IMPROVED POLYNOMIAL TRANSITION REGIONS ALGORITHM FOR ALIAS-SUPPRESSED SIGNAL SYNTHESIS

Dániel Ambrits and Balázs Bank
Budapest University of Technology and Economics,
Dept. of Measurement and Information Systems,
H-1521 Budapest, Hungary
ambrits.daniel@gmail.com, bank@mit.bme.hu

ABSTRACT

One of the building blocks of virtual analog synthesizers is the oscillator algorithm producing simple geometric waveforms, such as saw or triangle. An important requirement for such a digital oscillator is that its spectrum is similar to that of the analog waveform, that is, the heavy aliasing that would result from a trivial modulo-counter based implementation is reduced. Until now, the computationally most efficient oscillator algorithm with reduced aliasing was the Polynomial Transition Regions (PTR) method. This paper shows that the efficiency can be increased even further by eliminating the phase offset of the PTR method. The new Efficient PTR (EPTR) algorithm produces the same output as the PTR method, while requires roughly 30% less operations, making it the most efficient alias-reduced oscillator algorithm up to date. In addition to presenting an EPTR sawtooth algorithm, the paper extends the differentiated parabolic wave (DPW) triangle algorithm to the case of asymmetric triangle waves, followed by an EPTR implementation. The new algorithm provides continuous transition between triangle and sawtooth signals, while still requires low computational power.

1. INTRODUCTION

Analog synthesizers produced in the 60s and 70s are still very popular among musicians for their characteristic timbre, and the sound of these classic synthesizers has become an inherent part of many modern musical genres. However, the original synthesizers are hard to find, expensive, and usually do not provide sufficient control (e.g., via MIDI) as required by today's musicians. Therefore, some companies provide modern analog synthesizers with digital control, but an even more cost-effective solution is to simulate the analog signal chain via digital signal processing. The first such synthesizer was the Clavia NordLead, which paved the way for virtual analog synthesis. For an excellent overview on related research, see [1].

In an analog synthesizer the signal flow starts with an oscillator generating geometric waveforms, such as square, sawtooth, triangle, sine, and sometimes a noise generator

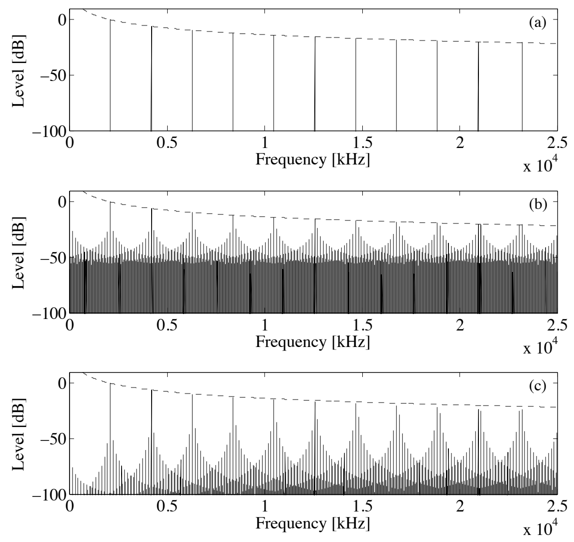

Figure 1. Spectrum of (a) the ideal, (b) the trivial and (c) the DPW sawtooth signals. The dashed line is the envelope of the ideal spectrum.

is also provided. Then this signal is fed to a filter that is controlled by envelope generators, low frequency oscillators (LFO), etc., and finally its gain is adjusted by an amplifier, again controlled by an envelope and LFO. This paper concentrates on the first part, that is, the digital modeling of the oscillator.

The trivial option for creating a digital replica of a geometric signal is to generate samples that correspond to the sampling of the analog waveform. In the case of the sawtooth signal, this results in a simple modulo counter, which can be implemented very effectively. However, as expected, this results in a heavy aliasing due to the non-bandlimited nature of the analog signal from which it originates [2]. This is displayed in Fig. 1 (b), together with the spectrum of the ideal (analog) sawtooth in Fig. 1 (a). A general remedy to the problem of aliasing is oversampling, that is, running the modulo counter at a significantly higher sampling rate, and then decimating. However, this leads to a considerable increase of computational complexity.

Therefore, special algorithms have been developed that reduce the aliased components while still keep the computational requirements low. Note that it is not required to eliminate aliasing completely, since aliased components below a certain level are inaudible due to masking effects [3].

The approaches include waveform generation based on a band-limited impulse train [4, 5] and band-limited step function [2, 6, 7], and the distortion and filtering of sine waves [8]. The simplest, yet still practically usable method is the differentiated parabolic wave (DPW) algorithm [9], that is based on the spectral tilt modification of the continuous-time signal before sampling. Later the higher-order extension of the method has also been presented [10], providing better alias suppression at the expense of larger complexity.

