SIMPLIFYING ANTIDERIVATIVE ANTIALIASING WITH LOOKUP TABLE INTEGRATION

Leonardo Gabrielli

Dept. of Information Engineering
Università Politecnica delle Marche
Ancona, IT
l.gabrielli@staff.univpm.it

Stefano Squartini

Dept. of Information Engineering
Università Politecnica delle Marche
Ancona, IT
s.squartini@staff.univpm.it

ABSTRACT

Antiderivative Antialiasing (ADAA), has become a pivotal method for reducing aliasing when dealing with nonlinear function at audio rate. However, its implementation requires analytical computation of the antiderivative of the nonlinear function, which in practical cases can be challenging without a symbolic solver. Moreover, when the nonlinear function is given by measurements it must be approximated to get a symbolic description. In this paper, we propose a simple approach to ADAA for practical applications that employs numerical integration of lookup tables (LUTs) to approximate the antiderivative. This method eliminates the need for closed-form solutions, streamlining the ADAA implementation process in industrial applications. We analyze the trade-offs of this approach, highlighting its computational efficiency and ease of implementation while discussing the potential impact of numerical integration errors on aliasing performance. Experiments are conducted with static nonlinearities (tanh, a simple wavefolder and the Buchla 259 wavefolding circuit) and a stateful nonlinear system (the diode clipper).

1. INTRODUCTION

Traditionally, reducing aliasing in nonlinear digital circuits was done by means of oversampling, which has a toll on the computational cost and only improves linearly with the increase of the oversampling order. Antiderivative Antialiasing (ADAA) was introduced by Parker et al. in 2016 [1] as a novel means to deal with aliasing without recurring to oversampling, or to be used in conjunction with mild oversampling.

The original ADAA method involves converting a discrete-time input signal into a continuous-time representation using linear interpolation. The nonlinear function is then applied to this continuous signal. To prevent aliasing, the output is low-pass filtered by applying a rectangular or triangular continuous-time kernel before resampling to obtain the discrete-time output. This process effectively reduces aliasing at a lower cost with respect to oversampling, and in principle it can be extended to different filter kernels. In [2], the ADAA technique has been extended to IIR kernels and, stemming from this work, the technique has been adapted to wavetable synthesis in [3], also proving that the Differentiated Polynomial Waveform (DPW) [4] technique can be seen as a special case of ADAA apt to the generation of classical synthesizer waveforms.

The ADAA technique has been also adopted for stateful systems [5, 6] and wave-digital filters [7], improving the range of applications in audio processing and circuit simulation.

An early paper from Bilbao et al. [8], showed that a numerical framework akin to those used in finite difference time-domain simulations allows to obtain the same solution as in [1] and extend it to higher orders using finite-difference operators. The approach makes it simpler to obtain the numerical expression to update the system output for arbitrary order. However, a notable challenge remains: the computation of the antiderivative of the nonlinear function requires symbolic calculus, which can be cumbersome without appropriate tools.

To address this, we propose a simplified approach to ADAA that utilizes trivial numerical integration and lookup tables (LUTs) to approximate the antiderivative. By digging into the literature it seems that this approach, although simple, was only previously hinted in [8], where the use of LUTs was mentioned in the experimental section as a workaround to computing the antiderivatives.

In this paper, the approach is extensively explored to assess its validity in aliasing reduction. The method eliminates the need for closed-form solutions, streamlining the ADAA implementation process and broadening its accessibility to applications where symbolic computation is impractical. We analyze the trade-offs of this approach, highlighting its computational efficiency and ease of implementation while discussing the potential impact of numerical integration errors on aliasing performance. Experimental results demonstrate the feasibility with memoryless and stateful systems, providing valuable insights for practitioners seeking a viable and efficient antialiasing solution.

The remainder of the paper is organized as follows. Section 2 presents the proposed solution, detailing the background of ADAA, the challenges of analytical integration, and the introduction of the LUT-based method. Section 3 describes the experimental setup and evaluates the aliasing reduction performance, LUT size trade-offs, applicability to stateful nonlinearities, and computational efficiency. Section 4 summarizes the conclusions and discusses future research directions.

2. PROPOSED SOLUTION

2.1. Background

The ADAA method introduced in [1] consists of three main steps to compute the output of a nonlinear waveshaping system with reduced aliasing. The first one consists in approximating the conversion from discrete-time to continuous-time using piecewise linear interpolation between the discrete-time input values $x[n]$:

$$\tilde{x}(t) = \begin{cases} x_1 + \tau(x_0 - x_1), & 0 \leq t < 1, \\ x_2 + \tau(x_1 - x_2), & 1 \leq t < 2, \\ \vdots & \\ x_n + \tau(x_{n-1} - x_n), & (n-1) \leq t < n. \end{cases} \tag{1}$$

