ARBITRARY-ORDER IIR ANTIDERIVATIVE ANTIALIASING

Pier Paolo La Pastina
Orastron Srl
Sessa Cilento, Italy
pierpaolo.lapastina@orastron.com

Stefano D'Angelo
Independent Researcher
Sessa Cilento, Italy
s@dangelo.audio

Leonardo Gabrielli
Department of Information Engineering,
Università Politecnica delle Marche
Ancona, Italy
l.gabrielli@staff.univpm.it

ABSTRACT

Nonlinear digital circuits and waveshaping are active areas of study, specifically for what concerns numerical and aliasing issues. In the past, an effective method was proposed to discretize nonlinear static functions with reduced aliasing based on the antiderivative of the nonlinear function. Such a method is based on the continuous-time convolution with an FIR antialiasing filter kernel, such as a rectangular kernel. These kernels, however, are far from optimal for the reduction of aliasing. In this paper we introduce the use of arbitrary IIR rational transfer functions that allow a closer approximation of the ideal antialiasing filter, required in the fictitious continuous-time domain before sampling the nonlinear function output. These allow a higher degree of aliasing reduction and can be flexibly adjusted to balance performance and computational cost.

1. INTRODUCTION

The design of virtual analog filters, oscillators and nonlinear components is an active area of research, involving expertise in diverse areas including analog electronics, digital signal processing and numerical analysis. Nowadays, methods to design band-limited oscillators [1] and linear filters are well established [2]. Current trends in this research field are towards implementing stable time-varying structures [3] and accurate nonlinear devices [4, 5].

Nonlinear devices are generally known to expand the bandwidth of an input signal and may, thus, lead to aliasing. Historically, the only method that was known in this regard relies on increasing the discrete-time signal sampling rate, i.e. *oversampling*. This method is rather straightforward to implement and is quite effective for large oversampling factors, however, it can lead to a remarkable increase in computational cost. A few years ago, a novel solution to this issue was found by Parker et al. in [6]. This method, generally named *antiderivative antialiasing* (AA), is based on the conversion from discrete-time to continuous-time, the application of the nonlinear function, and the conversion of the new signal back to discrete-time domain. This whole process would not be computationally feasible without applying some simplifying hypotheses, that are: *(a)* the conversion from discrete- to continuous-time using linear interpolation, *(b)* the conversion from continuous- to discrete-time using a finite-support kernel, such as a rectangular kernel.

Extensions to the AA method have been introduced by [7], with a state-space formulation that makes the method suitable to nonlinearities with memory, and [8] which applies the concept to wave-digital filters. The paper in [9] formulates the method in terms of numerical operators, therefore extending the method to higher order of the nonlinearity antiderivative.

The effectiveness of the AA method depends on the upsampling filter, for converting from discrete-time to continuous-time and the anti-aliasing filter, used to return to discrete-time. Current AA techniques employ simple linear interpolation as the anti-imaging filter. Upsampling filters have been discussed in [10], showing that, although far from optimal, linear interpolation is sufficient for many applications, while the spectral rolloff of the antialiasing filter has a larger impact in aliasing reduction. For this reason we will concentrate our efforts on the design of the antialiasing filter and leave some additional details in the Appendix.

As for the antialiasing filters, in [6] the method describes a rectangular and a triangular kernel, but in general, a closed-form solution can be generalized for any piecewise polynomial kernel. Any such filter will have finite support and therefore far from the ideal antialiasing filter (i.e. the sinc function). Another disadvantage of extending the filter order is that the expressions for computing the reduced-aliasing output get more complex with increasing order and ill-conditioning problems may become a concern.

Infinite impulse response (IIR) filters allow for a steeper rolloff at a low computational cost and many well known design methods exist. This makes them interesting candidates as antialiasing filters in the AA method provided that a method can be derived for their application. In this work we extend the antiderivative antialiasing method to IIR filter kernels. The only hypothesis is that the filter transfer function must be rational, which anyway is the standard way of dealing with filters in digital signal processing. We provide a closed form solution for several special cases and show the superior antialiasing performance of the method. The original method will be indicated as AA-FIR in contrast to the proposed AA-IIR method.

The paper is organized as follows. Section 2 summarizes the AA-FIR method, while Section 3 introduces the proposed AA-IIR method, deriving a formulation for all filter pole types, and finally reporting an overview of the method. Experiments are provided in Section 4 showing the performance of the method and discussing the computational cost. Finally, conclusions are drawn in Section 5. We provide some more details in the appendices. Appendix A discusses the effect of linear interpolation for the upsampling, which is common to other techniques as well. Appendix B relates the method to the impulse invariance transform by studying the case of a linear function. Appendix C reformulates the proposed method for a practical implementation where complex numbers are avoided.

