ANTIALIASED STATE TRAJECTORY NEURAL NETWORKS FOR VIRTUAL ANALOG MODELING

Lasse Köper and Martin Holters

Department of Signal Processing and Communication
Helmut Schmidt University – University of the Federal Armed Forces
Hamburg Germany
lasse.koeper@hsu-hh.de | martin.holters@hsu-hh.de

ABSTRACT

In recent years, virtual analog modeling with neural networks experienced an increase in interest and popularity. Many different modeling approaches have been developed and successfully applied. In this paper we do not propose a novel model architecture, but rather address the problem of aliasing distortion introduced from nonlinearities of the modeled analog circuit. In particular, we propose to apply the general idea of antiderivative antialiasing to a state-trajectory network (STN). Applying antiderivative antialiasing to a stateful system in general leads to an integral of a multivariate function that can only be solved numerically, which is too costly for real-time application. However, an adapted STN can be trained to approximate the solution while being computationally efficient. It is shown that this approach can decrease aliasing distortion in the audioband significantly while only moderately oversampling the network in training and inference.

1. INTRODUCTION

In the realm of audio signal processing, nonlinear systems are widely used to create certain musical effects. The main class of these effects is comprised of clipping or overdrive effects, which add an amount of harmonic frequency content to the output, in order to achieve a distorted sound. Historically these kinds of effects were designed in the analog domain. The sound and behaviour of these analog devices are sought after until today. Consequently, there is a natural interest in recreating them in digital models, a process referred to as virtual analog modeling. Over the last decades many different approaches have been developed for converting the analog circuit into a virtual model. In [1, 2, 3], the authors show that Wave-Digital filters can be used to create a digital model of nonlinear stateful systems. [4] and [5] construct a discrete-time state-space model from circuit schematics, while [6] uses a Port-Hamiltonian formalism to create gueranteed-passive systems. Another approach, which in the recent years experienced an increase in popularity, is to use artificial neural networks. In this work we are using so called state-trajectory networks, which approximate the trajectory in the state-space using a feedforward neural network.

Since nonlinear systems introduce harmonic frequency content that can exceed the Nyquist frequency, all before-mentioned approaches can suffer from aliasing distortion. The most commonly used technique to reduce this aliasing distortion is to oversample the signal with a very high sampling rate. Obviously this increases computational complexity and memory consumption. In order to keep real-time capabilities as good as possible, we propose in this work an antialiased neural network model, which uses only modest oversampling.

This paper is organized as follows. In section 1.1 we give a detailed problem statement, followed by a derivation of the proposed method in section 2. Sections 3 and 4 discuss the neural network structure and training data generation. In section 5 the proposed method is applied to different example circuits, providing the results for this work, followed by some concluding remarks in section 6.

1.1. Problem statement

We consider an analog circuit which is described by an implicit ordinary differential equation

$$g\big(\dot{x}(t), y(t), x(t), u(t)\big) = 0 \tag{1}$$

where the state $x(t)$, the input $u(t)$, and the output $y(t)$ may be vector-valued. We assume an explicit solution

$$\dot{x}(t) = f_{\mathrm{x}}\big(x(t), u(t)\big) \tag{2a}$$

$$y(t) = f_{\mathrm{y}}\big(x(t), u(t)\big) \tag{2b}$$

exists, but does not have a closed form, i.e. may only be determined numerically from (1) using an iterative approach. The system can be transformed to discrete-time by various well-known methods such as the trapezoidal rule, resulting in an implicit update rule of the form

$$\bar{g}\big(\bar{x}(n), \bar{y}(n), \bar{x}(n-1), \bar{u}(n); T\big) = 0 \tag{3}$$

where T denotes the sampling interval, $\bar{u}(n) = u(nT)$ is the sampled input signal and $\bar{y}(n) \approx y(nT)$ approximates samples of the output signal subject to the error introduced by the discretization scheme. The states $\bar{x}(n)$ maintain a relationship to the continuous-time states $x(t)$, but do not necessarily correspond to samples thereof.

There are three problems associated with this approach:

1. Numerical solution of (3) is computationally expensive and may prevent real-time operation. (Note that using an explicit discretization scheme such as forward Euler will not help here due to the implicit nature of (1)).
2. The discretization scheme introduces an error which may become significant for high-frequency signals.
3. Even if the approximation error of the discretization scheme is tolerable, samples of $y(t)$ may not be the desired output: The continuous-time output may contain high-frequency

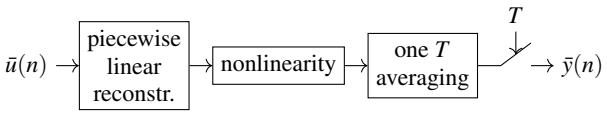

Figure 1: *Underlying idea of antiderivative antialiasing.*

content due to harmonics introduced by the nonlinearity of the system which results in aliasing distortion when sampled. Ideally, one would wish $\bar{y}(n)$ to be samples of a band-limited version of $y(t)$.

