ANTIALIASING PIECEWISE POLYNOMIAL WAVESHAPERS

Kurt James Werner [*]
Soundtoys, Inc.
Burlington, VT
kurt.james.werner@gmail.com

Emma Azelborn
Native Instruments, GmbH.
Boston, MA
emma.azelborn@native-instruments.com

ABSTRACT

Memoryless waveshapers are commonly used in audio signal processing. In discrete time, they suffer from well-known aliasing artifacts. We present a method for applying antiderivative antialising (ADAA), which mitigates aliasing, to any waveshaping function that can be represented as a piecewise polynomial. Specifically, we treat the special case of a piecewise linear waveshaper. Furthermore, we introduce a method for for replacing the sharp corners and jump discontinuities in any piecewise linear waveshaper with smoothed polynomial approximations, whose derivatives match the adjacent line segments up to a specified order. This piecewise polynomial can again be antialiased as a special case of the general piecewise polynomial. Especially when combined with light oversampling, these techniques are effective at reducing aliasing and the proposed method for rounding corners in piecewise linear waveshapers can also create more "realistic" analog-style waveshapers than standard piecewise linear functions.

1. INTRODUCTION

Memoryless nonlinearities are commonly used in audio signal processing algorithms, as waveshapers and wavefolders used in synthesizers [1–3], to model guitar amplifier distortion [4, 5], ring modulators [6], as models of electrical elements in virtual analog modeling (for instance, real electrical components like Shockley's diode model [7] or imaginary electrical components [8]), as gain computers in dynamic range controllers, as utility saturators to restrict a signal output to a certain range, etc. Given a continuous-time input signal $x(t)$, a memoryless nonlinearity $f(\cdot)$ produces a continuous-time output signal $y(t)$ by

$$y(t) = f\left(x(t)\right) \ . \tag{1}$$

In a digital signal processing context, with discrete time index n, a naive implementation is

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

This naive implementation suffers from one well-known artifact: aliasing distortion. The nonlinear function $f(\cdot)$ expands the bandwidth of the input signal $x[n]$, and if any frequency component arises that is above the Nyquist rate (half of the sampling rate: $f_s/2$), it will end up aliased down into the baseband. Although aliasing is sometimes introduced deliberately as a creative effect or to mimic a hardware device with audible aliasing distortion [9], it is usually considered detrimental and bad-sounding. As such, audio DSP designers typically use several types of approaches to mitigate aliasing distortion: 1. Oversampling, 2. Non-oversampled antialiasing (such as ADAA), and/or 3. Altering the waveshape to expand the signal bandwidth less. Oversampling is a classic and effective approach [4, 10], but can be quite costly if a large oversampling ratio is needed. Non-oversampled antialiasing is a class of techniques that include those specialized to waveform synthesis and for memoryless nonlinearities, including the recent and well-known Antiderivative Antialiasing (ADAA) technique [11–17]. In fact, we do not have to choose just one technique, but can often combine them. For instance, ADAA works best when combined with modest (at least $2\times$) oversampling.

If a designer is willing to allow small changes to the shape of their memoryless nonlinearity, the last option should be considered quite attractive. First of all, sacrificing a small bit of accuracy for increased sound quality is often a valid tradeoff. Second of all, in certain circumstances, smoothing out and/or rounding corners on a memoryless nonlinearity is often more "realistic," in the sense that waveshapers created with analog electronic circuits (including diodes, tubes, transistors) typically have very soft corners, whereas some of the classic techniques for creating digital memoryless nonlinearities, such as piecewise-linear representations (PWL) have sharp corners. For this reason, it can be useful to specify a PWL model augmented with smoothness controls for each corner. In fact, this approach is commonly used in digital dynamic range control systems, which typically offer "knee" parameters on their gain computers.

In this paper, we will present a method for applying ADAA to any piecewise-polynomial (PWP) waveshaping function. The special case of a piecewise linear (PWL) waveshaper is discussed, including how to convert from the more convenient breakpoint representation to the standard line-segment representation, for which ADAA can be applied in the identical fashion. Finally, we introduce a technique for replacing any rounded corners and jump discontinuities in a PWL waveshaper with rounded corners, making it a technique that combines *all three* of the aforementioned approaches. This method has a number of attractive features: 1. It's parameterized by piecewise line segments and finite jump discontinuities. Jump discontinuities should be attended to, since they appear commonly in memoryless nonlinearities, such as "dead-zone" and bitcrushers; 2. It has smooth corners made of finite-order polynomials; and 3. It has controllable finite width of corners and degree of smoothness at splice points.

This method combines the well-known ADAA approach with aspects of the canonical piecewise linear representations (CPLR) [18, 19] and the smoothed corners are based on a piecewise polynomial model of certain soft clipper curve, whose smoothness can be set to an arbitrary level, as introduced and discussed at length

[*] Early stages of this research were conducted during the 1st author's employment at iZotope, Inc., Boston, MA.

Table 1: *Polynomial coefficients for the piecewise polynomial from* (5) *and Fig. 1 and its antiderivative.*

j	$p_{j,0}$	$p_{j,1}$	$p_{j,2}$	$\overline{C}_j$	$\hat{C}_j$	C_j	$P_{j,1}$	$P_{j,2}$	$P_{j,3}$
0	3	2	0	0	0	$\frac{5}{3}$	3	2	0
1	0	0	1	0	$-\frac{5}{3}$	0	0	0	$\frac{1}{3}$
2	$\frac{1}{2}$	0	0	0	$-\frac{11}{6}$	$-\frac{1}{6}$	$\frac{1}{2}$	0	0

by Robert Bristow-Johnson, Olli Niemitalo, et al. [20–22], which stands in as a smoothed approximation to the sign function, and a related curve which is a smooth approximation of the absolute value function. The first curve is related to a family of curves known in the image processing field, including the simplest case, SmoothStep [23] and higher-order generalizations, and its lowest-order case is identical to a cubic soft clipper that is well-known in audio signal processing [24, 25].

The paper is structured as follows. §2 reviews piecewise polynomial (PWP) waveshaping functions and shows how to apply Antiderivative Antialiasing (ADAA) to them. §3 details a special case of the PWP waveshaper, the piecewise linear (PWL) waveshaper, again showing how ADAA can be performed. §4 shows how to round the corners of a PWL waveshaper using polynomial segments, which can be parameterized by a PWL representation, corner widths, and the order of smoothness enforced at each corner, again showing how ADAA can be applied. §5 presents several different ways to arrive at the same method for rounding the corners, based on ensuring smoothness up to a certain degree. §6 gives two brief case studies. §7 concludes.

