ASSESSING NONLINEARITY AND MEMORY EXTENT IN AUDIO SYSTEMS

by

Ethan Randall Hoerr

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Electrical Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

July 22, 2021

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

ABSTRACT

Creating digital models of existing audio devices is useful for increasing access to audio effects and for preserving audio history. In the work covered by this dissertation, we investigate the use of Volterra series modeling to assess the degree of nonlinearity of a system and time-delayed mutual information (TDMI) to estimate the length of the recovered impulse response.

Using an arctangent function as an example system, comparing empirically generated Volterra series models containing anywhere from first- to fourth-order system kernels revealed that including the odd-ordered first and third kernels yielded the best-performing model. We propose that this benchmarking method can aid a system modeler by elucidating details about a system's nonlinear behavior.

We also assess the utility of time-delayed mutual information (TDMI) as a method for revealing which samples of a recovered impulse response of a nonlinear system are significant. As a nonlinear metric of correlation between an input signal $x[n]$ and output signal $y[n]$, the TDMI method in MATLAB simulations accurately predicted the significant samples of delay lines, FIR moving average filters, and Schroeder all-pass filters. The TDMI method was less informative when applied to IIR low pass filters with and without an arctangent function appended to the output.

Finally, we applied the TDMI approach to a real-world audio device, a distortion effects pedal designed for electric guitar players. In the presence of increasing nonlinear distortion, the calculated TDMI curve took the shape of a pronounced peak starting at $\tau = 0$ samples delay between $x[n]$ and $y[n]$, with τ increasing as the distortion increased. A similar phenomenon was observed when lowering the pedal's low-pass filter cutoff frequency from 36.7 kHz to 620 Hz; in the 620 Hz test, the TDMI peak was significantly lower than the other test cases and featured a more gradual decay to the estimator bias noise floor.

In summary, we demonstrated that Volterra series models are useful for assessing the degree of nonlinearity of a system and that time-delayed mutual information can inform which samples of a recovered impulse are significant. Both of these insights can aid in deciding how many Volterra series kernels and how much kernel memory to include when creating a black box system model.

CHAPTER ONE

INTRODUCTION

Creating digital models of audio devices has myriad applications in the world of music production and performance. With the ability to use hardware devices outside of their original context, producers, engineers, and musicians can replicate the characteristics of an analog device in the form of a software plug-in or embedded hardware device. This enables the user to easily access signal processing chains that might otherwise require physically acquiring hard-to-find or obsolete hardware. Often, one desirable property of analog devices to be modeled is the presence of nonlinear distortion, such as in overdrive or fuzz guitar effects, or in more subtle forms such as tape saturation in analog tape delay units. The focus of this dissertation, while not performing actual modeling of analog hardware devices, is to investigate the use of two techniques in diagnosing the nonlinear behavior of audio systems. The intent is that information gleaned from such diagnostic techniques could aid a system modeler using a black-box approach where assumptions about the system are made, compared to "white-box" approaches that require meticulous emulation of a given device's circuit behavior.

To motivate the work that follows, we provide examples of commercial audio products that take advantage of digital modeling, such as the Roland TB-03 "Bass Line" synthesizer, Strymon's dTape analog tape modeling technology, and the Sintefex Replicator effects unit. We then provide an overview of the contribution of our work.

The Roland TB-03 is a 2016 remake of a short-lived yet widely popular bass synthesizer, the TB-303; see Figure 1.1. Originally marketed in 1981 along with the TR-606 drum machine, the TB-303 was designed for solo guitarists who wanted to easily add synthetic

(a) Original Roland TB-303 "Bass Line" synthesizer, introduced in 1981.

(b) Roland TB-03, boutique recreation of the original TB-303 device introduced in 2016.

Figure 1.1: Roland TB-303 (a) and TB-03 (b)

yet realistic bass guitar accompaniment to their performances. Due to its unpopularity for this intended purpose, only 10,000 units were produced before manufacturing ceased in 1984. However, the TB-303 experienced a resurgence in interest in techno and house music production, particularly due to its synthetic, squelching bass sounds as used in the artist Phuture's 1987 acid house track, "Acid Tracks" [23]. Today, due to the TB-303's rarity and high resale value, demand for devices that perform similarly drove numerous independent manufacturers over the years to replicate the functionality of device, with at least 13 such "clones" being introduced as of the writing of this document [1, 3]. Eventually, the original manufacturer Roland introduced the TB-03 in 2016, a boutique model of the original device that uses their proprietary Analog Circuit Behavior (ACB) modeling framework in an embedded hardware context to closely recreate the TB-303's original functionality. While Roland has not released technical details regarding exactly how the ACB technology works, their marketing materials reflect the objective of closely replicating the behavior of the original analog circuits [30].

The Strymon El Capistan and Timeline delay effect pedals (Figure 1.2) are two commercial audio devices that mimic the behavior of vintage tape delay units that were common in music production during the 1960's and 1970's, such as the Roland Space Echo

(a) Strymon El Capistan dTape Echo effects pedal.

(b) Strymon Timeline effects pedal, featuring the dTape algorithm among several other delay algorithms.

Figure 1.2: Strymon El Capistan (a) and Timeline (b) delay effects.

and Maestro Echoplex. Instead of choosing a single device to model, the software designer distilled many desirable properties of the tape delay units into a proprietary framework known as Strymon dTape Technology [6]. With a completely digital model of a custom tape delay unit, the user is given control over several parameters inherent to the sonic character created by analog tape delays, such as tape speed, feedback, and magnetic record and playback head configuration. Also accessible are controls for simulated parameters would be difficult to modulate in real-time with a vintage tape delay, such as tape degradation, wow and flutter, and record head bias level. Finally, other convenient features including the saving of the current device as a preset, and can quickly synchronize the tape speed to a beat by tapping in the timing with a footswitch. Thus, while vintage tape machines are still coveted and provide unique sonic characteristics, digital modeling of these devices allows a wider audience of people to use these effects, and makes them significantly more user-friendly.

The Sintefex FX8000 Replicator (Figure 1.3) is our final example of an audio modeling

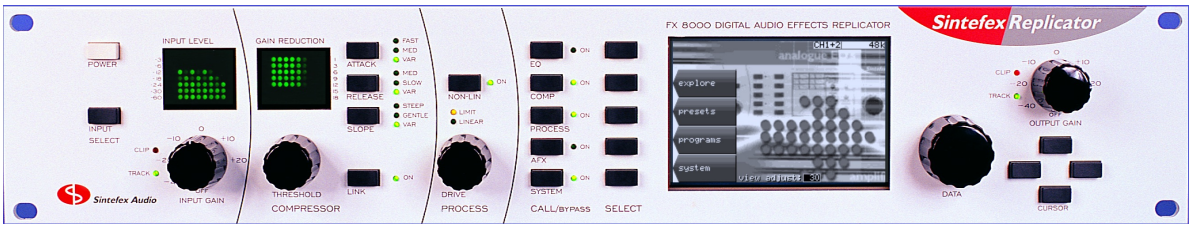


Figure 1.3: Sintefex FX8000 Replicator front panel.

product, which uses "Dynamic Convolution" to emulate the behavior of analog audio devices such as equalizers, compressors, channel strip preamplifiers, guitar amplifiers, and even the audio characteristics of analog tape running at a given speed [13, 14]. In order for the nonlinear nature of these devices to be modeled, unit impulse signals over a range of amplitudes are passed through the target device and saved into memory. When the resulting Dynamic Convolution model processes an audio signal, the underlying algorithm will choose one of the measured impulse responses based on the amplitude of the incoming sample. That is, the algorithm will convolve the input signal with the impulse response for which the amplitude of the analysis signal most closely matches that of the incoming audio sample. Although this product is now long out of production, the same underlying technology has been licensed for use in products manufactured by third parties, such as the Focusrite Liquid Mix (Figure 1.4). According to Sintefex, the Liquid Mix is a

> "...unique DSP based Firewire tool that recreates up to 32 channels of classic analogue equalisers and compressors for inclusion into your desktop project without loading the CPU. The unique Dynamic Convolution and analogue sampling technology brings true to life emulations of professional analogue equipment. [2]"

The remainder of the dissertation is organized as follows. In Chapter 2 we give an overview of an existing modeling technique using Wiener and Volterra series kernels and our

Figure 1.4: Focusrite Liquid Mix desktop 32-channel equalizer and compressor processor, using Dynamic Convolution technology under license from Sintefex Audio.

application of this technique for assessing the degree of nonlinearity of a given audio system. Then, in Chapter 3 we introduce time-delayed mutual information (TDMI) and assess its sensitivity to the presence of memory in FIR and IIR digital filter simulations. Finally, in Chapter 4 we apply the TDMI method from Chapter 3 to a real-world analog audio device with filtering and nonlinear distortion, the ProCo Rat 2 distortion effects pedal. Conclusions and future work directions are noted in Chapter 5.

CHAPTER TWO

USING VOLTERRA SERIES MODELING TECHNIQUES TO ESTIMATE NONLINEARITY

Note: material in this chapter is based on the author's submission for the 2019 Audio Engineering Society's 147th Convention, Paper 10225 [11], accepted based on a peer-reviewed abstract and 750-word precis.

2.1 Abstract

Digital models of various audio devices are useful for simulating audio processing effects, but developing good models of nonlinear systems can be challenging. This chapter reports on determining attributes of black-box audio devices using Volterra series modeling techniques. In general, modeling an audio effect requires determination of whether the system is linear or nonlinear, time-invariant or -variant, and whether it has memory. For nonlinear systems, we must determine the degree of nonlinearity of the system, and the required parameters of a suitable model. We explain our work in making educated guesses about the order of nonlinearity in a memoryless system, and then discuss the extension to nonlinear systems with memory.

2.2 Introduction

With the proliferation of digital signal processing in both software and embedded hardware implementations, emulating existing audio systems is a natural curiosity. There is no shortage of evidence to support this: consider the contemporary work on modeling nonlinear audio systems such as distortion pedals, guitar amplifiers, and audio limiters [4, 9, 26]. If an audio system of interest exhibits linear, time-invariant (LTI) behavior,

methods such as periodic impulse excitation, maximum-length sequences, and time-delay spectrometry exist to represent the system with either its impulse response or frequency response function [8, 16, 24]. However, many interesting audio devices exhibit nonlinear and/or time-varying properties, rendering the well-known LTI techniques ineffective. In particular, this chapter is concerned with nonlinear system modeling considerations.

In theory, the ability to abstract audio devices sounds convenient, but what goes into the process of determining a digital model? The process of replicating a given system depends on our knowledge of its inner workings. At one extreme, if we completely know the electronic circuit of a device we can create a parametric or "white box" model of the effect; on the other hand, if we know nothing about the device, we employ a variety of "black box" modeling techniques, which mandate a degree of assumptions [9]. Supposing an optimal solution exists within our chosen model framework, we seek to minimize the error between the actual and modeled system. To improve our model it is helpful to identify basic system properties such as memory, linearity, time-invariance, and causality. Assuming time-invariance, stability, causality and no memory, by benchmarking the audio device-under-test (DUT) against a linear Volterra series model of the system, we hope to glean information regarding the system's degree of nonlinearity. With this information, we seek an informed approach to determining a Volterra series model resulting in a lower overall mean-square-error compared to the DUT.

The organization of this chapter is as follows. First, we provide a review of basic properties from systems theory, providing context when dealing with audio systems. Second, we present a summary of approximating audio systems with Volterra series models, and discuss considerations when modeling causal, stable, time-invariant, and memoryless systems that may be nonlinear. Then, we generate linear and nonlinear models of known systems in an attempt to establish rules of thumb in deciding degree of nonlinearity. Finally, we summarize the practical considerations and limitations of this approach and how it may

apply to real-world systems.

2.3 Black Box Model Considerations

To aid discussion of system modeling, we first review the key system properties of memory, linearity, time-invariance, causality, and stability from [18]. We wish to contrast the conveniences of modeling linear time-invariant systems with the difficulties of modeling nonlinear systems. If a system is linear and time-invariant (LTI), it can be represented by an impulse response or frequency response function. However, we will see that this framework breaks down if linearity is violated.

2.3.1 Basic System Properties and Their Implications

Memory In a system with memory, there exists some mechanism to retain energy or information such as a capacitor, kinetic energy, or digital memory. The instantaneous output of a system with memory may be a function of the present input as well as past or future inputs. In contrast, the output of a memoryless system is only dependent on the input at the present time. Example audio systems with memory include reverberation units or a filter, whereas a memoryless system could be a simple gain block.

Linearity A linear system obeys the superposition principle [18]. When considering sinusoidal input signals, a linear system's output will be some linear combination or superposition of the input frequencies, whereas a nonlinear system may include nonlinear combinations of the input frequencies. A fuzz pedal and tube amplifier are known to exhibit nonlinear behavior, manifesting as harmonic distortion [33].

Time-invariance In a time-invariant system, a time-shift of the input signal results in the same time-shift of the output signal. Intuitively, the state or settings of a time-varying system vary with time. While a wah-wah guitar effects pedal set to a fixed position can be viewed as a time-invariant system, modulating the pedal position invokes the time-varying

nature of the system.

Causality A causal system only operates on present and past inputs. All physical systems are causal as they cannot anticipate future inputs. A real-time pitch shifter is a causal system due to only operating on present and past inputs, whereas a real-time implementation of an audio time-reversal would require knowledge of future inputs that have not yet occured.

Stability A system is stable if inputs with bounded amplitude produce outputs whose amplitude is also bounded; conversely, given bounded inputs, an unstable system may produce outputs with unbounded amplitude. In an orientation prone to feedback, a microphone connected to an amplifier and loudspeaker is example of an unstable system resulting in an unbounded growth of the output amplitude. A stable audio system could be the impulse response of a room, whose total acoustic energy decays with each wave reflection/absorption.

2.3.2 Audio System Assumptions

In this work we make the assumption that our systems are causal, stable, time-invariant, and memoryless, but may be linear or nonlinear. In the case of a nonlinear system with memory, a suitable black box framework is the Volterra series representation [19]. An example memoryless nonlinear system is that of a Taylor series expansion [33]; we demonstrate that this type of system may be modeled by the Volterra series, which is essentially a Taylor series with memory [15].

2.4 Volterra series modeling approach

Schetzen [25] gives an illuminating presentation of the Wiener and Volterra theories of nonlinear systems, which are both ways of relating the output of a time-invariant system $y(t)$ to an input $x(t)$. While it is possible and valid to produce a Wiener or Volterra

series model given a characteristic equation for a known system, our concern here is with the empirical determination of a "black box" Volterra series model based on a cross-correlation measurement technique, known as the Lee-Schetzen method. In brief, this method exploits statistical properties of white Gaussian noise to isolate and measure Wiener kernels corresponding to each degree of nonlinearity of a system. It is important to note that the measured Wiener kernels are dependent on the variance of the identification signal. The Wiener kernels can be thought of as impulse responses of various dimensions from zero (DC term), one (linear impulse response), up to infinity. The nonlinear behavior of a system is captured by the n-dimensional impulse responses for $n \geq 2$. We then obtain the homogeneous Volterra kernels by combining all Wiener kernels of the same order. The total response of a system can be approximated by infinite order and memory of Wiener/Volterra kernels, but in practice we must truncate order and memory. The Volterra series representation does have limitations; for example, it is not suited for modeling extreme nonlinearities such as hard clipping [8, 33].

The aforementioned Lee-Schetzen method is a useful tool in determining the Volterra kernels of a black box system, and the technique is still being improved upon [19]. For example, the authors in [20] apply their novel multivariance approach to determine models of tube amplifiers, demonstrating the method's improved root-mean-square error (RRMSE) when compared to the single-variance approach of the classic Lee-Schetzen method [22]. Whereas this novel multivariance technique yields lower error measurements by a different approach to the generating a Volterra series model, we focus on improving the single variance approach by establishing guidelines for choosing model order and memory length. Our hope is that by combining heuristics regarding model order and memory length with either the single- or multi-variance approach, we can produce more accurate models.

2.4.1 Volterra Series Model Parameters

Once again, the systems we are concerned with here are causal, stable, time-invariant, and memoryless, with no assumption made about linearity. When applying the cross-correlation technique to determine a Volterra series model, we have several parameters to choose: length of identification signal, identification signal variance, Volterra series model order, and model memory.

Input signal length is important as the Volterra theory is closely linked to the Wiener theory of nonlinear systems. Measurement of the Wiener kernels, which the Volterra kernels are based on, relies on Wiener G-functionals that are orthogonal when the input signal has a white Gaussian distribution [25]. In practice, the longer our test signal, the closer we approach an ideal Gaussian distribution, resulting in better kernel estimation.

Input signal variance results in the excitation of different levels of nonlinearity. Consider an arctangent function: for low input amplitudes, the function may be approximated as linear. As input amplitude increases, the nonlinear behavior of the function manifests as the output is no longer a homogeneous scaling of the input. Thus, the choice of input variance of the identification signal will affect the linear or nonlinear excitation of the system.

Model order refers to the highest-order Volterra kernel we wish to identify. We want to strike a balance between low orders and low model error, as computational cost increases exponentially with additional higher-order kernels [20].

Model memory refers to the energy or information storage aspect of the Volterra series model. A system with finite memory may be compared to a system with a finite impulse response. For a system with a memoryless nonlinearity, we only need one value for each kernel, as the output is only dependent on the instantaneous input. This is akin to a finite impulse response with only one coefficient. For systems with memory, determining kernel memory is a necessary step to optimizing our model. As mentioned in the previous subsection, kernels may be thought of as the n-dimensional impulse response of the system.

Capturing the total impulse response for all dimensions is important for preserving the nature of the nonlinear system, although in practice some tradeoff must be made between model size/dimension and computational performance.

2.4.2 Determining system properties given a Volterra series model

As mentioned at the beginning of this section, modeling nonlinear audio systems with the Volterra series is well-known. In this subsection we will discuss how to make educated guesses about the linearity and memory of the DUT, which influence our choices when creating a Volterra series model. Again, we assume only that the system be causal, stable, time-invariant, and possibly nonlinear. Given some known stimulus, we use the relative root mean square error metric (RRMSE) in Equation 2.1 to compare the responses of the DUT and our model, y_i and $\hat{y}_i$, respectively. Qualitative metrics such as human factors listening tests are also important, but are beyond the scope of this work.

$$RRMSE = \frac{\sqrt{\sum_{i=0}^{N} (y_i - \hat{y}_i)^2}}{\sqrt{\sum_{i=0}^{N} (y_i)^2}} \tag{2.1}$$

We first wish to answer the question, "is the system linear?" By comparing the harmonic content of the input to and output from the DUT, total harmonic distortion (THD) is a familiar way to check whether our system adheres to superposition, as a linear system should have no harmonic distortion [33]. Our proposed approach is to create various linear Volterra series models of the reference system, and to use (2.1) to quantify model performance against the actual system. Because measured Volterra kernels depend on the variance of the measurement signal, we will use multiple variances to generate multiple models. We predict that as RRMSE values increase for a given model, the linear model becomes worse at approximating the nonlinear behavior of the reference device.

Then, if the system appears to be nonlinear we ask, "to what extent is the system

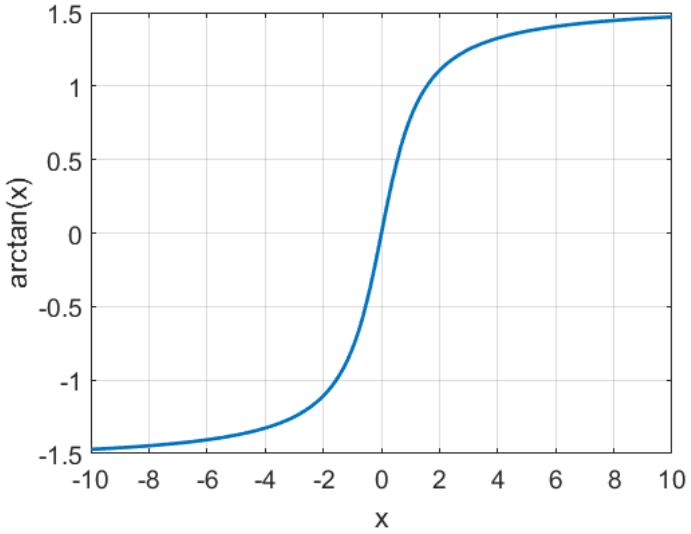


Figure 2.1: Plot of arctan(x) for $x = [-10, 10]$. This sigmoidal function may be approximated linearly for small inputs x, but exhibits increasingly nonlinear behavior as x increases.

nonlinear?" For example, if we know our DUT is a square-law operator, we should only need up to a second-order Volterra series model. With black box systems we may lack this precise knowledge, however. By creating multiple linear models of the DUT using increasing input signal variance, we can get a rough idea of whether increased variances are exciting nonlinearities in the system.

Orcioni, et al. [20] offer guidelines to determine the memory extent of an audio amplifier, which has a symmetric distribution of kernel values about some peak. To summarize the procedure, the first-order kernel is identified using a large memory value due to low computation cost. Then, the significant kernel values are kept by truncating values below a noise threshold. Higher-order kernels also have truncated memories based around the peak of the first-order kernel, to reduce computational complexity. We focus only on memoryless nonlinearities here; the extension to systems with memory remains a future development.

2.5 Experiment: Nonlinear, memoryless system

As mentioned, we seek to establish some guidelines for assessing the degree of nonlinearity of a system by conducting experiments with a known memoryless nonlinearity. Our hope is that by developing insights from a known system in a controlled context, we will have more confidence in approaching unknown, real-world black box systems.

For our memoryless, time-invariant nonlinear system we choose $y = \arctan(x)$. The response of this sigmoidal function becomes increasingly nonlinear as input amplitude increases; see Figure 2.1. Though our reference system is primitive compared to a real physical system, we feel this choice is justified as we see the contemporary use of the hyperbolic tangent function–also a sigmoidal function–for emulating the nonlinear behavior of distortion and guitar amplifier audio systems [9]. With a reference system decided, we next choose our experimental parameters: Volterra series model order, input signal duration in samples, input signal variances, and validation signal variances.

2.5.1 Experiment parameters

The first parameter we control is the Volterra series model order, or degree of nonlinearity of the model. We begin by creating a first-order (linear) Volterra series model of the system, then subsequently increase the order of our models by one until we reach a fourth-degree model. This is to see if adding nonlinear terms offers improvement over a purely linear approximation.

Secondly, input signal duration is important because the measurement of Wiener and thus Volterra kernels depends on identification signals with ideal white Gaussian distributions [25]. Practically speaking, we can never get a perfect white Gaussian distribution with finite-length sequences, but the approximation improves as we include more samples. We choose 10^6 samples for our input signal length, as this seems to be a good balance between

computation time and nearly white Gaussian distribution of our finite data.

Our third parameter is the variance of our white Gaussian identification signal, σ_{id}^2. This parameter directly influences our measured kernel values. For instance, from [25] consider the expression for the first-order kernel $k_1(\tau)$ as a function of σ_{id}^2 and the time-average of the reference system response $y(t)$ and "n-dimensional delayed input" $y_1(t)$:

$$k_1(\tau) = \frac{1}{\sigma_{id}^2}\overline{y(t)y_1(t)}. \tag{2.2}$$

Higher-order kernels are also variance-dependent; see [22] for a thorough discussion of such expressions. To model the reference system under various input conditions, we will use a logarithmic range of σ_{id}^2 values from 0.01 to 5.

Fourth and finally, the variance parameter σ_{val}^2 refers to the variance of the signals used to validate each model. As noted in [19], a Volterra series kernel performs best when the input signal has a variance nearly that of the signal used to measure the kernel. RRMSE values will be computed for each combination of σ_{id}^2 and σ_{val}^2; we expect to see local minima in RRMSE where these variances are a close match. Our validation signal variance ranges logarithmically between 0.001 and 50; note that this range covers more variances than than the identification signal variance, to simulate testing models with variances beyond those expected.

