Alias-Free Digital Synthesis of Classic Analog Waveforms

Tim Stilson (stilti@ccrma.stanford.edu)
Julius Smith (jos@ccrma.stanford.edu)
CCRMA (http://www-ccrma.stanford.edu/)
Music Department, Stanford University

Abstract

Techniques are reviewed and presented for alias-free digital synthesis of classical analog synthesizer waveforms such as pulse train and sawtooth waves. Techniques described include summation of bandlimited primitive waveforms as well as table-lookup techniques. Bandlimited pulse and triangle waveforms are obtained by integrating the difference of two out-of-phase bandlimited impulse trains. Bandlimited impulse trains are generated as a superposition of windowed sinc functions. Methods for more general bandlimited waveform synthesis are also reviewed. Methods are evaluated from the perspectives of sound quality, computational economy, and ease of control.

1 Introduction

Any analog signal with a discontinuity in the waveform (such as pulse train or sawtooth) or in the waveform slope (such as triangle wave) must be *bandlimited* to less than half the sampling rate before sampling to obtain a corresponding discrete-time signal. Simple methods of generating these waveforms digitally contain *aliasing* due to having to round off the discontinuity time to the nearest available sampling instant. The signals primarily addressed here are the impulse train, rectangular pulse, and sawtooth waveforms. Because the latter two signals can be derived from the first by integration, only the algorithm for the impulse train is developed in detail.

2 Why Simple Discrete-Time Pulse Trains are Aliased

The "obvious" way to generate a discrete-time version of an impulse train is to approximate it by a unit-sample-pulse train. The unit sample pulse $\delta(n)$ is defined as

$$\delta(n) \stackrel{\Delta}{=} \begin{cases} 1, & n = 0 \\ 0, & |n| = 1, 2, 3, \ldots \end{cases}$$

The unit-sample pulse is only defined for integer n, so we have a problem: Suppose the desired impulse-train frequency is $f_1 = 1/T_1$, then the period in samples has to be $P = T_1/T_s = F_s/f_1$, where $F_s = 1/T_s$ is the sampling rate, and P is rarely an integer. Because pitch perception

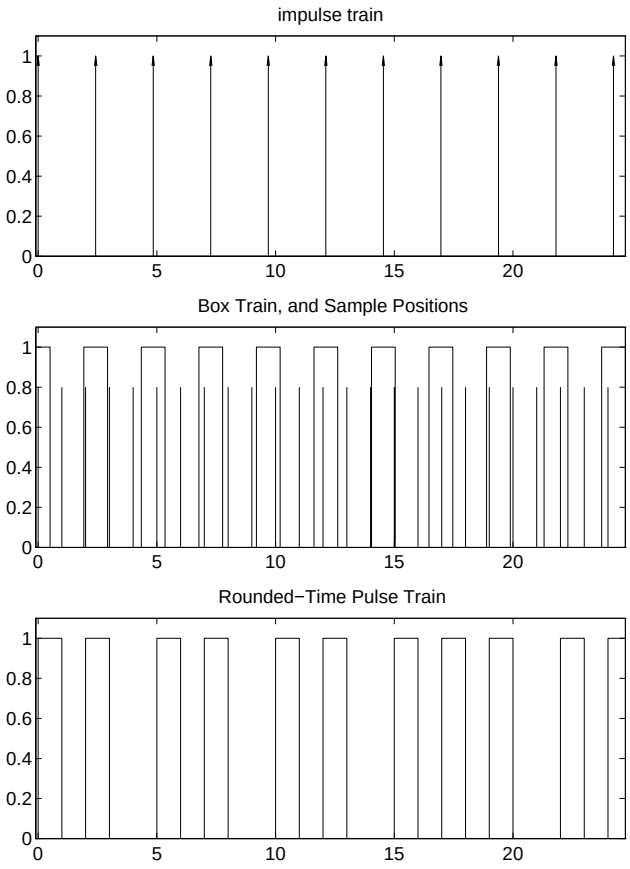

Figure 1: *Rounded-Time Impulse Train as a Sampled Version of an Ideal Rectangular Pulse Train*

is so accurate , it does not work to round M to the nearest integer, except at frequencies so low that the error is on the order of a tenth of a percent (around 100 Hz or below for a 50 kHz sampling rate). Therefore, to make the pitch right, it is necessary to compute the impulse arrival times very accurately and round each arrival time to the nearest sample instant. This process can be modeled as pitch-period jitter which obviously adds noise to the signal. It can also be modeled as a uniform sampling of a periodic sequence of rectangular pulses one sample wide (Figure 1):

$$p(t) \stackrel{\Delta}{=} \begin{cases} 1, & |t| \le T_s/2 \\ 0, & |t| > T_s/2 \end{cases}$$

The Fourier transform of this pulse is a *sinc function*

$$P(\omega) \stackrel{\Delta}{=} \int_{-\infty}^{\infty} p(t) e^{-j\omega t} dt = T_s \,\mathrm{sinc}(fT_s)$$

where $\omega = 2\pi f$, and

$$\mathrm{sinc}(x) \stackrel{\Delta}{=} \frac{\sin(\pi x)}{\pi x}.$$

A periodic sequence of these rectangular pulses is constructed as

$$x(t) \stackrel{\Delta}{=} \sum_{l=-\infty}^{\infty} p(t + lT_1)$$

and its Fourier transform is, by the shift theorem and properties of delta functions,

$$\begin{aligned} X(\omega) &\stackrel{\Delta}{=} \sum_{l=-\infty}^{\infty} e^{j\omega l T_1} P(\omega) \\ &\propto P(\omega) \sum_{l=-\infty}^{\infty} \delta(\omega \Leftrightarrow l\omega_1) \\ &\propto \begin{cases} P(\omega_l), & \omega = \omega_l \\ 0, & \omega \neq \omega_l, \forall l \end{cases} \end{aligned}$$

where $\delta(\omega)$ denotes the delta function, and $\omega_l \stackrel{\Delta}{=} 2\pi l F_1$. Thus, the rectangular-pulse train has an infinite harmonic spectrum weighted by a sinc function having zeros at multiples of the sampling rate F_s (because the original rectangular pulse was taken to be one sampling interval wide). Since the sinc function has a lot of energy above its first zero crossing, sampling $x(t)$ will cause aliasing of an infinite number of harmonics whose amplitudes fall off at 6 dB per octave (since $\mathrm{sinc}(fT_s)$ falls off as $1/f$). (Harmonics under the outer half of the main lobe of the sinc function also alias.) Sampling, we get

$$\begin{aligned} Y(e^{j\omega T_s}) &\propto \sum_{k=-\infty}^{\infty} X(\omega + k2\pi F_s) \\ &= \sum_{k=-\infty}^{\infty} P(\omega + k2\pi F_s) \\ &\quad \times \sum_{l=-\infty}^{\infty} \delta(\omega + k2\pi F_s \Leftrightarrow l\omega_1) \end{aligned}$$

The desired pulse-train spectrum is only the $k = 0$ term above. Each nonzero k term contributes a string of aliased of harmonics across the entire frequency band. The amount of aliasing is highly significant and audible, as can be predicted from looking at Fig. 6.