2. PIECEWISE POLYNOMIALS WAVESHAPERS

A piecewise polynomial (PWP) with $J+1$ segments is given by

$$f(x) = \begin{cases} p_0(x), & x < x_1 \\ p_1(x), & x_1 \le x < x_2 \\ \cdots & \cdots \\ p_{J-1}(x), & x_{J-1} \le x < x_J \\ p_J(x), & x_J \le x \end{cases} \tag{3}$$

each of the $J+1$ polynomials has an order Φ_j and is defined as

$$p_j(x) = \sum_{\rho=0}^{\Phi_j} p_{j,\rho} x^\rho = p_{j,0} + p_{j,1}x + \cdots + p_{j,\Phi_j} x^{\Phi_j} \ . \tag{4}$$

An illustrative piecewise polynomial

$$f(x) = \begin{cases} p_0(x) = 2x + 3, & x < (x_1 = -1) \\ p_1(x) = x^2, & (x_1 = -1) \le x < (x_2 = 1) \\ p_2(x) = \frac{1}{2}, & (x_2 = 1) \le x \ . \end{cases} \tag{5}$$

is shown in the top of Fig. 1.[1]

[1]It is worth mentioning two efficient methods for evaluating polynomials: Horner's method and Estrin's scheme. Horner's method uses fewer operations but Estrin's scheme is more parallelizable, so the more efficient one will depend on implementation context. For higher-order polynomials, it may be more efficient to bake the waveshapes into lookup tables rather than evaluating the polynomials directly.

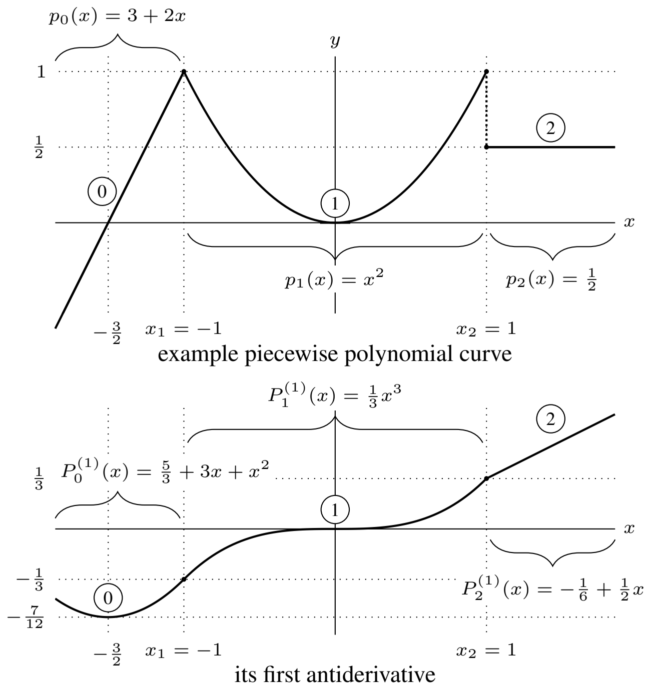


Figure 1: *An example of a piecewise polynomial curve (top) and its first antiderivative (bottom), with constants of integration chosen to enforce C^0 smoothness.*

2.1. Antialiasing Piecewise Polynomial Waveshapers

We can antialias any piecewise polynomial using the well-known Antiderivative Antialiasing (ADAA) method [11], which works by approximating the process of upsampling, applying a nonlinearity, and downsampling without actually doing anything at an upsampled rate. For a nonlinearity $f()$, ADAA produces expressions that use additions, multiplications, and divisions of various antiderivatives $F^{(N)}()$, where N indicates the antiderivative order. This results in potential divisions by zero (or near zero, which can cause numerical issues), which must be dealt with using "escape conditions": special cases where a small signal linearization that avoids the problematic division substitutes for the original expression.

The simplest form is first-order ADAA

$$y[n] = \begin{cases} \frac{F^{(1)}(u[n]) - F^{(1)}(u[n-1])}{u[n]-u[n-1]} \ , & \epsilon < |u[n] - u[n-1]| \\ f\left(\frac{u[n]+u[n+1]}{2}\right) \ , & \text{otherwise} \ , \end{cases} \tag{6}$$

where ϵ is a very small number (we use $\epsilon = 10^{-8}$) and the second case is used to escape numerical ill-conditioning issues. Again, $F^{(1)}()$ is the first antiderivative of $f()$. u is the input signal.[2]

The antiderivative of a polynomial of order Φ is another polynomial of order $\Phi + 1$

$$P_j^{(1)}(x) = C_j + \sum_{\rho=1}^{\Phi_j+1} \frac{p_{j,\rho-1}}{\rho} x^\rho \ . \tag{7}$$

[2]Note that we draw a distinction between the input to the waveshaping function or its antiderivate (which we always call x) and the input signal itself, which we call u. This is because the input to the waveshaping functions or antiderivatives (x) are sometimes filtered versions of the input signal itself (u), i.e., during the escape condition.

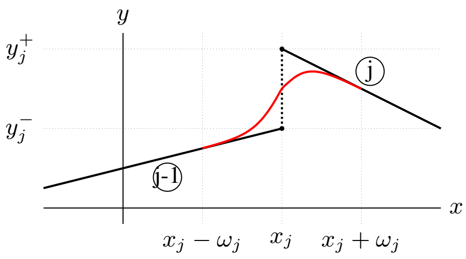


Figure 2: *Naming conventions around the single jth breakpoint, where ω_j is the "double-width" of the curved segment for the cases of a PWL waveshaper with rounded corners, as discussed in §4.*

However, we must be careful in setting the constants of integration C_j. In general, leaving them at zero will lead to large jump discontinuities in the antiderivative. Instead, we must set the constants of integration C_j so that the $P^{(1)}_{j-1}(x_j) = P^{(1)}_j(x_j)$, $j \in \{1, 2, \cdots, J\}$. We can also consider one last degree of freedom, the "global" vertical offset of the function. This can be set entirely arbitrarily, since it will cancel out immediately in the numerator of the antiderivative antialiasing equation (6). We choose it so that $F^{(1)}(0) = 0$, mainly for visual appeal on our plots.

To accomplish this, we start with a piecewise polynomial $\overline{F}^{(1)}(x)$ with all constants of integration zero: $\overline{C}_j = 0, j \in \{0, 1, \cdots, J\}$. Next, we form a piecewise polynomial $\hat{F}^{(1)}(x)$ by first setting $\hat{C}_0 = 0$. Then, for all other values of $j \in \{1, 2, \cdots, J\}$, we choose $\hat{C}_j$ so that the segments line up with C^0 smoothness:

$$\hat{C}_j = \hat{C}_{j-1} + \sum_{\rho=1}^{\Phi_{j-1}+1} \left(\frac{p_{j-1,\rho-1}}{\rho} x^\rho\right) - \sum_{\rho=1}^{\Phi_j+1} \left(\frac{p_{j,\rho-1}}{\rho} x^\rho\right). \quad (8)$$

This can be accomplished by finally setting

$$C_j = \hat{C}_j - \hat{F}^{(1)}(0),\ j \in 0, 1, 2, \cdots, J\,. \quad (9)$$

Which finally gives us our polynomial $F^{(1)}(x)$. All of the polynomial coefficients and intermediate steps of finding the constants of integration for this example PWP curve are shown in Tab. 1 and the resulting curve is shown in the bottom of Fig. 1.

