A Study of Sinusoid Generation Using Recursive Algorithms

Juhan Nam

This paper describes an efficient recursive algorithm to realize a sinusoidal oscillator in the digital domain. In general the algorithm calculates successive samples along a sine waveform creating a sinusoid with very low levels of harmonic distortion and noise.

Several previously described recursive algorithms are examined in [1]. Some leave room for optimization while others have problems such as large dynamic range in intermediate nodes or interactions between amplitude and frequency. The waveguide oscillator described in [2] is also examined. While it may appear cheaper with only 1 multiply and 3 additions required per sample, it suffers from both a large dynamic range requirement and a complex amplitude normalization function.

This paper introduces a recursive algorithm called the "elliptical oscillator" which requires only two multiplies and two additions per sample without a large dynamic range requirement when applied over the audio range. A third multiply provides a constant amplitude of oscillation even when the frequency changes arbitrarily. The coefficient for this multiply is calculated using a relatively simple formula which need be evaluated only when the frequency changes.

Coupled Form

In linear algebra, matrix rotation is shown as below.

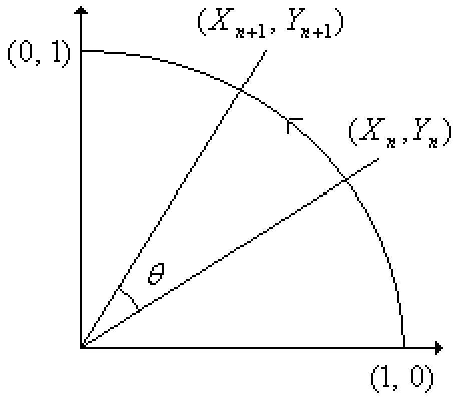

Figure 1. Matrix Rotation

Starting from (X_0, Y_0) = (1, 0), the two state-variables X_{n+1}, Y_{n+1} (where *n* is 0, 1, 2, 3…) can be obtained from the following recursive matrix equations.

$$\begin{pmatrix} X_{n+1} \\ Y_{n+1} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X_n \\ Y_n \end{pmatrix} \quad (\theta \text{ is a constant}) \qquad (1)$$

They move along the unit circle with a constant angular rate. By substituting θ with $2\pi f$, a sinusoidal oscillation with frequency *f* radians is generated on X_n or Y_n. This is called the "coupled form" [1] or CORDIC algorithm [3]. The coupled form guarantees a sinusoid of high quality in a digital system, and the amplitude is fixed for any arbitrary frequency. In addition, all samples move along the unit circle so the computation is free from overflow problems. But, the calculation cost of this algorithm, which requires four multiplies and two additions per sample, is high and leaves room for optimization. Moreover, the generated oscillation could exhibit exponentially decaying amplitude drift because quantization of the two coefficients makes the sum of their squares less than one for most frequencies [2].

Transformation via Ladder/Lattice Filter Structures

(1) can be thought of as a kind of digital two-pair network which has two inputs and two outputs. Based on the idea of two-pair forms of an all-pass filter [4], the coupled form can be represented as Figure 2.

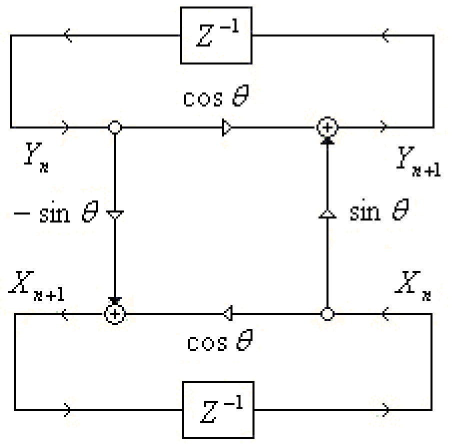

Figure 2. The coupled form represented as a four-multiply structure with delay components

The structure in Figure 2 resembles a four-multiply normalized ladder filter very much. However it is different in that X_{n+1}, Y_{n+1} are fed back to X_n, Y_n via two delay elements to build the recursion. In fact, the advantages of the four-multiply normalized ladder filter are similar to those of the coupled form described in the previous paragraph. With this analogy, other equivalent forms of the four-multiply normalized ladder filter will be applied to recursive structures for sinusoid generation in order to improve calculation efficiency and robustness against quantization errors while preserving the advantages of the coupled form.

In reducing the number of multiplies, the three-multiply ladder structure is derived first. In general, $\sin\theta$ portions are replaced by a parameter *k* and $\cos\theta$ portions are combined into one portion with $1-k^2$ to create the three-multiply ladder filter. But, the transformation can be regarded as simply redistributing the weight of each node in a rather asymmetrical way such that $\sin\theta$ is not required to match to *k*. In the following paragraphs, three cases that have different layouts of *k*-based parameters among the 4 branches will be derived and analyzed. Although their outputs are similar, their numerical characteristics are quite different. There are actually more ways to arrange the *k*-based parameters, but they all reduce to one of three cases.

< CASE I >

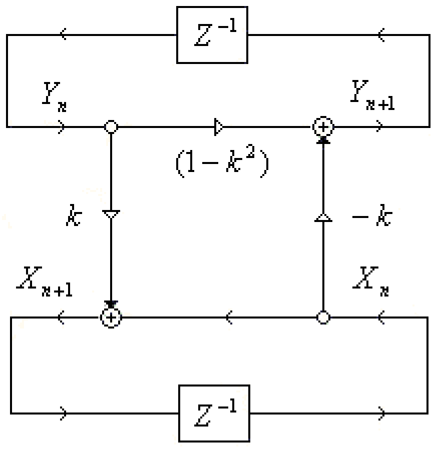

Figure 3. CASE I: three-multiply structure

In Case I, k and $-k$ are put in the place of $-\sin\theta$ and $\sin\theta$ in Figure 2, respectively and the two $\cos\theta s$ are combined into $(1-k^2)$ as in the top portion of Figure 3. Then from the resulting three-multiply structure, the recursive equations of (1) are transformed as follows.

$$X_{n+1} = X_n + k^* \; Y_n, \tag{2a}$$
$$Y_{n+1} = -k^* X_n + (1-k^2)^* Y_n \tag{2b}$$

(2b) can be manipulated to include the terms of (2a):

$$Y_{n+1} = -k^*(X_n + k \; ^*Y_n) + Y_n \; = -k^* X_{n+1} + \; Y_n \tag{2c}$$

This numerical manipulation deletes the $1-k^2$ coefficient and saves one multiply. This then leads to another transformation from a three-multiply structure into a two-multiply structure without moving weights, using (2a) and (2c). The two-multiply structure is shown in Figure 4 below. It resembles a two-multiply IIR lattice filter. The recursive equation (2a) and (2c) are the “magic circle” [1].