2. ANTIDERIVATIVE ANTIALIASING WITH FINITE SUPPORT KERNELS

In this section we will briefly introduce the formalism used in the original AA-FIR method. Let f be a (possibly) nonlinear function, and x_n a discrete-time input signal to be processed by the nonlinear function. The AA-FIR method is based on a (fictitious) conversion from discrete-time to continuous-time using linear interpolation as follows

$$\tilde{x}(t) = x_n + (t-n)(x_{n+1} - x_n) \tag{1}$$

with $n \leq t \leq n+1$. The continuous-time signal $\tilde{x}$ is now fed to the nonlinear function, generating $\tilde{y}(t) = f(\tilde{x}(t))$. The nonlinear output $\tilde{y}$ is generally not bandlimited, therefore an antialiasing filter must be applied in the continuous-time domain before converting the signal back to the discrete-time domain. The simplest lowpass filter that can be applied to solve the problem analytically is a unitary rectangular kernel defined for $0 \leq t \leq 1$ (assuming $F_s = 1$). Such a kernel allows to solve the continuous-time integral explicitly as follows

$$\begin{aligned} y_n &= \int_{-\infty}^{+\infty} h(t) f(\tilde{x}(n-t))dt \\ &= \int_0^1 f(\tilde{x}(n-t))dt \\ &= \int_0^1 f(x_{n-1} + (1-t)(x_n - x_{n-1}))dt. \end{aligned} \tag{2}$$

By applying the following substitution $\xi = x_{n-1} + (1-t)(x_n - x_{n-1})$, the integral becomes

$$\begin{aligned} y_n &= \int_{x_{n-1}}^{x_n} \frac{f(\xi)}{x_n - x_{n-1}} d\xi \\ &= \frac{1}{x_n - x_{n-1}} \int_{x_{n-1}}^{x_n} f(\xi) d\xi \end{aligned} \tag{3}$$

From there, the solution is easy to obtain and is directly expressed in terms of the discrete-time input signal

$$y_n = \frac{F_1(x_n) - F_1(x_{n-1})}{x_n - x_{n-1}}, \tag{4}$$

where F_1 is the first antiderivative of f. This is where the term antiderivative antialiasing stems. As can be seen, the transition to continuous-time is fictitious. Please notice that, from now on, the mean integral will be simply denoted as

$$\frac{1}{b-a}\int_a^b dx = \fint_a^b dx$$

The process can be extended to higher orders, by integrating the kernel and thus obtaining, e.g. a triangular kernel or other piecewise polynomial kernels, as shown originally in [6]. Increasing the order of the kernel improves antialiasing performance, however, it is prone to more severe ill-conditioning, which in turn may require a higher numerical precision. Furthermore, the computational cost increases. The reason for this increase is twofold: *(a)* increasing the order of the AA-FIR method increases the number of past input samples for FIR filtering, thus the expression becomes more complex; *(b)* higher order antiderivatives are required, which for transcendental functions may become very expensive. In these cases, it is suggested to reduce the computational cost by adopting look-up tables.

3. ANTIDERIVATIVE ANTIALIASING WITH IIR KERNELS

The proposed method takes pace from the AA-FIR, however it formulates the solution allowing for antialiasing IIR rational transfer functions of arbitrary order and design, including the most common ones, such as Butterworth, Chebyshev or Elliptical. We will consider only low-pass causal kernels for the antialiasing filter. Under the hypothesis of a causal kernel, the support of h is $[0, +\infty)$ and we assume the same for $\tilde{x}$. We also assume $f(0) = 0$ for convenience[1]. Under these hypotheses, the first integral of Eq. (2) reduces to

$$y_n = \int_0^n h(t) f(\tilde{x}(n-t))dt. \tag{5}$$

To calculate this, we can divide the integration interval into n intervals of unitary length and apply linear interpolation from Eq (1)

$$\begin{aligned} y_n &= \sum_{k=0}^{n-1} \int_k^{k+1} h(t) f(\tilde{x}(n-t))dt \\ &= \sum_{k=0}^{n-1} \int_k^{k+1} h(t) f(x_{n-k-1} + (k+1-t)(x_{n-k} - x_{n-k-1}))dt. \end{aligned}$$

Similarly to what was done in the AA-FIR case we can apply the substitution $\xi = x_{n-k-1} + (k+1-t)(x_{n-k} - x_{n-k-1})$. This leads to

$$y_n = \sum_{k=0}^{n-1} \fint_{x_{n-k-1}}^{x_{n-k}} f(\xi) h\left(k + 1 - \frac{\xi - x_{n-k-1}}{x_{n-k} - x_{n-k-1}}\right) d\xi$$

or, equivalently,

$$y_n = \sum_{k=0}^{n-1} \fint_{x_k}^{x_{k+1}} f(\xi) h\left(n - k - \frac{\xi - x_k}{x_{k+1} - x_k}\right) d\xi. \tag{6}$$

In the general case, this expression cannot be further simplified. The expression has three unknowns and is not amenable, in the general case, to a recursive implementation. More specifically, y_{n+1} can be calculated recursively only if $h\left(n+1-k-\frac{\xi - x_k}{x_{k+1} - x_k}\right)$ can be expressed in terms of $h\left(n-k-\frac{\xi - x_k}{x_{k+1} - x_k}\right)$. To overcome these issues, we need to impose further constraints.

Filters, in signal processing, are usually expressed as rational functions of the Laplace or the Z variable. We can, thus, express the desired filter kernel as a rational function in the Laplace domain, and derive more appropriate expressions for real-time implementation of the AA-IIR method. This will lead to tractable expressions that lend themselves to real-time use.

