Alias-Suppressed Oscillators Based on Differentiated Polynomial Waveforms

Vesa Välimäki, *Senior Member, IEEE*, Juhan Nam, Julius O. Smith, and Jonathan S. Abel

***Abstract*—An efficient approach to the generation of classical synthesizer waveforms with reduced aliasing is proposed. This paper introduces two new classes of polynomial waveforms that can be differentiated one or more times to obtain an improved version of the sampled sawtooth and triangular signals. The differentiated polynomial waveforms (DPW) extend the previous differentiated parabolic wave method to higher polynomial orders, providing improved alias-suppression. Suitable polynomials of order higher than two can be derived either by analytically integrating a previous lower order polynomial or by solving the polynomial coefficients directly from a set of equations based on constraints. We also show how rectangular waveforms can be easily produced by differentiating a triangular signal. Bandlimited impulse trains can be obtained by differentiating the sawtooth or the rectangular signal. An objective evaluation using masking and hearing threshold models shows that a fourth-order DPW method is perceptually alias-free over the whole register of the grand piano. The proposed methods are applicable in digital implementations of subtractive sound synthesis.**

***Index Terms*—Acoustic signal processing, antialiasing, audio oscillators, music, signal synthesis.**

I. Introduction

Subtractive synthesis is a traditional sound generation principle used in music synthesizers. Recently, it has become interesting because it is used in digital modeling of analog sound synthesis. The basic idea in subtractive synthesis is first to generate a signal with a rich spectral content, and then to filter that signal with a time-varying resonant filter. The input signal is usually a periodic classical waveform, such as a sawtooth, a rectangular, or a triangular wave, or a sum of two or more such waveforms with different fundamental frequencies. In developing a digital implementation of subtractive synthesis, the main challenge is to find computationally efficient algorithms for generating periodic waveforms that do not suffer from excessive aliasing distortion. This paper discusses a new method for the generation of the classical waveforms with reduced aliasing. It is based on differencing a digital signal that is obtained by sampling a piecewise polynomial waveform.

Manuscript received March 31, 2009; revised June 03, 2009. Current version published April 14, 2010. The work of V. Välimäki was supported by the Academy of Finland under Project 126310. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Dan Ellis.

V. Välimäki is with the Department of Signal Processing and Acoustics, TKK—Helsinki University of Technology, FI-02015 TKK, Espoo, Finland, and also with the Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, Stanford, CA 94305 USA (e-mail: vesa.valimaki@tkk.fi).

J. Nam, J. O. Smith, and J. S. Abel are with the Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, Stanford, CA 94305 USA (e-mail: juhan@ccrma.stanford.edu; jos@ccrma.stanford.edu; abel@ccrma.stanford.edu).

Digital Object Identifier 10.1109/TASL.2009.2026507

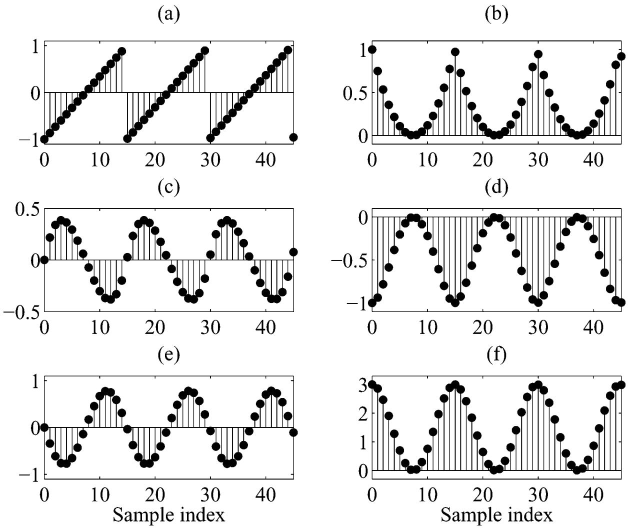


Fig. 1. Discrete-time signals generated by sampling piecewise polynomial periodic functions. (a) The linear ramp ($N = 1$), or the trivial sawtooth waveform, (b) the parabolic ($N = 2$), (c) the cubic ($N = 3$), (d) the fourth-order ($N = 4$), (e) the fifth-order ($N = 5$), and (f) the sixth-order ($N = 6$) polynomial function. The fundamental frequency of the signals is 2960 Hz (MIDI note #102) and the sampling rate is 44.1 kHz.

A straightforward algorithm to generate a discrete-time sawtooth wave with signal values between -1 and 1 samples the continuous-time rising ramp function and shifts it to remove the dc offset

$$s(n) = 2(nf_0T \bmod 1) - 1 \tag{1}$$

where $n = 0, 1, 2, \ldots$ is the sample index, f_0 is the fundamental frequency of the tone, and T is the sampling interval. An example of the signal produced using this method is given in Fig. 1(a). This bipolar modulo counter can be realized easily with a computer or a signal processor. Unfortunately, the discrete-time audio signal obtained using (1) is notorious for its distorted sound quality [1], [2]. The problem is that the continuous-time sawtooth wave is not bandlimited, but it has an infinite number of spectral components falling off at approximately -6 dB per octave. The discretization of the waveform leads to aliasing where all spectral components above the Nyquist limit $f_s/2$ are reflected down to the audible frequency range, when f_s is the sampling rate, see Fig. 2(a). This is heard as disturbing interference.

An easy way to improve the quality of the trivial sawtooth wave is to oversample the signal (see [1, pp. 116–117]). The aliasing resulting from using (1) is less severe when the fundamental frequency is small with respect to the sampling rate. In

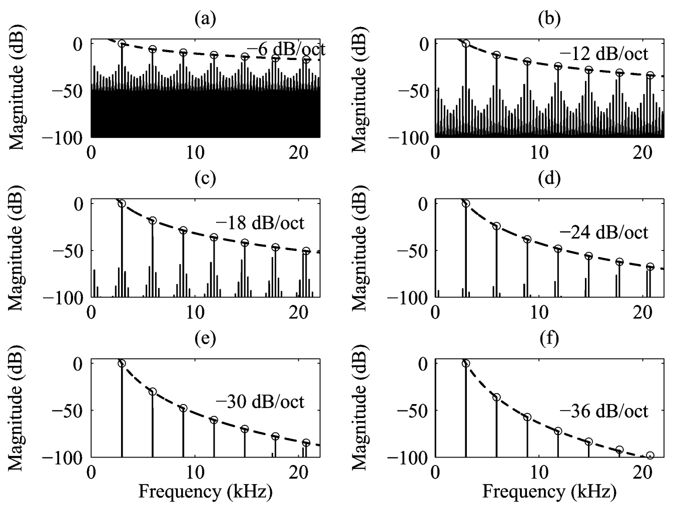


Fig. 2. Spectra of signals shown in Fig. 1. (a) $N = 1$, (b) $N = 2$, (c) $N = 3$, (d) $N = 4$, (e) $N = 5$, and (f) $N = 6$. The spectra were computed from 1-s-long signals using a 262 144-point fast Fourier transform (FFT) with a Chebyshev window that has its sidelobes at the level of -120 dB. The circles indicate the desired harmonic components. The dashed line shows in each case the spectral slope, which is approximately $-6N$ dB/octave.

practice, this approach leads to a highly oversampled and therefore inefficient system, because the spectrum of the sampled sawtooth wave decays slowly [3].

An ideal bandlimited sawtooth wave can be computed using additive synthesis, in which each harmonic component is generated separately using a sinusoidal oscillator [4], [5]. The ideal sawtooth waveform can be expressed as

$$s_{\mathrm{id}}(n) = -\sum_{k=1,2,3,\ldots}^{K} \frac{1}{k}\sin(2\pi k f_0 nT) \qquad (2)$$

where $K = \mathrm{floor}(f_s/2f_0)$ is the number of harmonics below the Nyquist limit that will be computed. The drawback of this approach is that the number of sinusoidal oscillators becomes large when the fundamental frequency is low. Computational savings can be obtained by omitting the highest frequencies or by storing sets of sinusoidal components in several wavetables, as in group additive synthesis [6]. Nevertheless, this approach is considerably more costly than (1).

Generation of alias-free waveforms for subtractive synthesis has received consideration over the years. Winham and Steiglitz [7] and Moorer [8] have proposed the use of a closed-form summation formula to generate a waveform containing a specified number of sinusoidal components. A low-order digital filter can shape the roll-off rate of the spectrum so that, for example, the sawtooth waveform can be closely approximated. While this method does not require as many operations per sample as additive synthesis at low frequencies, it has its drawbacks: it requires a division per sample, and this leads to numerical errors that must be controlled. One solution is to compute many versions of the bandlimited sawtooth waveform using (2) with a different number of harmonic components, and then use them in wavetable synthesis [4]. This is efficient to compute and free of numerical problems, but it consumes much memory. A variation that saves memory consists of a small set wavetables, such as one per octave, which are played at the various speeds using sample rate conversion techniques [9].

Stilson and Smith [10], [11] have introduced the idea of generating alias-free waveforms using a bandlimited impulse train (BLIT) and filters. This method uses an oversampled windowed sinc function that is stored in memory. Recently, the modified frequency modulation technique has been proposed for synthesizing an approximately bandlimited pulse train [12]. Brandt has proposed to linearly combine a trivial waveform with a bandlimited pulse to reduce aliasing [13]. The pulse used is a minimum-phase bandlimited step function (minBLEP) obtained by integrating a minimum-phase windowed sampled sinc function [13]. The computational load of the BLEP-based algorithm depends on the fundamental frequency. The algorithm consumes some memory, because an oversampled BLEP residual sequence must be designed and stored in advance for real-time implementation. The BLEP residual is the difference of an ideal step function and the BLEP approximation [2]. Recently, it has been shown that the BLEP residual sequence can be obtained by sampling a low-order polynomial curve [2]. The table lookup is not needed then and the oversampling factor is replaced with a fractional delay variable, which can vary continuously.

