

# School of Physics and Astronomy



## Modelling Non-linear Behaviour in Valve Amplifiers

### MPhys Project Report

Ewan Hemingway  
18th March 2011

#### Abstract

Various black box modelling techniques are employed to investigate the nature of linear filtering and non-linear behaviour guitar valve amplifiers. Generalisations for existing Volterra methods are presented and evaluated in the context of valve distortion, and neural network methods are used to build a working model of a real guitar overdrive pedal. A discussion of the practicalities of implementing these methods is also included.

Signature:

Date:

**Supervisor:** Dr. J. Kemp

# Personal Statement

During the initial few weeks of the project, I spent time researching what types of method were currently being used elsewhere in the field with specific focus on Volterra series representations; an overview these findings can be found in the Literature Review [1]. Having built up a collection of potential black box techniques, I then proceeded to test the suitability of these for non-linear system analysis, specifically in the context of vacuum tube distortion.

Initially, I spent time implementing the exponential sine sweep method in MATLAB, testing first with virtual systems and then later with real system output. I then developed a generalised version of the method described by Farina et al. capable of calculating an arbitrary number of Volterra kernels (Sect. III 2.3).

I then worked on recreating the results of parameter estimation methods as described by Vörös and although a working implementation was produced, the method was shown to be generally unsuitable for use in real world valve modelling (Sect. III 3.2).

After that, I investigated the stability of various neural network models. Once a suitable model was found, I evaluated the effectiveness of various training algorithms and signals, particularly with the aim of improving the low amplitude response of the simulation. A reasonable audio output was eventually obtained and can be heard at Appendix A.4, with the proviso that further filtering may be needed before this can be considered an accurate model (Sect. III 4).

Returning to the Volterra methods, I then implemented a wide-noise version of the Hawkford method, based on original code by Dr. Kemp. Using this I was able to find the Volterra kernels for both a virtual “test system” and for a real guitar pedal. I was also able to place estimates on the amount of memory of the pedal along with a predictions about the ordering of linear and non-linear components in the system. (Sect. III 2.2.1).

Finally I evaluated the computational efficiency of the methods, with particular reference to current hardware.

## Note

A full colour version of this report complete with MATLAB code can found online at [2].

# Contents

<b>I</b>	<b>Introduction</b>	<b>4</b>
<b>II</b>	<b>Background and Theory</b>	<b>6</b>
1	Recap of memory and non-linearities	6
2	Working with Digital Audio	7
3	Analysis of Tube Distortion	8
4	Systems Tested	10
4.1	Technical Considerations . . . . .	11
<b>III</b>	<b>Black Box Methods</b>	<b>12</b>
1	Black Box Methodology	12
2	Volterra Series Methods	12
2.1	Introduction . . . . .	12
2.2	Hawksford Method . . . . .	13
2.2.1	Kemp modification: Multiple Wide Noise Sequences . . . . .	14
2.3	Exponential Sine Sweep Method . . . . .	16
2.4	Comparison of Volterra Methods . . . . .	20
3	Parameter Estimation: Least Squares	22
3.1	Introduction . . . . .	22
3.2	Vörös Technique . . . . .	22
4	Neural Networks	26
4.1	Introduction to Neural Networks . . . . .	26
4.2	Practical Implementation . . . . .	26
4.3	Results . . . . .	27
<b>IV</b>	<b>Evaluation of Practical Implementation</b>	<b>31</b>

<b>V</b>	<b>Conclusions</b>	<b>32</b>
<b>A</b>	<b>Instructions for Code</b>	<b>34</b>
A.1	Wide Hawksford . . . . .	34
A.2	Exponential Sine Sweeps . . . . .	34
A.3	Voros Parameter Estimation . . . . .	34
A.4	Neural . . . . .	34
A.5	Background Noise Analysis . . . . .	35
A.6	Brute Force . . . . .	35

## Part I

# Introduction

The theoretical basis of Digital Signal Processing (DSP) has been well established for many years now, but it is only fairly recently that we have seen consumer level hardware capable of implementing the more advanced techniques. In this project we follow a fairly recent trend of using newly available computational power to simulate a vintage (and what some may describe as obsolete) technology by modelling the valve amplifier. Here we mean obsolete in the sense of function rather than aesthetic: modern digital amplifiers work “perfectly” well in the sense of accurately amplifying a signal.