2.2. An example

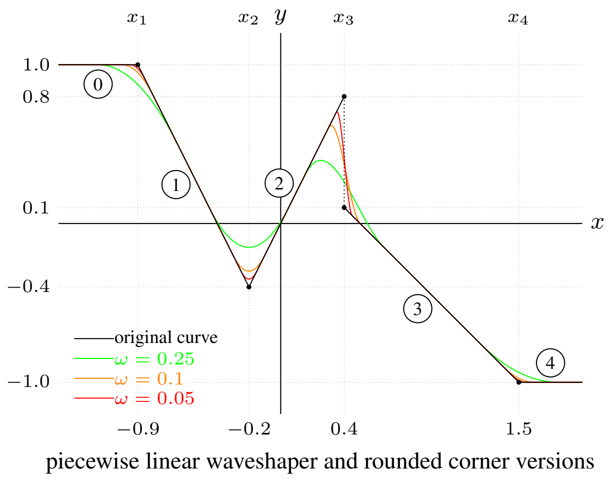
Starting with our example piecewise polynomial (5), we can antidifferentate them to form the intermediate polynomial

$$\overline{F}^{(1)} = \begin{cases} \overline{P}^{(1)}_0(x) = 3x + 2x^2, & x < -1 \\ \overline{P}^{(1)}_1(x) = \frac{1}{3}x^3, & -1 \le x < 1 \\ \overline{P}^{(1)}_2(x) = \frac{1}{2}x, & x \le 1 \end{cases}, \quad (10)$$

i.e., one where the unadjusted constants of integration are $\overline{C_0} = 0$, $\overline{C_1} = 0$, and $\overline{C_2} = 0$.

Using (8), we can form a piecewise polynomial of the antiderivative with no jump discontinuities, where $\hat{C}_0 = 0$, $\hat{C}_1 = -\frac{5}{3}$, and $\hat{C}_2 = -\frac{11}{6}$,

$$\hat{F}^{(1)} = \begin{cases} \hat{P}^{(1)}_0(x) = 3x + 2x^2, & x < -1 \\ \hat{P}^{(1)}_1(x) = -\frac{5}{3} + \frac{1}{3}x^3, & -1 \le x < 1 \\ \hat{P}^{(1)}_2(x) = -\frac{11}{6} + \frac{1}{2}x, & x \le 1 \end{cases}. \quad (11)$$



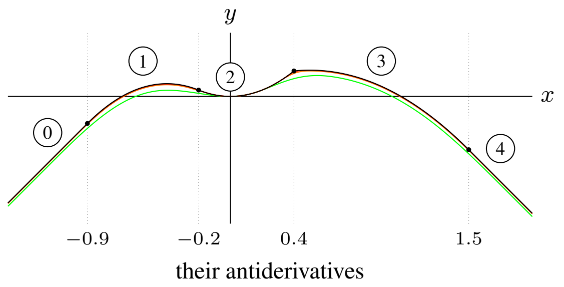


Figure 3: *An example of a piecewise linear curve (top) and its antiderivative (bottom), including three versions with spliced-in rounded corners of "double-width" ω, as discussed in §4. Here we have $\omega_j = \omega, \forall j$.*

Finally, by evaluating $\hat{F}^{(1)}(0) = -\frac{5}{3}$, we form the actual constants of integration $C_0 = \frac{5}{3}$, $C_1 = 0$, and $\hat{C}_2 = -\frac{1}{6}$ and arrive at the actual constants of integration

$$F^{(1)} = \begin{cases} P^{(1)}_0(x) = \frac{5}{3} + 3x + 2x^2, & x < -1 \\ P^{(1)}_1(x) = \frac{1}{3}x^3, & -1 \le x < 1 \\ P^{(1)}_2(x) = -\frac{1}{6} + \frac{1}{2}x, & x \le 1 \end{cases}. \quad (12)$$

This is now suitable for use in the first-order ADAA equation (6) or higher-order generalizations.

2.3. Higher-order antiderivatives

Higher-order [13, 26, 27] and other variations [12, 14, 16] on antiderivatives antialiasing algorithms exist, for instance 2nd or 3rd order ADAA, which involve expressions involving higher order antiderivatives such as $F^{(2)}(x)$, $F^{(3)}(x)$, etc., as well as more complex expressions and escape conditions. The process of defining $F^{(N)}(x)$ from $F^{(N-1)}(x)$ for any order N is identical to the process of forming $F^{(1)}(x)$ from $f(x)$, so needs not be discussed in detail here. Because the antiderivative of any polynomial is another polynomial of one higher order, we are guaranteed that we can analytically produce as many antiderivatives as are required. For other types of functions, analytic antiderivatives do not always exist (although they can always be approximated numerically).

Table 2: *Breakpoint locations and extremal slopes and derived slope and offsets for the example curve shown in Fig. 3. The y-values and σ value associated with jump discontinuities are shaded.*

j	x_j	y_j^-	y_j^+	b_j	m_j	β_j	μ_j	α_j	σ_j	$\hat{C}_j$	$\overline{C}_j$	C_j	$P_{j,1}$	$P_{j,2}$
0	——	——	——	1.0	0.0	——	——	——	——	0	0	0.73	1.0	0.0
1	−0.9	1.0	1.0	−0.8	−2.0	−0.8	−1.0	−1.0	0.0	0	−0.81	−0.08	−0.8	−1.0
2	−0.2	−0.4	−0.4	0.0	2.0	0.0	0.0	2.0	0.0	0	−0.73	0.0	0.0	1.0
3	0.4	0.8	0.1	0.5	−1.0	0.0	0.5	−1.5	−0.35	0	−0.69	0.04	0.5	−0.5
4	1.5	−1.0	−1.0	−1.0	0.0	0.5	−0.5	0.5	0.0	0	0.435	1.165	−1.0	0.0