3.1. Rational Function of Real Poles

Let us now consider a rational transfer function $H(s) = \frac{F(s)}{G(s)}$, where F and G are real-valued polynomials of order q and p, respectively and $q \leq p$. Denominator G can be decomposed as the

[1] If this does not hold true, it is possible to define a function $f^\star(x) = f(x) - f(0)$, apply the following reasoning to $f^\star(x)$ and finally add constant term $f(0)$ in the end

product

$$G(s) = A(s-\alpha_1)^{m_1}\dots(s-\alpha_p)^{m_p}(s-\beta_1)^{\mu_1}(s-\overline{\beta}_1)^{\mu_1}\dots \\ \dots(s-\beta_q)^{\mu_q}(s-\overline{\beta}_q)^{\mu_q}, \quad (7)$$

where $\alpha_1,\dots,\alpha_p$ are the real poles of H, $\beta_1,\overline{\beta}_1,\dots,\beta_q,\overline{\beta}_q$ are the complex poles and $m_1,\dots,m_p,\mu_1,\dots,\mu_q$ the multiplicities of $\alpha_1,\dots,\alpha_p,\beta_1,\dots,\beta_q$, respectively. It is useful to recall that the condition for stability is $\alpha,\dots,\alpha_p$, $\Re(\beta_1),\dots,\Re(\beta_q) < 0$. The application of a partial fraction decomposition to $H(s)$ results in the following

$$\begin{aligned} H(s) = A_0 &+ \frac{A_{11}}{s-\alpha_1} + \frac{A_{12}}{(s-\alpha_1)^2} + \dots \frac{A_{1m_1}}{(s-\alpha_1)^{m_1}} \\ &+ \frac{A_{p1}}{(s-\alpha_2)} + \frac{A_{p2}}{(s-\alpha_2)^2} + \dots \frac{A_{pm_p}}{(s-\alpha_2)^{m_p}} + \dots \\ &+ \frac{B_{11}}{s-\beta_1} + \frac{B_{12}}{(s-\beta_1)^2} + \dots + \frac{B_{1\mu_1}}{(s-\beta_1)^{\mu_1}} \\ &+ \frac{\overline{B}_{11}}{s-\overline{\beta}_1} + \frac{\overline{B}_{12}}{(s-\overline{\beta}_1)^2} + \dots + \frac{\overline{B}_{1\mu_1}}{(s-\overline{\beta}_1)^{\mu_1}} + \dots \end{aligned} \quad (8)$$

where the terms A_{ij}, B_{kl} are the residues. Eq. 8 informs us that each term can be treated separately and then summed to the others. Specifically, for any λ either real or complex, the inverse Laplace transform of $\frac{1}{(s-\lambda)^m}$ is $\frac{t^{m-1}}{(m-1)!}e^{\lambda t}\cdot u(t)$, where $u(t)$ is the unit step function. Therefore we will consider the effect of each individual pole and derive equations for real-time solution in the following paragraphs. Since $u(t) = 0\ \forall t < 0$ the outcome is a causal filter. The inverse Laplace of the constant term A_0 is trivially $h(t) = A_0\delta(t)$, therefore $A_0 f(x(t))$ is simply added to the final result.

3.1.1. Simple Real Pole

The simplest case is that of a single real pole. In this case the inverse Laplace transform is $h(t) = Ae^{\alpha t}u(t)$ ($\alpha < 0$), which can be plugged in Eq. (6) yielding for time $n+1$

$$\begin{aligned} y_{n+1} &= A\sum_{k=0}^{n}\oint_{x_k}^{x_{k+1}} f(\xi)e^{\alpha\left(n+1-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}d\xi \\ &= A\sum_{k=0}^{n-1}\oint_{x_k}^{x_{k+1}} f(\xi)e^{\alpha\left(n+1-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}d\xi + \\ &+ A\oint_{x_n}^{x_{n+1}} f(\xi)e^{\alpha\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)}d\xi \\ &= e^{\alpha}y_n + A\oint_{x_n}^{x_{n+1}} f(\xi)e^{\alpha\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)}d\xi. \end{aligned} \quad (9)$$

This allows to compute y_{n+1} recursively by storing its previous value y_n and computing the integral. As can be seen, Eq. 9, compared to the three variables integral of Eq. (6), has now only two unknowns, and can computed recursively. Please note that this is possible due to $h(t)$ being reduced to an exponential. The integral can be solved analytically, numerically, or can be stored in a lookup table. Expressions such as e^{α} can be precomputed and the value of the previous output can be stored in memory.

3.1.2. Multiple Real Poles

From Eq. (8) we can observe that for those poles with multiplicity $m > 1$, the inverse transform has the form $A\frac{t^r}{r!}e^{\alpha t}u(t)$ for $r = 0,\dots,m-1$. Therefore, by plugging $h(t) = A\frac{t^r}{r!}e^{\alpha t}u(t)$, into Eq (6)

$$y_n = \frac{A}{r!}\sum_{k=0}^{n-1}\oint_{x_k}^{x_{k+1}} f(\xi)\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)^r e^{\alpha\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}d\xi. \quad (10)$$

Let

$$I_{k,r,n} = \oint_{x_k}^{x_{k+1}} f(\xi)\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)^r e^{\alpha\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}d\xi$$

the solution can be now written as

$$y_n = \frac{A}{r!}\sum_{k=0}^{n-1} I_{k,r,n}.$$

for $k = 0,\dots,n-1$,

The evaluation of the integrals $I_{k,r,n}$ can be done in a recursive way. At time step $n+1$ it is only necessary to compute the integral

$$I_{n,r,n+1} = \oint_{x_n}^{x_{n+1}} f(\xi)\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)^r e^{\alpha\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)}d\xi.$$

While, for $k < n$, one has

$$I_{k,r,n+1} = \oint_{x_k}^{x_{k+1}} f(\xi)\left(n+1-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)^r \cdot \\ \cdot e^{\alpha\left(n+1-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}d\xi.$$

By recalling $(a+1)^r = \sum_{l=0}^{r}\binom{r}{l}a^l$, one gets

$$I_{k,r,n+1} = e^{\alpha}\oint_{x_k}^{x_{k+1}} f(\xi)\sum_{l=0}^{r}\binom{r}{l}\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)^l \cdot \\ \cdot e^{\alpha\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}d\xi,$$

thus,

$$I_{k,r,n+1} = e^{\alpha}\sum_{l=0}^{r}\binom{r}{l}I_{k,l,n}. \quad (11)$$

Therefore, to compute the coefficient related to the exponential r, the previous exponentials must be known. However, this does not add further computations because, in general, the previous exponentials appear in other terms of the inverse transform.

