Audio Engineering Society

Convention Paper 10191

Presented at the AES 157th Convention 2024
October 8–10, New York, NY, USA

This paper was peer-reviewed as a complete manuscript for presentation at this convention. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

A Survey of Methods for the Discretization of Phonograph Record Playback Filters

Benjamin R. Thompson[1], Tre DiPassio[1], Jenna Rutowski[1], and Michael C. Heilemann[1]

[1] *University of Rochester*

Correspondence should be addressed to Benjamin R. Thompson (benjaminrthompson.mail@gmail.com)

ABSTRACT

Since the inception of electrical recording for phonograph records in 1924, records have been intentionally cut with a non-uniform frequency response to maximize the information density on a disc and to improve the signal-to-noise ratio. To reproduce a nominally flat signal within the available bandwidth, the effects of this cutting curve must be undone by applying an inverse curve on playback. Until 1953, with the introduction of what has become known as the RIAA curve, the playback curve required for any particular disc could vary by record company and over time. As a consequence, anyone seeking to hear or restore the information on a disc must have access to equipment that is capable of implementing multiple playback equalizations. This correction may be accomplished with either analog hardware or digital processing. The digital approach has the advantages of reduced cost and expanded versatility, but requires a transformation from continuous time, where the original curves are defined, to discrete time. This transformation inevitably comes with some deviations from the continuous-time response near the Nyquist frequency. There are many established methods for discretizing continuous-time filters, and these vary in performance, computational cost, and inherent latency. In this work, several methods for performing this transformation are explored in the context of phonograph playback equalization, and the performance of each approach is quantified. This work is intended as a resource for anyone developing systems for digital playback equalization or similar applications that require approximating the response of a continuous-time filter digitally.

1 Introduction

Almost all electrically recorded records are cut with an equalization curve that allows the cutter to switch between a constant-velocity response and a constant-amplitude response for different bands within the audible spectrum. For laterally-cut grooves, this allows for smaller groove widths, and thus more information over a given surface area. When high-frequency pre-emphasis is included in the cutting profile, it has the additional effect of increasing the overall signal-to-noise ratio upon playback as high-frequency surface noise is suppressed. When a record is played, an equalization that is the inverse of the cutting curve must be applied to accurately decode the information on the disc [1, 2].

From 1924, when the research arm of Western Electric, Bell Labs, introduced a system for electrically recording phonograph records [3], until the 1950s when the recording industry settled on the RIAA curve as the standard [4], the equalization applied when cutting a master recording varied from record company to record company, and over time as the technology evolved. The properties of some of these curves, particularly for

smaller record companies, are unpublished, and even for the most widely used curves, there is disagreement among professionals about their exact properties [5, 6]. Additionally, even when cutting equalization information is known for a particular label, it is often unclear which curve was utilized for any individual release [7]. As a result, a restoration engineer working with disc recordings must have the ability to implement multiple established playback curves, as well as the means to adjust the properties of the equalization *in situ* to suit a particular recording.

This equalization is often performed using analog equipment. Because all playback curves were originally defined in continuous time, if the cutting transfer function is known, an inverse transfer function can be implemented more or less exactly with this approach. However, many analog playback equalizers are limited to a relatively small number of turnover frequencies for each filter element [8, 9], with the consequence that, if a required turnover frequency falls outside of those values, the correct equalization cannot be implemented with that device. In addition, when performing playback equalization using an analog equalizer in the context of digitizing a disc, it is common to apply the playback curve before recording. If, at some later time, it is decided that a different curve is desirable, the information must be re-recorded. For historic recordings, this may be unacceptable because of the degradation each playback imposes on the medium. Instead, if the signal is recorded without equalization and the equalization is applied in the digital domain, it can be adjusted without consequence to the original media. Digital approaches are more versatile and less expensive than their analog counterparts, however, it is not possible to exactly replicate the response of an analog filter at all frequencies with a digital filter.

The following sections present a survey of several established methods for discretizing continuous-time filters in the context of implementing phonograph playback equalization in the digital domain.

2 The RIAA Curve as a Prototype Continuous-Time Filter

What is now known as the RIAA curve was developed by RCA Victor as the "New Orthophonic" curve and was published in 1953 [10]. This curve was adopted as a standard by a number of recording and broadcast organizations including the Recording Industry Association of America, eventually becoming a universal standard applied to all vinyl records [7].

