ELECTRONOTES
1016 Hanshaw Rd
Ithaca, NY 14850
607-266-8492

APPLICATION NOTE NO. 367

June 2006

SOME COMMENTS ON FIXED-POINT DIGITAL OSCILLATOR DESIGN

What is the difference between a filter and an oscillator? Basically, it is the so-called "Q" of the filter. A "high-Q" filter is a very sharp filter, emphasizing a particular frequency. If we then somehow make the Q infinite, the filter essentially decides to make its own signal, thus becoming an oscillator. In a more mathematical sense, what we are trying to do is place a pair of complex conjugate poles in a particular position. For analog sinewave oscillators, the poles are placed exactly on the $j\Omega$-axis in the s-plane (Fig. 1). For a digital sinewave oscillator, the poles are placed exactly on the unit circle in the z-plane (Fig. 2).

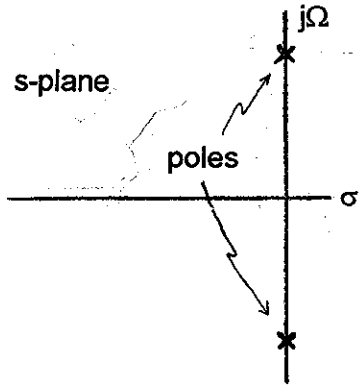

Fig. 1 Analog Sinewave Oscillator

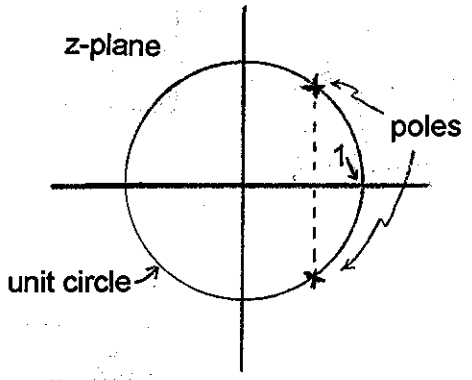

Fig. 2 Digital Sinewave Oscillator

The difficulty of doing this in the analog case can be appreciated by considering that the placement of the poles is a matter of choosing exact values for resistors and capacitors in a network, and using ideal active elements. In practice, components have associated tolerances, and active elements are non-ideal. Without exception, reliable placement of poles can not be done, and we end up with oscillators that are unstable: either they blow up and clip against power supply limits, or they simply die exponentially toward zero. Making a workable analog oscillator is thus a matter of including some sort of amplitude regulation. For example, the poles could be placed ever so slightly inside the right half of the s-plane, so that the oscillator is continually trying to increase its amplitude, while at the same time, secondary circuitry detects peaks and removes "energy" as is appropriate. This is much the same as an adult, monitoring the amplitude of a child vigorously "pumping" a swing, would reach up on occasion and subtract a bit of energy to assure the child's safety.

In fact, the employment of actual analog sinewave oscillators is infrequent in such designs as test-bench oscillators and electronic music equipment. Instead, a "function generator" approach is likely used. For this case, the basic oscillation is not a sine-wave, but rather a ramp-based "triangle" or "sawtooth." The slope of this ramp, and ultimately the frequency, is precisely controlled by a current source driving a capacitor, and the amplitude is controlled by reset or turn-around voltages. This results in a well stabilized, non-sinusoidal oscillation, which is then shaped into a sinewave approximation (as well as other waveforms). Typically this results in harmonic distortions that may be as low as 0.05% or as high as 2%.

In the case of a digital filter network, intended for conversion to an oscillator, one problem which we do not expect to encounter is the variability among individual units. In analog oscillators, one example might have a nominal 10k resistor that was actually 9994Ω while a second example might have a resistor that is 10066Ω, and so on. In the case of a digital network, once we set a multiplier coefficient to, say, 3.67842, all versions will be identical.

There are several issues to consider. The most fundamental issue is whether or not a particular structure and corresponding implementation will actually maintain a sustained oscillation. Secondary issues relate to whether or not the frequency will be the desired value, and to how much harmonic distortion we can expect. Since our preference would likely be to achieve a particular frequency exactly, with zero total harmonic distortion, we need to first ask if a digital sequence with these properties exists.