2.5.2 Experiment method

A single test case consists of a choice of model order, identification signal variance σ_{id}^2, and validation signal variance σ_{val}^2. Our output from each test case is an RRMSE which compares our reference system against the generated model with the given order, σ_{id}^2, and σ_{val}^2. Within our range of experimental parameters we generate Volterra series models with the Lee-Schetzen cross-correlation method by adapting code provided in [19].

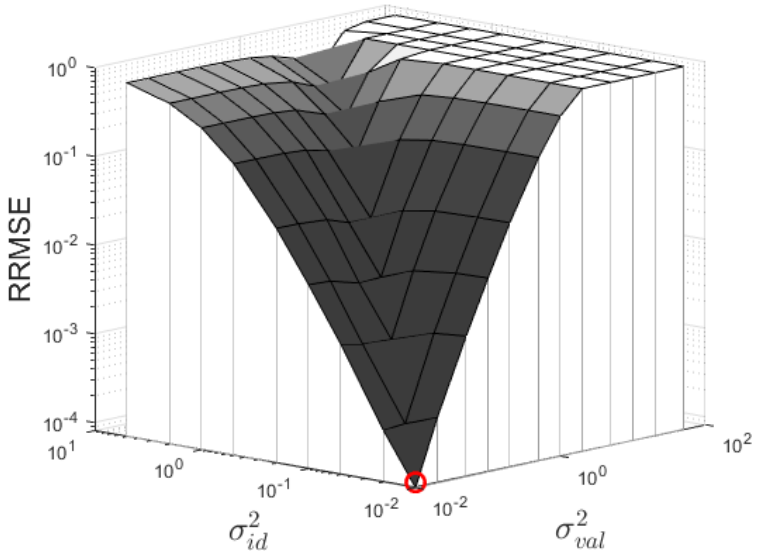


Figure 2.2: RRMSE values for linear approximations of arctan(x), each obtained as a first-order memoryless Volterra kernel. A minimum RRMSE of 8.17×10^{-5} occurs where σ^2_{id}, σ^2_{val} = 0.01 (circled).

2.5.3 Experimental results: Linear model

In Figure 2.2 we see that the global minimum of our linear model occurs where input signal variance $\sigma^2_{id} = 0.01$ evaluated with a signal variance $\sigma^2_{val} = 0.01$. This point is marked with a circle; the RRMSE here is 8.17×10^{-5}. As we increase the validation signal variance, moving along the axis labelled σ^2_{val}, the RRMSE increases. This is because higher variances excite more nonlinear behavior of the function compared to the lower variance used to measure the linear Volterra kernel. Note that beyond a certain threshold, we manually clipped the plot where RRMSE ≥ 1. Errors greater than or equal to 1, according to our RRMSE equation (2.1), suggest a poor-fitting model. Corroborating with the findings in [19], an underside view of Figure 2.2 (not shown) indicates local minima where $\sigma^2_{id} = \sigma^2_{val}$. This phenomenon suggests that a better linear approximation exists for validation input signals whose variance is greater than that corresponding to the absolute minimum RRMSE.

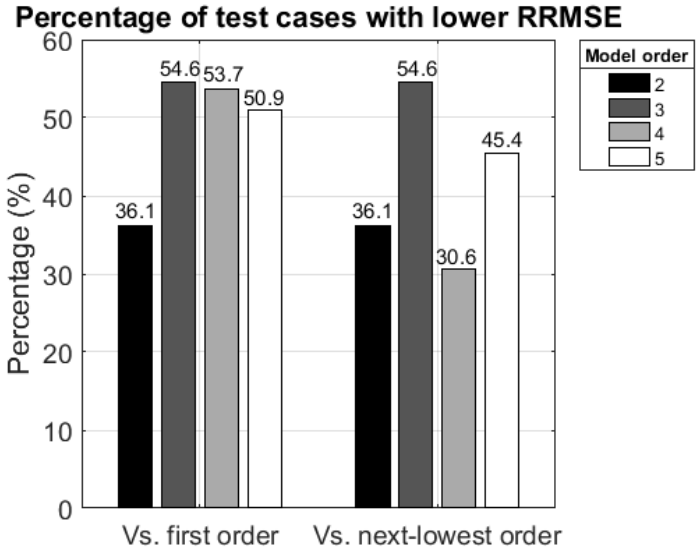


Figure 2.3: Percentage of test cases with lower RRMSE compared against the linear model and next-lowest order model. Model order indicated by color code in legend.

2.5.4 Experimental results: Nonlinear models

With linear models established for each σ_{id}^2, we proceed to identify 2nd- through 5th-order nonlinear Volterra series models. Instead of focusing on where global minima occur in each model, we are more concerned with how often the nonlinear model performs better than the linear model. As we create models that include higher-order Volterra kernels, we want a way to benchmark against our linear model.

For each test case (that is, each combination of σ_{id}^2 and σ_{val}^2), we first find the RRMSE for the various-ordered models. Then, we compare this RRMSE, $RRMSE_{Nonlinear}$, with that of the linear model, $RRMSE_{Linear}$. This results in a ΔRRMSE measure; see Equation 2.3. Negative ΔRRMSE values are better, indicating that the nonlinear model error is less than the linear model error. To indicate how often a nonlinear model performs better, we compute the percentage of test cases with negative ΔRRMSE's, shown in Figure 2.3. To get a broad idea of how much the nonlinear model improves upon the linear model, we take the

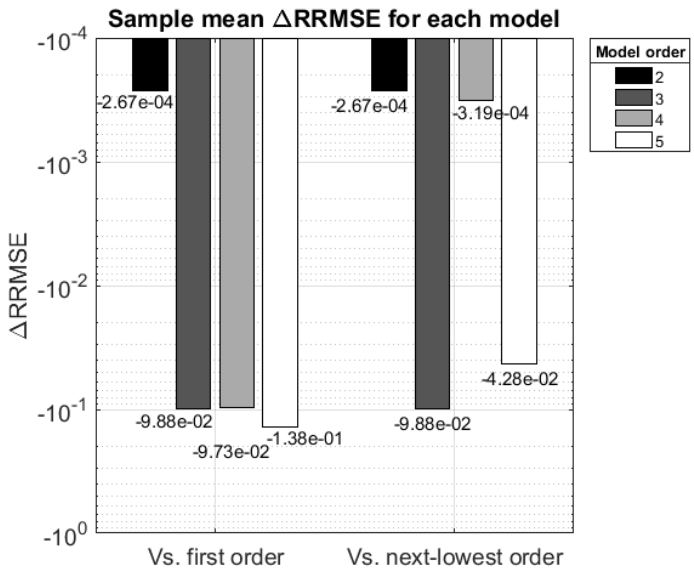


Figure 2.4: Sample mean of negative ΔRRMSE test cases for each nonlinear model, compared against the linear model and next-lowest order model. Model order indicated by color code in legend.

sample mean of all negative ΔRRMSE's. For each nonlinear model comparison, we provide identical difference metrics against both the linear model and the next-lowest degree model's RRMSE values, respectively shown in the left and right bar groups. For example, the 3rd-degree model is compared against both the linear model and the 2nd-degree model, and so on. This helps give an idea of the relative performance of each subsequent model.

$$\Delta RRMSE = RRMSE_{Nonlinear} - RRMSE_{Linear} \tag{2.3}$$

2.6 Discussion

From the left bar groups in Figures 2.3-2.4 we can see that adding a 2nd-order term offers little RRMSE improvement over the linear model, on the order of 10^{-4}. We see that

both the 3rd- and 4th-order models give a ΔRRMSE of about -0.09, while the 5th-order model has a sample mean of -0.14 in the cases where it did better than the linear models. Because the improvement upon the linear model offered by the 3rd and higher models is so similar, in the right bar group we show a comparison of each model order against the next-lowest order model. Although the sample mean of the ΔRRMSE for the 3rd- and 4th-order models was similar, we see that adding the 4th order term offers little improvement over the 3rd-order term, on the order of 10^{-5}. The ΔRRMSE for the 2nd-order model is the same in both bar groups because of the comparison against the linear model in both cases. The disparity between the 2nd- and 3rd-order model performance highlights the benefit of adding the 3rd-order term. Thus, if given the choice between model orders 1-5, we suspect that a 3rd-order model may best approximate the reference system. This is likely due to the Taylor series approximation for arctan(x) having only odd-order terms.

Taking a step back, the process of determining the various-ordered, memoryless Volterra series models of the reference system amounts to empirically determining a generic Taylor series approximation of the form $y(t) = \sum_{n=0}^{N} a_n x^n(t)$, where N is our highest model order, and the Taylor series coefficients a_n are a function of input variance σ_{id}^2. In a practical modeling situation, the engineer is tasked with deciding what is "good enough" for a given model. This Volterra series modeling process is equally valid if the system has memory, but we limited our focus to a memoryless nonlinearity. As mentioned earlier, including higher-order kernels may improve RRMSE at the expense of computation cost, especially when the kernels include memory.

2.7 Conclusion

Developing models of nonlinear audio systems can be challenging, especially if the degree of nonlinearity is unknown. We offer one approach to making educated guesses about the degree of nonlinearity of a system by comparing various measured Volterra series models

against the reference system. This was done by comparing 2nd- through 5th-order Volterra series models against a linear model and against each next-lowest order model to judge incremental improvements. For the memoryless, nonlinear reference system arctan(x), we saw that between orders 1-5, a 3rd-order model seemed to offer the best approximation in a relative root-mean-square error (RRMSE) sense, and that even-order nonlinear terms offered relatively low improvement. We attribute these observations to the fact that an ideal Taylor series approximation of the arctan function uses only odd-order terms.

Future extensions of the work include considering nonlinear systems with memory; that is, systems with finite impulse responses with more than one coefficient. As model order increases, the presence of memory causes computation time to increase exponentially. To address this, we might draw inspiration from [9] by approximating finite impulse responses with iteratively-determined second-order IIR filters. Another further development would be combining our model parameter estimation techniques here with the multivariance approach described in [19]. Although the comparative advantage of a multivariance model over a single-variance model has been shown, we theorize that the ideal choice of σ_{id}^2 for each Volterra kernel might be found experimentally to give an even better multivariance Volterra series model. In addition, though our error metric was quantitative, we acknowledge the importance of subjective listening tests, as a better RRMSE score does not necessarily equate to better human perception [26]. We could also try benchmarking the model and original system by looking at the error of both system's outputs given a musical input signal $x[n]$ instead of Gaussian white noise. Finally, we might consider additional statistical properties of the various ΔRRMSE measurements beyond just sample mean, such as variance, for quantifying "how much better" a given Volterra series model is than another.

CHAPTER THREE

ESTIMATING NONLINEAR IMPULSE RESPONSE LENGTHS WITH TIME-DELAYED MUTUAL INFORMATION

Note: material in this chapter is based on the author's submission for the 2020 Audio Engineering Society's 149th Convention, Paper 10416 [12], accepted based on a peer-reviewed abstract and 750-word precis.

3.1 Abstract

Using impulse responses measured from various audio systems has become common in audio signal processing. However, determining the significant values of a recovered impulse response from a nonlinear system can be problematic, as these systems violate the linearity principle from signals and systems theory. A "lumpy" or "spiky" tail in the measured impulse response a tell-tale symptom of nonlinearity. In this chapter, we investigate the use of time-delayed mutual information (TDMI), a concept from the field of information theory, to identify the useful portion of a recovered nonlinear impulse response. Initial test systems include linear, time-invariant FIR and IIR filters, and IIR filters with a static nonlinear distortion added. We find that TDMI accurately predicts the number of FIR filter taps and their time-locations when applied to a delay, FIR moving-average filter, and Schroeder all-pass filters. For IIR lowpass filters based on analog RC filter prototype using the impulse invariance method, the TDMI method reasonably predicted the significant impulse response values for cutoff frequencies $f_c = 10$ and 20 kHz, but performance degraded for 5, 2, and 1 kHz cutoff frequencies. After adding a static nonlinearity to these same IIR filters, the TDMI method tended to underreport the total number of significant impulse response values.

3.2 Introduction

The problem of noise contaminating the recovered impulse response of a system with nonlinearities is well-known [8, 24, 31]. Often, this noise will manifest as a "lumpy" tail in the impulse response. This phenomenon can complicate the issue of determining the actual length of the impulse response. When using the measured response for filtering audio signals, excluding noisy samples improves distortion immunity [8]. Applying this same line of reasoning to nonlinear Volterra series models investigated in the previous chapter, truncating the measured Volterra kernels of tube amplifiers showed improved model performance and reduced computational complexity [20].

This chapter investigates estimating the impulse response length of nonlinear systems using time-delayed mutual information (TDMI), a technique originating from the information theory field [7]. TDMI is not a modeling technique, rather, it is a method to quantify the nonlinear correlation between two signals across various time-shifts [5]. In our case, the two signals are the input and output of a nonlinear system, and the TDMI should indicate the memory extent or significant values of the system's recovered impulse response. Thus, we hope to demonstrate that TDMI can help distinguish a nonlinear system's recovered impulse response from noise due to nonlinearities.

To evaluate how well this technique estimates a system's memory length, we first apply it to linear, time-invariant FIR and IIR filters to develop confidence in applying the same method to nonlinear systems. We evaluate the TDMI performance by comparing the results to the expected impulse response length. By establishing a strong foundation on known systems, we can then address known nonlinear systems with confidence, and eventually, unknown systems that may exhibit nonlinear behavior.

3.3 Background: Time-Delayed Mutual Information

In this section, we explain the theory behind time-delayed mutual information (TDMI), starting from the basic concept of entropy. In information theory, *Shannon entropy*–or simply *entropy*–measures the amount of information in a signal as a function of the predictability of observing a given sample [21]. *Kullback entropy* quantifies the excess entropy when assuming a signal has one underlying probability distribution, when in actuality it may have a different distribution [7, 27]. *Mutual information* is a specific form of Kullback entropy that measures the deviation from statistical independence between two random variables or signals, say X and Y [7]. In other words, mutual information tells us how knowledge of X informs our knowledge of Y, and vice versa. Finally, *time-delayed mutual information* (TDMI) introduces some sensitivity to direction by applying a time-delay to one of the signals [27, 32]. This chapter focuses on estimating the impulse response length of nonlinear systems by evaluating the TDMI over a range of time-delays τ applied to the output signal $y[n]$.

3.3.1 Entropy

In the field of information theory, *entropy* measures the amount of information in a message, proportional to the uncertainty of the message [21]. For example, one message, or *symbol*, chosen out of a list of 10 equally-probable and unique possible symbols has lower uncertainty or "randomness" than a symbol chosen out of 10^6 such symbols, and thus has lower entropy. Because the probability of any given symbols is equal for all symbols, such random processes have *uniform* probability distributions. In Equation 3.1, entropy H is represented as the number of binary bits needed to encode each equally-probable symbol, or all n possible values, of some random process.

$$H = \log_2(n) \text{ bits/symbol} \tag{3.1}$$

There is a base 2 logarithmic relationship between entropy and the number of possible values a random process may take. Entropy depends on the probabilities of each possible symbol in a random process, not the actual value of the symbols [7]. Note that despite using bits as a unit of measurement, entropy can take on non-integer values. Consider the entropy for two processes P and Q, which have $n_p = 4$ and $n_q = 5$ equally-likely symbols, respectively:

$$\begin{aligned} H_p &= \log_2(n_p) \\ &= \log_2(4) \\ &= 2 \text{ bits/symbol} \end{aligned}$$

$$\begin{aligned} H_q &= \log_2(n_q) \\ &= \log_2(5) \\ &= 2.321 \text{ bits/symbol} \end{aligned} \tag{3.2}$$

Generally, when representing unsigned integers between 0 and 7 with the binary number system, we use 3 bits. Yet, H_q indicates that we don't need 3 bits to represent the 5 possible outcomes of process Q, we need only 2.321. While fractions of bits can seem confusing for those familiar with the binary number system, entropy simply tells us the *minimum number of bits* needed to encode all outcomes of a random process.

Measuring entropy for equally-probable messages is a useful learning tool, but real-world random processes often have nonuniform probability distributions. A more general measure of entropy is *Shannon entropy*, based on the probability distribution $p(i)$ of a random variable, say, I. Shannon's measure of entropy is given as the summation of each symbol's probability

$p(i)$ multiplied by the base 2 logarithm of the probability's reciprocal, $\frac{1}{p(i)}$:

$$\begin{aligned} H_I &= \sum_i p(i) \log_2(1/p(i)) \\ &= \sum_i p(i)\{\log_2(1) - \log_2 p(i)\} \\ &= \sum_i p(i)\{0 - \log_2 p(i)\} \\ &= -\sum_i p(i) \log_2 p(i) \text{ bits/symbol} \end{aligned} \tag{3.3}$$

Revisiting the random processes P and Q from before, the probability of observing one of 4 values from P is $p_p(i) = \frac{1}{4}$, and the probability of observing one of 5 values in Q is $p_q(i) = \frac{1}{5}$. Using Shannon's measure of entropy we can again find the entropy for these processes. Notice that the resulting value for H_P in Equation 3.4 matches the value found earlier in Equation 3.2:

$$\begin{aligned} H_P &= -\sum_i p_p(i) \log_2 p_p(i) \\ &= -\{p_p(0) \log_2 p_p(0) + p_p(1) \log_2 p_p(1) + \cdots + p_p(3) \log_2 p_p(3)\} \\ &= -\{\frac{1}{4} \log_2 \frac{1}{4} + \frac{1}{4} \log_2 \frac{1}{4} + \frac{1}{4} \log_2 \frac{1}{4} + \frac{1}{4} \log_2 \frac{1}{4}\} \\ &= -\frac{1}{4}\{-8\} \\ &= 2 \text{ bits/symbol} \end{aligned} \tag{3.4}$$

Due to its generality, Shannon's entropy formula enables entropy calculations for random processes with nonuniform probability distributions. A fair or unbiased coin toss has a $\frac{1}{2}$ probability of being heads or tails and thus has an entropy of 1 bit/symbol. Now we

find the entropy for a *biased* coin toss C that is heads $\frac{2}{3}$ of the time and tails $\frac{1}{3}$ of the time:

$$\begin{aligned} H_C &= -\sum_i p_c(i) \log_2 p_c(i) \\ &= -\{p_c(\text{heads}) \log_2 p_c(\text{heads}) + p_c(\text{tails}) \log_2 p_c(\text{tails})\} \\ &= -\{\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3}\} \\ &= -\{-0.390 + -0.528\} \\ &= 0.918 \text{ bits/symbol} \end{aligned} \tag{3.5}$$

Because the biased coin will be heads more often than tails, it has less entropy than a fair coin, in which both outcomes are equally probable. Thus, while an unbiased coin would need 1 bit to encode its outcomes, a biased coin needs less than 1 bit because we can better predict its outcomes.

3.3.2 Kullback Entropy

Kullback entropy is a measure of the excess entropy of a random process with probability distribution $p(i)$ while assuming it has a different probability distribution, $q(i)$ [7, 27]. More intuitively, the excess entropy quantifies the coding inefficiency if we use $q(i)$ instead of $p(i)$. Kullback entropy is also known as *Kullback-Leibler distance*, *Kullback-Leibler divergence* or *relative entropy*. The expression for Kullback entropy is:

$$\begin{aligned} K_I = D(p||q) &= \sum_i p(i) \log_2 \frac{p(i)}{q(i)} \\ &= \sum_i p(i) \{\log_2 p(i) - \log_2 q(i)\} \end{aligned} \tag{3.6}$$

Kullback entropy is always nonnegative, only 0 if $p = q$, and infinite if there are any $q(i) = 0$ for $p(i) \neq 0$. The number of bits required to encode process P assuming the distribution q is $H_p + D(p||q)$ [7].

Revisiting the biased coin example from (3.5), in (3.7) we compute an excess entropy or coding inefficiency of 0.0850 for an unbiased coin if we assume it is a biased coin. Here, $p(i)$ and $q(i)$ refer to the probabilities of the unbiased coin and biased coin, respectively:

$$\begin{aligned} D(p||q) &= \sum_i p(i) \log_2 \frac{p(i)}{q(i)} \\ &= p(\text{heads}) \log_2 \frac{p(\text{heads})}{q(\text{heads})} + p(\text{tails}) \log_2 \frac{p(\text{tails})}{q(\text{tails})} \\ &= 0.5 \log_2 \frac{0.5}{\frac{2}{3}} + 0.5 \log_2 \frac{0.5}{\frac{1}{3}} \\ &= -0.2075 + 0.2925 \\ &= 0.0850 \end{aligned} \tag{3.7}$$

In addition, Kullback entropy may be used with a conditional probability $p(i|j)$, the probability of I given J. More specifically, this allows measuring the coding inefficiency of using $q(i|j)$ instead of $p(i|j)$. The following expression considers only one possible state or outcome of the process J:

$$K_j = \sum_i p(i|j) \log_2 \frac{p(i|j)}{q(i|j)} \tag{3.8}$$

When taking the summation over all states of the process J, the single $p(i|j)$ term becomes $p(i, j)$:

$$K_{I|J} = \sum_{i,j} p(i, j) \log_2 \frac{p(i|j)}{q(i|j)} \tag{3.9}$$

3.3.3 Mutual Information

Mutual information is a specific application of Kullback entropy that measures the reduced uncertainty about I given information from J [7]. In Equation 3.10 below, we assume I and J are statistically independent processes, when they may have a joint probability

$p(i,j)$. Mutual information does not indicate whether I influences J or vice versa, just that some dependency exists. The expression for mutual information M_{IJ} is

$$M_{IJ} = \sum_{i,j} p(i,j) \log_2 \frac{p(i,j)}{p(i)p(j)}. \tag{3.10}$$

As with any Kullback entropy measure, mutual information is always greater than or equal to 0. If mutual information is 0, then $p(i,j) = p(i)p(j)$, which means I and J are statistically independent. Theorem 2.4.1 from [7] gives useful algebraic relationships between various mutual information and entropy quantities as shown below:

$$\begin{aligned} M_{IJ} &= H_I - H_{I|J} \\ M_{IJ} &= H_J - H_{J|I} \\ M_{IJ} &= H_I + H_J - H_{I,J} \end{aligned} \tag{3.11}$$

To give a mutual information example, we will use the third relationship from (3.11) along with Table 3.1, the joint probabilities $p(i,j)$ for statistically independent I and J. Calculating mutual information with the third expression in (3.11) above, H_I and H_J are the entropies of the marginal probabilities for I and J in Table 3.1, respectively. We find the marginal probabilities P_I and P_J by summing the bold numbers from Table 3.1:

$$\begin{aligned} P_I &= 0.5 + 0.5 = 1 \\ P_J &= 0.5 + 0.5 = 1 \end{aligned} \tag{3.12}$$

Having calculated the marginal probabilities P_I and P_J, we then use the Shannon entropy from Equation 3.3 to find the marginal probabilities H_I and H_J–both are 1 bit/symbol. $H_{I,J}$, the joint entropy of $p(i,j)$, is 2 bits/symbol. Mutual information M_{IJ} is

	$p(I=0)$	$p(I=1)$	Marginal $p(J)\downarrow$
$p(J=0)$	0.25	0.25	**0.5**
$p(J=1)$	0.25	0.25	**0.5**
Marginal $p(I)\rightarrow$	**0.5**	**0.5**	

Table 3.1: Joint probabilities $p(i,j)$ for statistically independent I, J

zero, thus indicating I and J are statistically independent:

$$\begin{aligned} M_{IJ} &= H_I + H_J - H_{I,J} \\ M_{IJ} &= 1 + 1 - 2 \\ M_{IJ} &= 0 \text{ bits/symbol} \end{aligned} \tag{3.13}$$

Thus, since mutual information is 0, $p(i,j) = p(i)p(j)$, which means I and J are statistically independent. Now consider Table 3.2, containing joint probabilities $p(p,q)$ for two dependent, random variables P and Q. The marginal entropies H_P and H_Q respectively are 1.4855 and 0.9341 bits/symbol, and the joint entropy $H_{P,Q}$ is 2.2842 bits/symbol. Again using (3.11), the Mutual Information M_{PQ} for P and Q is

$$\begin{aligned} M_{PQ} &= H_P + H_Q - H_{P,Q} \\ M_{PQ} &= 1.4855 + 0.9341 - 2.2842 \\ M_{IJ} &= 0.1354 \text{ bits/symbol} \end{aligned} \tag{3.14}$$

The positive M_{IJ} in (3.14) reflects some joint probability between P and Q. This suggests that knowledge about P or Q would reduce the uncertainty of the other by 0.1354 bits.