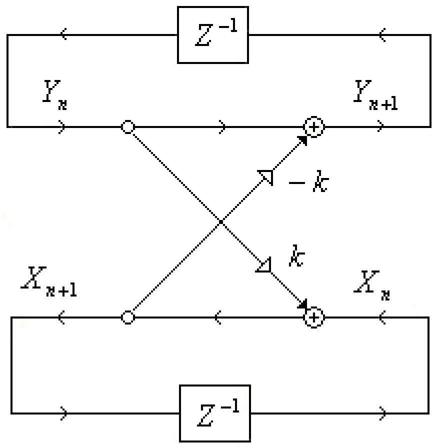

Figure 4. CASE I: two-multiply structure

< CASE II >

In the second case, k is put in the place of $\cos\theta$, $\sin\theta$ and $-\sin\theta$ are combined into $-(1-k^2)$ in the left portion. Figure 5(a) shows the resulting three-multiply structure. The equations from Figure 5(a) can be manipulated as in Case I resulting in the two-multiply structure in Figure 5(b).

$$Y_{n+1} \;=\; X_n + k{*}Y_n \tag{3a}$$
$$X_{n+1} = k{*}X_n - (1-k^2){*}Y_n = k{*}(X_n + k{*}Y_n) - Y_n = k{*}Y_{n+1} - Y_n \tag{3b}$$

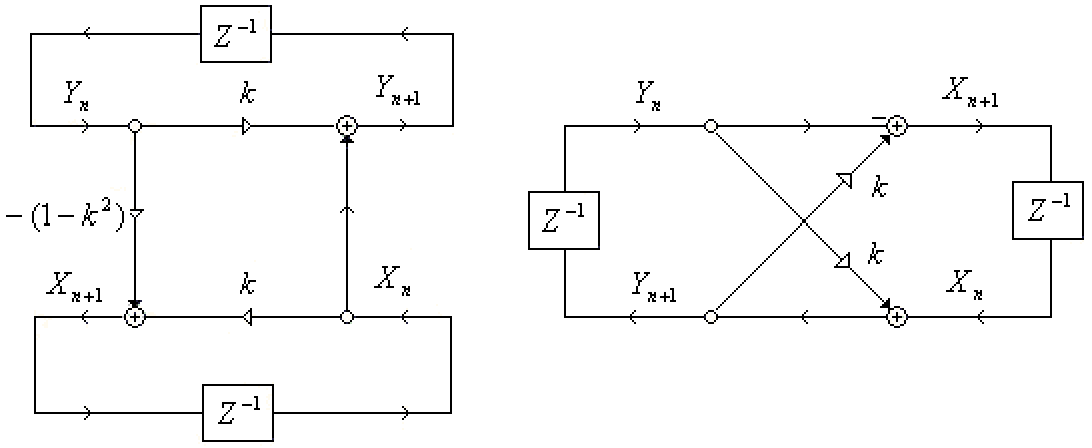

Figure 5. CASE II: (a) three-multiply and (b) two-multiply structure

< CASE III >

In the last case, k is put in the place of $\cos\theta$ as in Case II, but $\sin\theta$ and $-\sin\theta$ are combined into $-(1-k^2)$ on the right portion as in Figure 6(a). The recursive equations are below.

$$X_{n+1} = k{*}X_n + Y_n, \tag{4a}$$
$$Y_{n+1} = k{*}Y_n - (1-k^2){*}X_n = k{*}(Y_n + k{*}X_n) - X_n = k{*}X_{n+1} - X_n \tag{4b}$$

Then from (4a) and (4b), the two-multiply structure in Figure 6(b) can be obtained.

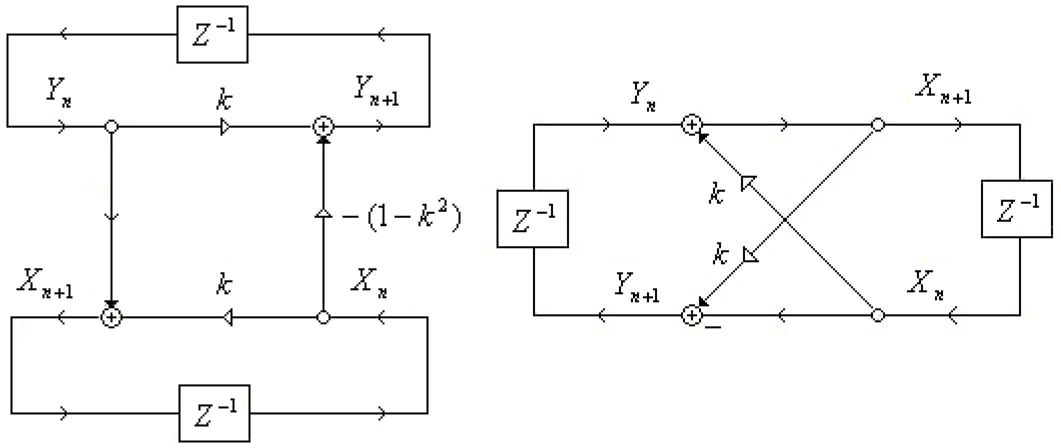

Figure 6. CASE III: (a) three-multiply and (b) two-multiply structure

Now three different two-multiply structures have been derived from three-multiply structures. To evaluate and compare them, the next section will first prove that they generate sinusoidal oscillation using filter-based analysis. Then, for each case, the frequency coefficient formula is determined and the amplitude in each node is calculated to compare the numerical dynamic range. The most desirable structure will have an easy to evaluate frequency coefficient formula and a relatively small dynamic range at frequency extremes.

Analysis by Viewing as a Filter

In the recursive equations in (2), (3) and (4), X_n and Y_n are initialized to 1 and 0 respectively to trigger oscillation. Assuming that the initial value of X_n is an impulse input inserted between the output X_n and the delay element, the two-multiply structures in Figures 4, 5(b) and 6(b) can be interpreted as IIR filters as illustrated in Figure 7.

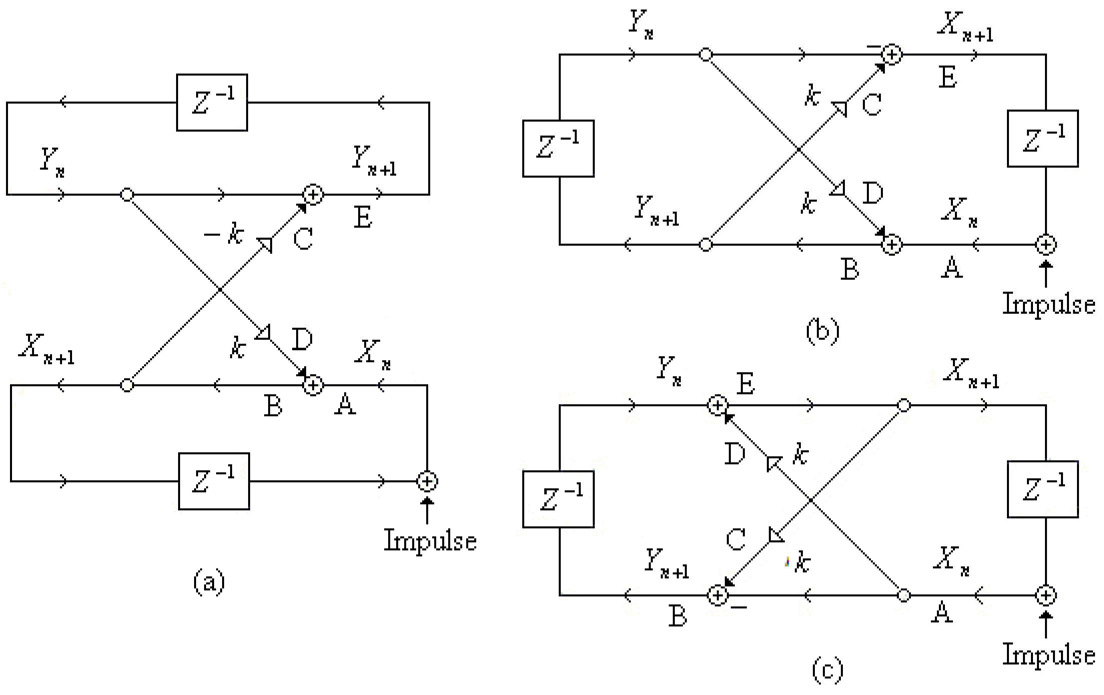

Figure 7. two-multiply structures viewed as an IIR filter with an impulse input
(a) CASE I, (b) CASEII, (c) CASE III

In general, a two-pole IIR filter generates constant oscillation when its poles are located on the Z-plane unit circle. So, by examining the location of poles in the three filters derived from the two-multiply structures, it can be proven that the output from the recursive equations (2), (3), and (4) is a sinusoidal oscillation.