Then, a nonlinear function $f(t)$ is virtually applied in the continuous-time domain obtaining $y(t) = f(\tilde{x}(t))$. To return to the discrete-time domain, one is left with applying an antialiasing filter and computing the value of the output at the discrete intervals n. The output is, thus,

$$y[n] = \bar{y}(t) \mid_{t=nT} \tag{2}$$

where $\bar{y}(n)$ is obtained as the convolution between $y(t)$ and a lowpass filter kernel, which in [1] is chosen as a rectangular kernel $h_{rect}(t)$, which can be thought as the simplest lowpass filter choice. Their convolution leads to

$$y[n] = \int_{-\infty}^{\infty} h_{\text{rect}}(u) y(n-u)\, du \tag{3}$$

$$= \int_0^1 y(n-u)\, du \tag{4}$$

$$= \int_0^1 f(\tilde{x}(n-u))\, du \tag{5}$$

which ultimately leads to the result

$$y[n] = \frac{F_0(x_n) - F_0(x_{n-1})}{x_n - x_{n-1}}. \tag{6}$$

which can be computed at the original audio rate, given that the antiderivative $F_0(x) = \int_a^b f(x)dx$ is known at least in a plausible input range $[a, b]$ for the application at hand.

The only caveat to this method is the possibility of the denominator to get to zero or close to zero. In such a case, the output can be computed as

$$y[n] \simeq f(\frac{x_n + x_{n-1}}{2}). \tag{7}$$

Extension of the method to higher orders of the kernel can be computed. For example, using a triangular kernel (resulting in the convolution of two rectangular kernels), the output is computed as

$$y[n] = \frac{x_n(F_0(x_n) - F_0(x_{n-1})) - (F_1(x_n) - F_1(x_{n-1}))}{(x_n - x_{n-1})^2} + \frac{x_{n-2}(F_0(x_{n-2}) - F_0(x_{n-1})) - (F_1(x_{n-2}) - F_1(x_{n-1}))}{(x_{n-2} - x_{n-1})^2} \tag{8}$$

where $F_1(x) = \int F_0(x)dx$.

It must be noted that the first step (Eq. 1) results in a rough approximation of the continuous-time signal, which can impair the result. Therefore, as shown in [2], adopting a very steep lowpass antialiasing filter in the third step may not improve effectively the aliasing reduction performance.

2.2. Issues with ADAA

The described ADAA framework has several benefits with respect to oversampling: it uses lower phase delay filters, which can be an advantage when the nonlinearity is in a feedback loop; it has a better performance for the same computational cost. However, each time a new nonlinear function must be implemented, its antiderivative must be calculated analytically. Applying the method at scale may require longer development times than oversampling, which is more suitable to an object-oriented language and allow code reuse. Furthermore, the method cannot be employed when a nonlinearity has no closed-form expression, as is the case when a nonlinearity is extracted from data measurements or drawn by a user.

Furthermore, some analytical expressions are not easy to integrate manually, they may be piecewise (thus, multiple expressions must be calculated for the entire domain).

Finally, the $F_0(x)$ often has an increased computational cost with respect to $f(x)$. As an example, let us consider the following expression, which is typical of a number of problems in analog electronics

$$V_{out}[n] = \alpha(tanh(V_{in}[n]/\beta)). \tag{9}$$

To reduce the effect of aliasing, the nonlinearity can be treated using the ADAA method. The antiderivative is

$$F_0(x) = \alpha\beta log(cosh(x/\beta)). \tag{10}$$

This exhibits two transcendental functions in place of a single one. Furthermore, if the antialiasing performance must be improved, the second antiderivative must be employed, which is

$$F_1(x) = \frac{1}{2}(Li_2(-e^{-2x}) - x(x + 2log(e^{-2x} + 1) - 2log(cosh(x)))), \tag{11}$$

with $Li_n(x)$ being the polylogarithm function. This expression is definitely hard to derive manually and expensive to compute in real-time. In such a case, in typical engineering applications, some workarounds can be found, such as approximating the $tanh$ with a similar function (with a simpler antiderivative), or using numerical approximations, such as the Lambert W [9]. However, the proposed method largely simplifies the entire problem.

2.3. Proposed Method

The solution we propose, from now on referred to as ADAA-LUT, is simple, yet flexible. It consists in the numerical integration of $f(x)$ and in storing the obtained values in a LUT. This way, $F_n(x)$ is precomputed over a finely sampled range.

In detail, we propose sampling the function $f(x)$ at a finite number of points K, equally spaced by Δx to get $\hat{f}$. This can be either obtained by physical measurement (e.g. of a circuit) or by direct computation of a closed-form expression at a number of meaningful input values. Suppose that a function is sampled using K points in the range $[a, b]$, leading to a LUT $\hat{f}$. From this LUT, numerical integration can be performed according to the following, to obtain the first antiderivative:

$$\hat{F}_0(k) = \rho + \sum_{k=0}^{K} \hat{f}(a + k\Delta x) \cdot \Delta x. \tag{12}$$

where the scalar ρ is the integration constant. In general, this can be suitably chosen, e.g. to get $F_0(x = 0) = 0$, however, it is