3.1.3. Simple Complex Conjugate Poles

Complex conjugate addenda from Eq. (8) that have multiplicity $\mu = 1$ can be considered together. Their contribution in the time domain is $h(t) = (Be^{\beta t} + \overline{B}e^{\overline{\beta}t})u(t) = 2\Re(Be^{\beta t})u(t)$, with $B,\beta \in \mathbb{C}$ and $\Re(\beta) < 0$. From (6) we can, thus, compute the output as

$$y_n = 2\sum_{k=0}^{n-1}\oint_{x_k}^{x_{k+1}} f(\xi)\Re\left(Be^{\beta\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}\right)d\xi.$$

To ease the computation of the recursion it is better to calculate first

$$\hat{y}_n = 2B\sum_{k=0}^{n-1}\oint_{x_k}^{x_{k+1}} f(\xi)e^{\beta\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}d\xi,$$

and then

$$y_n = \Re(\hat{y}_n). \tag{12}$$

Following the same procedure of Section 3.1.1, we obtain

$$\hat{y}_{n+1} = e^{\beta}\hat{y}_n + 2B\int_{x_n}^{x_{n+1}} f(\xi)e^{\beta\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)}d\xi \tag{13}$$

and convert this to y_{n+1} using Eq. (12). In the Appendix we will reformulate this in terms of real numbers, for the sake of completeness, which is more apt to real-time implementation.

3.1.4. Multiple Complex Conjugate Poles

Let $h(t) = \frac{t^r}{r!}(Be^{\beta t} + \overline{B}e^{\overline{\beta}t})u(t) = 2\frac{t^r}{r!}\Re(Be^{\beta t})u(t)$, with B and β as discussed in the previous section. Eq. (6) gives now

$$y_n = \frac{2}{r!}\sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} f(\xi)\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)^r \cdot \cdot \Re\left(Be^{\beta\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}\right)d\xi.$$

As in Section 3.1.3, we first compute

$$\hat{y}_n = \frac{2B}{r!}\sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} f(\xi)\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)^r e^{\beta\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}d\xi, \tag{14}$$

and then $y_n = \Re(\hat{y}_n)$. The integral can be defined as

$$I_{k,r,n} = \int_{x_k}^{x_{k+1}} f(\xi)\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)^r e^{\beta\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)}d\xi$$

for $k = 0, \ldots, n-1$, thus yielding

$$\hat{y}_n = \frac{2B}{r!}\sum_{k=0}^{n-1} I_{k,r,n}.$$

A usable equation for computing the output recursively can be obtained as done in Section 3.1.2. The integral at step $n+1$, is obtained as

$$I_{n,r,n+1} = \int_{x_n}^{x_{n+1}} f(\xi)\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)^r e^{\beta\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)}d\xi.$$

while the integrals $I_{k,r,n+1}$ for $k < n$ can be computed as follows

$$I_{k,r,n+1} = e^{\beta}\sum_{l=0}^{r}\binom{r}{l}I_{k,l,n}.$$

3.2. Algorithm Properties and Overview

In the above, we have derived expressions to compute the nonlinear output for rational functions by residual decomposition. We would like to summarize here the steps of the algorithm for the design and implementation of the AA-IIR method.

- Devise IIR filter specifications according to the problem at hand and the computational constraints;
- Compute the Laplace filter coefficients in terms of rational polynomials $F(s), G(s)$;
- Conduct Partial Fraction Decomposition;
- For each real pole/complex conjugate pair, derive a computable expression according to Sections 3.1.1-3.1.4. The definite integrals can be computed numerically or analytically using the fundamental theorem of calculus;
- Sum all the partial outputs from each pole/conjugate pole to obtain the final expression.

Similarly to the AA-FIR method, the computational cost of the overall algorithm depends on both the IIR filter order and on the nonlinear function. Fortunately, increasing the order of the filter, linearly increases the computational cost, and savings can be done by storing expressions that are used multiple times. As for ill conditioning, differently from the AA-FIR methods, where the kernel order is proportional to the nonlinear function antiderivative order, with our AA-IIR method, raising the order of the Laplace-domain filter only adds more addenda as per Eq. (8). It must be reminded that the filter introduces phase distortion. Since causal IIR filters are not linear phase, this issue should be carefully considered when implementing the method.

Experiment	Parameters
OVS-2	Oversampling factor: 2
OVS-8	Oversampling factor: 8
AA-FIR-1	Rectangular kernel
AA-FIR-2	Triangular kernel
AA-IIR-1	Butterworth order 2, $F_c = 0.45 \cdot F_s$
AA-IIR-2	Chebyshev type II, order 10, $F_{bs} = 0.61 \cdot F_s$

Table 1: The methods selected for comparison. Please note that the base sampling rate is 44100 for all experiments. The oversampling methods uses a Chebyshev type I filter, order 8, cutoff at $0.8 \cdot F_s$ (Matlab's default).