It is entirely possible that the reader may not have previously come across the valve amplifier at all. The earliest design was conceived by the English physicist John Fleming whilst working for Marconi during the burgeoning telecommunications industry at the turn of the 19th century [3]. Amplifiers were initially required to boost long distance telephone signals, though the applications later spread to “wireless” radios and television. It was not long before commercial guitar amplifiers appeared, and the many of the innovators of that age such as Charlie Gibson and Leo Fender will no doubt be recognised by many guitarists today. Whilst improvements in the technology led to larger wattages and greater stability, the underlying technology remained the same [4]. Valve amplifiers fell out of favour somewhat after the arrival of the much cheaper transistor amplifiers, but has still remained popular with audio enthusiasts and has enjoyed something of a revival in recent years. My personal involvement and interest in valve technology is at least threefold and the following outlines my motivation for the project.

**As a Physicist** I personally find the physics of the vacuum tube fascinating. Even the simplest of vacuum tube amplifiers exhibit some unknown combination of both complex non-linear behaviour and hysteretic effects. Many of the topics covered in this report apply quite generally to non-linear systems: there are gains to be made in areas other than valve modelling or even audio signal processing in general.

**As a Musician** Motivation also comes from my experiences as a musician. Traditionally the main proponents of older vacuum tube technology are guitarists / bass guitarists: this is understandable as a large part of the early electric guitar sound was defined by the tone of valve amplifiers. However my background is based more on recent electronic music, for example in using valve based pedals for adding distortion to synthesiser parts. I find that discrete use of distortion can really harmonically enrich a sound (for some of the physics of this see Sect. II 3). At the other end of the spectrum, I sometimes employ fully blown distortion for a really abrasive and dramatic sound. Hopefully part of this project will provide me with a better physical understanding of tube distortion, and should help apply to my future music making endeavours.

**Commercial** Many of the methods described in this project and in the Literature Review [1] have received much commercial attention. The audio equipment industry is often on the lookout for new technology for modelling vintage equipment, and despite several products offering “true valve modelling”, these are generally not yet fully accepted by the music community; valve modelling is still generally seen as being one of the unsolved problems in digital audio. While it is ambitious to suppose that this project will result in any commercial software, the techniques explored may form the basis for future work in this area.

Ultimately, the aim of the project is to produce a “working” model of valve amplifier based distortion. However the main benefit is actually in the process of arriving at this model (or models) rather than the end result. This report documents my process in evaluating the various techniques used in the field, exploring which aspects work particularly well but also where the weaknesses lie. The process of system identification, particularly in acoustics, can be very much one of experimentation. Prior knowledge of the system can certainly help, but more often than not we are working with pure black box systems (Part III). The structure of the report should hopefully reflect this, documenting the “trial and error” nature of the process.

Initially we will cover some of the background physics of valve behaviour along with a more mathematically rigorous description of the problem. We will also review the physics of some of the more subjective characteristics of the “valve sound” and introduce the pedals / systems that we will be validating the models against. The report then details the methods used throughout the project along with evaluations of their stability and effectiveness. Finally we provide both a mathematical and descriptive analysis of the audio output of the models and discuss how practical an implementation in consumer level hardware is.

## Part II

# Background and Theory

## 1 Recap of memory and non-linearities

Although this report assumes some basic knowledge of acoustics, it is worth quickly establishing some of the key concepts that appear frequently, even if only to avoid any ambiguities. Specifically, there are two fundamental aspects of the “valve problem” as introduced in the Literature Review [1]: memory and non-linearity.