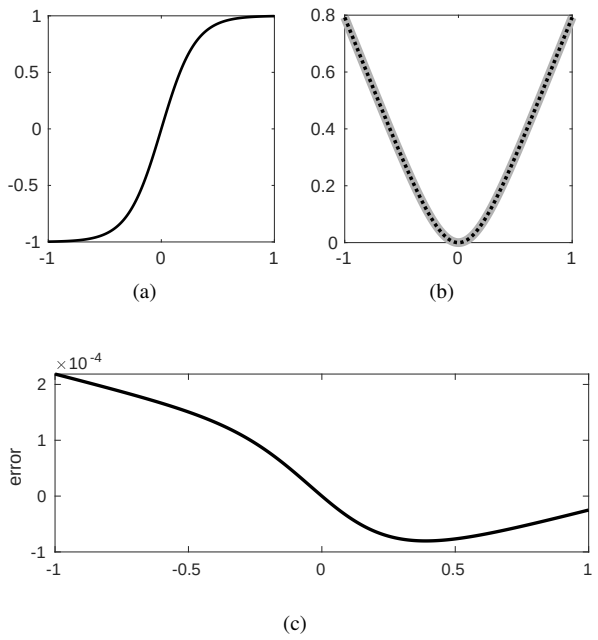


Figure 1: Input-output relationship of the hyperbolic tangent with $\alpha = 1, \beta = 0.3$ (a), its antiderivatives (b) and numerical error (c). The error in (c) is the difference between the symbolic antiderivative (plot (b), solid gray line) and the numerical antiderivatide (plot (b), dotten black line).

canceled out when used in Eq. (6) or Eq. (8), therefore its choice is not highly relevant and will not be discussed further. By applying again the numerical integration we can also derive the LUT $\hat{F}_1$ for the second antiderivative to be used in Eq. (8).

The LUT of the antiderivative can be now employed to compute Eq. (6), using any form of interpolated read, the simplest one being linear interpolation, i.e.

$$\hat{F}_0(x_n) = \hat{F}_0(i) + d(\hat{F}_0(i+1) - \hat{F}_0(i)) \tag{13}$$

with the reading index being $i = \lfloor (K/2 \cdot x_n) + K/2 \rfloor$ and the fractional part being $d = (K/2 \cdot x_n) + K/2 - i$.

More advanced forms of interpolation can be used as well, such as cubic spline interpolation, which provides a smoother estimate by considering N surrounding points. In this case, the LUT value is obtained as

$$\hat{F}_0(x_n) = \sum_{j=0}^{N-1} S_j(x_n) \hat{F}_0(i + j - N/2) \tag{14}$$

where $S_j(x_n)$ are the cubic basis functions that depend on the fractional position d.

One advantage of the approach is that it eliminates the need for symbolic computation, simplifying the implementation and allowing for faster implementation of several nonlinear functions, since the algorithm stays the same. It is easy to implement this efficiently in, e.g. C++ code, by reading the LUT from a file and thus allowing code reuse. The computational efficiency is also extremely improved, since no transcendental function must be computed. Linear interpolation is fast to compute, while other improved interpolation methods may be slower, but still quite easy to handle and sufficiently fast for real-time applications.

Another advantage of this method is that it does not require closed-form nonlinear function but can employ any memoryless input-output relationship obtained from physical measurements.

3. EXPERIMENTS

To show the feasibility of the approach we conducted numerical experiments in Matlab ($F_s = 44100$ Hz). Four use cases were considered:

- the static saturating $tanh$ nonlinearity of Eq. (9),
- a static piecewise asymmetrical wavefolding nonlinearity,
- a model of the Buchla 259 wavefolder, as described in [10], with a heavy distortion
- a diode clipper [11], which is a nonlinear circuit with memory.

Details about these nonlinear systems follow. The saturating nonlinearity (Eq. 9) has been sampled using values $\alpha = 1, \beta = 0.3$, resulting in the shape of Figure 1(a).

The wavefolding nonlinear function is the following

$$f(x) = \begin{cases} 2\tau - x & x > \tau, \\ x & -\tau \leq x \leq \tau, \\ -2\tau - x & x < -\tau, \end{cases} \tag{15}$$

with $\tau = 0.7$. Since the function is odd, in all our experiments, a bias term $b = 0.1$ is added to the signal before the nonlinear function to allow even harmonics at the output.

The Buchla 259 wavefolder has been implemented according to Eqns. (13-18) from [10], therefore it does not account for the aliasing reduction techniques proposed in the cited paper, leaving alias reduction to the ADAA method proposed in our paper.

The diode clipper has been often taken as a use case in virtual analog research and has been studied in [11]. In our implementation we take the implementation proposed in [9], with the nonlinear function being the exact Wright Omega function (i.e. none of the approximations proposed in the cited paper are considered).

The first experiments have been conducted on the static nonlinearities to study the antialiasing properties of the ADAA-LUT method compared to the original analytic ADAA method (from now on referred to as ADAA-ORIG). The size of the LUT has been varied to show its impact on the aliasing by measuring the signal to noise ratio (SNR). Among these tests, the Buchla 259 circuit stands out for its extremely nonlinear characteristic, which has been taken as a testbench to validate the proposed method under heavy aliasing conditions. For this reason, a third order ADAA method will also be employed with this circuit.

Finally, tests on the diode clipper circuit show that, as expected, the method is also suitable to stateful nonlinear systems.

Computational load has been tested with a C++ implementation of the ADAA-ORIG and ADAA-LUT method to assess the computational cost of the different solutions.