4. EXPERIMENTS

In order to evaluate the effectiveness of this method we consider the hard clipping function, defined as

$$f(x) = \begin{cases} x & -1 \le x \le 1 \\ sgn(x) & otherwise \end{cases} \tag{15}$$

This nonlinear function is commonly used for comparing antialiasing techniques [6]. We will compare oversampling, AA-FIR and AA-IIR and, for a fruitful comparison, we will describe the computational cost of each method.

Considering that each method has different settings that can be adjusted to trade cost and performance, we have provided a number of configurations, shown in Table 1. For the proposed method we have taken a 2nd order Butterworth filter, which requires computing only one pair of complex conjugate poles (one integral), and a 10th-order Cebyshev type II filter, with a steep transition and stopband attenuation of -60dB. The Laplace-domain filters used for our method are shown in Figure 1 and compared to the rectangle kernel used by the AA-FIR method. A reference Matlab implementation for the proposed method is provided online[2].

For a fair comparison, OVS-2 and OVS-8 employ linear interpolation for the upsampling.

[2] http://www.dangelo.audio/dafx2021-aaiir.html

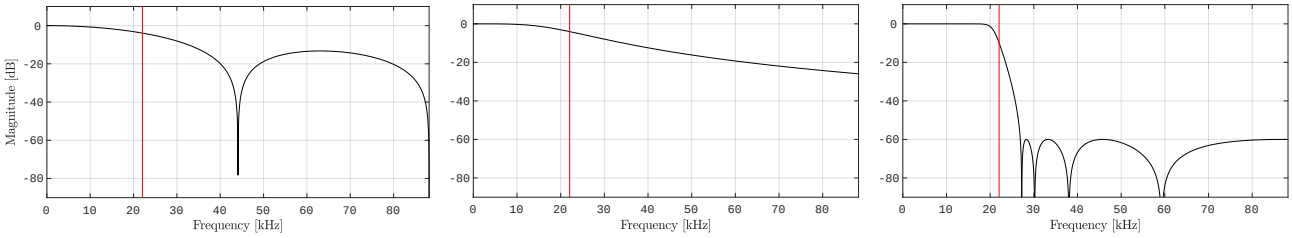

Figure 1: Comparison of the antialiasing filters used in AA-FIR-1 (left), AA-IIR-1 (middle), AA-IIR-2 (right). The red line indicates the Nyquist frequency at 44100 Hz.

4.1. Aliasing Reduction

We first perform a linear sine sweep for all the methods considered in this work, running through the hard clipper after multiplication with an input gain of 10. Spectrograms are computed using 2048 bins, a Blackman-Harris window and an overlap of 16 samples, as shown in Figure 2. The figures show that the AA-FIR-1 and AA-IIR-1 are comparable, with the AA-IIR-1 slightly superior to AA-FIR-1, while AA-IIR-2 is superior to AA-FIR-2. Oversampling by a factor 8 is among the best methods, while oversampling by a factor 2 is the worst among the antialiasing methods.

To precisely assess the performance of the methods, we conducted a batch estimate of the SNR for the whole range of a piano keyboard. We fed a constant frequency sine signal with input gain 10, as above, through the different hard clipping methods, and estimated the SNR considering all components but the partials as noise. Figure 3 reports data from this benchmark. As can be seen, the performance of AA-IIR-1 is similar or slightly better than AA-FIR-1, while the performance of AA-IIR-2 outperforms all other methods by a large margin, with exception of OVS-8. Please note, however, that the proposed method performs almost as good as OVS-8 when the fundamental frequency increases, i.e. where antialiasing performance matters more.

Since the proposed method allows generalizing to any kind of filter, there is headroom for improvement, and the performance can be optimized for the specific task. As an example, with Butterworth filters, the cutoff can be decreased if the higher region of the spectrum is not perceptually relevant. With Chebyshev II filters the filter cutoff can be carefully selected to make the stopband ripple notches be reflected just above DC, to reduce the perceptual impact of aliased components that occur below the fundamental of a tone. Additionally, the order of the filter can be adjusted to meet the desired performance and computational cost trade off.

4.2. Computational Cost

To conclude the experimental section we illustrate the computational cost involved with the discussed solutions for the hard clipper case.

The trivial method only consists of two max and min operations. Oversampling requires running interpolation and decimation filters and running the trivial method at a higher sampling rate. As an example, in this work we employed linear interpolation for interpolation and an 8-th order IIR decimation filter. The cost of the OVS-8 method is, thus, 312 floating point operations for the filters, plus the hard clipper, which can be implemented with 8 max and 8 min floating point operations per sample.

The AA-FIR-1 method requires computing the first integral of the hard clipping function and dividing by the first-order difference, i.e. 5 floating point MUL and SUM and 1 DIV. The cost largely increases with the AA-FIR-2 method, where the required operations are: 25 SUM and MUL, 2 DIV, unless the first order difference is below a small numerical threshold for which less operations are required.

With the proposed methods, computing the exponential functions is the most expensive operation. For each pair of complex conjugate poles, i.e. we have - in the worst case - 2 exponentials, 1 DIV, 16 SUM and MUL. The worst case, however, occurs only for those low probability cases where $(x[n] > 1$ AND $x[n-1] < -1)$ or $(x[n] > -1$ AND $x[n-1] < 1)$, which is uncommon with audio signal. On the other hand, for those cases where either $x[n]$ or $x[n-1]$ exceed the clipping threshold, the result of the integral is a constant, therefore in practical applications the computational cost is lower than the worst case. We drawn a statistics by feeding our method with a logarithmic sine sweep up to $F_s/2$ with gain 10, and counting the occurrences of the worst and best case solutions over a number of $44100 \cdot 10$ samples. We observed that only 16.3% of the input samples require computing the solution according to the worst case, while 73.2% of the integrals can be solved using the precomputed constant (percentage have been rounded to the nearest tenth).