The only frequency which does not suffer aliasing is DC since the sinc spectral envelope goes through zero at all multiples of the sampling rate. For this reason, the aliasing is reduced at very low frequencies relative to the sampling rate. This provides another explanation why the aliasing is not as objectionable for low fundamental frequencies (where the nearest-integer roundoff is an insignificant fraction of the period): The aliased frequencies that would tend to be most noticeable (those near or below the fundamental frequency) are most attenuated.

The above analysis extends immediately to rectangle waves.

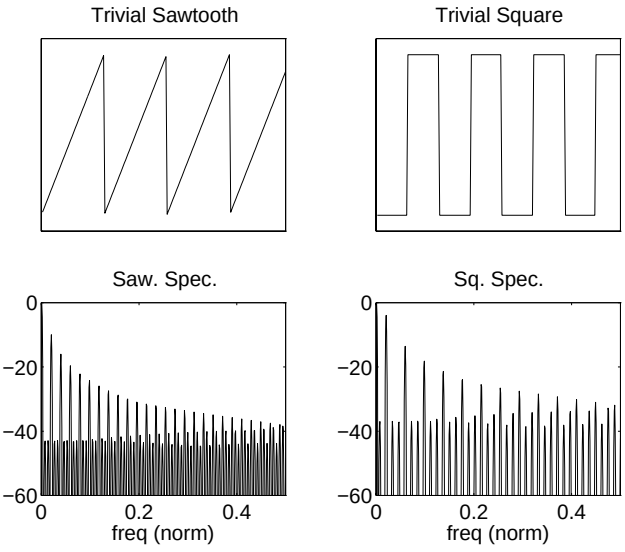

Figure 2: *Spectra of Trivial Sawtooth and Square-Wave Signals, Low Frequency*

A similar argument applies to explain why sawtooth signals alias: If the unit-amplitude sawtooth is generated by

$$s(n) = (f_1 n T_s) \bmod 1$$

where f_1 is the desired frequency, then it can be shown to be simply a sampling of a continuous-time ("analog")

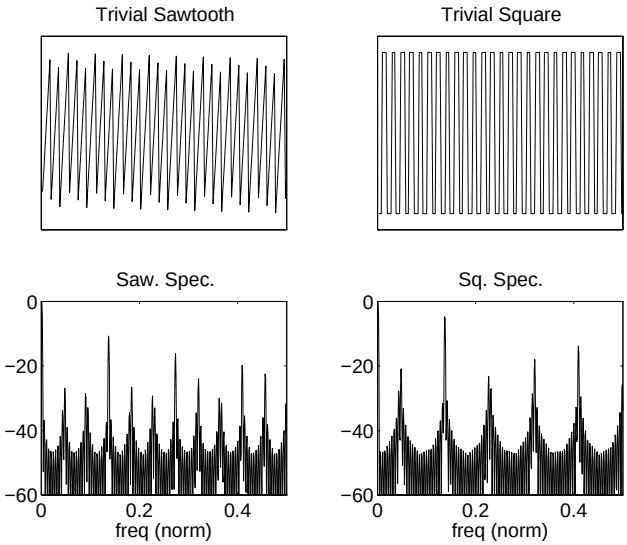

Figure 3: *Spectra of Trivial Sawtooth and Square-Wave Signals, High Frequency*

sawtooth of equivalent slope. Since the analog sawtooth is not bandlimited, the sampled version will be aliased.

This is slightly different from a rounded-time retriggered sawtooth which is better described as the convolution of a single cycle of a sawtooth wave with a rounded-time impulse train, which was just shown to be aliased as well. Since a single cycle of a sawtooth has a Fourier transform which falls off at 6 dB per octave (all discontinuous waveforms have spectral fall-offs no faster than 6 dB per octave [Papoulis 1991]), it follows that the spectrum of the rounded-time retriggered sawtooth also falls off at 6 dB per octave. Therefore, sampling it causes aliasing in an amount similar to that in the sampled rectangular pulse train.

By convolving the rounded-time impulse train (sampled rectangular pulse train, with pulses of width T_s) with an arbitrary fixed filter, *formant synthesis* such as VOSIM [Kaegi and and S. Tempelaars 1978], Chant [Rodet *et al.* 1989] (see also [Roads 1996]) is described. By multiplying the rounded-time impulse train by a periodic amplitude envelope and convolving it with a Blackman FFT window, *window-function synthesis* is described [Goeddel and Bass 1984] (summarized in [Roads 1996]). In all these cases, a pitch-period is associated with the impulse response of the formant filter (or lowpass filter in the case of window-function synthesis), perhaps overlapping with those of other periods, and the impulse-response start times are quantized to the nearest sample, thus causing pitch-period jitter. Eliminating this jitter requires resampling the filter impulse response each period which would be very expensive, so it appears not to be done in practice.

3 Bandlimited Synthesis

In this section, various preexisting methods for synthesis of bandlimited waveforms will be reviewed.

3.1 Additive Synthesis

Additive synthesis is trivially bandlimited simply by not generating harmonics higher than $F_s/2$; typically this also presents a computational savings. Additive synthesis systems that use the inverse FFT to compute the oscillators are bandlimited by definition because the inverse FFT only generates frequencies up to $F_s/2$. In inverse-FFT synthesis, *time*-aliasing becomes the error to minimize.

3.2 Wavetable Synthesis

One of the very earliest synthesis techniques used in computer music was periodic wavetable synthesis [Mathews 1969] (not to be confused with sample playback synthesis which is also called wavetable synthesis these days). In this technique, a wavetable contains only one period of the desired tone, sampled at a high rate, such as $N = 512$ samples per period. Playing out the table repeatedly generates a periodic waveform with fundamental frequency F_s/N, where F_s is the sampling rate. Skipping every other sample on playback yields a fundamental frequency of $2F_s/N$ and so on. Intermediate frequencies f_0 are obtained using a non-integer skip-factor, or "phase increment," given by $\text{Inc} = N f_0/F_s$; in such cases, the wavetable address has a fractional part which is often either discarded or used to round to the nearest integer (reasonable for very large N and/or heavily oversampled waveforms), or it is used to determine an interpolated table value. By far the most common interpolation technique is *linear* interpolation. However, higher order interpolation methods, especially Lagrange [Schafer and Rabiner 1973] and bandlimited interpolation [Smith and Gossett 1984], have been used commercially.

