PHASESHAPING OSCILLATOR ALGORITHMS FOR MUSICAL SOUND SYNTHESIS

Jari Kleimola[1], **Victor Lazzarini**[2], **Joseph Timoney**[2], **and Vesa Välimäki**[1]

[1] Department of Signal Processing and Acoustics, Aalto University, Espoo, Finland

[2] Sound and Digital Music Technology Group, National University of Ireland, Maynooth, Ireland

jari.kleimola@tkk.fi,victor.lazzarini@nuim.ie,jtimoney@cs.nuim.ie,vesa.valimaki@tkk.fi

ABSTRACT

This paper focuses on phaseshaping techniques and their relation to classical abstract synthesis methods. Elementary polynomial and geometric phaseshapers, such as those based on the modulo operation and linear transformations, are investigated. They are then applied to the generation of classic and novel oscillator effects by using nested phaseshaping compositions. New oscillator algorithms introduced in this paper include single-oscillator hard sync, triangle modulation, efficient supersaw simulation, and sinusoidal waveshape modulation effects. The digital waveforms produced with phaseshaping techniques are generally discontinuous, which leads to aliasing artifacts. Aliasing can be effectively reduced by modifying samples around each discontinuity using the previously proposed polynomial bandlimited step function (polyBLEP) method.

1. INTRODUCTION

The generation of complex musical timbres has been approached from various angles in sound computing. One elegant solution, which has provided a wide scope for research and implementation, has been that of distortion techniques. Within this area, various techniques have been put forward, such as frequency modulation (FM) [3], phase distortion (PD) [5,9], nonlinear waveshaping [1,10,14,16], and discrete summation formulae (DSF) [11]. These are in many cases equivalent and can be used as alternative ways to describe and implement a given algorithm, as discussed in [7].

In particular, the waveshaping method provides a computationally simple means to produce potentially rich spectra. Its principle is quite straightforward, starting with a discrete-time sinusoidal signal,

$$x(n) = \sin(\omega n), \tag{1}$$

where ω is the angular frequency and n is the discrete sample index, a complex (i.e., non-sinusoidal) spectrum can be obtained via a mapping such as

$$y(n) = f\left[x(n)\right], \tag{2}$$

where $f[.]$ is an arbitrary nonlinear function called a *waveshaper*. The well-known classic FM synthesis equation, for instance, can be rewritten as a waveshaping expression

$$\begin{aligned}&\cos\left[\omega_c n + I\,x(n)\right] = \\ &\cos(\omega_c n)\cos\left[I\,x(n)\right] - \sin(\omega_c n)\sin\left[I\,x(n)\right],\end{aligned} \tag{3}$$

where ω_c is the carrier frequency and the waveshapers cos(.) and sin(.) act on the sinusoidal modulation signal $x(n)$ of Equation 1.

Similarly, it is possible to describe PD as a form of waveshaping. This is demonstrated by starting with the following expression defining a sinusoidal oscillator:

$$y(n) = \cos\left[2\pi\phi(n)\right]. \tag{4}$$

The function $\phi(n)$ is the normalized phase defined by

$$\phi(n) = \left[\phi(n-1) + f_0 / f_s\right] \bmod 1, \tag{5}$$

where f_0 is the fundamental frequency, f_s is the sampling rate, and $x \bmod 1 = x - \lfloor x \rfloor$, and $\lfloor x \rfloor$ is the floor function denoting the largest integer that is not greater than x. To implement a PD oscillator, the phase is then applied to a nonlinear function $g(x)$:

$$y(n) = \cos\left\{2\pi\, g\left[\phi(n)\right]\right\}. \tag{6}$$

A linear $g(x)$ would result in a sinusoid whose frequency is transposed. However, with nonlinear $g(x)$, the shape of the output waveform is modified.

From a waveshaping perspective, Equation 4 can be described as a sinusoidal waveshaper acting on a complex input waveform $s(n) = 2\pi\phi(n)$:

$$y(n) = f\left[s(n)\right]. \tag{7}$$

This transforms the phase signal $s(n)$ into the output signal $y(n)$ by means of a waveshaper $f(x) = \cos(x)$. The waveshaper can be implemented as a function or as a lookup table that acts on the normalized value of the phase signal. The typical phase generator producing $s(n)$ is the unipolar modulo counter $\phi(n)$ of Equation 5, which is also a unipolar geometric non-bandlimited sawtooth wave.

In this vein, Equation 6 can be seen as based on a form of double waveshaping, where two functions, $g(x)$ and $\cos(x)$, are applied to an input sawtooth wave $\phi(n)$. This is perfectly equivalent to the principle of distorting the phase function $\phi(n)$ of a sinusoidal oscillator.

In this paper, the term ‘phaseshaping’ [7] is used to describe the generalization of the phase function distortion $g[\phi(n)]$. The aim here is to investigate elementary polynomial and geometrical phaseshapers, and then discuss their application in classic and novel oscillator algorithms.

2. ELEMENTARY PHASESHAPERS

The investigation is began by proposing two fundamental phaseshaping concepts, entitled nested phaseshaping and phaseshaper entities. *Nested phaseshaping* is related to function composition, in which the result of the inner function serves as the input to the outer function. Equation 8 shows an example of nesting at three levels, expressed in the basic form in Equation 8a and its equivalent shorthand notation in Equation 8b.

$$y(n) = f\{g[h(x)]\}, \tag{8a}$$

$$y(n) = f \circ g \circ h(x). \tag{8b}$$

For the purposes of this paper, x is assumed to be a signal which flows from the inner function towards the outer ones, transforming at each step into the final shape given by the outmost function. The graphical representation of this composition is thus a signal block diagram, similar to the one shown in Figure 1.

$$\begin{tikzcd} \boxed{x(n)} \arrow[r] & \boxed{h} \arrow[r] & \boxed{g} \arrow[r] & \boxed{f} \arrow[r] & y(n) \end{tikzcd}$$

Figure 1. Graphical representation of Equation 8.