Of course it does. We have only to consider ordinary sampling. If we want to represent a sinewave $x(t) = A \sin(2\pi ft)$ by a sequence of numbers, we have only to choose:

$$x(n) = A \sin(2\pi fnT) \qquad (1)$$

where A is the amplitude, f is the frequency, and T is a sampling interval that is smaller than 1/2f. That is, the "Sampling Theorem" tells us that if we sample at at least twice the bandwidth of the signal, we can recover x(t). The remaining problems relate to quantization of x(n), and whether or not any particular oscillator structure can give us a satisfactory version of x(n).

Generally speaking, quantization of x(n) from equation (1) results in harmonic distortion. This means that in addition to the frequency f, we expect, for example, smaller components with frequencies 2f, 3f, and so on, approaching but not reaching 1/2T. This distortion may be large at low amplitudes (amplitudes comparable to the quantization interval itself) but may well be minor if the amplitude is large enough (perhaps 100 times the quantization linterval or larger). What we do not expect is any significant deviation from the amplitude and frequency as specified. Some digital sinewave generation methods may actually use look-up tables and interpolation methods to obtain the samples in the manner of equation (1).

Of more interest to us are the simple digital networks that generate the sequence x(n) iteratively based on previous samples. Three such networks are shown in Fig. 3a (Direct Form), Fig. 3b (State-Variable Form) and Fig. 3c (Coupled Form). In general, the State-Variable Form is very similar in performance to the Direct Form. Here we will concentrate on the Direct Form and the Coupled Form which are significantly different in practice. We also need to pay attention to the particular arithmetic proposed. We can get different results with floating point as compared to fixed point; and with fixed point, for example, truncation will differ from rounding.

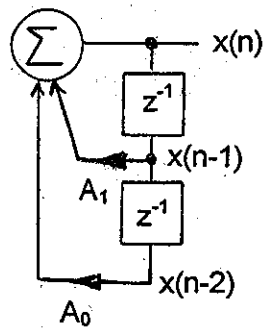

Fig. 3a
Direct Form

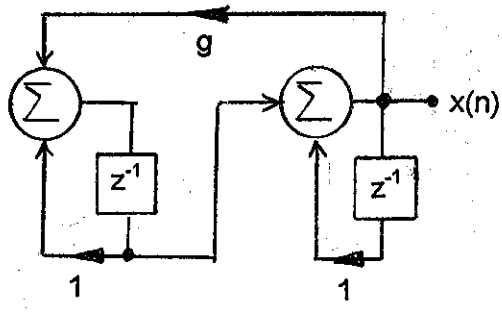

Fig. 3b
State-Variable Form

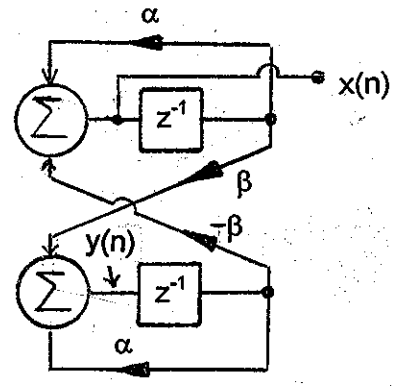

Fig. 3c
Coupled Form

With infinite precision, all three of these networks perform identically, if properly initialized. Clearly Fig. 3a is easy to initialize by setting x(n-1) and x(n-2). In Fig. 3c we might set the x output to x(n), but there is no place to set (and then use) x(n-1) directly (it will be discarded when the network clocks). Instead, we must set y somehow. It is easy to show that y(n) should be set to:

$$y(n) = [x(n-1) - \alpha\, x(n)] / \beta \qquad (2)$$

where we assume that α and β have been set for poles on the unit circle (see below).

With finite precision, the first question of interest is whether or not a stable oscillation can be achieved. This is equivalent to asking if the poles of the network can be set exactly on the unit circle. The poles of Fig. 3a are found as the roots of:

$$1 - A_1 z^{-1} - A_0 z^{-2} \qquad (3)$$

By the quadratic equation, it appears that the complex roots of equation (2) have a radius of exactly 1 if A_0=-1. At the same time, the frequency is determined by the angle of these poles: $\theta = \cos^{-1}(A_1/2)$. So the answer is that for Fig. 3a, we can apparently set the poles exactly on the unit circle, as we expect no error to result from multiplying by -1. In fact, this oscillator is generally satisfactory in actual practice, but perhaps not according to this particular explanation.

