reference waveform with the help of table-driven lookup techniques. The software NCO has an advantage over the hardware analog VCO (voltage controlled oscillator) that find use in analog based PLLs, specifically its ability to control both frequency and phase of the generated waveform. Typical hardware VCOs only allow frequency adjustments.

The NCO

During my previous investigation of table-based techniques for waveform generation, I noted that the position in the table is equivalent to the instantaneous phase of the sine wave. Further, motion though the table (how many entries an algorithm skips each time it extracts a sample) is related to the frequency. The following equation drives the lookup table (LUT) technique

$$q(n+1) = q(n) + f(n).$$

where $q(n)$ is the waveform's present phase (position of last extracted sample in the table) and $f(n)$ is the present frequency.

Before finishing the discussion of the NCO, let's move on to a few definitions. In the PLL literature you'll often encounter a mention of the "order" of a PLL. In one sense that term refers to how many integrators are in the feedback path. More importantly, "order" refers to the loop's ability to acquire lock to different input signal conditions and reduce the feedback error term to zero. Now consider several different possibilities:

First order -- A first-order loop can acquire lock to the incoming signal and adjust the phase of the reference waveform to match it in phase. However, the loop can't modify the frequency of the reference oscillator and therefore can't track a received waveform that has a constant frequency difference with a zero error signal. Because frequency is the rate of change of phase, some ability to track frequency is available because the loop can make constant phase adjustments.

Second Order -- The second-order loop is the most popular implementation, and it provides the ability to track both frequency and phase differences. This class of loop can track a constant frequency offset (by adjusting the oscillator's frequency) and thus produce a zeroed error signal. Note, though, that this loop can't track a received waveform that experiences acceleration (rate of change in frequency) with a zeroed error signal.

Third Order -- Although we won't consider it here, there is a third-order loop that can track phase, frequency and acceleration. These loops are important when the transmitter or receiver is experiencing Doppler shifts (frequency deviation due to motion or changes in motion). Important examples are airborne and satellite-based transmitters or receivers.

Recall from Physics 101 that objects with constant speed but circular motion experience constant acceleration (the definition of acceleration include changes in direction).

Simplified model

To gain a better understanding of loop operation with these three classes in mind, start by examining Fig 2, which depicts a simplified model of the loop where we consider only phase. The received waveform has an instantaneous phase of $f(n)$, and the oscillator has a phase of $q(n)$. The error or feedback waveform is the difference between these two phases. The scheme then provides a scaled version of this error signal as feedback to the oscillator.

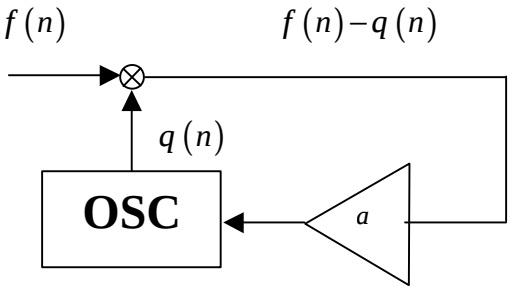

Fig 2 -- It's convenient to define PLL loop operation in terms of instantaneous phase angles.

Back to the NCO

Considering the first-order case (adjusting phase only), we can modify the equation for the LUT generator as follows (ignoring frequency for the moment):

$$q(n+1) = q(n) + a\,e(n).$$

In this equation, $e(n)$ refers to the error feedback term, which is the product of the output of the arm filters. The scheme uses the scalar a to control the amount of feedback going to the oscillator. It's also possible to highlight the operation of a first-order loop with the Z-transform.

$$\begin{aligned}\Theta(z)\,z &= \Theta(z) + a\,E(z)\\ &= \Theta(z) + a\left(\Phi(z) - \Theta(z)\right)\\ \Theta(z)\left(z - 1 + a\right) &= a\Phi(z)\\ \frac{\Theta(z)}{\Phi(z)} &= \frac{a}{\left(z - 1 + a\right)}\end{aligned}$$

This manipulation reveals that the relationship between the received waveform phase and the reference waveform phase is a recursive lowpass filter with one pole, which is located on the real axis. Further, the feedback parameter a controls the pole's position and hence filter bandwidth. As long as $a < 1$, the system is bounded and stable. The closer a comes to unity, the narrower the filter and the longer it takes to adapt.

$$z - 1 + a = 0$$

$$z = 1 - a$$

Move on to the second-order loop by modifying the LUT driving equation to include both frequency and the adjustment of frequency. Now you have two difference equations and an additional feedback parameter, b.

$$q(n+1) = q(n) + a\,e(n) + f(n)$$

$$f(n+1) = f(n) + b\,e(n)$$

Again by examining the Z–transform of these time equations you can develop an understanding of the loop operation. This time the system is a filter with two poles.

$$F(z) = \frac{b\,E(z)}{z-1}$$

$$\Theta(z)\,z = \Theta(z) + F(z) + a\,E(z)$$

$$= \Theta(z) + a\,E(z) + \frac{b\,E(z)}{z-1}$$

$$\frac{\Theta(z)}{\Phi(z)} = \frac{a\,z + b - a}{z^2 + (a-2)z + 1 + b - a}$$

This time you control the location of the poles and thus filter operation by the choice of feedback parameters a and b. Because the polynomial in the denominator has real coefficients, you can categorize their locations into three distinct cases. The poles can be real but not equal; real and equal; or complex conjugates of each other. If the poles are complex the filter oscillates about the angle of the pole and thus acts as a bandpass filter.

You want a lowpass filter, so make both poles real. If you solve for the case where two real poles are equal you end up with the fastest adaptation time. In control theory this is the case of critical damping. Because this discussion hasn't yet addressed the effect of the feedback parameter on loop operation, you can solve for the relationship between a and b that provides the critical damped case for any a. Using the quadratic equation, the answer is

$$b = a^2 / 4 .$$

So where are we?

With all this knowledge in hand, you can create a closed loop. For either a first- or second-order loop you can mix the received waveform with a reference one and construct a feedback signal. Using the feedback signal you now have equations for updating the frequency and phase of the reference oscillator. Further, it should now be clear that the update equations lead us to a 1- or 2-pole filter relationship between the instantaneous incoming and reference phases. The interested reader can use the same techniques outlined for first- and second-order loops for investigating loops of higher orders. Already you can see that choices in the feedback parameters affect these filters, and an inappropriate choice could generate an unstable filter and in turn an unstable feedback mechanism. What remains, then, is to develop an understanding of these feedback equations with a variety of input signal types -- a great topic for my next column.

Author's biography

Eric Hagemann (ehagemann@home.com) is an electrical engineer who has been programming DSPs for fifteen years. When not writing code, he spends time with his wife and two cats endlessly remodeling their house.