It is further assumed that the source of the chain is the unipolar modulo counter $\phi(n)$ of Equation 5, and that the rightmost block is the waveshaper producing the final output signal. The blocks or functions between these two extremes are called phaseshapers, because they act on the phase signal $\phi(n)$ and because the input of the final waveshaper is essentially a phase signal as well. Having said this, note that in some cases the output of the chain is the phase signal itself instead of the product of the waveshaper.

Phaseshaper entities are frequently used phaseshapers that have fixed predefined semantics. These include

$$\text{mod}_1[x(n)] = x(n) \bmod 1 \tag{9a}$$

$$\text{mod}_m[x(n), m] = x(n) \bmod m \tag{9b}$$

$$g_\text{b}[x(n)] = 2x(n) - 1 \tag{9c}$$

$$g_\text{u}[x(n)] = 0.5x(n) + 0.5, \tag{9d}$$

where mod_1 is the modulo-1 operation, mod_m is the real-valued modulo-m operation ($m \in$ R), g_b is the bipolar transformation converting a unipolar signal into its bipolar form, and g_u its opposite unipolar transformation.

2.1 Ramp-like Fractional Period Phase Signals

Phaseshaper entities g_b and g_u are linear transformations, whose general expression is given by the phaseshaper

$$g_\text{lin}[x(n), a_{1,0}] = a_1 x(n) + a_0 , \tag{10}$$

where a_1 and a_0 are the scaling and shifting factors, respectively. Assuming that $x(n)$ is given by $\phi(n)$ – which is restricted to values between 0 and 1 – one notices that the output of g_lin is no longer constrained to the range [0,1], which is the expected normalized phase range of most waveshaper terminals of the shaper chain.

The output of g_lin should therefore be normalized. One way of doing this is to apply the mod_1 phaseshaper entity to obtain

$$y(n) = \text{mod}_1 \circ g_\text{lin}[x(n), a_{1,0}]. \tag{11}$$

The effect of this normalization is seen in Figure 2, which plots the output of Equation 11 using parameter values a_1 = 1.5 and a_0 = 0. The sampling rate f_s = 44.1 kHz is used in all examples of this paper. In this example, the modulo operation is activated first within the context of mod_1 (producing the full-height phase cycle) and then within the context of g_lin (producing the fractional phase cycle)[1]. Parameter a_1 thus controls the length of the phase period (when a_1 > 1) or the slope of the phase signal (when a_1 < 1). The shifting term a_0 contributes to the DC offset of the produced phase signal.

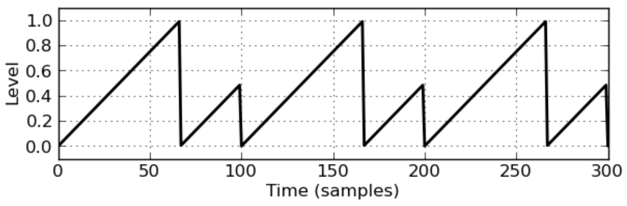

Figure 2. Ramp-like phase signal with a fractional phase period (a_1 = 1.5 and a_0 = 0). The fundamental frequency is f_0 = 441 Hz, as in all plots of this section.

2.2 Triangular Fractional Period Phase Signals

The unipolar modulo counter signal $\phi(n)$ can be transformed into a bipolar sawtooth waveform by applying the g_b phaseshaper entity of Equation 9c. Then, by feeding this sawtooth waveform through the absolute value function, a unipolar triangular signal [20] is obtained, which can be further shaped by g_lin and mod_1 to get phaseshaper

$$g_\text{tri}[x(n), a_{1,0}] = \text{mod}_1 \circ g_\text{lin} \circ \text{abs}\{g_\text{b}[x(n)]\} . \tag{12}$$

Alternatively the abs{.} term of Equation 12 can be replaced with the piecewise linear triangular waveform definition

$$s_\text{tri}(x) = \begin{cases} 2x, & \text{when } 0 \le x < 0.5, \\ 2 - 2x, & \text{when } 0.5 \le x \le 1. \end{cases} \tag{13}$$

The fractional period phase signal produced by g_tri is depicted in Figure 3, which shows that because the slope of the triangle wave is two times steeper than that of a

[1] Here the term *phase cycle* is adapted to describe the segment that takes the phase value from 0 to 1, and the term *phase period* to describe the total period of the modulo counter signal $\phi(n)$.

sawtooth, the frequency of the phase cycle is doubled. The phase period of g_{tri} therefore contains two complete periods of Equation 11 and, as expected, the latter period is reversed in time. Because of this symmetry, g_{tri} produces less dramatic effects on the output of the shaper chain.

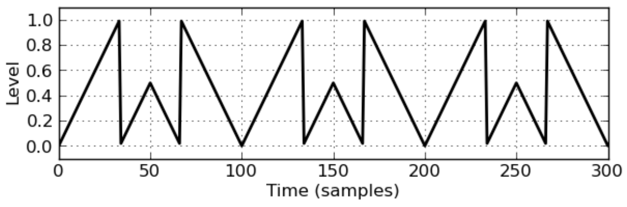

Figure 3. Triangular phase signal with a fractional phase period (a_1 = 1.5 and a_0 = 0).

