applied sciences

MDPI

Article

Antiderivative Antialiasing for Stateful Systems †

Martin Holters

Department of Signal Processing and Communication, Helmut Schmidt University, 22043 Hamburg, Germany; martin.holters@hsu-hh.de

† Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19).

Received: 21 November 2019; Accepted: 16 December 2019; Published: 18 December 2019

Abstract: Nonlinear systems, such as guitar distortion effects, play an important role in musical signal processing. One major problem encountered in digital nonlinear systems is aliasing distortion. Consequently, various aliasing reduction methods have been proposed in the literature. One of these is based on using the antiderivative of the nonlinearity and has proven effective, but is limited to memoryless systems. In this work, it is extended to a class of stateful systems which includes but is not limited to systems with a single one-port nonlinearity. Two examples from the realm of virtual analog modeling show its applicability to and effectiveness for commonly encountered guitar distortion effect circuits.

Keywords: antialiasing; aliasing mitigation; nonlinear signal processing; virtual analog modeling

1. Introduction

Nonlinear systems play an important role in musical signal processing. In particular, there are many effects categorized as overdrive, distortion, or fuzz, whose primary objective is to introduce harmonic distortion to enrich the signal. Usually, the nonlinear behavior is combined with (linear) filtering to spectrally shape the output signal or to make the amount of introduced distortion frequency-dependent. While many of these systems were originally designed in the analog domain, naturally, there is interest in deriving digital models for them, e.g., [1–4].

One major problem encountered in digital nonlinear systems, whether designed from scratch or derived by virtual analog modeling is aliasing distortion. Once the additional harmonics introduced by the nonlinearity exceed the Nyquist limit at half the sampling frequency, they get folded back to lower frequencies, just as if the corresponding analog signal had been sampled without appropriate band-limiting. Contrary to the desired harmonic distortion, aliasing distortion is usually perceived as unpleasant. Therefore, methods to suppress or reduce aliasing distortion are needed.

The conceptually simplest aliasing reduction method is oversampling. However, if the harmonics decay slowly with frequency, the oversampling factor has to be high, making the approach unattractive due to the rising computational demand. Consequently, various alternatives have been proposed, e.g., [5–11]. These methods, however, usually come with certain limitations, most commonly the restriction to memoryless systems. In this work, an extension of [10] is presented that loosens the restriction from memoryless systems to a certain class of stateful systems.

This article is an extension of our previous conference paper [12]. The main differences are the generalization of the method to the case of multiple parallel nonlinear functions and an added discussion of nonlinearities with vector-valued input.

2. Antiderivative-Based Aliasing Reduction for Memoryless Nonlinear Systems

As the proposed method builds upon the approach from [10], we shall briefly summarize the latter. Conceptually, the digital signal is converted to a continuous-time signal using linear interpolation

between consecutive samples, the nonlinearity is applied, and the result is lowpass-filtered by integrating over one sampling interval before sampling again to obtain the digital output signal. Thus, the nonlinear system

$$y(n) = f\big(u(n)\big), \tag{1}$$

where $f(u)$ denotes the nonlinear function, mapping input sample $u(n)$ to output sample $y(n)$, is transformed into

$$y(n) = \int_0^1 f\big(\bar{u}_n(\tau)\big)d\tau \qquad \text{with} \qquad \bar{u}_n(\tau) = (1-\tau)u(n-1) + \tau u(n). \tag{2}$$

Only rarely, the integral will have a closed solution and numerical integration at run time is unattractive considering the computational load. However, by switching the integration variable, we obtain

$$y(n) = \frac{1}{u(n) - u(n-1)} \int_{u(n-1)}^{u(n)} f\big(\bar{u}\big)d\bar{u}, \tag{3}$$

where we may apply the fundamental theorem of calculus and by further special-casing $u(n) \approx u(n-1)$ to avoid numerical issues, we finally arrive at

$$y(n) = \tilde{f}\big(u(n), u(n-1)\big) = \begin{cases} \frac{F(u(n)) - F(u(n-1))}{u(n) - u(n-1)} & \text{if } u(n) \not\approx u(n-1) \\ f\big(\frac{1}{2}u(n) + \frac{1}{2}u(n-1)\big) & \text{if } u(n) \approx u(n-1) \end{cases} \tag{4}$$

where $F(u) = \int f(u)du$ is the antiderivative of $f(u)$. Again, the antiderivative may not have a closed solution, but being a function in one variable, it may be pre-computed and stored in a lookup table.

In addition to reducing aliasing artifacts, the approach introduces a half-sample delay and attenuates signal components not only above the Nyquist limit, but also at high frequencies below it, i.e., it acts as a low-pass filter in the audio band. This can be readily seen when using the identity function $f(u) = u$ instead of a true nonlinearity. In that case, straightforward calculation yields

$$y(n) = \tfrac{1}{2}u(n) + \tfrac{1}{2}u(n-1), \tag{5}$$

which is a first-order finite impulse response low-pass filter with a group delay of half a sample. The low-pass effect can be countered by a modest amount of oversampling (e.g., by a factor of two) and the delay usually is of no concern.

The Higher-Dimensional Case

Before turning to stateful systems, we shall briefly discuss how to lift the restriction to systems with one scalar input $u(n)$ and one scalar output $y(n)$ quietly assumed so far. To keep the notation concise, we collect the inputs $u_1(n), u_2(n), \ldots, u_{K_\mathrm{u}}(n)$ of a system with K_u scalar inputs into an input vector $\boldsymbol{u}(n) = \begin{pmatrix} u_1(n) & u_2(n) & \cdots & u_{K_\mathrm{u}}(n) \end{pmatrix}$ (using bold letters to indicate vectors here and in the following). Likewise, multiple outputs are collected into an output vector $\boldsymbol{y}(n) = \begin{pmatrix} y_1(n) & y_2(n) & \cdots & y_{K_\mathrm{y}}(n) \end{pmatrix}$. We may first notice that the extension to vector-valued output is straightforward: the above derivation does not depend at all on the fact that the non-linear function (or its antiderivative) is scalar-valued.