3. PIECEWISE LINEAR WAVESHAPERS

A special case that often arises is the case of piecewise linear (PWL) waveshapers. A graphical representation of an example piecewise linear curve is shown in Fig. 3. These are in fact a special case of the previously discussed peicewise-polynomial waveshapers, where, again for J breakpoints x_j, $j \in 1, 2, \cdots, J$ and $J+1$ segments, each polynomial segment is just a line segment

$$p_j(x) = p_{j,0} + p_{j,1}x \tag{13}$$

To be more specific that it is a line we will call these $\ell(x)$ and to match the conventional notation for a line, we will use m for slope and b for offset. Now, the entire PWL function is given by

$$f(x) = \begin{cases} \ell_0(x), & x < x_1 \\ \ell_1(x), & x_1 \le x < x_2 \\ \cdots & \cdots \\ \ell_{J-1}(x), & x_{J-1} \le x < x_J \\ \ell_J(x), & x_J \le x \end{cases} \tag{14}$$

where each line segment is defined by

$$\ell_j(x) = b_j + m_j x_j, \quad j \in \{0, 1, 2, \cdots, J\}\,. \tag{15}$$

We require that the x_j, coordinates be ordered as

$$x_1 < x_2 < x_3 < \cdots < x_J\,. \tag{16}$$

It's often convenient, in defining a PWL function, to not deal with the slopes and offsets directly, but rather to specify the endpoints of each line segments. A piecewise linear function with J breakpoints and $J+1$ line segments (which may have jump discontinuities between them) can be fully specified by

1. the slope of the furthest-left line segments: m_0.
2. J breakpoints, $j \in \{1, 2, \cdots, J\}$. They are written in the form (x_j, y_j), with a single y coordinate, when there is no jump discontinuity. They are written in the form (x_j, y_j^-, y_j^+) when there is a jump discontinuity, where the superscript indicates which y coordinate is associated with the $(-)$ or right $(+)$ line segment. When there is no jump discontinuity, we will still need to refer to the y coordinate associated with each line segment, but in that case we simply have $y_j^- = y_j^+ = y_j$.
3. the slope of the furthest-right line segments: m_J.

Again, the x coordinates must be ordered by $x_1 < x_2 < x_3 < \cdots < x_J$, essentially meaning that none of the line segments may be vertical—instead, an instantaneous vertical jump should be represented as a jump discontinuity between two line segments—and also that the line segments are indexed in increasing order from left to right. Finally, all quantities should have finite values. These naming conventions are shown graphically in Fig. 2.

Given this, the slopes m and offsets b are determined from the breakpoints and extremal slopes by

$$m_j = \frac{y_{j+1}^- - y_j^+}{x_{j+1} - x_j}\,, \qquad j \in \{1, \cdots, J-1\} \tag{17}$$

$$b_0 = y_1^- - m_0 x_1 \tag{18}$$

$$b_j = y_j^- - m_{j-1}x_j = y_j^+ - m_j x_j,\ j \in \{1, \cdots, J-1\} \tag{19}$$

$$b_J \qquad = y_J^+ - m_J x_J\,. \tag{20}$$

and m_0 and m_J are already given. The resulting slopes and offsets for the example piecewise linear curve (Fig. 3) are tabulated alongside the original breakpoint locations in Tab. 2.

3.1. Antialiasing Piecewise Linear Waveshapers

Applying antialiasing to the PWL waveshaper is done identically to the more general piecewise polynomial. For this special case, the antiderivative of the PWL function is a piecewise-quadratic function, i.e., the order of each line segment in the PWL is $\Phi_j = 1$, so the order of the each segment of its antiderivative is $\Phi_j = 2$ and

$$P_j^{(1)}(x) = C_j + \frac{p_{j,0}}{1}x + \frac{p_{j,1}}{2}x^2 = C_j + \frac{b_j}{1}x + \frac{m_j}{2}x^2\,. \tag{21}$$

The constants of integration are set identically to the PWP case.

4. ROUNDED PIECEWISE LINEAR WAVESHAPERS

Now we will introduce a formulation that splices a rounded corner function $g_j(x)$, $j \in \{1, 2, \cdots, J\}$ between each line segment $\ell_{j-1}(x)$ and $\ell_j(x)$. Without yet saying much the exact shape of these rounded corners, the main quantity that they must be parameterized by is the double-width of each corner, denoted by $\omega_j > 0$, which may be different for each corner. These must obey

$$x_j + \omega_j < x_{j+1} - \omega_{j+1}, \quad j \in \{1, 2, \cdots, J-1\} \tag{22}$$

essentially guaranteeing that the corners do not overlap and that the line segments do not shrink to have empty domains.

This representation is given by

$$\widetilde{f}(x) = \begin{cases} \ell_0(x), & x < x_1 - \omega_1 \\ g_1(x), & x_1 - \omega_1 \le x < x_1 + \omega_1 \\ \ell_1(x), & x_1 + \omega_1 \le x < x_2 - \omega_2 \\ \cdots & \cdots \\ \ell_{J-1}(x), & x_{J-1} + \omega_{J-1} \le x < x_J - \omega_J \\ g_J(x), & x_J - \omega_J \le x < x_J + \omega_J \\ \ell_J(x), & x_J \le x \end{cases} \tag{23}$$

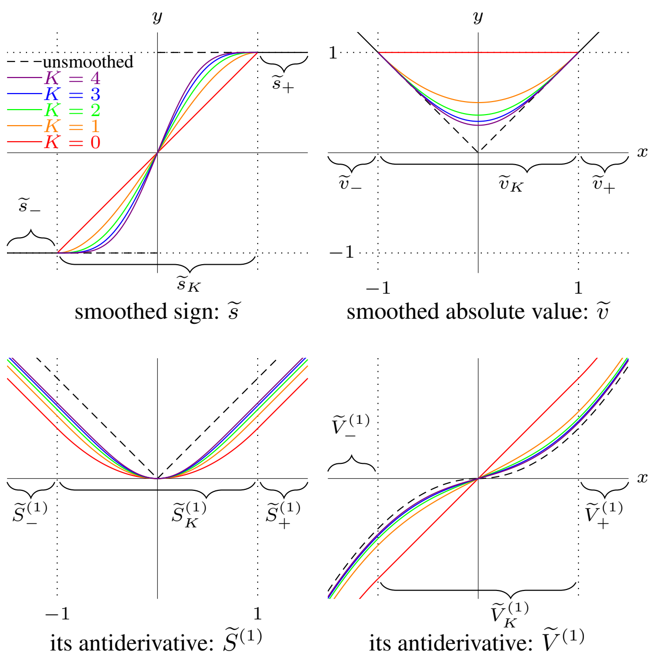


Figure 4: *Plots of the sign function* $\operatorname{sgn}(x)$ *and its smoothed variant* $\widetilde{s}_K(x)$ *(top left), the absolute value function* $|x|$ *and its smoothed variants* $\widetilde{v}_K(x)$ *(top right), as well as their antiderivatives* $\widetilde{S}_1^{(x)}$ *(bottom left) and* $\widetilde{V}_1^{(x)}$ *(bottom right), all for* $K \in \{0, 1, 2, 3, 4\}$.

where each line segment $\ell_j(x)$ is defined as in (15), and where, still without saying much about their shape, each rounded corner is given by an equation of the form [18]

$$g_j(x) = \beta_j + \mu_j x + \alpha_j \omega_j\, \widetilde{v}_j\left(\tfrac{x - x_j}{\omega_j}\right) + \sigma_j\, \widetilde{s}_j\left(\tfrac{x - x_j}{\omega_j}\right) \quad (24)$$

where

$$\beta_j = g_j(0) - \mu_j |x_j| = \begin{cases} y_j^- - m_j x_j , & x_j < 0 \\ \frac{y_j^+ + y_j^-}{2}, & x_j = 0 \\ y_{j-1}^+ - m_{j-1} x_j , & 0 < x_j \end{cases} \quad (25)$$

$$\mu_j = \frac{m_j + m_{j-1}}{2}, \qquad \alpha_j = \frac{m_j - m_{j-1}}{2} \quad (26)$$

$$\sigma_j = \frac{y_j^+ - y_j^-}{2}, \quad j \in \{1, 2, \ldots, J\}\,. \quad (27)$$

where each slope m_j is defined as in (17).