2.3 Rectangular Signals

The unipolar modulo counter signal $\phi(n)$ can also be shaped into a unipolar square wave by first replacing the abs{.} term of g_{tri} with the signum function and then applying the unipolar transformation entity g_{u} to the result. Unfortunately, this construction does not allow for variable-width duty cycles.

Variable-width pulse signals can be generated by subtracting two out-of-phase ramp signals from each other [19], and then by offsetting the difference with the duty width, it is possible to obtain their unipolar representations. The generating phaseshaper is given by

$$g_{\text{pulse}}\left[x(n), w\right] = x(n) - x(n + wP) + w, \tag{14}$$

where w defines the pulse width ($0 \le w \le 1$) and $P = f_s/f_0$ is the period of $x(n)$. Since Equation 14 is linear and does not thus introduce aliasing, it is well suited for situations where $x(n)$ is a bandlimited or an antialiased signal. However, if aliasing problems are not a concern, the trivial unipolar pulse waveform definition

$$s_{\text{pulse}}(x) = \begin{cases} 1, & \text{when } 0 \le x < w \\ 0, & \text{when } w \le x \le 1 \end{cases} \tag{15}$$

is able to produce similar results more efficiently.

Although a rectangular signal is not a useful phase signal by itself, it may be combined with other phaseshapers for two-segment phase sequences. For instance, the expression for the variable-slope phase signal of Figure 4 is $x(n)\{1 + g_{\text{pulse}}[x(n) - 1, w]\}$. Another application for rectangular signals is the algebraic sawtooth shifter described in [4].

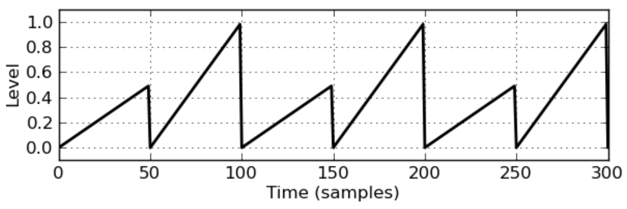

Figure 4. Variable-slope phase signal (w = 0.5).

2.4 Tilted Triangular Fractional Period Phase Signals

Instead of subtracting two sawtooth waveforms from each other, subtracting two out-of-phase parabolic waveforms produces a variable-slope triangle wave [13]. Using phaseshaping techniques, this can be implemented as

$$s_{\text{vtri}}\left[x(n), w\right] = a_{\text{T}}\left\{ g_{\text{b}}^{2}[x(n)] - g_{\text{b}}^{2}[x(n-w)] \right\} + b_{\text{T}}, \tag{16}$$

where w is the duty width, $a_{\text{T}} = 1/[8(w - w^2)]$, and b_{T} = 0.5. Although Equation 16 may be used as a standalone phase generator, it can be further generalized by shaping it with a g_{lin} and mod_1 sequence. This results in the phaseshaper

$$g_{\text{vtri}}\left[x(n), w, a_{1,0}\right] = \text{mod}_1 \circ g_{\text{lin}} \circ s_{\text{vtri}}\left[x(n)\right]. \tag{17}$$

Figure 5 plots a fractional period phase signal generated by g_{vtri}. Comparing this with Figure 3, it is noted that the slopes of the up- and down-ramp cycles are weighted by the duty width w. As expected, with w = 0.5 the slopes become equal in magnitude, at which point g_{vtri} and g_{tri} produce identical results. Therefore, Equation 17 can be seen as a generalization of Equation 12.

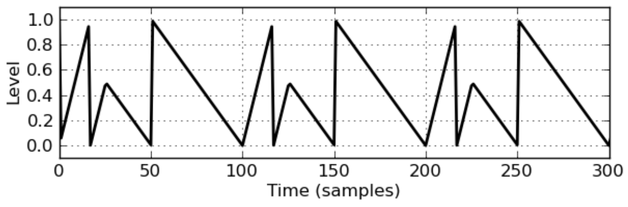

Figure 5. Variable-slope triangular phase signal with a fractional phase period (a_1 = 1.5, a_0 = 0, w = 0.75).

2.5 Phase Signals with Ripples

The definition of the general modulo operation of Equation 9b is

$$x(n) \bmod m = x(n) - m\left\lfloor x(n)/m \right\rfloor, \tag{18}$$

where $m \in$ R is the real-valued wrapping modulus. For efficiency reasons, practical applications usually set m = 1, making Equation 18 equal to the fractional part of $x(n)$. In some applications, however, it is desirable to generate a phaseshaper whose output is decorated with small-amplitude ripples. This can be achieved by utilizing the phaseshaper entity mod_m (with a low fractional m value), as in

$$g_{\text{ripple}}\left[x(n), m\right] = x(n) + \text{mod}_m\left[x(n), m\right]. \tag{19}$$

An example phase signal generated by this phaseshaper is shown in Figure 6.

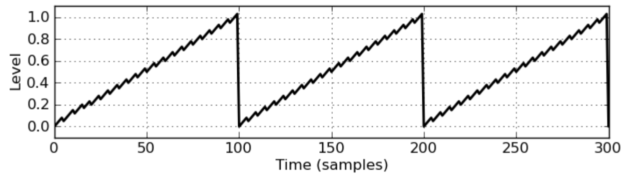

Figure 6. Phase signal with ripples (m = 0.05).

3. OSCILLATOR ALGORITHMS

This section describes the application of the elementary phaseshapers in classic and novel oscillator algorithms.