For vector-valued input, on the other hand, things do not work out so nicely anymore. In particular, we can no longer rewrite (2) as (3). So unless the integral in (2) happens to have a closed solution, to avoid numerical solving at run time, we are almost stuck with precomputing and tabulating the solution for all combinations of $\boldsymbol{u}(n)$ and $\boldsymbol{u}(n-1)$ at a certain density. Thus, if the input is K_u-dimensional, having to use two subsequent input vectors stacked on top of each other to form the lookup (index) vector, we would require a $2K_\mathrm{u}$-dimensional lookup table. A small improvement is possible by realizing that we always integrate along lines, and a line can be parameterized by

$2K_\mathrm{u} - 2$ parameters in a K_u-dimensional space. Along a line, we can precompute the antiderivative with respect to the scalar coordinate along the line. This antiderivative with a one-dimensional argument and $2K_\mathrm{u} - 2$ parameters could be stored in a $2K_\mathrm{u} - 1$-dimensional lookup table. While this may be reasonable for $K_\mathrm{u} = 2$, it quickly becomes infeasible for larger K_u due to prohibitive memory requirements.

3. Extension to Stateful Systems

The half-sample delay introduced by the method of [10] becomes problematic if the nonlinearity is embedded in the feedback loop of a stateful system. As noted in [10], for the particular case of an integrator following the nonlinearity and using trapezoidal rule for time-discretization, one can simply replace the numerator of the discretized integrator's transfer function with the filter introduced by antialiasing. This fusing of antialiased nonlinearity and integrator then has no additional delay compared to the system without antialiasing, hence can be used inside a feedback system without problems.

Here, we consider systems that do not necessarily have an integrator following the nonlinearity. In particular, we shall consider the general discrete nonlinear state-space system

$$\boldsymbol{x}(n) = \boldsymbol{f}_{\mathrm{x},0}(\boldsymbol{p}_{\mathrm{x},0}(n)) + \cdots + \boldsymbol{f}_{\mathrm{x},L}(\boldsymbol{p}_{\mathrm{x},L}(n)) \tag{6}$$

$$\boldsymbol{y}(n) = \boldsymbol{f}_{\mathrm{y},0}(\boldsymbol{p}_{\mathrm{y},0}(n)) + \cdots + \boldsymbol{f}_{\mathrm{y},M}(\boldsymbol{p}_{\mathrm{y},M}(n)), \tag{7}$$

with

$$\boldsymbol{p}_{\mathrm{x},l}(n) = \boldsymbol{C}_{\mathrm{p}_{\mathrm{x},l}}\boldsymbol{x}(n-1) + \boldsymbol{D}_{\mathrm{p}_{\mathrm{x},l}}\boldsymbol{u}(n) \tag{8}$$

$$\boldsymbol{p}_{\mathrm{y},m}(n) = \boldsymbol{C}_{\mathrm{p}_{\mathrm{y},m}}\boldsymbol{x}(n-1) + \boldsymbol{D}_{\mathrm{p}_{\mathrm{y},m}}\boldsymbol{u}(n), \tag{9}$$

where $\boldsymbol{x}(n) \in \mathbb{R}^{K_\mathrm{x}}$ is the state vector, $\boldsymbol{u}(n) \in \mathbb{R}^{K_\mathrm{u}}$ is the input vector, and $\boldsymbol{y}(n) \in \mathbb{R}^{K_\mathrm{y}}$ is the output vector. The nonlinearity of the system is captured by the (potentially) nonlinear functions $\boldsymbol{f}_{\mathrm{x},l} : \mathbb{R}^{K_{\mathrm{p}_{\mathrm{x},l}}} \to \mathbb{R}^{K_\mathrm{x}}$ and $\boldsymbol{f}_{\mathrm{y},m} : \mathbb{R}^{K_{\mathrm{p}_{\mathrm{y},m}}} \to \mathbb{R}^{K_\mathrm{y}}$ for state update and output, respectively. Their arguments $\boldsymbol{p}_{\mathrm{x},l}(n) \in \mathbb{R}^{K_{\mathrm{p}_{\mathrm{x},l}}}$ and $\boldsymbol{p}_{\mathrm{y},m}(n) \in \mathbb{R}^{K_{\mathrm{p}_{\mathrm{y},m}}}$ are calculated by (8) and (9) as a linear combination of previous states and current input where the coefficient matrices $\boldsymbol{C}_{\mathrm{p}_{\mathrm{x},l}} \in \mathbb{R}^{K_{\mathrm{p}_{\mathrm{x},l}} \times K_\mathrm{x}}$, $\boldsymbol{D}_{\mathrm{p}_{\mathrm{x},l}} \in \mathbb{R}^{K_{\mathrm{p}_{\mathrm{x},l}} \times K_\mathrm{u}}$, $\boldsymbol{C}_{\mathrm{p}_{\mathrm{y},m}} \in \mathbb{R}^{K_{\mathrm{p}_{\mathrm{y},m}} \times K_\mathrm{x}}$ and $\boldsymbol{D}_{\mathrm{p}_{\mathrm{y},m}} \in \mathbb{R}^{K_{\mathrm{p}_{\mathrm{y},m}} \times K_\mathrm{u}}$ depend on the system. Some remarks are in order:

- We not only allow multiple states, collected in the vector $\boldsymbol{x}(n)$, but similarly multiple inputs in $\boldsymbol{u}(n)$ and multiple outputs in $\boldsymbol{y}(n)$. In many applications, input and output will however be scalar.
- While having only one nonlinear function on the right-hand side, i.e., $L = M = 0$, is perfectly valid, we allow the decomposition into a sum of multiple functions. We do this because, as discussed above, nonlinear functions with scalar or very low-dimensional argument are much better suited to the proposed method than those with higher-dimensional argument. Thus, while we theoretically allow vector-valued $\boldsymbol{p}_{\mathrm{x},l}(n)$ and $\boldsymbol{p}_{\mathrm{y},m}(n)$ of arbitrary dimension, from a practical viewpoint, their dimensionalities $K_{\mathrm{p}_{\mathrm{x},l}}$ and $K_{\mathrm{p}_{\mathrm{y},m}}$ should be low. Therefore, if the system's nonlinearity can be decomposed into a sum of nonlinear functions with low-dimensional arguments, it should be. It has to be noted, though, that unfortunately not all systems allow their nonlinearity to be decomposed in such a way.
- It may often be desirable to differentiate between linear and nonlinear parts of this system. By letting $\boldsymbol{f}_{\mathrm{x},0}(\boldsymbol{p}_{\mathrm{x},0}(n)) = \boldsymbol{p}_{\mathrm{x},0}(n)$ and $\boldsymbol{f}_{\mathrm{y},0}(\boldsymbol{p}_{\mathrm{y},0}(n)) = \boldsymbol{p}_{\mathrm{y},0}(n)$ be the identity function and further