$\widetilde{v}(x)$ is a "smooth" approximation of the absolute value

$$\widetilde{v}(x) \approx |x| = \begin{cases} -x\,, & x \le 0 \\ x\,, & 0 \le x \end{cases} \quad (28)$$

and $\widetilde{s}(x)$ is a "smooth" approximation of the sign function

$$\widetilde{s}(x) \approx \operatorname{sign}(x) = \begin{cases} -1\,, & x < 0 \\ 0\,, & x = 0 \\ 1\,, & 0 < x \end{cases}. \quad (29)$$

In the case where $\widetilde{v}(x) := |x|$ and $\widetilde{s}(x) := \operatorname{sign}(x)$, the representation (24) would be identical to the standard PWL representation (with no rounded corners). The idea of decomposing a corner with a possible jump continuity into a constant, linear, scaled absolute value, and scaled sign function comes from the "canonical piecewise linear representation" literature [28]. The advantage of performing this decomposition is that we only need to deal with defining $\widetilde{v}(x)$ and $\widetilde{s}(x)$ once, rather than defining a unique new function for each corner, and can then shift and scale them appropriately using the α_j, σ_j, ω_j, and x_j parameters used in (24)–(27). The resulting constants for the example piecewise linear curve (Fig. 3) are tabulated in Tab. 2. Again, we have $\omega_j = \omega, \forall j$ for these curves, where Fig. 3 shows plots for three values of ω whereas the values in Tab. 2 are calculated for the case of $\omega = 0.25$.

5. DERIVATIONS FOR SMOOTHED CORNERS

We could imagine defining $\widetilde{v}(x)$ and $\widetilde{s}(x)$ in many ways. Here, we will propose a specific way of defining these functions based on polynomials that match the values and one or more derivatives of the linear functions they are spliced to.

Now we will give details on how to derive functions that are suitable for $\widetilde{v}(x)$ and $\widetilde{s}(x)$. The key concept here is that these functions are both piecewise polynomials, comprising three segments each, with the outer two segments as straight lines and the inner segment, defined on the open interval $x \in\,]-1, 1[$, is a polynomial that is designed to match the value and a certain number of derivatives of the outer segments at the splice points $x = \pm 1$.

The smoothed absolute value function is

$$\widetilde{v}(x) = \begin{cases} \widetilde{v}_-(x) = -x, & x \le -1 \\ \widetilde{v}_K(x), & -1 \le x \le 1 \\ \widetilde{v}_+(x) = x, & 1 \le x \end{cases} \quad (30)$$

and the smoothed sign function is

$$\widetilde{s}(x) = \begin{cases} \widetilde{s}_-(x) = -1, & x \le -1 \\ \widetilde{s}_K(x), & -1 \le x \le 1 \\ \widetilde{s}_+(x) = 1, & 1 \le x \end{cases}. \quad (31)$$

$\widetilde{v}(x)$ and $\widetilde{s}(x)$ with no subscripts refer to the entire smoothed absolute value and sign functions, whereas the subscripted versions $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$ refer to the inner polynomial segments. The subscript K refers to the degree of smoothness—Our goal will be to choose the polynomial coefficients so that $\widetilde{v}(x)$ and $\widetilde{s}(x)$ have a specified degree of differentiability C^K. Specifically, C^k smoothness means that all derivatives up to order k exist and are continuous. Since $\widetilde{v}_-(x)$, $\widetilde{v}_K(x)$, $\widetilde{v}_+(x)$, $\widetilde{s}_-(x)$, $\widetilde{s}_K(x)$, and $\widetilde{v}_+(x)$ (and all polynomials) all have C^∞ smoothness, this amounts choosing the $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$ coefficients so that their values and first K derivatives match those of $\widetilde{v}_\pm(x)$ and $\widetilde{s}_\pm(x)$ at $x = \pm 1$. For a given degree of smoothness K, these constraints are:

$$\widetilde{v}_K(\pm 1) = \widetilde{v}_\pm(\pm 1) \qquad = 1 \quad (32)$$

$$\frac{\mathrm{d}}{\mathrm{d}x}\widetilde{v}_K(\pm 1) = \frac{\mathrm{d}}{\mathrm{d}x}\widetilde{v}_\pm(\pm 1) \quad = \pm 1 \quad (33)$$

$$\frac{\mathrm{d}^k}{\mathrm{d}x^k}\widetilde{v}_K(\pm 1) = \frac{\mathrm{d}^k}{\mathrm{d}x^k}\widetilde{v}_\pm(\pm 1) = 0, \quad k \in \{2, 3, \cdots, K\} \quad (34)$$

$$\widetilde{s}_K(\pm 1) = \widetilde{s}_\pm(\pm 1) \qquad = \pm 1 \quad (35)$$

$$\frac{\mathrm{d}}{\mathrm{d}x}\widetilde{s}_K(\pm 1) = \frac{\mathrm{d}}{\mathrm{d}x}\widetilde{s}_\pm(\pm 1) \quad = 0 \quad (36)$$

$$\frac{\mathrm{d}^k}{\mathrm{d}x^k}\widetilde{s}_K(\pm 1) = \frac{\mathrm{d}^k}{\mathrm{d}x^k}\widetilde{s}_\pm(\pm 1) = 0, \quad k \in \{2, 3, \cdots, K\}\,. \quad (37)$$

Table 3: *Tabulated coefficients for smoothed signum $\widetilde{s}_K(x)$ and absolute value $\widetilde{v}_K(x)$ for the lowest 7 values of K.*