Please note that although some of the nonlinearities employed in these tests are antisymmetric, we did not exploit this property to halve the LUT size, in order to keep the discussion general.

3.1. Aliasing reduction performance

As done in previous works, the aliasing reduction performance of the algorithms is highlighted by spectrogram plots showing the

output of the system with a swept sine input. The spectrograms are shown in Figure 2 and compare the trivial method (i.e. simple evaluation of the $tanh$ or the wavefolder nonlinear function at sampling rate), with the ADAA-ORIG and ADAA-LUT. Specifically, two variations of the ADAA-LUT method are shown, one with $K = 8192$, one with $K = 1024$. All spectrograms exhibit aliasing to some extent, with the trivial methods obviously showing it more severely. From visual inspection, all the ADAA methods look exactly the same, hinting at the fact that they will sound identical to the human ear.

To investigate further this issue, we repeated these experiments with a fixed frequency sine, and measured the SNR at the output of the nonlinearity. As an example, Figure 3 shows spectra from several experiments. The fundamental and the higher harmonics are indexed. All the remaining components, i.e. the aliasing, are treated as noise. The SNR is computed excluding the harmonics and the DC component.

As can be seen, there is no noticeable difference between the ADAA-ORIG, ADAA-LUT 8192 and ADAA-LUT 1024. This is reflected in the SNR which is almost identical between all these methods. The ADAA-LUT 128 case exhibits some noise, coming from the quantization of the LUT. However, this is lower than the aliasing components, and is kept below -100 dB which makes it unnoticeable.

From these tests we can conclude that, as long as the first order ADAA method is employed, all options are almost identical, with only the smaller LUT ($K = 128$) exhibiting some quantization noise.

3.2. Higher-order ADAA and Interpolation

In order to push the limits of the method we employed the 3rd order ADAA method from [8] (see Eq. (16) from the paper). The recursion is easy to implement, while the computation of the third-order antiderivative can be carried out numerically as proposed earlier. This makes it effortless to compute complex nonlinear effect processors such as the Buchla 259 wavefolder. In our experiments, we computed the input-output LUT by sweeping linearly an input voltage from -10V to 10V. The obtained nonlinear function has been integrated numerically three times to compute the antialiased output (with integration constant $\rho = 0$). An input sine wave with range -8V:8V and bias 2V has been employed to drive the system. The resulting output has been studied for the trivial case, the first-order ADAA method and the third-order ADAA method using 8192 LUT points and linear interpolation. Additionally, a 5-point cubic spline interpolation has been tested with the third-order ADAA. Results are shown in Figure 4. As can be seen, aliasing components are reduced by a large amount by increasing the ADAA order. However, the quantization error due to linear interpolation grows. This can be reduced by introducing the cubic spline method, that reduces the noise floor and leaves the harmonics unaffected. Please note that despite the significant improvement in perceived quality, the SNR value does not decrease substantially because a few high-amplitude spurious components remain. These dominant residuals continue to limit the overall SNR, making the effect of the reduction on this metric relatively small.

3.3. Discussion on LUT size

The tests show that the proposed method is suitable to replace the original ADAA method without any loss of quality, except when the LUT is considerably small or the ADAA order is high. In such cases, the quantization noise may become noticeable, which can be overcome by using advanced interpolation methods or larger LUTs.

In most cases, LUT size will not be a issue (except under severe hardware constraints). Indeed, the algorithms discussed in this field are most probably implemented on laptop computers, mobile devices or embedded digital signal processors. The latter is the most challenging case, since computational and memory resources are more limited than the previous two.

Within embedded digital signal processors, memory buses create bottlenecks, therefore any data that is repeatedly accessed at random positions should be allocated in arrays as close as possible to the main processing unit. However, such low-level *cache* memories are limited and must be used efficiently. For example a current ADSP-SC592 signal processor[1], has only 640KB L1 cache memory. However, even the largest LUT considered in our tests takes slightly more than 1% of the available cache size, while the smaller ones are takes a negligible space in the L1 cache.

If memory is critical, a small LUT can be employed, at the expense of a higher computational cost required to implement a higher complexity interpolation method.

3.4. Stateful nonlinearities

The ADAA method has been successfully employed with stateful nonlinear systems, therefore it is sound to expect that the ADAA-LUT method will perform similarly. For the sake of completeness, we verified that the proposed method does not alter the behavior of a stateful system such as a diode clipper. Figure 5 shows spectrograms of the diode clipper output with a swept sine input. The trivial approach consists in computing the exact solution of the diode characteristic using the Lambert W function as done in [9]. This solution exhibits aliasing. The ADAA method can be fruitfully employed by either computing the antiderivative in closed form or in numerical form, with identical results. Figure 6 shows the behavior of the output in the time domain. Some portions of the swept sine are highlighted. The ADAA method shows some slight differences from the non-antialiased method at high frequency due to the filtering involved in the continuous-time domain. However, there is no noticeable difference between the two ADAA implementations. Indeed the error between the two is of the order of 10^{-4} or lower.