renaming $\boldsymbol{A} = \boldsymbol{C}_{\mathrm{p}_{\mathrm{x},0}} \in \mathbb{R}^{K_\mathrm{x} \times K_\mathrm{x}}$, $\boldsymbol{B} = \boldsymbol{D}_{\mathrm{p}_{\mathrm{x},0}} \in \mathbb{R}^{K_\mathrm{x} \times K_\mathrm{u}}$, $\boldsymbol{C}_\mathrm{y} = \boldsymbol{C}_{\mathrm{p}_{\mathrm{y},0}} \in \mathbb{R}^{K_\mathrm{y} \times K_\mathrm{x}}$, and $\boldsymbol{D}_\mathrm{y} = \boldsymbol{D}_{\mathrm{p}_{\mathrm{y},0}} \in \mathbb{R}^{K_\mathrm{y} \times K_\mathrm{u}}$, we therefore rewrite our system as

$$\boldsymbol{x}(n) = \boldsymbol{A}\boldsymbol{x}(n-1) + \boldsymbol{B}\boldsymbol{u}(n) + \boldsymbol{f}_{\mathrm{x},1}(\boldsymbol{p}_{\mathrm{x},1}(n)) + \cdots + \boldsymbol{f}_{\mathrm{x},L}(\boldsymbol{p}_{\mathrm{x},L}(n)) \tag{10}$$

$$\boldsymbol{y}(n) = \boldsymbol{C}_\mathrm{y}\boldsymbol{x}(n-1) + \boldsymbol{D}_\mathrm{y}\boldsymbol{u}(n) + \boldsymbol{f}_{\mathrm{y},1}(\boldsymbol{p}_{\mathrm{y},1}(n)) + \cdots + \boldsymbol{f}_{\mathrm{y},M}(\boldsymbol{p}_{\mathrm{y},M}(n)). \tag{11}$$

A visual representation is given as a block diagram in Figure 1.

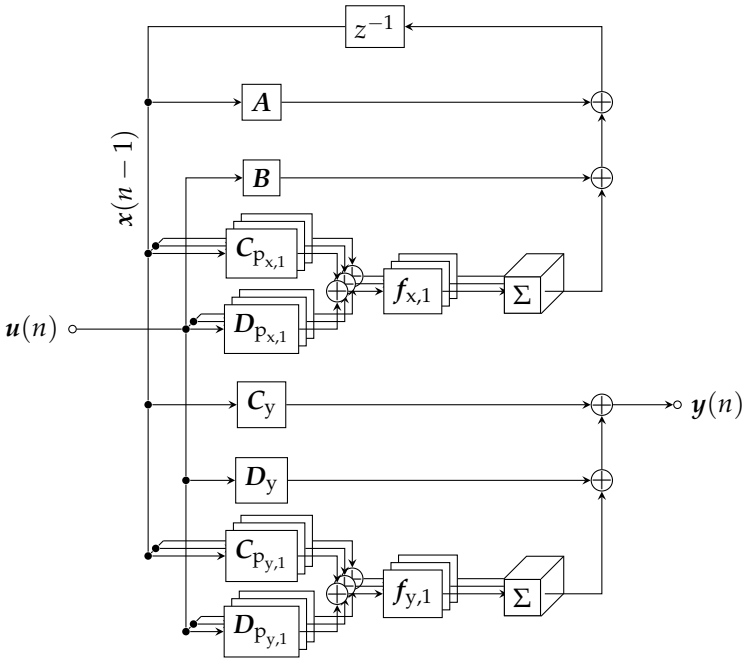


Figure 1. Block diagram of the considered system according to (8)–(11) (for $M = L = 3$).

- If the system is obtained in the context of virtual analog modeling, usually the nonlinear functions will only be given implicitly (as the solution of what is sometimes referred to as a delay-free loop), making solving a nonlinear equation necessary. However, they are typically based on a common set of functions $\boldsymbol{f}_m$, only applying different weighting to their outputs, i.e.,

$$\boldsymbol{f}_{\mathrm{x},m}(\boldsymbol{p}_{\mathrm{x},m}(n)) = \boldsymbol{W}_{\mathrm{x},m}\boldsymbol{f}_m(\boldsymbol{p}_m(n)) \tag{12}$$

$$\boldsymbol{f}_{\mathrm{y},m}(\boldsymbol{p}_{\mathrm{y},m}(n)) = \boldsymbol{W}_{\mathrm{y},m}\boldsymbol{f}_m(\boldsymbol{p}_m(n)) \tag{13}$$

  with $L = M$ and $\boldsymbol{p}_m(n) = \boldsymbol{p}_{\mathrm{x},m}(n) = \boldsymbol{p}_{\mathrm{y},m}(n)$. While this redundancy should be kept in mind for optimizing an implementation, we will derive our method for the more general case of possibly independent nonlinear functions for state update and output.

In a first step, we may consider only applying the aliasing suppression to $\boldsymbol{f}_{\mathrm{y},m}(\boldsymbol{p}_{\mathrm{y},m})$, as they are not part of any feedback loop. We have to be careful though, and may not just replace $\boldsymbol{f}_{\mathrm{y},m}$ with $\tilde{\boldsymbol{f}}_{\mathrm{y},m}$ in (11), as that would lead to a misalignment in time of the linear and antialiased nonlinear terms. Instead, we have to apply the antialiasing also to the linear part $\boldsymbol{f}_{\mathrm{y},0}$, even though it obviously does not introduce aliasing distortion. Fortunately, the integral has a closed solution no matter the dimensionality, namely

$$\tilde{\boldsymbol{f}}_{\mathrm{y},0}(\boldsymbol{u}(n), \boldsymbol{u}(n-1)) = \frac{1}{2}\boldsymbol{C} \cdot \big(\boldsymbol{x}(n-1) + \boldsymbol{x}(n-2)\big) + \frac{1}{2}\boldsymbol{D} \cdot \big(\boldsymbol{u}(n) + \boldsymbol{u}(n-1)\big), \tag{14}$$

resulting in

$$y(n) = \frac{1}{2}\boldsymbol{C}\cdot\big(\boldsymbol{x}(n-1)+\boldsymbol{x}(n-2)\big)+\frac{1}{2}\boldsymbol{D}\cdot\big(\boldsymbol{u}(n)+\boldsymbol{u}(n-1)\big)+\sum_{m=1}^{M}\tilde{f}_{\mathrm{y},m}(\boldsymbol{p}_{\mathrm{y},m}(n),\boldsymbol{p}_{\mathrm{y},m}(n-1)). \quad (15)$$

However, any aliasing distortion introduced into $\boldsymbol{x}(n)$ by (10) will not undergo any mitigation (except for the lowpass filtering).