The RIAA playback curve makes an appropriate platform for evaluating the methods presented here because it contains high-frequency de-emphasis resulting in the presence of features near what will become the Nyquist frequency, which makes it challenging to discretize accurately. Additionally, its ubiquity means that it would likely be included in any system for applying playback equalization digitally.

The original RIAA playback curve is defined by three time constants: $\tau_0 = 318\mu s$, $\tau_1 = 75\mu s$, and $\tau_2 = 3180\mu s$. Though an amendment adding a fourth time constant at $7950\mu s$ was adopted in 1976, it was later withdrawn [11], and in this article, we will use only the original three time constants. The transfer function for this filter, with a gain factor applied such that the magnitude response at 1 kHz is approximately equal to 0 dB, is

$$H(s) = \frac{\frac{1}{\tau_1}\left(s+\frac{1}{\tau_0}\right)}{\left(s+\frac{1}{\tau_1}\right)\left(s+\frac{1}{\tau_2}\right)}. \tag{1}$$

Expressed as a ratio of polynomials, the transfer function becomes,

$$H(s) = \frac{\frac{1}{\tau_1}s+\frac{1}{\tau_1\tau_0}}{s^2+\frac{\tau_1+\tau_2}{\tau_1\tau_2}s+\frac{1}{\tau_1\tau_2}}. \tag{2}$$

3 Discretization Methods

Eight discretization methods were chosen to be evaluated in the context of playback equalization. Each of the methods is order-preserving in the sense that the order of the denominator of the transfer function, N, remains unchanged by the discretization. However, if the order of the numerator of the continuous-time filter, M_s, is less than N, it may be increased by the discretization process such that the order of the numerator of the discrete-time filter, M_z, becomes equal to N. The following subsections contain a brief theoretical overview of each method.

3.1 Zero-Order Hold

This method entails approximating the output of a continuous filter, $H(s)$, when the input signal, $x[n] = x(nT)$, is a sampled representation of a continuous-time signal, $x(t)$, with a sampling period of T and integer sample index n. For a zero-order hold, the sample value at $x[n]$ is directly held constant for the entire sampling period, resulting in the $x(t) = x(nT) = x[n]$ for the interval from $nT \leq t < (T + nT)$. This results in an approximated continuous-time signal that resembles a piece-wise function. From [12], the discrete-time transfer function, $H(z)$, may be approximated from $H(s)$ as,

$$H(z) = \left(1 - z^{-1}\right) \mathcal{Z}\left[\frac{H(s)}{s}\right], \tag{3}$$

where $\mathcal{Z}$ is the Z-transform. In this work, the Zero-Order Hold method was implemented using MATLAB's *c2d* function [13], with the Zero Order Hold ('zoh') method [14] selected.

3.2 Triangle Approximation

This hold-based method is also referred to as a first-order hold and is similar to the Zero-Order Hold method discussed in Sect. 3.1. For the Triangle Approximation, the continuous signal, $x(t)$, is approximated in the sampling interval from $nT \leq t < (T + nT)$ by connecting adjacent sampled values $x(nT)$ and $x(T + nT)$ with a straight line. This method is related to linear interpolation, which is not a causal process in continuous time. However, following [12], the discrete-time equivalent process is causal, and is given by,

$$H(z) = \frac{\left(1 - z^{-1}\right)^2}{Tz} \mathcal{Z}\left[\frac{H(s)}{s^2}\right]. \tag{4}$$

In this work, the Triangle Approximation method was implemented using MATLAB's *c2d* function [13], with the First-Order Hold ('foh') method [14] selected.

3.3 Impulse Invariant Method

The Impulse Invariant method, as outlined in [15], produces a discrete-time transfer function whose impulse response is identical to the continuous-time impulse response at intervals of nT. The frequency response of the z-domain transfer function, however, is an aliased version of the frequency response of the s-domain transfer function.

To perform a transformation using the Impulse Invariant method, a partial fraction expansion (PFE) to first-order terms is performed to find the residues (numerators of first-order fractions after PFE) and poles of the continuous-time system. For the prototype s-domain filter specified by Eq. 2, the transfer function after the PFE takes the form,

$$H(s) = \frac{\frac{\tau_1\tau_2 - \tau_0\tau_2}{\tau_0\tau_1^2 - \tau_0\tau_1\tau_2}}{s + \frac{1}{\tau_1}} + \frac{\frac{\frac{\tau_2}{\tau_0} - 1}{\tau_2 - \tau_1}}{s + \frac{1}{\tau_2}}, \tag{5}$$

where τ_0, τ_1, and τ_2 are the time constants defined in Sect. 2.