K	N	s_1	s_3	s_5	s_7	s_9	s_{11}	s_{13}
0	1	1						
1	3	1.5	−0.5					
2	5	1.875	−1.25	0.375				
3	7	2.1875	−2.1875	1.3125	−0.3125			
4	9	2.4609375	−3.28125	2.953125	−1.40625	0.2734375		
5	11	2.70703125	−4.51171875	5.4140625	−3.8671875	1.50390625	−0.24609375	
6	13	2.9326171875	−5.865234375	8.7978515625	−8.37890625	4.8876953125	−1.599609375	0.2255859375
K	N	v_0	v_2	v_4	v_6	v_8	v_{10}	v_{12}
0	0	1						
1	2	0.5	0.5					
2	4	0.375	0.75	−0.125				
3	6	0.3125	0.9375	−0.3125	0.0625			
4	8	0.2734375	1.09375	−0.546875	0.21875	−0.0390625		
5	10	0.24609375	1.23046875	−0.8203125	0.4921875	−0.17578125	0.02734375	
6	12	0.2255859375	1.353515625	−1.1279296875	0.90234375	−0.4833984375	0.150390625	−0.0205078125

The polynomials are defined as

$$\widetilde{v}_K(x) = \sum_{\rho=0,2,\cdots}^{\Phi_\text{e}} v_\rho x^\rho = v_0 + v_2 x^2 + \cdots + v_{\Phi_\text{e}} x^{\Phi_\text{e}} \tag{38}$$

$$\widetilde{s}_K(x) = \sum_{\rho=1,3,\cdots}^{\Phi_\text{o}} s_\rho x^\rho = s_1 x + s_3 x^3 + \cdots + s_{\Phi_\text{o}} x^{\Phi_\text{o}} \,. \tag{39}$$

Because $|x|$ is an even function, $\widetilde{v}(x)$ and hence $\widetilde{v}_K(x)$ must be even functions, so all of the odd coefficients of $\widetilde{v}_K(x)$ are zero and its polynomial order Φ_e must be an even ("e") positive integer. Because $\text{sign}(x)$ is an odd function, $\widetilde{s}(x)$ and hence $\widetilde{s}_K(x)$ must be odd functions, so all of the even coefficients of $\widetilde{s}_K(x)$ are zero, and its polynomial order Φ_o must be an odd ("o") positive integer.

In practice, since both $\widetilde{s}(x)$ and $\widetilde{v}(x)$ may contribute to the shape of the curve at each corner, we set their orders together by

$$\Phi_\text{e} = 2K, \qquad \Phi_\text{o} = 2K + 1 \,. \tag{40}$$

The first five ($K \in \{0, 1, 2, 3, 4\}$) instances of $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$ are shown in Fig. 4. Note that the $K = 0$ case is only shown as an aid to understanding—it is not actually of any real use, since it is equivalent to just making a PWL with twice as many line segments, and does not actually increase the smoothness compared to the original PWL representation.

Our final task is to come up with the polynomial coefficients $v_0, v_2, \cdots, v_{\Phi_\text{e}}$ and $s_1, s_3, \cdots, s_{\Phi_\text{o}}$ of $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$.

5.1. Linear algebra

The simplest conceptual approach is to write all of the constraints from (35)–(37) and (32)–(34) as two systems of equations, which can each written as a single linear algebra equation and solved with standard methods (matrix inversion, etc.).

For example, the $\widetilde{s}_0(x)$ setup would be

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \,, \tag{41}$$

and the $\widetilde{s}_1(x)$ setup would be

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & -2 & 3 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} \,. \tag{42}$$

This process can be repeated to systematically form a linear equation of arbitrary order, by forming each row of the matrix as

$$\begin{bmatrix} 1 & (\pm 1) & (\pm 1)^2 & (\pm 1)^3 & \cdots & (\pm 1)^{\Phi_\text{o}} \end{bmatrix} \left(\mathbf{D}^k\right)^\top \tag{43}$$

where the matrix $\mathbf{D}$ represents differentiation

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 2 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \Phi_\text{o} - 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \,. \tag{44}$$

The whole system can then be solved as-is, or we can preferably enforce the odd symmetry of the smoothed sign function by eliminating the even columns (since the even coefficients are guaranteed to be zero) and eliminating the even rows (since the constraints at $x = -1$ are now redundant, since odd symmetry is enforced).

The process for the smoothed absolute value function is nearly identical, although the constraints on the right hand side will need to match those in (32)–(34) instead, and even symmetry is instead enforced by eliminating the odd columns.

In both cases, the coefficients $v_0, v_2, \cdots, v_{\Phi_\text{e}}$ and $s_1, s_3, \cdots, s_{\Phi_\text{o}}$ of $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$ can be found with standard linear algebra, such as matrix inversion. Tabulated coefficients of $\widetilde{s}_K(x)$ and $\widetilde{v}_K(x)$ for $K \in \{0, 1, \cdots, 6\}$ are given in Tab. 3.

5.2. Closed-form

As an alternative to using linear algebra, we can used a closed-form equation that arises from a recursive construction. Robert Bristow-Johnson has discussed a soft clipper which is identical to our smoothed sign function [20–22].

In [21], Olli Niemitalo gives a useful recursive construction

$$\widetilde{s}_0(x) = x \tag{45}$$

$$\widetilde{s}_K(x) = \widetilde{s}_{K-1}(x) + \frac{(2K)!}{4^K (K!)^2} (1 - x^2)^K x \tag{46}$$

$$= \sum_{k=0}^{K} \frac{(2k)!}{4^k (k!)^2} (1 - x^2)^k x \,. \tag{47}$$

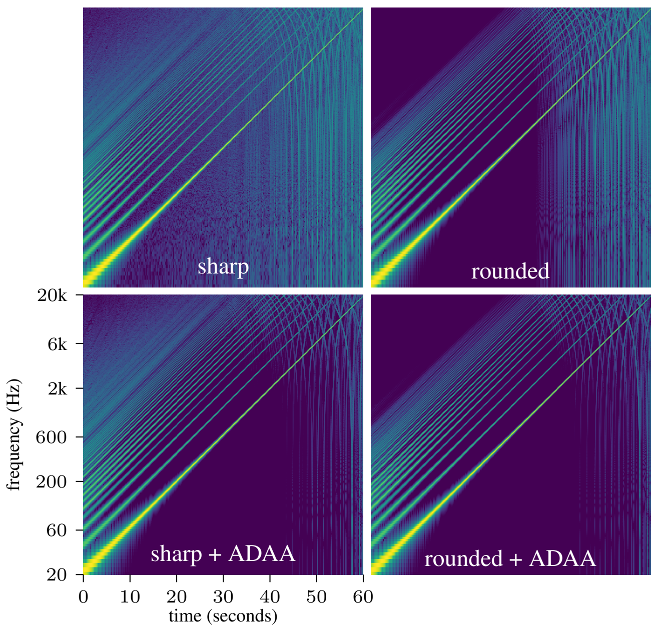


Figure 5: *Chirp response spectrograms showing the effect of rounding corners and/or applying first-order ADAA.*

The coefficients of $\widetilde{s}_K(x)$ are given in closed form by [21]

$$s_{2k+1} = \frac{(-1)^k (2K+1)!}{4^K K!(2k+1)k!(K-k)!}, \; k \in \{0, 1, \cdots, K\} \,. \quad (48)$$

We can use this same technique to generate the smoothed absolute value function $\widetilde{v}_K(x)$ by recognizing that—in the same way that $|x|$ is an antiderivative of $\text{sgn}(x)$—the smoothed absolute value function is itself an antiderivative of the smoothed sign function $\widetilde{s}_K(x)$, as can be seen in Fig. 4. This means that we can produce $\widetilde{s}_K(x)$ by first producing $\widetilde{v}_K(x)$ using Niemitalo's method and then integrating it. Since these are piecewise functions, we need to attend to the constants of integration in the same way that we discussed in §2. The only difference is that in the final step, we should not have $\widetilde{v}_K(0) = 0$, but rather $\widetilde{v}_K(\pm 1) = 1$.