By noting that the DPW algorithm modifies only the samples around the discontinuity of the analog signal, a more efficient implementation is possible. This algorithm is called Polynomial Transition Regions (PTR), and is based on precomputing correction polynomials for the samples in the transition, while the linear regions of the signal are offset by a constant value [11].

This paper presents an even more efficient version of the PTR algorithm, which will be called EPTR throughout the paper. The method is based on the fact that the offset of DPW and PTR waveforms compared to the trivial (modulo-counter generated) waveform is due to a phase shift of the DPW and PTR signals. When this phase shift is removed, the linear regions of the waveform can be taken simply as the trivial waveform values, eliminating the need for an extra addition. For the sawtooth signal this leads to the reduction of the number of operations by around 30%.

By modulating the pulse width of a square wave, very interesting sonic variations can be created. Accordingly, many classic and virtual analog synthesizers offer this kind of PWM signal. A similarly interesting effect can be achieved by modulating the symmetry of triangle waves. This way the triangle signal can be continuously transformed into a sawtooth waveform. Two of the rare examples generating triangle waves with variable symmmetry are the Moog Little Phatty and Sub Phatty analog synthesizers [12]. This paper first extends the DPW algorithm for the case of asymmetric triangle waves, then provides a highly efficient implementation by the use of the new EPTR algorithm.

The rest of this paper is organized as follows. Section 2 reviews the DPW algorithm for the case of the sawtooth signal and provides an extension for the case of the asymmetric triangle wave. This is followed by the basic idea of the PTR method in Sec. 3, while Sec. 4 proposes the new EPTR algorithm for the saw and asymmetric triangle waves. Finally, Sec. 5 compares the computational complexity of the DPW, PTR and the EPTR algorithms.

2. DIFFERENTIATED POLYNOMIAL WAVEFORM ALGORITHM

First, let us consider the steps of generating an alias-suppressed signal with the Differentiated Polynomial Waveform (DPW) algorithm. In the Nth-order method the continuous signal is integrated $N-1$ times. This is equivalent to processing the sampled signal with an Nth-order polynomial waveshaper [10]. As a result, the spectrum of the sawtooth signal decreases by $6N$ dB per octave instead of 6 dB. This way the aliasing is significantly reduced. Then the

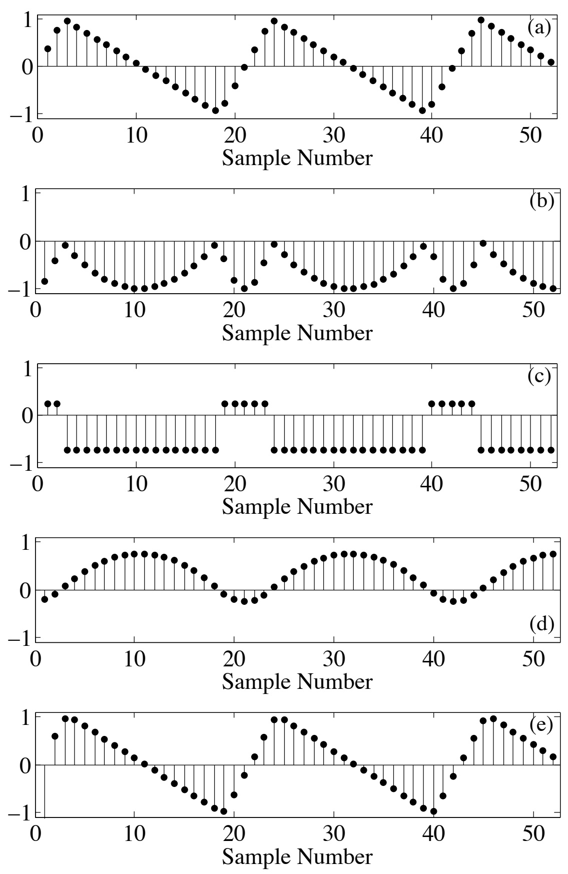

Figure 2. Generation of the triangle signal with variable symmetry. (a) The trivial signal is first processed with the x^2-1 function. (b) The result is then multiplied with (c) a scaled rectangular wave to get (d) the parabolic waveform. Finally differentiating and scaling produce (e) the desired alias-suppressed waveform.

signal is differentiated $N-1$ times in the discrete-time domain to restore its spectral tilt, which means filtering with the $(1-z^{-1})^{N-1}$ transfer function. Finally it is scaled to the desired amplitude [10].

For $N=2$ the integral of a piecewise linear signal is a piecewise parabolic ($x^2/2$) function. The trivial sawtooth signal corresponds to a periodic counter ranging from -1 to 1. The alias-suppressed sawtooth signal can be generated by squaring the signal, then differentiating the resulted piecewise polynomial waveform and finally scaling by a sufficient value [9].