The poles of the digital filter are defined by transforming the poles of the analog filter with the substitution,

$$z = e^{sT}. \tag{6}$$

The residues of the digital filter are defined to be the same as the residues of the continuous-time system scaled by a factor of T. The digital transfer function for this case is given by,

$$\hat{H}(z) = \frac{T\frac{\tau_1\tau_2 - \tau_0\tau_2}{\tau_0\tau_1^2 - \tau_0\tau_1\tau_2}}{z - e^{\frac{-1}{\tau_1}T}} + \frac{T\frac{\frac{\tau_2}{\tau_0} - 1}{\tau_2 - \tau_1}}{z - e^{\frac{-1}{\tau_2}T}}. \tag{7}$$

The first-order terms defined by these new poles and residues can then be added together to generate a new ratio of polynomials in the z-domain.

In this work, the Impulse Invariant method was implemented using MATLAB's *c2d* function [13], with the Impulse-Invariant Mapping ('impulse') method selected [14].

3.4 Bilinear Transform

In the bilinear transform, or Tustin approximation, the $j\omega$ axis in the s-plane is mapped to the unit circle in the z-plane. Because the $j\omega$ axis is infinite, and the unit circle is finite, this results in some frequency warping, meaning that behavior at one frequency in the continuous-time system may be shifted to a different frequency in the discrete-time system. This frequency warping is most pronounced near the Nyquist frequency. There is a modification to the bilinear transform known as pre-warping that allows perfect matching between the continuous and discrete-time filter responses at one

frequency. This can be useful if the filter has a significant feature at one frequency, such as the center frequency of a band-pass filter. In this work, the bilinear transform for the application of playback equalization is considered without the inclusion of pre-warping, though a method of pre-mapping the continuous time spectrum to minimize the frequency warping incurred by performing the bilinear transform is explored in Sect. 3.8.

The bilinear transform from the s-domain to the z-domain is accomplished by making the substitution,

$$s = \frac{2}{T}\frac{z-1}{z+1}. \tag{8}$$

For the purposes of this work, the Bilinear Transform method was implemented using MATLAB's *c2d* function [13], with the Tustin ('tustin') method selected [14].

3.5 Zero-Pole Matching

Zero-Pole Matching (also called the Matched-z Transformation [15]) is accomplished by mapping s-domain poles and zeros to poles and zeros in the z-domain directly. Each pole from the continuous system is mapped to a pole in the discrete system by the equivalency in Eq. 6 [12]. The s-domain zeros are also mapped to the z-domain by Eq. 6, with the exception of $s = \infty$, which is ignored in this analysis as it is not relevant to this application.

For the prototype filter, combining Eq. 6 with Eq. 1 gives,

$$\hat{H_0}(z) = \frac{\left(z - e^{\frac{-1}{\tau_0}T}\right)}{\left(z - e^{\frac{-1}{\tau_1}T}\right)\left(z - e^{\frac{-1}{\tau_2}T}\right)}. \tag{9}$$

The gain of the digital filter is then set to match the gain of the analog filter in Eq. 1 at 0 Hz, resulting in a discretized transfer function,

$$\hat{H}(z) = G\hat{H}_0(z), \tag{10}$$

where

$$G = \frac{H(s=0)}{\hat{H}_0(z=1)}. \tag{11}$$

In this work, the Zero-Pole Matching method was implemented using MATLAB's *c2d* function [13], with the Zero-Pole Matching Equivalents method ('matched') selected [14].

3.6 Complex Error Minimization

This method determines the numerator and denominator of the digital filter, $\hat{B}(z)$ and $\hat{A}(z)$ respectively, by minimizing the weighted squared error between the complex frequency response of the continuous filter, $H(j\omega)$, and the complex frequency response of the discrete filter, $\hat{H}(j\omega)$, given by

$$\frac{\hat{B}(j\omega)}{\hat{A}(j\omega)}. \tag{12}$$

This is accomplished in two stages. First, the weighted squared equation error is minimized. For a desired sample rate of f_s, this minimization takes the form,

$$\min_{\hat{B}_0,\hat{A}_0} \sum_{k=0}^{n-1} w_k |\hat{A}(j\omega_k)H(j\omega_k) - \hat{B}(j\omega_k)|^2, \tag{13}$$

where $\hat{B}_0$ and $\hat{A}_0$ are the numerator and denominator of the digital filter at this stage respectively, ω is a length-n vector of equally-spaced angular frequencies spanning from $\omega = 0$ rad/s to $\omega = \frac{n-1}{n}\pi f_s$ rad/s, w is a length-n array of weights, and k is an indexing variable. The minimization in Eq. 13 can be accomplished in a single iteration by solving a system of linear equations.