In general, the cost of functions including an exponential can be highly reduced by taking advantage of the Lambert W function [11], while the cost of the exponential itself can be reduced by taking advantage of the floating point representation. In this case, it requires only 9 floating point operations and some integer and casting operations. In consideration of this, the AA-IIR-1 method, on average, has a computational cost comparable to AA-FIR-2 or inferior, while the AA-IIR-2 is still much cheaper than OVS-8.

5. CONCLUSIONS

This paper described a technique for aliasing reduction with static nonlinearities that allows extended flexibility in both performance and computational cost. The proposed method extends the work by Parker et al. to rational IIR filters, being based on the application of the nonlinearity in the continuous-time domain and an antialiasing filter before discretizing back the signal to discrete time. While previously a computable solution was known for antialiasing FIR kernels, in our work we describe how to implement rational IIR filter transfer functions at a reasonable computational cost. We derive the expressions for simple and multiple real poles, single and conjugate complex poles, which are used separately to compute

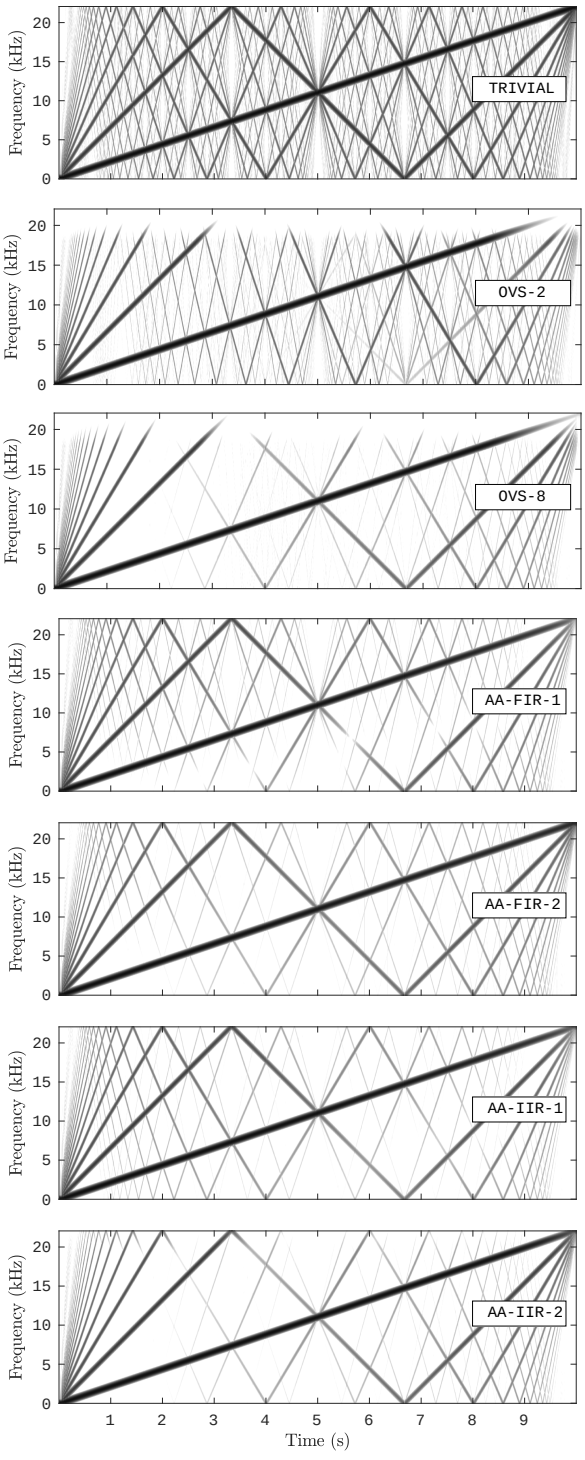

Figure 2: Spectrograms of a linear sine sweep through the hard clipper, with input gain 10, for each of the antialiasing methods in Table 1.

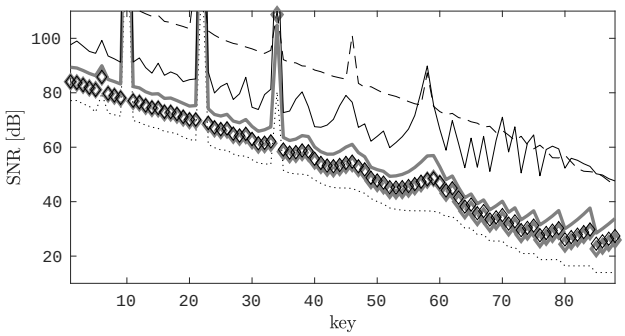

Figure 3: SNR performance of the hard clipper antialiasing methods for the 88 piano keyboard notes, with a sine input with gain 10: trivial (fine dotted line), AA-FIR-1 (gray diamonds), AA-IIR-1 (black diamonds), AA-FIR-2 (solid gray), AA-IIR-2 (solid black), OVS-8 (dashed line).

the final result. The results in terms of performance depend on the filter design, which can be flexible and can reduce aliasing by a large amount at high frequency with a reduction of the computational cost with respect to oversampling techniques, as shown by our experiments.