While only symmetrical triangle wave is considered in the literature [10], it can be easily extended to asymmetric case. The method is explained by using Fig. 2. The gradient of the ascending region is $A>0$, then the gradient of the descending region is $B=-A/(A-1)<0$. Following the generation of the symmetric triangle signal for $N=2$ [10], the trivial waveform Fig. 2(a) is first processed with the x^2-1 function, giving Fig. 2(b). (In the general case, when the value of the peak is not 1, the waveshaper would be $x^2-p_{\text{peak}}^2$.) Then it is multiplied with a rectangular waveform with $1/A$ duty cycle so that the parabolic regions of the signal are alternately positive and negative. This rectangular wave is generated according to the counting direction of the trivial signal which, for the symmetric triangle holds the value 1 when the trivial waveform is ascending and -1 when it is descending. Note that by using this ± 1 rectangular wave the absolute values of the peaks are 1 in both regions but the width of these parabolic regions are not the same. Thus at the transition of two successive regions the gradients are different and this would cause jumps in the differentiated signal. This problem can be solved by scaling the regions with $1/|A|$ and $1/|B|$ factors so that the transition is smooth (see Fig. 2(c) and (d)). This step is the only difference between the symmetric and asymmetric case. Therefore the waveshapers are $(x^2-1)/A$ for the ascending region and $(x^2-1)/B$ for the descending region. Then the polynomial waveform is differentiated and multiplied by a scaling factor.

3. POLYNOMIAL TRANSITION REGIONS ALGORITHM

The signal generated with the Nth-order DPW algorithm differs from the trivial waveform by only $N-1$ samples per period. The differing samples are in the transition region, the linear sections only have an offset. This means unnecessary additional computation, since the integration and differentiation is computed even for the linear regions. The Polynomial Transition Regions algorithm was introduced to decrease the computation cost based on this observation. In the PTR method the sample values are derived in a closed form for each section, and the final signal is generated from the trivial signal using these general forms [11]. To show that the computational cost can be reduced even further, the linear section is discussed for $N=2$.

Two successive samples in the linear section are $p[n-1] = p_0 - 2AT$ and $p[n] = p_0$, where p_0 is the current value of the trivial signal generator, A is the gradient of the section. ($A=1$, when the signal increases from -1 to 1 during one period. If $A<0$, the signal decreases.) $T = f_0/f_s$, where f_0 is the fundamental frequency of the signal and f_s is the sampling frequency. For $N=2$ the waveshaper is x^2 according to the DPW algorithm, then the differentiation and scaling leads to:

$$y[n] = \frac{(p[n])^2 - (p[n-1])^2}{4AT} = p_0 - AT. \qquad (1)$$

For the linear section an addition operation is required for each sample. The value of the offset can be both positive and negative depending on whether the signal is ascending or descending, thus also a branch operation is needed. The $-AT$ offset represents a half sample delay from the trivial generator as seen in Fig. 3(a). This delay comes from the behavior of the discrete differentiation. We will see in the next section that a more efficient algorithm can be derived by eliminating this half sample delay, and so the need for the addition operation.

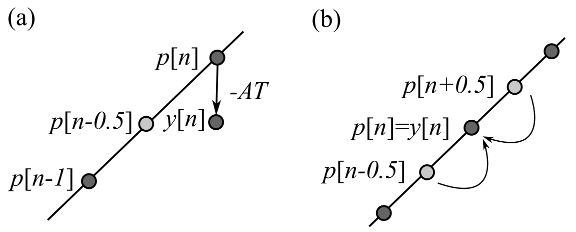

Figure 3. Discrete-time differentiation using (a) the trivial signal (dark dots) and (b) the waveform shifted with half sample (ligth dots).

4. EFFICIENT POLYNOMIAL TRANSITION REGION ALGORITHM

4.1 Eliminating the Half Sample Delay in the Linear Region

The simplest way to avoid the unnecessary computation in the linear region is using the trivial generator as the output. This is equivalent to using the differentiation on the adjacent samples which are at half sample distance from the origin as seen in Fig. 3(b). The $p[n-0.5]$ and $p[n+0.5]$ are the values of the continuous signal halfway between the samples.

$$\begin{aligned} y[n] &= \frac{(p[n+0.5])^2 - p[n-0.5])^2)}{4AT} = \\ &= \frac{(p_0+AT)^2 - (p_0-AT)^2}{4AT} = p_0. \end{aligned} \qquad (2)$$

In other words, the PTR algorithm is applied on a trivial signal which is with half sample in advance to the desired signal. Therefore also the samples in the transition region should be calculated from this shifted trivial waveform. However, this does not require that these samples are known during the wave generation, an explicit form of the correction can be calculated in advance.