**Non-linearity** From a purely functional point of view, a “perfect amplifier” should be completely linear across its operating range; that is for input signal  $x(t)$ , the corresponding output  $y(t)$  satisfies:

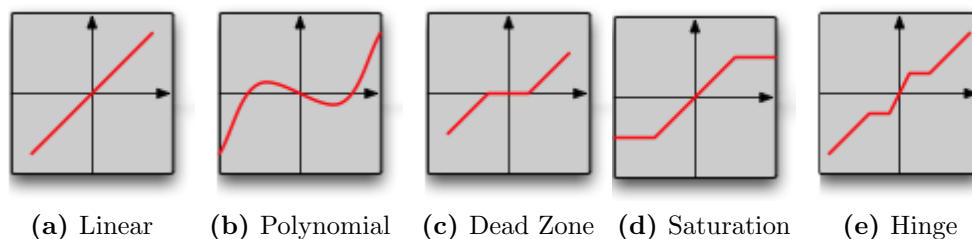
$$y(x_1 + x_2) = y(x_1) + y(x_2)$$

$$y(ax) = ay(x)$$

**Additivity**

**Homogeneity**

Valve amplifiers generally do not behave linearly. It was partly this non-linear behaviour that led to the decline in popularity of the valve, ironically a characteristic that is now very much desired! Fig 1. illustrates some common types of non-linearities. For the most part this project deals with polynomial non-linearities, particularly as the tube sound tends to exhibit soft clipping (see Sect. II.3) rather than the harsher clipping produced by saturation. However there are cases where dead zone type non-linearities can occur, for example due to power supply sag, and while Vörös explores this type of system in [5] it is not considered in detail here. The polynomial form is most sensible as most of the other types of non-linearity can be at least be roughly approximated by a polynomial, albeit a high order one.



**Figure 1:** Schematic of some common type of non-linearities, with the vertical axis representing output  $y(t)$  with  $x(t)$  on the horizontal axis.

**Memory** The other important component of the problem is memory. Put most generally, this is the idea that a system reacts differently depending on what signal has passed previously. In more practical terms, this can be thought of as a kind of linear filter and can be defined as:

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})}x(t)$$

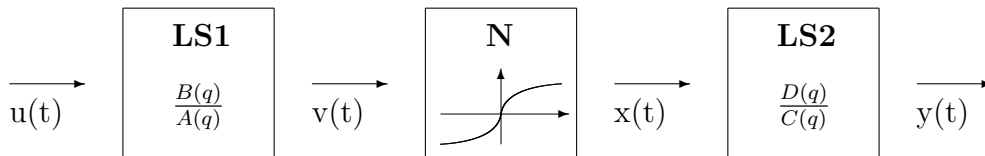
$A$  and  $B$  are polynomials of the delay operator  $q$ :

$$\begin{array}{ll} A(q) = 1 + a_1q^{-1} + \dots + a_{n_a}q^{-n_a} & \text{feedback coefficients} \\ B(q) = b_0 + b_1q^{-1} + \dots + b_{n_b}q^{-n_b} & \text{feedforward coefficients} \end{array}$$

where the delay operator is defined as  $q^{-n}x(t) = x(t-n)$ . The order of the filter is defined as the larger of the two polynomial lengths  $n_a$  and  $n_b$ . The set of filter coefficients  $\{a\}$  and  $\{b\}$  are also usually normalised with respect to  $a_0$ .

The physical origin of this memory is not immediately obvious for the vacuum tube. Even high quality circuit components can have an intrinsic filtering effect on a signal, which may introduce some memory effects. Additionally, there may be some lag after large changes in signal amplitude as the power supply to the tube recovers.

**Composite Systems** Individually these two properties are not difficult to identify: it is the combination that is problematic. A system comprising of linear then non-linear parts is known as a Wiener system; the reverse is called a Hammerstein system. These can be combined to create Wiener-Hammerstein systems (linear  $\rightarrow$  non-linear  $\rightarrow$  linear) such as in Fig. 2 and various other composite systems. The exact combination present in the valve is unknown, the most general statement that can be made is that the system contains at least one linear filter and one non-linear unit. It is also possible that stages contain only very weak non-linearities or filters, and there is a chance that these may safely be ignored without severely affecting the accuracy of the method.