In Figure 7, each node after a multiplication or addition is marked as A, B, C, D, and E for the three filters. Then the Z-transform for each marked node in each filter is derived. These results are shown in equations (5a) - (7e) below.

CASE I

$$H1_A(z) = \frac{1-(1-k^2)z^{-1}}{1-(2-k^2)z^{-1}+z^{-2}} \quad (5a) \qquad H1_B(z) = \frac{1-z^{-1}}{1-(2-k^2)z^{-1}+z^{-2}} \quad (5b)$$

$$H1_C(z) = \frac{-k(1-z^{-1})}{1-(2-k^2)z^{-1}+z^{-2}} \quad (5c) \qquad H1_D(z) = \frac{-k^2z^{-1}}{1-(2-k^2)z^{-1}+z^{-2}} \quad (5d)$$

$$H1_E(z) = \frac{-k}{1-(2-k^2)z^{-1}+z^{-2}} \quad (5e)$$

CASE II

$$H2_A(z) = \frac{1-kz^{-1}}{1-2kz^{-1}+z^{-2}} \quad (6a) \qquad H2_B(z) = \frac{1}{1-2kz^{-1}+z^{-2}} \quad (6b)$$

$$H2_C(z) = \frac{k}{1-2kz^{-1}+z^{-2}} \quad (6c) \qquad H2_D(z) = \frac{kz^{-1}}{1-2kz^{-1}+z^{-2}} \quad (6d)$$

$$H2_E(z) = \frac{k-z^{-1}}{1-2kz^{-1}+z^{-2}} \quad (6e)$$

CASE III

$$H3_A(z) = \frac{1-kz^{-1}}{1-2kz^{-1}+z^{-2}} \quad (7a) \qquad H3_B(z) = \frac{k^2-1}{1-2kz^{-1}+z^{-2}} \quad (7b)$$

$$H3_C(z) = \frac{k(k-z^{-1})}{1-2kz^{-1}+z^{-2}} \quad (7c) \qquad H3_D(z) = \frac{k(1-kz^{-1})}{1-2kz^{-1}+z^{-2}} \quad (7d)$$

$$H3_E(z) = \frac{k-z^{-1}}{1-2kz^{-1}+z^{-2}} \quad (7e)$$

In the Z-transforms above observe that the coefficient of z^{-2} in every node of every case is 1. This implies that the poles of all of the Z-transforms are located on the unit circle. Therefore X_n and Y_n are the outputs of a constant amplitude, self-oscillating IIR filter; a pure sinusoidal oscillation.

Thus the desired result is obtained with a calculation cost of just one coefficient, two multiplies and two additions per sample. This is much cheaper than the original coupled form.

The frequency of oscillation of the self-oscillating filter is determined by the location of its poles. So the frequency coefficient of the sinusoidal oscillator can be obtained from the poles of the Z-transforms of its nodes (5), (6) and (7). From the denominator $1-(2-k^2)z^{-1}+z^{-2}$ in CASE I and $1-2kz^{-1}+z^{-2}$ in CASE II and III, the poles are calculated like this:

$$Z_p = \frac{2-k^2}{2} \pm j\frac{\sqrt{4k-k^2}}{2} \qquad \text{in CASE I} \qquad (8a)$$

$$Z_p = k \pm j\sqrt{1-k^2} \qquad \text{in CASE II, III} \qquad (8b)$$

When the poles are represented in the form of $Z_p = \cos\omega \pm j\sin\omega$, the relation between k and the frequency is given as:

$$Z_p = \frac{2-k^2}{2} \pm k\frac{\sqrt{4k-k^2}}{2} = \cos\omega \pm j\sin\omega \quad \text{in CASE I,}$$

$$k = 2\sin\frac{\omega}{2} \qquad (9a)$$

$$Z_p = k \pm j\sqrt{1-k^2} \;=\; \cos\omega \pm j\sin\omega \quad \text{in CASE II and III,}$$

$$k = \cos\omega \qquad (9b)$$

($\omega = \frac{2\pi f}{fs}$ where f is the frequency of oscillation and fs is the sampling rate.)

The results in (9a) and (9b) are the same as the parameters of the "magic circle" and the direct-from resonator, respectively [1].

Analysis of Numerical Dynamic Range

Thinking ahead to actual use of these algorithms in real hardware, an important attribute is the dynamic range of the values at the nodes as a function of oscillation frequency. A relatively small dynamic range will lessen the numeric precision required in a hardware implementation.

Now, having formulas for the frequency coefficients, the amplitudes of the numeric values at all of the nodes can be calculated and from those, the dynamic range. This is done using the Z-transforms developed above for each of the nodes. When (5) ~ (7) are transformed to the time domain, they all share the following difference equation:

$$y[n] + a1y[n-1] + y[n-2] = b1x[n] + b2x[n-1] \tag{10}$$

Note that $x[n]$ and $y[n]$ in (10) are different from X_n and Y_n. $x[n]$ is the impulse input and $y[n]$ is the output of each node. The output of each filter for the impulse input is a sinusoidal oscillation which can be represented as $y[n] = A\cos(\omega n + \lambda)$. This is also a homogeneous solution of the difference equation when A and λ are determined by the initial conditions from the coefficients $b1$ and $b2$ in the right term of (10). By setting $n = 0,\ \ n = 1$ and the impulse input $x[n]$, we have:

$$\begin{aligned} y[0] &= b1x[0] = b1 = A\cos(\lambda) \\ y[1] &= -a1y[0] + b1x[1] + b2x[0] = -a1A\cos(\lambda) + b2 = A\cos(\omega + \lambda) \end{aligned}$$

From the equations above, the amplitude and phase are given as

$$A = \sqrt{b1^2 + \left(\frac{b1\cos\omega + a1b1 - b2}{\sin\omega}\right)^2} \tag{11a}$$

$$\lambda = \cos^{-1}\left(\frac{b1}{A}\right) \tag{11b}$$

By putting all $a1$, $b1$, and $b2$ obtained from the Z-transforms in (5) ~ (7) into (11a) and (11b), formulas for the amplitude and phase of each node in Case I, II and III were obtained. The result is shown in the following table (the phase is calculated only for X_n and Y_n).