The transition region is a one sampling time wide section. The shifted trivial signal causes that the samples to be corrected can be found before or after the break. So unlike in the PTR algorithm where this section was the [0,1] sample interval after the discontinuity, here it can be found in the [-0.5,0.5] interval with the transition in the centre. When we detect that the trivial signal generator is in this region, the position of the sample must be inspected and the correction must be applied according to the result. In the next section various types of transitions are derived for $N=2$.

4.2 Sawtooth

4.2.1 Derivation of the sawtooth wave generation

The continuous signal with A gradient jumps from the value $p_{\max}$ to $p_{\min}$ as can be seen in Fig. 4. When the sample of the discrete trivial waveform is $p[n] = p_0$ before the discontinuity, the next sample is $p[n+1] = p_0 + 2AT - (p_{\max} - p_{\min})$. These samples are processed by the x^2 waveshaper. The correction depends on whether the sample to be corrected is before or after the discontinuity of the continuous waveform.

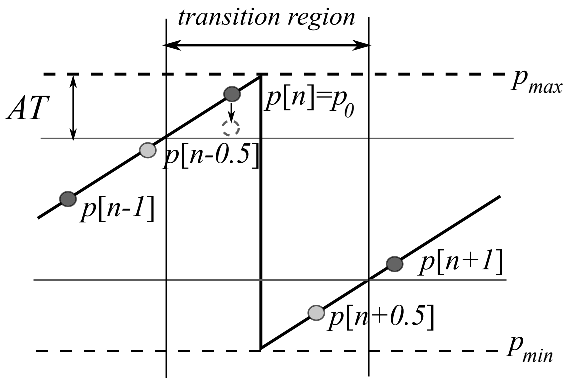

Figure 4. Correction of the sawtooth signal. During the derivation the half-sample delayed signal (light dots) is used instead of the trivial signal (dark dots). $p[n]$ is corrected to the desired value (dashed dot).

1. The sample is before the discontinuity ($p_0 > p_{\max} - AT$), as in Fig. 4

According to Section 4.1 the calculation must be applied using the adjacent samples at half sample distance from the output sample. The trivial sample has a value of $p[n] = p_0$, as seen in Fig. 4. The previous sample $p[n - 0.5] = p_0 - AT$, the next sample would be $p_0 + AT$ but it is higher than the maximum value, so $p[n + 0.5] = p_0 + AT - (p_{\max} - p_{\min})$. Applying the DPW algorithm leads to the desired output:

$$y_A[n] = \frac{(p[n + 0.5])^2 - (p[n - 0.5])^2}{4AT} = \\ = \frac{(p_0 + AT + p_{\min} - p_{\max})^2 - (p_0 - AT)^2}{4AT}. \quad (3)$$

For a sawtooth ranging from -1 to 1, we have $A = 1$, $p_{\max} = 1$, and $p_{\min} = -1$. In this special case the calculations lead to

$$y[n] = p_0 - \frac{p_0}{T} + \frac{1}{T} - 1. \quad (4)$$

2. The sample is after the discontinuity ($p_0 < p_{\min}+AT$)

The next sample is $p[n + 0.5] = p_0 + AT$. The previous sample can be found before the discontinuity (since $p_0 - AT < p_{\min}$), so it has a value of $p[n - 0.5] = p_0 - AT + (p_{\max} - p_{\min})$.

$$y_B[n] = \frac{(p[n + 0.5])^2 - (p[n - 0.5])^2}{4AT} \\ = \frac{(p_0 + AT)^2 - (p_0 - AT + p_{\max} - p_{\min})^2}{4AT}. \quad (5)$$

For the usual sawtooth signal $A = 1$, $p_{\max} = 1$ and $p_{\min} = -1$. In this special case the calculations lead to

$$y[n] = p_0 - \frac{p_0}{T} - \frac{1}{T} + 1. \quad (6)$$

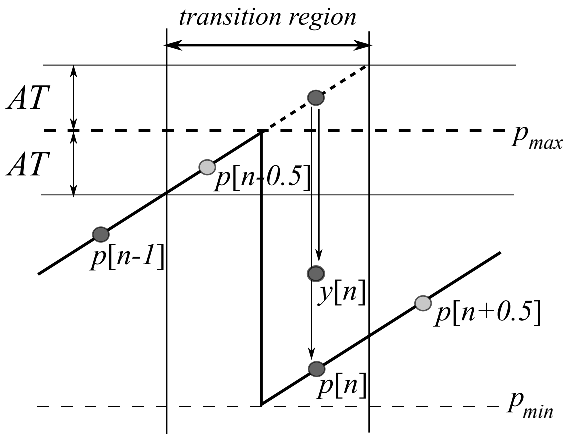

Figure 5. EPTR algorithm of the sawtooth signal.