Interpolated wavetable synthesis is not guaranteed to be bandlimited when the phase increment is larger than one sample. In these cases, one is performing the equivalent of a decimation of the waveform, with the decimation factor being equal to the phase increment in samples. Therefore, for nonsinusoidal wavetables, an upper bound is imposed on the phase increment by the highest harmonic of the signal in the wavetable ($\max(\text{Inc}) = N f_0/(F_s n_h)$, where n_h is the harmonic number of the

highest harmonic.). If the restriction on exact bandlimiting is relaxed, then the phase increment can have a higher bound, based on the highest harmonic whose amplitude is large enough to be objectionable when aliased.

Interpolated wavetable synthesis is equivalent in principle to *additive synthesis* employing a digital sinusoidal oscillator on every harmonic. In practice, wavetables are generally computed as a weighted sum of harmonics up to some maximum harmonic number. Thus, we may either add the outputs of N oscillators together, or we may add their effective wavetables together with the same weightings to obtain a single wavetable oscillator. If the highest harmonic frequency is well below half the sampling rate, and/or if the harmonic amplitudes decrease rapidly with harmonic number, as normally happens in natural waveforms, inexpensive interpolation techniques such as linear interpolation give high quality results.

3.3 Bandlimited Interpolation

For a general discrete-time signal $x(nT_s)$ which has been uniformly sampled at twice its highest frequency, exact bandlimited interpolation, which we call *sinc interpolation*, is carried out as

$$\begin{aligned} x(t) &= \sum_{n=-\infty}^{\infty} x(nT_s) h_s(t - nT_s) \qquad (1) \\ &= \frac{\sin(\pi F_s t)}{\pi F_s} \sum_{n=-\infty}^{\infty} x(nT_s) \frac{(-1)^n}{t - nT_s} \end{aligned}$$

where

$$h_s(t) \triangleq \mathrm{sinc}(F_s t) \triangleq \frac{\sin(\pi F_s t)}{\pi F_s t}.$$

To resample $x(t)$ at a new sampling rate $F_s' = 1/T_s'$, we need only evaluate Eq. (2) at integer multiples of T_s'.

The summation in Eq. (2) cannot be implemented in practice because the "ideal lowpass filter" impulse response $h_s(t)$ actually extends from minus infinity to infinity. It is necessary in practice to *window* the ideal impulse response so as to make it finite. This is the basis of the *window method* for digital filter design [Rabiner and Gold 1975]. (See also the `fir1()` function in Matlab.) While many other filter design techniques exist (e.g., [Bellanger 1996]), the window method is simple and robust, especially for very long impulse responses. The filter impulse response is very long because it is heavily oversampled. Interpolation techniques of this nature which approximate sinc interpolation in the frequency domain are generally referred to as *bandlimited interpolation* techniques [Schafer and Rabiner 1973, Crochiere and Rabiner 1983, Smith and Gossett 1984].

3.4 Exact Wavetable Interpolation

For best results using interpolated periodic wavetable synthesis, sinc interpolation is ideal as always. Since the signal is periodic, Eq. (2) can be collapsed to a finite sum over one period, thus making it implementable in ideal form [Schanze 1995]:

$$\begin{aligned} x(t) &= \frac{\sin(\pi t)}{2N} \sum_{n=-L}^{M-1} x(n)(-1)^n \left[\cot\left(\pi \frac{t - n}{2N}\right) \right. \\ &\qquad \left. + (-1)^{N+1} \tan\left(\pi \frac{t - n}{2N}\right) \right] \end{aligned}$$

where the sampling interval is normalized to be $T = 1$, and the period is $N = L + M$ samples. Intuitively, this can be understood as a *time aliasing* of Eq. (2) over one period of $x(t)$, i.e., it can be shown that

$$x(t) = \sum_{\text{period}} x(nT_s) \mathrm{Sinc}_N[F_s(t - nT_s)] C_N(n) \qquad (2)$$

where

$$C_N(n) \triangleq \begin{cases} 1, & N \text{ odd} \\ \cos(\pi F_s(t - n)/N), & N \text{ even} \end{cases}$$

and

$$\mathrm{Sinc}_N(t) \triangleq \frac{\sin(\pi t)}{N \sin(\pi t/N)}$$

can be recognized as $\mathrm{sinc}(t)$ time-aliased on a block of N samples. The form $N\mathrm{Sinc}_N(N\omega)$ arises often as the Discrete-Time Fourier Transform (DTFT) of the N-sample pulse (N successive unit samples preceded and followed by zeros). The present occurrence is simply the dual: $\mathrm{Sinc}_N(F_s t)$ is the N-periodic continuous-time signal given by the inverse Fourier series expansion of a uniform bank of N harmonics at amplitude $1/N$, centered about and including DC, and extending from $-F_s/2$ to $F_s/2$.

Unifying with the previous section, each of the N samples of a periodic signal contributes a copy of $\mathrm{Sinc}_N(F_s t)$ to the interpolation output (for odd N), while each sample of a general discrete-time signal contributes a copy of $\mathrm{sinc}(F_s t)$ to the interpolation output.

A difficulty with "Sinc_N Synthesis" is that the wavetable contents must be changed with frequency. Since frequency is constantly varying due to vibrato, onset skew, etc., the wavetable must be recomputed on the fly or else stored for all pitches. One way to accomplish this is to have an efficient means for real-time generation of Sinc_N, and synthesize an arbitrary N-periodic signal as a sum of N Sinc_N oscillators. However, the end result is no different from summing N sinusoidal oscillators, so ordinary additive synthesis would be preferable.

3.5 DSF Synthesis

In [Moorer 1975] (partially summarized in [Dodge and Jerse 1985, pp. 149–154] and [Roads 1996, pp. 260–261]), Discrete-Summation Formulae (DSF) are proposed for synthesizing bandlimited periodic signals based on the identity

$$\begin{aligned}\sum_{k=0}^{N-1} a^k \sin(\theta + k\beta) &= \frac{1}{1 \Leftrightarrow 2a\cos(\beta) + a^2} \times \\ [\sin(\theta) \Leftrightarrow a\sin(\theta \Leftrightarrow \beta) &\Leftrightarrow a^N \sin(\theta + N\beta) \\ &+ a^{N+1}\sin[\theta + (n \Leftrightarrow 1)\beta]\end{aligned}$$

The above closed-form expression is derived in a straightforward manner using the identity $2j\sin(x) = e^{jx} \Leftrightarrow e^{-jx}$ on the left-hand side, and applying the closed-form expression for a geometric series:

$$\sum_{k=0}^{N-1} z^k = \frac{1 \Leftrightarrow z^N}{1 \Leftrightarrow z}$$

By setting θ and β to various values, a wide class of bandlimited waveforms can be generated. Bandlimiting is achieved by controlling N such that the highest frequency generated is less than $F_s/2$ (for $\theta = 0$ and $\beta = f_1 t, N \Leftrightarrow 1 = \lfloor F_s/(2f_1) \rfloor$). This also easily generalizes to lower bandlimit frequencies. Note that Csound's `buzz` and `gbuzz` unit generators implement this method.