The poles of Fig. 3c are easily found to be at $\alpha \pm j\beta$, so the poles should be on the unit circle if $\alpha = \cos(\theta)$ and $\beta = \sin(\theta)$. With finite precision, this is impossible to do, at least in general. No matter how many decimal places (or bits) we have for a and b, we can not set them so that $\sqrt{\alpha^2+\beta^2} = 1$. Accordingly, we expect that we may be able to fine tune α and β so that the network either blows up or dies fairly slowly, but we do not expect to be able to obtain stability. In actual implementations, instability of amplitude is usually observable in a matter of a second or less of actual running, although this if often hard to see by simulating just a few hundred samples of x(n).

Now for the surprises. (1) Perhaps the Direct Form should not work, although it does with both fixed and floating point implementations. (2) Coupled Form does not work with floating point, but it often seems to for fixed point. There is a more or less common explanation for both these surprises, relating to how these "machines" jump through a sequence.

The apparent problem with Direct Form is that while there is no error associated with the multiplication by A_0=-1, there is a roundoff error associated with multiplication by A_1. The results of the two multiplies are added before becoming the newest value of x(n). How does the oscillator "know" that the error came from A_1 and not from A_0? Of course, it can't. It must be true that A_0 does not have to be exactly -1. By actual experiment, we can easily show that the oscillator works with values of A_0 differing slightly (up to 0.3%) from -1 with much larger frequency swings as a result (up to 20%), but nonetheless, stable. This variation in A_0, which does not destabilize Direct From, is much greater than the errors in α and β that do not allow Coupled Form to be stable. Something else must be stabilizing the Direct Form.

Also, as suggested, Coupled Form may sometimes work, with fixed point arithmetic. Typically what sometimes happens (see examples below) is that the output sequence begins by either decaying or blowing up, as expected, but then wonders into a stable oscillation, generally at some amplitude, frequency, and distortion levels away from the ones we intended.

What these two have in common is that to understand them, we need to recognize that each is a "machine" that jumps to a new value based on a limited number of past values. Specifically, the devices are trying to make a small local segment of a sinewave. This jump is followed by some degree of quantization. It is as though you were given instructions to move through a city by moving in a certain direction for a certain distance, and then turning. When the specified distance is achieved, and you are about to turn, you can suppose that you may well not be at an actual intersection. In such a case, you would likely turn at the nearest intersection (or perhaps at the next) thus quantizing your distance instruction. Following a series of such instructions, you would expect to end up near your intended destination, but because of the quantization, you may be a couple of blocks off.

In a similar way, each of the iterations of the oscillator is one step in moving toward the "destination" of producing a sinewave-like sequence. Each state consists of starting from two previously quantized numbers, calculating a jump, and then quantizing the result. Because we can equate a quantization in amplitude to a jump in time, with respect to the intended sinewave, at times we may get slightly ahead of where we might want to be ideally, and at times behind. With a finite number of possible states, with the periodic sinewave, we expect to reachieve our starting point. This guarantees stability and periodicity. At the point of re-linking, we probably have not broken even with respect to gains and losses due to quantization. The result will be a (perhaps very small) frequency error. Further, the particular sequence achieved is not just a matter of oscillator coefficients, but of the choice of initial states. This means that two identical digital oscillators, initialized to different starting states (for example, one for cosine phase and the other for sine phase) may well have slightly different frequencies. When this is true, the set phase relationship will drift apart slowly with time, often destroying a desired result.

We can understand the Coupled Form oscillator working by the fact that as it blows up or decays, it is in some sense searching out a stable sequence (often called a limit cycle) into which it may eventually blunder and be captured. The direct form also ends up on a limit cycle (at least with fixed point), but there is reason to believe that the feedback itself is the major stabilizing influence.

Examples:

The examples given here will show many of the design considerations that we likely encounter during the actual implementation of digital sinewave oscillators. A variety of problems, instability, incorrect frequency, and unwanted additional frequency components (harmonics, secondary frequencies, and sub-harmonics) are illustrated.

Fig. 4a shows a case that turns out to be near-nominal. This case uses the Direct Form structure with an A_1 coefficient selected ($A_1 = 2\cos(\theta)$) such that the separation angle θ between samples should be 30 degrees. Initial conditions set the first two samples of the desired sequence to 10 and 20, and the arithmetic is integer (with rounding). The network returns to the initial state in exactly 12 iterations, so the frequency is exactly what we wanted, and the harmonic distortion is very low.

