The Costas Loop – Setting the Loop

by Eric Hagemann

In previous columns (available from the online archives) I introduced a structure suitable for implementing a phase-locked loop in software – the Costas loop. Those articles described the loop's structure and operation, but I've postponed a discussion of how to "set the loop" until now.

The Costas loop implementation consists of a feedback structure where a derived error term forces the loop into a lock condition. As in any feedback arrangement, applying an incorrect amount can cause the loop either to not adapt (too little feedback) or blow up (too much feedback). In this installment I'll begin to investigate procedures for properly choosing the feedback coefficients for efficient operation.

A loop review

Let's start by refreshing you on a few fundamentals. Fig 1 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 create 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.

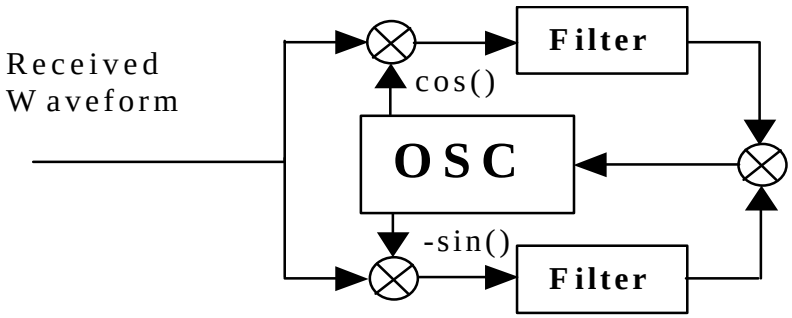

Fig 1 -- A Costas loop uses a quadrature mix and lowpass filters to generate an error signal suitable for adjusting the oscillator into lock.

The loop is a nonlinear structure, and so understanding its operation is somewhat tricky. Last month I developed a linearized model (detailed in Fig 2) to help the discussion. Using this model, the column showed how to construct equations relating the input phase

to the phase of the internal oscillator for several cases. Focusing on the 2nd-order case (where the loop can compensate for both frequency and phase offsets), it used the following time equations for updating the internal oscillator's phase and frequency (q, f), respectively.

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

Solving for the transfer function of the linearized loop reveals a 2nd-order equation defining the relationship between the instantaneous phase of the received signal and the local oscillator.

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

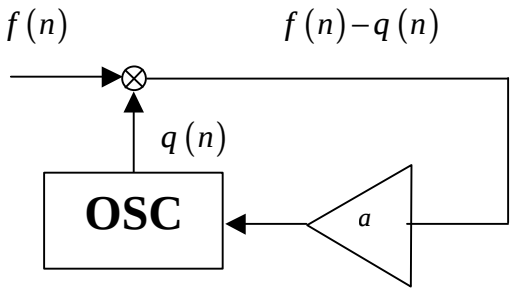

Fig 2 -- It's convenient to define PLL loop operation in terms of instantaneous phase angles.

While this transfer function defines the operation of the linearized model and not the real loop, it's a useful basis to guide the choice of the feedback constants a and b. Having a 2nd- order denominator, this polynomial indicates that the system has two poles. For critical damping you set these to be both real and equal, leaving a relationship of

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

Using the quadratic equation and the relationship between a and b, you can find the poles are located at

$$\frac{2-a}{2}$$

which supports a stable system when

$$0 < a \le 4.$$

In addition, you must consider the numerator where the single zero is located at

$$\frac{4-a}{4}.$$

Typical ranges for a include 0.01 and smaller.

Acquisition

Now that you have a structure and a basic understanding of its operation, probably the primary remaining questions are these: Over what frequency range can the loop acquire lock? And, even more basic, exactly what is lock ?

A good observation point for watching loop behavior is the phase “error term” as provided into the NCO. Ideally this term should become zero when the loop reaches a stable locked operating point. As a simplification, the author calculated the next few plots using a Costas loop that had a complex input signal (received waveform). The multiplication of sine and cosine was a full complex multiplication. This modification, while only useful in limited applications, allows you to concentrate on loop operation without concern of the arm filters. You still compute phase error in a similar fashion: as the product of the real (in phase) and imaginary (quadrature) results of the down conversion.