Now, if we naively rewrite (10) as we did with (11), we modify our system in an unwanted way as we introduce additional delay in the feedback. But we do that in a very controlled way: the unit delay in the feedback is replaced by a delay of 1.5 samples. As a delay of 1.5 samples corresponds to a delay of one sample at $\frac{2}{3}$ of the sampling rate, we can compensate for the extra half-sample delay by designing our system for this reduced sampling rate, but then operating it at the original sampling rate. We thus arrive at

$$\boldsymbol{x}(n) = \frac{1}{2}\tilde{\boldsymbol{A}}\cdot\big(\boldsymbol{x}(n-1)+\boldsymbol{x}(n-2)\big)+\frac{1}{2}\tilde{\boldsymbol{B}}\cdot\big(\boldsymbol{u}(n)+\boldsymbol{u}(n-1)\big)+\sum_{l=1}^{L}\tilde{\boldsymbol{f}}_{\mathrm{x},l}(\boldsymbol{p}_{\mathrm{x},l}(n),\boldsymbol{p}_{\mathrm{x},l}(n-1)) \quad (16)$$

$$\boldsymbol{y}(n) = \frac{1}{2}\tilde{\boldsymbol{C}}\cdot\big(\boldsymbol{x}(n-1)+\boldsymbol{x}(n-2)\big)+\frac{1}{2}\tilde{\boldsymbol{D}}\cdot\big(\boldsymbol{u}(n)+\boldsymbol{u}(n-1)\big)+\sum_{m=1}^{M}\tilde{\boldsymbol{f}}_{\mathrm{y},m}(\boldsymbol{p}_{\mathrm{y},m}(n),\boldsymbol{p}_{\mathrm{y},m}(n-1)), \quad (17)$$

with

$$\boldsymbol{p}_{\mathrm{x},l}(n) = \tilde{\boldsymbol{C}}_{\mathrm{p}_{\mathrm{x},l}}\boldsymbol{x}(n-1)+\tilde{\boldsymbol{D}}_{\mathrm{p}_{\mathrm{x},l}}\boldsymbol{u}(n) \quad (18)$$

$$\boldsymbol{p}_{\mathrm{y},m}(n) = \tilde{\boldsymbol{C}}_{\mathrm{p}_{\mathrm{y},m}}\boldsymbol{x}(n-1)+\tilde{\boldsymbol{D}}_{\mathrm{p}_{\mathrm{y},m}}\boldsymbol{u}(n), \quad (19)$$

where all coefficients are calculated for the reduced sampling rate $\tilde{f}_\mathrm{s} = \frac{2}{3}f_\mathrm{s}$. We can only do this because we do not have a delay-free loop. Or rather, the delay-free loop is hidden inside the nonlinear functions: Instead of worrying about a nonlinearity within a delay-free loop, we treat the solution of the delay-free loop as the nonlinearity to apply aliasing reduction to. Note that the behavior for frequencies above $\frac{1}{2}\tilde{f}_\mathrm{s} = \frac{1}{3}f_\mathrm{s}$ is ill-defined, but with the mild oversampling suggested by [10] anyway, we do not have to worry about this.

The increased delay is not the only effect of the modification. There is also the low-pass filtering. To study this in more detail, assume all $\boldsymbol{f}_{\mathrm{x},l}(\boldsymbol{p}_{\mathrm{x},l})$ and $\boldsymbol{f}_{\mathrm{y},m}(\boldsymbol{p}_{\mathrm{y},m})$ to be linear so that we have a linear system, and let $H(z)$ denote the transfer function obtained from (8)–(11). If we instead use (16)–(19) without adjusting the coefficients, it is straightforward to verify that the resulting transfer function fulfills

$$\tilde{H}(z) = \frac{1}{2}(1+z^{-1})\cdot H\Big(\big(\tfrac{1}{2}(z^{-1}+z^{-2})\big)^{-1}\Big). \quad (20)$$

We may observe two effects: the well-known filtering with a factor on the outside and the substitution $z \leftarrow \big(\frac{1}{2}(z^{-1}+z^{-2})\big)^{-1}$ in the argument of H. Evaluating the latter for $z = e^{j\omega}$, we note that

$$\big(\tfrac{1}{2}(e^{-j\omega}+e^{-2j\omega})\big)^{-1} = \frac{1}{\cos(\frac{1}{2}\omega)}e^{\frac{3}{2}j\omega} \quad (21)$$

depicted in Figure 2.

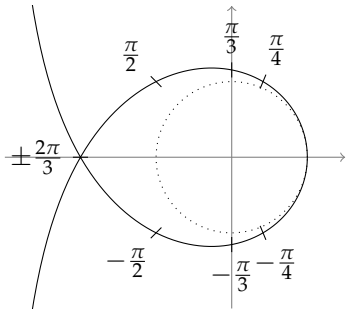


Figure 2. Trajectory of $\left(\frac{1}{2}e^{-j\omega}+\frac{1}{2}e^{-2j\omega}\right)^{-1}$ compared to the unit circle $e^{j\omega}$ (dotted).

While in the original system $H(z)$ is evaluated on the unit circle $e^{j\omega}$ (shown dotted) to obtain the frequency response, for the modified system, the original $H(z)$ is evaluated on the trajectory of (21). We notice that, in addition to the frequency scaling by $\frac{3}{2}$, there is an additional scaling away from the unit circle, increasing with frequency. Importantly, as we only evaluate $H(z)$ for z on or outside the unit circle, we preserve stability, i.e., if $H(z)$ is stable, so is $\tilde{H}(z)$. Nevertheless, especially for higher frequencies, this may cause a significant distortion of the frequency response.

An extreme example would be an all-pass filter with high Q-factor, where the transformation might result in the zero moving onto the frequency axis, turning a flat frequency response into one with a deep notch. As the examples will demonstrate, many typical systems are rather well-behaved under the transformation, but one has to be aware of this pitfall.