**Figure 2:** Schematic of a Wiener-Hammerstein system: Linear System (LS1), Non-linearity (N), Linear System (LS2). A version of this model is explicitly solved by Vörös (see Sect. III.3.2 for details).

## 2 Working with Digital Audio

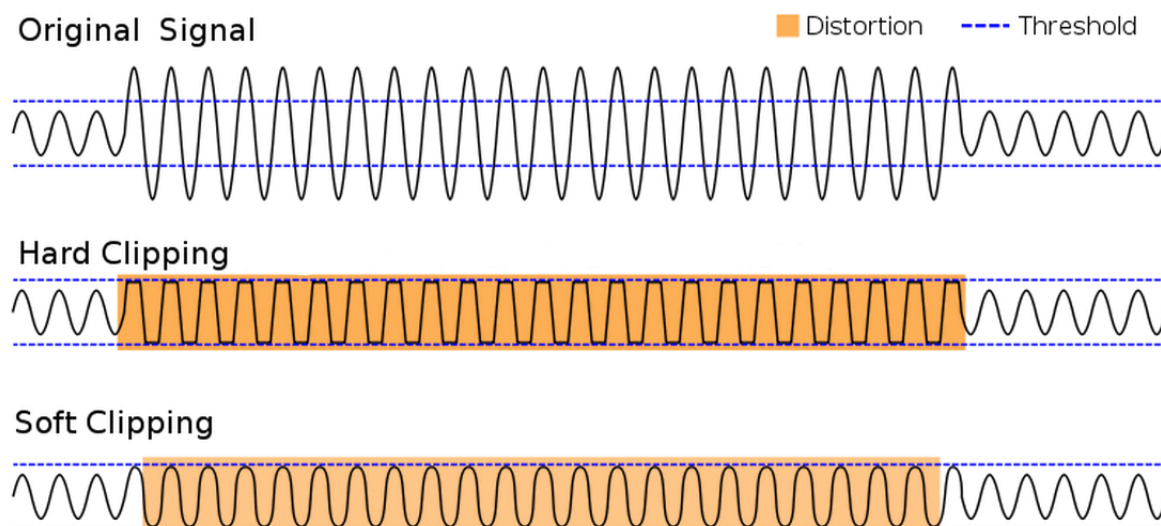
Some of the methods in this project are initially presented in a continuous time form and generally the transition to the discrete version is trivial. However, occasionally issues can arise with representing impulses with digital audio - if the impulse doesn't occur exactly at the point when a sample is recorded the impulse can be blurred out over the intermediate samples. Higher sampling rates can mitigate the effect to a certain extent but this is a fundamental problem with digital audio which can never be fully resolved.

For several of the methods, the use of cyclic deconvolution means that a slight latency in recordings from the sound card\* can safely be ignored. However it was found that the sound card was occasionally overcompensating for latency which *does* produce erroneous results. Hawksford proposes using a preamble sequence of the form  $[0, 0, 1, 1, -1, -1, 0, 0, \dots]$  for manually checking synchronisation [6] and this resolved the issue.

Finally, it is worth pointing out that some care must be taken before writing audio files to disk. Additional distortion can be introduced if normalisation is not carried out correctly, and this can lead to spurious results as we will see next.

### 3 Analysis of Tube Distortion

For low gain input signals, valves act in a fairly linear fashion. However the really interesting physics occurs for high gains, where we start to see the signal distorting and this is key area that we will be looking at during this report. Generally in signal processing, distortion is seen as a unwanted/negative characteristic but for the guitarist this can be a key element of the sound. When we refer to distortion in the context of valve amplifiers, it is specifically “clipping” that is of interest.