In the second stage of the algorithm, the results of the equation error minimization, $\hat{B}_0$ and $\hat{A}_0$, are used as the initial estimate for an iterative minimization of the weighted squared output error using the damped Gauss-Newton method. This minimization takes the form,

$$\min_{\hat{B}(z),\hat{A}(z)} \sum_{k=0}^{n-1} w_k |H(j\omega_k) - \hat{H}(j\omega_k)|^2. \tag{14}$$

For the purposes of this work, the Complex Error Minimization method was implemented using MATLAB's *invfreqz* function [16]. The maximum number of iterations was set to 100 and the gradient vector norm threshold for convergence was set to 0.001.

3.7 Magnitude Error Minimization

This method is similar to the Complex Error Minimization method with the exception that it takes in as its argument only the sampled magnitude response of the prototype analog filter, and then constructs a minimum-phase complex response around it using the cepstral method [17]. This is then fed into the Complex

Error Minimization algorithm described in Sect. 3.6. Compared to the Complex Error Minimization method, this method yields a discrete filter with a magnitude response closer to that of the prototype filter, at the expense of a larger error in the phase response.

3.8 Nyquist Band Transform

The Nyquist Band Transform (NBT) [18] uses conformal mapping to pre-map a continuous-time filter so that when it is discretized using the bilinear transform, the amount of frequency warping is significantly reduced when compared to using the bilinear transform alone.

As discussed in Sect. 3.4, the bilinear transform results in frequency warping because it compresses the infinite range of frequencies along the positive $j\omega$ axis in the s-domain, $[0,\infty)$ radians per second, to the finite range of frequencies along the positive half of the unit circle in the z-domain, $[0,\pi)$ radians per sample. The NBT attempts to address this by pre-mapping the Nyquist band in the s-domain so that $\omega = [0,\omega_0)$ maps to $\omega = [0,\infty)$, where ω_0 is one half of the desired sample rate in radians per seconds. The ideal result is that, after performing the bilinear transform, the Nyquist band in the s-domain maps to the Nyquist band in the z-domain.

A detailed description of the NBT algorithm can be found in Appendix A.

4 Oversampling

The filters produced by each of the discretization methods presented in this work suffer from some amount of error when compared to the continuous-time prototype filter. In the frequency domain, this error is most pronounced near the Nyquist frequency. This is particularly an issue for audio signals recorded at sample rates where the Nyquist frequency is near the upper limit of the audible band. One common method of mitigating this is to move the Nyquist frequency further away from the audible band by oversampling. In this approach, the audio signal to be filtered is first upsampled, then filtering is performed at this higher sampling rate, after which the signal is downsampled back to the original sample rate.

Oversampling can significantly reduce the error in the audible band; however, this improvement does not come without penalty. Oversampling introduces an increase in computational expense, both because the filter must be implemented over more samples, and because the resampling process itself requires computation. Additionally, resampling requires the implementation of anti-aliasing and anti-imaging filters. Performing this filtering with linear-phase, half-band FIR filters, as is common, introduces delay and pre-ringing into the time domain response. The pre-ringing and delay artifacts introduced by the oversampling process are shown in Fig. 1, which shows the sampled impulse response of the continuous-time prototype filter along with the impulse response of the filter after discretization with the bilinear transform using an oversampling factor of 2.

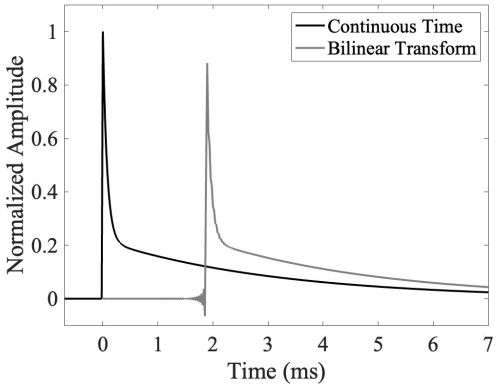

Fig. 1: The sampled impulse response of the continuous-time prototype filter along with the impulse response of a discretized version generated by the bilinear transform and an oversampling factor of 2. The anti-aliasing filters used in this work result in a delay of 1.87 ms for this configuration. The pre-ringing introduced by the oversampling process can be observed in the discretized filter impulse response just before the 2 ms mark.