4. Computational Cost

In addition to potentially altering the system behavior for high frequencies, the price to pay for applying the proposed antialiasing approach is computational cost. Obviously, the exact total number of operations needed depends on the actual system the method will be applied to, but we may observe some trends. First, by the necessary oversampling by a factor of two, the computational cost doubles even for the operations we do not otherwise alter. This is, of course, still better than just using a higher oversampling factor and no other antialiasing at all. If the input and output sampling rates are fixed, the required resampling necessitates interpolation and decimation filters, which incur additional cost, but for the proposed approach and naive oversampling alike.

To judge the computational cost of the modifications made by the proposed method, let us assume that both the nonlinear functions and their antiderivatives are stored in lookup tables with the same resolution and lookup/interpolation method. Then replacing the nonlinear functions by (4) introduces a division and slightly increases the number of required table lookups. Note that the number of table lookups does not double, as $\boldsymbol{F}(\boldsymbol{p}(n))$ may be stored to avoid the lookup of $\boldsymbol{F}(\boldsymbol{p}(n-1))$ in the next time step. Only when switching between the two cases in (4), an additional lookup is needed. However, the computational cost of the interpolation during table lookup grows exponentially with its dimensionality.

So if all nonlinear functions have only scalar arguments, the extra effort per time step is a number of additions and multiplications by $\frac{1}{2}$ and one division per nonlinear function, and the occasional extra table lookup. This will almost certainly be more efficient than increasing the sampling rate even further. However, when the nonlinear functions have arguments of higher dimensionality, the (almost) doubled dimensionality for the antiderivatives leads to a quickly increasing computational cost for the table lookups, making the method unattractive in that case, in addition to the exploding memory requirements discussed above.

5. Examples

We shall exemplify the method with two circuits well-suited for it as they result in nonlinear functions with scalar input. The Julia source code implementing the derived models can be found at https://github.com/martinholters/ADAA_Examples.jl.

5.1. Diode Clipper

As the first example, we consider the diode clipper of Figure 3.

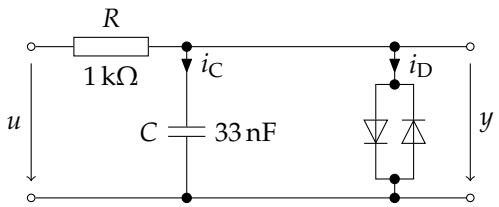


Figure 3. Schematics of the modeled diode clipper.

The circuit is simple enough that we briefly repeat the modeling process here. From Kirchhoff's and Ohm's laws and $i_C = C\dot{y}$, we immediately obtain

$$y = u - R \cdot (i_C + i_D) = u - RC \cdot \dot{y} - Ri_D. \tag{22}$$

Summing over two subsequent sampling instances, we get

$$y(n) + y(n-1) = u(n) + u(n-1) - RC \cdot (\dot{y}(n) + \dot{y}(n-1)) - R \cdot (i_D(n) + i_D(n-1)). \tag{23}$$

We now use the trapezoidal rule to substitute

$$\dot{y}(n) + \dot{y}(n-1) = 2f_s \cdot (y(n) - y(n-1)) \tag{24}$$

and obtain

$$y(n) + y(n-1) = u(n) + u(n-1) - 2RCf_s \cdot (y(n) - y(n-1)) - R \cdot (i_D(n) + i_D(n-1)). \tag{25}$$

Collecting all quantities from time step $n-1$ into canonical states

$$x(n-1) = (2RCf_s - 1) \cdot y(n-1) + u(n-1) - Ri_D(n-1) \tag{26}$$

allows simplification to

$$y(n) = x(n-1) + u(n) - 2RCf_s y(n) - Ri_D(n). \tag{27}$$

Using Shockley's equation for the diodes, we get

$$i_D(n) = I_S \cdot \left(e^{y(n)/v_T} - 1\right) - I_S \cdot \left(e^{-y(n)/v_T} - 1\right) = 2I_S \sinh\left(y(n)/v_T\right), \tag{28}$$

where saturation current and temperature voltage have been chosen as $I_S = 1\,\mathrm{fA}$ and $v_T = 25\,\mathrm{mV}$ respectively. Inserting (28) into (27) and introducing

$$p_y(n) = x(n-1) + u(n) \tag{29}$$

then leads to the implicit equation

$$y(n) = p_y(n) - 2RCf_s y(n) - 2RI_S \sinh\left(y(n)/v_T\right) \tag{30}$$

for $y(n)$. Note that we do not treat this as a delay-free loop and apply the antialiasing to the sinh-function. Instead, we let $f_\mathrm{y}(p_\mathrm{y}(n)) = y(n)$ denote the solution of the implicit equation. The resulting function is depicted in Figure 4 (obtained using an iterative solver).

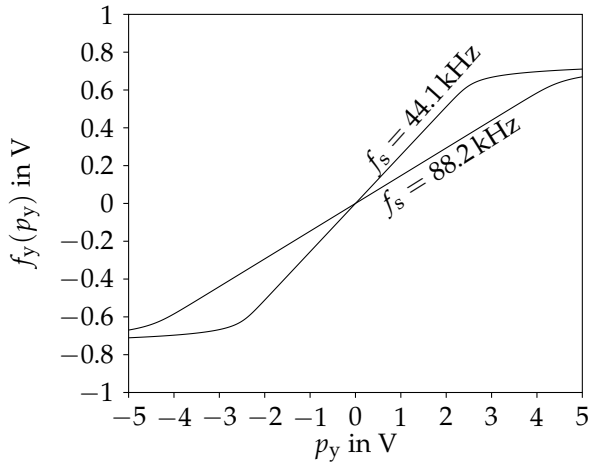


Figure 4. Nonlinear function $f_\mathrm{y}(p_\mathrm{y})$ of the diode clipper for two different sampling rates f_s.

To obtain the state update equation, we rearrange (27) to

$$(2RCf_\mathrm{s} - 1) \cdot y(n) + u(n) - Ri_\mathrm{D}(n) = -x(n-1) + 4RCf_\mathrm{s}y(n) \tag{31}$$

and note by comparing with (26) that the left-hand side equals $x(n)$. Thus introducing

$$f_\mathrm{x}(p_\mathrm{x}(n)) = 4RCf_\mathrm{s}f_\mathrm{y}(p_\mathrm{y}(n)) \tag{32}$$

with $p_\mathrm{x}(n) = p_\mathrm{y}(n)$, we arrive at

$$x(n) = -x(n-1) + f_\mathrm{x}(p_\mathrm{x}(n)) \tag{33}$$
$$y(n) = f_\mathrm{y}(p_\mathrm{y}(n)) \tag{34}$$

of the desired form.