[In these examples, the time waveform is shown across the top. Usually this is one full cycle of the sequence (which is however often several cycle of the sinewave). The same time-domain sequence is also plotted in the phase-plane (consecutive samples plotted against each other), which gives an simple overall picture of what is going on. The spectrum is shown based on the magnitude of the FFT. Overplotted on the FFT, as an x, is the actual desired frequency.]

Fig. 4b, another Direct Form case, shows again a separation angle of 30 degrees. Here however the oscillator was set to initial samples of 0 and 5. Once again we get the hoped-for 12 samples per cycle, and thus the correct frequency. This case however has noticeable second-harmonic distortion. This can be seen in the time domain as an imbalance between the samples on either side of the peak (e.g., samples 3 and 5 are not the same), and more directly, in the FFT.

As we get to Fig. 4c, Direct Form, we find a more annoying case where we seem to get the correct frequency, but with a noticeable "amplitude modulation." Here we have chosen initial samples as 20 and 30, with a separation angle of 36 degrees. We thus would expect 10 samples per cycle. The actual sequence is length 90 and includes 9 cycles of the sinewave. The rub is that the 9 cycles are not identical. The phase-plane view here perhaps give the best notion of the variability. The FFT shows what looks like a second smaller oscillation, corresponding to 10 cycles at 9 samples per cycle, so that the actual length-90 sequence may be interpreted as a beating of these two components. Another case, not shown is identical to this case except the initial state, here set to 20, is instead set to 21. In this case, the sequence is length 10, not 90, and overall, the result greatly resembles Fig. 4a.

Fig. 4d, our final Direct Form example, shows yet another probematic case where we ask for a separation angle of 31 degrees, starting at 0 and 5. The sequence locks after 3 sinewave cycles or 34 samples total length. At 31 degrees per sample, 34 samples would be 1054 degrees, as compared to 1080 degrees (3 x 360 degrees) for three cycles. The sequence thus links up early, and the frequency is high by about 2.5%. The FFT shows what is now a subharmonic at 1. Different choices of initial conditions, in general, will result in different frequency errors. That is, identical oscillators, initialized to different starting amplitudes and/or phases can result in frequencies that are different, often just by tiny amounts, but different. This can cause major problems, for example, if two oscillators are

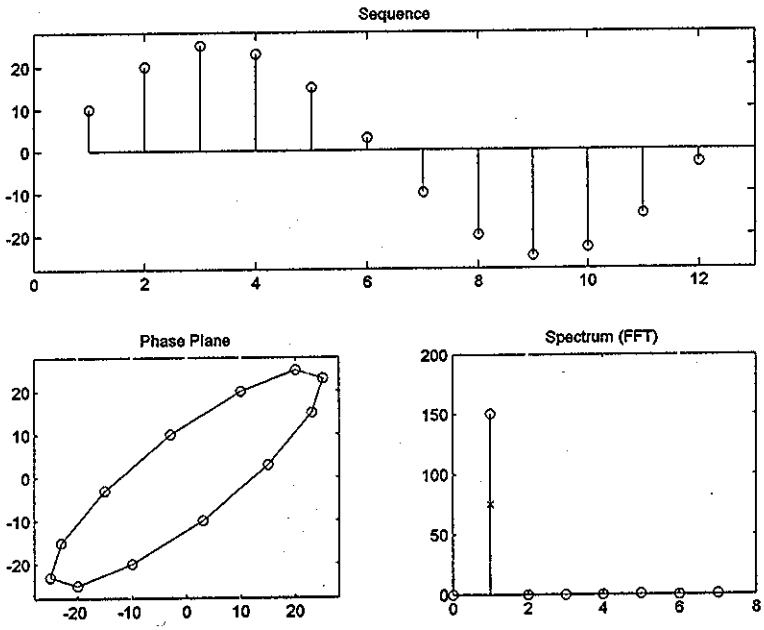

Fig. 4a [x,n,fa]=osc(30,10,20) A "near nominal" case using direct form. Samples are 30° apart resulting in 12 samples/cycle.