In this paper, all oversampling was performed using linear-phase, half-band FIR interpolators and decimators, created with MATLAB's *designHalfbandFIR* function [19]. The details of the structure of these filters can be viewed in the companion MATLAB live script [20].

	Bark-Weighted RMSE (20Hz-20kHz)								
	Magnitude (dB)			Phase (°)			Complex		
Method	×1	×2	×4	×1	×2	×4	×1	×2	×4
Zero-Order Hold	0.51	0.124	0.0308	22.5	11.0	5.47	0.103	0.0498	0.0246
Triangle Approx.	1.41	0.261	0.0620	**0.509**	**0.015**	**8.04E-4**	0.0180	0.00431	0.00128
Impulse Invariant	1.14	0.475	0.217	21.2	10.8	5.48	0.145	0.0709	0.0349
Bilinear Transform	1.39	0.256	0.0607	0.791	0.187	0.0463	0.0176	0.00419	0.00125
Zero-Pole Matching	0.504	0.123	0.0305	22.5	11.0	5.47	0.103	0.0498	0.0246
Complex Error Min.	1.34	0.237	0.0556	1.11	0.259	0.0644	**0.0166**	**0.00389**	**0.00119**
Mag. Error Min.	**0.0584**	**0.027**	**0.00802**	17.2	8.17	4.07	0.0716	0.0362	0.0183
Nyquist Band Trans.	0.430	0.132	0.0321	15.1	6.99	3.46	0.123	0.0448	0.0174

Table 1: Magnitude, phase, and complex Bark-weighted mean squared error between the sampled continuous-time prototype filter and the discrete-time filters generated using each method. Each measurement was made at 1 Hz resolution over the bandwidth 20 Hz-20 kHz. The base sample rate in each case was 48 kHz. The factors above each column indicate the amount of oversampling that was implemented. ×1 indicates no oversampling, ×2 indicates a oversampling factor of 2 for a maximum sampling frequency of 96 kHz, and ×4 indicates a oversampling factor of 4 for a maximum sampling frequency of 192 kHz. The lowest error values in each column are bolded.

5 Results & Discussion

5.1 Error Weighting

A method for weighting error in the frequency domain was implemented such that perceptually equivalent deviations in the responses of the filters account for equivalent increases in the overall error reported. This was accomplished by assigning a weight to each frequency bin that is the inverse of the bandwidth of the Bark band in which that frequency falls. Bark, or critical, bands represent, "a subdivision more closely related to the manner in which the ear itself appears to carry out the process," compared to purely linear or purely logarithmic groupings [21]. This weighting scheme has the result that each Bark band has equal weight in the error calculations. A similar weighting system is recommended in [18] for use in the optimization of a free parameter that influences frequency warping. In this work, this weighting scheme is used in the evaluation of the magnitude and phase error in the results section, as well as in the minimizations required by the Complex Error Minimization and Magnitude Error Minimization methods. Additionally, this weighting is used in the optimization of the free parameter, γ, used in the Nyquist Band Transform method (see Appendix A).

5.2 Error Metrics

The prototype RIAA filter was discretized using each method described in Sect. 3. A base sampling rate of 48 kHz was chosen because the Nyquist frequency at this sample rate is close enough to 20 kHz for the error in the audible band to be pronounced. The frequency-domain performance of each discretization is characterized by three metrics. The accuracy of the magnitude response is calculated as,

$$RMSE_{mag} = \sqrt{\frac{\sum_{k=0}^{L-1} w_k \left(20\log_{10}\left[\frac{|H(j\omega_k)|}{|\hat{H}(j\omega_k)|}\right]\right)^2}{\sum_{k=0}^{L-1} w_k}}, \quad (15)$$

where L is the number of frequency bins to be analyzed, H is the sampled complex frequency response of the continuous-time prototype filter, $\hat{H}$ is the complex frequency response of the discretized filter, ω is a length-n vector of angular frequencies, w is a length-L array of Bark weights, and k is an indexing variable. To accommodate a 1 Hz resolution over the 20 Hz-20 kHz bandwidth, $L = 19,981$ is chosen.

The phase response error[1] is calculated as,

$$RMSE_{ang} = \sqrt{\frac{\sum_{k=0}^{L-1} w_k \left(\angle H(j\omega_k) - \angle \hat{H}(j\omega_k)\right)^2}{\sum_{k=0}^{L-1} w_k}}. \quad (16)$$

[1] Note that when reporting the phase response error for the oversampled tests, a linear phase offset was added to the discrete-time filter phase response to compensate for the group delay introduced by the FIR anti-aliasing and anti-imaging filters.