Applying the aliasing mitigation only to the output equation is particularly simple in this case, giving

$$y(n) = \tilde{f}_\mathrm{y}(p_\mathrm{y}(n), p_\mathrm{y}(n-1)) \tag{35}$$

with $\tilde{f}_\mathrm{y}$ defined according to (4). The required antiderivative $F_\mathrm{y}(p_\mathrm{y})$ of $f_\mathrm{y}(p_\mathrm{y})$, depicted in Figure 5, has to be determined numerically. For the results below, it has been precomputed at 1024 uniformly distributed points in the relevant range and stored in a table, using cubic interpolation during lookup.

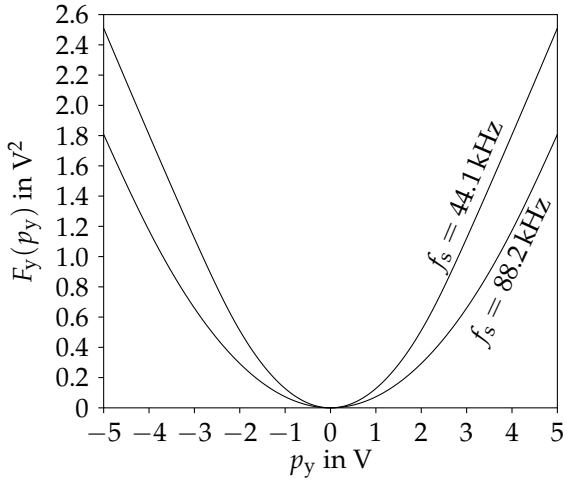


Figure 5. Antiderivative $F_y(p_y)$ of $f_y(p_y)$ of the diode clipper for two different sampling rates f_s.

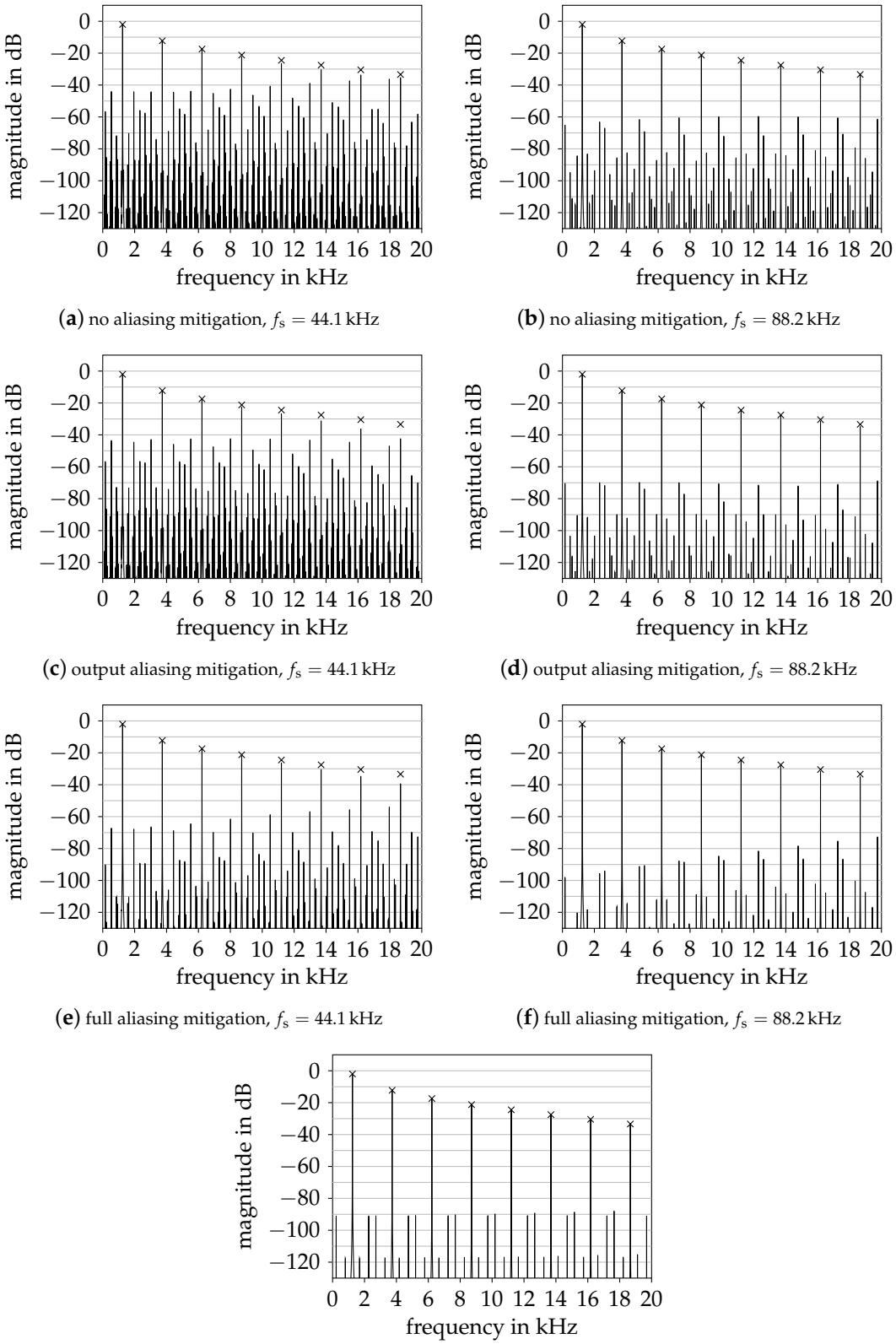
To also apply aliasing mitigation to the state update equation, we have to change it to

$$x(n) = -\tfrac{1}{2}\big(x(n-1) + x(n-2)\big) + \tilde{f}_x(p_x(n), p_x(n-1)) \tag{36}$$

and substitute $\tilde{f}_s = \frac{2}{3} f_s$ for f_s in (30) and (32). Note that this immediately leads to

$$\tilde{f}_x(p_x(n), p_x(n-1)) = 4RC\tilde{f}_s\tilde{f}_y(p_y(n), p_y(n-1)). \tag{37}$$

To study the effectiveness of the method, we consider Figure 6, where the output spectra for a sinusoidal excitation is depicted for various model configurations. Figure 6a,b give the baseline, the system without any aliasing mitigation at sampling rates $f_s = 44.1\,\mathrm{kHz}$ and $f_s = 88.2\,\mathrm{kHz}$, respectively. Only applying aliasing mitigation to the output equation according to (35) is of little benefit, as seen when considering Figure 6c,d in comparison. We do note, however, the low-pass effect in Figure 6c, where higher harmonics exhibit an attenuation of up to 10 dB.