From (5a)~ (5e) of CASE I,

$$A_A = A_B = A_E = \frac{\sqrt{2(1-\cos\omega)}}{\sin\omega} = \frac{1}{\cos(\frac{\omega}{2})} \qquad A_C = A_D = \frac{2(1-\cos\omega)}{\sin\omega} = 2\tan(\frac{\omega}{2})$$

$$X_n = \frac{\cos(\omega n)}{\cos(\frac{\omega}{2})} \qquad Y_n = \frac{\cos(\omega n + \lambda)}{\cos(\frac{\omega}{2})} \quad \lambda = \omega + \frac{\pi}{2} \tag{12}$$

From (6a)~(6e) of CASE II,

$$A_A = A_E = 1 \qquad A_B = \frac{1}{\sin\omega} \qquad A_C = A_D = \frac{1}{\tan\omega}$$

$$X_n = \cos(\omega n) \qquad Y_n = \frac{\cos(\omega n + \frac{\pi}{2})}{\sin(\omega)} = \frac{\sin(\omega n)}{\sin(\omega)} \tag{13}$$

From (7a)~(7e) of CASE III,

$$A_A = A_E = 1 \qquad A_B = \sin\omega \qquad A_C = A_D = \cos\omega$$

$$X_n = \cos(\omega n) \qquad Y_n = \sin(\omega)\cos(\omega n + \frac{\pi}{2}) = -\sin(\omega)\sin(\omega n) \tag{14}$$

(12) ~ (14) show that the three two-multiply structures all generate a sinusoidal oscillation but the amplitude and phase of the output and intermediate nodes are different. It is as if the filter topology influences the numerical characteristics of each node [5].

In Case I, the amplitudes of both X_n and Y_n in (12) depend on the frequency of the oscillation, and also the phase difference between X_n and Y_n is not constant (this means that the sum of their squares is not equal to 1). So, it needs extra calculation at initialization to obtain constant amplitude. Also it is complex to normalize the amplitude when frequency arbitrarily changes, because both amplitude and phase are affected by frequency.

On the other hand, in Case II and III, the amplitude of X_n is constant, and the phase difference between X_n and Y_n is fixed to $\frac{\pi}{2}$ like in the coupled form. So, these forms can have constant amplitude for any frequency with the same initial values, and it is easy to evaluate the relative equation between X_n and Y_n. But, while some amplitudes in Case II are the reciprocal form of trigonometric functions which requires quite a large dynamic range in calculation, all amplitudes in Case III are limited to values less than 1.

So in conclusion, although all cases generate a sinusoid in common, Case III is the best choice for numerical characteristics, especially when implemented in fixed-point arithmetic. When Y_n is divided by its amplitude $-\sin\omega$ in (14), the sum of squares of X_n and Y_n is unity so the oscillation of these two variables can be represented as an ellipse when plotted one against the other.

Figure 8 below shows such a trace of X_n and Y_n for three different frequencies. Each starts at $X_0 = 1$ and $Y_0 = 0$ and the sequence of plotted points rotates clockwise around an ellipse. For this reason, the recursive equation of Case III will be called an "elliptical oscillator".

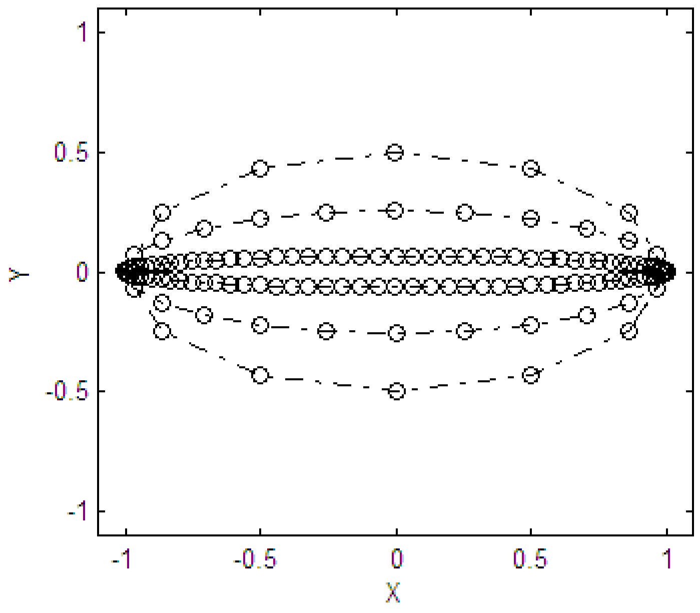

Figure 8. Elliptical oscillator plot of Y_n vs X_n for 500Hz (inner), 2000Hz (middle), and 4000Hz (outer) at 48kHz sampling rate

Tuning Frequency in Real-Time

In the previous section, it was assumed that the frequency of the oscillator was constant. However, for all two-multiply algorithms including the elliptical oscillator, when the frequency changes, the amplitude of the output also changes. Figure 9 shows the effect of a step frequency change on the amplitude for a change from 1278Hz to 2005 Hz.

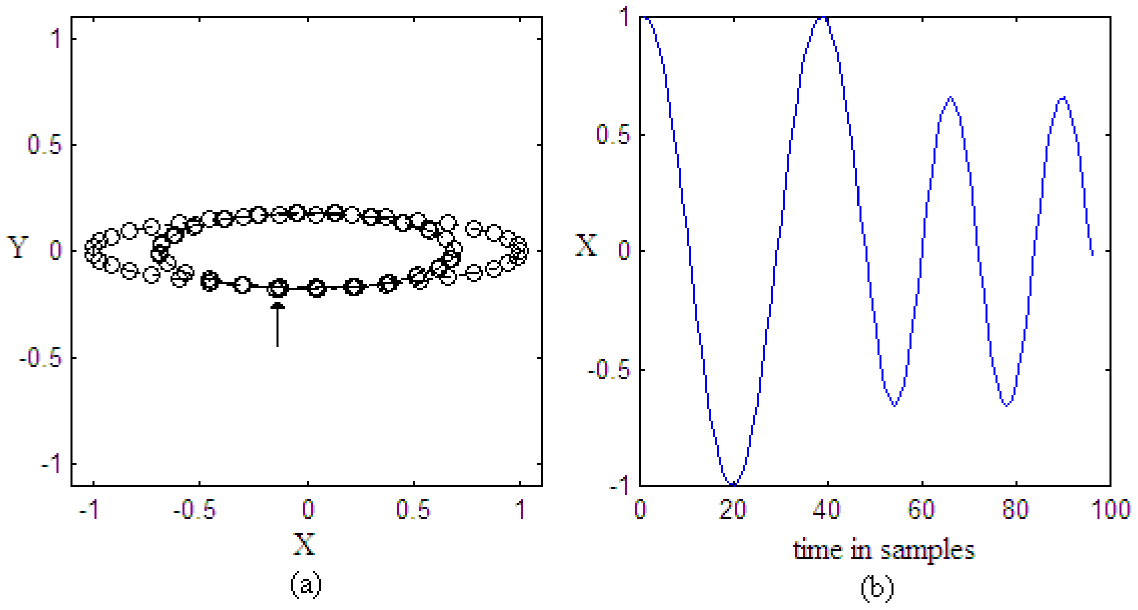

Figure 9. (a) Plot of X_n and Y_n for a change from 1278Hz (outer) to 2005Hz (inner)
(b) Plot of X_n vs time for a change from 1278Hz to 2005Hz

The reason why the amplitude changes can be explained as follows. From (14) and Figure 8, it can be seen that the X_n, Y_n point traces a path along different ellipses for different frequencies. When the frequency changes, the new path must be contiguous with the old path, i.e., the two paths must meet at the point of change. The only points where the paths for all possible frequencies meet is (X_n, Y_n) = (1, 0) and (-1, 0) as can be seen in Figure 8. Actually if the frequency change occurs at exactly one of those points there will be no amplitude change but conforming to such a restriction in practice would be impractical.

When the change occurs at some other point, the new frequency's ellipse that shares that point on the *y*-axis will necessarily have a different X_n excursion (amplitude). This can be seen in Figure 9a above where the arrow marks the point of frequency change.