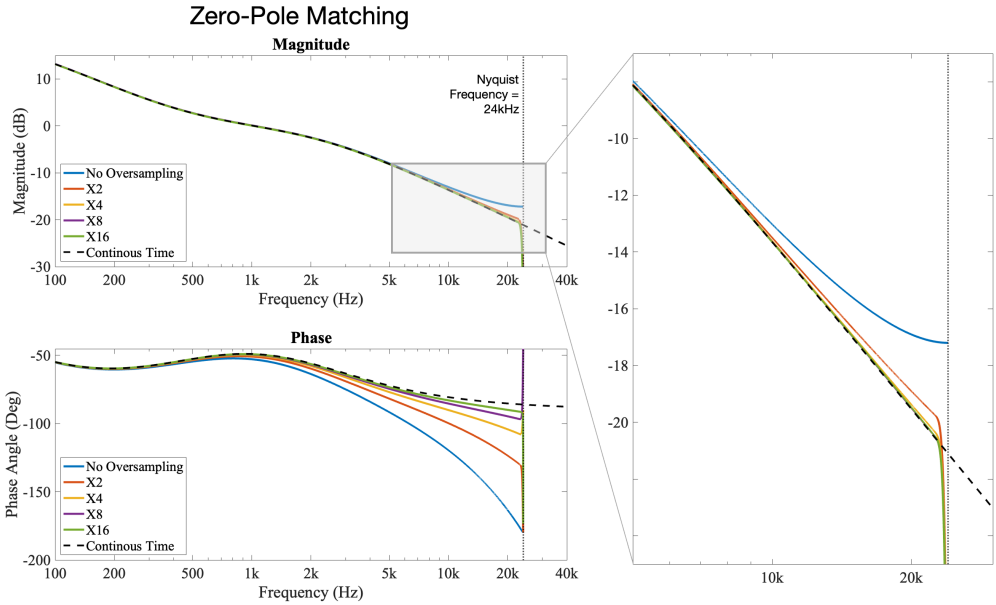

Fig. 2: Magnitude and phase responses of the discrete-time filters generated using the Zero-Pole Matching method with differing amounts of oversampling, along with the sampled magnitude and phase response of the continuous-time prototype filter. All discrete filters have a base sample rate of 48 kHz.

The complex response error is calculated as,

$$RMSE_{comp} = \sqrt{\frac{\sum_{k=0}^{L-1} w_k |H(j\omega_k) - \hat{H}(j\omega_k)|^2}{\sum_{k=0}^{L-1} w_k}}. \quad (17)$$

This metric equates to the square root of the mean of the squared Euclidean distance between the points in the complex plane.

5.3 Accuracy

The accuracy with which each discretization method reproduces the continuous-time response of the prototype filter as measured by the metrics above is presented in Table 1 for oversampling factors of $\times 2$ and $\times 4$ as well with no oversampling. The lowest magnitude, phase, and complex errors were achieved by the Magnitude Error Minimization, Triangle Approximation, and Complex Error Minimization methods, respectively. The first and third results are unsurprising as these methods iteratively minimize an error metric that is related to the ones that we are using for evaluation. While these methods are highly accurate, as discussed below, they are the most computationally expensive of the methods, and are therefore potentially unsuitable for some real-time applications. Of the non-iterative methods, the NBT exhibits the lowest magnitude error with no oversampling, and Zero-Pole Matching achieves the lowest error when oversampling is implemented. In terms of complex accuracy, of the non-iterative methods, the Bilinear Transform exhibits the lowest error. For each method, all three metrics are improved as the oversampling factor is increased.

The sampled response of the continuous-time prototype filter along with the responses of discrete-time filters generated using the Zero-Pole matching method at several levels of oversampling are shown in Fig. 2. From this figure, we can see that the responses are quite accurate in the lower band, and begin to fail as we approach the Nyquist frequency. There is also an improvement in accuracy as the amount of oversampling is increased. The effect of the anti-aliasing filters for the oversampled cases can be observed in the steep dropoff in the magnitude response at the Nyquist frequency. Although only the Zero-Pole matching method is shown, these general trends are repeated for all meth-

ods. Fig. 2 was generated with the companion MATLAB live script, which can be used to make similar figures for any of the presented methods [20].