4.2.2 The EPTR sawtooth wave algorithm

The PTR algorithm assumes that the trivial sawtooth signal is given, that is, the trivial signal generation and correcting the samples in the transition region are handled separately [11]. If we were doing the same, then two branch operations would be required: one for detecting the jump of the trivial signal and one for finding the transition region. However, a computationally simpler algorithm can be realized by merging the trivial signal generation and sample correction.

A further advantage of this choice is that since the trivial signal is generated by us, we are able to run the trivial counter even over $p_{\max}$ without forcing it to jump and apply the correction using that value. Indeed, substituting $p + 2$ into (6) leads to (4). Therefore, there is no need to check whether the sample to be corrected is before or after the discontinuity and the two cases can be handled in the same way. When the transition region is detected, the corrected output sample is computed, and then the trivial signal jumps, while the relative position of the sample compared to the transition is irrelevant. This is shown in Fig. 5.

The next source code shows how the algorithm can be programmed to generate a sawtooth signal ranging from -1 to 1.

```
p = p + 2*T;
if p > 1 - T
  y = correct(p);
  p = p - 2;
else
  y = p;
```

The function $\mathrm{correct}(p)$ is responsible for correcting the sample in the transition region. For the usual ± 1 sawtooth waveform we simply use (4):

$$\mathrm{correct}(p) = p - \frac{p}{T} + \frac{1}{T} - 1. \quad (7)$$

For the general case, see Table 1.

Note that the result would be the same if the trivial signal first jumped, then (6) was applied for correction. The resulting waveform is equivalent to the signal generated

region	$A, p_{\max}, p_{\min}$ (general case)	$A = 1, p_{\max} = 1, p_{\min} = -1$
linear region	p	p
correct(p)	$p + \frac{p_{\min}-p_{\max}}{2AT}p + \frac{p_{\min}-p_{\max}}{2} + \frac{(p_{\min}-p_{\max})^2}{4AT}$	$p - \frac{p}{T} + \frac{1}{T} - 1$

Table 1. Correction functions for the EPTR sawtooth algorithm.

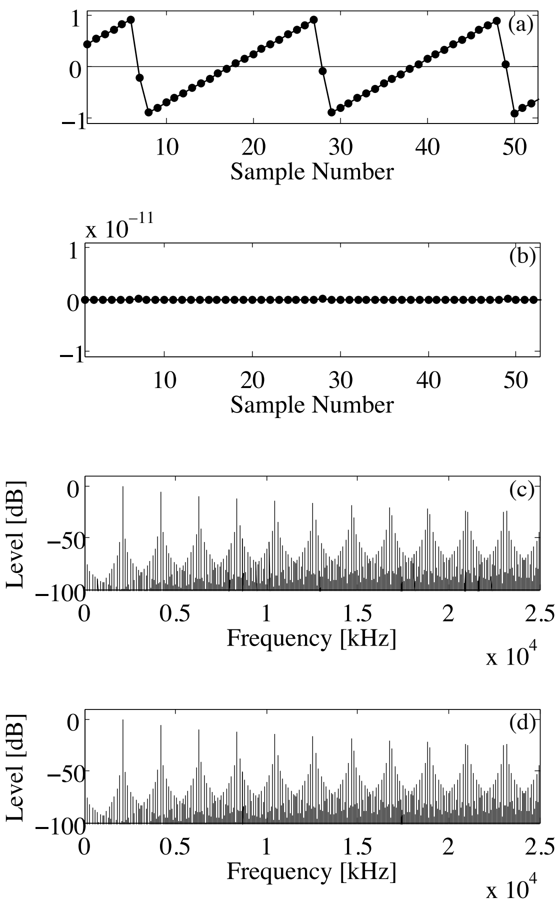

Figure 6. The DPW and EPTR sawtooth waveforms (a) and their difference (b), and the spectrum of the signal generated with the DPW (c) and EPTR (d) algorithms.

by the DPW and PTR algorithms as seen in Fig. 6. (The starting phase of the counter p was offset by a half sample for the EPTR algorithm so that the two curves match perfectly).