To avoid this change in X_n excursion, we need to transfer to a point X_n, Y_n on the new ellipse that shares the same point on the *x*-axis as the old ellipse. The transfer is implemented by inserting another multiply into the Y_n path. The new multiply is called an "amplitude coefficient" in [2]. The amplitude coefficient is derived as follows.

If points on the old ellipse and the new ellipse are X_n, Y_n and $X_n^{'}$, $Y_n^{'}$ respectively, and frequency in radians changes from ω to $\omega^{'}$, then from (14)

$$X_n^{\,2} + \frac{Y_n^{\,2}}{\sin^2(\omega)} = 1, \qquad X_n^{'2} + \frac{Y_n^{'2}}{\sin^2(\omega^{'})} = 1$$

At the moment that frequency changes, they share the same point on the *x*-axis, thus $X_n = X_n^{'}$. By putting it into the above equations, $Y_n^{'} = \dfrac{\sin(\omega^{'})}{\sin(\omega)} Y_n$. Therefore, the amplitude coefficient is given as:

$$A(n) = \frac{\sin(\omega(n))}{\sin(\omega(n-1))}\ ,\ \omega(n) = \frac{2\pi f(n)}{fs}(\ fs = sampling\ rate) \qquad (15)$$

When frequency is constant, the amplitude is not modulated so $A(n) = 1$ and the multiply need not be performed. When frequency changes, the amplitude coefficient will deviate from unity for only one time sample in order to normalize the amplitude. This introduces a discontinuity into the Y_n signal but the output waveform, X_n, remains continuous. Figure 10 is the elliptical oscillator with the amplitude coefficient multiplier inserted.

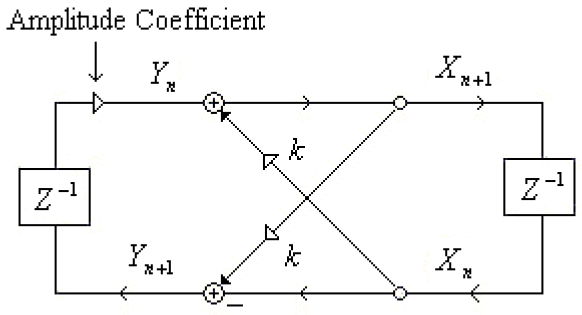

Figure 10. Elliptical oscillator with amplitude coefficient multiplier

If the amplitude coefficient is applied to the conditions of Figure 9, the X_n vs Y_n plot changes as shown in Figure 11(a). Thus the amplitude of X_n remains constant and normalized as shown in Figure 11(b). The arrow in Figure 11(a) shows the point of transfer from the old frequency ellipse to the new frequency ellipse. The step change in Y_n excursion is readily apparent.

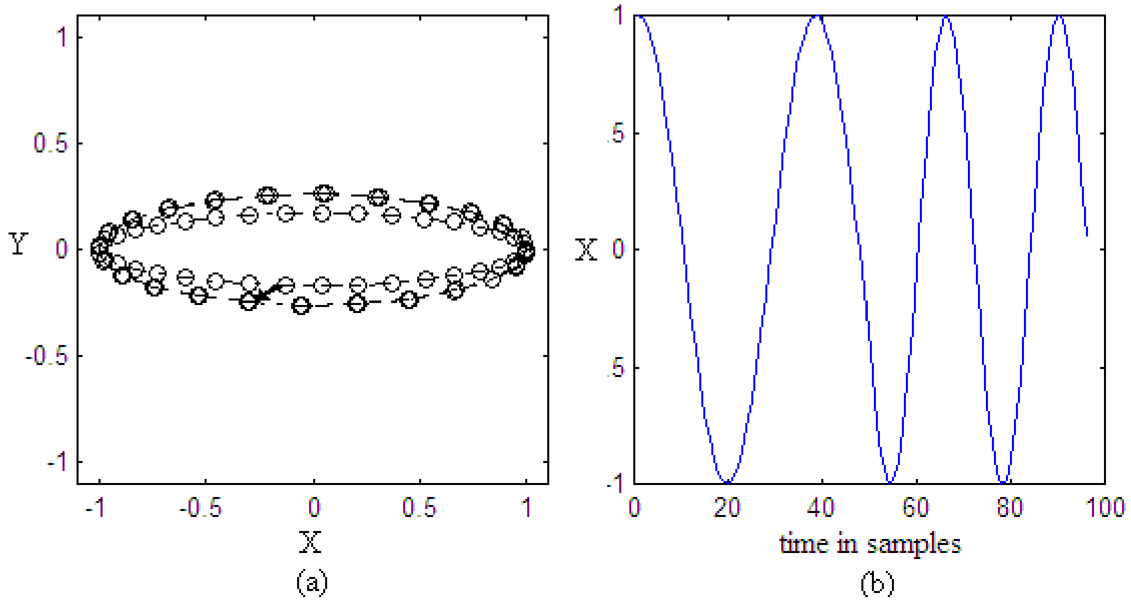

Figure 11. (a) Plot of Y_n vs X_n for a change from 1278(outer) to 2005Hz (inner) with amplitude coefficient (b) Plot of Y_n vs time for a change from 1278 to 2005Hz with amplitude coefficient

Until now we have not been concerned with the phase of the elliptical oscillator's output which is actually a cosine waveform that begins its oscillation from unity. However it is possible to start the oscillation from an arbitrary point on the cosine waveform by properly setting initial values of X_n and Y_n. Figure 12 shows that the initial values X_0 and Y_0 are set to where the X_0, Y_0 point has an angular distance of λ from the *y*-axis.

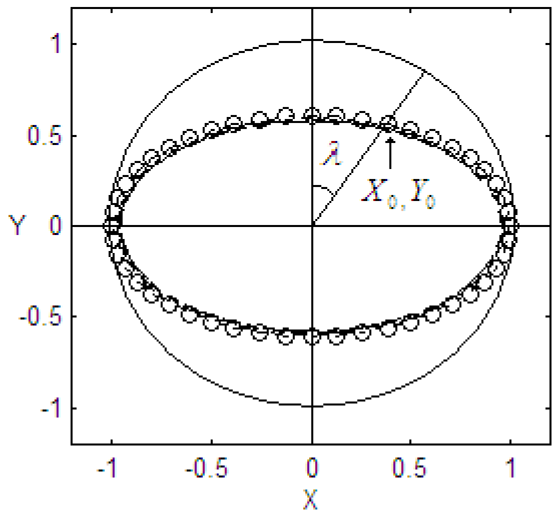

Figure 12. Initial phase of the elliptical oscillator

In (14), X_n and Y_n were initialized to 1 and 0, which is a specific case where λ is $\frac{\pi}{2}$ in Figure 12. So, if (14) is generalized to include λ, X_n and Y_n in (14) can be modified to yield the following formulas.

$$X_n = \cos(\omega n + \lambda - \frac{\pi}{2}) = \sin(\omega n + \lambda)$$

$$Y_n = -\sin(\omega)\sin(\omega n + \lambda - \frac{\pi}{2}) = \sin(\omega)\cos(\omega n + \lambda) \qquad (16)$$

From the fact that X_n is the elliptical oscillator's output, it can be seen that λ is exactly the phase of the output. Sine waveforms starting from zero are desirable for additive synthesis as anything else tends to generate undesirable initial clicks. For the sine wave case, the formula (16) above simplifies to:

$$X_n = \sin(\omega n)\,,\ \ Y_n = \sin(\omega)\cos(\omega n) \qquad (17)$$

So from (17), X_n and Y_n should be initialized to 0 and $\sin(\omega)$ to generate the sine waveform. Note the similarity to (15). In fact initialization can be treated as simply a frequency change from 0Hz at (X_n, Y_n) = (0, 0) to the desired initial frequency. Thus nothing need be added to the structure of Figure 10 and the same frequency change calculation logic can be used for initialization as well.