Lane *et al.* introduced a different approach to derive antialiasing oscillator algorithms. They full-wave rectify a sine wave and modify the resulting signal with digital filters [14]. Classical waveform approximations computed with Lane's algorithm contain some aliased components, but they are suppressed in comparison to the trivial waveforms. Pekonen and Välimäki used recently highpass and comb filtering as a postprocessing method to suppress aliasing in waveforms [15].

It has been shown that a signal that closely approximates the sawtooth wave but having less aliasing can be produced by differentiating a piecewise parabolic waveform [16]. The simplest version of this algorithm generates the bandlimited sawtooth waveform approximation in four stages: First a trivial sawtooth waveform is generated using the bipolar modulo counter (1), then the signal is raised to the second power, it is differentiated with a first difference filter with transfer function $1 - z^{-1}$, and, finally, the obtained waveform is scaled. A two-times oversampled version of the differentiated parabolic wave algorithm yields improved alias suppression and leaves room for optimization by using various choices of decimation filters [16], [2].

In this paper, we propose an extension to the differentiated parabolic wave method mentioned above. Our ultimate design goal is not to suppress the aliasing completely, but to reduce it sufficiently at selected frequency areas so that it will be inaudible. We take advantage of the frequency-dependent sensitivity and the frequency masking phenomenon of the human auditory system [17]. Other goals are computational efficiency and small memory space requirement. In this paper we introduce a new class of polynomial waveforms that must be differentiated more than once to obtain an improved version of the sampled sawtooth signal. Suitable polynomials of order higher than two can be derived either by analytically integrating a previous lower order polynomial or by solving the polynomial coefficients directly from a set of equations based on constraints. A multirate version of the algorithm oversamples and decimates the polynomial signal with a simple filter prior to differentiation. Another new class of polynomial signals can be differentiated to produce alias-suppressed triangular signals.

The basics of the new approach are introduced and a multirate method is considered in Section II. Section III suggests algorithms to synthesize triangular, rectangular, and impulse train signals. The audio quality of the produced waveforms is analyzed and is compared against previous methods in Section IV. Section V concludes this paper.

II. Sawtooth Signal Generation With the Differentiated Polynomial Wave Method

In this section, we show that it is possible to extend the principle used in the differentiated parabolic wave method for the generation of sawtooth waveforms to polynomials of order higher than two. The motivation to use a high-order piecewise polynomial signal with polynomial order N is that its spectrum decays about $-6N$ dB per octave. For comparison, the spectrum of the sawtooth wave falls off only about -6 dB per octave. The faster roll-off rate leads to suppressed aliasing [16], [18], [2]. Multiple differentiations can restore the desired spectral tilt for the harmonic components. In the following, it is shown how the desired polynomials can be derived and how the resulting signal can be scaled. A multirate implementation is also discussed.

A. Derivation of Polynomials Using Analytical Integration

It was shown in a previous work [16] that a digital signal that closely resembles the sawtooth wave, but with a smaller aliasing error, can be generated by differentiating a sampled, piecewise parabolic waveform. The piecewise parabolic waveform can be obtained by integrating one period of the sawtooth waveform, which is a linear function within the period, that is

$$f_2(x) = \int x\mathrm{d}x = \frac{x^2}{2} + C. \tag{3}$$

Scaling can be dealt with separately, so we may use the polynomial x^2 for synthesis. Fig. 1(b) shows the sampled parabolic waveform, and Fig. 2(b) shows its spectrum, which decays about 12 dB per octave.

In principle, differentiating (3) restores the piecewise linear function. However, when a discrete-time signal consisting of samples taken from the second-order polynomial curve is differentiated with a digital filter, the result is different from a trivial sawtooth waveform: one or more samples near the joint of two contiguous periods will be different. This is seen in Fig. 3(b), which shows the resulting discrete-time signal when a first-order difference $1 - z^{-1}$ has been applied. A single modified sample forms a smooth transition from one period to the next one. This time-domain property of the differentiated polynomial waveforms (DPW) waveform reduces aliasing. Fig. 4(b) shows the spectrum of the second-order DPW sawtooth waveform, which

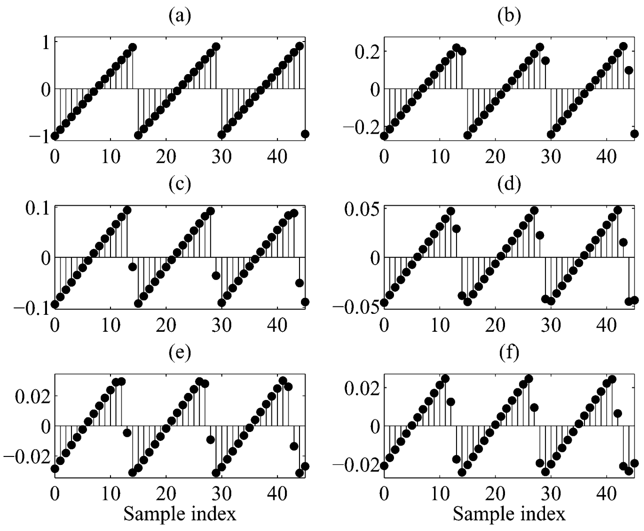


Fig. 3. Alias-reduced sawtooth waveforms generated by differencing polynomial periodic functions $N-1$ times. (a) The trivial sawtooth waveform, (b) the parabolic, (c) the cubic, (d) the fourth-order, (e) the fifth-order, and (f) the sixth-order DPW signals.

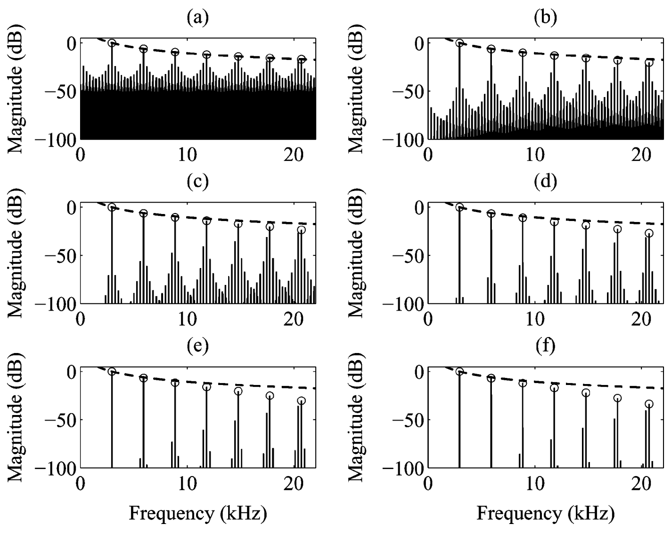


Fig. 4. Spectra of the sawtooth waveforms shown in Fig. 3. (a) The trivial sawtooth waveform, (b) the parabolic, (c) the cubic, (d) the fourth-order, (e) the fifth-order, and (f) the sixth-order polynomial function. The parameters used in the spectral analysis are the same as in Fig. 2. The circles indicate the desired harmonic components. The dashed line shows the ideal spectral slope of the sawtooth signal, which is approximately -6 dB/octave.

has generally weaker aliased components than the spectrum of the trivial sawtooth waveform, cf. Fig. 4(a).

We can continue integrating (3), but an arbitrary constant of integration C must be assumed:

$$\begin{aligned} f_3(x) &= \int [f_2(x)]\,\mathrm{d}x \\ &= \int \left(\frac{x^2}{2} + C\right)\mathrm{d}x = \frac{x^3}{6} + Cx. \end{aligned} \tag{4}$$

The constant C can be determined by the reasonable restriction that function $f_3(x)$ must not contain a discontinuity when repeating, that is, $f_3(-1) = f_3(1)$. We then obtain $C = -1/6$, and thus $f_3(\mathrm{x}) = (x^3 - x)/6$. Again, we may skip the scaling here and use $f_3(x) = x^3 - x$ instead. Fig. 1(c) plots the sampled version of this polynomial signal.

Fig. 2(c) shows that the spectrum of the cubic waveform has about 6 dB per octave steeper slope than the parabolic waveform. This leads to further reduction of aliasing. Filtering twice with a first-order difference yields a sawtooth-like signal, which has a transition region of two samples between each period, see Fig. 3(c). The corresponding spectrum is shown in Fig. 4(c). We also see that the level of harmonics at high frequencies above 10 kHz is several dB too low. This is caused by the poor approximation of the differentiation by a first-order finite difference. We have found that it is possible to compensate for this approximation error using a second-order IIR equalizer, if desired. Then the level of harmonics will not deviate more than a fraction of a decibel in the audio range.

By integrating the cubic polynomial, we obtain the fourth-order polynomial, and so on. However, we will next show how to derive the polynomial functions directly.

B. Derivation of Polynomials by Solving Linear Equations

In this section, we show that it is possible to directly solve for the coefficients of an Nth-order polynomial from constraints that describe certain desired characteristics of the polynomials. Let

$$f_N(x) = a_N x^N + a_{N-1}x^{N-1} + \cdots + a_1 x + a_0 \quad (5)$$

denote the Nth-order polynomial. The coefficients are unconstrained at this point. Without loss of generality, we can let the polynomial be monic (i.e., $a_N = 1$), because a scaling factor can separately deal with the overall level. The key idea in the class of techniques that we propose is to impose a maximally flat wrap-around at the switch-back point in the sawtooth signal. Variable x ranges from -1 to 1, and we do not want a constant term for this first-order polynomial, so we can set $a_{N-1} = 0$. There remain $N-1$ coefficients $a_k, k = 0, \ldots, N-1$ that we may choose so as to maximize spectral roll-off rate in the signal (cf. (1)

$$s_N(n) = f_N[2(nf_0T \bmod 1) - 1]. \quad (6)$$

As is well known in the context of windows used for Fourier analysis, spectral roll-off rate is maximized by means of "maximally flat" tapering at the discontinuity points (window endpoints). Specifically, a maximally flat "splice" of the function $f(x)$ is obtained at x by matching the maximum number of leading terms in the Taylor expansion of f about x to zero. In the present context, we wish to equate as many leading derivatives of $f(x)$ as possible at the "wrap-around points" $x = \pm 1$. That is, we desire

$$\begin{aligned} f_N(-1) &= f_N(1) \\ f'_N(-1) &= f'_N(1) \\ f''_N(-1) &= f''_N(1) \end{aligned} \quad (7)$$

and so on, for as many derivatives as possible. (Note that cubic splines, being third-order polynomials, match both function value and slope at each spline junction, or "knot.")