Problems 2 and 3 can be mitigated by oversampling the system, but this obviously aggravates problem 1. Problem 3 can also be approached by antiderivative antialiasing [7, 8, 9], but only for stateless and a very limited class of stateful systems [10, 11]. A recent trend to tackle problem 1 is to approximate the solutions

$$\bar{x}(n) = \bar{f}_{\mathrm{x}}\big(\bar{x}(n-1), \bar{u}(n)\big) \tag{4a}$$

$$\bar{y}(n) = \bar{f}_{\mathrm{y}}\big(\bar{x}(n-1), \bar{u}(n)\big) \tag{4b}$$

of (3)[1] (which lack a closed form) with neural networks [12] which are more efficient to evaluate than iterative numerical solution.

In this work, we explore the combination of the approximation with neural networks with the idea underlying the development of antiderivative antialiasing to attack all three problems simultaneously without imposing new restrictions on the considered systems.

2. ANTIALIASED NEURAL NETWORK APPROACH

Perfect aliasing suppression could be obtained by operating in the continuous-time domain. That is, for the samples $\bar{u}(n)$, the continuous input signal $u(t)$ could be formed by using an ideal reconstruction lowpass filter. Then, the continuous-time non-linear system of (2) could be applied to obtain $y(t)$. Bandlimiting to half the sampling-rate with an ideal lowpass filter before sampling would then allow to obtain an aliasing-free output $\bar{y}(n)$. While theoretically perfect, this is clearly impractical.

However, by using non-ideal lowpass filters, one may actually arrive at a practical system. In particular, consider linear interpolation for the reconstruction filter and averaging over one sampling interval for bandlimiting as depicted in figure 1. When implementing a digital system, the continuous signals obviously still pose a problem. The key insight of antiderivative antialiasing is that for a stateless nonlinearity, an equivalent system can be derived that operates solely on the sampled signals, but requires the antiderivative of the nonlinear mapping function—hence the method's name (see [7] for details.) Unfortunately, this only works for stateless nonlinear systems and for a limited class of stateful systems after some modification [10, 11].

But now consider a general stateful nonlinear system of the form (1). Focusing on the time interval from $(n-1)T$ to nT, we first observe that given both the state $x((n-1)T)$ at the beginning of that interval and the input $u(t)$ for the whole interval, the state trajectory $x(t)$ and the output $y(t)$ are fully determined during that interval. Now if $u(t)$ is assumed piecewise linear, i.e. linear in that interval, it in turn is fully determined by its values at the beginning and the end of the interval, namely its samples $\bar{u}(n-1)$ and $\bar{u}(n)$. So from these input samples and the initial state $\bar{x}(n-1) = x((n-1)T)$, the state trajectory $x(t)$ and the output $y(t)$ can be determined up to the interval's end nT. In particular, this allows to determine the state $\bar{x}(n) = x(nT)$ at the end of the interval to be used as the initial state for the subsequent time interval. Furthermore, we can apply the averaging operation to the output to obtain the antialiased output samples

$$\bar{y}(n) = \frac{1}{T}\int_{(n-1)T}^{nT} y(t). \tag{5}$$

So to summarize, we may conclude that input samples $\bar{u}(n-1)$ and $\bar{u}(n)$ and initial state $\bar{x}(n-1)$ are sufficient to determine the next state $\bar{x}(n)$ and the antialiased output $\bar{y}(n)$, or in other words, that there exist functions

$$\bar{x}(n) = \bar{f}_{\mathrm{x}}\big(\bar{x}(n-1), \bar{u}(n-1), \bar{u}(n)\big) \tag{6a}$$

$$\bar{y}(n) = \bar{f}_{\mathrm{y}}\big(\bar{x}(n-1), \bar{u}(n-1), \bar{u}(n)\big) \tag{6b}$$

that correspond to the approach from figure 1 applied to an arbitrary stateful system. However, these functions lack a closed form and in fact, evaluating them numerically requires a scheme like (3) operating at a sampling rate high enough that discretization error and aliasing distortion are sufficiently low. Thus, we are back to oversampling, but with deliberately simple interpolation and decimation filters.