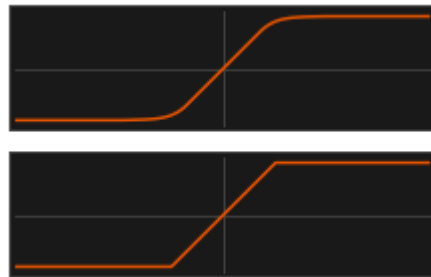


**Figure 3:** Demonstration of the various types of clipping for an overdriven signal. Valve distortion tends to exhibit soft clipping. Diagram modified from [7].

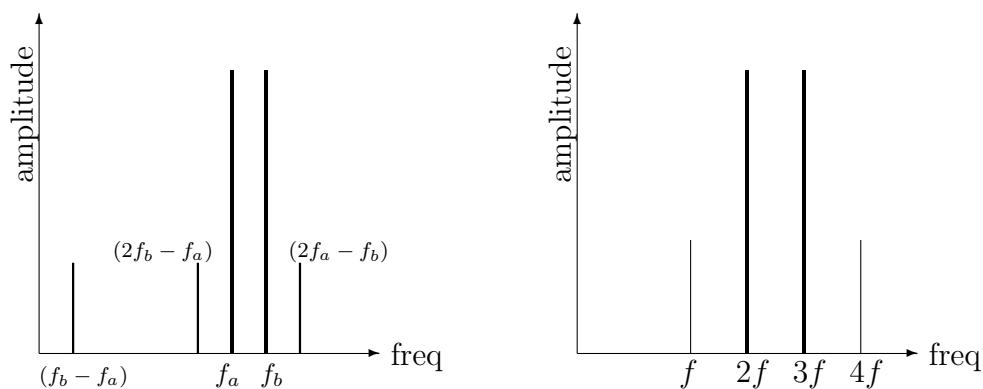
Clipping occurs when an input signal is larger than the system threshold, as shown in Fig. 3. Most modern transistor amplifiers react to this by using the threshold value for any signal outside this range: this is referred to as hard clipping. For an overdriven sine wave, this would tend to produce a square wave. This exclusively features odd integer harmonics up to quite a high order, and can have a rather buzzy tone (without any further equalisation). This is often cited as one of the reasons for why valve distortion is preferred.

\*The EDIROL UA-25EX was used throughout the project

For the valve amplifier, the nature of the input-output relation means the valve acts as a compressor: as an input signal nears the threshold value, the valve scales it back in a *non-linear* fashion meaning some semblance of the original signal is preserved (see **Soft Clipping** in Fig. 3). As the resulting signal is less square-like, the odd harmonics produced are generally of a lower order resulting in a smoother tone [8]. However as the gain is increased on the amplifier, higher order harmonics do start to appear and the fuzzier tone returns. Another observed property of valve distortion is that it can be asymmetric, that is to say that negative values from the input signal may be treated differently from positive values. This has the effect of introducing even harmonics, and this is part of what gives the valve amp its distinct sound.



**Figure 4:** Example symmetric non-linearities from a waveshaper plugin for Ableton Live (input level on x-axis, output on y-axis). The top profile produces a softer clipping whereas the bottom acts as a limiter producing hard clipping.



(a) In general, intermodulation distortion does not produce harmonic difference tones.

(b) If input frequencies are related by an integer ratio then difference tones are harmonically related.

**Figure 5:** Intermodulation effects: thicker lines are the frequencies initially fed into the system, thinner lines show some of the new distortion products. Figure 5b shows the special case when input frequencies are related by integer ratios.

An interesting side effect of increased harmonic content in non-linear systems is the appearance of “difference tones” or intermodulation effects. When multiple frequencies are present in a distorting system, new frequencies can be produced. For example if we feed two input frequencies  $f_a$  and  $f_b$  into a non-linear system, linear integer combinations of these may be produced as shown in Fig. 5a. These new difference tones are not necessarily harmonically related to the original signal which can often sound discordant: this can be avoided by making sure input frequencies are related by an integer ratio. For example a perfect fifth (with ratio  $2f:3f$ ) would produce difference tones at  $f$  and  $4f$  (see Fig. 5b). This phenomena is the main reason why power chords are often used in rock and pop music that involves distortion.