An upper bound in terms of SNR is imposed by the use of linear interpolation for conversion from discrete- to continuous-time, as discussed in the Appendix. Future works should concentrate on removing this constraint by finding alternative ways to formulate the problem.

A. APPENDIX: EFFECT OF THE LINEAR INTERPOLATION

As discussed in the introduction of the paper, linear interpolation provides one quick way to upsample the signal in oversampling methods, and is the only currently known method to provide an analytical solution in AA methods. However, it allows spurious components into the upsampled signal (or into the continuous-time signal, in our case). These components are, then, processed by the nonlinearity, introducing additional undesired tones. We can easily show why linear interpolation impairs the signal and, thus, why it produces spurious components that cannot be eliminated during conversion from continuous- to discrete-time, and thus, represent an upper bound to SNR performance.

The operation of upsampling a discrete-time into continuous-time using linear interpolation is equivalent to convolution with a triangular kernel

$$h_t(t) = \begin{cases} t+1 & -1 \leq t \leq 0 \\ 1-t & 0 \leq t \leq 1 \\ 0 & otherwise \end{cases} \tag{16}$$

whose Laplace transform is

$$H(s) = \frac{e^{-s} + s - 1}{s^2}. \tag{17}$$

The transfer function of such a kernel, therefore, is $1/2$ for $s = 0$ and falls to 0 only at $+\infty$, allowing replicas of the discrete-time spectrum to pass through, impairing the reconstruction quality.

The aliasing reduction performance of any AA method will, thus, be impaired by the amount of spurious components allowed by the linear interpolation filter. For this reason, the OVS-8 and the AA-IIR-2 methods are very close: they are reaching the upper SNR bound imposed by the upsampling filter. Of course, oversampling can be implemented with arbitrary upsampling filters, while the currently proposed AA methods are defined only for linear interpolation upsampling. Future works should aim at finding viable solutions using AA methods with better upsampling filters.

B. APPENDIX: RELATION TO THE IMPULSE INVARIANCE TRANSFORM

It is worth applying the proposed method to the linear case, in order to gain further insight. Let, thus, $f(x) = x$, if the antialiasing filter is a first order lowpass with pole $\alpha < 0$ and unity gain, its transfer function and impulse response are

$$H(s) = -\frac{\alpha}{s-\alpha},$$

$$h(t) = -\alpha e^{\alpha t} u(t).$$

Eq. (9) is, thus

$$y_{n+1} = e^{\alpha} y_n - \alpha \int_{x_n}^{x_{n+1}} \xi e^{\alpha\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)} d\xi.$$

This can be solved analytically, yielding

$$y_{n+1} = e^{\alpha} y_n - \frac{e^{\alpha}-\alpha-1}{\alpha} x_{n+1} - \frac{(\alpha-1)e^{\alpha}+1}{\alpha} x_n.$$

The transfer function is, thus

$$H(z) = -\frac{1}{\alpha} \frac{(e^{\alpha}-\alpha-1) + ((\alpha-1)e^{\alpha}+1)z^{-1}}{1-e^{\alpha}z^{-1}}.$$

This transfer function preserves unity gain at DC and the denominator has the same form of the impulse invariance transform [12]. This may add further insight to this family of antialiasing methods as they may be understood in relation to known linear filter discretization techniques. Please note that F_s is unitary as in the rest of the paper.

C. APPENDIX: REAL-VALUED FORMULATION FOR COMPLEX POLES

IIR antialiasing filter may often have simple or multiple complex conjugate poles in their transfer function. In Sections 3.1.3-3.1.4 we derived expressions for the AA-IIR method keeping the complex notation and converting the result to real values (see Eq. 12). In this section we express the same algorithm in terms of real values, which is more convenient to implement. We will derive the equations for simple complex conjugate poles and multiple complex conjugate poles.

C.1. Simple Complex Conjugate Poles

By letting $\hat{y}_n = y_n + j z_n$ and considering again Eq. (13), one gets

$$\begin{aligned} y_{n+1} =& \Re(\hat{y}_{n+1}) \\ =& \Re(e^{\beta}) y_n - \Im(e^{\beta}) z_n \\ &+ 2\Re(B) \int_{x_n}^{x_{n+1}} f(\xi) \Re(e^{\beta\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)}) d\xi \\ &- 2\Im(B) \int_{x_n}^{x_{n+1}} f(\xi) \Im(e^{\beta\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)}) d\xi. \end{aligned} \tag{18}$$

Let

$$\begin{aligned} I_n^{(R)} &:= \int_{x_n}^{x_{n+1}} f(\xi) \Re(e^{\beta\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)}) d\xi, \\ I_n^{(I)} &:= \int_{x_n}^{x_{n+1}} f(\xi) \Im(e^{\beta\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)}) d\xi, \end{aligned} \tag{19}$$

then Eq. (18) can be rewritten as

$$y_{n+1} = \Re(e^{\beta}) y_n - \Im(e^{\beta}) z_n + 2\Re(B) I_n^{(R)} - 2\Im(B) I_n^{(I)}. \tag{20}$$

Similarly for the imaginary part

$$z_{n+1} = \Im(e^{\beta}) y_n + \Re(e^{\beta}) z_n + 2\Im(B) I_n^{(R)} + 2\Re(B) I_n^{(I)}. \tag{21}$$

Such a formulation allows to find a recursive expression involving only y_n. By rewriting equations (20) and (21) for time $n+2$ and solving the system, one gets

$$y_{n+2} = 2\Re(e^{\beta}) y_{n+1} - |e^{\beta}|^2 y_n + 2\Re(B) I_{n+1}^{(R)} \tag{22}$$

$$- 2\Re(B e^{\bar{\beta}}) I_n^{(R)} - 2\Im(B) I_{n+1}^{(I)} + 2\Im(B e^{\bar{\beta}}) I_n^{(I)}. \tag{23}$$

In the linear case ($f(x) = x$) this becomes a second order linear filter with two complex conjugate poles, not unlike what was shown in Appendix B.