Even and odd polynomials are of interest in this work, because the derivative of an even-order polynomial is odd, and vice versa. Furthermore, every polynomial can be written as the sum of an even and odd polynomial. Every even polynomial satisfies a desired property that $f(-1) = f(1)$, and the same is also true for every even-order derivative of f. Since every other coefficient of an even polynomial is zero, the number of odd derivatives is comparable to the number of coefficients. Thus, substantially all degrees of freedom in an even polynomial are devoted to satisfying the maximally flat constraints we seek.

For odd polynomials, the only solution for $f(-1) = f(1)$ is the zero solution, so they must have the property that their coefficients sum to zero, which leads to $f(1) = 0$. Since f is odd, we obtain $f(-1) = 0$ automatically. Any even number of derivatives of an odd polynomial is odd, so similar remarks apply. An odd number of derivatives gives an even polynomial, which is spontaneously maximally flat at the wrapping points, as discussed in the previous paragraph.

Choosing an even or odd starting polynomial—depending on whether N is even or odd—constrains every second coefficient to zero. Additionally, we can argue that a_0 can be zero: in the even case, it is arbitrary, and in the odd case, it must be zero, so it may as well be zero in every case.

For even N, we have a polynomial of the form

$$f_{\text{even}}(x) = x^N + a_{N-2}x^{N-2} + a_{N-4}x^{N-4} + \cdots + a_4x^4 + a_2x^2 \quad (8)$$

and $f(-1) = f(1)$ already no matter what coefficient values are chosen. For odd N, we have a polynomial of the form

$$f_{\text{odd}}(x) = x^N + a_{N-2}x^{N-2} + a_{N-4}x^{N-4} + \cdots + a_3x^3 + a_1x \quad (9)$$

and we must have $1 + a_{N-2} + a_{N-4} + \cdots + a_3 + a_1 = 0$.

Going back to the even case, to obtain our first constraint, we must differentiate and equate the coefficient sum to zero. Then we differentiate twice more, equate the sum of coefficients to zero, and so on, until we get down to order 3 which determines a_{N-2} (because $a_N = 1$). Finally, we back-solve the previous equations in reverse order to obtain all coefficients. In matrix form, it is a simple upper triangular system with integer entries made from every other derivative. Matrix inversion is not needed, as the equations can be solved starting from the bottom.

The odd case goes the same way, except that we start with $f(x)$ instead of its derivative, i.e., the coefficients of $f(x)$ must sum to zero, as well as those for $f''(x), f''''(x)$, and so on down to order 2, where a_{N-2} is again determined and the rest can be found by back-solving.

With the above constraints, it is possible to derive a polynomial function for any N. A few example cases are shown next.

1) Examples: The second-order case is of the form x^2, and there are no other terms or coefficients, because we must have $a_1 = 0$ (to keep it even) and we can choose $a_0 = 0$ (no effect), that is

$$f_2(x) = x^2. \quad (10)$$

This case has been known for several years from an earlier work [16], where it was derived by integrating a single period of a linear ramp function, as in (3).

TABLE I
POLYNOMIAL FUNCTIONS OF ORDER 1 TO 6 FOR GENERATING SAWTOOTH WAVEFORMS AND THEIR COMPUTATIONAL LOAD. THE NUMBER OF MULTIPLICATIONS (MUL) AND ADDITIONS (ADD) TO EVALUATE EACH POLYNOMIAL AND TO IMPLEMENT N DIFFERENTIATORS AND THE SCALING ARE CONSIDERED

Polynomial order	**Polynomial function**	**MUL**	**ADD**	**MUL (scaling)**	**ADD (diff)**	**Total**
$N = 1$	x	0	0	0	0	0
$N = 2$	x^2	1	0	1	1	3
$N = 3$	$x^3 - x$	2	1	1	2	6
$N = 4$	$x^4 - 2x^2$	3	1	1	3	8
$N = 5$	$x^5 - 10x^3/3 + 7x/3$	5	2	1	4	12
$N = 6$	$x^6 - 5x^4 + 7x^2$	5	2	1	5	13

For the third-order case ($N = 3$), the form is $f(x) = x^3 + ax$. Since N is odd (making the polynomial odd), these coefficients must sum to zero. This yields $a = -1$, and the cubic polynomial becomes

$$f_3(x) = x^3 - x. \tag{11}$$

This is equivalent to the third-order function that we obtained previously with the integration method in Section II-A

For $N = 4$, the form is $f(x) = x^4 + ax^2$. Differentiating with respect to x yields $f'(x) = 4x^3 + 2ax$. Setting $4 + 2a = 0$ yields $a = -2$. The fourth-order polynomial is thus

$$f_4(x) = x^4 - 2x^2. \tag{12}$$

For $N = 5$, the form is $f(x) = x^5 + ax^3 + bx$ yielding $1 + a + b = 0$ for our first constraint. Next, we need the first and second derivatives, $f'(x) = 5x^4 + 3ax^2 + b$ and $f''(x) = 20x^3 + 6ax$. Setting $20 + 6a = 0$ yields $a = -10/3$, and back-solving gives $b = 10/3 - 1 = 7/3$. The fifth-order polynomial is then

$$f_5(x) = x^5 - \frac{10}{3}x^3 + \frac{7}{3}x. \tag{13}$$

For $N = 6$, we start with $f(x) = x^6 + ax^4 + bx^2$ and go to $f'(x) = 6x^5 + 4ax^3 + 2bx$, giving $6 + 4a + 2b = 0$, or $3 + 2a + b = 0$ for (1). Next, $f''(x) = 30x^4 + 12ax^2 + 2b$ and $f'''(x) = 120x^3 + 24ax$, which gives $a = -120/24 = -5$, and back-solving gives $b = -2a - 3 = 7$. The sixth-order polynomial is therefore

$$f_6(x) = x^6 - 5x^4 + 7x^2. \tag{14}$$

Similarly, it is possible to derive a polynomial of any order N for this purpose. We will use the above polynomials of orders 1 to 6 in the examples of this paper. They are also included in Table I. For practical implementations, we suggest to use the first-order difference $1 - z^{-1}$ as an approximate for derivative, as was done previously in the second-order case [16].

C. Scaling Factors

When the linear ramp function is raised to a high power and differenced several times, the signal level becomes small. For this reason, signal scaling is a crucial part of the synthesis method. We derive next the scaling factors needed for different orders of polynomials.

1) Continuous Polynomial Scaling: When an Nth-order monic polynomial, such as (5) with $a_N = 1$, is differentiated $N - 1$ times, it becomes a first-order polynomial of the form $N!\,x + C$, where $N!$ denotes the factorial of N, and C is zero by construction in our case (since we employ only even or odd polynomials). Therefore, the scale factor for ideal differentiators is $1/N!$.

2) Discrete Waveform-Preserving Scaling: When the ideal differentiator is replaced by a first-order finite difference, we have instead $D\,s_N(n) = s_N(n) - s_N(n-1)$, where D denotes the first-order difference operator, and $s_N(n)$ is defined in (6). If there is no modulo operation between time-steps $n - 1$ and n, then the highest-order term in n is found to be $(2f_0/f_s)N\,n^{N-1}$. Thus, the n^N-order term is canceled and the polynomial order is reduced by one. After $N - 1$ applications of the difference operator, we obtain

$$D^{N-1}s_N(n) = \left(2\frac{f_0}{f_s}\right)^{N-1} N!\,n + C. \tag{15}$$

It can be shown that C corresponds to a half-sample delay per application of D, which we will ignore (since it is easily compensated by a time shift). Thus, our waveform-preserving scaling factor for the finite-difference case becomes

$$\hat{c}_N = \frac{f_s^{N-1}}{(2f_0)^{N-1}N!} = \frac{P^{N-1}}{2^{N-1}N!} \tag{16}$$

where $P = f_s/f_0$ denotes the waveform period in samples. The scaling factors of this form are presented in Table II, second column, for $N = 2$ to 6. Notice that the scaling factors are proportional to P^{N-1} and inversely proportional to the fundamental frequency f_0 raised to the power $N - 1$. The scaling factors also contain a large integer in the denominator. To avoid the division, when the fundamental frequency is changed, the scaling factors can be precomputed and stored in a table.