## 4 Systems Tested

While a lot of the early work in validating the various models could be performed using virtual models in MATLAB, the interesting experiments are those performed on real hardware. Two pieces of vacuum tube driven equipment were used throughout the project:

**Korg Electribe ESX1** The Electribe is one of KORG's recent "groovebox" synthesizers. The machine is basically a glorified sampler, but is laid out in a similar fashion to many drum machines. As with most samplers, the internal workings are completely digital with the sampling rate set at a standard 44.1kHz. However the Electribe has the addition of a tube stage in what KORG call "valve force circuitry". The tube design itself consists of a pair of 12AX7s, a commonly used valve that is often found in guitar amplifiers, and dates back to 1947. The exact integration of this with the rest of the synth is unclear as schematics are unavailable, however from experimentation it seems that the tubes act as an auxiliary send which is mixed back into the master channel, meaning that some component of the signal is always bypassing the tube stage. The Electribe was used extensively for testing during the initial months of the project but was eventually abandoned for the TD-1 following a series of poor results which can probably be attributed to loss of information during analogue-to-digital (ADC) stages or corruption with the clean signal from the master channel.



Figure 6: The KORG Electribe ESX1

**Guyatone TD-1** The TD-1 is a classic guitar distortion pedal from the 1980s. Its simplicity makes it ideal for this project: not only does it only use a single vacuum tube, the signal path is entirely analogue which avoids any artefacts that may occur during the ADC processes. It has only two controls: *gain*, which specifies how much the tube stage is driven and *level*, which controls output volume. An issue the pedal shares with other vintage equipment is that there is a noticeable 50Hz mains hum, although this can be filtered out without too much effort (see Sect. II.4.1 below). As the pedal is reasonably old, it was not possible to open it up to examine the internal workings, but online guitar communities have suggested that the pedal also uses a 12AX7 tube [9, 10]. The pedal was kindly loaned by Dr Jonathan Kemp.