Analysis of the Noise Created by Continuous Frequency Modulation

Figure A-1 of Appendix A shows that recursive algorithms generate almost noise-free sinusoids when the frequency is constant. In this section the case of continuous frequency modulation will be examined to verify that the amplitude normalization process described above is a complete, noiseless solution to the amplitude modulation problem. For evaluation, frequency modulation of the elliptical oscillator will be compared to frequency modulation of the MATLAB Cosine function. The comparison will be made primarily in the frequency domain where even small deviations from perfection show up as noise between the frequency modulation sideband components.

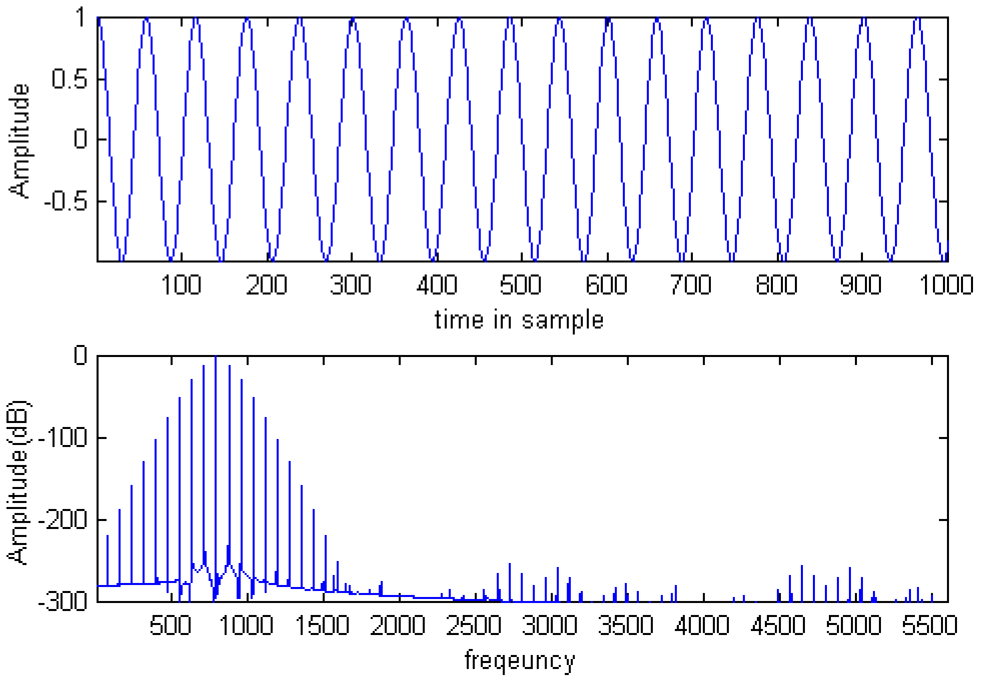

Figure 13. Continuous frequency modulation of the elliptical oscillator (Carrier freq. 800Hz, Modulator freq. 80Hz, Modulation amount 5%)

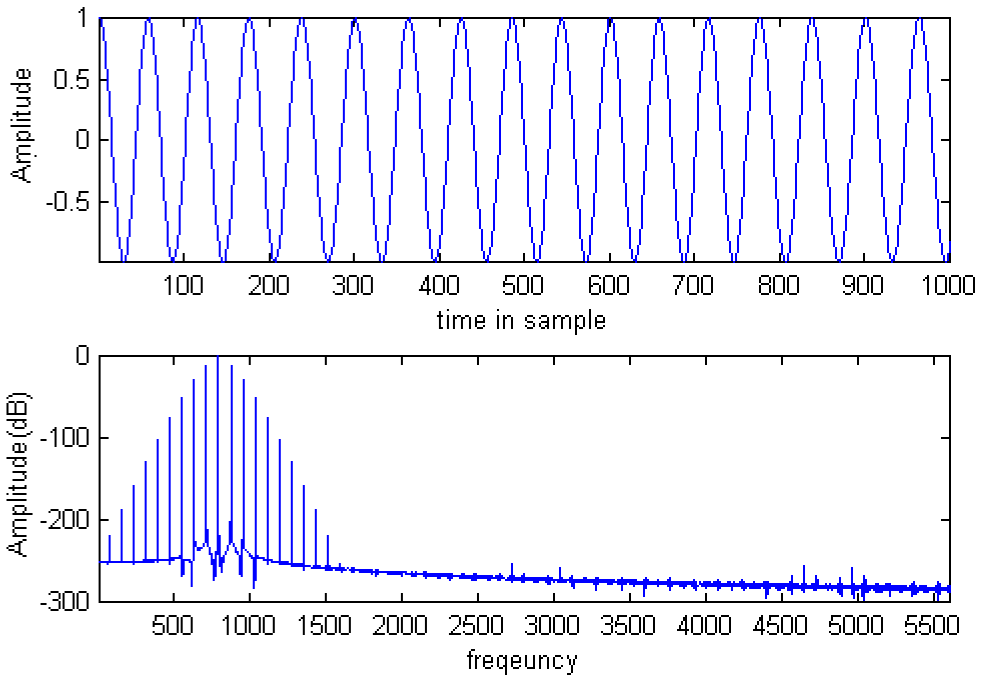

Figure 14. Continuous frequency modulation of the MATLAB Cosine function using the same parameter values as Figure 13.

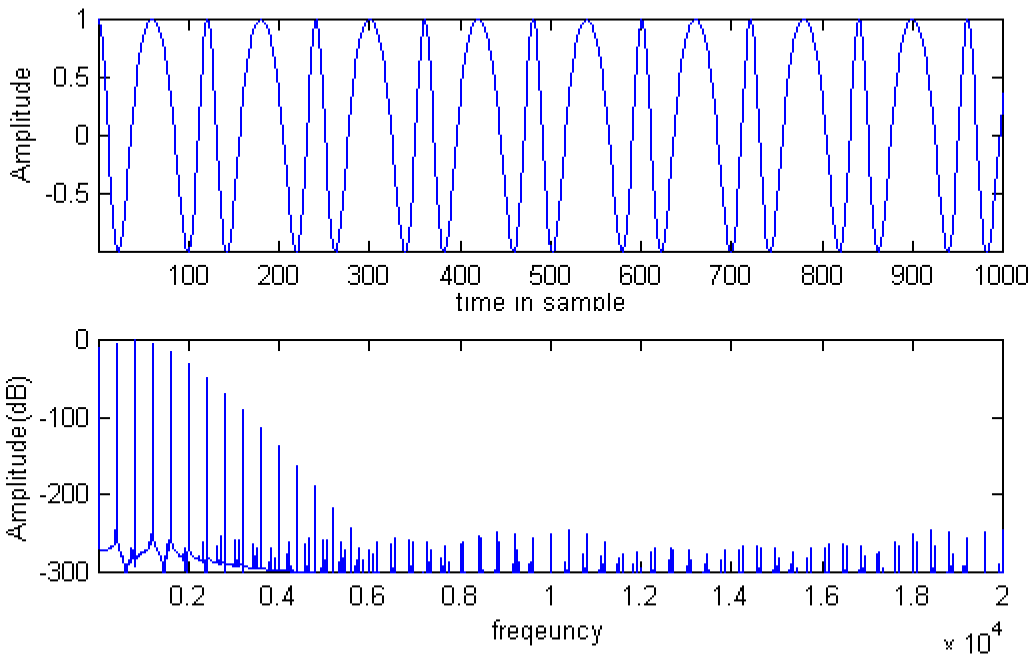

Figure 15. Continuous frequency modulation of the elliptical oscillator (Carrier freq. 800Hz, Modulator freq. 400Hz, Modulating amount 50%)