3) Fundamental-Matching Scaling: The previously derived scale factor for differenced polynomials exactly preserves the sawtooth waveform at the sample points over regions not including the modulo discontinuity. While some applications may prefer this type of scaling, it is more typical in audio applications to optimize the scaling according to *psychoacoustic* criteria. Since a scale factor offers only one degree of freedom, it can be argued that the best use for it is to maintain correct amplitude at the fundamental frequency, especially as that frequency

TABLE II
SCALING FACTORS FOR DIFFERENTIATED POLYNOMIAL WAVEFORMS OF ORDERS $N = 1$ TO 6. THE VARIABLE P DENOTES THE RATIO f_s/f_0

Polynomial order	**Basic scaling factor**	**Improved scaling factor**
$N = 1$	1	1
$N = 2$	$P/4$	$\pi/[4\sin(\pi/P)]$
$N = 3$	$P^2/24$	$\pi^2/(6\times[2\sin(\pi/P)]^2)$
$N = 4$	$P^3/192$	$\pi^3/(24\times[2\sin(\pi/P)]^3)$
$N = 5$	$P^4/1920$	$\pi^4/(120\times[2\sin(\pi/P)]^4)$
$N = 6$	$P^5/23040$	$\pi^5/(720\times[2\sin(\pi/P)]^5)$

is changed dynamically. Thus, we derive a scale factor that cancels the frequency-dependent difference between the ideal differentiator and the first-difference filter at the fundamental.

The frequency response of an ideal differentiator is $j\omega$ while that of a first-order difference is $e^{-j\omega T/2}2j\sin(\omega T/2)$, where ω is radian frequency, j is the imaginary unit, and T denotes the sampling interval. Neglecting the linear phase term corresponding to a half-sample delay, correcting the amplitude at $\omega_0 = 2\pi f_0$ is accomplished by the scale factor

$$c_N = \hat{c}_N \left[\frac{\omega_0 T}{2\sin\left(\frac{\omega_0 T}{2}\right)}\right]^{N-1} = \frac{\pi^{N-1}}{N!\left[2\sin\left(\frac{\omega_0 T}{2}\right)\right]^{N-1}}. \tag{17}$$

This scaling factor is included in the third column of Table II for each polynomial order N.

The scaling factors according to (17) are displayed in Fig. 5(left) for polynomial orders $N = 2$ to 6. It is seen that the gain is very large, when the fundamental frequency is low. For example, for $N = 6$, the required scaling factor for the lowest note on the piano ($f_0 = 27.5$ Hz) is 4.9×10^{11}, or about 233 dB. Implementing a multiplication with a large number like this requires special care in finite-accuracy arithmetic systems, such as in 16-bit fixed-point signal processors. A solution is to implement the scaling in different stages: The first part of the scaling factor can be used after evaluating the polynomial and the rest between the first-difference filters, which reduce the amplitude of low-frequency signals. In practice, the multistage scaling means that more multiplications than just one must be used, though all but one of them can be approximated by the nearest power of 2.

Using the scaling factors presented in Table II, the amplitude of the sawtooth signals generated using the DPW oscillator can be limited not to exceed 1.0. This can be seen in Fig. 5(right), where the amplitude of sawtooth wave signals with 88 fundamental frequencies ranging from 27.5 Hz to 4186 Hz is shown. The amplitude has been estimated as the absolute maximum of sample values from a 1-s-long signal. For high fundamental frequencies, the amplitude is less than one, or 0 dB, but no more than about 2.5 dB less.

4) Bounded Scaling: We may alternatively solve for the scaling factors that are needed to maintain a maximum value of one. We again assume that the first difference operation is used to approximate differentiation. In the case of the parabolic

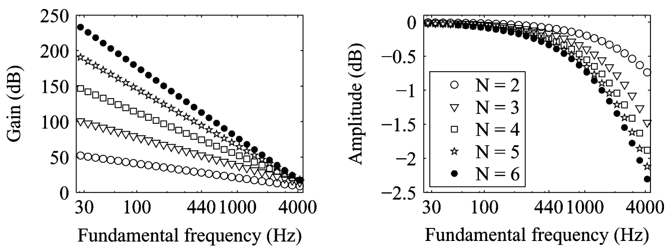


Fig. 5. (left) Gain factors of the differentiated polynomial waveforms of orders $N = 2$ to 6, see Table II, third column, and (right) amplitude of the sawtooth waveforms using these scaling factors. For $N = 6$, the first octave is opted out in the right figure, since the scaling factor is very large. Fundamental frequencies between 27.5 Hz (MIDI note #21) and 4186 Hz (MIDI note #108) are considered.

polynomial of (10), this can be done by requiring that the scaled difference of the largest and second largest sample values is 1.0, that is

$$c_2\left[1^2 - \left(1 - \frac{2}{P}\right)^2\right] = 1 \tag{18}$$

where c_2 is the scaling factor. We can solve c_2 as

$$c_2 = \frac{P}{4\left(1 - \frac{1}{P}\right)}. \tag{19}$$

As mentioned in [18], this can well be replaced with a simplified version $c_2 = P/4$, which is also the factor that we obtained previously using another method, as seen in Table II. This approximation deviates from (19) by less than 1 dB, when the fundamental frequency is below 4.0 kHz and the sampling frequency is 44.1 kHz.

Instead of deriving exact polynomial formulas for the scaling factors of high-order methods in this way, we propose to use simple approximations shown in Table II.

D. Computational Complexity

Next we discuss the computational load of the proposed methods. Table I presents the number of operations needed for different parts of the algorithms, such as evaluating the polynomials and scaling. It is seen that the number of multiplications, including scaling, is about N. The total number of operations turns out to be about $2N$, where N is the polynomial order. It is assumed that the first-difference filter is used to implement all derivatives.

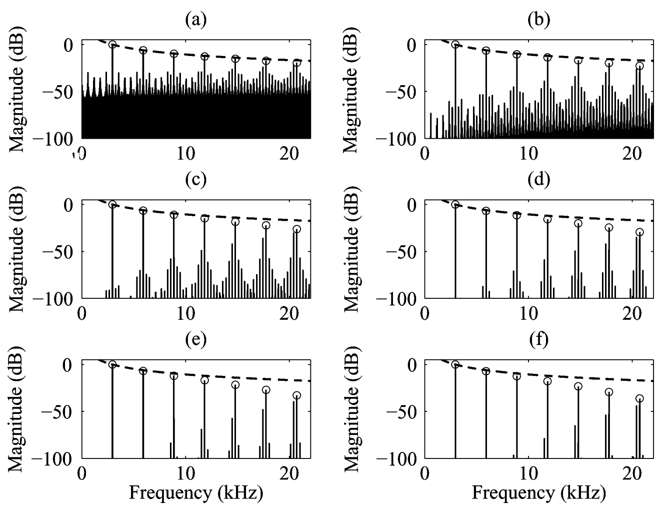


Fig. 6. Spectra of the sawtooth waveforms generated using the two times oversampled method, where a two-point averager is used as the decimation filter. (a) The trivial sawtooth waveform, (b) the parabolic, (c) the cubic, (d) the fourth-order, (e) the fifth-order, and (f) the sixth-order polynomial function. The circles indicate the desired harmonic components, and the dashed line is the spectral envelope of the ideal sawtooth waveform.

E. Multirate Method

It has been shown previously that an oversampled version of the second-order DPW algorithm can improve the alias-reduction at low frequencies [16], [2]. The multirate version of the algorithm oversamples and decimates the polynomial signal with a first-order filter prior to differentiation. It is expected that this idea should perform very well also when third or higher order polynomials are used. The steepened spectral roll-off rate, which is about -18 dB per octave for $N = 3$ and -6 dB more for each higher order, helps to further reduce aliasing.

Fig. 6 shows the spectra of sawtooth signals produced using polynomials of order $N = 2$ to 6 with two times oversampling. A two-point averaging filter is used for decimating the signal, as proposed in [16]. Fig. 6 should be compared against Fig. 4 to see whether the improvements are useful. It can be noticed that the aliasing is reduced in all cases at low frequencies, particularly below the first harmonic, which is at 2960 Hz in this case. The improvement gets better as N increases. In fact, the oversampled method of order $N = 4$ [see Fig. 6(d)] is practically as good as the nonoversampled method of order $N = 5$ [see Fig. 4(e)].

However, since the computational complexity is also doubled, when two samples must be generated in each sampling interval, it is impractical to use the oversampling method. Notice that according to Table I, increasing the polynomial order generally increases the computational burden less than oversampling. An exception is when N is increased from 2 to 3, since then the number of operations is exactly two times larger. However, the regular method with $N = 3$ is still better than the oversampled method with $N = 2$, so oversampling does not give any advantage even in this case. The DPW method is itself a very efficient signal processing method, which cannot be improved with a simple trick.

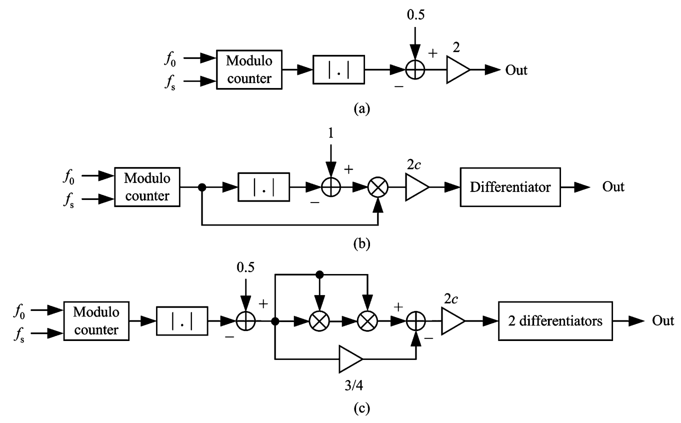


Fig. 7. Algorithms to generate (a) the trivial triangular, (b) the differentiated bipolar parabolic ($N = 2$), and (c) the differentiated cubic ($N = 3$) triangular waveforms. The modulo counter produces a trivial sawtooth signal with values between -1 and 1. The block containing $|\cdot|$ full-wave rectifies its input signal.

III. Synthesis of Other Waveforms

In the section, we will show how to realize various classical waveforms using higher-order DPW methods. These include the triangular and square waveforms and the bandlimited impulse train.