3.1 Waveslices

The waveforms produced by physical analog oscillators diverge from trivial piecewise linear sawtooth, pulse, and triangle waveshapes. Although these deviations are subtle in the spectral domain, they contribute to the characteristic sound of the synthesizer [15].

These nonlinear waveshapes may be approximated with higher order polynomial or sinusoidal waveshapers. For example, Figure 7a shows an approximation of the Minimoog Voyager sawtooth waveform, which was generated using

$$y(n) = g_{\mathrm{b}} \circ \sin\{2\pi g_{\mathrm{lin}}[\phi(n), a_1 = 0.25]\}. \qquad (20)$$

Parameter a_1 is set to a value smaller than unity so that only a portion of the entire sine wave period is included in the output. The spectrum of the waveform produced by Equation 20 is shown in Figure 7b. As can be seen, the abrupt transition caused by the modulo operation of $\phi(n)$ introduces a questionable amount of aliasing.

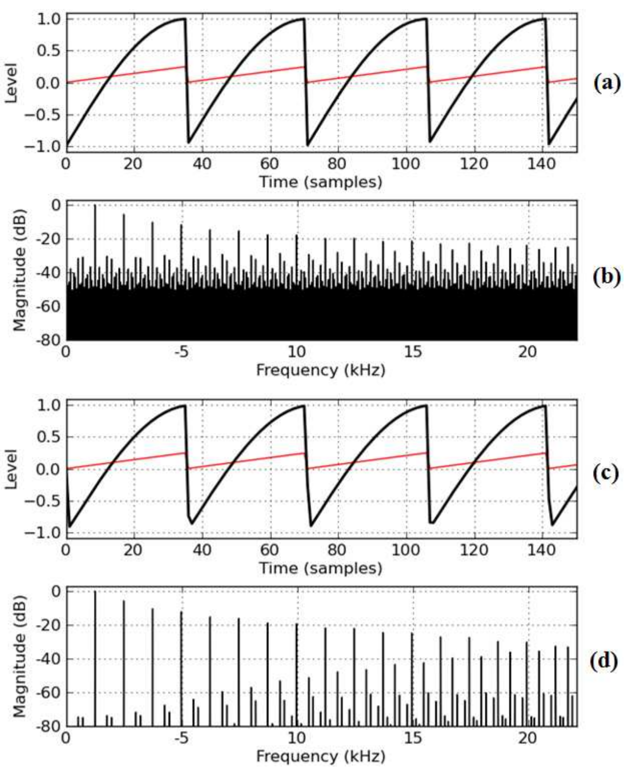

Figure 7. Approximation of the Minimoog Voyager sawtooth waveform. (a,b) Trivial and (c,d) aliasing-suppressed implementation. The thin lines of (a) and (c) plot the phase signal, while the thick lines show the waveshaper output (f_0 = 1245 Hz).

3.2 Antialiasing

The amount of aliasing can be suppressed by smoothing the transition in the time domain. An efficient method to accomplish this is the polynomial bandlimited step function (polyBLEP) [18], which is a simplification of the minBLEP method originally proposed by Brandt [2]. PolyBLEP modifies the values of two samples that are located before and after the modulo transition by evaluating a second-order correction polynomial and adding the result to the values of the two original waveform samples.

Figures 7c and 7d show the aliasing-suppressed waveform and spectrum of Equation 20 after applying the polyBLEP method. The aliasing is suppressed considerably at low and middle frequencies and, although the artifacts are still clearly visible in the spectrum plot, their effect is greatly diminished because of the properties of human hearing. The effect of transition smoothing is also visible in the time domain as the minima of the waveform do not reach the level of –1. Interestingly, the same effect is also observable in the original analog Minimoog Voyager waveform.

This suggests yet another phaseshaper entity that applies the polyBLEP method to its input signal, thereby performing a soft modulo-1 operation. This antialiasing phaseshaper is denoted as

$$\mathrm{mod}_{\mathrm{s}}[x(n), T_{\mathrm{s}}, h] = \mathrm{polyBLEP}[x(n), T_{\mathrm{s}}, h], \qquad (21)$$

where $T_s = f_0 / f_s$ is the phase increment of signal $x(n)$ and h is the maximum height of the discontinuity. The sign of h should be negative for falling transitions. A detailed explanation of the polyBLEP is out of the scope of this paper, but interested readers may consult Reference [18] and the source code published in the companion page of this paper[2].

3.3 Oscillator Synchronization

In classic oscillator hard synchronization (hardsync), the phase of the slave oscillator is reset each time the master oscillator finishes its cycle [2,17]. As shown in Figure 2, modulo-based phaseshaping is capable of producing similar effects by first utilizing the linear transformation phaseshaper g_{lin} and then processing the result with the modulo-1 phaseshaper entity mod_1. The latter operation synthesizes the free-running cycles of the slave oscillator, while the former generates the hardsynced transition. A computationally efficient trivial single-oscillator hardsync implementation is therefore given by the phaseshaping composition

$$y(n) = g_{\mathrm{b}} \circ \mathrm{mod}_1 \circ g_{\mathrm{lin}}[x(n), a_1] = 2[a_1 x(n) \bmod 1] - 1. \qquad (22)$$

The synchronization rate between the master and the slave oscillator is modeled by a_1, which is given in terms of the classic hardsync implementation as

$$a_1 = f_{\mathrm{slave}} / f_{\mathrm{master}}, \qquad (23)$$

where f_{slave} is the slave and f_{master} is the master oscillator frequency, respectively. Figure 8 shows the waveform and spectrum produced by the aliasing-suppressed single-oscillator hardsync algorithm for a_1 = 2.5.