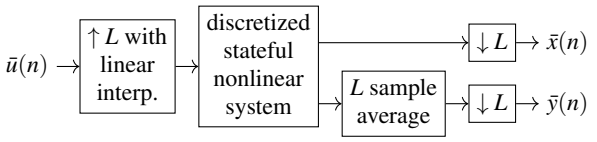

Figure 2: *Oversampled system to obtain training data for approximation with neural network.*

At this point, the neural network comes into play. Given that (6) describes a system with the desired properties except for the functions being computationally rather costly, it suggests itself to approximate them using a neural network. We may therefore boil down our approach to the following: Feed a suitable training stimulus $\bar{u}(n)$ to an oversampled, classical simulation approach to obtain corresponding sequences of $\bar{x}(n)$ and $\bar{y}(n)$ and use these as training data for a neural network to approximate (6). But note that the oversampling has to be of a particular form: Upsampling of the input uses linear interpolation, downsampling of the output uses averaging over one sampling interval. In contrast, downsampling of the states uses no decimation filter at all, as these act like snapshots of the oversampled system from which it could be restarted; i.e. it must be possible to reconstruct the original state trajectory, which would be impossible if the states were filtered. Thus, the system for generating training data finally looks as depicted in figure 2.

The simple interpolation and decimation filters are required to avoid additional states from appearing. Comparing (4) and (6), only $\bar{u}(n-1)$ needs to be newly introduced. It would be possible, however, to utilize more sophisticated filters as long as more samples of $\bar{u}$ are provided to fully cover the filters' combined support.

[1] Some modeling approaches yield an output equation dependent on $\bar{x}(n)$ instead of $\bar{x}(n-1)$, i.e. of the form $\bar{y}(n) = \bar{f}^*_{\mathrm{y}}\big(\bar{x}(n), \bar{u}(n)\big)$. It may not always be possible to express the solution of (3) like that, however. On the other hand, it is always possible to rewrite from $\bar{f}^*_{\mathrm{y}}$ to $\bar{f}_{\mathrm{y}}$ via $\bar{f}_{\mathrm{y}}\big(\bar{x}(n-1), \bar{u}(n)\big) = \bar{f}^*_{\mathrm{y}}\big(\bar{f}_{\mathrm{x}}\big(\bar{x}(n-1), \bar{u}(n)\big), \bar{u}(n)\big)$, so the case considered here is the more general one.

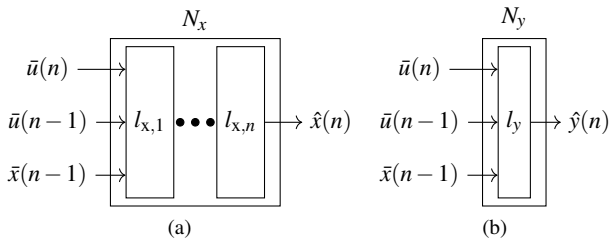

Figure 3: *Network structure during training, (a) trained on downsampled states, (b) trained on downsampled and averaged output.*

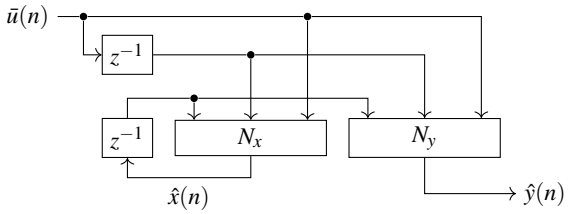

Figure 4: *Network structure during inference time.*

3. NEURAL NETWORK ARCHITECTURE

For approximating a stateful nonlinear system like (6), many suitable neural network modeling approaches can be found. To name a few, Wright et al. [13] employ recurrent neural networks in a black-box model approach, while in [14] time-varying effects are modeled with neural grey box models. Wilczek et al. [15] approximate the underlying nonlinear differential equation by a neural network and combine it with a numerical solver.

However, given that the system is in state-space form, it is preferable to apply the neural network directly in the state-space. Therefore using state-trajectory networks [12] seems to be the natural choice. Furthermore, this has the advantage that modeling of states and output can be performed in two different networks. This can be helpful to reduce stability issues during inference, since one network can be trained to solely model the dynamics of the system, whereas the other is only responsible for the nonlinear mapping from states to output.

Figure 3 shows the two models during training. The amount of hidden layers in the state predicting model can be adjusted to the complexity of the modeled analog circuit. In this work it was sufficient to use small networks with up to three hidden layers, which is also beneficial in terms of realtime capability. For the network predicting the output, it is even possible to use only one layer. Regarding the hidden layer type, we opted for fully connected layers followed by a hyperbolic tangent activation function. The last layer is a fully connected layer with no bias and linear activation.