A. Triangular Wave

The trivial triangular waveform can be obtained by sampling a function that goes linearly from -1 to $+1$ and linearly back to -1 in each period. This waveform can be generated from the bipolar module counter (1) in the following way:

$$s_t(n) = 1 - 2|s(n)|. \tag{20}$$

The block diagram of this algorithm is shown in Fig. 7(a).

It has been shown in [18] that differentiating a bipolar parabolic waveform leads to an alias-suppressed triangular signal. The bipolar parabolic waveform is obtained as

$$s_b(n) = s(n)[1 - |s(n)|]. \tag{21}$$

Here the term $[1 - |s(n)|]$ corresponds to the fullwave rectified sawtooth signal, which is a triangular signal with a dc offset. Note that it deviates from (20) because of its dc level. The multiplication of the bipolar modulo counter signal and its shifted, full-wave-rectified version leads to a piecewise parabolic waveform that has its every second cycle inverted (nonpositive), see Fig. 8(b). Differentiation and scaling produces the alias-suppressed triangular signal:

$$s_{t,2}(n) = cDs_b(n). \tag{22}$$

Depicted in Fig. 7(b), this new algorithm is simpler than the one presented earlier by Välimäki and Huovilainen [18], but both algorithms produce the same output signal.

Alias-suppressed triangular signals with differentiated high-order polynomial signals can be derived as follows. The polynomial signals can be obtained by applying a polynomial function f to the trivial sawtooth wave (1). Each polynomial function f of order N must be an odd function between points $x =$

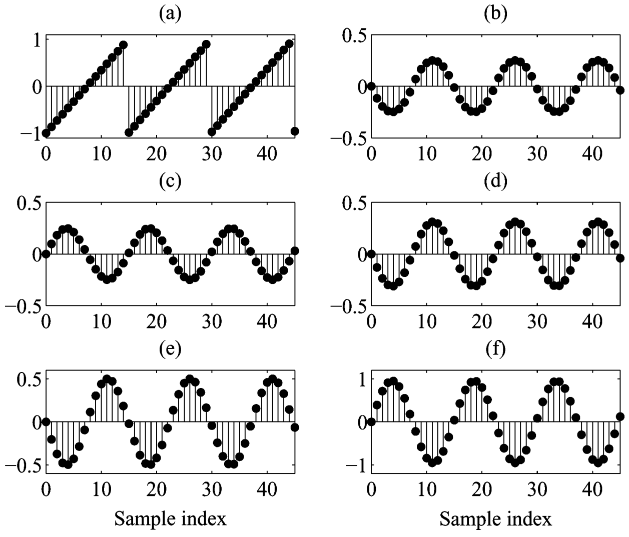


Fig. 8. Signals generated by sampling periodic polynomial functions used for producing triangular signals. (a) The bipolar module counter (trivial sawtooth waveform), (b) the bipolar parabolic ($N = 2$), (c) the cubic ($N = 3$), (d) the fourth-order ($N = 4$), (e) the fifth-order ($N = 5$), and (f) the sixth-order ($N = 6$) polynomial signals.

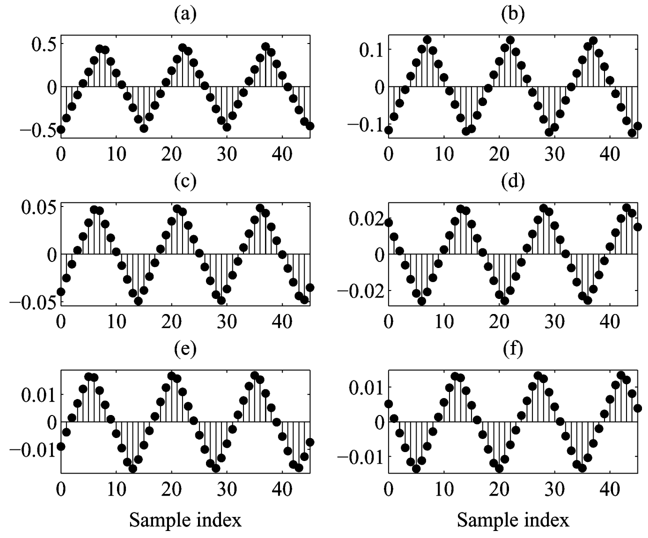


Fig. 9. (a) Trivial triangular and alias-reduced triangular waveforms of orders (b) $N = 2$, (c) $N = 3$, (d) $N = 4$, (e) $N = 5$, and (f) $N = 6$ generated by differencing polynomial functions of Fig. 8 $N - 1$ times.

$-(1/2)$ and $x = (1/2)$. Additionally, the polynomial must have a special property that its two negative and positive quadrants are symmetric with each other with respect to the points $x = -(1/2)$ and $x = (1/2)$, that is, $f((1/2) - x) = f(x + (1/2))$, because the waveform must contain only odd harmonics of the fundamental. The continuous-time triangular waveform itself fulfills these requirements.

We use monic polynomials and skip the constant terms, as was previously done in Section III. Interestingly, we must now include fullwave rectified terms $|x|$, like in (20) and (21). Note that even powers of x, such as x^2 and x^4, are even functions between -0.5 and 0.5, and thus we must replace them with $x|x|$ and $x^3|x|$, respectively. We may again investigate even and odd polynomial functions separately, as in Section III. For even N, we use a polynomial function of the following general form:

$$f_{\text{even}}(x) = a_N x^{N-1}|x| + a_{N-1}x^{N-1} + a_{N-2}x^{N-3}|x| + a_{N-3}x^{N-3} + \cdots + a_4 x^3|x| + a_3 x^3 + a_2 x|x| + a_1 x. \quad (23)$$

For odd N, we use the following form:

$$f_{\text{odd}}(x) = a_N x^N + a_{N-1}x^{N-2}|x| + a_{N-2}x^{N-2} + \cdots + a_3 x^3 + a_2 x|x| + a_1 x. \quad (24)$$

Values of coefficients a_k can be derived for any N by applying continuity conditions for the function itself and its derivatives. We match the derivatives from each side at $x = 0$, that is

$$\begin{aligned} f'(x = 0^+) &= f'(x = 0^-), \\ f''(x = 0^+) &= f''(x = 0^-), \\ f'''(x = 0^+) &= f'''(x = 0^-) \end{aligned} \quad (25)$$

and so on, where 0^+ means that zero is approached from the positive side. In addition, for odd-order polynomials we match the derivatives at $x = 0.5$ using the following conditions:

$$\begin{aligned} f'(x = 0.5^+) &= f'(x = 0.5^-) \\ f''(x = 0.5^+) &= f''(x = 0.5^-) \\ f'''(x = 0.5^+) &= f'''(x = 0.5^-) \end{aligned} \quad (26)$$

and so on. Since the polynomial functions are symmetric about the point $x = 0.5$, the above conditions are always true for even-order derivatives. Therefore, for odd-order derivatives only the following conditions are necessary:

$$\begin{aligned} f'(x = 0.5) &= 0, \\ f'''(x = 0.5) &= 0, \\ f'''''(x = 0.5) &= 0 \end{aligned} \quad (27)$$

and so on.

Based on the matching of derivatives at each side of zero, the terms of the form $x^{N-1}|x|$, when N is even, must have a weight zero. The reason is that the sign of the derivative is different on each size, and therefore the only way to match the term is to set their weighting coefficient to zero.

We can verify that (19) uses a second-order polynomial function ($N = 2$) of the form $f_2(x) = x|x| + a_1 x$ that satisfies the derivative matching at $x = 0.5$. The derivative is $2x + a_1$. Evaluating it at $x = 0.5$ gives $1 + a_1$. Setting $1 + a_1 = 0$ yields $a_1 = -1$. Thus the second-order polynomial function is

$$f_2(x) = x|x| - x = x(|x| - 1). \quad (28)$$

Fig. 8(b) presents the waveform obtained by inserting the bipolar modulo counter signal into this polynomial function. Differentiation yields the triangular signal approximation shown Fig. 9(b). Fig. 7(b) shows the block diagram of the whole algorithm.

For $N = 3$, the form is $f_3(x) = x^3 + a_2 x|x| + a_1 x$. We need to compute the first and second derivatives of the function on the positive and negative side of zero to see that a_2 must be

TABLE III
POLYNOMIAL FUNCTIONS OF ORDER 1 TO 6 AND THE INPUT SIGNALS USED TO GENERATE THE TRIANGULAR WAVEFORM. THE NUMBER OF MULTIPLICATIONS, ADDITIONS AND ABSOLUTE VALUE OPERATIONS TO EVALUATE EACH POLYNOMIAL AND TO IMPLEMENT N DIFFERENTIATORS AND THE SCALING ARE ALSO GIVEN

Polynomial order	Polynomial function (triangular)	Input signal	MUL	ADD	ABS	MUL (scaling)	ADD (diff)	Total
$N = 1$	$1 - 2\|x\|$	$s(n)$	1	1	1	0	0	3
$N = 2$	$x\|x\| - x$	$s(n)$	1	1	1	1	1	5
$N = 3$	$x^3 - 3x/4$	$½ - \|s(n)\|$	3	2	1	1	2	9
$N = 4$	$x^3\|x\| - 2x^2 + x$	$s(n)$	3	2	1	1	3	10
$N = 5$	$x^5 - 5x^3/2 + 25x/16$	$½ - \|s(n)\|$	4	3	1	1	4	13
$N = 6$	$x^5\|x\| - 3x^5 + 5x^3 - 3x$	$s(n)$	7	3	1	1	5	16