4.3 Triangle

4.3.1 Derivation of the asymmetric triangle wave generation

First let us consider computing the maximum peak of the triangle signal. Due to symmetry, the minimum peak can be calculated similarly. Around the maximum peak the trivial signal ascends to the value $p_{\max}$ with gradient A, then it descents with gradient B. When the trivial signal has a value of $p[n] = p_0$ before the peak, after its value is $p[n+1] = p_{\max} + 2BT - (p_{\max} - p_0 + 2AT) \cdot B/A$. The waveshapers are $(x^2 - p_{\max}^2)/A$ for the ascending and $(x^2 - p_{\max}^2)/B$ for the descending regions, as discussed

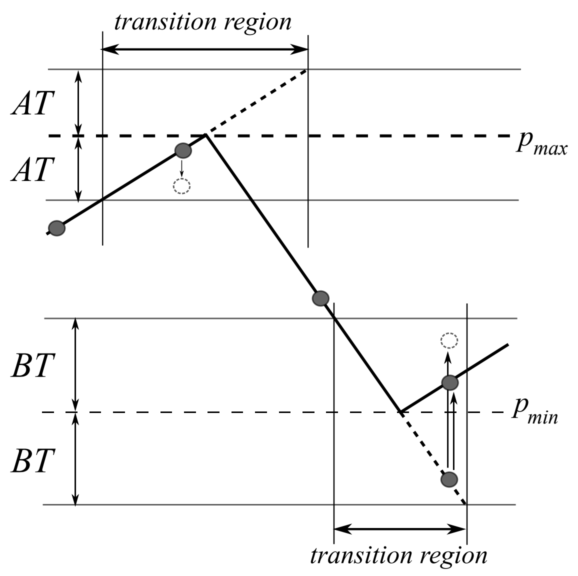

Figure 7. EPTR algorithm of the triangle signal.

in Section 2. Similarly to generating the sawtooth signal, the two adjacent values of the continuous signal are used which are at half sample distance from the output sample.

When the sample to be corrected is before the maximum peak ($p_0 > p_{\max} - AT$), the two adjacent values used are $p[n-0.5] = p_0 - AT$ and $p[n+0.5] = p_{\max} + BT - (p_{\max} - p_0) \cdot B/A$. The differentiation and scaling gives

$$\frac{((p[n+0.5])^2 - p_{\max}^2)/B - ((p[n-0.5])^2 - p_{\max}^2)/A}{4T} = \\ = a_2 p_0^2 + a_1 p_0 + a_0 \qquad (8)$$

which is included as the correctMax function in Table 2. Similarly, when the sample is after the peak ($p_0 > p_{\max} + BT$), the used value to the right is $p[n+0.5] = p_0 + BT$ and point to the left is still before the peak, so its value is $[p - 0.5] = p_{\max} + AT - (p_{\max} - p_0 + 2BT) \cdot A/B$. However, if we apply the same trick as in Sec. 4.2.2, that is, we are merging the trivial signal generation and sample correction, we are able to run the trivial counter p above $p_{\max}$, and one function (8) can handle both cases.

The derivation is similar for the minimum peak. After determining the adjacent values, the differentiation and scaling gives

$$\frac{((p[n+0.5])^2 - p_{\min}^2)/A - ((p[n-0.5])^2 - p_{\min}^2)/B}{4T} = \\ = b_2 p_0^2 + b_1 p_0 + b_0 \qquad (9)$$

which is included as the correctMin function in Table 2.

It is possible that a high gradient section fits between two samples. The condition for this case is that $|A| \leq$

Linear region	p
correctMax(p)	$a_2p^2 + a_1p + a_0$
correctMin(p)	$b_2p^2 + b_1p + b_0$

Coefficient	General	Special
a_2	$\frac{B-A}{4A^2T}$	$\frac{-1}{4(A-1)T}$
a_1	$\frac{AT(A+B)+p_{\max}(A-B)}{2A^2T}$	$\frac{2AT-4T+2}{4(A-1)T}$
a_0	$\frac{(B-A)(AT-p_{\max})^2}{4A^2T}$	$\frac{-(AT-1)^2}{4T(A-1)}$
b_2	$\frac{A-B}{4B^2T}$	$\frac{-1}{4(B+1)T}$
b_1	$\frac{BT(B+A)+p_{\min}(B-A)}{2B^2T}$	$\frac{2BT+4T-2}{4(B+1)T}$
b_0	$\frac{(A-B)(BT-p_{\min})^2}{4B^2T}$	$\frac{-(BT+1)^2}{4T(B+1)}$

Table 2. The polynomial correcting functions for the EPTR asymmetric traingle algorithm. In the special case $B = -A/(A-1), p_{\max} = 1, p_{\min} = -1$.

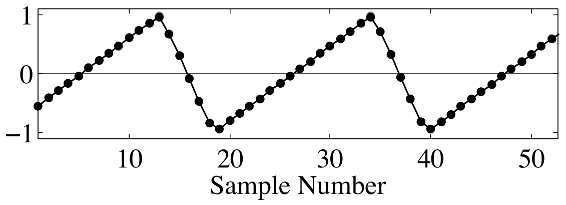

Figure 8. The PTR (gray) and the EPTR (black) asymmetric triangle waveforms.