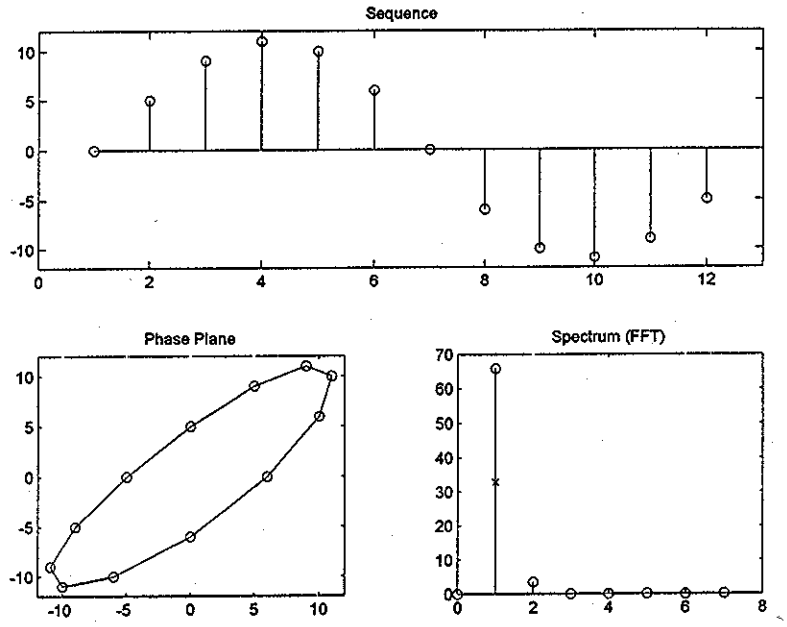

Fig. 4b [x,n,fa]=osc(30,0,5) Here with lower amplitude quantization is more severe, and while we still get the desired frequency, a fair amount of 2nd harmonic distortion is seen in the time domain, and in the FFT.

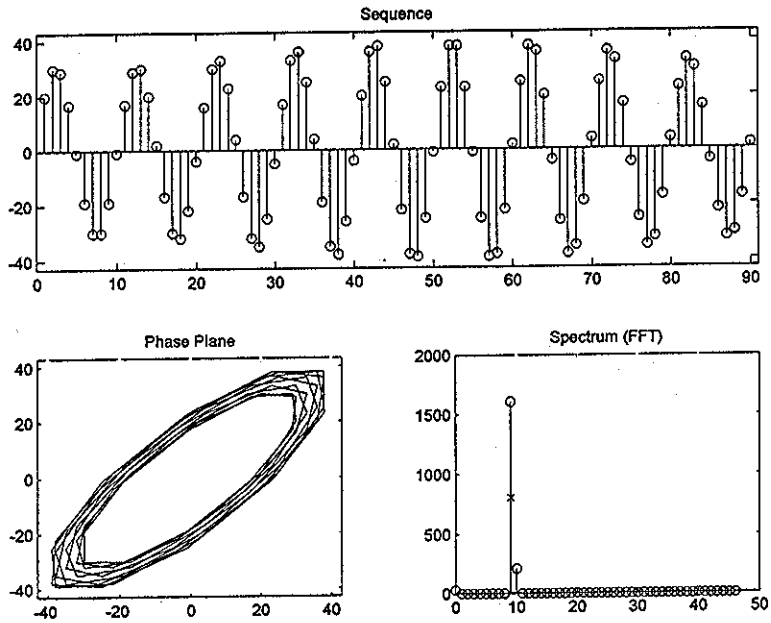

Fig. 4c [x,n,fa]=osc(36,20,30) Here we see a case of direct from which has severe amplitude variations from cycle to cycle. We have the "correct frequency" but as a superposition of 9 cycles (10 samples per cycle) and 10 cycles (9 samples per cycle)