[2] http://www.acoustics.hut.fi/go/smc2010-phaseshaping

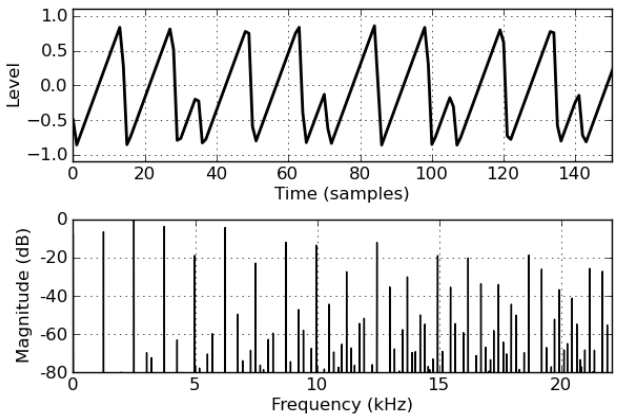

Figure 8. (top) Waveform and (bottom) spectrum of the single-oscillator hardsync algorithm in which the polyBLEP method is used to suppress aliasing (a_1 = 2.5, f_0 = 1245 Hz).

Instead of resetting the phase of the slave, oscillator soft synchronization (softsync) inverts the phase increment of the slave oscillator at the points of synchronization. The trivial single-oscillator softsync implementation utilizes the output of the phaseshaper g_{tri} of Equation 12 either directly or indirectly through a triangular waveshaper function $s_{\text{tri}}\{x\}$:

$$y(n) = g_{\text{b}} \circ g_{\text{tri}}[x(n), a_1] \tag{24a}$$

$$y(n) = g_{\text{b}} \circ s_{\text{tri}}\{g_{\text{tri}}[x(n), a_1]\}. \tag{24b}$$

Figure 9 shows the phase signal g_{tri} (thin line) on top of the resulting waveshaping operation of Equation 24b (thick line). The phase signal does not produce softsync in a strict sense, because the slopes of both ramps are inverted after the synchronization instant. However, this does not have a profound effect on the produced timbre.

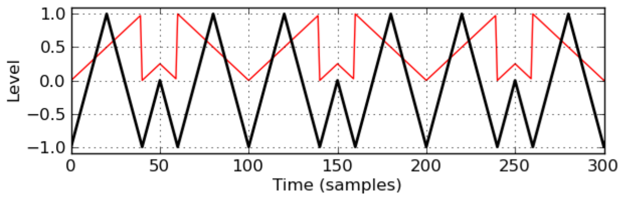

Figure 9. Trivial single-oscillator softsync effect. The thin line plots the phase signal, while the thick line shows the result of the waveshaping acting on that phase, as in all waveform plots in the subsequent examples (a_1 = 1.25 and f_0 = 441 Hz).

3.4 Pulse-width Modulation

Pulse-width modulation (PWM) changes the relative durations of the high and low state segments of a rectangular signal, while the frequency and the amplitude of the signal remain constant [17]. This can be achieved in two ways:

$$y(n) = g_{\text{b}} \circ g_{\text{pulse}}[x(n), w] \tag{25a}$$

$$y(n) = s_{\text{pulse,b}}\{\text{mod}_1 \circ g_{\text{lin}}[x(n), a_1]\}, \tag{25b}$$

where $s_{\text{pulse,b}}$ is the bipolar transformation of Equation 15. Both forms produce classic PWM when $0 < a_1 = w < 1$. When $a_1 > 1$, Equation 25b produces a trivial hardsynced square wave.

3.5 Triangle Modulation

One of the first commercial virtual analog synthesizers, the Roland JP-8000, introduced three original oscillator effects [15]. One of these effects is *triangle modulation* (see Figure 10a), which can be implemented using a scaled bipolar triangular phase signal $x_{\text{T}}(n)$ with a ceiling function:

$$\begin{aligned} x_{\text{T}}(n) &= a_{\text{TM}}\, g_{\text{tri}}\left[x(n), 2, -1\right] \\ y(n) &= 2\left(x_{\text{T}}(n) - \lceil x_{\text{T}}(n) - 0.5 \rceil\right), \end{aligned} \tag{26}$$

where a_{TM} is the modulation amount in the range [0.7, 1], and $\lceil x \rceil$ denotes the ceiling function, which returns the smallest integer not less than x. Figure 10b shows both signals of Equation 26 with a_{TM} = 0.82, corresponding to the Roland JP-8000 triangle modulation offset parameter value 64/127.

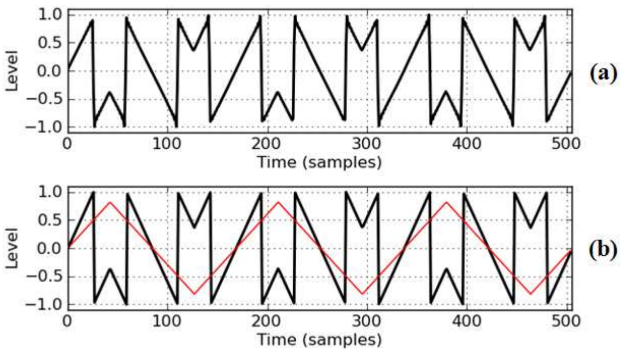

Figure 10. (a) Roland JP-8000 triangle modulation and (b) its simulation. The thin line plots the scaled phase signal $x_{\text{T}}(n)$, while the thick line shows the output signal $y(n)$ (a_{TM} = 0.82, f_0 = 261.63 Hz, and JP-8000 offset parameter = 64/127).