5.3. Shifting polynomials

When we are not applying ADAA, we can shift and scale the normalized corner components $\widetilde{v}_K(x)$ and $\widetilde{s}_K(x)$ without any further effort, as in (24). However, when we are applying ADAA, we may prefer to deal with the raw polynomial coefficients directly. In this case, we need to be able to apply shifting and scaling operations to the polynomials. Vertically scaling a polynomial is trivial

$$Ap(x) = Ap_0 + Ap_1 x + Ap_2 x^2 + \cdots + Ap_\Phi x^\Phi \,. \quad (49)$$

Horizontal scaling is also fairly simple

$$p(Ax) = p_0 + Ap_1 x + A^2 p_2 x^2 + \cdots + A^\Phi p_\Phi x^\Phi \,. \quad (50)$$

Horizontal shifts are more complicated; we use the synthetic division based algorithm from [29].

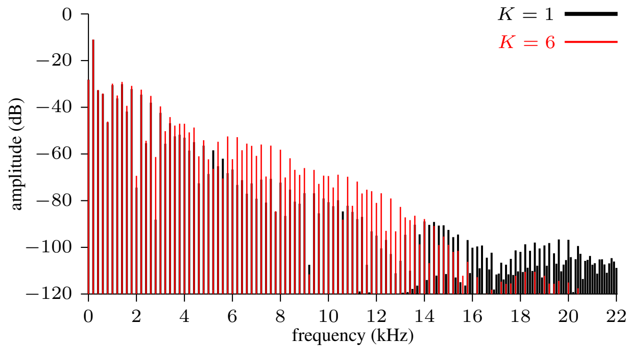


Figure 6: *Overtone amplitudes for* $K \in \{1, 6\}$ *for a* 5*-second-long* 200 *Hz sinusoid with an amplitude of* 2.0.

6. CASE STUDIES

We will now look at the chirp response for the PWL waveshaper shown in Fig. 3. Our input signal is a 60 second long logarithmic sinusoidal chirp from 20 Hz to 20 kHz, with an amplitude of 2.0 to drive it to hit all of the segments. Fig. 5 shows spectrograms of the chirp response of the "sharp" PWL waveshaper, a rounded version (all corners with $\omega = 0.25$), and versions of both with 1st-order ADAA applied. In all cases, the sampling rate is $f_s = 44100$ Hz (with no oversampling), and the curve smoothness is $K = 2$.

The "sharp" PWL waveshaper produces excessive aliasing, whereas the rounded version has far less, due to generally producing fewer overtones. For both the "sharp" PWL and rounded cases, applying ADAA is successful at eliminating most of the aliasing, coherent with results [11] for other waveshapes presented in the literature. In practice, due to the inherent filtering of ADAA, it should be used with at least 2× oversampling, which will mitigate the filtering as well as adding extra alias suppression.

A second case study looks at what happens when the order of smoothness K is varied. Fig. 6 shows the strength of the overtones for a 5-second-long static 200 Hz sine wave with an amplitude of 2.0 as input, fed into the same waveshaper from shown in Fig. 3, again with $\omega = 0.25$, and with 1st-order ADAA applied. We can glean a few things from this plot. First, the amplitudes of the lower overtones are somewhat unaffected by K. Second, we can see that increasing K raises the amplitudes of the middle-frequency overtones. Third, increasing K decreases the amplitudes of the higher-frequency overtones.

7. CONCLUSION

In this paper, we've shown how to apply the ADAA technique to any piecewise polynomial waveshaper, including the crucial step of manipulating each segment's constant of integration to ensure at least C^0 smoothness of each antiderivative used. This greatly expands the class of waveshaping functions that can be used with ADAA, while also providing a template for how to handle other piecewise waveshaping functions. We looked specifically into two special cases: the classic case of piecewise linear waveshaping functions and a proposed technique for rounding the corners on a piecewise linear waveshaper using special polynomial corners that enforce smoothness up to a certain order.

For these smoothed corners, we've shown how to derive $\widetilde{s}_K(x)$

and $\widetilde{v}_K(x)$ for positive integer K, which enforces C^K smoothness on our rounded corners. If one wanted to use K as a smoothly controllable parameter, the discrete nature of the set of curves that make up $\widetilde{s}_K(x)$ and $\widetilde{v}_K(x)$ (as shown in Fig. 4) will be disappointing. However, there is an easy way to form a set of polynomials with continuous K with smoothness $C^{\lfloor K \rfloor}$,

$$\widetilde{s}_K(x) = (\lceil K \rceil - K)\widetilde{s}_{\lfloor K \rfloor}(x) + (K - \lfloor K \rfloor)\widetilde{s}_{\lceil K \rceil}(x)\ , \quad (51)$$

which would allow the corner shape to be varied smoothly.

For the case of the piecewise linear waveshaper with rounded corners, an obvious question arises: what order of smooothnes should we choose? Increasing the order of a polynomial waveshaper increases the number of overtones it produces, so it may seem that increasing the order of the smoothness would increase the number of overtones. However, in fact the *number* of overtones in a non-trivial piecewise polynomial waveshaper is infinite. So, reasoning about the number of overtones is not the right approach. Thinking instead of the relative amplitudes of the overtones, we saw in the second case study that increasing the smoothness did not greatly affect the low-frequency overtones, increased the amplitude of middle-frequency overtones, and decreased energy for the higher overtones. It is hard to say conclusively for all situations whether this is good or bad—What we can say for sure is that being able to control the spectral profile in this way is a useful timbral control that has implications for the amount of aliasing, since suppressing the amplitude of high-frequency overtones (beyond the Nyquist limit) reduces aliasing. One downside of increasing the order is that it can increases numerical error in the calculation of the coefficients and evaluation of the polynomials. In practice, algorithm designers will have to be careful not to create a situation where this numerical error, which manifests as harsh buzzing, is audible. So, for now, we can only recommend choosing the order to taste—a more rigorous evaluation and recommendation could be the subject of future work.