Fig 3 shows a loop locking to a small (0.001) frequency offset. Fig 4 shows a loop locking to larger frequency offset (0.01). All frequency terms are normalized as

$$f_N = \frac{f}{f_{sample}}.$$

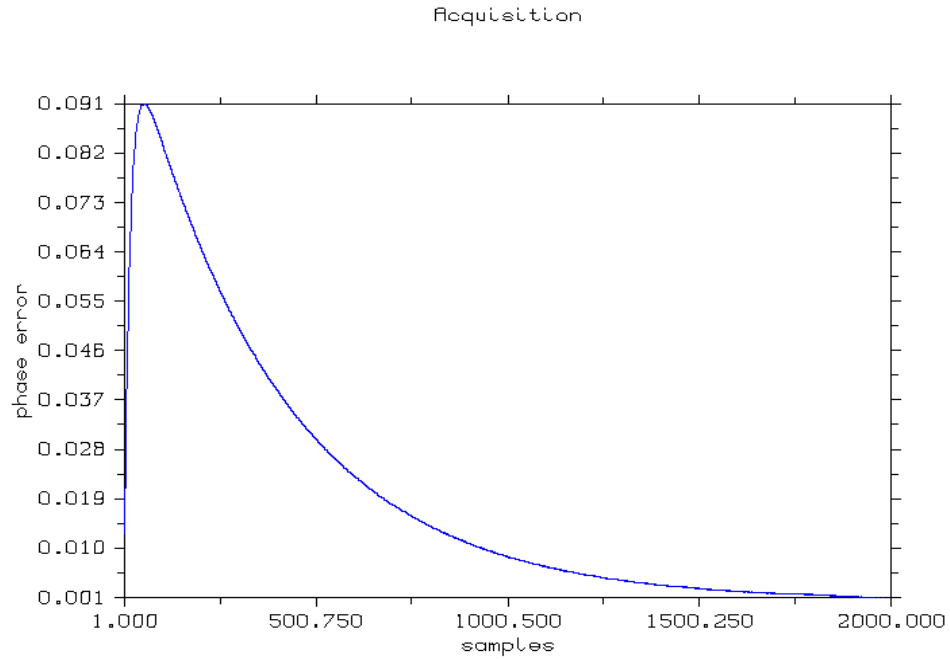

Fig 3-- Phase-error curve when locking to minor frequency offset, fn = 0.001

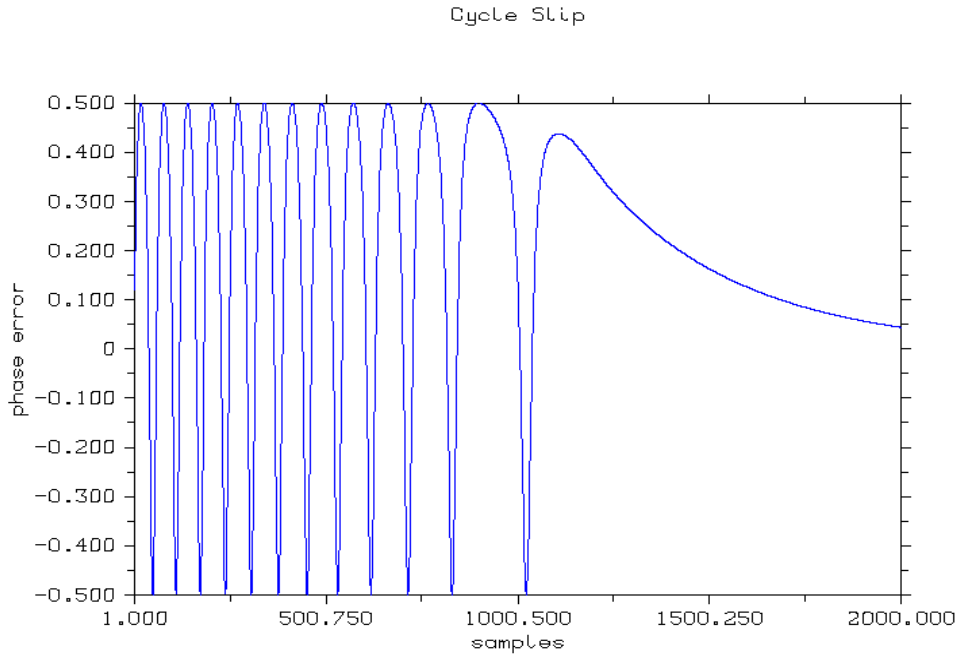

Fig 4 -- Phase-error curve showing cycle slip and lock-in
for larger frequency difference fn = 0.01.