Higher amounts of modulation increase the slope of the ramp and the magnitude of the v-shaped segments. At the maximum modulation $a_{\text{TM}} = 1$ the magnitude of the v-shapes becomes 0.5. Figure 11 shows the effect of a_{TM} to the lower half of the baseband spectrum. As can be seen, the spectrum consists of odd harmonics only, the 3rd partial being the most prominent throughout the entire parameter range. The relative strengths of other harmonics change dynamically with a_{TM}, producing sweeping formant-like oscillator synchronization type effects.

The timbre that is produced by the maximum modulation amount a_{TM} = 1 can also be synthesized using the bitwise logical modulation [6]. This is not surprising, because the bitwise XOR operation is related to the staircase functions mod(.) and ceiling(.) employed here. The expression for the equivalent logical triangle modulation is

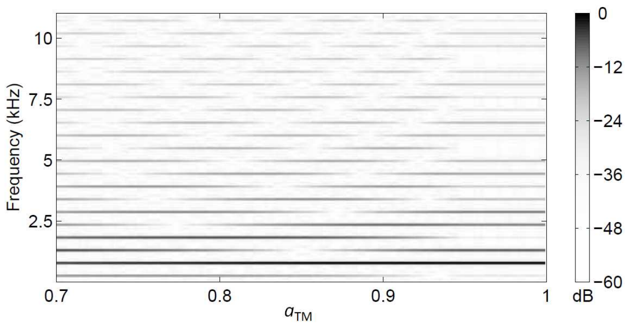

Figure 11. The effect of the modulation amount a_{TM} to the Roland JP-8000 triangle modulation spectrum.

$$y(n) = s_{\text{tri}}\{2g_{\text{tri}}[x(n)]\} \text{ xor } 0.5. \qquad (27)$$

3.6 Supersaw

The most well-known Roland JP-8000 oscillator effect is *supersaw*, which emulates a bank of seven slightly detuned oscillators [15]. Previously, an algorithm for producing the supersaw signal using the bandlimited impulse-train method has been proposed in [12]. However, instead of utilizing seven oscillators, our supersaw simulation employs only one sinusoidal waveshaper that is driven by a slightly modified g_{ripple} phaseshaper:

$$y(n) = g_{\text{b}} \circ \sin\{\text{mod}_{m}[x(n), m_1] + \text{mod}_{m}[x(n), m_2]\}, \quad (28)$$

where m_1 and m_2 are the ripple amounts, and $x(n)$ = $g_{\text{lin}}[\phi(n),a_1]$ = $a_1\phi(n)$. The difference between the g_{ripple} phaseshaper of Equation 19 and that of Equation 28 is the added modulo operation of the first term.

Figure 12 shows three waveforms produced by the supersaw simulation algorithm, using three different ripple amounts m_1. Since $a_1 < 2\pi$, only a portion of the entire sine wave cycle is used as a virtual analog sawtooth oscillator. However, because $a_1 > 1$, the phase signal extends beyond a single phase cycle – thereby introducing an additional discontinuity to the ripple-edged waveform.

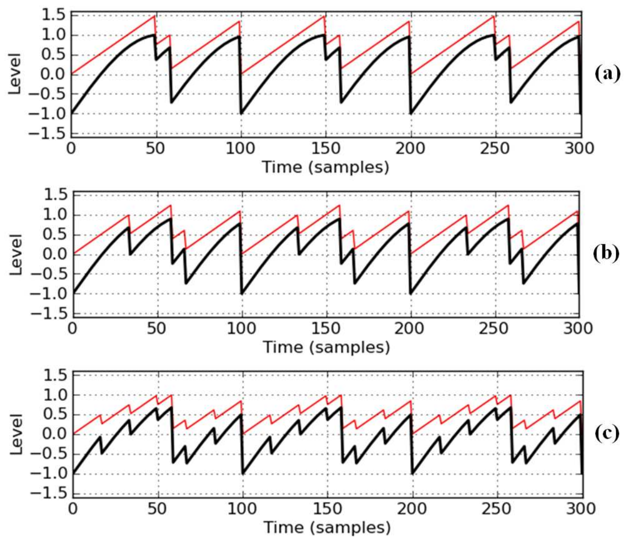

Figure 12. Supersaw simulation. (a) m_1 = 0.75, (b) m_1 = 0.5, (c) m_1 = 0.25 (a_1 = 1.5, m_2 = 0.88, and f_0 = 441 Hz).

Although Equation 28 is capable of synthesizing characteristic spectrally rich supersaw timbres, the sound is still not a convincing simulation of a multi-oscillator setup. This is due to a lack of timbral variations over time, which is a distinctive feature of a slightly detuned oscillator bank. To overcome this, a low frequency oscillator (LFO) may be connected to the m_1 parameter of the algorithm, as shown in Figure 13.

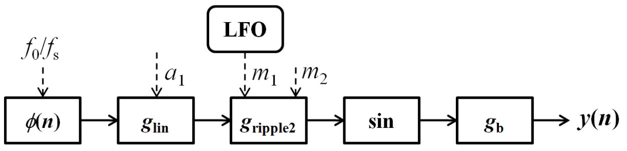

Figure 13. Block diagram of the supersaw simulation algorithm.

Figure 13 shows also that nested phaseshaping is a practical tool that provides a modular approach to sound synthesis and is therefore instantly applicable in systems such as Max, Pure Data, and Reaktor. However, some implementations might opt for minimizing the number of function calls in the code. An example of this is shown in Equation 22.