All trainings were performed using a mean-squared error loss function and an NAdam optimizer following the hyperparameter settings of [16]. Figure 4 shows the model during inference. The networks N_x and N_y correspond to the two networks trained in figure 3 respectively. The predicted state vector $\hat{x}(n)$ is then used in the next time step as an input of N_x following the typical structure of a state-trajectory network. Similarly, by using the current and previous input sample, as well as the previously predicted state, the network N_y can approximate the averaging and downsampling operation from figure 2 employing a simple static mapping.

4. TRAINING DATA GENERATION

Generating suitable training data for the models is a crucial point in this work, since the antialiasing operation is in great part performed during training data generation. One of the first questions arising is whether to use measurement or simulation data. In this work it is obvious that we have to use simulation data, since the antialiasing lowpass filters in measurement hardware are in general not in the particular form required for the presented approach. Therefore the proposed antialiasing scheme could not be applied. The training data is generated by simulation using the ACME framework[2] in Julia[3]. This simulation is based on a state-space modeling approach, discretizing a continuous-time state-space system obtained from circuit dynamics using the trapezoidal rule [5].

We now use a low sampling rate f_{s1} to create our input signal $\bar{u}(n)$. This sampling rate should already be a reasonably high audio sampling rate, because this scheme is most effective if combined with modest oversampling. Therefore, we choose $f_{s1} = 96\,\text{kHz}$ for all examples in the following section. Afterwards the input signal is upsampled to a sampling rate f_{s2} like we saw in figure 2. Unless otherwise noted, we use $f_{s2} = L \cdot f_{s1}$ with $L = 4$, i.e. $f_{s2} = 384\,\text{kHz}$. The simulation is now run using the high sampling rate. After averaging the output, as well as downsampling states and output, the data is ready to be used for training. In order to evaluate the proposed method's antialiasing capabilities, we also create a reference training signal. This signal is obtained by just running the simulation at the low sampling rate f_{s1} with the signal $\bar{u}(n)$ as its input. The reference signal and the antialiased training data are the basis for all comparisons in the following examples. That means we compare the network structure from 3, trained with the antialiased data, against a standard STN using only the current input sample and trained with the reference signal.

As the training input stimulus, we use the exponential sine sweep

$$\bar{u}(n) = A \sin\left(\frac{\Omega_l\,(N-1)}{\log\frac{\Omega_h}{\Omega_l}} \exp\left(\frac{n}{N-1}\log\frac{\Omega_h}{\Omega_l}\right) - 1\right), \quad (7)$$

where N is the signal length in samples, Ω_l and Ω_h are the lower and upper normalized angular frequency $\Omega_{\cdot} = 2\pi\frac{f_{\cdot}}{f_s}$ respectively and A is the amplitude.

The signal length is chosen as $N = 960000$ and the lower and upper frequency as $f_l = 20\,\text{Hz}$ and $f_u = 2000\,\text{Hz}$, respectively. This stimulus is then repeated for different amplitudes A from a range differing between the examples as given in the next section.

5. APPLICATION

5.1. Example 1: 2nd Order Diode Clipper

As a first example we study the aliasing behavior of a second order diode clipper. The schematic can be seen in figure 5 and the corresponding component values are listed in table 1. For the training, the model structure from figure 3 was used with two

[2]Analog Circuit Modeling and Emulation for Julia (v0.10.0): https://github.com/HSU-ANT/ACME.jl

[3]The Julia Programming Language (v1.8.5): https://julialang.org/

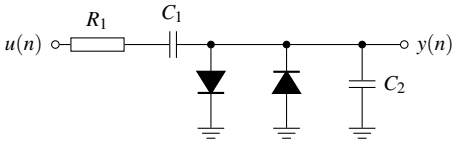

Figure 5: *Second-Order Diode Clipper.*

Table 1: *Second-Order Diode Clipper - Component List.*

Element	Value
R_1	2.2 kΩ
C_1	470 nF
C_2	10 nF
Diodes	$I_s = 1\,\text{pA}$, $v_t = 25\,\text{mV}$, $\eta = 1$

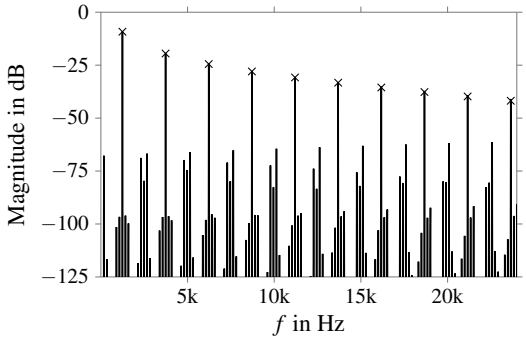

Figure 6: *Second-order diode clipper - reference model without antialiasing, crosses mark the desired harmonics.*