3.5. Computational Cost

Assessing the computational cost of the ADAA-LUT with respect to the original ADAA method is not trivial, since transcendental functions in modern C libraries are usually implemented using several approximation methods based on the input value and the floating point precision. These may include linear, Pade and Taylor approximations, lookup tables and more. On the other hand, the proposed approach requires a handful of operations, based on the interpolation method. For this reason, we conducted an experimental test to benchmark by implementing the two methods in C, to provide a hint of the computational savings.

The benchmarks involved computing the ADAA output using the antiderivative of Eq. 10 and comparing it to the ADAA output obtained using the ADAA-LUT method, with a precomputed

[1] https://www.analog.com/en/products/adsp-sc592.html

lookup table of 8192 points. Tests were conducted on an Intel-based laptop (13th Gen Core i7-1360P) using a single-threaded application, with calculations performed in double-precision floating-point arithmetic. We used the GNU C/C++ `math.h`[2] mathematical libraries available with the GNU compiler (gcc/g++) version 11.4.0 available with Ubuntu 22.04. We compiled a simple C application running from commandline with no graphical frontend using either no compiler optimization (flag `-O0`) and maximum optimization (flag `-O3`). The application processes a batch of 44,100 samples for each test, with either methods. We repeated this test 20 times and averaged the elapsed times obtained from the application to get a stable estimate of the execution time. The Linux kernel system timer was `CLOCK_MONOTONIC`.

Optimization	ADAA	ADAA-LUT
-O0	571 µs	227 µs
-O3	495 µs	162 µs

Table 1: Benchmark results for analytical AADA vs AADA-LUT with different compiler optimization.

Results are shown in Table 1. The AADA-LUT method significantly reduced computation time by 60.8% reduction in time compared to the analytical method without optimization. With maximum optimization, the gain increases to 67%, meaning that the optimizer was able to improve the ADAA-LUT algorithm better than the analytic ADAA method.

4. CONCLUSIONS

In this paper, we proposed a novel approach to Antiderivative Antialiasing (ADAA) that employs numerical integration of lookup tables (LUTs) to approximate the antiderivative, effectively eliminating the need for closed-form solutions. This method, referred to as ADAA-LUT, significantly simplifies the implementation process, making it more accessible for practical applications, particularly in industrial and embedded systems where computational efficiency is a priority. By removing the need for analytical integration it paves the way for faster time-to-market in industrial applications.

Through extensive numerical experiments, we demonstrated that the ADAA-LUT method achieves aliasing reduction performance identical to the original analytical ADAA method while offering significant computational savings. Specifically, the proposed method reduced computation time by up to 67% in our benchmarks, making it a viable alternative for real-time digital signal processing applications. The use of LUTs enables flexible and scalable implementation, supporting arbitrary nonlinear functions, including those derived from empirical measurements, thus broadening the scope of ADAA applications.

Our results also highlighted the trade-offs associated with LUT size. While large LUTs (e.g., 8192 or 1024 points) provided results indistinguishable from the analytical approach, smaller LUTs introduced minor quantization noise, which can be mitigated with higher-order interpolation techniques. Additionally, LUT size can be reduced by leveraging symmetries in the nonlinear function. However, given the relatively low memory footprint of even the largest LUTs, this trade-off is unlikely to pose a significant limitation in modern processing environments.

[2]https://www.gnu.org/software/libc/manual/html_node/Mathematics.html

We also pushed the method further by testing it on a highly nonlinear function, the Buchla 259 wavefolding circuit, which introduces stronger aliasing. We employed a third order ADAA-LUT method with 5-point cubic spline interpolation to reduce numerical errors and, hence, the noise floor.

Finally, we showed the applicability of the ADAA-LUT method to stateful nonlinearities, such as the diode clipping circuit, confirming that the method preserves the advantages of ADAA without introducing unintended artifacts.

5. REFERENCES

[1] Julian D Parker, Vadim Zavalishin, and Efflam Le Bivic, "Reducing the aliasing of nonlinear waveshaping using continuous-time convolution," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16), Brno, Czech Republic*, 2016, pp. 137–144.

[2] Pier Paolo La Pastina, Stefano D'Angelo, and Leonardo Gabrielli, "Arbitrary-order IIR antiderivative antialiasing," in *2021 24th International Conference on Digital Audio Effects (DAFx)*. IEEE, 2021, pp. 9–16.

[3] Leonardo Gabrielli, Stefano D'Angelo, Pier Paolo La Pastina, and Stefano Squartini, "Antiderivative antialiasing for arbitrary waveform generation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2743–2753, 2022.

[4] Vesa Valimaki, Juhan Nam, Julius O Smith, and Jonathan 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, 2009.

[5] Pier Paolo La Pastina and S D'Angelo, "Antiderivative antialiasing with frequency compensation for stateful systems," in *Proc. Int. Conf. Digital Audio Effects*, 2022, pp. 40–47.

[6] Martin Holters, "Antiderivative antialiasing for stateful systems," *Applied Sciences*, vol. 10, no. 1, pp. 20, 2019.