In Fig 4, where the initial frequency offset is larger, the system experiences cycle slips as the phase of the received and reference waveforms 'slip' past each other. This effect continues until the system can bring the frequency of the reference oscillator close enough to the received stream to start the lock-in process. When cycle slipping, the phase-error term is biased to one direction or the other (in this case positive) as the received waveform leads the reference oscillator in phase.

In the case of the complex Costas loop, you can compute the time to lock versus initial frequency offset. For the next two plots, the author set the received waveform to fn = 0.25 and the reference waveform to a variety of values between fn = 0 and 0.5. Fig 5 details this result for an a of 0.05, while Fig 6 shows the time to lock (in terms of samples) for an a of 0.01.

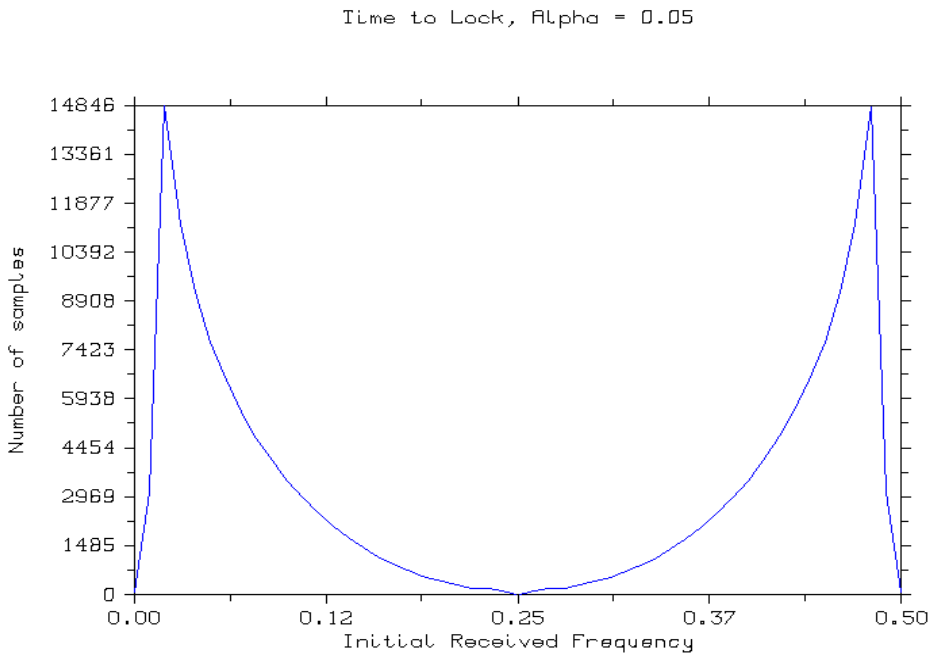

Fig 5 – Lock time (in samples) of a complex Costas loop versus initial frequency difference, a = 0.05