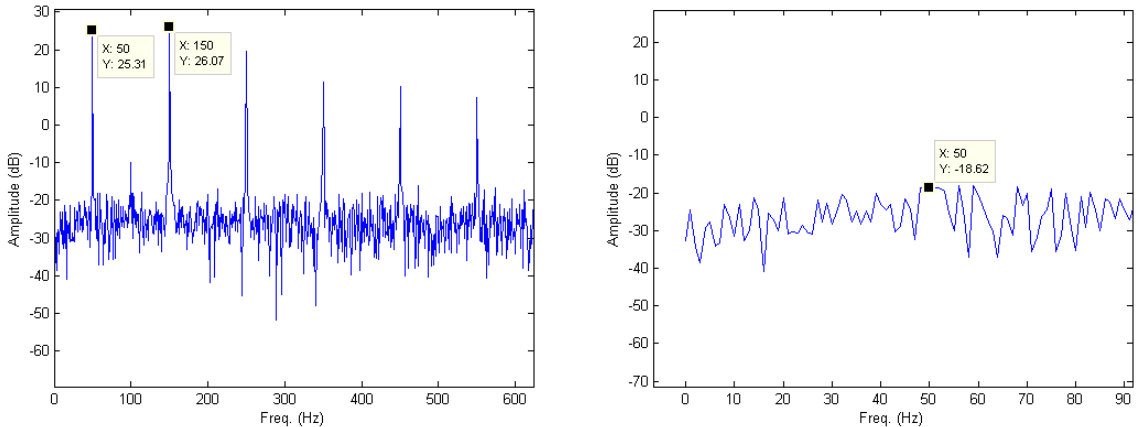


Figure 7: The Guyatone TD-1 guitar pedal

## 4.1 Technical Considerations

One of the problems with experimenting with real hardware is the interference caused by “mains hum”. Many effects pedals and other audio hardware use an AC (alternating current) mains power supply to function, and unfortunately this can create unwanted interference. Fortunately, this is highly localised to a frequency of 50Hz and can be filtered out to a certain extent. For example, Fig. 8a shows the frequency spectrum for a background noise recording taken from the pedal. Not only is the  $f = 50\text{Hz}$  peak clearly seen; odd harmonics at  $3f, 5f, \dots$  can also be observed, implying that the source of the interference occurs before some non-linearity. However, it is established later in the project that tube distortion produces both odd and even harmonics so it’s possible that this effect is due to some other non-linearity in the power supply itself. This theory is supported by the presence of these odd higher order harmonics even with the gain at the lowest setting.

We can manually remove these peaks in the frequency domain by flattening off the peak in the region of 50Hz to the level of the neighbouring values (see Fig. 8b), although we must remember to preserve conjugate symmetry [11]. This could in theory be repeated for all higher harmonics, although the main contribution is from the fundamental frequency at 50Hz (and maybe the first odd harmonic). It is possible to programmatically remove these peaks (see Appendix A.5) although for this to work the 50Hz peak must be exactly situated on a frequency bin. i.e. the sample length must be a multiple of  $\frac{1}{50}^{\text{th}}$  of a second.



(a) Wide view illustrating the appearance of odd harmonics

(b) Close up demonstrating removal of the 50Hz peak by using neighbouring values

**Figure 8:** Plot of frequency spectra for the background noise of the Guyatone TD-1, recorded at a gain setting of 7.

## Part III

# Black Box Methods

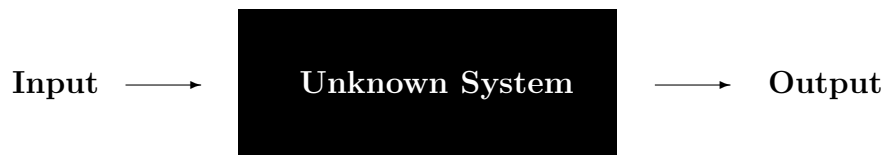


Figure 9: The black box system

## 1 Black Box Methodology

All the techniques presented in this project fall under the class of black box methods, that is they only require input and output signals to model a system. While this does mean that no knowledge of the internal workings is explicitly *required*, it's not to say it can't be beneficial. Many system identification techniques are known to be stable for a specific type of system, e.g. Wiener-Hammerstein. However with valve modelling there is currently no consensus on the exact ordering of non-linearities and linear filters. By exploring the stability of some of these black box methods, we hope to place constraints on the this structure.

Valve amplifiers can have any number of tube stages which complicates matters further. Say, for example, that a single valve could be described as a Wiener-Hammerstein system. An amp with three tube stages would become a W-H-W-H-W-H system, and methods that could cope with solving a single stage may fail for this composite system. This could be part of the reason why early experiments with multi-tube systems such as the KORG Electribe failed to produce many positive results.

## 2 Volterra Series Methods

### 2.1 Introduction

Although the key theories of the Volterra series have already been covered in the literature review [1], it is worth quickly recapping the basics here. The somewhat daunting Volterra series representation in its full form (Eq. 1) is computationally impractical as the  $n^{\text{th}}$  order Volterra series requires an  $n$ -dimensional integral, i.e. memory requirements and processing time grow exponentially.

$$y(t) = k_0(t) + y_1(t) + y_2(t) + \dots \quad (1)$$
$$y_n(t) = \int \dots \int k_n(\tau_1, \tau_2, \dots, \tau_n) x(t - \tau_1) x(t - \tau_2) \dots x(t - \tau_n) d\tau_1 d\tau_2 \dots d\tau_n \text{ for } n \geq 1$$

However by only considering terms where  $\tau_n$  equals some  $\tau$  for all  $n$ , the expression reduces to the much simpler form (this process is referred to as taking diagonal Volterra coefficients):

$$y(t) = k_0 + k_1(t) \otimes x(t) + k_2(t) \otimes x(t)^2 + \dots + k_M(t) \otimes x(t)^M \quad (2)$$

The Volterra series already loses a degree of accuracy by truncating after only a few terms: this simplification to only include “diagonal terms” affects the accuracy further. However the method has historically produced positive results [12, 13] and successes with non-linear system identification make it a very sensible path to investigate.