3.7 Phaseshaping for a Sinusoidal Waveshaper

3.7.1 Sinusoid with a Variable-slope Ramp Phase

Figure 14a shows the output of a sinusoidal waveshaper acting on the variable-slope phase signal of Figure 4. The waveshape consists of concatenated half- and full-cycle sine wave segments alternating at a frequency ratio of 1:2. The spectrum contains all harmonics and decays

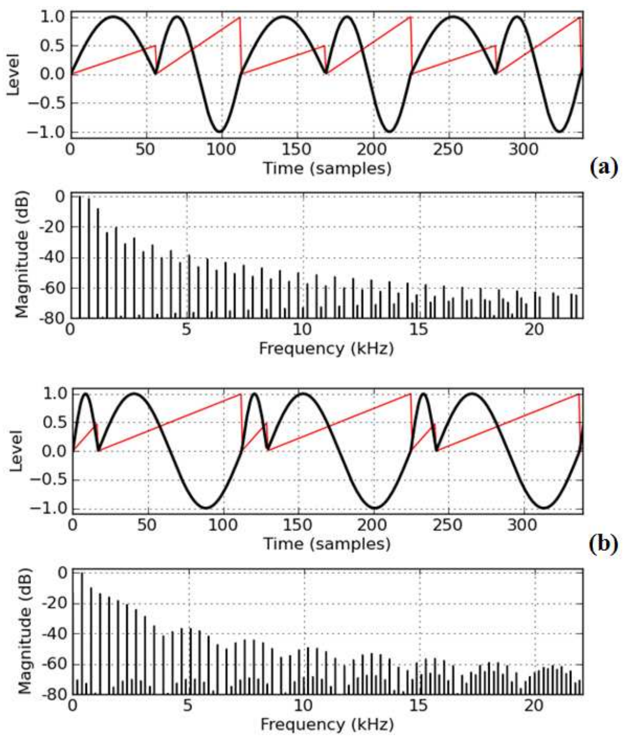

Figure 14. Variable-slope phase signal applied to a sinusoidal waveshaper. (a) Duty width w = 0.50, (b) duty width w = 0.85 (f_0 = 392 Hz).

fairly rapidly because the waveform has discontinuities only in its derivatives.

The phase signal of Figure 14a was generated by multiplying a ramp signal with a square waveform. By replacing the 50% duty-width square with a variable width pulse signal, it becomes possible to alter the relative widths of the half- and full-cycle sine segments, as shown in Figure 14b. As can be seen, the fundamental frequency component is reinforced as the width of the full-cycle segment is increased. The spectrum also shows modest formant regions that sweep across the baseband when the pulse width is modulated with an LFO.

3.7.2 Sinusoid with a Variable-slope Triangular Phase

The variable-slope triangular phase generator g_{vtri} of Equation 17 is closely related to the phase shape of the previous section. However, there are two major differences as can be seen in Figure 15. First, applying a sinusoidal waveshaper to the output of g_{vtri} produces a more prominent formant region, whose position may be controlled using the a_1 parameter. Second, outside this formant region, every fourth harmonic is missing from the spectrum. The aliasing artifacts are also more pronounced, because the symmetrical nature of the phaseshaper is reflected as the sharp peaks of the waveshaped output.

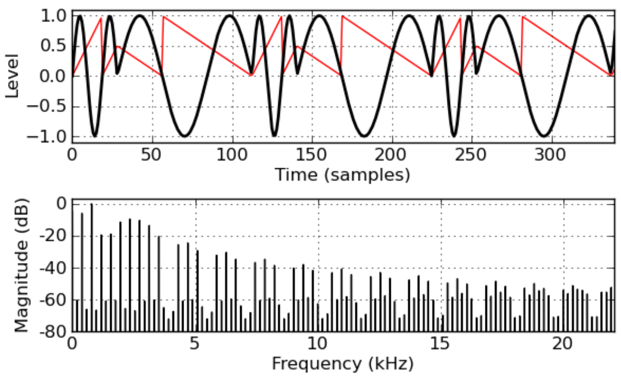

Figure 15. Variable-slope triangular phase signal applied to a sinusoidal waveshaper (a_1 = 1.5, w = 0.75, and f_0 = 392 Hz).

Decreasing the value of parameter a_1 below 1 bends the phase signal from a perfect triangle (a_1 = 0.5) towards a rising ramp shape (a_1 = 0). At a_1 = 0.5, the waveshaper output is a half-cycle sine wave, which gradually bends towards the extreme quarter-cycle segment shown in Figure 7. In between, the spectral tilt becomes less steep, thereby making it possible to control the amount of high end spectral content, as shown in Figure 16.

Lower values of a_1 produce more high end content, and at the same time, the amount of aliasing increases. By comparing Figure 16 to Figure 7, it is noted that polyBLEP provides better aliasing suppression than the sinusoidal waveshaping in effect here.

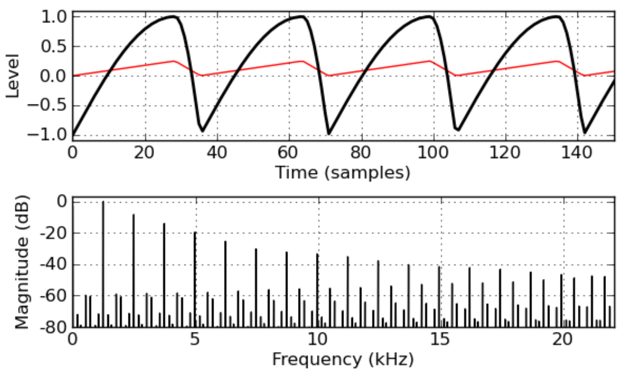

Figure 16. Bent sinusoidal half-cycle (a_1 = 0.25, w = 0.2, and f_0 = 1245 Hz).