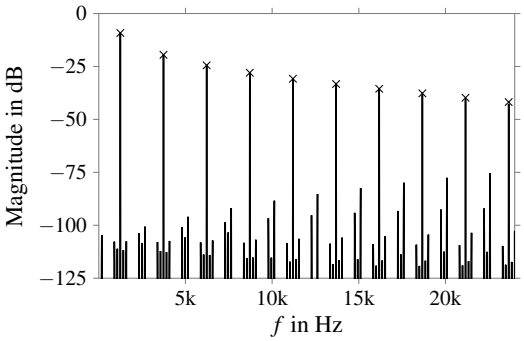

Figure 7: *Second-order diode clipper - model with antialiasing L=4, crosses mark the desired harmonics.*

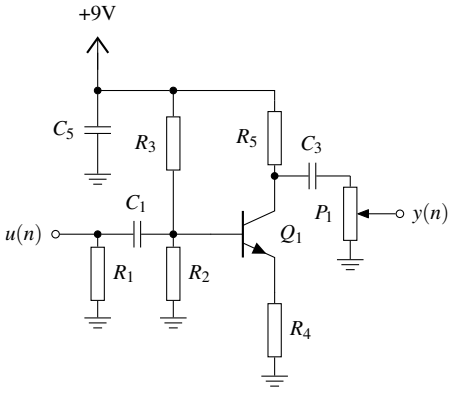

Figure 8: *Birdie Treble Booster.*

hidden layers, each comprising 12 neurons and a hyperbolic tangent activation function. The output model from figure 3 uses one hidden layer with 8 neurons.

In order to train the model on a variety of signal amplitudes, the sinesweep from (7) was evaluated for a set of different amplitudes and combined afterwards to form the final training signal. With the goal of achieving a high amount of aliasing to show the capabilities of the proposed method, we opted for rather high amplitudes in the training signal, ranging from 0.2 V to 12 V. The model was trained over 40 epochs after which the loss did not decrease significantly further. The batch size was set to 1024 samples. To test the aliasing reduction of the proposed model, a single sinusoid with an amplitude of 10 V and a frequency of 1244.5 Hz was applied to the reference(no antialiasing) and the antialiased model. Figure 6 shows the frequency content of the test signal, after being applied to the reference model. As expected we can observe a high amount of aliasing distortion between the desired harmonics. In comparison, the output of the proposed model in figure 7 shows a high reduction of aliasing in the audio band. The aliasing components for higher frequencies experience a much smaller reduction than for the low frequencies. However, this is a well-known property of the antiderivative antialiasing approach from [7] and was to be expected, since the proposed method is based on it.

It should be noted that the diode clipper is simple enough so that it could also be treated with the antialiasing approach of [10, 11]. Indeed, applying that approach yields very similar results. Furthermore, when based on a lookup table, it is more straight-forward to design than training a neural network and computationally cheaper at runtime. So the diode clipper serves as a proof-of-concept, but does not exhibit the advantage of the present method. That will be different for the following cases, where the method of [10, 11] is no longer applicable.

5.2. Example 2: Birdie - Treble Booster

Our next case study is the guitar treble booster *The Birdie*. It is sold as a DIY soldering kit by *musikding* [4] and is based on the Electro-Harmonix *Screaming Bird*. Figure 8 shows the schematic and table 2 the corresponding component values of the circuit. The circuit itself is based on a common emitter amplifier, with a simple highpass filter comprising R_1, C_1 and R_2 at the input. Note that the capacitor C_5 is only used to stabilize the supply voltage. However, we assume an ideal voltage source for the supply and can therefore safely omit C_5 from the simulation, reducing the circuit's dynamics to second order. We will also model the circuit at a fixed level of the volume potentiometer P_1 of 0.5.