$1/T = f_s/f_0$. This is not equivalent to the sawtooth signal in which the gradient is infinite, thus this case should be handled separately. When the trivial signal has a value of $p[n] = p_0$ before the maximum peak, after it the value is $p[n+1] = p_0 + p_{\min}(1 - A/B) - p_{\max}(1 - A/B)$. Since both of the adjacent samples are on linear sections with the same gradient, the calculation can be performed similarly to the sawtooth waveform.

4.3.2 The EPTR asymmetric triangle wave algorithm

Figure 7 explains the algorithm for generating an asymmetric triangle signal. Similarly to the sawtooth signal, the trivial waveform generation and the corrections are merged, thus checking whether the trivial generator is in the transition region is sufficient. The next code segment shows the implementation of the algorithm for a triangle waveform ranging from -1 to 1, with variable symmetry.

```
if dir == 1     // counting up?
  p = p + 2*A*T;
  if p > 1 − A*T
  //transition region?
    y = correctMax(p);
    p = 1 + (p − 1)*B/A;
    dir = −1;
  else  //linear region
    y = p;
else   // counting down
  p = p + 2*B*T;
  if p < −1 − B*T
 //transition region?
    y = correctMin(p);
    p = −1 + (p + 1)*A/B;
    dir = 1;
  else  //linear region
    y = p;
```

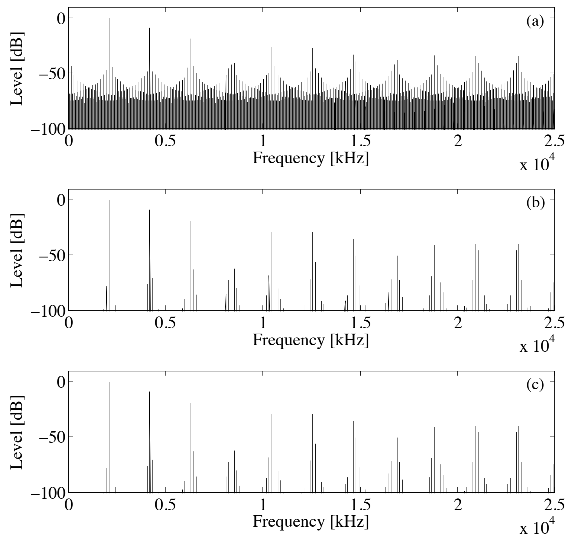

Figure 9. The spectrum of (a) the trivial, (b) the DPW and (c) the EPTR asymmetric triangle signal with 25% symmetry.

The correcting functions can be found in Table 2. First the counting direction must be determined, then the value of p is checked. When p is in the transition region, the corrected output sample is computed, the trivial counter is updated, and finally the counting direction is changed. If it is not in the transition region, the output simply equals the trivial counter $y = p$. The generated signal is equivalent to the DPW and PTR versions (see Fig. 8 and 9).

The previous code assumed that the values of $|A|$ and $|B|$ are not higher than f_s/f_0, so there is no region that fits between two samples. Although it is also possible to implement triangle waveforms with high gradient as discussed at the end of Sec. 4.3.1, it would result in a significantly more complicated algorithm. The allowed highest gradient case is close enough to the sawtooth waveform, therefore implementing the extra operations is not rewarding. Figure 10 shows the spectrum of a sawtooth with an infinitely sharp transition (a) and with a transition that lasts one sampling instant (b). The only drawback of limiting the gradient is a slight attenuation at high frequencies, on the other hand, the aliasing is reduced, since now we are correcting two samples around the transition. So the asymmetric triangle with a one sample-time transition can be safely used instead of the sawtooth signal. However, if there is still a need for the special case, the algorithm can be developed according to Sec. 4.3.1.

5. COMPARISON

The advantage of the EPTR method over PTR is the reduced computational load. For providing a fair comparison we merged the trivial signal generation and the correction

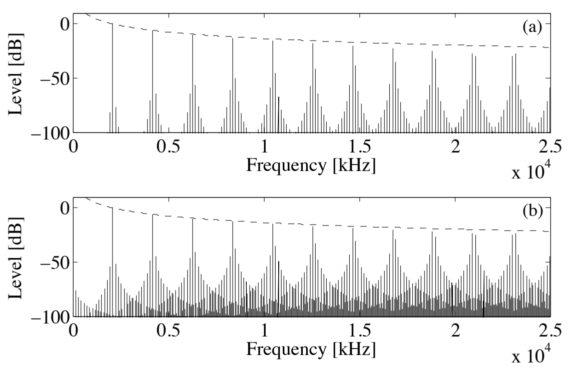

Figure 10. Spectrum of (a) the triangle signal with the highest allowed asymmetry and (b) the sawtooth signal. The dashed line is the envelope of the ideal spectrum.