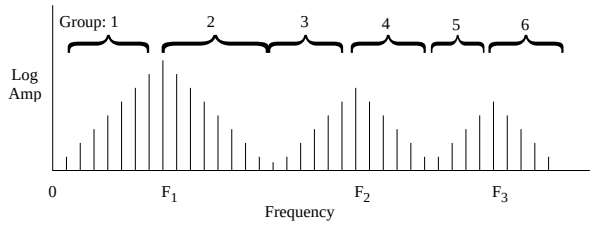

Figure 4: *Use of six DSF tables to synthesis voice.*

As an example, voice synthesis using three vocal formants can be carried out using DSF in a manner similar to FM voice synthesis [Chowning 1989]. In this technique, a row of exponentially rising or falling harmonics is used to approximate the harmonics falling under *half* a formant, as shown in Fig. 4. For K formants, we need $2K$ DSF units. In the three-formant case, we have six partial groups. Moorer also describes a variant of DSF that generates two-sided harmonic series. In this scheme, only three groups would be necessary (as long as the fall-off rate were the same for both sides).

3.6 Bandlimited Impulse Train (BLIT) Synthesis

Some of the previously mentioned techniques can be used to generate BLITs, but can also generate other waveforms. Now we will discuss techniques designed especially to generate BLITs, without consideration for extension to other waveforms (although it may be possible in some cases). By restricting to a single specific waveform, these methods can often be much more efficient than the other methods. These also benefit strongly from the fact that the signal to be generated is an impulse train, which is mathematically much simpler than most other waveforms.[1]

3.7 Sum of Sincs and Sinc$_M$

The standard operation before sampling is to apply an anti-aliasing filter. The ideal anti-aliasing filter has a continuous-time impulse response that is a sinc function with a zero-crossing interval of one sample:

$$h_s(t) \stackrel{\Delta}{=} \mathrm{sinc}(F_s t) \stackrel{\Delta}{=} \frac{\sin(\pi F_s t)}{\pi F_s t}.$$

The ideal unit-amplitude impulse train with period T_1 seconds is given by

$$x(t) = \sum_{l=-\infty}^{\infty} \delta(t + lT_1)$$

Applying the anti-aliasing filter h_s to this signal gives

$$\begin{aligned}x_f(t) \stackrel{\Delta}{=} (x * h_s)(t) &= \sum_{l=-\infty}^{\infty} h_s(t + lT_1) \\ &= \sum_{l=-\infty}^{\infty} \mathrm{sinc}(t/T_s + lP)\end{aligned}$$

where $P = T_1/T_s$ is the period in samples (not an integer). Since x_f is bandlimited to the frequency interval $(\Leftrightarrow F_s/2, F_s/2)$, it can now be sampled without aliasing to obtain

$$y(n) \stackrel{\Delta}{=} x_f(nT_s) = \sum_{l=-\infty}^{\infty} \mathrm{sinc}(n + lP)$$

As before, the above expression for $y(n)$ can be interpreted as a *time aliasing* of the sinc function about an interval of P samples. It can be shown that the time-aliased sinc becomes

$$y(n) = (M/P)\,\mathrm{Sinc}_M[(M/P)n] \qquad (3)$$

[1] The basic operation to be simulated is `Sample[ImpulseTrain(t) * sinc(t)]`, where "$*$" denotes convolution. Since ImpulseTrain is just impulses, the convolution is almost trivial. This operation would be much more difficult to calculate for more complex waveforms

where

$$\mathrm{Sinc}_M(x) \triangleq \frac{\sin(\pi x)}{M \sin(\pi x/M)}$$

This function provides a closed-form expression for the sampled bandlimited impulse train (BLIT), and it can be used directly for synthesis in a manner similar to DSF. While P is the period in samples, M is the number of harmonics. It is always odd because an impulse train has one "harmonic" at DC, and an even number of non-zero harmonics, provided no harmonic is allowed at exactly half the sampling rate (which we enforce). Note that M/P is always close to 1. When P is an odd integer, $P = M$, and $y(n)$ is simply $\mathrm{Sinc}_M(n)$. As P departs from M, Eq. (3) implements a time scaling along with a compensating amplitude scaling. We can relate the number of harmonics M to the period P of the impulse train as

$$M = 2\lfloor P/2 \rfloor + 1$$

i.e., M is the largest odd integer not exceeding the period P in samples.

The above expression for $y(n)$ can also be derived via Discrete-Summation Formulae and Eq. (2). DSF is often used with a constant number of harmonics, rather than harmonics that go all way out to $F_s/2$. This produces a slightly different result. (I think it is the same as the above equation for y, except that M is a constant.)

3.8 Sum of Windowed Sincs (BLIT-SWS)

A more efficient method for synthesizing digital impulse trains may be based on the windowed-sinc method for general bandlimited interpolation [Smith and Gossett 1984]. The technique is equivalent conceptually to bandlimited periodic wavetable synthesis of an impulse train, as mentioned earlier: Bandlimited interpolation is used to convert the sampling rate of a discrete-time unit sample pulse train from a pitch which divides the sampling rate (so that the period is an integer) to the desired pitch. The rate conversion causes each unit sample pulse $\delta(n)$ to be replaced by a windowed sinc function $w(t)h_s(t)$ sampled at some phase which generally varies each period.

Harmonic (aliasing) fall-off rate vs. number of zero crossings

Because the windowing imposes a finite fall-off rate in the harmonics, some aliasing is inevitable. We can, however, control this by our choice of window. It is also helpful to helpful to have an oversampling factor so that there is a good sized guard band between the upper limit of human hearing and half the sampling rate. This reduces the window length required.

Number of Harmonics vs. Sinc Overlap

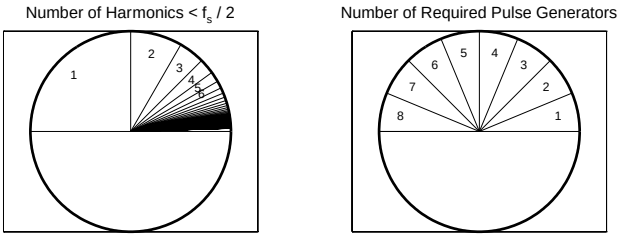

Figure 5: *Comparing number of harmonics to number of overlapped pulse instances for a pulse 8 samples long.*

A further optimization comes from comparing the number of sincs that must be overlapped in the BLIT-SWS method to the number of harmonics of the BLIT that land below $F_s/2$. At very high frequencies, the number of bandlimited harmonics becomes quite small. Indeed, in the top octave, only the fundamental is in band. Thus for a large percentage of the frequency range, it is quite likely that it may be more efficient to generate a BLIT (or any other harmonic waveform) by simple summation of sines. At lower frequencies, we can again revert to the BLIT-SWS method because it is obviously more efficient at low frequencies, where the number of harmonics is very large.