Like in the previous example, we go for a model with two fully connected hidden layers and 12 neurons each. The training signal from (7) is adapted to a different set of amplitudes ranging from 0.2 to 8 volts. The model was trained over 30 epochs with a batch size of 1024 samples. As a test signal for inference, we use again a single sinusoid with a frequency of 1244.5Hz but with a slightly smaller amplitude of 8 volts. The frequency domain

[4] The Birdie: https://www.musikding.de/docs/musikding/birdie/birdie_schalt.pdf

Table 2: *Birdie Treble Booster - Component List.*

Element	Value
R_1	1 MΩ
R_2	43 kΩ
R_3	430 kΩ
R_4	390 Ω
R_5	10 kΩ
C_1	2.2 nF
C_3	2.2 nF
C_5	100 µF
Q_1	2N5088
P_1	100 kΩ log

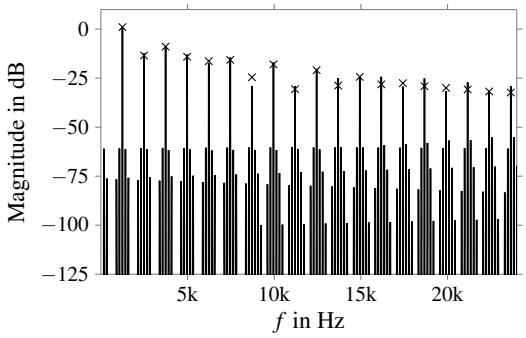

Figure 9: *Birdie - reference model without antialiasing, crosses mark the desired harmonics.*

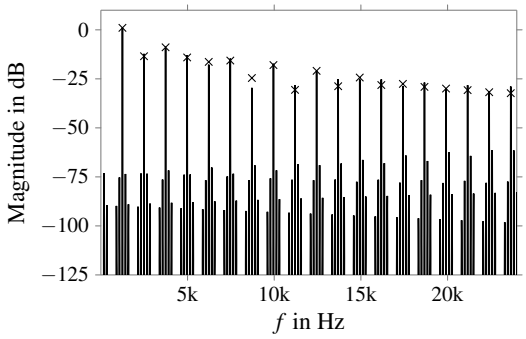

Figure 10: *Birdie - model with antialiasing L=4, crosses mark the desired harmonics.*

output to this test signal can be seen in figure 9 (reference) and in figure 10 (antialiased model). We can observe a similar behavior as for the diode clipper. The reference shows a decent amount of aliasing distortion between the desired harmonics of the output signal, whereas the antialiased model shows only a small amount of aliasing distortion. For low frequencies the aliasing components in figure 10 are barely visible, but they increase for higher frequencies.

5.3. Example 3: Fuzzface

To conclude this section we show the effectiveness of the proposed method on the *Fuzzface* guitar distortion effect. This circuit is a useful addition to the previous examples, because in comparison to the treble booster from the last section it provides a lot more

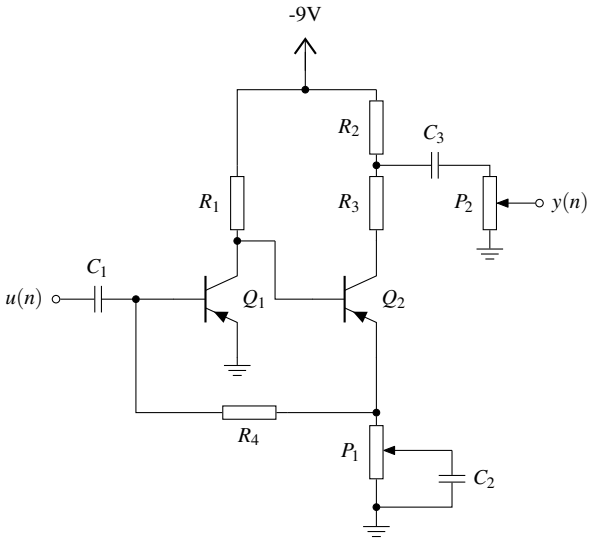

Figure 11: *Fuzzface.*

Table 3: *Fuzzface - Component List.*

Element	Value
R_1	33 kΩ
R_2	470 Ω
R_3	8.2 kΩ
R_4	100 kΩ
C_1	2.2 µF
C_2	20 µF
C_3	100 nF
Q_1, Q_2	AC128
P_1	1 kΩ linear
P_2	500 kΩ log

distortion to the output. Consequently it will produce more aliasing distortion, due to the high amount of harmonic frequency content.

The schematic of the *Fuzzface* can be seen in figure figure 11 and the component values can be found in table 3. The circuit basically consists of an input stage providing a high voltage gain, an output stage and a feedback loop to stabilize the circuit. The amount of distortion added to the input can be adjusted with the potentiometer P_1, which controls the amount of negative feedback via R_4. The output volume can be controlled with the remaining potentiometer P_2. For the sake of simplicity we train the models of the system with fixed values of 0.5 for both potentiometers.