(**a**) no aliasing mitigation, $f_s = 44.1\,\text{kHz}$

(**b**) no aliasing mitigation, $f_s = 88.2\,\text{kHz}$

(**c**) output aliasing mitigation, $f_s = 44.1\,\text{kHz}$

(**d**) output aliasing mitigation, $f_s = 88.2\,\text{kHz}$

(**e**) full aliasing mitigation, $f_s = 44.1\,\text{kHz}$

(**f**) full aliasing mitigation, $f_s = 88.2\,\text{kHz}$

(**g**) no aliasing mitigation, $f_s = 220.5\,\text{kHz} = 5 \cdot 44.1\,\text{kHz}$

Figure 6. Output spectra of various model configurations for the diode clipper when excited with a single sinusoid of 10 V amplitude at 1244.5 Hz. Crosses mark expected harmonics.

When also applying the aliasing mitigation to the state update equation according to (36), we observe a significant aliasing reduction in Figure 6e,f. As explained, the aliasing mitigation should be combined with (modest) oversampling. In this particular case, as verified in Figure 6e, the model is still a relatively good fit even without oversampling, which however must be mainly attributed to lucky coincidence. More relevant is the case of a sampling rate of $f_s = 88.2\,\mathrm{kHz}$, shown in Figure 6f. Comparing to oversampling to $f_s = 220.5\,\mathrm{kHz}$ without additional aliasing mitigation measures, as shown in Figure 6g, we see that the aliased components at low frequencies, where they are most easily perceived, are at a comparable level.

5.2. Tube Screamer-Like Distortion Circuit

As a second example, we consider the distortion circuit of Figure 7, inspired by the Ibanez Tube Screamer TS-808.

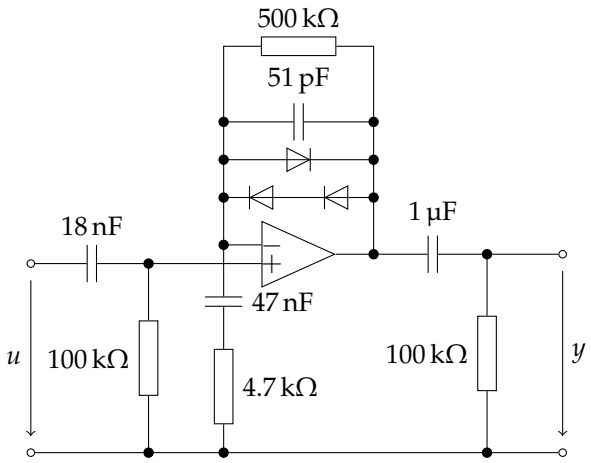
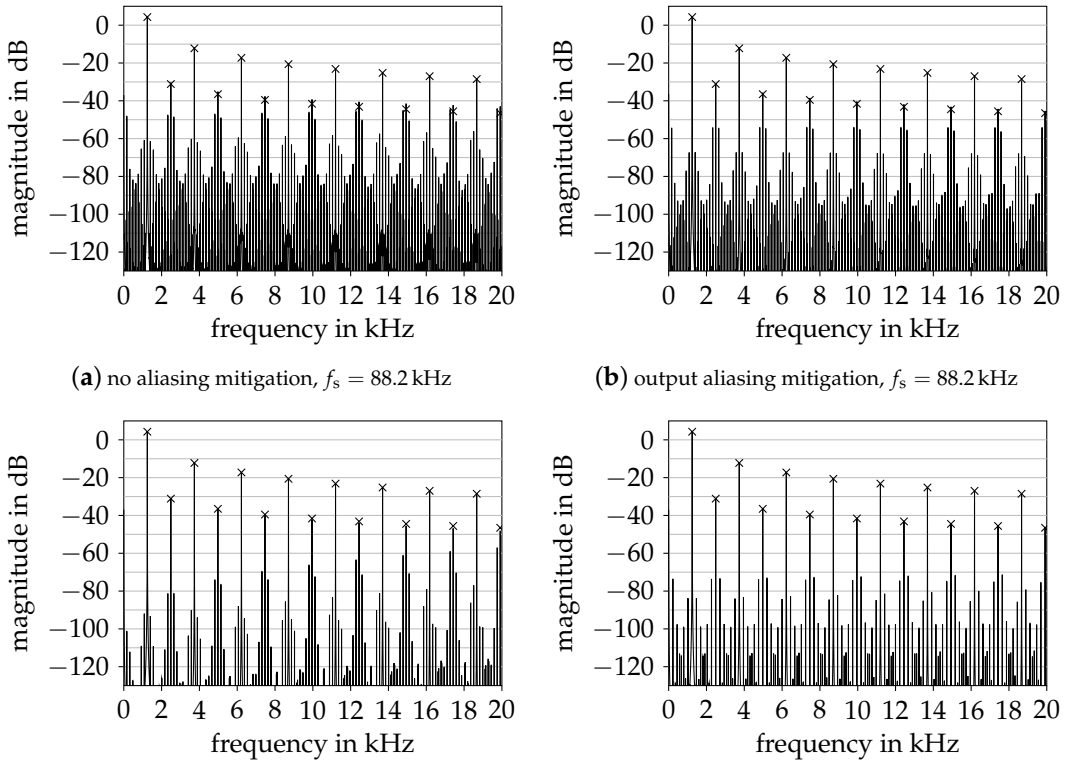


Figure 7. Tube screamer-like distortion circuit.

We shall not go into details of the modeling procedure, for which we have used ACME.jl (https://github.com/HSU-ANT/ACME.jl), but remark that if one allows the three diodes to be different, one can no longer derive a closed-form expression for their combined behavior. Instead, the nonlinear behavior is determined by a system of three equations. Nevertheless, using the dimensionality reduction approach of [13], the input $p_x(n) = p_y(n)$ to the nonlinearity can be reduced to a scalar value, formed by a linear combination of the input and the capacitor states. Hence, the proposed method is applicable.