Precisely where the tradeoff occurs depends on the system in which the algorithm is to be implemented. For an extreme example, in an FFT^{-1} system the tradeoff frequency moves all the way down to $F_s/(2N)$, because the system implements summed sines so efficiently. On the other hand, in a system where sine generation is significantly more expensive (say in a system where memory accesses are expensive enough to make even table lookup mildly expensive), the tradeoff frequency can easily be on the order of $F_s/8$. Of course, the amount of accuracy desired in the BLIT-SWS algorithm also affects the tradeoff, because it will affect the choice of the window length, thus affecting the number overlapping sincs at a given frequency.

Exact BL vs. some fall-off rate

In practice, the BLIT-SWS method has demonstrated another advantage over other, more exact bandlimited BLIT algorithms. In cases where the frequency is sweeping, such as vibrato or portamento, high harmonics of the signal will disappear or appear (depending on the direction of the frequency sweep) during the sweep.

In an exactly bandlimited system, the highest harmonic transitions between full amplitude and zero amplitude in the period of one sample. This causes an audible transient, especially at low sampling rates, where $F_s/2$ is well within the audible region. These transients are usually unwanted and distracting.

Like additive synthesis and bandlimited wavetable synthesis, and unlike DSF, in BLIT-SWS synthesis the highest harmonic need not audibly "pop" in or out as it comes down from or gets up to half the sampling rate, since the window function can be chosen to exhibit any amount of attenuation at $F_s/2$.

This effect has not been a historical problem in recorded digital music because of the use of non-ideal anti-aliasing filters. The finite fall-off rate of these filters allows harmonics to die out more slowly in upward sweeps (and appear more slowly in downward sweep), which avoids the above-mentioned transients. The BLIT-SWS method ends up implementing slower fall-off rates as an artifact of the windowing, and therefore gets this effect for free.

Some of the above-mentioned BLIT generation methods cannot easily implement a slower fallof. DSF, for example, has control only over the existence (or lack thereof) of harmonics via N, so it can only implement abrupt transitions in the number of active harmonics. DSF can implement a harmonic fall-off via the parameter a, but this fall-off must start at the first harmonic and increase through all the harmonics, rather than beginning only at the last few harmonics. (Sinc_M synthesis has a similar lack of control over fall-off rate; in fact, there is no fall-off control at all without additional filtering). A few hacks can be used to overcome this. For example, a second DSF group of harmonics can be placed at the top few harmonics with a small a to implement the fall-off rate. Another hack places a post- filter after the DSF to implement the high-frequency fall-off. This hack may have problems handling the harmonics' on-off transients, however.

Example Spectra

Figure 6 shows the spectrum of a rounded-time impulse train. Here unit samples are put out at the sample time nearest the ideal time, and there is massive aliasing.

Figure 7 shows the spectrum of a discrete-time impulse train using linear interpolation to interpolate the unit sample pulse locations. The aliasing is reduced, but it is still very strong.

Figure 8 shows the spectrum of a bandlimited impulse train generated using the BLIT-SWS method with 8 sinc zero-crossings under a Blackman window. While there is still considerable aliasing at high frequencies, at low

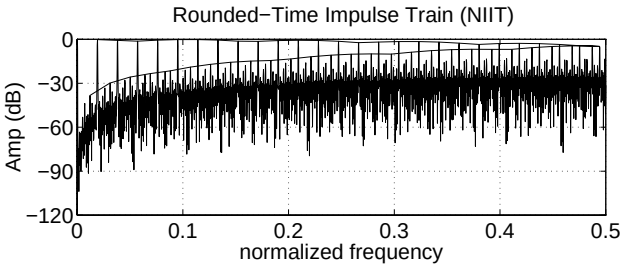

Figure 6: *Spectrum of rounded-time impulse train with a line drawn connecting the peaks of both the desired harmonics and the first string of aliased harmonics ("NIIT" stands for "Nearest-Integer Impulse Train").*

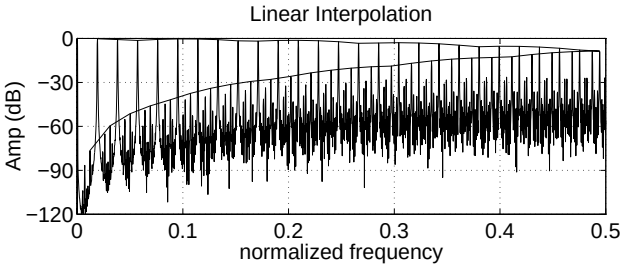

Figure 7: *Spectrum of linearly interpolated impulse train with a line drawn connecting the peaks of desired harmonics and aliased harmonics.*

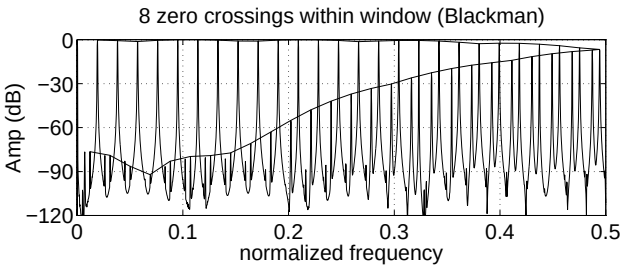

Figure 8: *Spectrum of windowed-sinc interpolated impulse train, the window spanning* 8 *zero-crossings of the sinc function.*

frequencies it is down 90 dB or so.

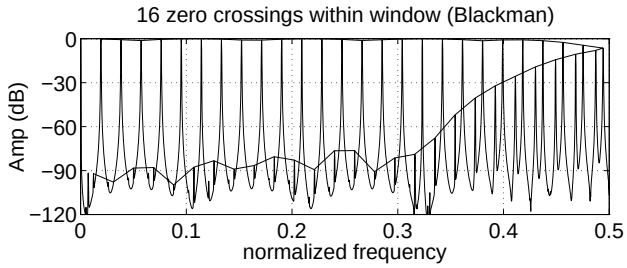

Figure 9: *Spectrum of windowed-sinc interpolated impulse train, the window spanning* 16 *zero-crossings of the sinc function.*

Figure 9 shows the same thing as Fig. 8 with the number of zero crossings raised from 8 to 16. This roughly halves the transition bandwidth of the window transform, and as a result, the aliasing is down 90 dB over approximately 60% of the spectrum.

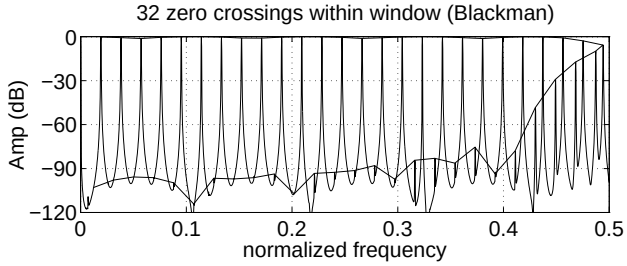

Figure 10: *Spectrum of windowed-sinc interpolated impulse train, the window spanning* 32 *zero-crossings of the sinc function.*

Figure 10 shows the same thing again with the number of zero crossings doubled again from 16 to 32. Again the transition width is halved, and now only the upper 20% of the spectrum is heavily aliased.