	$p(P=0)$	$p(P=1)$	$p(P=2)$	Marginal $p(Q)\downarrow$
$p(Q=0)$	0.20	0.05	0.40	**0.65**
$p(Q=1)$	0.10	0.15	0.10	**0.35**
Marginal $p(P)\rightarrow$	**0.3**	**0.2**	**0.5**	

Table 3.2: Joint probabilities $p(p,q)$ for statistically dependent P, Q

3.3.4 Time-Delayed Mutual Information

Because mutual information does not indicate causality between two random processes, introducing a time-delay τ in one of the processes can introduce some sensitivity to direction [27, 32]. Known as *directional* or *time-delayed mutual information*, $M_{IJ}(\tau)$ is expressed as

$$M_{IJ}(\tau) = \sum p(i[n], j[n-\tau]) \log_2 \frac{p(i[n], j[n-\tau])}{p(i[n])p(j[n-\tau])}. \tag{3.15}$$

Until now, none of the probability mass functions discussed have had a time argument. Further, in all of the example random processes, the probability of each random outcome is known. However, in observing real-world random processes, the underlying probability mass function is rarely known and instead must be empirically estimated. The process of determining a probability density or mass function from observed data is known as *density estimation* [28]. With this discrepancy between theory and practice in mind, the time-delayed mutual information $M_{IJ}(\tau)$ also depends on the accuracy of the estimated probability density functions $p(\cdot)$. Further, whether a positive value for $M_{IJ}(\tau)$ indicates directionality requires knowledge of the *estimator bias*. Only $M_{IJ}(\tau)$ values greater than the upper-bound of the estimator bias indicate a relationship between the two random processes in question [5].

Kernel density estimation (KDE) was used in [5] for the estimating time-delayed mutual information in nonuniformly-sampled data sets. Despite potentially high computation costs, KDE is a popular choice for creating assumption-free estimates of an underlying probability

density function [10]. For our purposes, the histogram is an equally good estimator, as our data is uniformly sampled. Further, the performance of the histogram and KDE methods in the context of TDMI calculations converges around sample sizes of 1000 and higher [5]. For each TDMI calculation, we look at 10^6 data points of the input $x[n]$ and output $y[n]$.

3.4 Experiment Background

In the context of the following experiments, mutual information measures the reduction in uncertainty about the probability of a random process X given information from another process Y [7]. In an audio signal processing context, the processes X and Y are the sampled input and output signals $x[n]$ and $y[n]$ of a time-invariant audio system. Calculating mutual information requires estimating the marginal and joint probabilities of X and Y; that is, $\hat{P}(X)$, $\hat{P}(Y)$, and $\hat{P}(X,Y)$. In this work, we use the histogram as our estimator, as it performs satisfactorily for the length of signals we will be using [5]. Next, the Shannon entropy of each of these estimated probabilities is computed using

$$\hat{H}_X = -\sum_{i}^{N} \hat{p}(x_i) \log_2 \hat{p}(x_i) \text{ bits/symbol} \tag{3.16}$$

$$\hat{H}_Y = -\sum_{j}^{M} \hat{p}(y_j) \log_2 \hat{p}(y_j) \text{ bits/symbol} \tag{3.17}$$

$$\hat{H}_{X,Y} = -\sum_{i,j}^{N,M} \hat{p}(x_i, y_j) \log_2 \hat{p}(x_i, y_j) \text{ bits/symbol} \tag{3.18}$$

With the entropies of each probability distribution computed, the mutual information MI is given by the sum of the marginal entropies minus the joint entropy [7]:

$$MI = \hat{H}_X + \hat{H}_Y - \hat{H}_{X,Y} \tag{3.19}$$

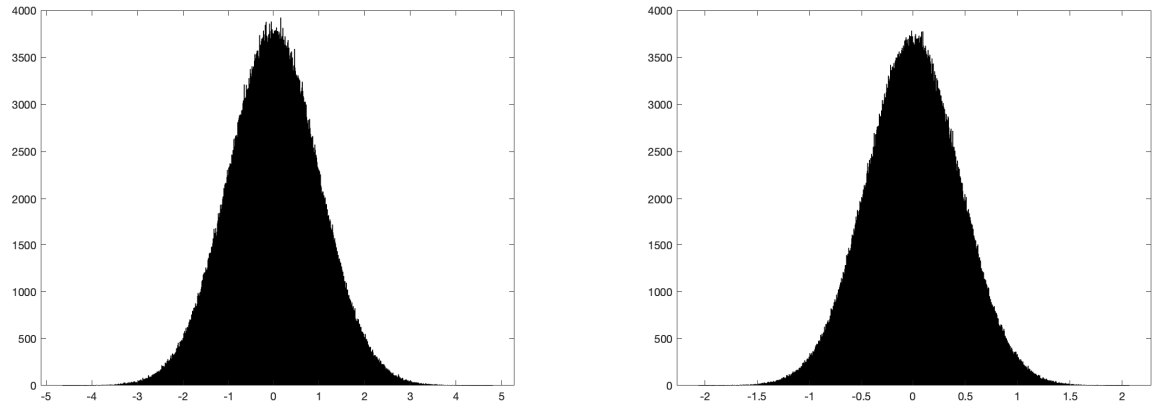
If the input and output signals are statistically independent – that is, $\hat{p}(x,y) = \hat{p}(x)\hat{p}(y)$ – MI in Equation 3.19 will be 0. Conversely, significant mutual information values are indicated where they are greater than the estimator bias or "noise floor" of the probability estimator; determining this estimator bias will be covered shortly. Now, introducing a time-shift τ in one of the random processes gives the time-delayed mutual information (TDMI):

$$MI(\tau) = \hat{H}_{X(\tau)} + \hat{H}_Y - \hat{H}_{X(\tau),Y} \tag{3.20}$$

By evaluating the TDMI in Equation 3.20 over a series of time-shifts τ, we estimate the impulse response length by counting how many samples it takes until the TDMI values fall below the estimator bias or "noise floor". To find the estimator bias, we first apply a random shuffle algorithm to the input signal $x[n]$, reordering the sequence elements randomly. Then, we measure the mutual information between the output signal $y[n]$ and the shuffled $x[n]$ delayed by τ [5]. Theoretically, there should be no mutual information between these signals; TDMI values less than or equal to this estimator bias shall be deemed insignificant.

3.5 Experiment

With the information theory fundamentals laid out, we now shift our attention to the experiments. We apply the time-delayed mutual information to a variety of time-invariant test systems in MATLAB: an FIR delay, FIR moving-average filter, Schroeder all-pass filters, IIR lowpass filters, and IIR lowpass filters with a static nonlinearity added. Based on the cross-correlation method used in [20] and [11], we use the MATLAB function `randn` to generate the test system's input signal $x[n]$, creating a pseudo-random Gaussian white noise sequence of length $N^2 = 10^6$ samples. We then capture the output signal $y[n]$ by passing $x[n]$ through the test system. Using $N = 10^3$ bins, the marginal probabilities $\hat{P}(X)$ and $\hat{P}(Y)$ and the joint probability $\hat{P}(X,Y)$ are respectively computed with the `histogram` and



(a) (b)

Figure 3.1: (a) Example 1D histogram used for $\hat{P}(X)$ estimation. (b) Example 1D histogram used for $\hat{P}(Y)$ estimation.

`histogram2` MATLAB functions, and we specify an array of sample-shifts τ over which to compute the TDMI. Example 1D histograms for $\hat{P}(X)$ and $\hat{P}(Y)$ are shown in Figure 3.1, and a 2D histogram example for the joint probability $\hat{P}(X, Y)$ is shown in Figure 3.2.

Once the signals $x[n]$, $y[n]$, the bin count N, and the array of sample-shifts τ are decided, they are passed into a custom MATLAB function, `tdmi`. This function iterates over all sample-shifts τ provided, computing each probability distribution and its associated Shannon entropy, as given in Equations 3.16-3.18. Then, the function computes the TDMI for each sample shift τ as shown in Equation 3.20. For each value τ, the function also determines the estimator bias as described in the previous section. The function then returns the sample-shifts, TDMI values, and estimator bias as separate arrays for analysis and plotting.

In the sections that follow, we start with relatively simple test systems for evaluating the TDMI method, including a delay line and FIR moving average filter. Then, we look at IIR lowpass filters, briefly discuss all-pass filters with various levels of attenuation, and finally, we examine the same IIR filters but with a nonlinear distortion added to the filter

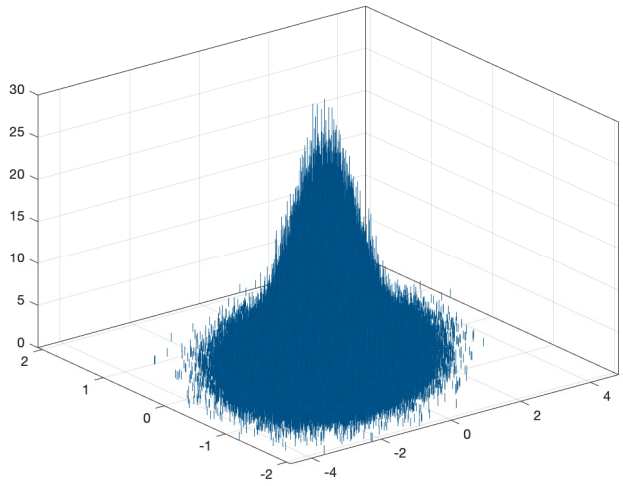


Figure 3.2: Example 2D histogram used to estimate joint probability $\hat{P}(X, Y)$.

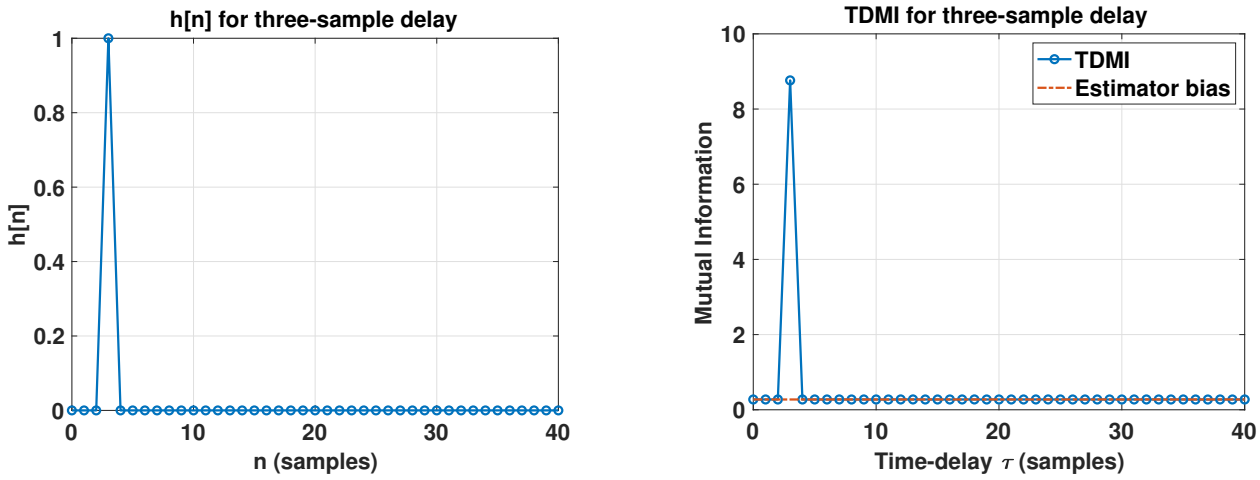
output.

3.5.1 First experiment: Simple delay system

As a straightforward first demonstration, our initial test system is simply a delay of three samples:

$$h[n] = \delta[n - 3] \tag{3.21}$$

Figure 3.3a shows the impulse response for the delay system described by Equation 3.21. After passing the input signal $x[n]$ through the system and recording the output $y[n]$, we then compute the TDMI and estimator bias over sample-shifts τ ranging from zero samples to 40 samples; see Figure 3.3b. Here we once again emphasize that the computed TDMI values do not serve as a *model* of the test system, but as an estimate of the significant impulse response values. Because $y[n]$ is simply a copy of $x[n]$ delayed by three samples, the maximum TDMI between $y[n]$ and x[n] occurs at $\tau = 3$. In contrast, the TDMI values at



(a) (b)

Figure 3.3: (a) Impulse response for delay system described in Eq. 3.21. (b) TDMI evaluated over $\tau = [0, 40]$ for same system.

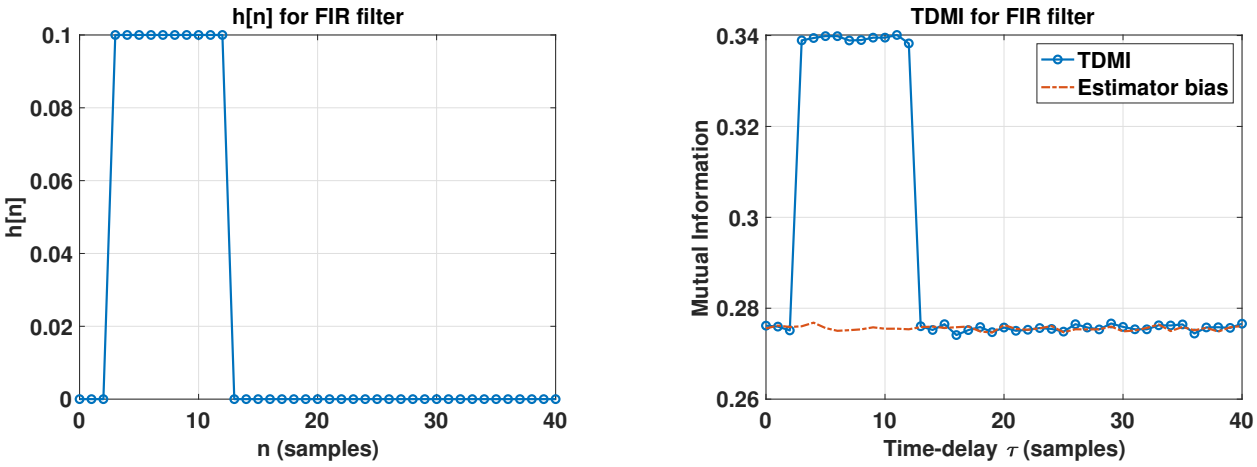
any other time-shift τ are indistinguishable from the estimator bias or TDMI "noise floor".

3.5.2 Second experiment: FIR Moving Average filter

To further evaluate TDMI as a method for estimating the significant impulse response values, we cascade the three-sample delay with a moving-average FIR filter containing 10 coefficients of $\frac{1}{10}$ each, resulting in the following impulse response:

$$h(n) = \begin{cases} \frac{1}{10}, & 3 \leq n \leq 12 \\ 0, & \text{otherwise} \end{cases} \tag{3.22}$$

The impulse response for this system is shown in Figure 3.4a; evaluating the TDMI across sample-shifts from $\tau = 0$ to $\tau = 40$ is shown in Figure 3.4b. In this example, the TDMI is clearly above the estimator bias for the active part of the filter, where $3 \leq \tau \leq 12$ sample-shifts. Thus, while we do not create a model of the test system, we can use the significant TDMI values to determine which values of the recovered impulse response to keep.



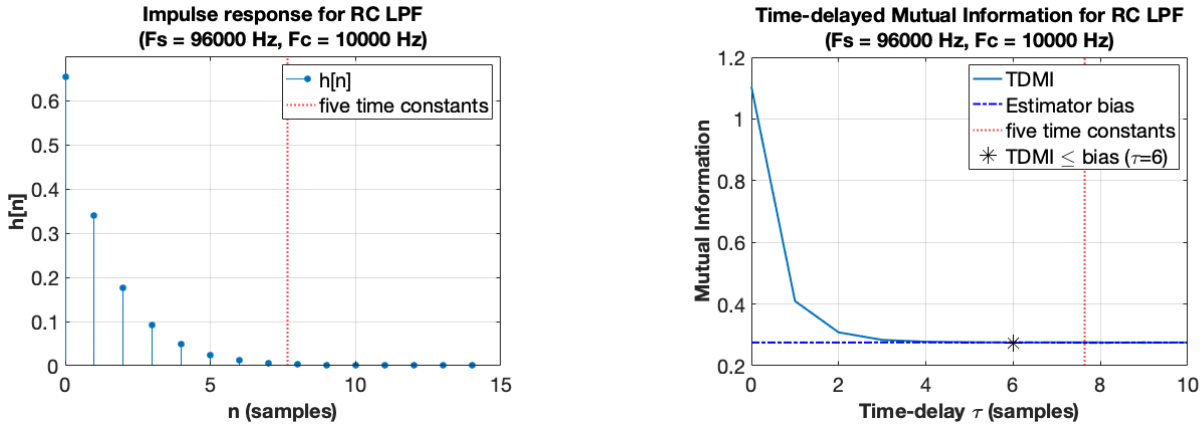
(a) (b)

Figure 3.4: (a) Impulse response of delayed, FIR moving-average system described in Eq. 3.22. (b) TDMI evaluated over $\tau = [0, 40]$ for same system.

3.5.3 Third experiment: IIR filters

With the examples of a delay line and a basic FIR filter validated, the next degree of complexity is to consider systems characterized by an infinite impulse response (IIR). Due to the analog nature of audio systems like amplifiers and filters, we must expect to encounter such IIR systems in the "real world". Hence, in this experiment, we investigate the performance of the TDMI approach on RC low-pass filters digitized using the impulse invariance method. Because RC low-pass filters are well understood and easy to design, we chose to focus on their digitized versions as our first IIR test systems. Despite the fact that the impulse response length of these filters is truly infinite, a common heuristic states that the impulse response decays after $5 \times RC$, or five time constants. Therefore, this rule of thumb will be useful when interpreting the calculated TDMI values.

To conduct the following IIR experiments, using the impulse invariance method, we designed RC lowpass filter prototypes with various cutoff frequencies of f_c =1, 2, 5, 10, and 20 kHz, using a sampling frequency of $f_s = 96$ kHz. The Laplace-domain transfer function



(a) (b)

Figure 3.5: (a) Impulse response for digitized RC Lowpass Filter with $f_c = 10$kHz. (b) TDMI evaluated over $\tau = [0, 10]$

for the RC filters is given by

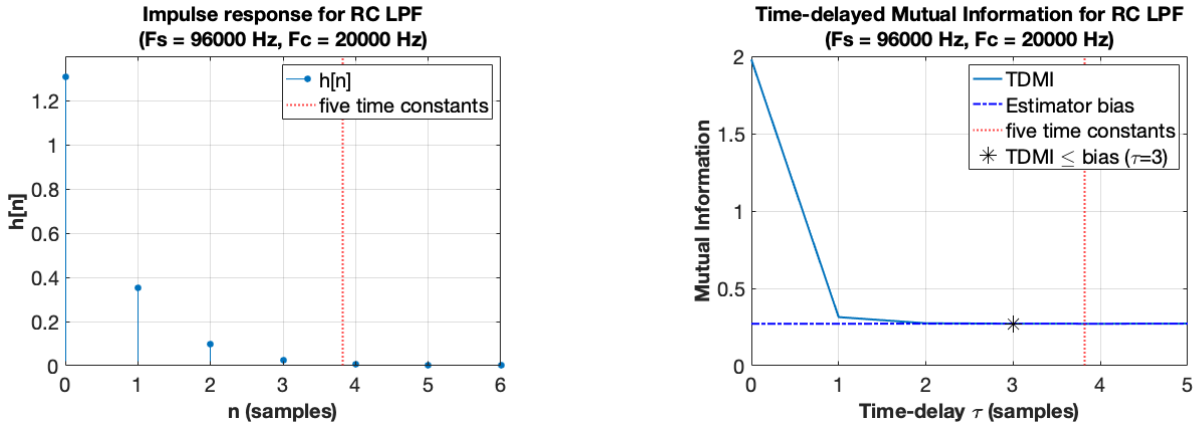
$$H(s) = \frac{2\pi f_c}{s + 2\pi f_c}, \tag{3.23}$$

where $2\pi f_c = \frac{1}{RC}$. We obtain the discrete-time RC filters using the impulse invariance relation from [17],

$$H(z) = \frac{2\pi f_c t_0}{1 - e^{-2\pi f_c t_0} z^{-1}}, \tag{3.24}$$

where $t_0 = 1/F_s$, for $F_s = 96$kHz. To evaluate the TDMI method for our lowpass filter prototypes, we apply the same method as before: pass pseudo-random Gaussian white noise through the system defined by $h[n]$, select sample-shifts τ over which to evaluate the TDMI, and pass $x[n]$, $y[n]$, and the lags into our MATLAB function `tdmi`.

3.5.3.1 *10kHz low-pass IIR filter* Starting with $f_c = 10$kHz, the impulse response of the designed low-pass filter is shown in Figure 3.5a. The abscissa is shown in samples based on F_s. Although five time constants in continuous-time does not correspond to an integer

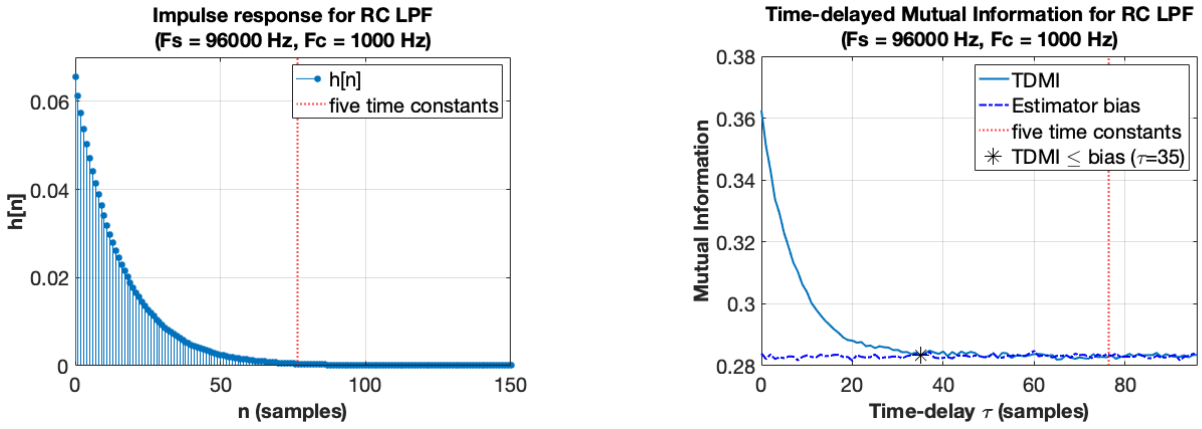


(a) (b)

Figure 3.6: (a) Impulse response for digitized RC Lowpass Filter with $f_c = 20$kHz. (b) TDMI evaluated for RC Lowpass Filter with $f_c = 20$kHz

number of samples, it is nonetheless plotted as a vertical line for reference. Figure 3.5b shows the TDMI calculation and estimator bias for the filter with $f_c = 10$kHz. Also in the TDMI calculation shown in Fig. 3.5b, we plot five time constants as a vertical dotted line for reference. In addition, we indicate where the TDMI is less than or equal to the estimator bias with a star. As seen in Fig. 3.5b, the significant portion of the TDMI occurs between $0 \leq n \leq 6$. With this particular filter, five time constants multiplied by F_s corresponds to 7.6394 samples. Hence, we have some confidence that the TDMI could estimate the length of the impulse response for $h[n]$ in this instance.