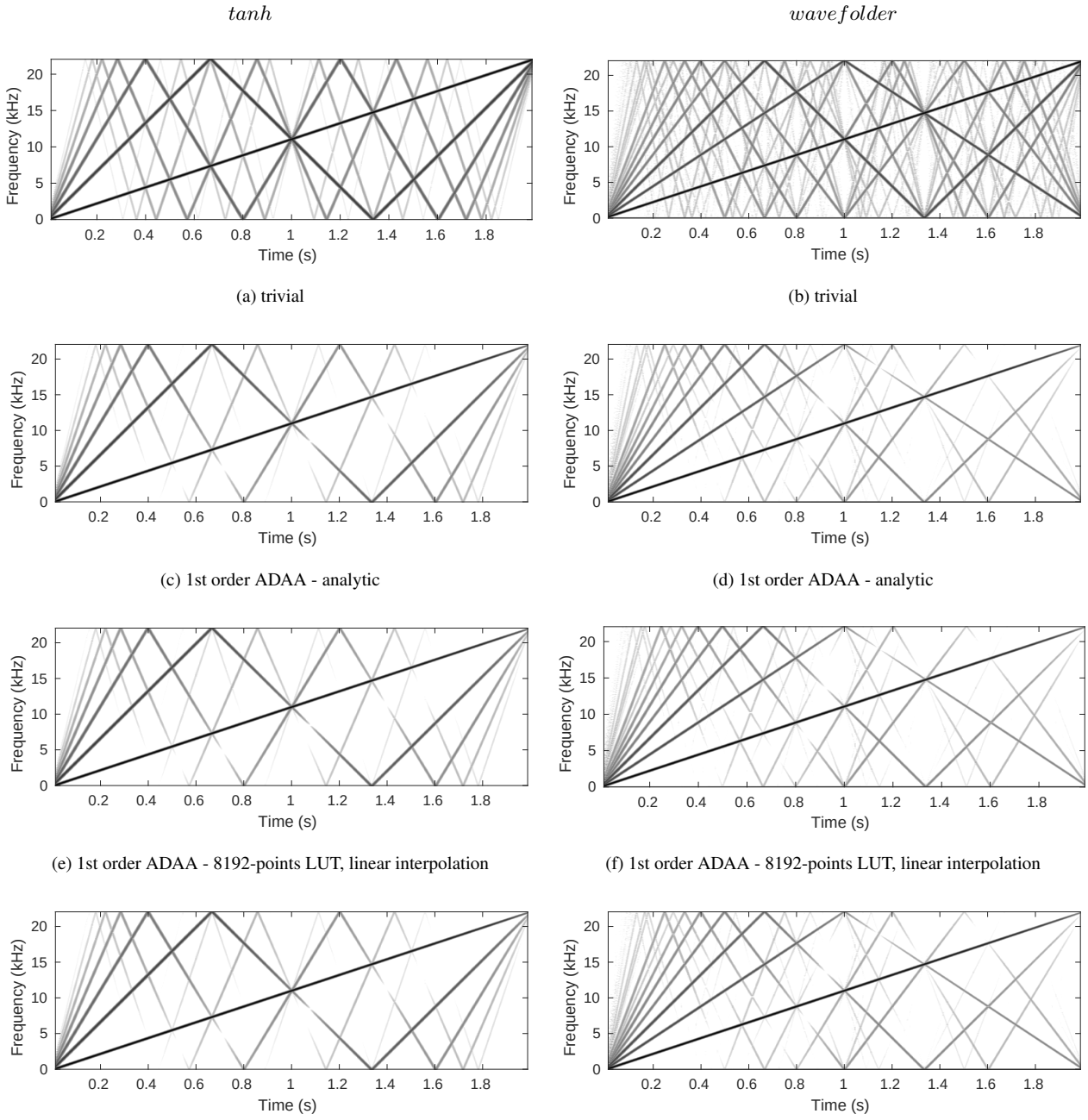
[7] Davide Albertini, Alberto Bernardini, Augusto Sarti, et al., "Antiderivative antialiasing techniques in nonlinear wave digital structures," in *Journal of the Audio Engineering Society*, 2021, vol. 69, pp. 448–464.

[8] Stefan Bilbao, Fabián Esqueda, Julian D Parker, and Vesa Välimäki, "Antiderivative antialiasing for memoryless nonlinearities," *IEEE Signal Processing Letters*, vol. 24, no. 7, pp. 1049–1053, 2017.

[9] Stefano D'Angelo, Leonardo Gabrielli, Luca Turchet, et al., "Fast approximation of the Lambert W function for virtual analog modelling.," in *Proceedings of Digital Audio Effects Conference*. Birmingham City University, 2019, pp. 238–244.

[10] Fabián Esqueda, Henri Pöntynen, Vesa Välimäki, and Julian D Parker, "Virtual analog Buchla 259 wavefolder," in *International Conference on Digital Audio Effects*. University of Edinburgh, 2017, pp. 192–199.

[11] D. T. Yeh, J. Abel, and J. O. Smith, "Simulation of the diode limiter in guitar distortion circuits by numerical solution of ordinary differential equations," in *Proceedings of the International Conference on Digital Audio Effects*, 2007, pp. 197–204.