This real-valued formulation of the method, however, may present further difficulties. Let $\beta = \beta_1 + j\beta_2$ and expressing the integrals from Eq. (19) explicitly, one gets

$$I_n^{(R)} = \int_{x_n}^{x_{n+1}} f(\xi) e^{\beta_1\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)} \cos\left(\beta_2\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)\right) d\xi$$

$$I_n^{(I)} = \int_{x_n}^{x_{n+1}} f(\xi) e^{\beta_1\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)} \sin\left(\beta_2\left(1-\frac{\xi-x_n}{x_{n+1}-x_n}\right)\right) d\xi$$

These can be harder to solve analytically compared to that in Eq (13).

C.2. Multiple Complex Conjugate Poles

A real-valued formulation for the method can be expressed also in the multiple complex conjugate poles case.

Let β be one of the complex conjugate poles and m its multiplicity. The partial fraction decomposition will show terms of the type

$$\frac{B}{(s-\beta)^{r+1}} + \frac{\bar{B}}{(s-\bar{\beta})^{r+1}}$$

where $r = 0, \ldots, m-1$. The related impulse response is $h(t) = \frac{t^r}{r!}(Be^{\beta t} + \bar{B}e^{\bar{\beta} t})u(t) = 2\frac{t^r}{r!}\Re(Be^{\beta t})u(t)$. For this impulse re-

sponse, the algorithm is, thus, for $0 \leq k < n$

$$\begin{aligned} I_{k,r,n} &= \int_{x_k}^{x_{k+1}} f(\xi)\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)^r e^{\beta\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)} d\xi, \\ \hat{y}_n &= \frac{2B}{r!}\sum_{k=0}^{n-1} I_{k,r,n}, \\ y_n &= \Re(\hat{y}_n), \end{aligned} \tag{24}$$

and the integrals $I_{k,r,n+1}$ for $k < n$ can be computed recursively as follows

$$I_{k,r,n+1} = e^{\beta}\sum_{l=0}^{r}\binom{r}{l} I_{k,l,n}.$$

To formulate the algorithm let $I_{k,r,n} = I^{(R)}_{k,r,n} + jI^{(I)}_{k,r,n}$, and

$$I^{(R)}_{k,r,n} = \int_{x_k}^{x_{k+1}} f(\xi)\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)^r \Re(e^{\beta\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)})d\xi$$

$$I^{(I)}_{k,r,n} = \int_{x_k}^{x_{k+1}} f(\xi)\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)^r \Im(e^{\beta\left(n-k-\frac{\xi-x_k}{x_{k+1}-x_k}\right)})d\xi.$$

By extracting the real part of $\hat{y}_n$ in (24) one gets

$$y_n = \frac{2}{r!}\sum_{k=0}^{n-1}(\Re(B)I^{(R)}_{k,r,n} - \Im(B)I^{(I)}_{k,r,n}),$$

and the recursive integrals become

$$\begin{aligned} I^{(R)}_{k,r,n+1} &= \sum_{l=0}^{r}\binom{r}{l}(\Re(e^{\beta})I^{(R)}_{k,l,n} - \Im(e^{\beta})I^{(I)}_{k,l,n}); \\ I^{(I)}_{k,r,n+1} &= \sum_{l=0}^{r}\binom{r}{l}(\Im(e^{\beta})I^{(R)}_{k,l,n} + \Re(e^{\beta})I^{(I)}_{k,l,n}) \end{aligned}$$

for $k < n$.

D. REFERENCES

[1] Vesa Valimaki and Antti Huovilainen, "Antialiasing oscillators in subtractive synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 116–125, 2007.

[2] Udo Zölzer, *DAFX-Digital Audio Effects*, John Wiley and Sons, 2011.

[3] Aaron Wishnick, "Time-varying filters for musical applications.," in *Proc. Int. Conf. Digital Audio Effects (DAFx-14)*, 2014, pp. 69–76.

[4] Julian Parker and Stefano D'Angelo, "A digital model of the Buchla lowpass-gate," in *Proceedings of the International Conference on Digital Audio Effects*, 2013, pp. 278–285.

[5] Fabián Esqueda, Henri Pöntynen, Julian Parker, and Stefan Bilbao, "Virtual analog models of the lockhart and serge wavefolders," *Applied Sciences*, vol. 7, no. 12, pp. 1328, 2017.

[6] Julian 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.

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

[8] Davide Albertini, Alberto Bernardini, and Augusto Sarti, "Antiderivative antialiasing in nonlinear wave digital filters," in *Proceedings of the International Conference on Digital Audio Effects*, 2020.

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

[10] Julen Kahles, Fabián Esqueda, and Vesa Välimäki, "Oversampling for nonlinear waveshaping: Choosing the right filters," *Journal of the Audio Engineering Society*, vol. 67, no. 6, pp. 440–449, 2019.

[11] Stefano D'Angelo, Leonardo Gabrielli, and Luca Turchet, "Fast approximation of the Lambert W function for virtual analog," in *Proceedings of the International Conference on Digital Audio Effects*, 2019.

[12] Alan V. Oppenheim and Ronald W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.