3.5.3.2 *20kHz low-pass IIR filter* Next, we look at the case where $f_c = 20$kHz. Its impulse response and TDMI plot are shown in Figures 3.6a and 3.6b, respectively. The results are similar to the $f_c = 10$kHz case: the TDMI is above the estimator bias between $0 \leq \tau \leq 3$ sample-shifts, slightly short of the five time constants mark of 3.8197 samples when scaled by $F_s = 96$kHz. Thus, while the TDMI method did not perfectly predict the impulse response length, it seemed to perform relatively well.



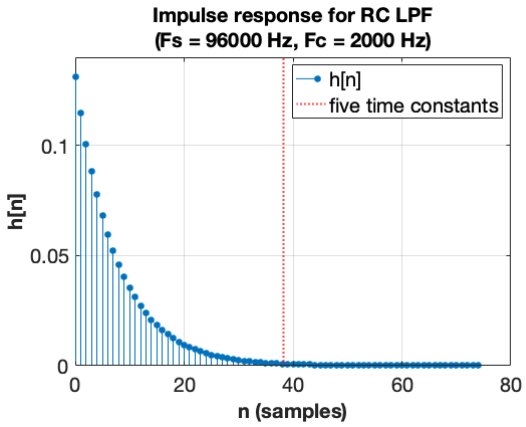
(a) (b)

Figure 3.7: (a) Impulse response for digitized RC Lowpass Filter with $f_c = 1$kHz. (b) TDMI evaluated for RC Lowpass Filter with $f_c = 1$kHz

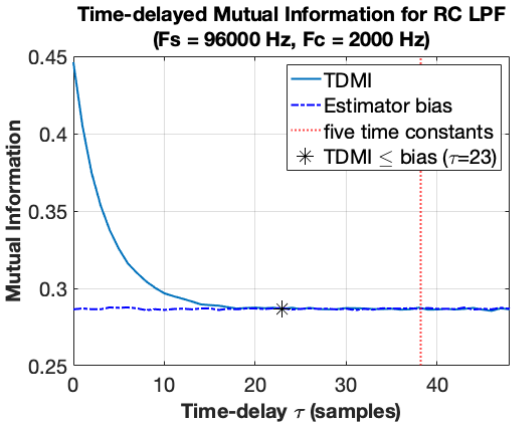
3.5.3.3 *1, 2, and 5kHz low-pass IIR filters* We noticed an interesting phenomenon when dealing with low-pass filters with narrower passbands. In this subsection, we examine the TDMI method as applied to filters with cutoff frequencies of 1, 2, and 5 kHz, respectively. The impulse response plots and TDMI calculations for these three filters are shown in Figures 3.7a through 3.9b.

We can see that as the passband shrinks by reducing f_c, the TDMI approach gets worse at predicting the impulse response length. In the 1kHz example, the TDMI falls below the estimator bias for $\tau \geq 35$ samples, yet the scaled five time constants occurs at 76.3944 samples. In the 2kHz example, the TDMI method indicates significant values between $0 \leq \tau \leq 34$, while the scaled five time constants should be 38.1972 samples. Thus, the TDMI already seems to improve compared to the 1kHz case. Finally, for $f_c = 5$kHz, TDMI is significant for $0 \leq \tau \leq 14$, and five time constants occurs at 15.2789 samples.

To explain the relationship between TDMI performance and passband width, it initially appears that the power of the filtered output signal $y[n]$ shrinks as we reduce the filter cutoff frequency, f_c. Intuitively, this is no surprise as the filter attenuates certain frequencies of the

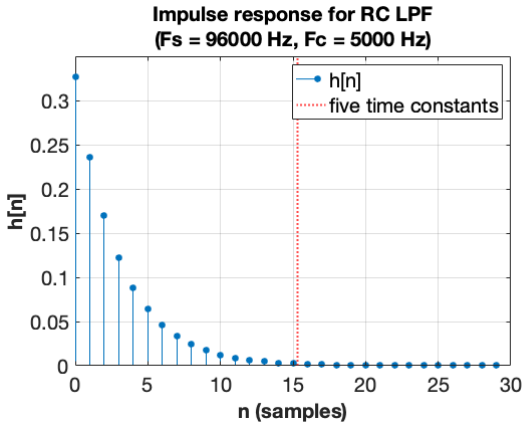


(a)

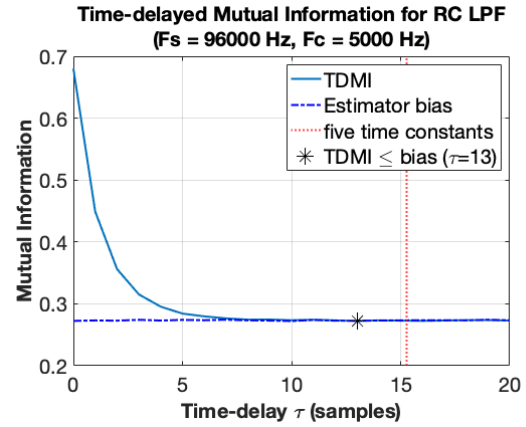


(b)

Figure 3.8: (a) Impulse response for digitized RC Lowpass Filter with $f_c = 2$kHz. (b) TDMI evaluated for RC Lowpass Filter with $f_c = 2$kHz



(a)



(b)

Figure 3.9: (a) Impulse response for digitized RC Lowpass Filter with $f_c = 5$kHz. (b) TDMI evaluated for RC Lowpass Filter with $f_c = 5$kHz

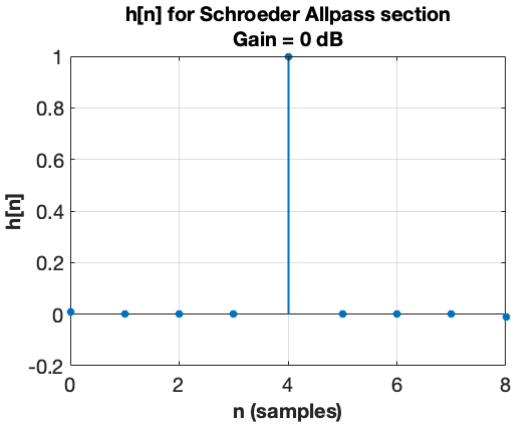
broadband noise in the test signal $x[n]$. To check whether the decreased power of the output signal was causing this phenomenon, in the next section we present tests using all-pass filters with gradually decreasing gains.

3.5.4 All-pass filters with decreasing gains

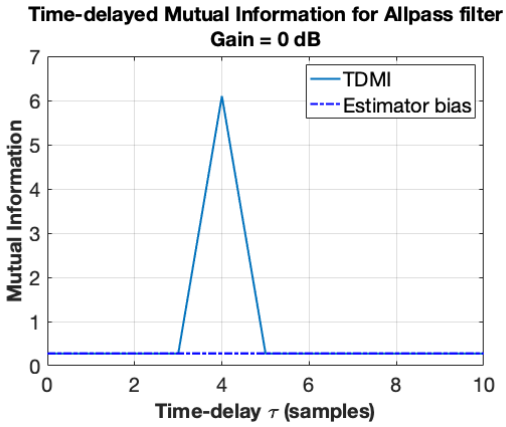
As mentioned in the previous subsection, the TDMI method when applied to our designed lowpass filters becomes unreliable as the pass band becomes narrower. Here, we apply the TDMI method to Schroeder all-pass filters to verify whether the reduction in the level of output signal $y[n]$ is attributable to the unreliable TDMI values. From [29] we use the following discrete-time transfer function to create Schroeder all-pass filter sections:

$$H(z) = A\frac{b_0 + z^{-M}}{1 + a_M z^{-M}}, \tag{3.25}$$

We arbitrarily set the delay line length to $M = 4$ samples and set the feedback and feedforward coefficients $b_0 = a_M = 0.01$. We scale $H(z)$ in Eq. 3.25 by the constants $A = 1, 2^{-10}$, and 2^{-20} to apply varying amounts of attenuation to the system. These scalars correspond to 0, -60, and -120 dB of gain, respectively. The resulting impulse response and TDMI results for an all-pass filter with unity gain are given in Figures 3.10a and 3.10b, respectively. The TDMI calculation for an all-pass filter with -120 dB gain is shown for comparison in Figure 3.11b. We did not show figures for the -60 dB case because, regardless of the attenuation we applied to the all-pass filter, the TDMI calculation always indicated that the significant portion of the impulse response occured at $\tau = 4$ samples, which corresponds to the impulse response peak as shown in Figure 3.10a. Thus, we cannot say that merely reducing the amplitude of $y[n]$ will produce unreliable TDMI estimations. The scope of our contribution does not further address the issue of TDMI performing less reliably when applied to lowpass filters with narrow passbands, and remains an item for future work.

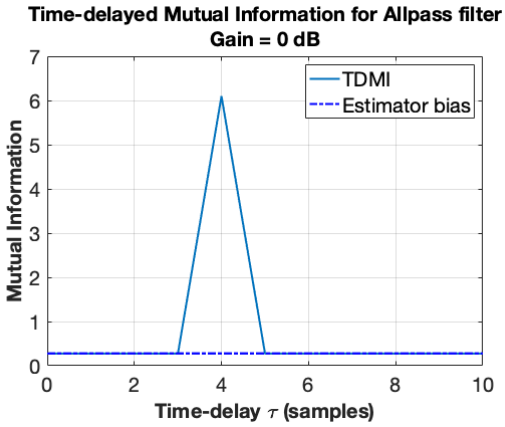


(a)

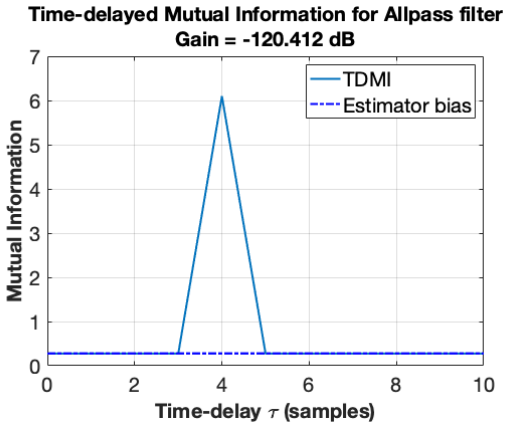


(b)

Figure 3.10: (a) Impulse response for all-pass filter with 0dB of attenuation. (b) TDMI evaluated for all-pass filter with 120dB of attenuation.



(a)



(b)

Figure 3.11: (a) TDMI evaluated for all-pass filter with 0dB of attenuation. (b) TDMI evaluated for all-pass filter with 120dB of attenuation

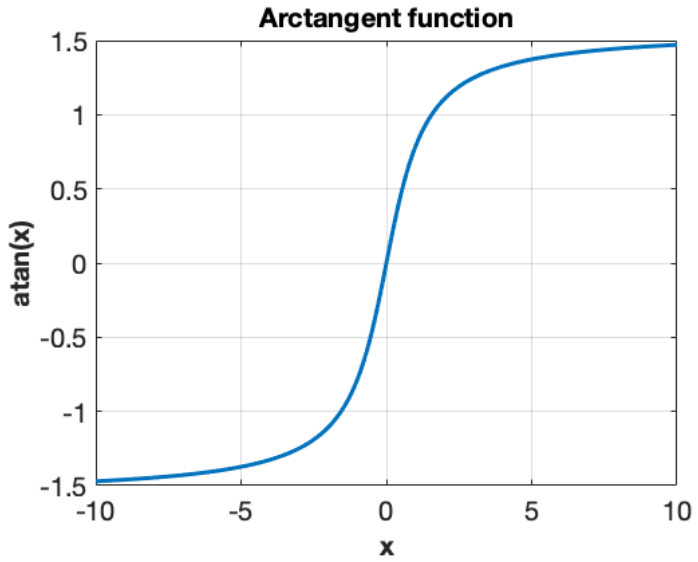
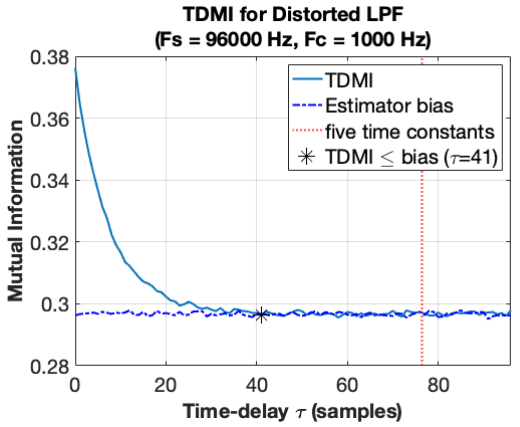


Figure 3.12: Arctangent function evaluated over $-10 \leq x \leq 10$

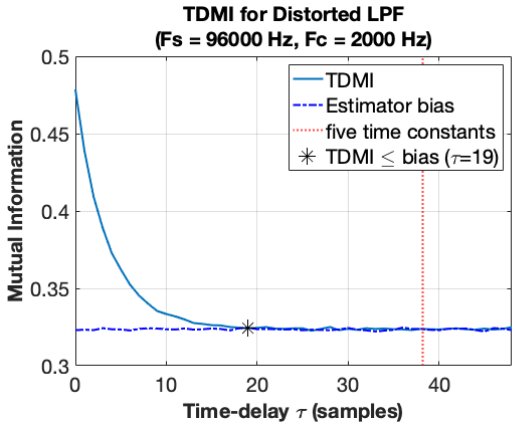
3.5.5 IIR Lowpass filters with static nonlinearity

Finally, we look at the TDMI method when applying a static nonlinearity to the output of the lowpass filters designed in Section 3.5.3. As in our prior work [11], we use the arctangent function to serve as a "soft-clipping" static nonlinearity; its function is shown in Figure 3.12. TDMI plots for the 1, 2, 5, 10, and 20 kHz lowpass filters cascaded with the arctangent distortion are shown in Figures 3.13a through 3.13e.

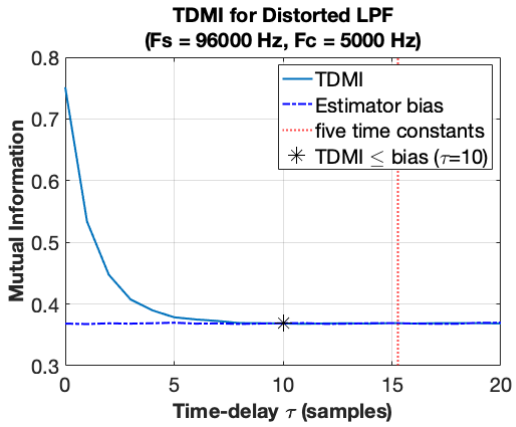
Comparing the distorted filters with their linear counterparts, the TDMI calculations included more of the impulse response length for the 1kHz case, and underestimated the impulse response length for the 2kHz, 5kHz, and 10kHz cases. The nonlinear 20kHz case appeared to perform similarly to the linear case, as it overestimated the impulse response length by about 1 sample. However, the linear 20kHz case underestimated the length by 1 sample. We did not observe the same trend in the linear experiments, where TDMI was less reliable as f_c decreased. To summarize, whereas the TDMI method performed better as the passband width increased in the linear IIR experiments, the TDMI method did not perform



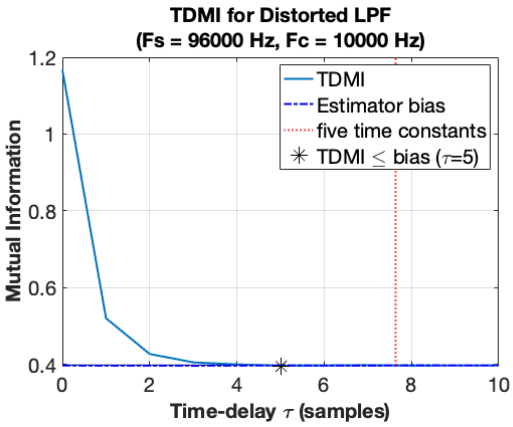
(a)



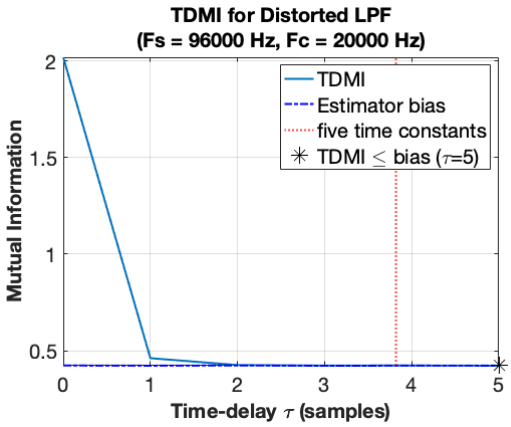
(b)



(c)



(d)



(e)

Figure 3.13: TDMI for distorted IIR lowpass filters, $f_c = 1, 2, 5, 10, 20$ kHz

predictably when applied to the nonlinear IIR filters.

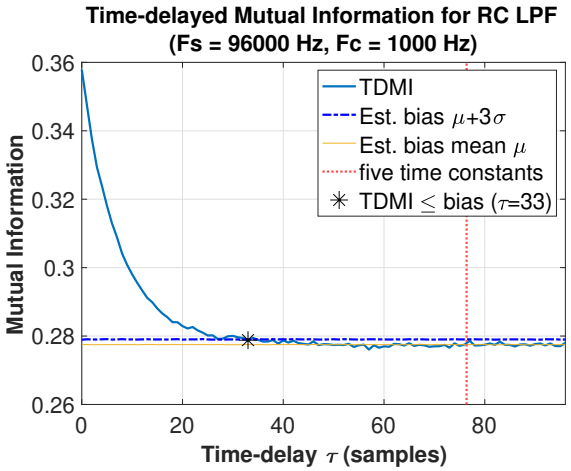
3.5.6 Recomputing TDMI Estimator Bias for Linear IIR Filters

In this section we demonstrate the same TDMI simulations with the linear IIR lowpass filters, except instead of computing the estimator bias once for every τ we compute it 1000 times to create an estimated probability distribution. For each of these 1000 calculations, we randomly shuffle $x[n]$ again before computing the estimator bias. We then look at where the calculated TDMI curve falls within three standard deviations of the estimator bias' estimated distribution to determine which values are significant. That is, we denote significant TDMI values when they exceed $\mu + 3\sigma$, where μ is the sample mean of the estimator bias and σ is the standard deviation. The subsequent TDMI plots are shown in Figure 3.14.

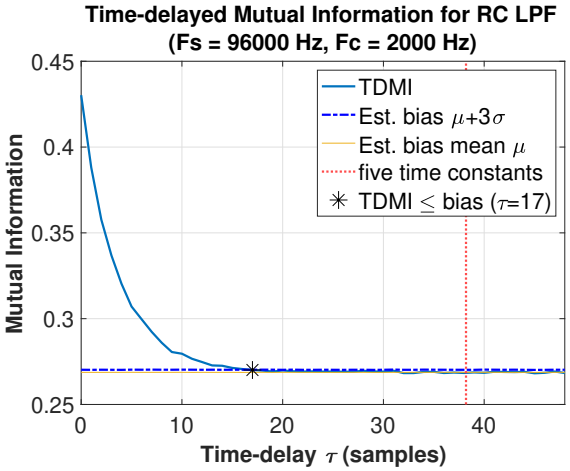
We find that by creating a probability distribution for the estimator bias at each τ, we have more confidence about where the TDMI falls below the noise floor. Compared to the previous TDMI computation with the linear IIR filters, the results are slightly more conservative. The estimated impulse response length is shortened by 2 samples for the 1kHz case, by 6 samples for the 2kHz case, 5 samples for the 5kHz case, and 1 sample for the 10kHz case. There was no distinction when comparing the TDMI estimation for the 20kHz case.

3.6 Discussion

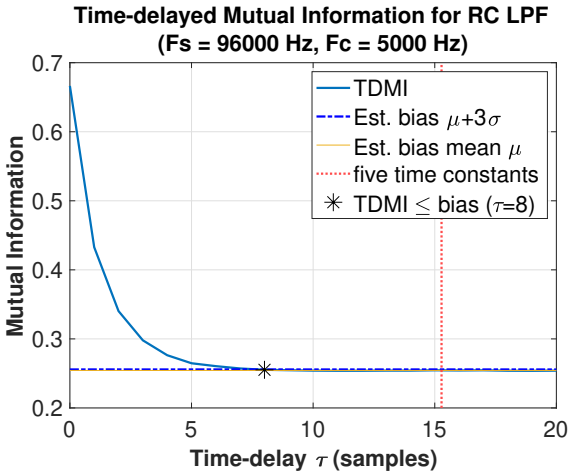
For fairly simple scenarios such as a delay line (Eq. 3.21), FIR moving-average filter (Eq. 3.22), and various Schroeder all-pass filters (Eq. 3.25), the TDMI method accurately reported the significant portion of the system's impulse response. Once we shifted our focus to IIR lowpass filters (Eq. 3.24), we noticed that the performance of the TDMI method worked for fairly generous passbands, but started to degrade as f_c was decreased. When adding the arctangent as a nonlinear distortion to these linear filters, the TDMI did worse for larger f_c



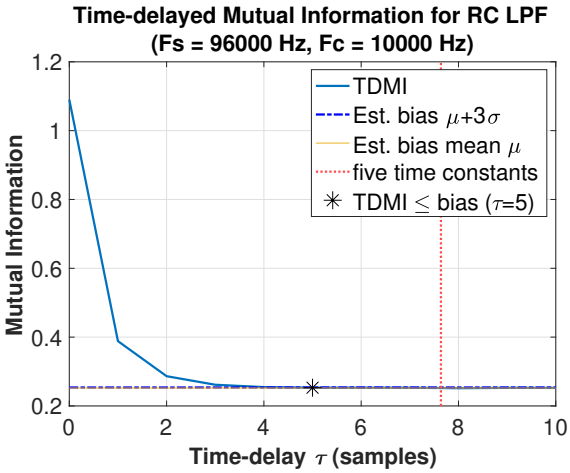
(a)



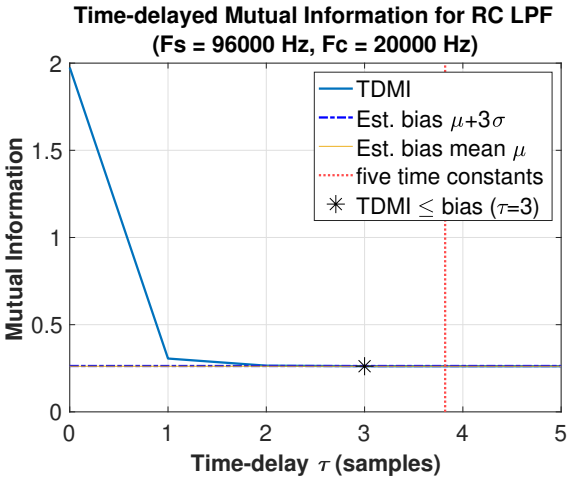
(b)



(c)



(d)



(e)

Figure 3.14: TDMI for IIR lowpass filters with 1000 estimator bias calculations per τ

but actually did better for $f_c = 1$kHz. There did not seem to be much difference between the linear and nonlinear 20kHz lowpass filter experiment. In the former, the impulse response length was underestimated by 1 sample, while in the latter, the length was overestimated by the same amount.

Though we do not address this further in the dissertation, the main outstanding issue from this chapter is the degradation of the TDMI method as f_c decreases in the linear case. We initially thought that the TDMI method might suffer if the amplitude of the output becomes extremely small, but the method worked perfectly on an all-pass filter with 120dB of attenuation. The TDMI calculations are dependent on the estimated probability distributions for the input and output signals, thus, the next logical step may be to investigate these distributions more closely as f_c varies. When adding the nonlinear distortion, the TDMI approach generally seemed to underestimate the impulse response length compared to the undistorted experiments. Again, investigating the calculated probability distributions in these cases may be worthwhile.