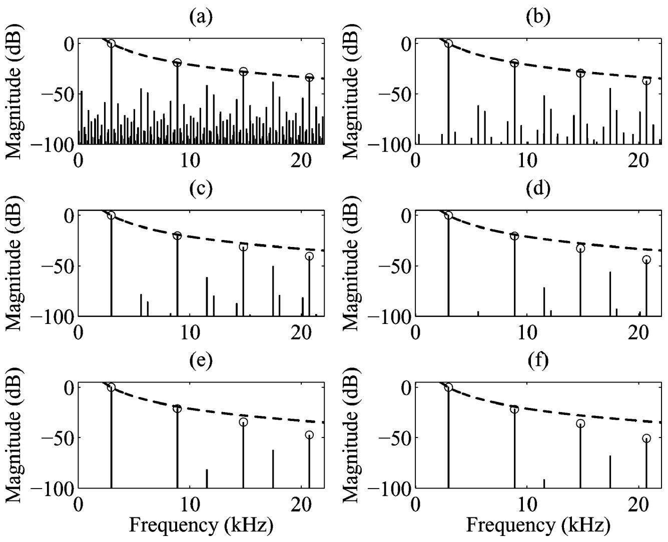


Fig. 10. Spectra of the triangular waveforms shown in Fig. 9. (a) The trivial triangular waveform and (b) the second-order, (c) the third-order, (d) the fourth-order, (e) the fifth-order, and (f) the sixth-order approximations. The parameters used in the spectral analysis are the same as in Fig. 2. The dashed line shows the spectral roll-off for the ideal triangular signal, which is about −12 dB/octave.

zero. We then set the derivative $2x^2 + a_1$ to zero at $x = 0.5$. This yields $(3/4) + a_1 = 0$, and the cubic polynomial becomes

$$f_3(x) = x^3 - \frac{3}{4}x. \quad (29)$$

This polynomial function also requires a special input signal, because it only works correctly between -0.5 and 0.5. Therefore, the input signal must repeat the same signal values in the first and the fourth quarter of the period. An appropriate input signal can be obtained by scaling (18) to have only values between -0.5 and 0.5:

$$\hat{s}_t(n) = \frac{s_t(n)}{2} = \frac{1}{2} - |s(n)|. \quad (30)$$

The block diagram of the third-order triangular algorithm, including the input signal generation, is presented in Fig. 7(c). The cubic waveform and the triangular signal approximation resulting after differencing twice are shown in Figs. 8(c) and 9(c), respectively.

For $N = 4$, the form is $f_4(x) = x^3|x| + a_3x^3 + a_2x|x| + a_1x$. From its second derivative on both the positive and negative side of zero, we see that $a_2 = 0$. We the first derivative $4x^3 + 3a_3x^2 + a_1$ to zero at $x = 0.5$. This yields $a_1 = 1$. We can then solve a_3 by setting the third derivative to zero at $x = 0.5$. We get $a_1 = -2$. The fourth-order polynomial is

$$f_4(x) = x^3|x| - 2x^2 + x. \quad (31)$$

The fourth-order waveform and its three times differenced version are presented in Figs. 8(d) and 9(d), respectively.

For the case $N = 5$ and for any larger N, we can similarly differentiate the general polynomial function, differentiate it multiple times and check the value on each side of points $x = 0$ and $x = 0.5$ to solve the coefficients. We skip the lengthy derivations of cases $N = 5$ and $N = 6$. The fifth- and sixth-order polynomial waveforms and the four and five times differenced waveforms are shows in Fig. 9(e), (f), 10(e), and (f), respectively.

The sample values of DPW triangular waveforms can be maintained between -1 and 1 by multiplying the signals with $2c$, that is, just two times larger scaling factors than the DPW sawtooth signals discussed previously.

We have gathered the polynomial functions for the cases $N =$ 1 to 6 in Table III. The third column in Table III shows the input signal for each case. Notice that when $N = 3$ and $N = 5$, a trivial triangular input signal must be used. Comparison against Table I reveals that the triangular algorithms generally need a few more operations per sample than the sawtooth algorithms. However, since the spectral roll-off rate of the triangular waveforms is steeper, it is expected that one of the low-order differentiated polynomial algorithms will be sufficient in practice.

Fig. 10 shows the spectra of the triangular waveforms of Fig. 9. It can be seen that the signals obtained by differencing high-order polynomial signals suffer from less aliasing than the trivial triangular signal [Fig. 10(a)]. The quality improves as the polynomial order is increased. When $N = 3$ or larger, the aliasing components below the first harmonic are suppressed more than 100 dB in this case. Notice also that in Fig. 10 the level of upper harmonics is always below the ideal spectral envelope that decays -12 db per octave. This is a consequence of the simple first-order difference used.

B. Rectangular Wave

The rectangular waveform with a 50% duty cycle, i.e., the square waveform, can be produced easily by differentiating a triangular signal. For the first-order case, this leads to the algorithm depicted in Fig. 11. The bipolar modulo counter signal is

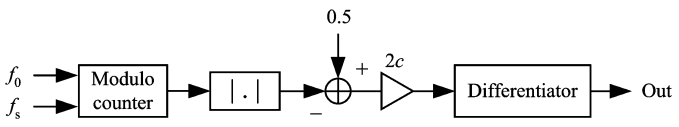


Fig. 11. First-order rectangular pulse generator based on differentiating the trivial triangular waveform.

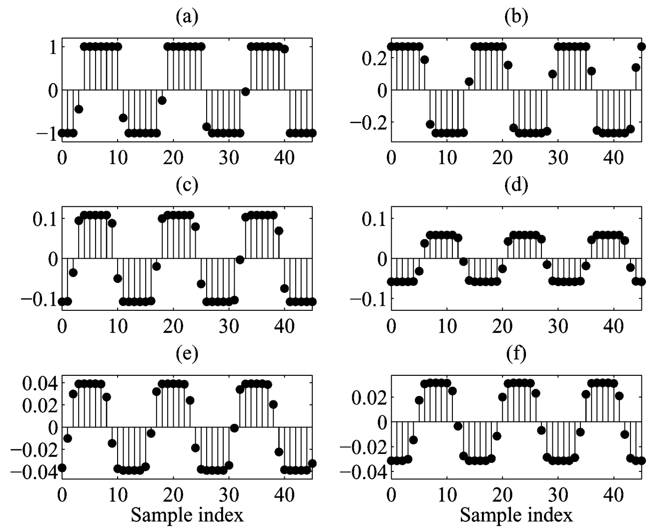


Fig. 12. Alias-reduced square waveforms generated by differencing the triangular signals Fig. 8. (a) The first-order, (b) the second-order, (c) the third-order, (d) the fourth-order, (e) the fifth-order, and (f) the sixth-order square waveform approximations.

full-wave rectified, inverted, shifted, scaled, and finally differenced. An example of the waveform and its spectrum are shown in Fig. 12(a) and Fig. 13(a), respectively. This waveform has a transition of one sample at each edge, so this is an improved version of the trivial square waveform.

Higher-order approximations for the square signal are obtained by differencing the higher-order triangular signals one more time. Fig. 12 shows the differenced versions of signals in Fig. 9. Notice how the transition region becomes smoother when N is increased. The corresponding spectra are shown in Fig. 13. In comparison to Fig. 10, the aliasing has been more suppressed at low frequencies in every case.

It has been shown in [18] that rectangular waveforms with a different duty cycle can be produced with the help of sawtooth signals [19], [18]. One can either subtract the output signals of two sawtooth oscillators with an appropriate phase shift (e.g., half a period for the 50% duty cycle) [18] or filter a sawtooth signal with a feedforward comb filter [19], which must use an interpolated delay line. Both methods facilitate also pulse-width modulation, which is a common technique in subtractive synthesis. When a high-order DPW sawtooth algorithm is used, it is more cost efficient to produce the rectangular pulses using an interpolated delay line method rather than running two DPW algorithms in parallel.

C. Bandlimited Impulse Trains

It is known that integrating an impulse train yields a sawtooth wave [10], [11]. Therefore, it is obvious that differentiation of the sawtooth waveform yields the impulse train. It is quite surprising, however, that when a sawtooth waveform obtained with the DPW method is differentiated, the result is an alias-suppressed pulse train, which can be obtained by inserting impulses into a B-spline filter. The theory of generating antialiasing impulse trains using interpolation and fractional-delay filter methods is discussed in a companion paper [20]. Differentiating a DPW square signal yields the bipolar BLIT sequence, in which there are two pulses in each period and every second of them is inverted.

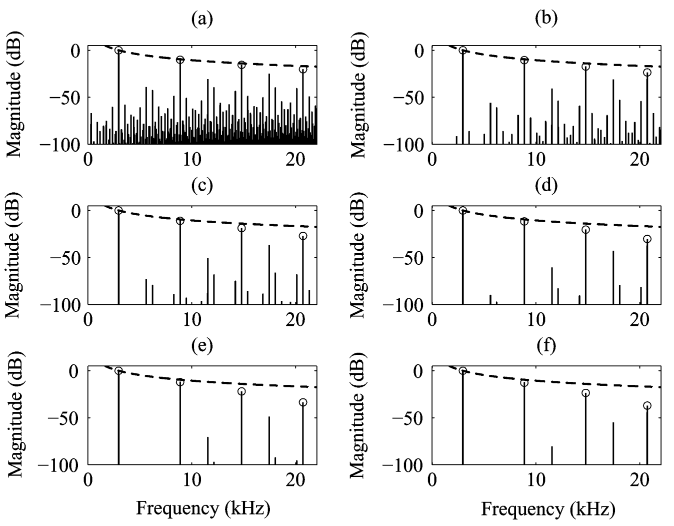


Fig. 13. Spectra of square wave approximations of Fig. 12. The dashed line shows the spectral roll-off for the ideal square signal, which is approximately −6 dB/octave.