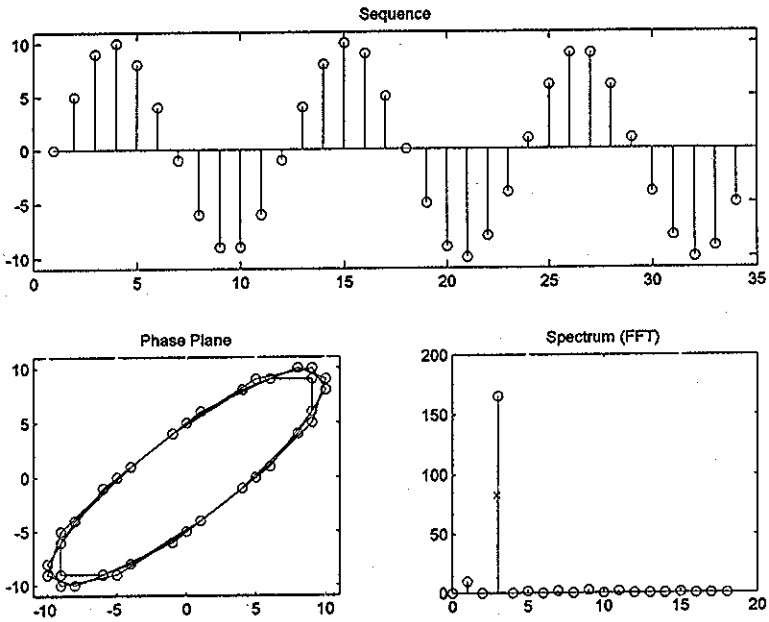

Fig. 4d [x,n,fa]=osc(31,0,5) With 31° separation angle, we expect to have frequency errors in most cases. Here we find three cycles over 34 samples total. The frequency is 2.5% high, and the FFT shows a sub-harmonic.

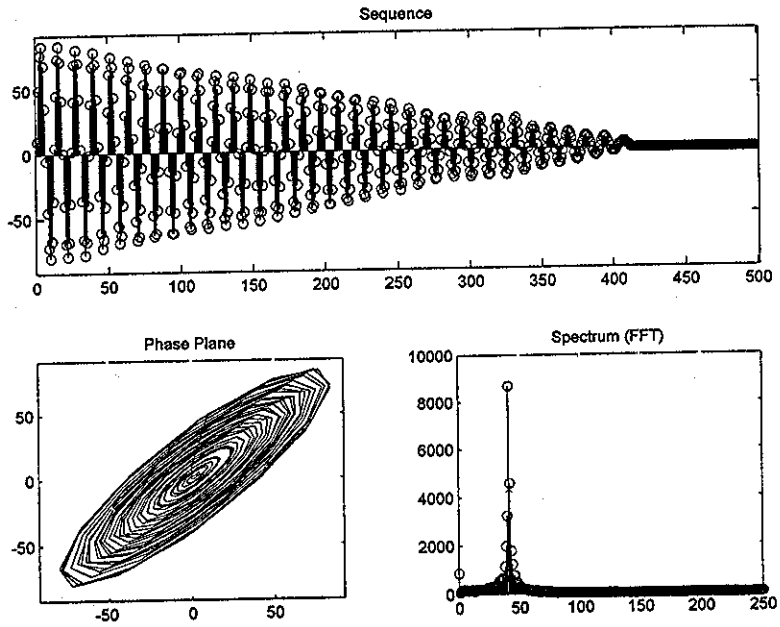

Fig. 5a [x,n,fa]=cfo(30,10,50,500) Coupled form showing decay to 0.

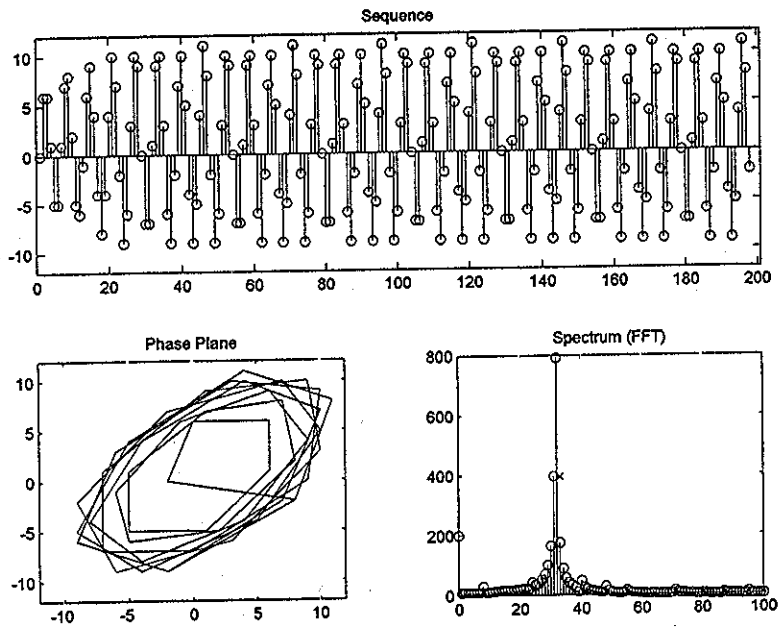

Fig. 5b [x,n,fa]=cfo(60,0,6,200) Coupled form showing an initial blow-op followed by capture in a limit cycle.