Figure 8 again shows the output spectra for a sinusoidal excitation. As can be seen in Figure 8a, with plain oversampling to $f_s = 88.2\,\mathrm{kHz}$, the signal contains strong aliasing components. Applying aliasing mitigation only to the output equation reduces the aliasing distortion to a limited extent, as shown in Figure 8b. In contrast, when also applying aliasing mitigation to the state update equation, the aliasing is significantly reduced, as seen in Figure 8c. Again, the aliasing mitigation is most effective at low frequencies, where it is also perceptually most relevant. As in the diode clipper example, for low frequencies the system with aliasing mitigation at $f_s = 88.2\,\mathrm{kHz}$ performs at least as good as an unmodified system at $f_s = 220.5\,\mathrm{kHz}$, see Figure 8d.



(**a**) no aliasing mitigation, $f_s = 88.2\,\text{kHz}$

(**b**) output aliasing mitigation, $f_s = 88.2\,\text{kHz}$

(**c**) full aliasing mitigation, $f_s = 88.2\,\text{kHz}$

(**d**) no aliasing mitigation, $f_s = 220.5\,\text{kHz} = 5 \cdot 44.1\,\text{kHz}$

Figure 8. Output spectra of various model configurations for the tube screamer distortion circuit when excited with a single sinusoid of 1 V amplitude at 1244.5 Hz. Crosses mark expected harmonics.

6. Conclusions and Outlook

The presented approach for aliasing reduction generalizes the approach of [10] to all nonlinear systems that can be cast in a way that the nonlinearity takes scalar or very-low-dimensional input. This includes, but is not limited to, all models of circuits with a single one-port nonlinear element. If the system contains a delay-free loop, it has to be re-cast such that the nonlinearity is defined as the solution of the delay-free loop. Then, the delay introduced by applying the method of [10] to the nonlinearity can be compensated by adjusting the system's coefficients, even if the nonlinearity is part of a feedback loop.

As is to be expected, the achieved aliasing reduction is comparable to that of [10], allowing to significantly reduce the required oversampling especially for systems which introduce strong distortion, while the additional computational load is modest. Assuming lookup tables are used for $f(u)$ (in general being implicitly defined) and its antiderivative $F(u)$, the main price to pay is in terms of memory used.

It should be noted that the extensions to higher-order antiderivatives as proposed in [14] or [15] should be straightforward, following the same principle. A more interesting future direction would be to lift the restriction on the nonlinear function to have only scalar or very-low-dimensional input. If the method of [10] (or even the higher-order extensions of [14] or [15]) could be generalized to nonlinear functions with multiple inputs without requiring excessively large lookup tables, the method proposed in the present paper would immediately generalize to all stateful nonlinear systems.

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Yeh, D.T.; Abel, J.S.; Smith, J.O., III. Simplified, Physically-Informed Models of Distortion and Overdrive Guitar Effects Pedals. In Proceedings of the 10th International Conference on Digital Audio Effects (DAFx-07), Bordeaux, France, 10–15 September 2007.
2. Holters, M.; Dempwolf, K.; Zölzer, U. A Digital Emulation of the Boss SD-1 Super Overdrive Pedal based on Physical Modeling. In Proceedings of the 131st AES Convention, New York, NY, USA, 20–23 October 2011.
3. Dunkel, W.R.; Rest, M.; Werner, K.J.; Olsen, M.J.; Smith, J.O., III. The Fender Bassman 5F6-A Family of Preamplifier Circuits—A Wave Digital Filter Case Study. In Proceedings of the 19th International Conference on Digital Audio Effects (DAFx-16), Brno, Czech Republic, 5–9 September 2016; pp. 263–270.
4. Esqueda, F.; Pöntynen, H.; Välimäki, V.; Parker, J.D. Virtual Analog Buchla 259 Wavefolder. In Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17), Edinburgh, UK, 5–9 September 2017; pp. 192–199.
5. Thornburg, H. Antialiasing for nonlinearities: Acoustic modeling and synthesis applications. In Proceedings of the International Computer Music Conference, Beijing, China, 22–27 October 1999; pp. 66–69.
6. Esqueda, F.; Välimäki, V.; Bilbao, S. Aliasing Reduction in Soft-Clipping Algorithms. In Proceedings of the 23rd European Signal Processing Conference (EUSIPCO), Nice, France, 31 August–4 September 2015; pp. 2059–2063.
7. Esqueda, F.; Välimäki, V. Rounding corners with BLAMP. In Proceedings of the 19th International Conference on Digital Audio Effects (DAFx-16), Brno, Czech Republic, 5–9 September 2016; pp. 121–128.
8. Esqueda, F.; Välimäki, V.; Bilbao, S. Antialiased Soft Clipping using an Integrated Bandlimited Ramp. In Proceedings of the 24th European Signal Processing Conference (EUSIPCO), Budapest, Hungary, 29 August–2 September 2016; pp. 1043–1047.
9. Esqueda, F.; Bilbao, S.; Välimäki, V. Aliasing Reduction in Clipped Signals. *IEEE Trans. Signal Process.* **2016**, *64*, 5255–5267. [CrossRef]
10. Parker, J.D.; Zavalishin, V.; Bivic, E.L. Reducing the Aliasing of Nonlinear Waveshaping Using Continuous-Time Convolution. In Proceedings of the 19th International Conference on Digital Audio Effects (DAFx-16), Brno, Czech Republic, 5–9 September 2016; pp. 137–144.
11. Muller, R.; Hélie, T. Trajectory Anti-Aliasing on Guaranteed-Passive Simulation of Nonlinear Physical Systems. In Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17), Edinburgh, UK, 5–9 September 2017; pp. 87–94.
12. Holters, M. Antiderivative Antialiasing for Stateful Systems. In Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19), Birmingham, UK, 2–6 September 2019.
13. Holters, M.; Zölzer, U. A k-d tree based solution cache for the non-linear equation of circuit simulations. In Proceedings of the 24th European Signal Processing Conference (EUSIPCO), Budapest, Hungary, 29 August–2 September 2016; pp. 1028–1032.
14. Bilbao, S.; Esqueda, F.; Parker, J.D.; Välimäki, V. Antiderivative Antialiasing for Memoryless Nonlinearities. *IEEE Signal Process. Lett.* **2017**, *24*, 1049–1053. [CrossRef]
15. Bilbao, S.; Esqueda, F.; Välimäki, V. Antiderivative antialiasing, lagrange interpolation and spectral flatness. In Proceedings of the 2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA, 15–18 October 2017; pp. 141–145.

© 2019 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).