5.4 Computational Efficiency

When a cutting curve is unknown or is in question, it is common that a restoration engineer will need to audition multiple playback equalizations. Under these conditions, it is beneficial for the operator to be able to adjust the filter parameters in real time during playback. For this reason, along with the accuracy with which each method reproduces the continuous-time response, we must also consider the computational efficiency of the algorithm. Quantifying the computational cost of each method is beyond the scope of this article, but we can say that, in general, the iterative approaches are likely to be the slowest, and methods that entail a simple substitution for s in terms of z, like the Bilinear Transform and Zero-Pole Matching methods, are likely to be the fastest. The companion MATLAB live script [20] includes a measurement of the time required to discretize the filters using each method for the system on which the script is running, which can provide a rough idea of the relative computational efficiencies of each approach.

6 Summary

In this article, we have surveyed eight methods for discretizing continuous-time systems, and explored each of them in the context of performing phonograph playback equalization in the digital domain. The results showed that the iterative methods with maximum oversampling provided the most faithful reproduction of the continuous-time magnitude and complex responses, but that methods with lower computational overhead can still generate very accurate filters. The intention of this article is not to determine which of these methods is best, but to quantify the compromises that each one entails in a way that might allow someone developing a system for playback equalization to make informed choices for their specific application and platform.

Each method presented here can be explored in detail using the interactive companion MATLAB live script found at `https://doi.org/10.60593/ur.d.26503432` [20].

References

[1] Galo, G. A., "Disc recording equalization demystified," *ARSC Journal*, 27(2), pp. 188–211, 1996.

[2] Read, O., *The Recording and Reproduction of Sound*, Howard W. Sams Co., Inc., 1952.

[3] Maxfield, J., *High Quality Recording and Reproduction of Music and Speech*, American Telephone and Telegraph Company, 1926.

[4] IEC 60098, "Analogue audio disk records and reproducing equipment," Standard, International Electrical Commission, IEC Central Office 3, rue de Varembé, CH-1211 Geneva 20, Switzerland, 2020.

[5] Galo, G. A., "The Columbia Lp Equalization Curve," *ARSC Journal*, 40(1), p. 40, 2009.

[6] Leister, W., "The London LP Curve: London LP Recording Characteristic and its Equalization in Playback," *ARSC Journal.*, 48(2), pp. 163–172, 2017.

[7] Copeland, P., *Manual of analogue sound restoration techniques*, British Library London, UK, 2008.

[8] Vad, J., *VADLYD MD12 MK4 78 RPM and RIAA Record Preamplifier*, JØRGEN VAD LYDTEKNIK, 2022.

[9] McINTOSH LABORATORY, I., *INSTRUCTION MANUAL McINTOSH MODEL C-8 and C-8p AUDIO COMPENSATORS*, McINTOSH LABORATORY, INC., 320 Water St., Binghamton, N.Y., U.S.A., 1955.

[10] Moyer, R. C., "Evolution of a recording curve," *Audio Engineering*, 37(3), pp. 19–22, 1953.

[11] Self, D., *Small signal audio design*, Focal Press, 2020.

[12] Franklin, G. F., Powell, J. D., Workman, M. L., et al., *Digital control of dynamic systems*, volume 3, Addison-wesley Reading, MA, 1998.

[13] MATLAB, "c2d: Convert model from continuous to discrete time," mathworks.com/help/ident/ref/dynamicsystem.c2d.html, n.d., from MATLAB help documentation.

[14] MATLAB, "Continuous-Discrete Conversion Methods," mathworks.com/help/ident/ug/continuous-discrete-conversion-methods.html, n.d., accessed 14-June-2024.

[15] Smith III, J. O., *Physical audio signal processing: For virtual musical instruments and audio effects*, 2010.

[16] MATLAB, "invfreqz: Identify discrete-time filter parameters from frequency response data," mathworks.com/help/signal/ref/invfreqz.html, n.d., accessed 14-June-2024.

[17] Smith, J. O., *Introduction to digital filters: with audio applications*, volume 2, Julius Smith, 2007.

[18] Darabundit, C. C., Abel, J. S., and Berners, D., "Nyquist Band Transform: An Order-Preserving Transform for Bandlimited Discretization," *Journal of the Audio Engineering Society*, 70(9), pp. 674–689, 2022.

[19] MATLAB, "designHalfbandFIR: Design and implement halfband FIR filter," mathworks.com/help/dsp/ref/designhalfbandfir.html, n.d., accessed 18-June-2024.

[20] Thompson, B. R. and DiPassio, T., "Companion MATLAB Live Script for A Survey of Methods for the Discretization of Phonograph Record Playback Filters," 2024, doi:10.60593/ur.d.26503432.