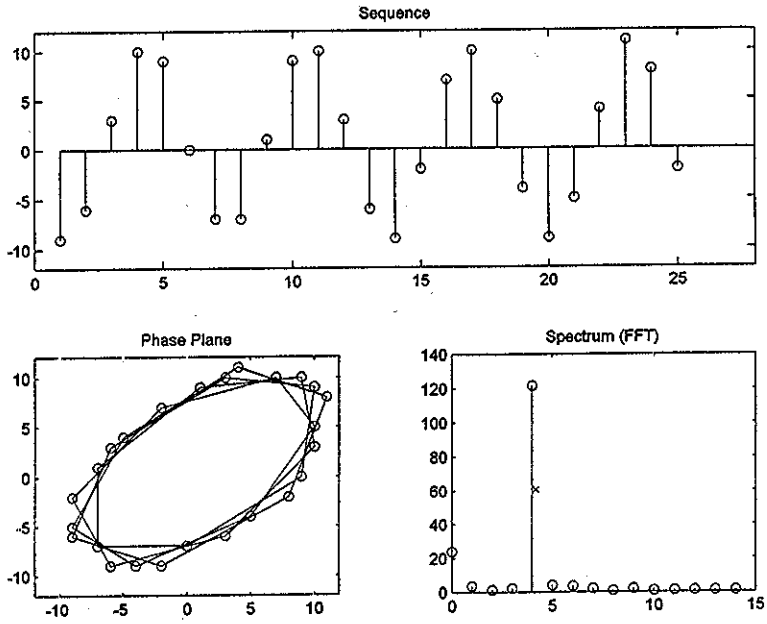

Fig. 5c [x,n,fa]=cfo(60,x(99),x(100),200) By resetting the case of Fig. 5b to samples 99 and 100, we obtain a stable oscillation.

expected to have a certain phase relationship, but this initial relationship drifts as frequency error accumulates into phase error, cycle by cycle.

All and all, the direct form oscillators do at least oscillate (sequences are stable), and usually with relatively minor problems which can be managed, or avoided. The Coupled Form is more problematic. Fig. 5a, a visually attractive case, shows an initialization to 10 and 50, with separation angle of 30 degrees, and the arithmetic is integer (with truncation). This oscillation simply dies, corresponding to poles slightly inside the unit circle. But it is also the case that this oscillation, as it dies, does not fall into a limit cycle what would stabilize the sequence.

Fig. 5b, also Coupled Form, with separation angle of 60 degrees and initial samples of 0 and 6, shows a case where the oscillation begins by blowing up (samples 1 to about 20), and then does find a stabilizing limit cycle. In such a case, the program will not find the network returning to its initial state. (It grew out of that state before it stabilized.) What we can do is simply reinitialize the state; for example, Fig. 5c shows the same network, initialized to outputs 99 and 100 of Fig. 5b. Having now started inside a stable limit cycle, the program does regain the initial state, in this case after 25 samples, or four cycles. With 60 degree separation angle, it should have taken only 24 samples, so the frequency is low by about 4%. Further, the FFT is "dirty" with a strong dc offset.