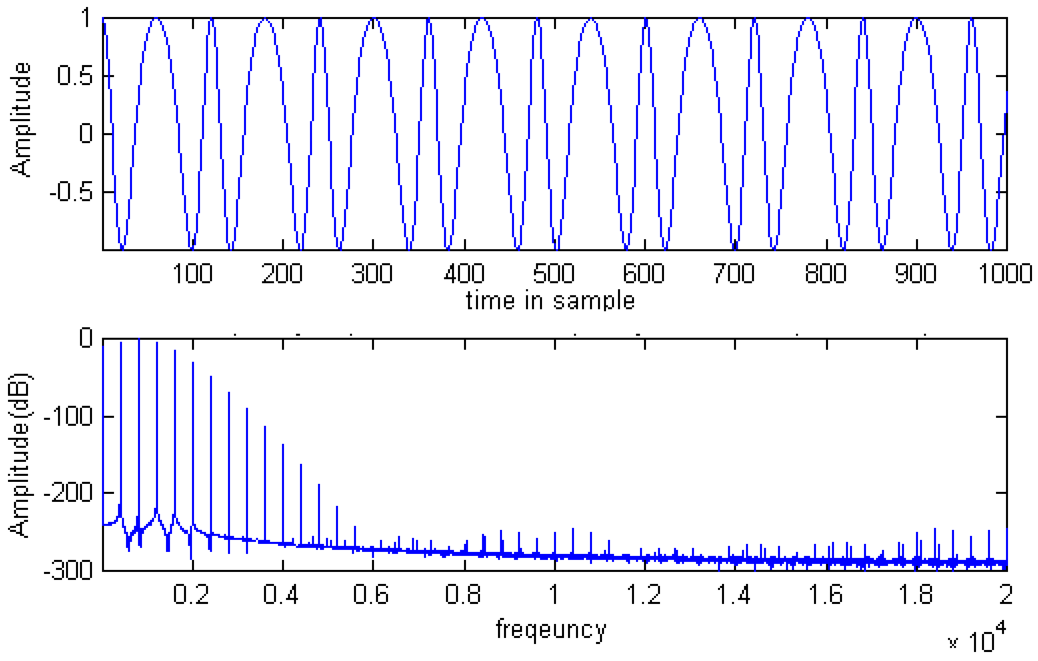

Figure 16. Continuous frequency modulation of the MATLAB Cosine function using the same parameter values as Figure 15.

In this first example, the modulating frequency and amount are quite low (10% of the carrier frequency and 5% amount, respectively). In Figure 13, we can see in the time domain plot that the amplitude is held constant by the normalization coefficient. Also, the frequency spectrum looks very clean except for the expected strong FM sidebands. When compared to Figure 14, some noise is evident in 2700~3200 Hz and 4500~5000 Hz ranges, but its level is well under -250dB and so is completely inaudible. Also, the levels of the FM sidebands are equal above the -200dB line.

The second example compares much more severe frequency modulation (50% of the carrier frequency and 50% amount, respectively). As in the first example, the strong FM sidebands look very similar. Some inharmonic partials above 2000Hz are observed in figure 14, and the overall noise pattern is different. But, the level of noise is still below -250dB so that it will not influence the quality of sound at all. As a result, these two examples show that the spectral purity of the sinusoid generated by an elliptical oscillator is not significantly degraded by arbitrarily changing its frequency.

Effects of Quantization

In digital systems, all variables and coefficients are quantized by the numerical precision used in the implemented system. Recursive algorithms are apt to be influenced by quantization because the round-off error created by quantization is accumulated at every sample. For this reason, the coupled form oscillator could generate an exponentially decaying or increasing output because round-off error in the two coefficients as well as the delay elements propagates around the structure at every sample. It is especially critical in systems with low precision.

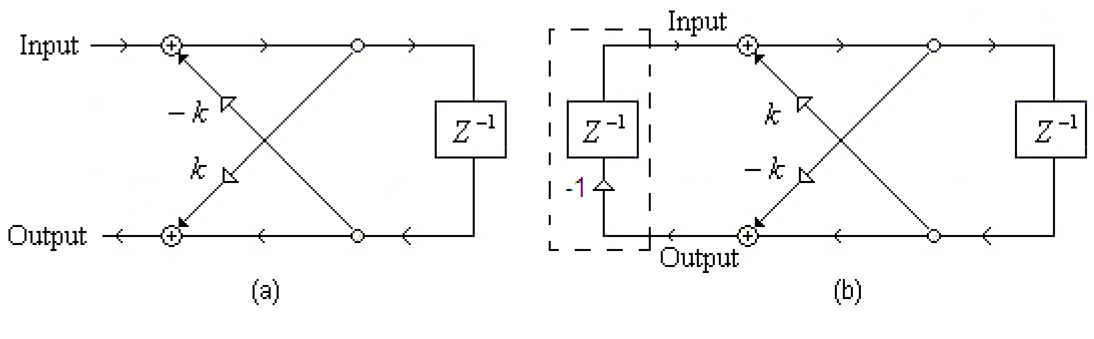

Figure 17. (a) Lattice structure of the first-order all-pass filter
(b) Transformation of (a) with a delay and sign inverter (note that (b) is identical to the elliptical oscillator)

Fortunately, the elliptical oscillator is robust against round-off error. This can be explained by studying the all-pass lattice filter. According to [4], all-pass filters with a two-multiply lattice form have the property that, upon quantizing the multiplier coefficient, the numerator polynomial and denominator polynomial retain a mirror-image relationship, and thus the all-pass characteristic is independent of any multiplier quantization as long as it is the same for both multipliers.

The elliptical oscillator is different from an all-pass lattice filter, but it has the same topology as illustrated in Figure 17. Here, the first-order lattice all-pass filter is transformed into the elliptical oscillator when the output is fed back to the input via a delay element with a sign inversion and the signs of the frequency coefficients are also inverted. The sign inversion and additional delay element do not influence the numerical characteristic of the all-pass filter. Therefore, the elliptical oscillator also has the same numerical characteristic of not being influenced by multiplier quantization. In other words, net accumulation of errors in the multiplications by the frequency coefficient is canceled out during each cycle of oscillation.

In the above paragraph, it is assumed that frequency is constant. When frequency changes, however, the round-off noise by the normalization coefficient can influence the amplitude of the oscillator. But, the multiplication of the amplitude coefficient occurs for only one time sample. In addition, the round-off (not truncation) error is represented as $-0.5 * 2^{-B} < e[n] < 0.5 * 2^{-B}$ (in the operation of quantizing a number to *(B+1)* bits) which is a uniformly distributed random function. So, although there is very low level of amplitude fluctuation, it will be not audible and the average effect would be zero.

Comparison with the Waveguide Oscillator

The second-order digital waveguide oscillator that is introduced in [2] can generate a sinusoidal oscillation with one multiply and three additions per sample for constant frequency, and with an additional multiply for arbitrary frequency. Although the elliptical oscillator needs one more multiply, it needs one less addition and substantially has less dynamic range for all variables in the recursive equation.