3.7 Summary

Measuring the impulse response of systems exhibiting nonlinear behavior by using linear systems theory is challenging. In this work, we present time-delayed mutual information (TDMI) as a potential technique for estimating the significant values of a measured impulse response for basic linear and nonlinear systems. From the field of information theoretics, mutual information is a nonlinear correlation metric of how much information from an output signal $y[n]$ may be caused by an input signal $x[n]$. By applying a time-delay to the input signal $x[n]$, e.g., $x[n-\tau]$, the TDMI calculation measures the shared information between the delayed $x[n]$ and $y[n]$. Thus, by repeating this process over various τ, we attempt to measure the impulse response length of the underlying system $h[n]$. For sample-shifts τ where the TDMI falls below the estimator bias, we may assume that the system's impulse

response $h[n]$ has fallen below the noise floor.

We validated the TDMI method by starting with simple systems such as a delay line, FIR moving-average filter, and Schroeder all-pass filters. We then investigated IIR lowpass filters to simulate encountering analog audio systems in the field. The TDMI method worked reasonably well for higher cutoff frequencies such as $f_c = 10$ and 20 kHz, but performance started to degrade as f_c was decreased down to 1kHz. We then tried the TDMI approach after applying a static nonlinearity to the IIR lowpass filters. In most instances, the TDMI approach tended to underreport the impulse response length compared to the linear cases.

We also looked at finding a probability distribution for the estimator biases in the linear IIR lowpass filter case. Compared to the other TDMI examples, we gain more certainty about where the TDMI estimation falls below the noise floor by looking at when it crosses within 3 standard deviations of the estimator bias sample mean. Aside from the 20kHz cutoff frequency filter, this updated approach tended to be more conservative when estimating impulse response length.

Future extensions of this work might consider investigating the effect of both nonlinear distortion and decreasing f_c on the TDMI performance, though this is beyond the scope of the dissertation. Initially we thought that the TDMI suffered due to the decreased energy in $y[n]$ as f_c was lowered. Yet, the TDMI method worked exceptionally for an all-pass filter with -120dB gain. We suspect that taking a closer look at the estimated probabilities of $x[n]$ and $y[n]$ may point to an explanation, as the TDMI calculation is inherently based on these probability estimations. Finally, a better error metric for estimating the significant portion of a measured impulse response may be to measure the energy of the significant response versus the under-estimated portion of the impulse response, say, by numerical integration. As the IIR lowpass filters have an exponentially decaying impulse response, most of the energy is present in the first few RC time constants. Therefore, weighting each sample of the impulse response within $5 \times RC$ may be unnecessarily penalizing.

CHAPTER FOUR

CASE STUDY - PROCO RAT 2 DISTORTION PEDAL

4.1 Abstract

In this chapter we look at how time-delayed mutual information (TDMI) changes as we separately vary the levels of nonlinear distortion and lowpass filtering in the ProCo Rat 2, a commercially available guitar distortion effects pedal. Operating under an assumption of time-invariance, for each parameter setting we stimulate the Rat pedal with Gaussian white noise that is scaled for a good signal-to-noise ratio without unintentionally overloading the device. We observed that except in the case of selective lowpass filtering, the general shape of the TDMI between input $x[n]$ and output $y[n]$ is a pronounced peak between $\tau = 0$ and 2 samples, depending on the parameter level. We attribute this observed time-shift to the increased phase shift of the pedal at higher levels of the Distortion or Filter settings. For a low cutoff frequency of 620 Hz, the TDMI curve is significantly reduced in amplitude and exhibits a gradual decay of many samples τ from the maximum value down to the estimator bias, which we attribute to a longer response time at this setting from the increased RC time constant. In conclusion, if one were to use this TDMI method on a system with unknown attributes, we predict the TDMI will exhibit relative insensitivity to distortion and mild filtering aside from time-shifting of the overall curve. In the presence of more pronounced lowpass filtering, we expect the TDMI curve to have a more subdued maximum and a more gradual decay from this maximum to the noise floor.

4.2 Introduction

Creating digital models of analog systems has plentiful applications in the realm of audio signal processing. Depending on how much is known about the inner workings of a

given analog audio system, the modeling process can vary from analytic solutions on one end to black-box modeling approaches on the other. In this work, we investigate the use of time-delayed mutual information (TDMI) and what information it can tell us about an analog audio effect that has filtering and nonlinear distortion capabilities, the ProCo Rat 2, herein referred to as the "Rat". While TDMI is not itself a modeling framework, we investigate its utility as a diagnostic method.

We previously explored the use of TDMI by applying the technique to simulations in MATLAB, with causal, time-invariant, and stable systems such as delay lines, FIR and IIR filters. We also appended a nonlinearity to the IIR filters to observe how TDMI responds to nonlinear distortion. In this chapter, we focus our efforts on physical audio systems; specifically we selected a guitar stompbox effect for our analysis. Certainly this technique could be used on other analog devices, such as preamplifiers, amplifiers, and other effects, although additional tests such as these are beyond the scope of the present proof-of-concept work. We incrementally vary the Rat's Distortion and Filter parameters separately, pass an analysis signal through the device, capture the system response, and compute the TDMI at various time-delays τ for each test. For each replication at a given parameter setting, we assume the device is time-invariant. After a representative sample of tests and TDMI calculations, we then look into what sort of information can be gleaned from the TDMI calculations as each parameter is varied.

The organization of this chapter is as follows. First, we discuss some of the practical considerations when applying the TDMI method to analog devices, including audio interface selection, test signal generation, and test signal scaling. Then, we discuss a method for adjusting the device's parameters in a controlled and measurable way to allow experiment repeatability. Next, we describe in detail the choices made for the Distortion and Filter settings, as well as the data collection process. After collecting our data, we then describe the TDMI calculation process, as well as two relevant tangents that arose during

experimentation: measuring phase shift while adjusting Filter settings, and performing TDMI calculations with fractional delay lines. We then interpret our findings and discuss general trends observed from the experiments, pointing out directions for future work.

4.3 Background and Motivation

In this section we briefly discuss distinctions between prior MATLAB experiments and the analog audio device experiments at hand. First, we discuss considerations when choosing an audio interface to pass signals into the device and collect the processed output signal from the device. We also must consider appropriate scaling of the test signal before it is converted from digital to analog. Next, because modulating device parameters now requires physically adjusting potentiometers, we discuss how to make precise changes with the ability to measure the settings for repeatability. After laying this groundwork, we can then focus on collecting data from the device under a variety of parameter settings, which we will use for subsequent calculations and analysis.

The systems we investigated in previous chapters were all pre-determined discrete-time systems in MATLAB that did not require any conversion between the digital and analog domains. This chapter deals with the practical considerations when applying the measurement and diagnostic techniques to a ProCo Rat 2 guitar effects pedal (Figure 4.1), which exhibits adjustable nonlinear distortion and features a passive RC low-pass filter with variable cutoff frequency. For instance, with a guitar pedal the range of possible input signal levels is limited by the op-amp rail voltages. When working with digital filters or other math operators in MATLAB, we have fewer practical constraints that dictate the test signal's amplitude; rather, we care mostly about the relationship between the input and output signals.

To interact with the effects pedal, we need an audio interface to pass test signals into the device and to record the device's corresponding output signal. Because the pedal is a

(a) Top view of ProCo Rat 2 guitar pedal. Knobs adjust potentiometers associated with the Distortion, Filter, and Volume circuits; the effect is engaged with the footswitch.

(b) Rear view of ProCo Rat 2 guitar pedal. The monophonic input and output signals are passed via 1/4" tip-sleeve connections.

Figure 4.1: ProCo Rat 2

single-input, single-output effect, we recommend an audio interface with at least two input channels to facilitate time-synchronized capture of the test signal and the system output. The choice of sampling frequency should also be considered, especially if trying to observe phenomena that affect spectral content beyond the interface's Nyquist frequency. In our tests, the interface was running at 48kHz which corresponds to a 24kHz Nyquist frequency. Further, as we are converting our digital test signals into analog signals, we want to know the relationship between digital amplitude and analog voltage; calibrating the audio interface with an oscilloscope gives us this understanding. Once an audio interface is selected and calibrated, we need to determine an appropriate test signal level for feeding the audio device. For instance, because this guitar pedal is designed to process weak, un-amplified signals from a magnetic guitar pickup, we must determine an appropriate level for our test signal that does not cause unintended distortion. We then use this maximum input signal level to scale

our Gaussian white noise measurement signal.

Once we have an appropriately-scaled Gaussian white noise signal to conduct our measurements, data collection comprises the bulk of the experiment. This collection is done by passing the noise signal through the device and recording the device's output. To investigate the pedal's behavior given various parameter settings, we replicate the measurement process after changing the parameters' potentiometers, removing them from the circuit, recording their resistance values, and placing them back in the circuit. We replaced the potentiometers' through-hole soldered connections with header pins and mating cables to streamline this process, retaining the pedal's original circuit behavior. We ensure repeatability by documenting the device settings for each test. For a given parameter setting, we then pass the analysis signals through the pedal and record the output. Further analysis takes place after gathering this data for a range of pedal settings. Mainly, we are interested in what time-delayed mutual information can reveal about the guitar pedal's nonlinear distortion and low-pass filtering.

4.4 Overview of Device-Under-Test

The ProCo Rat 2 (herein, the "Rat") was chosen as the device-under-test because it features a diode "soft-clipping" nonlinear distortion and has memory in the form of an RC low-pass filter. This type of clipping distortion deliberate in design, enabling the user to attain a specific sound timbre. That the author already possesses one of these devices was also a key decision factor. By memory, we mean that the signal processing chain contains an energy-storage element, in this case, the capacitor C in the passive RC low-pass filter. The simplified block diagram in Figure 4.2 shows that the Rat is designed to process the unamplified signal from an electric guitar before it reaches the guitar amplifier in the signal chain. The Rat introduces intentional harmonic distortion to an electric guitar signal, creating an aggressive, buzzy sound. There are three controls exposed to the user in the

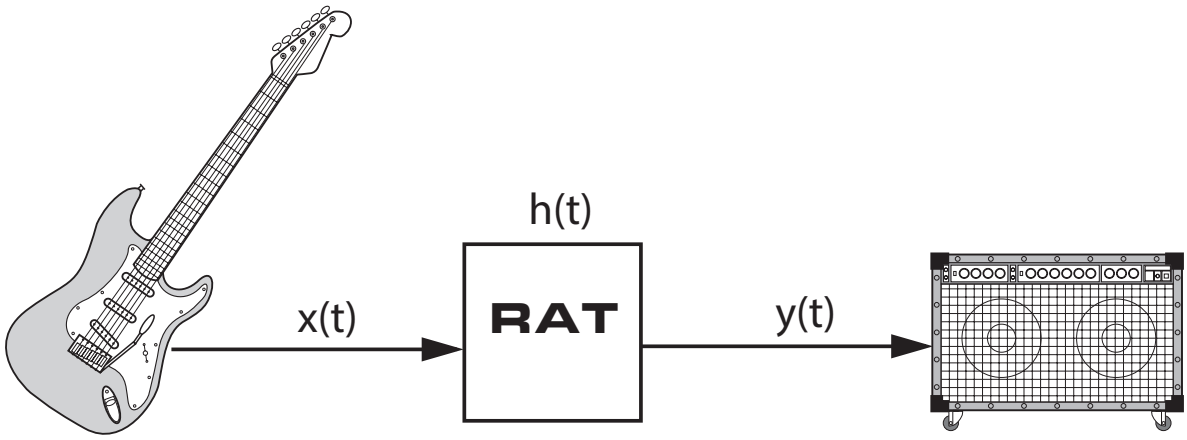


Figure 4.2: Block diagram indicating typical connection of Rat between an electric guitar and a guitar amplifier.

form of linear-taper, 100-kΩ potentiometers: Distortion, Filter, and Volume. Respectively, these controls allow the user to adjust overall distortion level, high-frequency attenuation, and final volume level of the effected signal. An overview of the Rat's signal processing chain is shown in the simplified schematic, Figure 4.3.

In the case of the Distortion parameter, the potentiometer alters the gain of the op-amp based amplifier shown in the first dashed-box subsection, labelled "Distortion" in Fig. 4.3. This op-amp gain stage also include two active high-pass filters, but the main property of interest is increasing the input signal amplitude so as to engage the dual-diode soft-clipping stage immediately following the amplifier. After the distortion stage, the signal then flows into the filter stage. Here, the Filter parameter alters the cutoff frequency of a passive RC low-pass filter shown in the second dashed-box subsection, labelled "Filter" in Fig. 4.3. This allows the user to attenuate some of the high frequency content of the distorted signal to their preference. The cutoff frequency range for this filter stage is between about 620 Hz and 36 kHz; lower potentiometer resistances correspond to a higher cutoff frequency or more restrictive passband. Finally, the Volume control allows the user to attenuate a

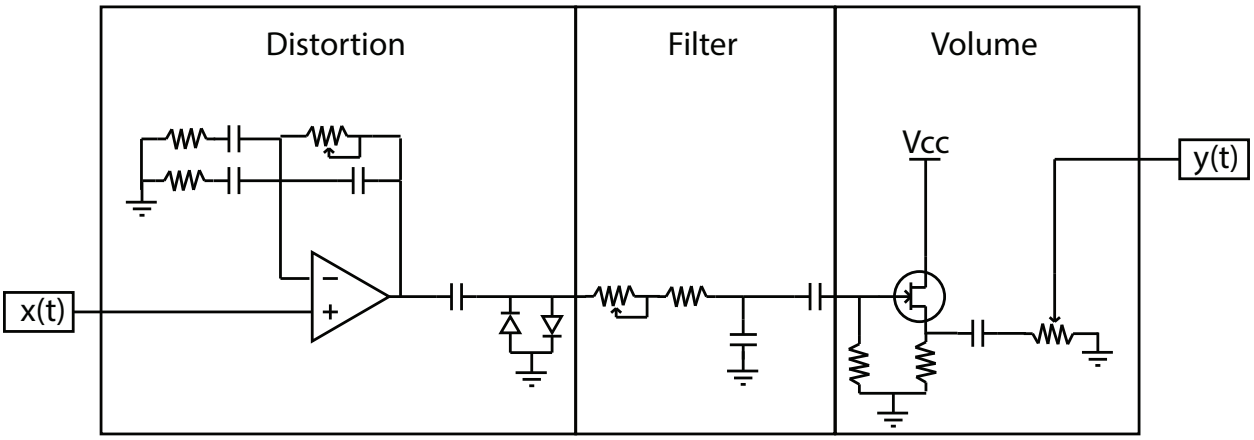


Figure 4.3: Rat op-amp gain and diode distortion stage. Signal enters on the non-inverting input of the op-amp and passes to the next processing block from the anode of the right-most diode. Op-amp gain is controlled by the Distortion potentiometer; as the signal level into the diodes increases, more of a soft-clipping effect is achieved. The distorted signal is then passed through the RC low pass filter, whose cutoff frequency is variable via the Filter potentiometer. The filtered signal is then buffered, and its amplitude can be attenuated by bleeding to ground through the Volume potentiometer.

buffered version of the distorted and filter signal; see the final dashed-box subsection labelled "Volume" in Fig. 4.3. In our case, we leave this setting at maximum, opting to control the overall volume level by selection of the input signal amplitude.

4.5 Experiment

4.5.1 Experiment Setup: Audio Interfacing

In this section we discuss the practical steps of generating a test signal, passing it to the device-under-test via an audio interface, and capturing the test system's response. We also explain our method for systematically altering the device parameters for each test by adjusting the potentiometers and measuring their resistance outside of the circuit before re-inserting it. We build our data set by recording the system's response to a given test signal at various documented operating points.

Figure 4.4: Expert Sleepers ES-8 USB audio interface.

In order to pass pre-recorded test signals into the Rat and to capture its response, we connected a USB audio interface to our computer running the free, open-source digital audio workstation, Audacity (https://www.audacityteam.org). We used an Expert Sleepers ES-8 class-compliant USB audio interface (Figure 4.4), which was readily available at the time of experimenting. The audio interface was configured for a sampling frequency of 48kHz and 16-bit bit depth. It is assumed that any similar professional analog/digital interface system would be equally appropriate in this context. It is worth mentioning that with a 48kHz sampling frequency, we will only be capturing frequency content up to the Nyquist frequency of 24kHz. This extends beyond the 20kHz approximate upper threshold of human-audible frequencies A block diagram showing the connections between the computer, audio interface, and Rat is shown in Figure 4.5.

Once we are ready to excite the Rat and capture its output, the next step is to determine the ideal signal level for our test signal, a pseudo-random Gaussian white noise sequence of 10^6 samples. However, now that we are dealing with an analog system with limited headroom,

we must carefully choose a signal level that won't unintentionally cause distortion but provide an adequate signal-to-noise ratio in data captures. The ES-8 audio interface has a maximum output signal amplitude of 10V, being designed for use with a Eurorack modular synthesizer system. This maximum amplitude far exceeds the headroom of the Rat, which uses a 9V DC power supply and a single-supply op-amp gain stage with a 4.5V operating point.

The process for determining an appropriate scaling for our white noise is as follows. First, we set the Rat's Distortion and Filter controls to their lowest positions to minimize the distortion effect and to allow as much high-frequency audio to pass through as possible. We set the Volume control to its maximum position so as not to unnecessarily attenuate the output of the Rat. With the pedal configured in this way, we then created a 1kHz sine test signal in MATLAB whose amplitude is 1 and is decreased by $2^{-(n/2)}$ every second for $n = [0, 20]$, or, approximately decreased by -3dB every second. Using the `audiowrite` MATLAB command we exported this signal to a 16-bit, 48kHz sample rate .wav audio file to be loaded as a monophonic audio track in Audacity, our digital audio workstation indicated in Figure 4.5. Then, we played back this audio track through the Rat, recording its output to a new mono audio track. Test signal attenuation levels of greater than about -30dBFS all resulted in undesired distortion, presumably from engaging the diode clipping stage without the assistance of the Distortion op-amp gain stage. Attenuating the test signal by -30 dBFS corresponded to a scaling of 1/32 of the original sine wave. This signal level, coming from the ES-8 audio interface, results in an analog sine waveform with an amplitude of 312.5mV or 10/32 volts. With a non-distorting signal level chosen, the next step is to create our scaled Gaussian white noise signal using the `randn` function. We created a zero-mean sequence of 10^6 samples with a standard deviation of 1, then normalized it so that the magnitude of the largest sample is 1/32, or 312.5 mV when converted to analog. Thus, the scaled noise signal will have a maximum power of about -30dBFS where this largest-magnitude sample occurs. As with the stepped-amplitude sine wave, we export this MATLAB vector as a 16-bit,

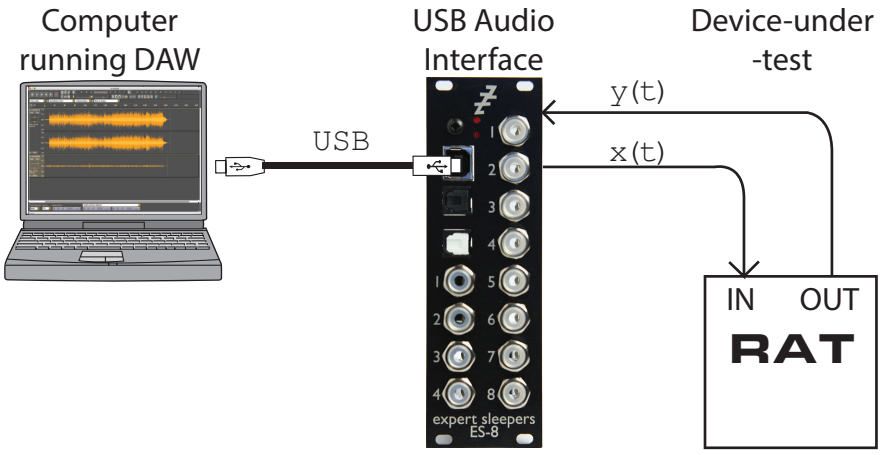


Figure 4.5: Rat audio measurement setup. A digital-audio workstation on the computer simultaneously sends the test signal $x[n]$ and records the output signal $y[n]$ from the pedal via the USB audio interface, which performs the output D/A and input A/D conversion.

48kHz sample rate .wav audio file with the `audiowrite` command. Finally, we also created a fixed-amplitude 1kHz sine waveform audio file whose amplitude is also 1/32 or 0.3125. This allows us to audition the effect of the Rat given different control settings, to look at the pedal's phase response at this frequency, and generally to capture time-domain signals whose audition and visual plots clearly demonstrate the distortion and filtering effects.

4.5.2 Experiment Setup: Pedal Settings

The ProCo Rat 2 has two particular characteristics of interest: a nonlinear distortion stage and memory in the form of a passive, single-pole RC low-pass filter. Given these two features, we can estimate both the degree of nonlinearity and the memory extent, as previously explored in Chapters 2 and 3, respectively. The data collection process is the same for both chapters: generate a Gaussian white noise signal, pass it through the system, and record the system output. To get a broad picture of the pedal's behavior, we must capture the pedal response at various settings: we chose to look at the effect of the Distortion and Filter parameters separately. For the Distortion tests, we varied the Distortion parameter

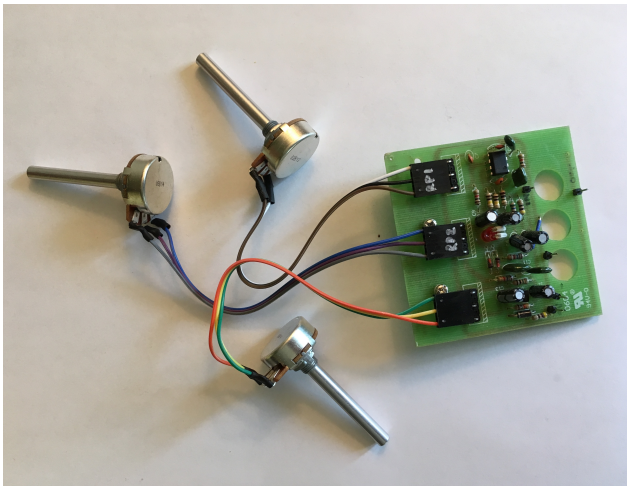


Figure 4.6: ProCo Rat 2 PCB with modified potentiometer connections.

while setting the Filter parameter such that minimal low-pass filtering occurred. For the Filter tests, we did the opposite. In both cases, the Volume control was set to maximum to avoid unnecessary attenuation as previously described.

In order to reproduce our data, we modified the guitar pedal by physically removing the Distortion, Filter, and Volume potentiometers from the printed circuit board and replacing them with header pins to which we can quickly connect and disconnect the potentiometers; see Figure 4.6. This allows us to easily change and measure the potentiometer setting without affecting the original circuit behavior.