Since this circuit is more complex with respect to the number of states and nonlinear behavior than the previous examples, it is necessary to adjust the simple STN from figure 3 in order to reduce exposure bias and ensure stability. For this reason the network is split into three separate sub-networks, where each network is trained to solely predict one of the three states. This allows a more precise prediction of each state. During inference the three networks are connected in parallel and each predicted state is fed to each individual network as the next input sample. The output network remains the same as in the previous examples. Note that the modification of the network does not effect the proposed antialiasing approach and is only necessary to obtain a stable model during inference.

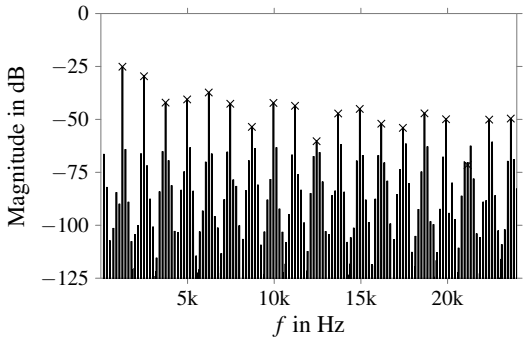

Figure 12: *Fuzzface - reference model without antialiasing, crosses mark the desired harmonics.*

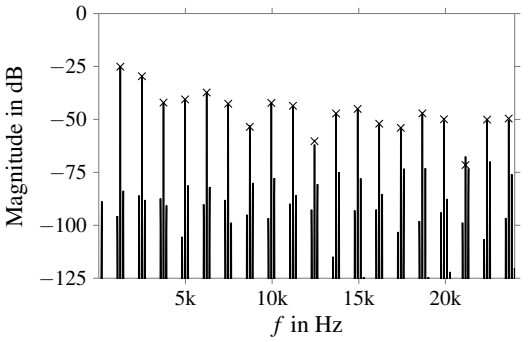

Figure 13: *Fuzzface - model with antialiasing L=4, crosses mark the desired harmonics.*

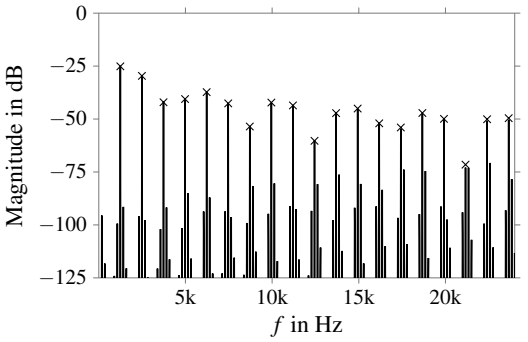

Figure 14: *Fuzzface - model with antialiasing L=8, crosses mark the desired harmonics.*

Regarding the model structure of each individual network, we use two hidden layers with 12 neurons each. For this example, we opted to normalize the states before training. This was necessary because when using the capacitor's voltages as states for the model, they become very difficult to train. The reason for this is that the bias voltage at the capacitors differ a lot from each other. Furthermore the amplitude around this bias values is fairly small. Training the network with these physically meaningful states resulted into a bad model performance. Consequently the states were normalized to

$$x_{i,norm} = \frac{x_i - \mu_{x_i}}{\max\left(x_i - \mu_{x_i}\right)}, \tag{8}$$

where x_i are the individual states and μ_{x_i} are the mean values for each state.

The models for this circuit were trained over 40 epochs using the same training sinesweep as before, but with adapted amplitudes in the more reasonable range of 0.2 to 3 volts. Figure 12 shows the frequency content of the reference model, when excited with the signal $u(n) = 2\sin\left(2\pi \cdot 1244.5\frac{n}{f_s}\right)$. Between the desired harmonics a lot of aliasing components are present. In comparison the antialiased model in figure 13 can reduce these aliasing distortion by a large margin using only an oversampling factor of $L = 4$. Using a higher oversampling factor than $L = 4$ for this model does not result into a significant increase in aliasing reduction. Figure 14 shows the frequency content for $L = 8$. It can be seen that the improvement in comparison to figure 13 is marginally small. However, since the model uses the downsampled data there is no computational overhead when running the model trained with $L = 8$ or higher.

6. CONCLUSION AND OUTLOOK

In this work, we presented an antialiasing approach for state-trajectory networks. This approach uses the general idea of antiderivative antialiasing [7, 8, 9, 10, 11] and applies an approximation of the method to the model's training data. In contrast to the original antiderivative antialiasing, the proposed method is not limited to systems having only one nonlinearity with scalar input. The necessary modification of the STN in order to incorporate the method is straight-forward and easily implemented. Only one additional input with the previous input sample has to be added.