Table 1 compares the maximum amplitudes in all of the nodes over the audio range in both the elliptical oscillator and the waveguide oscillator. Maximum amplitudes in the elliptical oscillator are limited to unity as expected from (14), whereas in the waveguide oscillator some variable amplitudes "blow up" as one approaches Nyquist. For example, the maximum amplitude of internal nodes B, C, D and E is about 3.732 for 20kHz at a 48KHz sampling rate and about 6.872 at a 44.1kHz sampling rate. This property will require either extra headroom to allow for word growth in the internal registers, or rescaling of all internal variables to avoid overflow.

Elliptical oscillator			Waveguide oscillator		
	Min	Max		Min	Max
A	1.000000000	1.000000000	A	1.000000000	1.000000000
B	0.002617991	1.000000000	B	0.001308998	3.732050807
C	0	0.999996573	C	0.001308998	3.732050807
D	0	0.999996573	D	1.000000000	3.732050807
E	1.000000000	1.000000000	E	0	3.232050807
			F	1.000000000	1.000000000

Table 1. Amplitude comparison for all nodes of the elliptical oscillator and waveguide oscillator
(The amplitude is the maximum value seen during a 1 second simulation at 48kHz sampling rate)

Conclusions

Recursive algorithms are capable of generating cleaner sinusoidal oscillation with fewer arithmetic operations per sample than other methods. Among several possible recursive algorithms, the elliptical oscillator introduced here can generate a normalized sinusoidal oscillation with two multiplies (three for arbitrary frequency) and two additions per sample. It is capable of clean arbitrary frequency modulation and requires a relatively small dynamic range from the arithmetic and storage elements. Finally it is robust against quantization error. Thus this algorithm is well suited for hardware implementation using fixed-point arithmetic elements with relatively small word lengths as are typically found in VLSI, FPGA and programmable DSP based systems.

Appendix A. Spectral Purity

The spectral plots below demonstrate the spectral purity of the two-multiply elliptical oscillator when generating 440Hz and 4427Hz sine waves at a sampling rate of 48KHz. MATLAB floating-point variables (IEEE floating-point standard. Floating-point numbers have a finite precision of roughly 16 significant decimal digits.) were used in the simulation. The coupled form and waveguide oscillators show very similar purity levels.

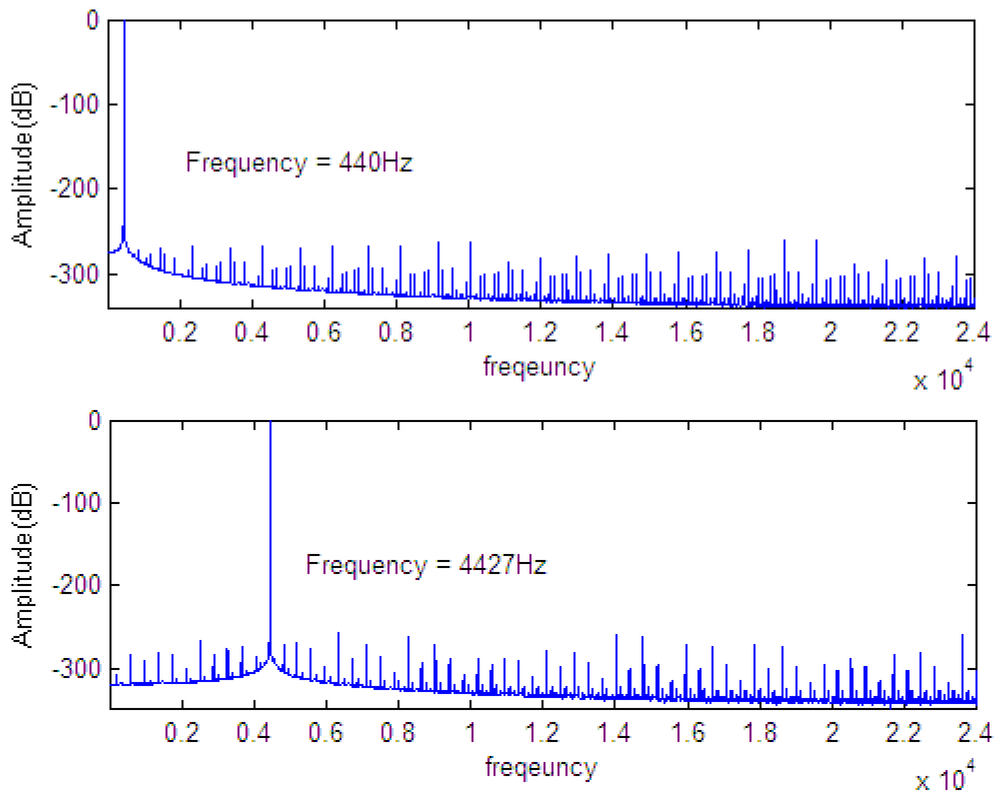

Figure A-1. Spectral purity of constant frequency elliptical oscillator output

Appendix B. MATLAB coding of two-multiply structure (Case III, elliptical oscillator) recursive equations

```
function [x,y] = sine_2mul(freq, fs, t, case)
%Sinusoid Oscillator
%  The sinusoid oscillator is generated by two-multiply form
%
%  function [x,y] = sine_2mul(freq, fs, t, case)
%
%    freq = sine wave form frequency.
%    fs   = sampling rate
%    t    = time in sec
%    case = case number ( refer to my paper on sinusoid generation)

% Juhan Nam, Kurzweil Co.,Ltd.
% History:  Jul-02-2005    v0.1  initial version
%           Aug-05-2005    v0.2  implemented all cases

%fs = 48000;

length = fs*t;
x = zeros(1,length);
y = zeros(1,length);

impulse = zeros(1,length);
impulse(1) = 1;

% frequency coefficients
k1 = cos(2*pi*freq/fs);
k2 = 2*sin(pi*freq/fs);

z1 = 0;
z2 = 0;

for i=1:length
   switch case
   case 1
       a = z1 + impulse(i);
       d = -k2*z2;
       b = a + d;
       c = k2*b;
       e = c + z2;
       x(i) = a;
       y(i) = z2;
       z1 = b;
       z2 = e;
   case 2
       a = z1 + impulse(i);
       d = k1*z2;
       b = d + a;
       c = k1*b;
       e = c - z2;
       x(i) = a;
       y(i) = z2;
       z1 = e;
       z2 = b;
   case 3
       a = z1 + impulse(i);
       d = k1*a;
       e = d + z2;
```

```
      c = k1*e;
      b = c - a;
      x(i) = a;
      y(i) = z2;
      z1 = e;
      z2 = b;
   end
end
```

References

[1] J. W. Gordon and J. O. Smith, “A sine generation algorithm for VLSI applications," in *Proceedings of the 1985 International Computer Music Conference, Vancouver*, Computer Music Association, 1985

[2] J.O. Smith and P.R. Cook, “The Second-Order Digital Waveguide Oscillator”, *Proceedings of the 1992 International Computer Music Conference, San Jose*, 1992.

[3] R. Andraka, “A survey of CORDIC algorithms for FPGA based computers”, *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, Monterey*, CA., 1998.

[4] P. A. Regalia, S. K. Mitra, and P. P. Vaidyanathan, “The Digital All-Pass Filter: A Versatile Signal Processing Building Block”, *Proceedings of the IEEE,* vol. 76, No. 1, 1988.

[5] J. Dattorro. “The Implementation of Recursive Digital Filters for High-Fidelity Audio”, *J. Audio Eng. Soc.,* vol. 36, No. 11, 1988.