4.5.3 Experiment: Distortion settings

To study the effect of the Distortion parameter, we drove the pedal with a 1kHz sine test signal, gradually increasing the Distortion potentiometer setting while qualitatively observing changes in the harmonic content of the output spectrum with the Analog Discovery 2. Figure 4.7 shows the original input sine wave in dark blue with a few output waveform examples captured under various Distortion settings. Once we reached a new distortion level, we measured the resistance between the potentiometer wiper (Pin 2) and the other

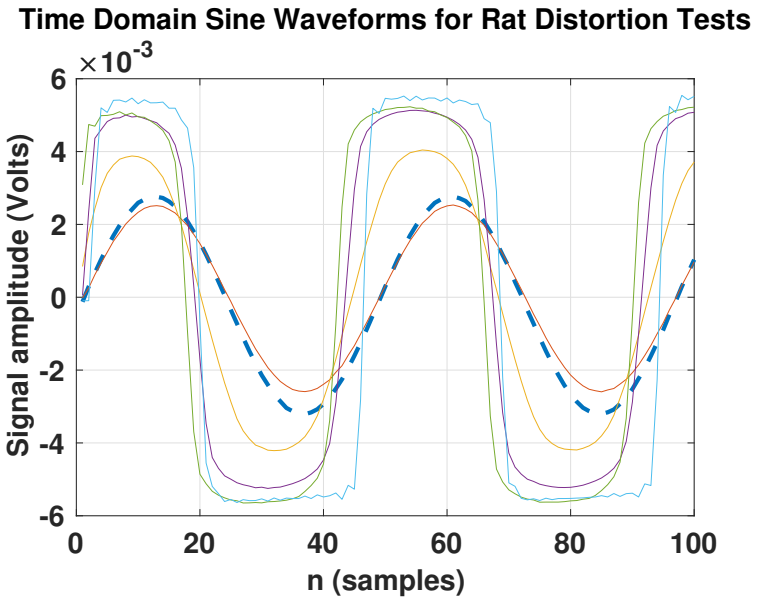


Figure 4.7: Time-domain plots of the original 1kHz test sine wave (dotted line) and output waveforms for various Distortion levels.

two terminals (Pins 1 and 3). The resistance R_{23} between Pins 2 and 3 determines the op-amp gain; see Figure 4.3. The resistance R_{21} between Pins 2 and 1 does not interact with the op-amp circuit, but is documented for completeness. Table 4.1 contains measured Distortion potentiometer resistances.

After adjusting the Distortion potentiometer to a new setting and documenting the changes, we began data collection. First, we pass the Gaussian white noise test signal through the pedal, recording the output into a new audio track in Audacity. The original and distorted noise signals will be used later for TDMI calculations. Then, we pass through a 1kHz sine signal of the same duration as the white noise, recording the pedal's output. While we don't use these distorted sine signals for any subsequent calculations, it is helpful for qualitative auditory and visual inspection.

Test No.	R_{23} (Ω)	R_{21} (Ω)
A	89200	4.90
B	89100	106
C	88600	450
D	88900	725
E	88200	1330
F	87300	2120
G	86900	2580
H	89100	20.7
I	85700	3540

Table 4.1: ProCo Rat 2 Distortion Settings/Potentiometer measurements

4.5.4 Experiment: Filter settings

To study the effect of the Filter parameter, we measured the minimum and maximum resistance values R_{pot} of the Filter potentiometer, then determined the respective maximum and minimum possible cutoff frequencies of the RC low-pass filter using Equation 4.1.

$$f_c = \frac{1}{2\pi(R_{pot} + 1600)C} \text{ Hz} \tag{4.1}$$

The $(R_{pot} + 1600)$ term in Equation 4.1 accounts for the 1.6-kΩ resistor in series with the Filter potentiometer R_{pot} as shown in Fig. 4.3. Table 4.2 shows the range of Filter potentiometer resistance values R_{pot} that correspond to ten linearly-spaced low-pass filter cutoff frequencies. The exact values of R_{pot} and the filter cutoff frequency are not critical; the intent here is to cover a representative sampling of the possible range of cutoff frequencies. Because we are running the audio interface at a 48kHz sampling frequency, we may not detect any interesting behavior for Test Numbers QA-TA, where $f_c \geq$ 24kHz, the Nyquist

Test No.	R_{pot} (Ω)	Cutoff Freq. (Hz)
QA	5.1	36724
RA	199.3	32761
SA	445	28825
TA	779	24778
UA	1260	20611
VA	1964	16539
WA	3063	12641
XA	5340	8493.7
YA	11210	4601.6
ZA	93500	619.83

Table 4.2: ProCo Rat 2 Low-pass Filter Cutoff Frequencies tested. The exact values of R_{pot} and the filter cutoff frequency are not critical, rather, they cover a representative range of parameter settings.

frequency. Nevertheless, with the list of R_{pot} settings used shown in Table 4.2, we passed our scaled Gaussian white noise signal through the Rat via our audio interface, capturing the output on a mono input channel on the interface.

4.5.5 TDMI Calculations

Upon collection of the Rat's response to our input test signals under various settings combinations, we then move on to the TDMI (time-delay mutual information) calculations to see how TDMI changes as we vary the Distortion and Filter controls.

To calculate the TDMI between our input $x[n]$ and output $y[n]$ signals we use the same method as in Chapter 3. First, we load into memory $x[n]$ and $y[n]$ from a single trial, located in a MATLAB struct containing our test data. We then create $x_s[n]$, a version of $x[n]$ whose samples are randomly shuffled once, for use in the estimator bias calculation

later on. From these signals we store the signal length N in samples, as well as $\sqrt{N}$, our bin count for subsequent histogram calculations. Second, we create a vector of time lags τ over which to compute the TDMI. In the cases that follow, we empirically chose all integer values $\tau = [-5, 35]$ for our time lags, as this range seemed to capture all significant TDMI values without including too many values that fell below the estimator bias or noise floor. Once the time lags are specified, we repeat the following steps for each τ:

1. Time-shift $y[n]$ by τ samples and store the resulting $y[n-\tau]$ in a temporary variable.

2. From the signals $x[n]$, $x_s[n]$, and $y[n-\tau]$, estimate the probability mass functions $\hat{P}(X)$, $\hat{P}(X_s)$, and $\hat{P}(Y)$ respectively with the MATLAB function `histogram` with $\sqrt{N}$ bins.

3. Use `histogram2` with $\sqrt{N}$ bins to estimate the joint probability mass functions $\hat{P}(X,Y)$ of $x[n]$ and $y[n-\tau]$, and $\hat{P}(X_s,Y)$ of $x_s[n]$ and $y[n-\tau]$.

4. Compute the estimated marginal and joint Shannon entropies $\hat{H}_x$, $\hat{H}_{x_s}$, $\hat{H}_y$, $\hat{H}_{x,y}$, and $\hat{H}_{x_s,y}$.

5. Compute TDMI at the current τ between $x[n]$, $y[n-\tau]$ using $\hat{H}_x + \hat{H}_y - \hat{H}_{x,y}$.

6. Compute estimator bias at the current τ using $\hat{H}_{x_s} + \hat{H}_y - \hat{H}_{x_s,y}$.

After repeating the above steps, we save the calculated TDMI and estimator bias for each lag τ for a given pair of $x[n]$ and $y[n]$. For fast data recall, we append the results back to the original MATLAB test data struct and save it as a .mat file.

4.5.6 Measuring phase shift: Filter tests

In addition to the TDMI calculations for the Filter tests above, we also calculated the phase shift between a 1kHz sine input signal and the output signal from the Rat for the

Filter tests mentioned before. To measure the phase shift for each Filter setting, we loaded into memory the recorded audio of both the input sine $x[n]$ and the filtered output $y[n]$, measuring the number of samples between both zero-crossings. Plots of $x[n]$, $y[n]$, and their zero crossings are shown in Figures 4.14 and 4.15 for each Filter test.

4.5.7 TDMI Tests with Fractional Delay Lines

As will be discussed in the Results section, there are some Distortion and Filter tests where there is not a distinct TDMI peak but two data points between which the maximum TDMI probably lies, e.g., Tests C, D, and UA to name a few. One explanation is that for these tests, the sampling period $T_s = 1/48000$ sec is not short enough to properly represent the mutual information at sub-sample intervals. This could be overcome by either using a higher sampling frequency F_s or by upsampling $x[n]$ and $y[n]$ in post-processing.

To explore this sub-sample TDMI phenomenon in more detail, we perform FIR fractional delay filtering in MATLAB using Lagrange interpolation [29]. The eleven simulated delay line lengths tested range from $n = 2.0 : 0.1 : 3.0$. Plots containing a single sample impulse $x[n]$ and the delayed output $y[n]$ are shown in Figures 4.8 and 4.9 with the actual fractional delay value indicated with a vertical dotted line. TDMI plots with interpretation are shown later in the Results section.

4.6 Results

In this section, we present the TDMI plots for the various Distortion and Filter tests, as well as a comparison between the phase of an input 1kHz sine wave and its output for the Filter tests. For the TDMI plots, we classify significant TDMI values where the TDMI curve (blue) is greater than the estimator bias (dashed orange). The estimator bias indicates the lower threshold of significant TDMI–it is generated by computing the TDMI between $y[n]$ and the randomly-shuffled $x_s[n]$.

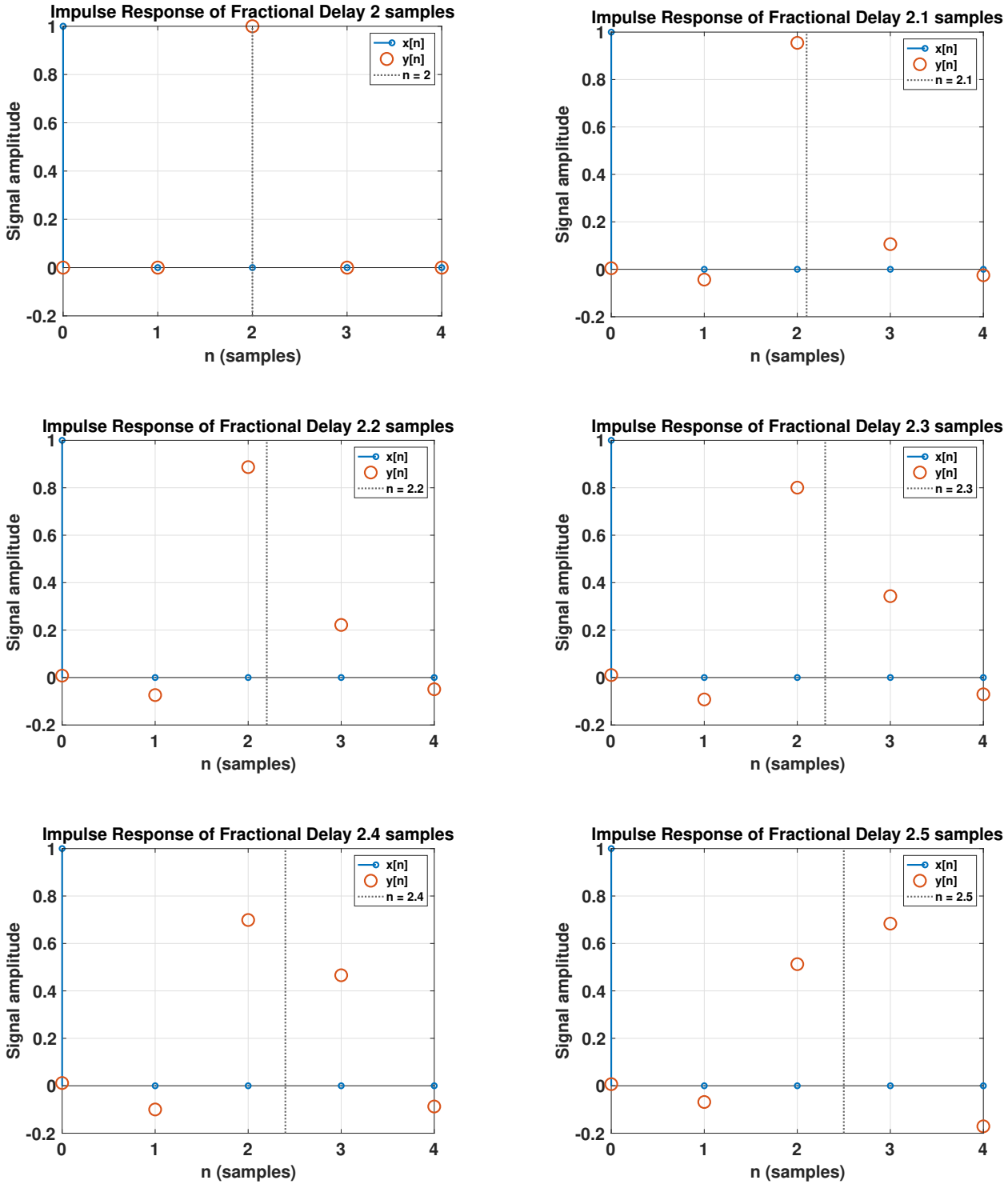


Figure 4.8: Impulse Responses for Fractional Delay lines of length $2 : 0.1 : 2.5$

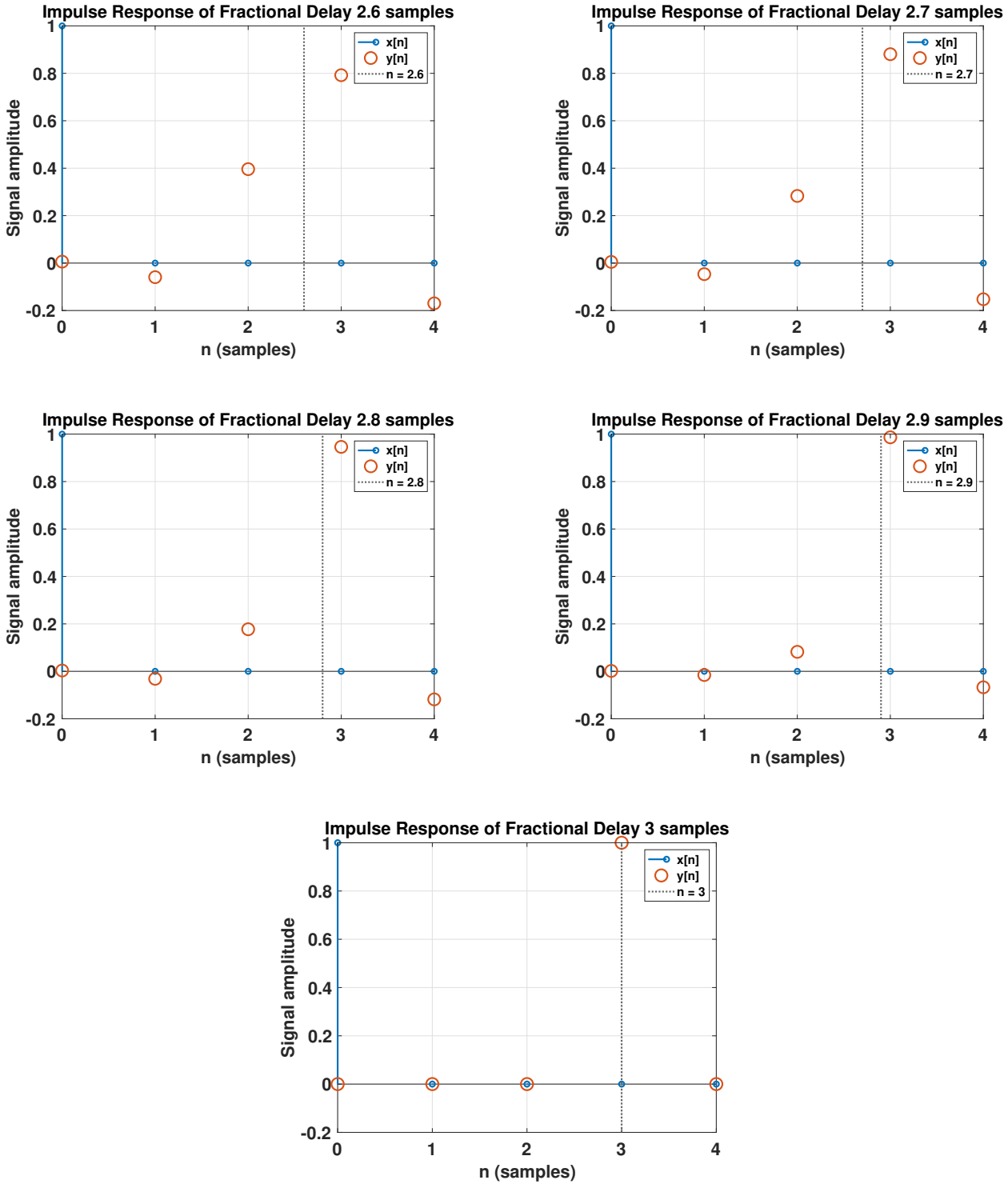


Figure 4.9: Impulse Responses for Fractional Delay lines of length 2.6 : 0.1 : 3.0

4.6.1 TDMI interpretation: Distortion tests

The plots in Figures 4.10 and 4.11 represent the calculated TDMI and estimator bias for each of the Distortion tests listed in Table 4.1. Note that Test H was excluded for clarity: its Distortion level lies between that of Tests A and B, and including its plot would detract from the subsequently-increasing Distortion setting in each figure.

As a first observation, when the Distortion setting is increased, the peak of the TDMI curve occurs at higher time-delays of τ samples. That is, the greatest mutual information between the input noise signal $x[n]$ and the distorted output signal $y[n]$ moves from $\tau = 0$ toward $\tau = 1$ for the range of values we tested. In Tests C and D, the y-distance between the TDMI peak and the next integer τ value decreases. Because the audio signals used were generated with a 48kHz sampling frequency, it is possible that the TDMI peak occurs at a non-integer multiple of the sampling period. In other words, the TDMI peak in Tests C and D may lie somewhere between $1 \cdot T_s$ and $2 \cdot T_s$, where $T_s = 1/48000$ seconds.

Another observation is that as the Distortion level increases, the overall estimator bias or noise floor increases from about 0.2 to 0.4. To explain this, recall that for a given test, the estimator bias is the TDMI between the randomly-shuffled input $x_s[n]$ and the output signal $y[n]$. Thus, as $y[n]$ is essentially an increasingly-distorted version of $x[n]$, it seems reasonable that it will have less mutual information compared to a shuffled version of the undistorted $x[n]$.

Finally, when comparing the TDMI curves against that of the Filter tests that follow, we notice that the overall shape does not change as the Distortion parameter is modulated. That is, there is generally a pronounced peak at some value τ that is time-shifted based on the level of Distortion present. We contrast this with the Filter curves, especially that of Test ZA, where as the Filter cutoff frequency is significantly lowered, the maximum TDMI value is significantly smaller than less-selective Filter tests, with a less steep trailing off of TDMI over τ after the primary peak.

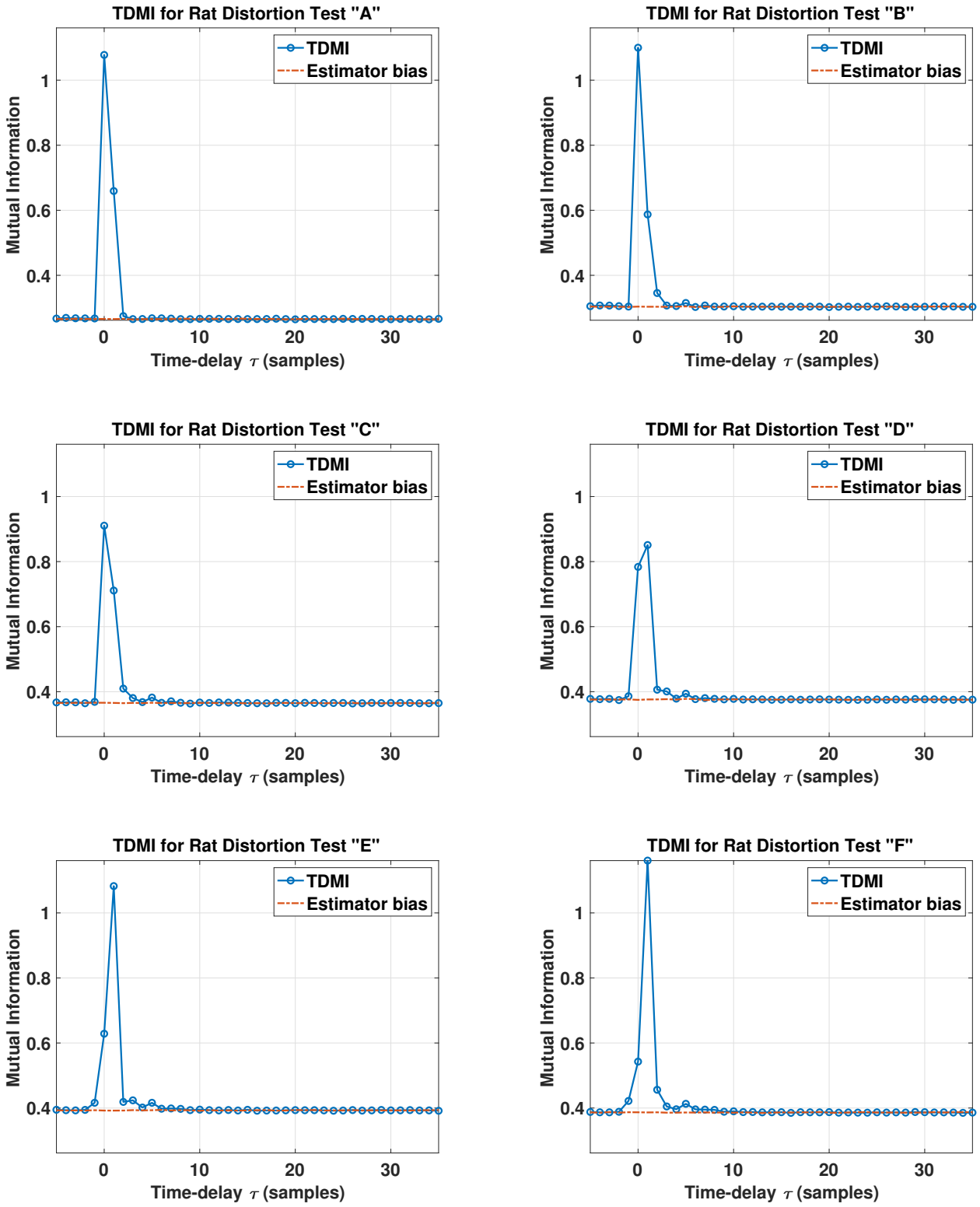


Figure 4.10: TDMI results for Rat Distortion Tests A-F

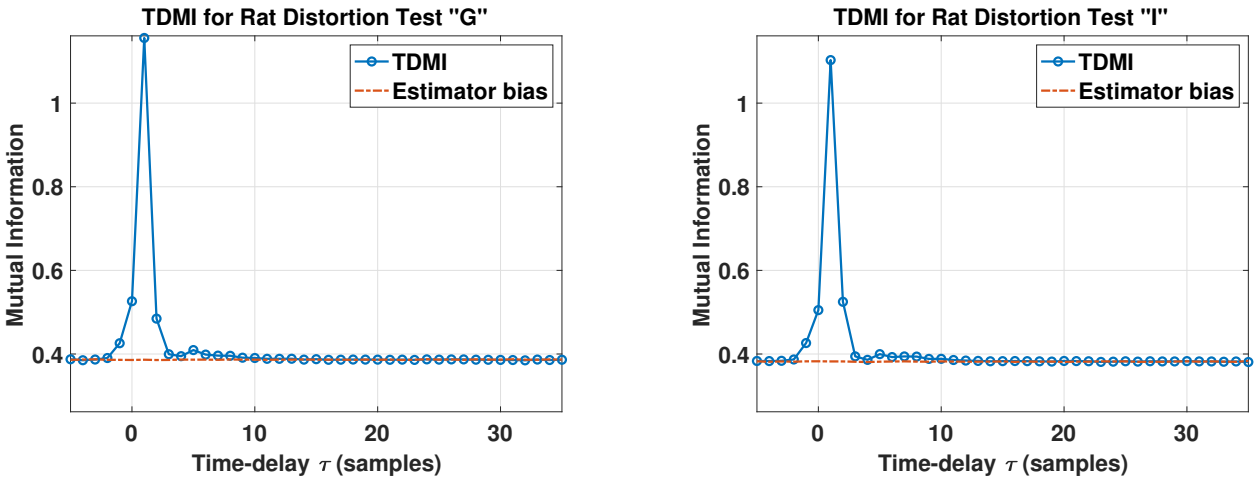


Figure 4.11: TDMI results for Rat Distortion Tests G and I

4.6.2 TDMI interpretation: Filter tests

The plots in Figures 4.12 and 4.13 represent the calculated TDMI and estimator bias for each of the Filter tests QA-ZA listed in Table 4.2. Once again we note that we used the USB audio interface at a sampling frequency of 48kHz, which corresponds to a 24kHz Nyquist frequency. As a result, we likely did not capture any interesting filter behavior for Tests QA, RA, SA, and TA whose cutoff frequencies ranged from 36.7 to 24.8kHz. In any case, similar to the Distortion tests, as we lower the Filter cutoff frequency by about 4kHz in each test, the TDMI peak shifts from $\tau = 0$ to higher values of τ. Another similarity is that as the Filter cutoff frequency is lowered, the maximum TDMI value may occur at non-integer multiples of τ, or $T_s = 1/48000$ seconds. If we used a higher sampling frequency, such as 192kHz, we might have more time-resolution for the TDMI data corresponding to these Filter settings.