8. ACKNOWLEDGMENTS

Thanks to Russell McClellan and Evan Lynch for helpful talks.

9. REFERENCES

[1] F. Esqueda, H. Pöntynen, V. Välimäki, and J. D. Parker, “Virtual analog Buchla 259 wavefolder,” in *Proc. Int. Conf. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, Sept. 5–9 2017, pp. 192–199.

[2] G. Gormond, F. Esqueda, H. Pöntynen, and J. D. Parker, “Waveshaping with Norton amplifiers: Modeling the Serge triple waveshaper,” in *Proc. Int. Conf. Digital Audio Effects (DAFx-18)*, Aveiro, Portugal, Sept. 2016, pp. 288–295.

[3] F. Esqueda, H. Pöntynen, and J. D. Parker, “Virtual analog models of the Lockhart and Serge wavefolders,” *Appl. Sci.*, vol. 7, no. 12, 2017, Art. #1328.

[4] H. Thornburg, “Antialiasing for nonlinearities: Acoustic modeling and synthesis applications,” in *Proc. Int. Comput. Music Conf.*, Beijing, China, Oct. 22–27 1999, pp. 66–69.

[5] J. Pakarinen and D. T. Yeh, “A review of digital techniques for modeling vacuum-tube guitar amplifiers,” *Comput. Music J.*, vol. 33, no. 2, pp. 85–100, Summer 2009.

[6] A. Bernardini, K. J. Werner, P. Maffezzoni, and A. Sarti, “Wave digital modeling of the diode-based ring modulator,” in *Proc. 144th Audio Eng. Conv.*, Milan, Italy, May 2018, pp. 66–69.

[7] K. J. Werner, V. Nangia, A. Bernardini, and J. O. Smith III A. Sarti, “An improved and generalized diode clipper model for wave digital filters,” in *Proc. 139th Audio Eng. Conv.*, New York, NY, Oct.–Nov. 2015.

[8] D. J. Gillespie and S. Schachter, “Model bending: Teaching circuit models new tricks,” in *Proc. Int. Conf. Digital Audio Effects (DAFx20in22)*, Vienna, Austria, Sept. 2022.

[9] D. T. Yeh, D. Nolting, and J. O. Smith III, “Physical and behavioral circuit modeling of the SP-12 sampler,” in *Proc. Int. Comput. Music Conf.*, Copenhagen, Denmark, Aug. 2007, pp. 299–302.

[10] J. Kahles, F. Esqueda, and V. Välimäki, “Oversampling for nonlinear waveshaping: Choosing the right filters,” *J. Audio Eng. Soc.*, vol. 67, no. 6, pp. 440–449, June 2019.

[11] J. D. Parker, V. Zavalishin, and E. Le Bivec, “Reducing the aliasing of nonlinear waveshaping using continuous-time convolution,” in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sept. 2016, pp. 137–144.

[12] D. Albertini, A. Bernardini, and A. Sarti, “Antiderivative antialiasing techniques in nonlinear wave digital structures,” *J. Audio Eng. Soc.*, vol. 69, pp. 448–464, July/Aug. 2021.

[13] K. J. Werner, “An equivalent circuit interpretation of antiderivative antialiasing,” in *Proc. Int. Conf. Digital Audio Effects (DAFx20in21)*, Vienna, Austria, Sept. 2021, pp. 17–24.

[14] M. Holters, “Antiderivative antialiasing for stateful systems,” in *Proc. Int. Conf. Digital Audio Effects (DAFx-19)*, Brimingham, U.K., Sept. 2019.

[15] M. Holters, “Combined derivative/antiderivative antialiasing,” in *Proc. Int. Conf. Digital Audio Effects (DAFx20in22)*, Vienna, Austria, Sept. 2022, pp. 86–93.

[16] P. P. La Pastina, S. D'Angelo, and L. Gabrielli, “Arbitrary-order IIR antiderivative antialiasing,” in *Proc. Int. Conf. Digital Audio Effects (DAFx20in21)*, Vienna, Austria, Sept. 2021, pp. 9–16.

[17] F. G. Germain, “Periodic analysis of nonlinear virtual analog models,” in *Proc. IEEE Work. Appl. Signal Process. Audio Acoust.*, New Paltz, NY, Oct. 2021, pp. 321–325.

[18] J.-N. Lin and R. Unbehauen, “Canonical representation: From piecewise-linear function to piecewise-smooth functions,” *IEEE Trans. Circuits Syst.—I: Fundamental Theory Appl.*, vol. 40, no. 7, pp. 461–468, July 1993.

[19] A. Bernardini and A. Sarti, “Canonical piecewise-linear representation of curves in the wave digital domain,” in *Proc. Europe. Signal Process. Conf.*, Kos, Greece, Aug.–Sept. 2017, pp. 1165–1169.

[20] R. Bristow-Johnson, “Family of soft clipping functions,” post on `music-dsp@music.columbia.edu` mailing list, 2014, `https://music-dsp.music.columbia.narkive.com/1JNTzu03/family-of-soft-clipping-functions#post1`.

[21] R. Bristow-Johnson, “Monotonic, symmetrical soft-clipping polynomial,” post on DSP Stack Exchange, Dec. 11 2016, `https://dsp.stackexchange.com/questions/36202/monotonic-symmetrical-soft-clipping-polynomial`.

[22] R. Bristow-Johnson, “Smoothstep sigmoid-like function: Can anyone prove this relation?,” post on Math Stack Exchange, Apr. 24 2017, `https://math.stackexchange.com/questions/2249296/smoothstep-sigmoid-like-function-can-anyone-prove-this-relation`.

[23] D. S. Ebert, Ed., *Texturing and modeling: A procedural approach*, AP Professional, Boston, 1994.

[24] J. O. Smith III, *Physical Audio Signal Processing*, W3K Publ., 2010.

[25] S. Enderby and Z. Baracskai, “Harmonic instability of digital soft clipping algorithms,” in *Proc. Int. Conf. Digital Audio Effects (DAFx-12)*, York, U.K., Sept. 2012.

[26] S. Bilbao, F. Esqueda, J. D. Parker, and V. Välimäki, “Antiderivative antialiasing for memoryless nonlinearities,” *IEEE Signal Process. Lett.*, vol. 24, no. 7, pp. 1049–1053, July 2017.

[27] A. Carson, “Aliasing reduction in virtual analogue modelling,” MSc. thesis, U. Edinburgh, Edinburgh, U.K., Sept. 2020.

[28] L. O. Chua and S. M. Kang, “Section-wise piecewise-linear functions: Canonical representation, properties, and applications,” *Proc. IEEE*, vol. 65, no. 6, pp. 915–929, June 1977.

[29] A. G. Akritas and S. D. Danielopoulos, “On the complexity of algorithms for the translation of polynomials,” *Computing*, vol. 24, pp. 51–60, Mar. 1980.