4. CONCLUSIONS

This paper investigated elementary phaseshapers, which were based on low-level entities such as modulo operations and linear transformations. All elementary phaseshapers were derived from the unipolar modulo counter signal, which is a common building block of digital sound synthesis systems.

The elementary phaseshapers were then arranged into nested higher-level topologies to form polynomial and geometrical phaseshaper compositions. These included fractional period, variable-width and variable-slope ramp, triangular, rectangular, and ripple-edged phaseshapers.

The phaseshaper compositions were finally utilized in classic and novel oscillator effect algorithms. The novel algorithms comprised single-oscillator hardsync, triangle modulation, efficient supersaw simulation, and sinusoidal waveshape modulation effects.

These synthesis algorithms produce evolving spectra, which can be manipulated with a continuous controller device or a control rate function generator, using a compact set of synthesis parameters. The algorithms are most useful in providing animation to the otherwise static timbres, and as such, respond well to secondary control streams that carry minute articulated expressions of the performer.

Because of the modulo operation, the produced waveforms are generally discontinuous, leading to aliasing artifacts. However, it was found that a previously proposed polynomial bandlimited step function (polyBLEP) is an efficient method to reduce aliasing.

The authors believe that nested phaseshaping is a flexible tool that has many practical uses in the design and implementation of modular sound synthesis applications. Furthermore, because the phase signal has a profound effect on the produced timbre, phaseshaping may also be used in sculpting yet-unheard sonic material.

Online sound examples and software are available at http://www.acoustics.hut.fi/go/smc2010-phaseshaping.

5. ACKNOWLEDGMENTS

This work has been supported by the European Union as part of the 7th Framework Programme (SAME project, ref. 215749) and by the Academy of Finland (project no. 122815).

6. REFERENCES

[1] D. Arfib: "Digital Synthesis of Complex Spectra by Means of Multiplication of Nonlinear Distorted Sine Waves," *Journal of Audio Engineering Society*, Vol. 27, No. 10, pp. 757–768, 1979.

[2] E. Brandt: "Hard Sync Without Aliasing," *Proceedings of the International Computer Music Conference (ICMC 2001)*, Havana, Cuba, Sept. 17-22, 2001.

[3] J. Chowning: "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," *Journal of Audio Engineering Society*, Vol. 21, No. 7, pp. 526–534, 1973.

[4] B. Hutchins: "Analog Circuits for Sound Animation," *Journal of Audio Engineering Society*, Vol. 29, No. 11, pp. 814–820, 1981.

[5] M. Ishibashi: "Electronic Musical Instrument," *U.S. Patent 4,658,691*, 1987.

[6] J. Kleimola: "Audio Synthesis by Bitwise Logical Modulation," *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08)*, pp. 67–70, 2008.

[7] V. Lazzarini and J. Timoney: "New Perspectives on Distortion Synthesis for Virtual Analog Oscillators," *Computer Music Journal*, Vol. 34, No. 1, pp. 28–40, 2010.

[8] V. Lazzarini, J. Timoney, and T. Lysaght: "Nonlinear Distortion Synthesis Using the Split-Sideband Method, with Applications to Adaptive Signal Processing," *Journal of Audio Engineering Society*, Vol. 56, No. 9, pp. 684–695, 2008.

[9] V. Lazzarini, J. Timoney, J. Pekonen, and V. Välimäki: "Adaptive Phase Distortion Synthesis," *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, 2009.

[10] M. Le Brun: "Digital Waveshaping Synthesis," *Journal of Audio Engineering Society*, Vol. 27, No. 4, pp. 250–266, 1979.

[11] J.A. Moorer: "The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulae," *Journal of Audio Engineering Society*, Vol. 24, No. 9, pp. 717–727, 1976.

[12] J. Nam, V. Välimäki, J.S. Abel, and J.O. Smith: "Efficient Antialiasing Oscillator Algorithms Using Low-order Fractional Delay Filters," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 18, No. 4, pp. 773–785, 2010.

[13] M. Puckette: *The Theory and Technique of Electronic Music*, World Scientific Press, 2007.

[14] J.-C. Risset: "An Introductory Catalog of Computer-Synthesized Sounds," Bell Laboratories, 1969. Published as part of The Historical CD of Digital Sound Synthesis, *Computer Music Currents 13*, Wergo WER 20332, 1995.

[15] Roland: *JP-8000 Synthesizer Owner's Manual*, Roland Corporation, 1996.

[16] R.A. Schaefer: "Electronic Musical Tone Production by Nonlinear Waveshaping," *Journal of Audio Engineering Society*, Vol. 18, No. 4, pp. 413–417, 1970.

[17] A. Strange: *Electronic Music: Systems, Techniques and Controls*, William C Brown Pub., 1983.

[18] V. Välimäki and A. Huovilainen: "Antialiasing Oscillators in Subtractive Synthesis," *IEEE Signal Processing Magazine*, Vol. 24, No. 2, pp. 116–125, 2007.

[19] V. Välimäki and A. Huovilainen: "Oscillator and Filter Algorithms for Virtual Analog Synthesis," *Computer Music Journal*, Vol. 30, No. 2, pp. 19–31, 2006.

[20] V. Välimäki, J. Nam, J.O. Smith, and J.S. Abel: "Alias-Suppressed Oscillators Based on Differentiated Polynomial Waveforms," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 18, No. 4, pp. 786–798, 2010.