Reference:

B. Hutchins, "Frequency Errors and Distortion in Digital Sinewave Oscillators," Electronotes, Vol. 18, No. 186, Sept. 1995, pp 3-50

Programs Used for this Note

```
function [x,n,fa]=osc(theta,w10,w00)
%  function [x,n,fa]=osc(theta,w10,w00)
%  theta in degrees
%
%      Digital Oscillator In Direct Form
%
%           SUM ------------> w0
%                              |
%           ^ ^                |
%           | |                |
%           | |                -1
%           | |               Z
%           | |                |
%           | |                |
%           | | 2cos(theta)    |
%           | |------<------ w1
%           |                  |
%           |                  |
%           |                  |
%           |                  -1
%           |                 z
%           |                  |
%           |                  |
%           |        -1        |
%           |--------<------ w2
%
%
%    B. Hutchins                                  Spring 1995


fd=theta/360;              % desired frequency based on cos(theta)
theta=theta*pi/180;
w0=w00;
w1=w10;
a1=2*cos(theta);

for m=1:20000
     w2=w1;
     w1=w0;
     w0=round(a1*w1)-w2;
     x(m)=w0;
       if (w1==w10 & w0==w00)
       n=m
       break
       else end
end
```

```
x=[x(length(x)-1),x(length(x)),x(1:(length(x)-2))];
figure(1)
subplot(211)
mx=max(abs(x));
if mx>10; mx=round(1.1*mx);end
if mx<=10; mx=mx+1;end
stem(x)
hold on
plot([1,1000],[0,0]);  %
hold off
axis([0,n+1,-mx mx]);
title('Sequence')
w1=[x, x(1)];
w2=[x(2:length(x)), x(1),x(2)];
%figure(2)
subplot(223)
plot(w1,w2)
axis([-mx mx -mx mx]);
hold;
if n<50; plot(w1,w2,'o'); end
hold off;
title('Phase Plane')
s=(abs(fft(x)));
%figure(3)
subplot(224)
lp=round(length(x)/2+2)
lpp=0:lp-1;
stem(lpp,s(1:lp))
hold;
k=fd*n;
[y,index]=max(s(1:lp));
y=y/2;
plot(k,y,'x')
hold off;
title('Spectrum (FFT)')
fa=(((index-1)/n)/fd);    % actual frequency as peak in FFT
```

```
function [x,n,fa]=cfo(theta,w1nm1,w1n,its)
%  function [x,n,fa]=cfo(theta,w1nm1,w1n)
%  theta in degrees
%
%  Coupled form of Digital Sinewave Oscillator
%
%
%                cos(theta)
%             ----------<-----------
%            |                      |
%            |            -1        |
%          SUM--w1-->-- z    --->---
%            ^                      |
%            |                      |
%           / \ -sin(theta)        \ / sin(theta)
%            |                      |
%            |        -1            |
%            |--<--- z  ---<---w2--SUM
%            |                      ^
%            |                      |
%             ----------->-----------
%                  cos(theta)
%
%   B. Hutchins                            Spring 1995

fd=theta/360;
theta=theta*pi/180;
a=cos(theta);
b=sin(theta);

w1=w1n;
w2=round((w1nm1 - a*w1n)/b)
w10=w1;
w20=w2;
n=its;
x(1)=w1nm1; %
x(2)=w1n;   %
%
for m=3:its  %  was m=1:100
     w2zm1=w2;
     w1zm1=w1;
 %    w1=round(a*w1zm1)-round(b*w2zm1);
 %    w2=round(a*w2zm1)+round(b*w1zm1);
     w1=floor(a*w1zm1)-floor(b*w2zm1);
     w2=floor(a*w2zm1)+floor(b*w1zm1);



     x(m)=w1;
     if (w1==w10 & w2==w20)
     n=m
     break
     else end
end
```

```
%x=[x(length(x)-1),x(length(x)),x(1:(length(x)-2))];
x=x(1:length(x)-2);
figure(1)
subplot(211)
mx=max(abs(x));
if mx>10; mx=round(1.1*mx);end
if mx<=10; mx=mx+1;end
stem(x)
hold on
plot([1,1000],[0,0])
hold off
axis([0, n+1 ,-mx,mx]);
title('Sequence')
w1=[x, x(1)];
w2=[x(2:length(x)), x(1),x(2)];
%figure(2)
subplot(223)
plot(w1,w2)
axis([-mx mx -mx mx]);
hold;
if n<50; plot(w1,w2,'o'); end
hold off;
title('Phase Plane')
s=(abs(fft(x)));
%figure(3)
subplot(224)
lp=round(length(x)/2+2);
lpp=0:lp-1;
stem(lpp,s(1:lp))
hold;

n=n-2;   % try

k=fd*n;
[y,index]=max(s(1:lp));
y=y/2;
plot(k,y,'x')
hold off;
title('Spectrum (FFT)')
fa=(((index-1)/n)/fd);
```