Finally Fig. 11 shows the previous case (32 zero crossings) with the cut-off frequency of the sinc function lowered below half the sampling rate. This means the transition band of the window transform is folded in half as it falls into half the sampling rate and reflects. The result is another halving of the aliased region to about 10% of the highest frequencies. If the limit of human hearing is 20 kHz, this means we need a 2 kHz guard band, so the sampling rate should be at least 44 kHz.

4 Square- and Sawtooth-Wave Generation

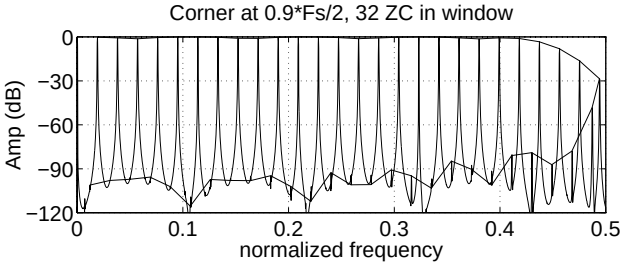

Figure 11: *Spectrum of windowed-sinc interpolated impulse train, the window spanning* 32 *zero-crossings of the sinc function, and the sinc function dilated so as to lower its cut-off frequency below half the sampling rate.*

The next major class of analog waveforms are Square waves and Sawtooth waves.[2] We will show how to easily derive these from a BLIT via integrations which are linear transforms (that can be implemented with trivial filters), so that they preserve the bandlimited nature of the BLIT.

4.1 Successive Integration of BLIT

Sawtooth

A sawtooth function can be generated as follows:

$$Saw_{\mathrm{CTS}}(t) = \int_0^t \mathrm{CIT}(\tau) \Leftrightarrow C_1 \, d\tau$$

where $\mathrm{CIT}(\tau)$ is a continuous-time impulse train, and $C_1 = \int_0^T \mathrm{CIT}(\tau) \, d\tau$, the DC component of the impulse train. This converts directly to discrete-time (via the impulse-invariant transform):

$$\begin{aligned} Saw(n) &= \sum_{k=0}^{n} BLIT(k) \Leftrightarrow C_2 \\ \Leftrightarrow Saw(z) &= \frac{z}{z \Leftrightarrow 1}(BLIT(z) \Leftrightarrow Z(C_2)) \end{aligned}$$

Which is trivally impementable with a single sum and one-pole digital filter. The offet C_2 is the average value of *BLIT*, which should be subtracted off to keep the integration from ramping off to infinity (or saturating). C_2 is a function of frequency. Depending on the initial conditions of the integrator, there will be a DC offset on the output of the integrator.

[2] To avoid confusion, we will use the following naming convention: A "square wave" is a rectangle wave with 50% duty cycle (i.e., "rectangle wave" means a wave that can have other duty cycles). A "triangle wave" can have asymmetric up/down slopes (including the 50% duty-cycle version), and a "sawtooth" wave must have infinite-slope transitions (either up or down). Thus a sawtooth wave is a triangle wave with either 0% or 100% duty cycle. All waves can have unipolar, bipolar, or arbitrary-offset versions.

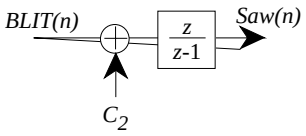

Figure 12: *Direct Sawtooth Generation*

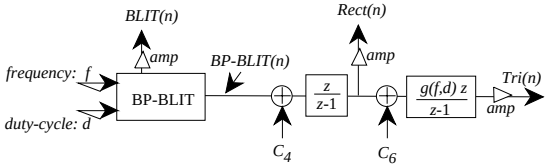

Figure 13: *Rectangle and Triangle Generation*

Rectangle

A rectangle wave can be computed as:

$$Rect_{\text{CTS}}(t) = \int_0^t \text{CIT}(\tau) \Leftrightarrow \text{CIT}(\tau \Leftrightarrow \text{t}_0) \Leftrightarrow C_3 \, d\tau$$

Which discretizes to:

$$\begin{aligned} Rect(n) &= \sum_{k=0}^{n} \text{BLIT(k)} \Leftrightarrow \text{BLIT(k} \Leftrightarrow \text{k}_0) \Leftrightarrow C_4 \\ &= \sum_{k=0}^{n} \text{BP-BLIT}_{k_0}(k) \Leftrightarrow C_4 \\ \Leftrightarrow Saw(z) &= \frac{z}{z \Leftrightarrow 1}(\text{BP-BLIT}_{k_0}(z) \Leftrightarrow Z(C_4)) \end{aligned}$$

Where BP-BLIT is a "BiPolar" BLIT, whose pulses alternate sign. See Section 5 for discussion on hacks for efficiently generating BP-BLIT. It turns out that a bipolar impulse train has a DC component of zero, which means that $C_3 = 0$ (and subsequently $C_4 = 0$). The rectangle width is controlled with k_0, which can be varied to give PWM (pulse-width modulation). The range of k_0 in these equations is [0,Period]. Depending on the implementation of the BLIT, the PWM control may also be in the range [0,1]. Depending on the initial conditions of the integrator, there will be a DC offset on the output.

Triangle

A triangle wave can be generated as:

$$Tri_{\text{CTS}}(t) = \int_0^t Rect_{\text{CTS}}(\tau) \Leftrightarrow C_5 \, d\tau$$

Where C_5 is the DC component of the rectangle wave: $C_5 = \int_0^T Rect_{\text{CTS}}(\tau) \, d\tau$. This discretizes to:

$$\begin{aligned} Tri(n) &= \sum_{k=0}^{n} Rect(k) \Leftrightarrow C_6 \\ \Leftrightarrow Tri(z) &= \frac{z}{z \Leftrightarrow 1}(Rect(z) \Leftrightarrow Z(C_6)) \end{aligned}$$

The offsets C_5 and C_6 are functions of the rectangle wave duty cycle and of a DC offset that arises from the initial conditions of the integration that produces the rectangle wave: $C_6 = k_0/Period + C_7$. C_7 is a function of $BLIT(0)$, the initial condition of the integration. To get appropriate amplitude on the triangle wave (so that its extrema are the same size as those of the BLIT and Rectangle wave), a frequency- and duty-cycle-dependant scaling must be performed on the Triangle integration:

$$\begin{aligned} Tri(n) &= \sum_{k=0}^{n} g(f,d)(Rect(k) \Leftrightarrow C_6) \\ g(f,d) &= \frac{2f}{d(1 \Leftrightarrow d)} \end{aligned}$$

Where f is the frequency in units of (cycles/sample), and d is the duty cycle ($d \in [0,1]$).

Why Rectangle and Triangle?

These particular waveforms were chosen especially because they can be easily generated from BLITs, which as explaied earlier are special in their ease of calculation. We were just lucky that they were also popular analog waveforms (or is it more than coincidence…?)