The method was successfully applied to three different nonlinear circuits. In comparison to a reference model, which was sampled with a sampling rate of 96 kHz, the antialiased model was able to attenuate most of the aliasing distortion while using the same sampling rate. Consequently the proposed method can reduce aliasing with only the cost of one additional input. The aliasing reduction works especially well for lower frequencies. Although high frequency aliasing components receive a smaller attenuation it is still an improvement compared to the reference model.

Generally speaking the proposed method is not only applicable to STNs, but basically to all neural virtual analog modeling approaches. The implementation should be straight-forward, since most of the modifications happen during training data generation. For the neural networks only minor adjustments should be necessary.

Another possible extension is the use of more complex filters for the up- and downsampling. E.g. one could replace the rectangular filter kernel of the decimation lowpass with a triangular one, just as was also done in [7] and then add $\bar{u}(n-2)$ as an additional input to the neural networks. But it is also possible to use more complex interpolation filters, which is problematic in antiderivative antialiasing. The only constraint is that the input samples provided to the neural network have to cover the combined support of interpolation and decimation filter. In practice, it may even be possible to violate this constraint and provide fewer input samples than theoretically needed.

In fact, preliminary simulations have shown that for many applications good results could be achieved by using only the current input sample $\bar{u}(n)$. This makes sense, because it basically reduces the linear interpolation filter to a zero-order hold. This approximation is acceptable if the change in amplitude between two adjacent

input samples is sufficiently small. Exploring this design space is left to future work.

7. REFERENCES

[1] K.J. Werner, *Virtual Analog Modeling of Audio Circuits using Wave Digital Filters*, Ph.D. thesis, 2016.

[2] K.J. Werner, E.J. Tebout, S. Cluett, and E. Azelborn, "Modeling and extending the RCA Mark II sound effects filter," in *Proc. 25th Int. Conf. on Digital Audio Effects (DAFx-20in22)*, Vienna, Austria, 2022.

[3] D.T. Yeh and J.O. Smith, "Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations," in *Proc. 11th Int. Conf. on Digital Audio Effects (DAFx-08)*, Espoo, Finland, 2008.

[4] D.T. Yeh, J.S. Abel, and J.O. Smith, "Automated physical modeling of nonlinear audio circuits for realtime audio effects – part I: Theoretical development," *IEEE Trans. Audio, Speech and Language Process.*, vol. 18, no. 4, pp. 728–237, 2010.

[5] M. Holters and U. Zölzer, "A generalized method for the derivation of non-linear state-space models from circuit schematics," in *Proc. European Signal Processing Conference (EUSIPCO-15)*, 2015, pp. 1073–1077.

[6] M. Danish, S. Bilbao, and M. Ducceschi, "Applications of port hamiltonian methods to non-iterative stable simulations of the Korg35 and Moog 4-pole VCF," in *Proc. 24th Int. Conf. on Digital Audio Effects (DAFx20in21)*, Vienna, Austria, 2021.

[7] J.D. Parker, V. Zavalishin, and E. Le Bivic, "Reducing the aliasing of nonlinear waveshaping using continuous-time convolution," in *Proc. 19th Int. Conf. on Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, 2016, pp. 137–144.

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

[9] S. Bilbao, F. Esqueda, and V. Välimäki, "Antiderivative antialiasing, lagrange interpolation and spectral flatness," in *2017 IEEE Workshop on Appl. of Signal Process. to Audio and Acoust. (WASPAA)*, New Paltz, NY, USA, 2017, pp. 141–145.

[10] M. Holters, "Antiderivative antialiasing for stateful systems," in *Proc. 22nd Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019.

[11] M. Holters, "Antiderivative antialiasing for stateful systems," *Appl. Sciences*, vol. 10, no. 1, 2020.

[12] J.D. Parker, F. Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. 22nd Int. Conf. on Digital Audio Effects (DAFx-19)*, Birmingham, UK, 2019.

[13] A Wright, E.P. Damskägg, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proc. 22nd Int. Conf. on Digital Audio Effects (DAFx-19)*, Birmingham, UK, 2019.

[14] A. Wright and V. Välimäki, "Neural modelling of periodically modulated time-varying effects," in *Proc. 23rd Int. Conf. on Digital Audio Effects (DAFx-20)*, Vienna, Austria, 2020.

[15] J. Wilczek, A. Wright, V. Välimäki, and E.A.P. Habets, "Virtual analog modeling of distortion circuits using neural ordinary differential equations," in *Proc. 25th Int. Conf. on Digital Audio Effects (DAFx20in22)*, Vienna, Austria, 2022.

[16] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. 4th Int. Conf. Learn. Repres. (ICLR 2016)*, San Juan, Puerto Rico, 2016.