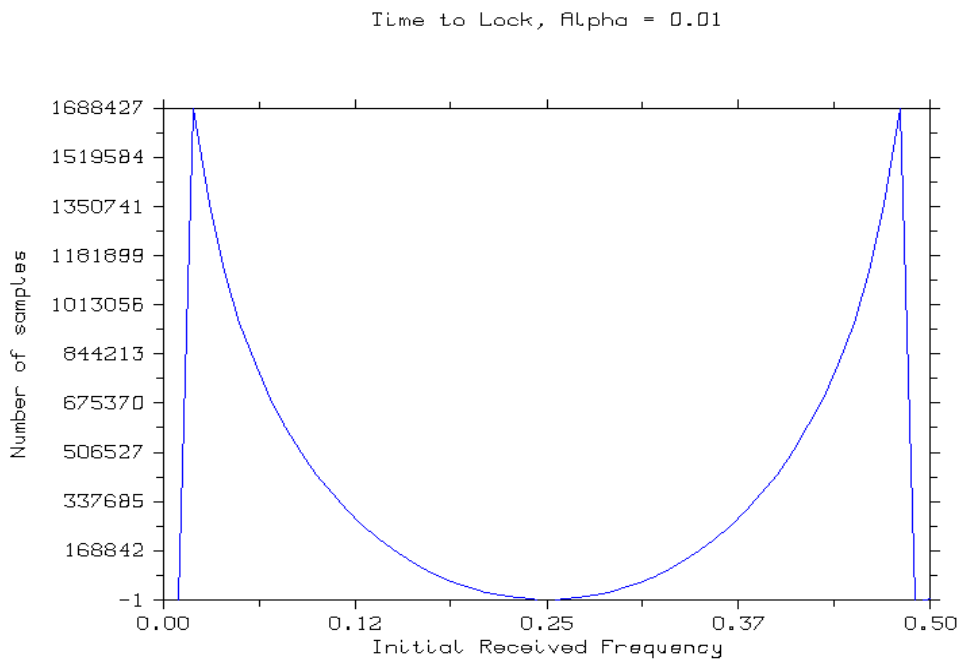

Fig 6 – Lock time (in samples) of a complex Costas loop versus initial frequency difference, $a = 0.01$

The behavior of the loop indicates that no matter what value for the input frequency difference, the loop always acquire lock – it just might take a while. The parameter a controls the amount of feedback and thus the time required to lock. The lower the value of a , the slower the loop is to adapt; conversely, the larger the faster.

Based on these results, you might expect that the operation of a loop as drawn in Fig 1 would behave the same. Except for the filters, this is essentially true. In the case of Fig 1 the arm filters are in place to remove the summed frequency term that results from the real multiplication. To make this loop work, the arm filters must be matched to each other and have sufficient bandwidth to pass the maximum expected initial frequency offset as well as attenuate the summed frequency term. Fig 7 shows a plot of lock time for a real Costas loop using FIR-based arm filters. This loop has a frequency-lock range of 0.0046 around the nominal frequency of 0.1. At a 10-kHz sample rate, this value gives an effective lock range of 46 Hz.

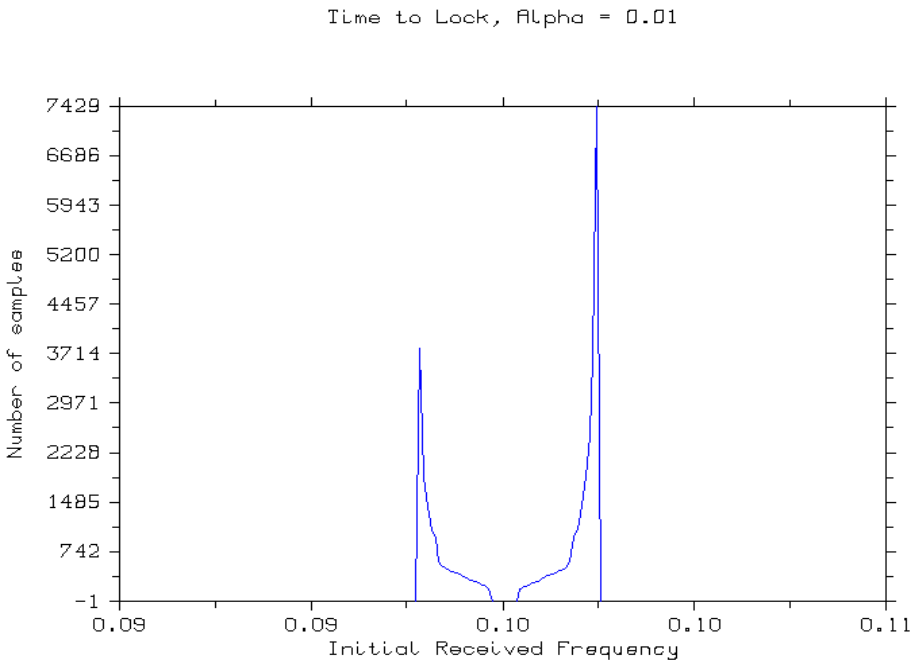

Fig 7 -- Lock time for Real Costas loop with FIR arm filters and nominal center frequency of fn = 0.01

What's left ?

With the basics of the Costas loop operation in hand and a general idea of how to set the loop's behavior, what's left to do? There remain two major factors that affect loop operation: noise and amplitude variations. All simulations for this series of columns thus far have been both noise free and non-varying in amplitude. As you might predict, examining the effect of both will be the focus of the 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.