One key difference compared to the Distortion tests is that for Test ZA, where the lowest tested cutoff frequency $f_c \approx 620$Hz, there is a gradual decay from $\tau = 2$ to about 20 samples of TDMI after the peak at $\tau = 1$. A possible explanation is that as the cutoff frequency

decreases, the RC time constant increases, causing the filter output to reach steady state in a longer time period. Further, Test ZA's maximum TDMI value of 0.272 is significantly lower than the previous tests QA-YA, whose maximum TDMI values vary between 0.865 and 1.24. This may be because the highly filtered output signal $y[n]$ in Test ZA has much less in common with the unfiltered input noise signal $x[n]$ compared to the previous tests with higher filter cutoff frequencies.

In an effort to draw a relationship between TDMI and the Filter controls, we next investigate the phase shift of a 1kHz input sine signal and the output of the Rat at the various Filter settings.

4.6.3 Phase comparison: Filter tests

Figures 4.14 and 4.15 show the measured phase shift at the zero-crossing of an input 1kHz sine and its output from the Rat. The phase shift between the zero-crossings of $x[n]$ and $y[n]$ is 0 samples between the 1kHz sine wave in Filter Tests QA-UA, where cutoff frequency f_c was 36.7 kHz, 32.8 kHz, 28.8 kHz, 24.8 kHz, and 20.6 kHz. A phase shift of 1 samples is measured for Filter Tests VA-XA, where f_c was 16.5 kHz, 12.6 kHz, and 8.49 kHz. A f_c of 4.60 kHz yielded a phase shift of 2 samples, and finally, at $f_c = 620$ Hz resulted in a phase shift of 8 samples. As the sampling period T_s is 1/48000th of a second, each n corresponds to 0.131 radians/sample or 7.5 degrees/sample, according to:

$$\frac{2\pi \cdot 1000[\frac{\text{rad}\cdot\text{Hz}}{\text{sample}}]}{48000\ [\text{Hz}]} = 0.131 \text{ radians/sample}$$

and

$$\frac{360 \cdot 1000[\frac{\text{deg}\cdot\text{Hz}}{\text{sample}}]}{48000\ [\text{Hz}]} = 7.5 \text{ deg/sample}$$

Comparing these phase shifts with the Filter test TDMI plots in Figs. 4.12 and 4.13,

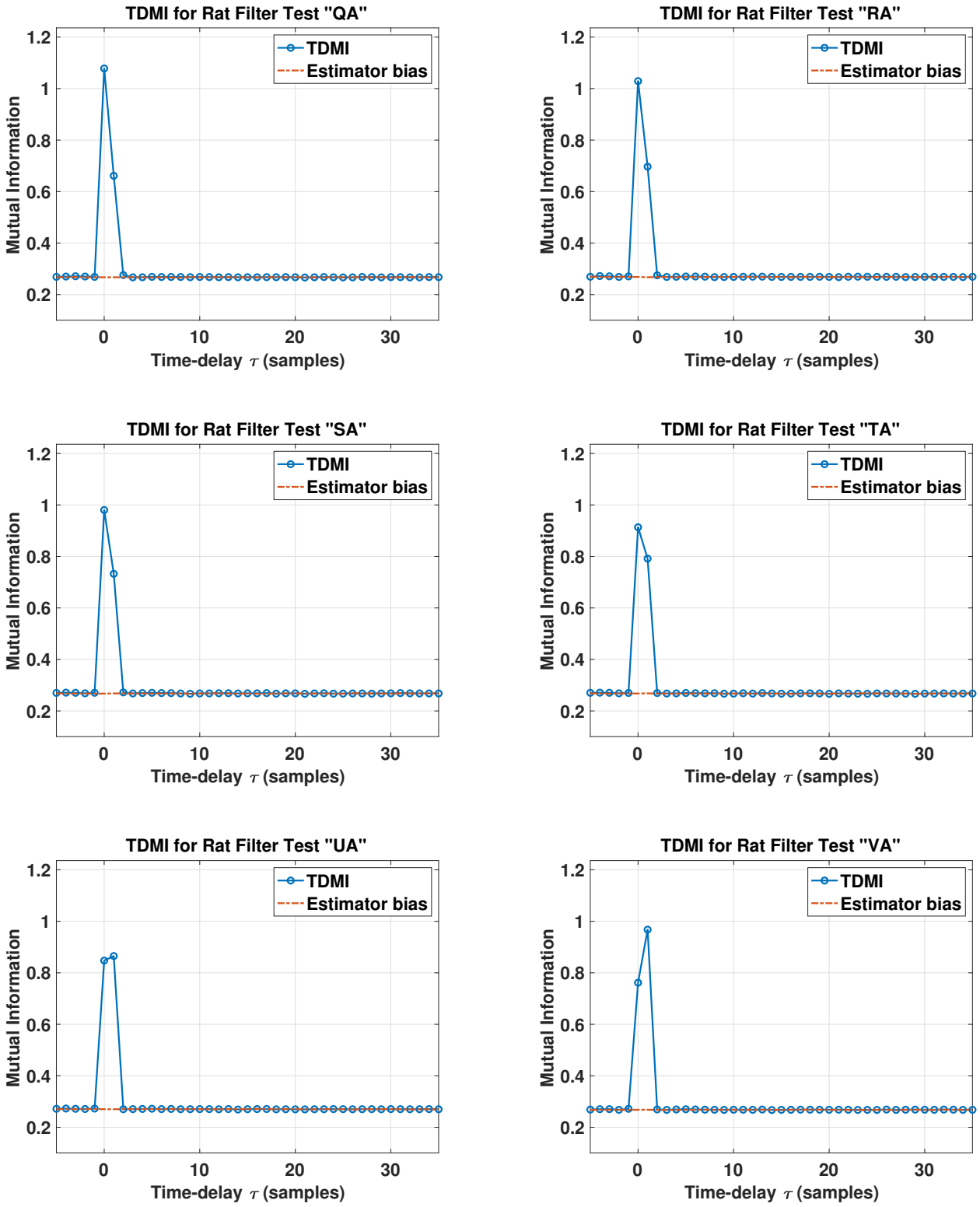


Figure 4.12: TDMI results for Rat Filter Tests QA-VA

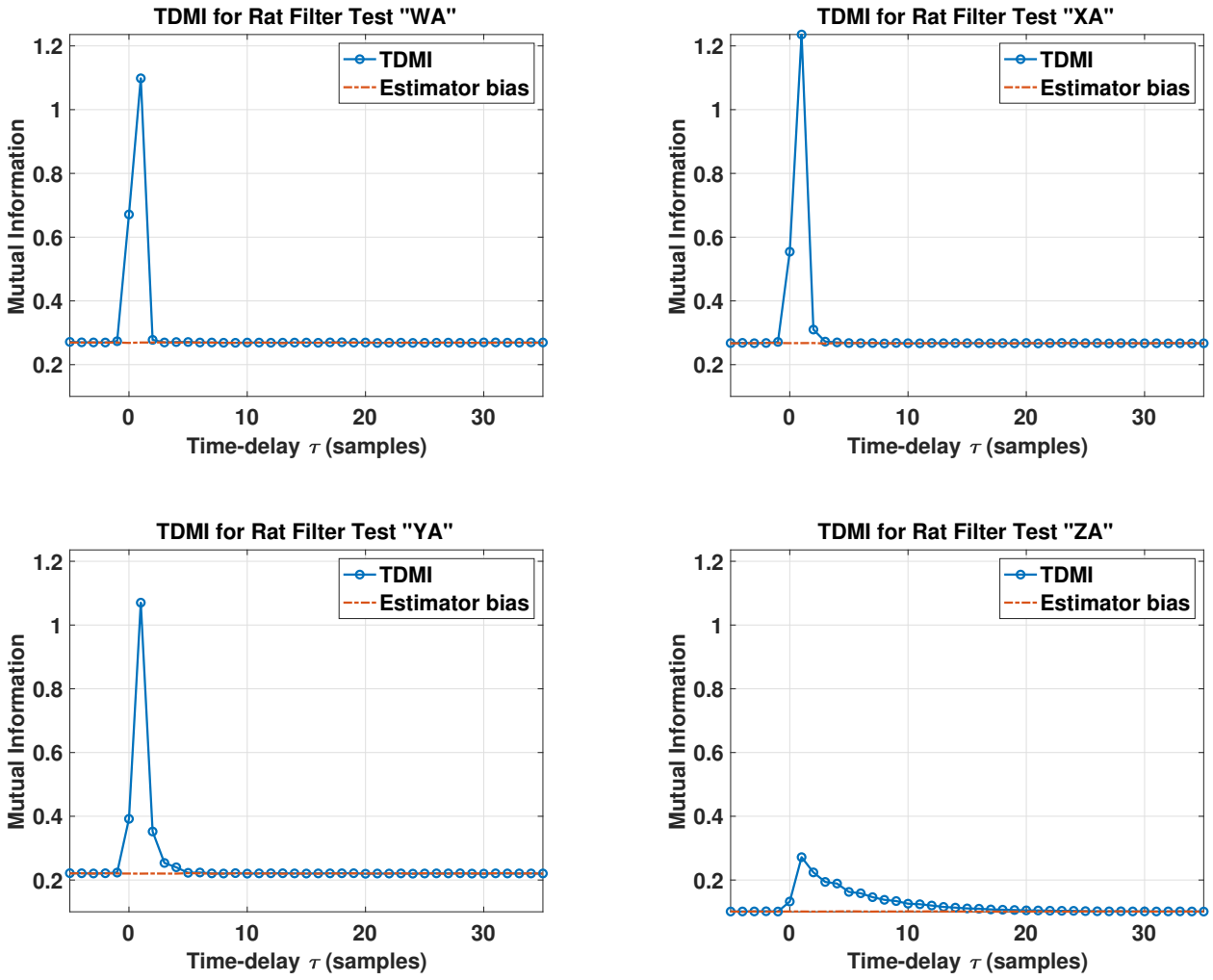


Figure 4.13: TDMI results for Rat Filter Tests WA-ZA

one possible explanation is that as f_c is decreased, the phase shift of the Rat at 1 kHz increases, causing the most mutual information between input $x[n]$ and output $y[n]$ to occur later in time at a given sample of $x[n]$.

4.6.4 TDMI interpretation: Fractional Delay line tests

Following the same TDMI calculation procedure in the Rat Distortion and Filter tests, we compute the TDMI for each of the fractional delay lines tested in MATLAB, shown in Figures 4.16 and 4.17. As can be seen in both the time-domain impulse response plots and the TDMI plots, the strongest relationship between the plots and the delay value occur at integer delay values. In other words, where fractional delay values exist, the maximum impulse response or TDMI values never occur exactly at the designed delay value.

4.7 Discussion

In both Distortion and Filter tests there is the possibility of a TDMI peak that occurs between samples. This may simply be an artifact of using a discrete-time signal and could possibly be addressed by using a higher sampling frequency if desired.

As Distortion is varied, the TDMI peak shape stays same but shifts in time. Estimator bias or noise floor increases. Thus, in the presence of nonlinear distortion, we expect the TDMI curve not to change significantly other than these attributes.

As Filter is varied peak shape generally stays same, shifts in time. As cutoff frequency is lowered, more of a tail begins to appear in TDMI curve after the maximum value. Then, Maximum TDMI value shrinks compared to tests with less selective passbands. Estimator bias also lowers as cutoff frequency lowers.

There seems to be a relationship between phase shift and Filter TDMI peak location based on our 1 kHz sine test. The most obvious explanation is that as cutoff frequency is lowered, there is more phase shift at this frequency.

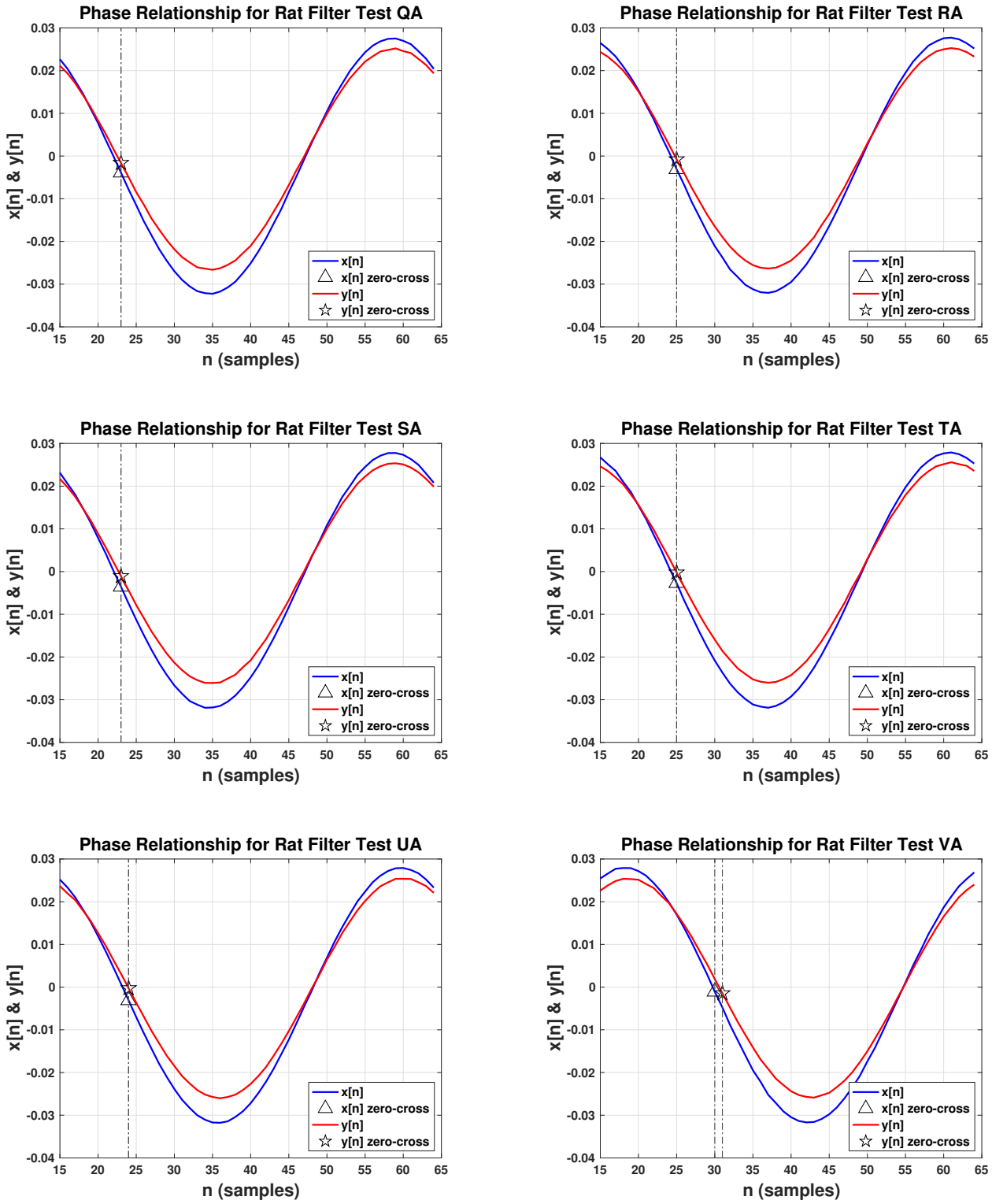


Figure 4.14: Phase Comparisons for Rat Filter Tests QA-VA

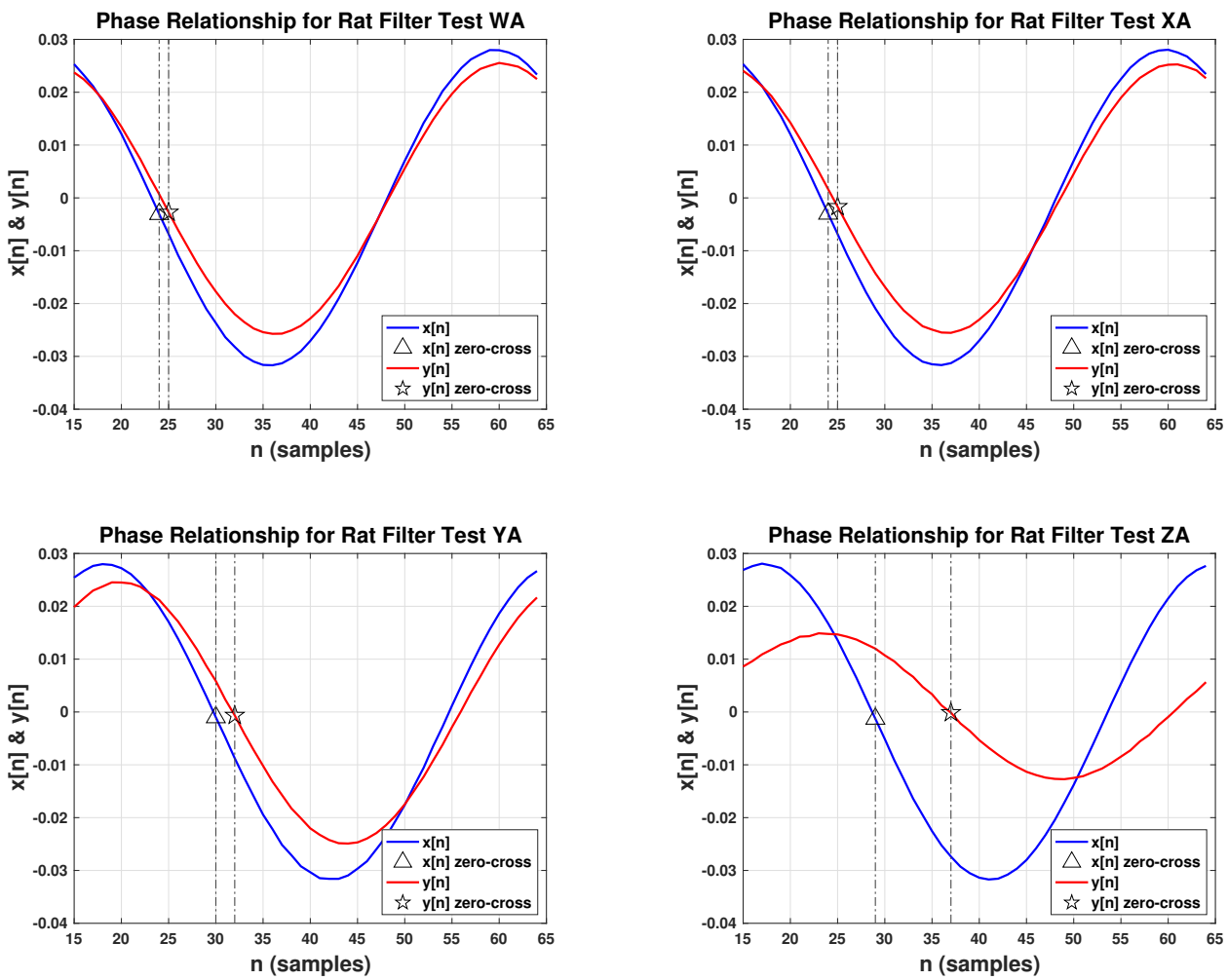


Figure 4.15: Phase Comparisons for Rat Filter Tests WA-ZA

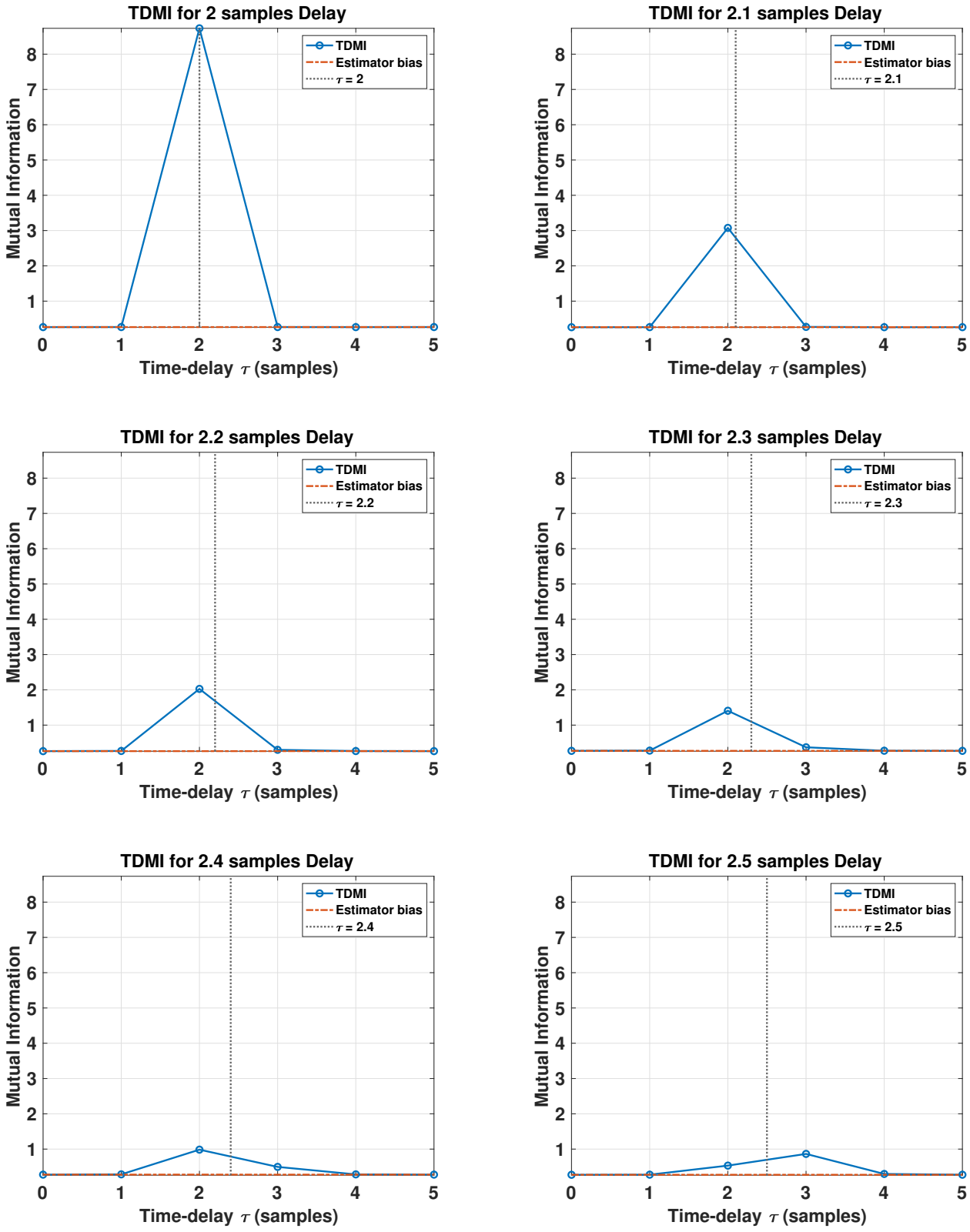


Figure 4.16: TDMI Calculations for Fractional Delay lines of length $2 : 0.1 : 2.5$