4.2 Can DSF do it?

One may wonder if DSF can be used to directly generate rectangle or triangle waves via appropriate settings of the a parameter. The answer is no, because (as can be shown from the integrations) the square-wave's harmonics' amplitudes fall of as $1/f$ (or, equivalently, as $1/n$, where n is the harmonic number), and the triangle's as $1/f^2$. The DSF harmonics fall off as a^k. This keeps the DSF from generating exact rectangle or triangle waves, but depending on the circumstance (i.e. the signal not being used as a control signal, etc), fitting an a^k fall-off rate to approximate $1/n$ or $1/n^2$ may be close enough (perceptually, for example).

4.3 Appropriate Scalings/Offsets and Non-Steady-State Fixups

In order to keep the integrators from ramping their outputs to infinity (or at least beyond the capabilities of the number system or the DACs), any DC offset in the input to the integrator must be avoided. As already noted, BP-BLIT has no DC offset, so there need be no special offsets for the square-wave integration. The initial conditions of the first integrator, however, can produce a DC offset on the output that must be canceled before the second integration. The value of this offset is also dependant on the duty-cycle of the signal[3], so that the correct initial condition will change based on: (1) desired phase, and (2) desired duty cycle. There is also a frequency-dependant scaling necessary for the second integration (because the triangle slopes are proportional to frequency), whose effects must be accounted for during frequency changes. It is important to remember that the old state (right before the change) acts as a new "initial condition" for the integrator when the parameters are changed.

Leaky Integrators

The use of pure integrators can present problems in the presence of numerical errors, such as roundoff. These errors accumulate in the integrators, causing unwanted (and unpredicatble) offsets in the signals, which can destroy the ability to create the desired waveforms, especially in the second integration. Therefore, we move the poles of the filters that implement the integration slightly in from the unit circle. These "leaky" integrators slowly forget bad initial conditions and numerical errors, so that they don't continue the build up forever. The impulse response of a leaky integrator is an exponential that slowly decays to zero. This has two more effects:

1. The decay rate places an effective lower bound on frequency (especially in cases where the signal is to be used as a control signal), when the period gets on the order of the decay time, the 'held' output (as in a rectangle wave), is no longer even close to being constant over its portion of the cycle. In audio signals, this is probably not a problem (it just attenuates the lowest-frequency harmonics, which may not be audible anyway), but in control signals, where the exact shape of the signal is important, this can be a big problem.
2. In steady-state, the outputs of the integrators will have no DC component (because BP-BLIT has none), regardless of initial conditions, since the leaky integrators eventually forget them. Thus, if one can live with occasional transient DC offsets (which decay at the leak rate), then just the presence of the leaky integrators can handle all offset cases[4].

If one still requires the absence of any DC offset, transient or not, then the presense of the leaky integrators makes the problem of computing appropriate offsets much more difficult.

Defs of Amplitude

Moore presents a discussion of amplitude compensation in his DSF paper. Similar compensation is necessary in BLIT generation. The compensation to be used depends on how one defines amplitude, which depends on how the signal is to be used. If the signal is to be used as an audio signal, signal power or some psychoacoustic loudness measure is appropriate, but if the signal is to be used as a control signal, a maximum-value (Chebychev) measure is more appropriate.

5 Generating Bipolar BLITs in DSF

The above sections describing the generation of rectangle waves and triangle waves from BLITs did not prescribe which method need be used to generate the BLITs. This is because it doesn't matter which technique is used (to within numerical accuracy differences). It does turn out, however, that there are one or two interesting methods that can be employed in generating the bipolar BLITs necessary to generate rectangle waves.

Difference of BLITs The standard (straight-forward) method of generating the bipolar BLITs is simply to take the difference to two BLITs, one shifted relative to the other. This is the technique used in BLIT-SWS.

DSF method In cases where DSF is acceptable (efficiency-wise), a few refinements to the algorithm can be put to good use to directly generate bipolar BLITs.

[3] Any amplitude dependance can be avoided by assuming unit amplitude in all the integrations and simply post-scaling the outputs. Thus the scaling gains are: on input of first integrator, 1.0; on input of second integrator $2/(Td(1-d))$ where d is the duty cycle ($\in [0,1]$) and T is the period in samples.

[4] The transient DC offsets can be reduced by temporarily increasing the integrators' leak rate at times when transients are expected. This causes the integrators to 'center' themselves faster (at the expense of low frequencies), and then return to the desired steady-state leak rate after DC has been fixed.

First, we note that BLITs can be generated via DSF by replacing the sin by cos in the DSF formulas (this ends up essentially replacing sin by cos in the numerator of the original DSF equation, the denominator stays the same). See Figure 14.

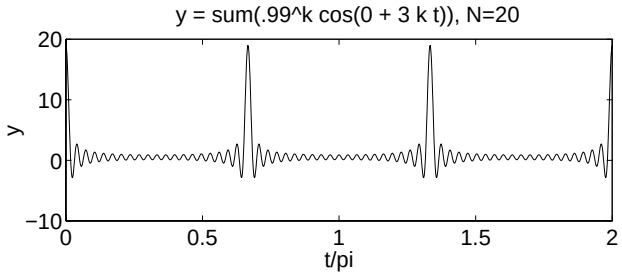

Figure 14: *Using cosine-DSF to generate BLIT*

50% duty cycle: First, it can be shown that using a negative a in the DSF formula $\sum_{k=1}^{N} a^k \sin(0 + k f_1 t)$ produces a signal that is shifted from the positive-a signal by exactly half a cycle (Figure 15), this gives a slightly more efficient (or elegant...) way of producing the shifted BLIT than offsetting t. This can lead to showing that:

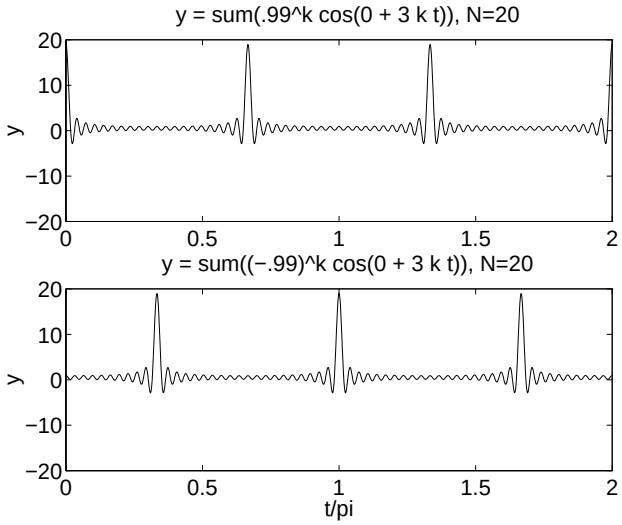

Figure 15: *DSF: half-cycle shift*

$$\sum_{k=1}^{N} a^k \sin(a+bk) \Leftrightarrow \sum_{k=1}^{N} (\Leftrightarrow a)^k \sin(a+bk)$$
$$= \; 2a \sum_{k=1}^{N/2} (a^2)^k \sin(\,(a+b) + 2bk)$$

The same applies to the sum-of-cosines DSF, which is shown in Figure 16. Thus a 50% duty-cycle bipolar DSF BLIT can be generated almost as efficiently as a single DSF BLIT.