[21] Zwicker, E., "Subdivision of the audible frequency range into critical bands (Frequenzgruppen)," *The Journal of the Acoustical Society of America*, 33(2), pp. 248–248, 1961.

APPENDIX

A Nyquist Band Transform Algorithm

The NBT is accomplished in four steps. In the first transform, the region $\omega = [0, \omega_0)$ is mapped back to itself, and also flipped and stretched to span the region $\omega = [\omega_0, \infty)$. In this mapping, dc is mapped to dc and ∞, and ω_0 is mapped to itself [2]. This transform takes the form,

$$G_1(s) = \frac{-2\omega_0^2 s}{s^2 - \omega_0^2}. \tag{18}$$

For our prototype filter from Eq. 2, the first stage is,

$$\mathbf{A}_1 \mathbf{c} = \mathbf{k}, \tag{19}$$

where $\mathbf{A}_1$ is the first transform in matrix form,

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & -2\omega_0^2 & 0 \\ 4\omega_0^4 & 0 & -2\omega_0^2 \\ 0 & 2\omega_0^4 & 0 \\ 0 & 0 & \omega_0^4 \end{bmatrix}, \tag{20}$$

$\mathbf{c}$ is a vector of the numerator or denominator coefficients of the prototype continuous-time filter,

$$\begin{bmatrix} c_2 \\ c_1 \\ c_0 \end{bmatrix}, \tag{21}$$

and $\mathbf{k}$ is a vector of the numerator or denominator coefficients of the resulting filter,

$$\begin{bmatrix} k_4 \\ k_3 \\ k_2 \\ k_1 \\ k_0 \end{bmatrix}. \tag{22}$$

All stages of the NBT prior to the fourth stage, the bilinear transform, are carried out independently for both the numerator and the denominator of the transfer function. The subscripts of the coefficients indicate the power of s in the term they are associated with; i.e., k_4 is the coefficient for the s^4 term, etc. For the numerator of our prototype filter, an s^2 term, $c_2 = 0$, has been added so that the number of terms in the numerator is the same as the number of terms in the denominator. Notice that the output of this transform is a filter with numerator and denominator orders of $2N$, where N is the order of the prototype filter.

The filter that results from the first transform is neither stable nor minimum phase. To address this, in the second stage of the NBT, any poles and zeros that fall on the right side of the s-plane are reflected over the $j\omega$ axis, resulting in a stable, minimum phase filter. This process generates a new vector of numerator or denominator coefficients, $\mathbf{m}$, with the same form as Eq. 22.

[2] In the original work [18], a distinct truncation frequency, ω_c, and a distinct critical frequency, ω_0, are specified. For our purposes, we will set $\omega_c = \omega_0 = \pi f_s$ rad/s, where f_s is the desired sampling frequency in Hz.

In the third stage of the NBT, the region $\omega = [0, \omega_0)$ is mapped to $\omega = [0, \infty)$. This is accomplished by the *inverse* second transform. The *forward* second transform is defined as,

$$G_2(s) = \frac{\gamma s}{s^2 + \omega_0^2}, \tag{23}$$

where γ is a free parameter that affects frequency warping. The value of γ can be tuned to perfectly map the continuous-time response at a single frequency to that same frequency in discrete time in the same way that pre-warping affects the result of a bilinear transform. As with the bilinear transform, this can be useful if there is a specific frequency of interest. Alternatively, γ can be set to reduce frequency warping across the spectrum. In this paper, γ was determined by minimizing the Bark-weighted mean squared error (see Sect. 5) between the original frequencies and the frequencies as mapped by the NBT from 20 Hz to 20 kHz. Values of γ for each desired sample rate were determined this way.

For our prototype filter, the second forward transform in matrix form, $\mathbf{A}_2$, is,

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & \gamma & 0 \\ \gamma^2 & 0 & 2\omega_0^2 \\ 0 & \omega_0^2\gamma & 0 \\ 0 & 0 & \omega_0^4 \end{bmatrix}. \tag{24}$$

To obtain the inverse second transform, the Moore-Penrose inverse of $\mathbf{A}_2$ is computed to generate $\mathbf{A}_2^+$. The third stage is then,

$$\mathbf{A}_2^+ \mathbf{m} = \mathbf{p}, \tag{25}$$

where $\mathbf{p}$ is a new vector of numerator or denominator coefficients with the same form as Eq. 21.

In the final stage, a standard bilinear transform is performed on the pre-mapped continuous-time filter to obtain the discretized filter.