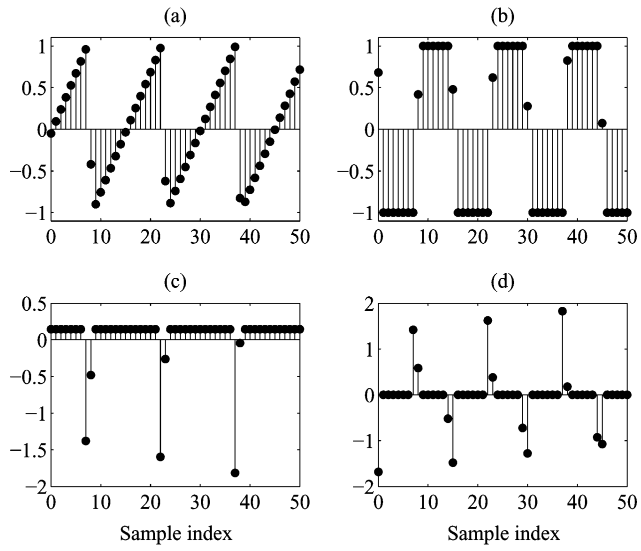


Fig. 14. (a) Second-order DPW sawtooth and (b) a first-order DPW square wave. (c) A unipolar and (d) a bipolar bandlimited impulse train obtained by differencing the signals (a) and (b), respectively.

Fig. 14 shows two examples, where a second-order DPW sawtooth [Fig. 14(a)] and the first-order DPW square wave [Fig. 14(b)] are differentiated with the first-order difference. Fig. 14(c) is the unipolar BLIT sequence obtained by differencing the sawtooth wave. Each pulse consists of two samples, which are also equivalent to the impulse response of the linear-interpolation FIR filter, which corresponds to the first-order B-spline interpolation. Additionally, this BLIT sequence contains a dc component. Fig. 14(d) shows the differenced square wave, which is a bipolar BLIT signal. Also these pulses can be obtained using the linear-interpolation filter.

While this observation may not be useful for generating BLITs, which can be synthesized directly without difference operations, it shows that the DPW methods and polynomial interpolation methods are closely related. This has not been known previously. The aliasing behavior of DPW and BLIT signals is also easily understood by knowing how they are related to each other.

IV. Evaluation and Comparison

In this section, we evaluate the audio quality of signals produced using the proposed DPW methods. It is of interest to understand how the quality is improved, when the polynomial order is increased. We also include in the evaluation the oversampled DPW algorithms and compare the results against a previous algorithm.

Previously, the sound quality of alias-suppressed signals has been assessed using various signal-to-noise (SNR) like measures, such as the power ratio of desired and undesired (i.e., aliased) components and its F-weighted version [16]. Välimäki and Huovilainen applied the noise-to-mask ratio (NMR), which is commonly used for evaluating the quality of perceptual audio codecs [2]. Timoney *et al.* introduced the use of Perceptual Evaluation of Audio Quality (PEAQ) for this purpose [12]. None of these measures is perfectly suited for evaluating the quality of harmonic tones with tonal disturbance, such as the classical waveforms with aliasing. We believe that a method that would account for the masking of the aliasing components by the partials would be the ideal method. This way it would be possible to decide case by case whether a synthesis method yields a perceptually alias-free output at a given fundamental frequency. We attempt to develop and use such method in this work.

We compute the FFT spectrum of each tone from a 1-s-long signal segment using a Hamming window, which facilitates the scaling of the signal power. We assume that the signals are played at the 96 dB SPL (sound pressure level). The signals are scaled so that their power is equivalent to the power of a sine wave played at 96 dB SPL.

Since the absolute signal level is known after scaling, we can use a hearing threshold function [21] to find which spectral components are inaudible. Additionally, we use the following asymmetric spreading function model for the masking effect of each partial on the dB scale [22]

$$B(\Delta z_{\mathrm{b}}) = L_{\mathrm{M}} + \cdots [-27 + 0.37 \max\{0, L_{\mathrm{M}} - 40\}\theta(\Delta z_{\mathrm{b}})]\,|\Delta z_{\mathrm{b}}| \quad (32)$$

where z_b is frequency in Bark units [17], L_M is the level of the masker (in dB), Δz_b is the frequency difference of the masker and the maskee, and $\theta(x)$ is a step function that is equal to zero for $x < 0$ and 1 otherwise. A masking curve model of (32) is allocated for each partial of the ideal waveform and is shifted down by 10 dB from the partials level. We combine the masking curves of each partial with the hearing threshold function using the max operation in the frequency domain.

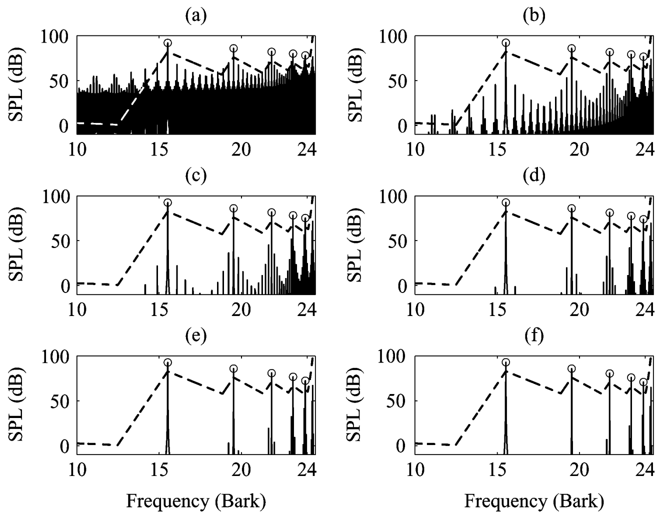


Fig. 15. Spectra (solid line) and masking curves (dashed line) of the DPW sawtooth signals of Fig. 3.

Fig. 15 shows examples of the spectra of sawtooth signals produced using the trivial method and the DPW methods of orders 2 to 6 with their overall masking curves (dashed lines). The dashed line indicates the overall masking curve for each signal. It is seen that at this fairly high fundamental frequency, many aliased components are above the masking curve in Fig. 15(a) and (b), but in all other cases only five harmonic components appear above the curve. This implies that aliasing should be audible in the trivial sawtooth [Fig. 15(a)] and in the sawtooth generated with the DPW with $N = 2$ [Fig. 15(b)]. For higher polynomial orders, the aliasing is completely hidden by the auditory masking phenomenon or by the disability of the human hearing to sense very quiet sounds.

Using this approach, we evaluated the sawtooth signals generated with the DPW methods of orders 1 to 6 for each fundamental frequency of the tempered scale ($\mathrm{A4} = 440$ Hz). We checked the highest fundamental frequency for which all aliased components remained below the masking curve. The results are presented in Table IV, second column. It can be noticed that increasing order N improves the alias-suppression considerably in every case. We repeated the same for the two times oversampled DPW algorithm (with a two-tap averager as the decimation filter), see Table IV, third column. The oversampling improves the performance in most cases, but, as discussed in Section II-E, it cannot be recommended, since it also doubles the computational burden.

For comparison, according to our evaluation, a sawtooth signal produced using the method proposed by Lane *et al.* [14] has the highest alias-free frequency of 600.0 Hz—the same result as with the second-order DPW method. (We used an optimized version of Lane's method in which the low-pass filter is opted out.) However, Lane's method has been estimated to have a larger computational cost than the second-order DPW method [16].

The highest tone of the piano, a C8, with the fundamental frequency of 4186 Hz, is often considered the highest pitch used in music. Thus, according to the results of Table I, the fourth-order DPW method provides a sufficient alias-suppression over

TABLE IV
HIGHEST FUNDAMENTAL FREQUENCY THAT IS PERCEPTUALLY ALIAS-FREE FOR THE DIFFERENTIATED POLYNOMIAL WAVEFORM METHODS OF ORDER 1 TO 6. THE RESULTS APPLY TO SAWTOOTH SIGNALS GENERATED AT 44.1 kHz AND AT 88.2 kHz (OVERSAMPLED)

Polynomial order	**Sawtooth**	**Sawtooth, oversampled**
$N=1$	N/A	N/A
$N=2$	600 Hz	1019 Hz
$N=3$	2037 Hz	3901 Hz
$N=4$	4593 Hz	6691 Hz
$N=5$	7851 Hz	9614 Hz
$N=6$	12221 Hz	12435 Hz

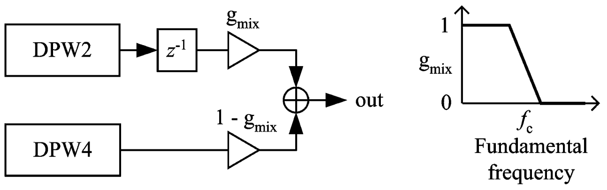


Fig. 16. Proposed implementation structure in which tones with a low fundamental frequency (below f_c) are produced with a low-order DPW algorithm (e.g., $N = 2$) and high fundamental frequencies are produced with a high-order DPW method (e.g., $N = 4$). A unit delay is needed to synchronize the signals.

the normally used range of fundamental frequencies, when the sample rate is 44.1 kHz. It would be superfluous to use the fifth or sixth-order DPW methods in that case.

A practical implementation structure for DPW sawtooth synthesis is presented in Fig. 16, where low-frequency tones are produced by a second-order DPW method and high frequencies are produced using a fourth-order method. The crossover frequency f_c may be about 500 Hz. Using a combination of two DPW algorithms it is possible to avoid a very large scale factor at low fundamental frequencies but still maintain a good audio quality at high fundamental frequencies.

V. CONCLUSION

This paper introduced novel classes of polynomial waveforms that can be differentiated one or more times to obtain an alias-suppressed sawtooth or triangular signal. The proposed differentiated polynomial waveform method extends the previous differentiated parabolic waveform method to higher orders than two. Square waves can be synthesized by applying the first-order difference operation to the triangular waveforms. Additionally, it was observed that the DPW methods are closely related to polynomial interpolation techniques, because differencing a DPW sawtooth or square signal yields a unipolar or a bipolar pulse train, respectively, consisting of impulse responses of low-order interpolation FIR filters.