(a) trivial

(b) trivial

(c) 1st order ADAA - analytic

(d) 1st order ADAA - analytic

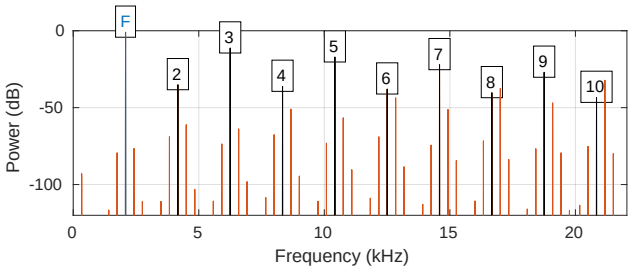
(e) 1st order ADAA - 8192-points LUT, linear interpolation

(f) 1st order ADAA - 8192-points LUT, linear interpolation

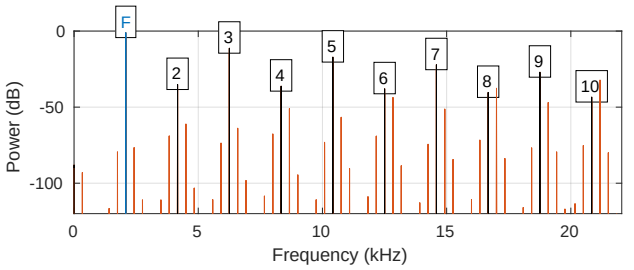
(g) 1st order ADAA - 1024-points LUT, linear interpolation

(h) 1st order ADAA - 1024-points LUT, linear interpolation

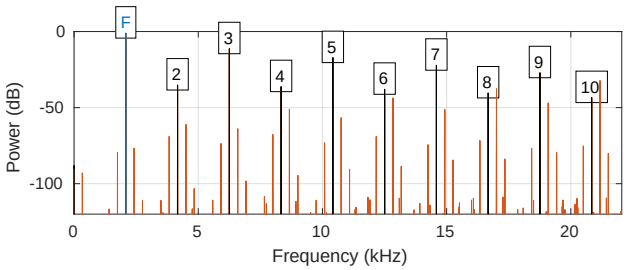
Figure 2: Spectrograms showing the output of the discussed nonlinearities using a swept sine input. Cases (a), (c), (e), (g): trivial $tanh$, 1st order AADA (analytic), 1st order AADA (8192- and 1024-points LUT with linear interpolation). Cases (b), (d), (f), (h): trivial wavefolder, 1st order AADA wavefolder (analytic), 1st order AADA wavefolder (8192- and 1024-points LUT with linear interpolation). The minimum spectrogram threshold is -90dB.



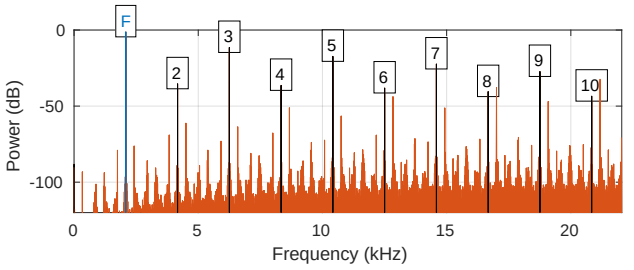
(a) 1st order ADAA - analytic (SNR: 29.39 dB)



(b) 1st order ADAA - 8192-points LUT (SNR: 29.39 dB)

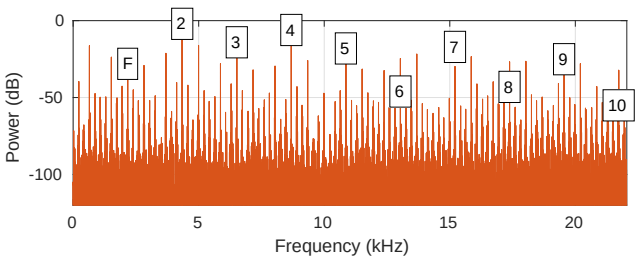


(c) 1st order ADAA - 1024-points LUT (SNR: 29.39 dB)

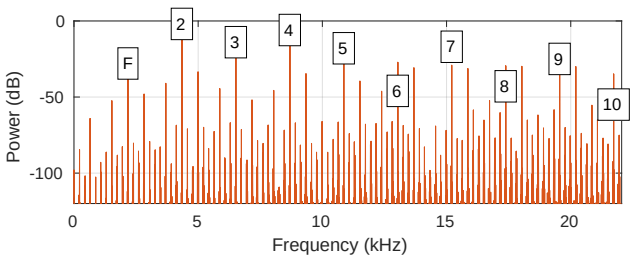


(d) 1st order ADAA - 128-points LUT (SNR: 29.36 dB)

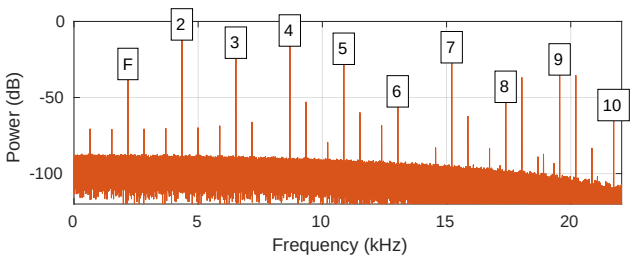
Figure 3: Spectra showing the aliasing components (highlighted in orange) for a sine input at 2093 Hz (C7 of a piano) with amplitude 3 and bias 0.5, being processed by the $tanh$ nonlinear function. The SNR for each case is shown. The fundamental is denoted as 'F', while the harmonics below $F_s/2$ are denoted by their number. For reference: SNR for the trivial method is 23.64 dB.