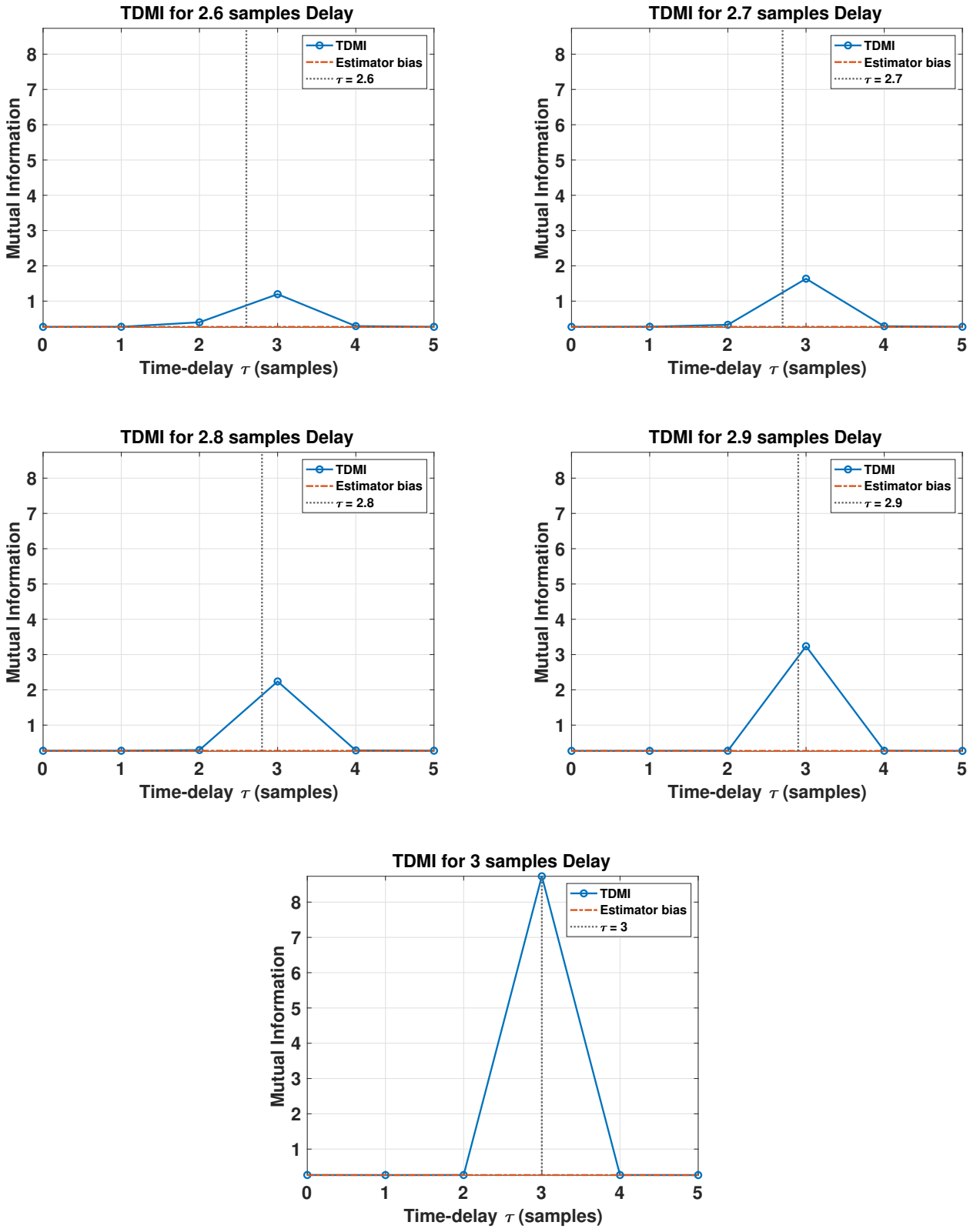


Figure 4.17: TDMI Calculations for Fractional Delay lines of length 2.6 : 0.1 : 3.0

4.8 Conclusions

In this chapter we investigated to what extent time-delayed mutual information (TDMI) can be useful in diagnosing the properties of a known analog audio system, the ProCo Rat 2 distortion pedal. To this end, we methodically varied the Distortion and Filter parameters separately, passing a test signal $x[n]$ of Gaussian white noise through the device, and recording the output $y[n]$ from the device. After building the data set, we then calculated the TDMI between $x[n]$ and $y[n]$ for each test across time-lags of $\tau = -5$ to 35. We also compute the TDMI estimator bias for the same values τ for each case to establish significant TDMI values.

In the tests where we increased the nonlinear distortion (Distortion parameter) from minimum to larger amounts, the TDMI curves in Figures 4.10 and 4.11 reveal a pronounced peak at $\tau = 0$ for no Distortion, with the TDMI peak moving further along the τ axis as Distortion is increased. In Tests C and D, which lack a distinct TDMI peak, we propose that the maximum mutual information between $x[n]$ and $y[n]$ occurs at a fractional delay value with respect to the sampling period, $T_s = 1/48000$ seconds. Thus, in the presence of increasing nonlinear distortion, we do not expect the general shape of the curve to drastically change aside from a time-delay of the maximum TDMI peak.

In the tests in which we gradually lowered the cutoff frequency f_c of the Rat's RC lowpass filter (Filter parameter), we similarly noticed a pronounced peak beginning at $\tau = 0$ and shifting further along the τ axis as f_c decreases. We point out that because the audio interface used a sampling frequency of 48kHz, we likely did not capture much of the lowpass filter's behavior for the cutoff frequencies $f_c \geq 24$kHz in Tests QA-TA. We also noticed the somewhat undefined peak phenomenon in Tests SA-WA as in the Distortion case, where the maximum mutual information likely occurs at a non-integer sampling interval. Of all the Filter tests, Test ZA for $f_c = 620$ Hz showed a significantly subdued peak with a longer

post-peak tail. We conjecture that as the RC filter in this configuration exhibits much mid- to high-frequency attenuation in the magnitude response, there is lower mutual information between $x[n]$ and $y[n]$. The tail of the TDMI curve in Test ZA is also lengthened compared to previous tests, which we attribute to the increased RC time constant of the single-pole passive filter.

To corroborate our theory that there is no pronounced TDMI peak where TDMI occurs at sub-sample values of τ, we applied the TDMI method to fractional delay lines with delay values of 2.0:0.1:3.0. Maximum TDMI occurs only at the integer delays $\tau = 2, 3$ and is lowest for 2.5 samples delay.

4.8.1 Future work

The general trends we observed are that TDMI is relatively insensitive to increasing nonlinear distortion except potentially in terms of phase shift or input-output delay. We also observed this phenomenon in the Filter cases except where f_c was 620 Hz, where the TDMI curve was significantly lower in overall level and exhibited a gradual decay to the estimator bias from the peak TDMI value. Though these were introductory experiments, we feel there are many additional experiments that could build upon our findings.

As a first future extension of this work, we varied the Distortion and Filter settings separately, so that there was never concurrent nonlinear distortion and low-pass filtering. Therefore it may be interesting to see how the TDMI curve changes in the presence of both nonlinear distortion and filtering, whether with this exact audio device or a similar system. Next, we did not try a comprehensive or methodical range of Distortion values as we did with the Filter control. This was not due to any theoretical or practical challenges, but was simply a qualitative selection of various distortion levels that proved interesting to the ear. In the Filter tests, we may have observed more interesting TDMI phenomena if we tried a sampling frequency higher than the 2×36.7kHz, the highest (theoretical) cutoff frequency

of the Rat's lowpass filter. Also, trying more cutoff frequencies between the 4.6 kHz and 620 Hz points may have revealed a gradual shift from a pronounced TDMI peak towards a lower-value TDMI peak with a more gradual decay to the estimator bias. Further, compared to the Filter tests, with the Distortion tests we did not explicitly seek a relationship between the TDMI measurements and the phase shift between input and output signals. As observed in both the Filter and Distortion tests, there were instances where the TDMI peak likely occurred at a fractional value of τ. Future tests may find that increasing the sampling frequency from 48 kHz to 96 or even 192 kHz may give better time-resolution in these instances in addition to capturing more of the Filter response for high cutoff frequencies.

Finally, part of the motivation for investigating TDMI as a diagnostic method when working with audio systems that exhibit memory and nonlinear distortion is to determine how many samples of the recovered transfer function to include when generating an impulse response or Volterra series model. Subsequent tests might consider finding a relationship between the TDMI measurements and how many samples are needed to create a satisfactory impulse response or Volterra series model of a given test device.

CHAPTER FIVE

CONCLUSION

This dissertation explores methods for diagnosing two properties of audio systems important for creating digital models: the system's degree of nonlinearity and the significance of recovered impulse response samples. When modeling a black-box, nonlinear audio system with a Volterra series representation, the work here informs which Volterra kernels to include and which values of the recovered linear kernel are significant. In this conclusion section, we summarize the key contribution of the dissertation by instructing the reader how to carry out diagnosis procedures that better inform Volterra series model design choices.

5.0.1 Diagnosing System Nonlinearity

As shown in the second chapter, we start by assessing the degree of nonlinearity of the test audio device, or determining which kernels to include when generating a Volterra series model. We created several Gaussian white noise signals whose variances σ_{id}^2 ranged logarithmically from 0.01 to 5, using the MATLAB function `randn` and a signal length of 10^6 samples. As discussed in the fourth chapter where we inspect an analog audio device, some care must be taken with respect to variance selection. When dealing with hardware audio devices, we advise choosing variance levels that do not unintentionally overload the device-under-test beyond what is desired, e.g., by railing out an op-amp.

Then, we pass each of the generated Gaussian white noise signals through the test device, simultaneously recording the input signal $x[n]$ to the device and the output signal $y[n]$ from the device. To accomplish this with an analog hardware audio device, we used a commercially available USB audio interface with at least two analog audio input channels. The choice of audio interface is not overly critical beyond the availability of at least a 44.1 kHz sampling rate, 16-bit sample resolution, two analog audio input channels, and one analog

audio output channel. From the recorded signals $x[n]$ and $y[n]$ we use the cross-correlation method from [19] to create several Volterra series models of the audio device whose kernels include everything from only the first-order or linear kernel, up to the fourth-order kernel.

Once the various Volterra series models are generated using the method from [19], we then analyze them by first creating a series of Gaussian white noise validation signals in the same manner as we created our identification signals. This time, we generate signals whose variances σ^2_{val} extend above and below the range of variances σ^2_{id} we used to create our identification signals, and we re-use our previously generated Gaussian white noise signals for the validation step. Then, we pass each of these test signals through each of the generated Volterra series models, save the output signals $\hat{y}[n]$ from each model, and compare the model response $\hat{y}[n]$ against the recorded response from the audio device $y[n]$ using Equation 5.1, saving this RRMSE result for each validation signal variance σ^2_{val}. To evaluate the relative performance of each model at a given variance σ^2_{val}, we take the difference of the RRMSE of each generated model against the linear model and against each lower-order model, resulting in a table of $\Delta RRMSE$ values. The two evaluation metrics we used were (a) the percentage of instances where a given model performed better than the linear and lower-order models indicated by a negative $\Delta RRMSE$ value, and (b) the sample mean of the $\Delta RRMSE$ for each such instance of better model performance. As an example taken from the second chapter, the left group of bar graphs in Figure 5.1 shows the percentage of test cases where a given nonlinear model performed better when compared to the linear model, and the right group of bar graphs shows the same percentage when comparing a model against the next-lowest order models. When applied to the `atan` function in MATLAB, we found that the third-order model performed best compared to the linear model and contributed the most in terms of capturing the nonlinear behavior of the test system. We explain this observation by noting that the Taylor series expansion of the arctangent function–a memoryless form of its Volterra series expansion–includes only odd power nonlinear terms. Thus, we show that this

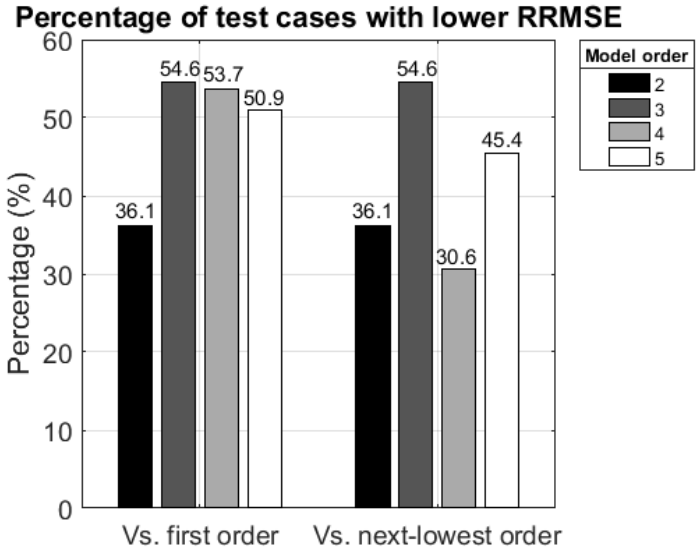


Figure 5.1: Percentage of test cases with lower RRMSE compared against the linear model and next-lowest order model. Model order indicated by color code in legend.

method of comparing Volterra series models of various degrees of nonlinearity helps inform which kernels are most significant, striking a balance between the model's computational efficiency and its fidelity to the original device being modeled.

$$RRMSE = \frac{\sqrt{\sum_{i=0}^{N} (y_i - \hat{y}_i)^2}}{\sqrt{\sum_{i=0}^{N} (y_i)^2}} \tag{5.1}$$

5.0.2 Diagnosing Impulse Response Length

With the optimal Volterra series kernels determined in the last section, the next useful insight is how many samples of the recovered kernel response to include in the model. As discussed in [20], the system modeler must decide how much memory to include in the Volterra series kernel by excluding measured linear kernel values that fall below the noise floor. In this section we elaborate on this process selecting significant kernel values by using

time-delayed mutual information (TDMI).

In the third chapter, we validated this TDMI approach by applying it to systems in MATLAB including simple delay lines, FIR moving average filters, Schroeder all-pass filters, IIR low pass filters, and IIR low pass filters followed by an arctangent function. Using the same type of Gaussian white noise signals from the previous section, we record both the input signals $x[n]$ passed into the test system and the system's response signals $y[n]$. To calculate the estimator bias or noise floor of the TDMI calculations, we create a new variable $x_s[n]$ by randomly shuffling or re-ordering all samples of $x[n]$. We then compute the TDMI between $x[n]$ and $y[n]$ and the estimator bias using the following procedure:

1. Time-shift $y[n]$ by τ samples to create $y[n-\tau]$.

2. From the signals $x[n]$, $x_s[n]$, and $y[n-\tau]$, estimate the probability mass functions $\hat{P}(X)$, $\hat{P}(X_s)$, and $\hat{P}(Y)$ respectively using the MATLAB function `histogram` with $\sqrt{N}$ bins.

3. Use `histogram2` with $\sqrt{N}$ bins to estimate the joint probability mass functions $\hat{P}(X,Y)$ for $x[n]$ and $y[n-\tau]$, and $\hat{P}(X_s,Y)$ for $x_s[n]$ and $y[n-\tau]$.

4. For each these estimated probability mass functions, compute the estimated marginal and joint Shannon entropies $\hat{H}_x$, $\hat{H}_{x_s}$, $\hat{H}_y$, $\hat{H}_{x,y}$, and $\hat{H}_{x_s,y}$ using Equation 3.3.

5. Compute TDMI at the current τ between $x[n]$, $y[n-\tau]$ using $\hat{H}_x + \hat{H}_y - \hat{H}_{x,y}$.

6. Compute estimator bias at the current τ using $\hat{H}_{x_s} + \hat{H}_y - \hat{H}_{x_s,y}$.

To determine the significant values of the recovered kernel response, we simply find the time delays τ where TDMI values exceed the estimator bias noise floor. When applied to the FIR filters in the third chapter, the TDMI method accurately predicted the significant values of the recovered response. We faced some difficulty in accurately predicting the impulse

response length of IIR low pass filters generated using the impulse invariance method applied to single-pole, analog RC low pass filter prototypes. Specifically, we expected the TDMI method to indicate significant values up to $5\times RC$ or five time constants of the low pass filters, but it fell short of this predicted duration for cutoff frequencies $\leq$ 5 kHz. However, when applied to the ProCo Rat 2 guitar pedal with low pass filtering applied with various cutoff frequencies f_c in Chapter 5, we found that the TDMI curve did indicate some sensitivity to f_c. That is, for a cutoff frequency of $f_c = 620$ Hz, the TDMI curve had a lower overall amplitude and longer decay from the maximum TDMI value to the estimator bias noise floor when compared to relatively higher cutoff frequency values of 4.6 kHz and above. Otherwise, the TDMI method seemed to accurately predict the total phase shift of the device when compared to the theoretical phase shift of the built-in RC low pass filter at each cutoff frequency.

In terms of the TDMI method's sensitivity to the presence of nonlinear distortion, the overall shape of the TDMI curves took the form of a pronounced peak near $\tau = 0$ and fast decay into the noise floor, with the peak moving along the positive-τ axis as the Distortion parameter was increased. Thus, we show that the presence of nonlinear distortion does not affect the performance of the TDMI method in predicting overall impulse response length. The key takeaway from these TDMI experiments is that the TDMI method is a viable approach for estimating which recovered kernel values are significant for inclusion in a Volterra series model, allowing the system modeler to make more informed decisions on how much memory to include in a Volterra series model.

5.0.3 Future Work

Regarding future directions that could expand upon the use of Volterra series models to assess a system's degree of nonlinearity, there may be other performance metrics to consider. In this work, we considered the sample mean of RRMSE in the test cases where

a given model performed better than some other model to measure its performance. Other statistical measures such as variance may reveal nuances in model performance that we did not otherwise investigate. Also, we used pseudo-random Gaussian white noice with an array of variances σ^2_{val} to evaluate the performance of our generated nonlinear models. It may be worthwhile to benchmark the performance of the models using a more realistic test signal, such as a recorded passage from a musical instrumental or singing voice. Finally, we only applied this nonlinearity assessment technique to a memoryless nonlinear system, the arctangent function. While we have no reason to believe the application of this technique would drastically vary in the presence of memory, it nonetheless will be interesting to consider benchmarking nonlinear systems with memory in this same manner.

Future directions regarding the use of time-delayed mutual information (TDMI) might include investigating how the technique behaves when applied to systems exhibiting simultaneous nonlinear distortion and filtering. Ultimately, we show that TDMI can be useful for determining how many samples of a recovered impulse response to keep when generating a black box model.

REFERENCES CITED

[1] Roland TB-303 Clones Contest. Which is the best emulator? Test Conclusions.

[2] Sintefex Audio Products Using Our Technology.

[3] The 8 best TB-303 clones according to the artists who use them, March 2020.

[4] A. Novak, L. Simon, F. Kadlec, and P. Lotton. Nonlinear System Identification Using Exponential Swept-Sine Signal. *IEEE Transactions on Instrumentation and Measurement*, 59(8):2220–2229, August 2010.

[5] D. J. Albers and George Hripcsak. Estimation of time-delayed mutual information and bias for irregularly and sparsely sampled time-series. *Chaos, Solitons & Fractals*, 45(6):853–860, June 2012. arXiv: 1110.1615 version: 1.

[6] Pete Celi. Experience a tape echo machine w/Strymon dTape Technology, July 2010.

[7] Thomas M. Cover and Joy A. Thomas. Elements of Information Theory. Wiley Series in Telecommunications., 1991.

[8] Chris Dunn and Malcolm J. Hawksford. Distortion Immunity of MLS-Derived Impulse Response Measurements. *J. Audio Eng. Soc*, 41(5):314–335, 1993.

[9] Felix Eichas and Udo Zölzer. Gray-Box Modeling of Guitar Amplifiers. *Journal of the Audio Engineering Society*, 66(12):1006–1015, December 2018.

[10] Alexander G. Gray and Andrew W. Moore. Rapid Evaluation of Multiple Density Models. In *AISTATS*, 2003.

[11] Ethan Hoerr and Robert C. Maher. Using Volterra Series Modeling Techniques to Classify Black-Box Audio Effects. Audio Engineering Society, October 2019.

[12] Ethan R. Hoerr and Robert C. Maher. Estimating Nonlinear Impulse Response Length using Time-Delayed Mutual Information. Audio Engineering Society, October 2020.

[13] Michael J. Kemp. Analysis and Simulation of Non-Linear Audio Processes Using Finite Impulse Responses Derived at Multiple Impulse Amplitudes. Audio Engineering Society, May 1999.

[14] Michael J. Kemp. Analysis and Simulation of Analogue Dynamic Compressors and Limiters in the Digital Domain. In *Audio Engineering Society Convention 109*, September 2000.

[15] M. Schetzen. Nonlinear system modeling based on the Wiener theory. *Proceedings of the IEEE*, 69(12):1557–1573, December 1981.

[16] Antonin Novak, Frantisek Rund, and Petr Honzik. Impulse Response Measurements using MLS Technique on Nonsynchronous Devices. *J. Audio Eng. Soc*, 64(12):978–987, 2016.

[17] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time Signal Processing*. Prentice-Hall signal processing series. Pearson, 2nd ed. edition, 1998.

[18] Alan V Oppenheim, Alan S Willsky, and Syed Hamid Nawab. *Signals & systems*. Prentice Hall, Upper Saddle River, N.J., 2nd edition, 1997. OCLC: 925123716.

[19] Simone Orcioni. Improving the approximation ability of Volterra series identified with a cross-correlation method. *Nonlinear Dynamics*, 78(4):2861–2869, December 2014.

[20] Simone Orcioni, Alberto Carini, Stefania Cecchi, Alessandro Terenzi, and Francesco Piazza. Identification of Nonlinear Audio Devices Exploiting Multiple-Variance Method and Perfect Sequences. *Journal of the Audio Engineering Society*, page 9, 2018.

[21] John R. Pierce. *An introduction to information theory : symbols, signals & noise*. Dover Publications, New York, 1980.

[22] Massimiliano Pirani, Simone Orcioni, and Claudio Turchetti. Diagonal Kernel Point Estimation of nth-Order Discrete Volterra-Wiener Systems. *EURASIP Journal on Advances in Signal Processing*, 2004(12):145457, September 2004.

[23] Simon Reynolds. *Energy flash : a journey through rave music and dance culture*. Faber and Faber, London, 2013.

[24] Douglas D. Rife and John Vanderkooy. Transfer-Function Measurement Using Maximum-Length Sequences. In *Audio Engineering Society Convention 83*, October 1987.

[25] Martin Schetzen. *The Volterra and Wiener theories of nonlinear systems*. Krieger, Malabar (FL.), 2006.

[26] Thomas Schmitz and Jean-Jacques Embrechts. Objective and Subjective Comparison of Several Machine Learning Techniques Applied for the Real-Time Emulation of the Guitar Amplifier Nonlinear Behavior. In *Audio Engineering Society Convention 146*, March 2019.

[27] Thomas Schreiber. Measuring Information Transfer. *Physical Review Letters*, 85(2):461–464, July 2000.

[28] B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman & Hall, London, 1986.

[29] J.O. Smith. *Physical Audio Signal Processing*. W3K Publishing, 2010.

[30] Roland U.S. What is Analog Circuit Behavior (ACB)? - Roland U.S. Blog, February 2014.

[31] John Vanderkooy. Aspects of MLS Measuring Systems. *J. Audio Eng. Soc*, 42(4):219–231, 1994.

[32] Andreas Wilmer, Marc de Lussanet, and Markus Lappe. Time-Delayed Mutual Information of the Phase as a Measure of Functional Connectivity. *PLoS ONE*, 7(9), September 2012.

[33] Udo. Zolzer. *DAFX : Digital Audio Effects, Second Edition*. John Wiley & Sons, Ltd, Chichester, 2011.