Perceptual evaluation of DPW sawtooth signals shows that aliasing may be audible in the previously introduced second-order DPW method at fundamental frequencies above 600 Hz, when the sampling rate is 44 100 Hz. The fourth-order DPW method is alias-free up to about 4.6 kHz and thus permits the synthesis of practically all fundamental frequencies commonly used in music.

ACKNOWLEDGMENT

The first author would like to thank A. Huovilainen for helpful discussions. The second author would like to thank Prof. M. Bosi for helpful discussions on auditory masking.

REFERENCES

[1] H. Chamberlin, *Musical Applications of Microprocessors*, 2nd ed. New York: Hayden, 1985.
[2] V. Välimäki and A. Huovilainen, "Antialiasing oscillators in subtractive synthesis," *IEEE Signal Process. Mag.*, vol. 24, no. 2, pp. 116–125, Mar. 2007.
[3] M. Puckette, *The Theory and Technique of Electronic Music*. Singapore: World Scientific, 2007.
[4] F. R. Moore, *Elements of Computer Music*. Englewood Cliffs, NJ: Prentice-Hall, 1990, pp. 270–271.
[5] A. Chaudhary, "Bandlimited simulation of analog synthesizer modules by additive synthesis," in *Proc. Audio Eng. Soc. 105th Conv.*, San Francisco, CA, 1998, paper no. 4779.
[6] P. Kleczkowski, "Group additive synthesis," *Comput. Music J.*, vol. 13, no. 1, pp. 12–20, 1989.
[7] G. Winham and K. Steiglitz, "Input generators for digital sound synthesis," *J. Acoust. Soc. Amer.*, vol. 47, pp. 665–666, Feb. 1970, part 2.
[8] J. A. Moorer, "The synthesis of complex audio spectra by means of discrete summation formulae," *J. Audio Eng. Soc.*, vol. 24, no. 9, pp. 717–727, Nov. 1976.
[9] D. C. Massie, "Wavetable sampling synthesis," in *Applications of Digital Signal Processing to Audio and Acoustics*, M. Kahrs and K. Brandenburg, Eds. Norwell, MA: Kluwer, 1998, pp. 311–341.
[10] T. Stilson and J. Smith, "Alias-free digital synthesis of classic analog waveforms," in *Proc. Int. Computer Music Conf.*, Hong Kong, China, 1996, pp. 332–335.
[11] T. Stilson, "Efficiently-variable non-oversampling algorithms in virtual-analog music synthesis—A root-locus perspective" Ph.D. dissertation, Dept. of Elect. Eng., Stanford Univ., Stanford, CA, Jun. 2006 [Online]. Available: http://ccrma.stanford.edu/~stilti/papers/
[12] J. Timoney, V. Lazzarini, and T. Lysaght, "A modified FM synthesis approach to bandlimited signal generation," in *Proc. 11th Int. Conf. Digital Audio Effects*, Espoo, Finland, Sep. 2008, pp. 27–33.
[13] E. Brandt, "Hard sync without aliasing," in *Proc. Int. Comput. Music Conf.*, Havana, Cuba, 2001, pp. 365–368.
[14] J. Lane, D. Hoory, E. Martinez, and P. Wang, "Modeling analog synthesis with DSPs," *Comput. Music J.*, vol. 21, no. 4, pp. 23–41, 1997.
[15] J. Pekonen and V. Välimäki, "Filter-based alias reduction in classical waveform synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Las Vegas, NV, Mar. 2008, pp. 133–136.
[16] V. Välimäki, "Discrete-time synthesis of the sawtooth waveform with reduced aliasing," *IEEE Signal Process. Lett.*, vol. 12, no. 3, pp. 214–217, Mar. 2005.
[17] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*. New York: Springer-Verlag, 1990.
[18] V. Välimäki and A. Huovilainen, "Oscillator and filter algorithms for virtual analog synthesis," *Comput. Music J.*, vol. 30, no. 2, pp. 19–31, 2006, Summer.
[19] D. Lowenfels, "Virtual analog synthesis with a time-varying comb filter," in *Proc. Audio Eng. Soc. 115th Conv.*, New York, Oct. 2003, paper no. 5960.
[20] J. Nam, V. Välimäki, J. S. Abel, and J. O. Smith, "Efficient implementation of anti-aliasing oscillators using low-order fractional delay filters," *IEEE Trans. Audio, Speech, Lang. Process.*, 2009, submitted for publication.
[21] E. Terhardt, G. Stoll, and M. Seewann, "Algorithm for extraction of pitch and pitch salience from complex tonal signals," *J. Acoust. Soc. Amer.*, vol. 71, no. 3, pp. 679–688, Mar. 1982.
[22] M. Bosi and R. Goldberg, *Introduction to Digital Audio Coding and Standards*. Norwell, MA: Kluwer, 2003.

Vesa Välimäki (S'90–M'92–SM'99) received the M.Sc., the Licentiate of Science, and the Doctor of Science degrees in technology, all in electrical engineering, from the Helsinki University of Technology (TKK), Espoo, Finland, in 1992, 1994, and 1995, respectively. His doctoral dissertation dealt with fractional delay filters and physical modeling of musical instruments.

He was a Postdoctoral Research Fellow at the University of Westminster, London, U.K., in 1996. In 1997 to 2001, he was Senior Assistant (cf. Assistant Professor) at the TKK Laboratory of Acoustics and Audio Signal Processing, Espoo, Finland. From 1998 to 2001, he was on leave as a Postdoctoral Researcher under a grant from the Academy of Finland. In 2001 to 2002, he was Professor of signal processing at the Pori unit of the Tampere University of Technology, Pori, Finland. Since 2002, he has been Professor of audio signal processing at TKK. He was appointed Docent (cf. Adjunct Professor) in signal processing at the Pori unit of the Tampere University of Technology in 2003. In 2006–2007, he was the Head of the TKK Laboratory of Acoustics and Audio Signal Processing. In 2008 to 2009, he was on sabbatical and spent several months as a Visiting Scholar at the Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, Stanford, CA. His research interests include sound synthesis, digital filters, and acoustics of musical instruments.

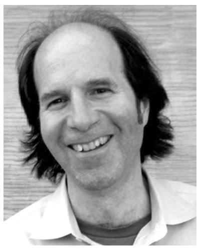
Prof. Välimäki is a member of the Audio Engineering Society, the Acoustical Society of Finland, and the Finnish Musicological Society. He was President of the Finnish Musicological Society from 2003 to 2005. In 2004, he was a Guest Editor of the special issue of the *EURASIP Journal on Applied Signal Processing* on model-based sound synthesis. In 2008, he was the chairman of DAFx-08, the 11th International Conference on Digital Audio Effects (Espoo, Finland). In 2000–2001, he was the secretary of the IEEE Finland Section. From 2005 to 2009, he was an Associate Editor of the IEEE SIGNAL PROCESSING LETTERS. In 2007, he was a Guest Editor of the special issue of the IEEE *Signal Processing Magazine* on signal processing for sound synthesis. He is currently an Associate Editor of the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING and of the *Research Letters in Signal Processing*. He is a member of the Audio and Electroacoustics Technical Committee of the IEEE Signal Processing Society. He is the Lead Guest Editor of the special issue of the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING on virtual analog audio effects and musical instruments.

Juhan Nam (S'09) was born in Busan, South Korea, in 1976. He received the B.S. degree in electrical engineering from Seoul National University, Seoul, Korea, in 1998. He is currently pursuing the M.S. degree in electrical engineering and the Ph.D. degree at the Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, studying signal processing applied to audio and music applications.

He was with Young Chang (Kurzweil Music Systems) as a Software Engineer from 2001 to 2006, working on software and DSP algorithms for musical keyboards.

Julius O. Smith received the B.S.E.E. degree in control, circuits, and communication from Rice University, Houston, TX, in 1975 and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1978 and 1983, respectively. His Ph.D. research was devoted to improved methods for digital filter design and system identification applied to music and audio systems.

From 1975 to 1977, he worked in the Signal Processing Department at ESL, Sunnyvale, CA, on systems for digital communications. From 1982 to 1986, he was with the Adaptive Systems Department at Systems Control Technology, Palo Alto, CA, where he worked in the areas of adaptive filtering and spectral estimation. From 1986 to 1991, he was with NeXT Computer, Inc., responsible for sound, music, and signal processing software for the NeXT computer workstation. After NeXT, he became an Associate Professor at the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University, teaching courses and pursuing research related to signal processing techniques applied to music and audio systems. Continuing this work, he is currently a Professor of Music and Associate Professor of Electrical Engineering (by courtesy) at Stanford University.



Jonathan S. Abel received the S.B. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1982, where he studied device physics and signal processing, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1984 and 1989, respectively, focusing his research efforts on statistical signal processing with applications to passive sonar and GPS.

He is currently a Consulting Professor at the Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, where his research interests include audio and music applications of signal and array processing, parameter estimation, and acoustics. From 1999 to 2007, he was a Co-Founder and Chief Technology Officer of the Grammy Award-winning Universal Audio, Inc. He was a Researcher at NASA/Ames Research Center, exploring topics in room acoustics and spatial hearing on a grant through the San Jose State University Foundation. He was also chief scientist of Crystal River Engineering, Inc., where he developed their positional audio technology, and a Lecturer in the Department of Electrical Engineering at Yale University. As an industry consultant, he has worked with Apple, FDNY, LSI Logic, NRL, SAIC, and Sennheiser, on projects in professional audio, GPS, medical imaging, passive sonar, and fire department resource allocation.

Prof. Abel is a Fellow of the Audio Engineering Society. He won the 1982 Ernst A. Guillemin Thesis Award for the best undergraduate thesis in the Department of Electrical Engineering and Computer Science.