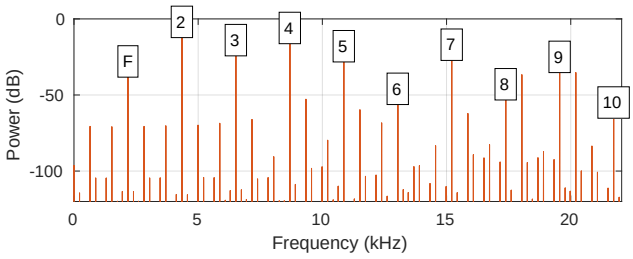
(a) Trivial (SNR: 0.14 dB)



(b) 1st order ADAA - 8192-points LUT (SNR: 6.94 dB)

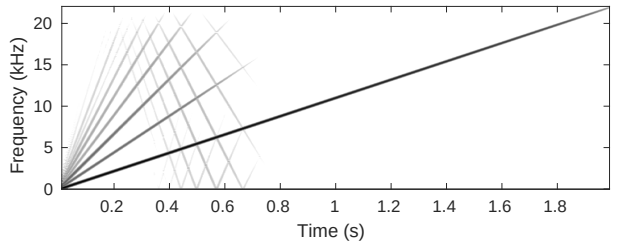


(c) 3rd order ADAA - linear interpolation - 8192-points LUT (SNR: 8.95 dB)

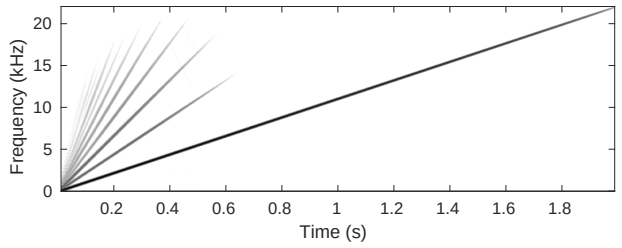


(d) 3rd order ADAA - spline interpolation - 8192-points LUT (SNR: 8.96 dB)

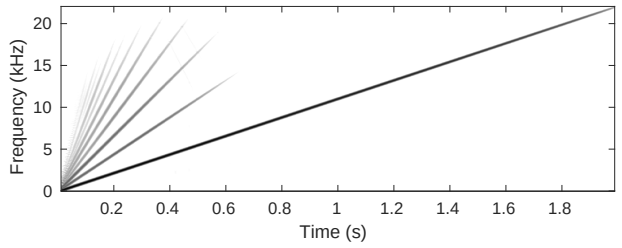
Figure 4: Buchla 259 wavefolder: spectra showing the aliasing components (highlighted in orange) for a sine input at 2093 Hz (C7 of a piano) with amplitude 8V and bias 0.1V. The fundamental is denoted as 'F', while the harmonics below $F_s/2$ are denoted by their number.



(a)

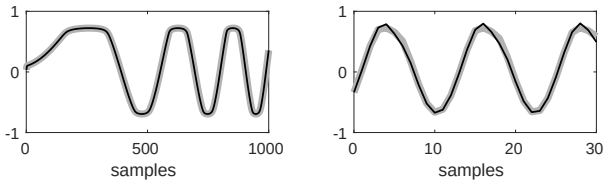


(b)

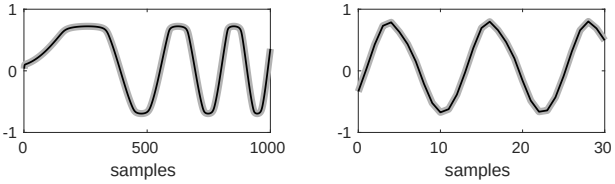


(c)

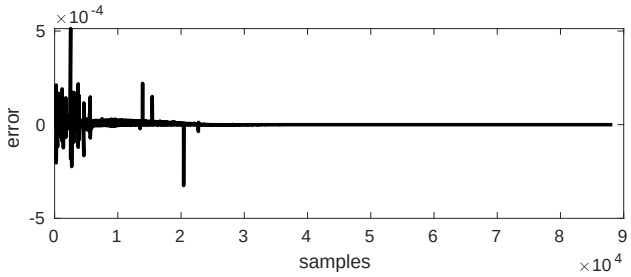
Figure 5: Spectrogram showing the output of the diode-clipper circuit using a swept sine input. Case (a) shows the exact Lambert W solution with no attempt at aliasing reduction. Case (b) shows the simulation using the AADA-ORIG method. Case (c) shows the simulation using the ADAA-LUT method. The minimum spectrogram threshold is -90dB.



(a) Exact solution (grey) vs. ADAA-LUT (black).



(b) ADAA-LUT (grey) vs. ADAA-ORIG (black).



(c) Error between ADAA-LUT and ADAA-ORIG along the full sweep.

Figure 6: Details from the time-domain signals of Figure 5. On the left, the beginning of the log sweep is shown, on the right a portion of signal around 4 kHz.