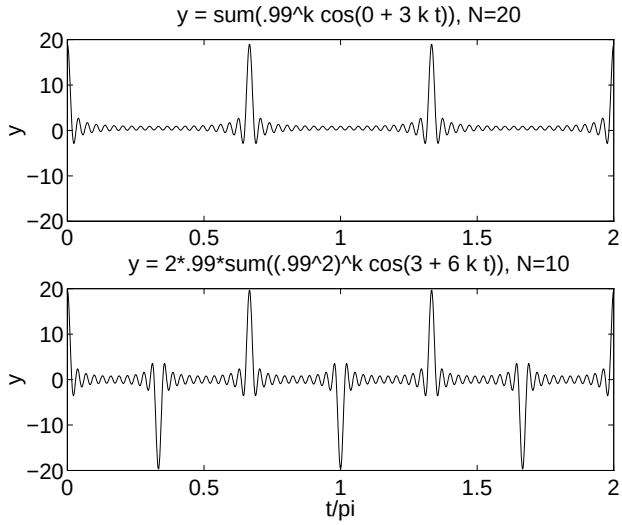

Figure 16: *50% duty-cycle BP-BLIT using DSF*

PWM: For other duty cycles, there is another variation on DSF that is of interest. Let a be complex and take either the real or imaginary part of the DSF, this imposes a $\sin(k\angle(a))$ (or cosine) amplitude envelope onto the harmonics, which is equivalent to a comb filtering, which in turn is equivalent to summing a real DSF with a shifted version of itself (possibly with a sign flip), all of which can be shown mathematically. See Figure 17. This method implements BP-BLIT in essentially the same complexity as evaluating the difference of two real DSF BLITs, but with a bit more elegance.

References

[Bellanger 1996] Bellanger, M. 1996. "Improved Design of Long FIR Filters using the Frequency Masking Technique." *In: Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Atlanta*. New York: IEEE Press. Paper DSP1.1.

[Chowning 1989] Chowning, J. M. 1989. "Frequency Modulation Sythesis of the Singing Voice.". *Pages 57–63 of:* Mathews, M. V., and J. R. Pierce (eds), *Current Directions in Computer Music Research*. Cambridge, MA: MIT Press.

[Committee 1979] Committee, D. S. P. (ed). 1979. *Programs for Digital Signal Processing*. New York: IEEE Press.

[Crochiere and Rabiner 1983] Crochiere, R., and L. R. Rabiner. 1983. *Multirate Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

[Dodge and Jerse 1985] Dodge, C., and T. A. Jerse. 1985. *Computer Music*. New York: Schirmer.

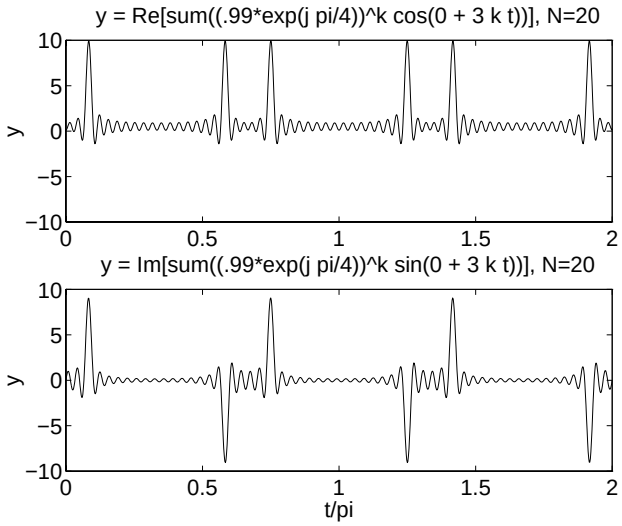

Figure 17: *Using a complex multiplier in DSF to generate BP-BLIT*

[Goeddel and Bass 1984] Goeddel, T. E., and S. C. Bass. 1984. "High Quality Synthesis of Musical Voices in Discrete Time." *IEEE Trans. Acoustics, Speech, Signal Processing*, 32(3):623–633.

[Kaegi and and S. Tempelaars 1978] Kaegi, and W. a S. Tempelaars. 1978. "VOSIM—A New Sound Synthesis System." *J. Audio Eng. Soc.*, 26(6):418–24.

[Mathews 1969] Mathews, M. V. 1969. *The Technology of Computer Music*. Cambridge, MA: MIT Press.

[Moorer 1975] Moorer, J. A. 1975. "The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulae." *J. Audio Eng. Soc.*, 24(Dec.):717–727 (Also svailable as CCRMA Report No. STAN-M-5).

[Papoulis 1991] Papoulis, A. 1991. *Probability, Random Variables, and Stochastic Processes, 3rd Edition*. New York, NY: McGraw-Hill, Inc.

[Rabiner and Gold 1975] Rabiner, L. R., and B. Gold. 1975. *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

[Roads 1996] Roads, C. 1996. *Computer Music Tutorial*. Cambridge, MA: MIT Press.

[Rodet *et al.* 1989] Rodet, X., Y. Potard, and J. Barrière. 1989. "The CHANT Project: From the Synthesis of the Singing Voice to Synthesis in General.". *Pages 449–465 of:* Roads, C. (ed), *The Music Machine*. Cambridge, MA: MIT Press.

[Schafer and Rabiner 1973] Schafer, R. W., and L. R. Rabiner. 1973. "A Digital Signal Processing Approach to Interpolation." *Proc. IEEE*, 61(June):692–702.

[Schanze 1995] Schanze, T. 1995. "Sinc Interpolation of Discrete Periodic Signals." *IEEE Trans. Signal Processing*, 43(6):1502–1503.

[Smith and Gossett 1984] Smith, J. O., and P. Gossett. 1984. "A Flexible Sampling-Rate Conversion Method." *Pages 19.4.1–19.4.2 of: Proc. Int. Conf. Acoustics, Speech, and Signal Processing, San Diego*, vol. 2. New York: IEEE Press (An expanded tutorial based on this paper is available in the directory `ftp://ccrma-ftp.stanford.edu/pub/DSP/Tutorials/`, file BandlimitedInterpolation.eps.Z, as is C code for implementing the technique in directory `ftp://ccrma-ftp.stanford.edu/pub/NeXT/`, file resample-n.m.tar.Z, where n.m denotes the latest version number. Note that the C source code is included so it is easy to port it to any platform supporting the C language. The tutorial can be browsed online with any Web browser at `http://www-ccrma.stanford.edu/~jos/`).

This paper can be found online at
`http://www-ccrma.stanford.edu/~stilti/papers`