Sawtooth	Add	Mul	Branch	Total
DPW	2	2	0	4
PTR	2	T	1	$3+T$
EPTR	$1+T$	T	1	$2+2T$
Triangle	Add	Mul	Branch	Total
DPW	3	3	0	6
PTR	$2+2T$	$6T$	2	$4+8T$
EPTR	$1+2T$	$6T$	2	$3+8T$

Table 3. Computational load of DPW, PTR and EPTR for sawtooth and triangle signals ($T = f_0/f_s$).

also for the PTR algorithm, although [11] and the adherent source codes [13] were handling them separately. Table 3 compares the two algorithms in operations per sample while generating sawtooth and asymmetric triangle waveforms.

The PTR algorithm uses an addition operation to increment the trivial counter. Then with a branch operation it decides whether the current sample is in the linear or the transition region. Finally, in the linear region an addition for the offset is applied and in the transition region a multiplication and an addition is necessary. When producing a sawtooth waveform with the EPTR algorithm, using the shifted trivial signal eliminates the addition operation in the linear region. Similarly, only the addition operation of the trivial counter is needed in the linear region during the asymmetric triangle signal generation.

The resulting waveforms generated with the two algorithms have the same spectrum, as we have seen in Sec. 4.

6. CONCLUSIONS

This paper has proposed a new version of the PTR algorithm. The Efficient Polynomial Transition Regions Algorithm requires around 30% lower number of operations compared to the PTR algorithm, while results in exactly the same waveform as that of the DPW and PTR algorithms. Thus, it is the most efficient alias-reduced algorithm up to date, making it an ideal choice for systems with low computational power requirements. In addition, the paper has extended the DPW algorithm for generating triangle waves with variable symmetry, and its EPTR implementation was also presented, allowing continuous transition between symmetric triangle and sawtooth signals. Future research includes the extension of the algorithm to higher orders, and to arbitrary waveforms composed of line segments (e.g., trapezoidal waves).

Acknowledgments

The work of Balázs Bank has been supported by the Bolyai Scholarship of the Hungarian Academy of Sciences.

The authors are thankful to Prof. Vesa Välimäki for his valuable comments.

7. REFERENCES

[1] J. Pekonen and V. Välimäki, "The brief history of virtual analog synthesis," in *Proc. 6th Forum Acusticum.* Aalborg, Denmark: European Acoustics Association, June 2011, pp. 461–466.

[2] V. Välimäki and A. Huovilainen, "Antialiasing oscillators in subtractive synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 116–125, March 2007.

[3] H.-M. Lehtonen, J. Pekonen, and V. Välimäki, "Audibility of aliasing distortion in sawtooth signals and its implications for oscillator algorithm design," *Journal of the Acoustical Society of America*, vol. 132, no. 4, pp. 2721–2733, October 2012.

[4] T. Stilson and J. Smith, "Alias-free digital synthesis of classic analog waveforms," in *in Proc. International Computer Music Conference*, Hong Kong, August 1996, pp. 332–335.

[5] S. Tassart, "Band-limited impulse train generation using sampled infinite impulse responses of analog filters," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 3, pp. 488–497, March 2013.

[6] E. Brandt, "Hard sync without aliasing," in *Proc. International Computer Music Conference*, Havana, Cuba, September 2001, pp. 365–368.

[7] V. Välimäki, J. Pekonen, and J. Nam, "Perceptually informed synthesis of bandlimited classical waveforms using integrated polynomial interpolation," *Journal of the Acoustical Society of America*, vol. 131, no. 1, pp. 974–986, January 2012.

[8] J. Lane, D. Hoory, E. Martinez, and P. Wang, "Modeling analog synthesis with DSPs," *Computer Music Journal*, vol. 21, no. 4, pp. 23–41, Winter 1997.

[9] V. Välimäki, "Discrete-time synthesis of the sawtooth waveform with reduced aliasing," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 214–217, March 2005.

[10] V. Välimäki, J. Nam, J. O. Smith, and J. S. Abel, "Alias-suppressed oscillators based on differentiated polynomial waveforms," *IEEE Transactions on Audio,*

Speech, and Language Processing, vol. 18, no. 4, pp. 786–798, May 2010.

[11] J. Kleimola and V. Välimäki, "Reducing aliasing from synthetic audio signals using polynomial transition regions," *IEEE Signal Processing Letters*, vol. 19, no. 2, pp. 67–70, February 2012.

[12] "Moog Slim Phatty user's manual," *Moog Music, Inc.*, 2010, URL: http://www.moogmusic.com/sites/default/files/slim_phatty_users_manual.pdf.

[13] J. Kleimola and V. Välimäki, "Polynomial Transition Regions [Online]," November 2011, URL: http://www.acoustics.hut.fi/go/spl-ptr.