To properly represent a system, a technique must test all possible frequencies, at least up to the Nyquist limit. Both the following methods try to ascertain the set of Volterra coefficients  $\{k\}$  in Eq. 2. They vary however in their choice of signal: the Hawksford method tests the system with a series of noise sources whereas the ESS method opts instead for sinusoidal signals of varying frequency.

## 2.2 Hawksford Method

In his 2005 paper [6], Hawksford describes a novel way of finding these Volterra coefficients  $\{k\}$ . While the concept is reviewed in detail in the literature review [1], the basic idea is that by using  $M$  independent white noise signals, we can end up with a set of  $M$  simultaneous equations relating input and output by Volterra series representation, and we can solve this by matrix inversion for coefficients,  $\{k\}$ .

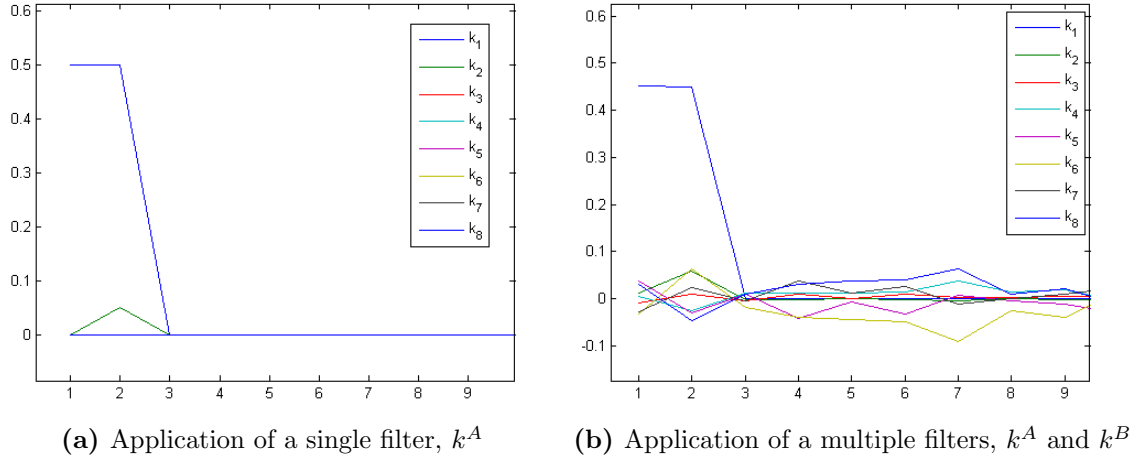
We begin testing with a simple filter and non-linearity. However, rather than considering this as a traditional Wiener system, it perhaps makes much more sense to think of this as a composite “Volterra filter” which contains both elements simultaneously. This is best demonstrated with an example filter  $A$  (ignoring  $k_0$  for now):

$$\begin{aligned} k_1^A &= [ 0.5, 0.5 ] \\ k_2^A &= [ 0, 0.05 ] \end{aligned} \quad (3)$$

The linear impulse response  $k_1$  in this case just a basic low pass filter. The impulse response to the signal squared,  $k_2$ , simply delays by 1 sample and scales by 0.05. When implemented in MATLAB, the Hawksford method copes well and can recover these responses without any difficulty. In fact the result is exactly correct, to floating point accuracy at least (see Fig. 10a).

But, as Kemp points out in a recent paper [14], this doesn’t hold if this system is followed by a second filter,  $B$ . For example, we can use:

$$\begin{aligned} k_1^B &= [ 0.9 ] \\ k_2^B &= [ 0.05 ] \end{aligned} \quad (4)$$

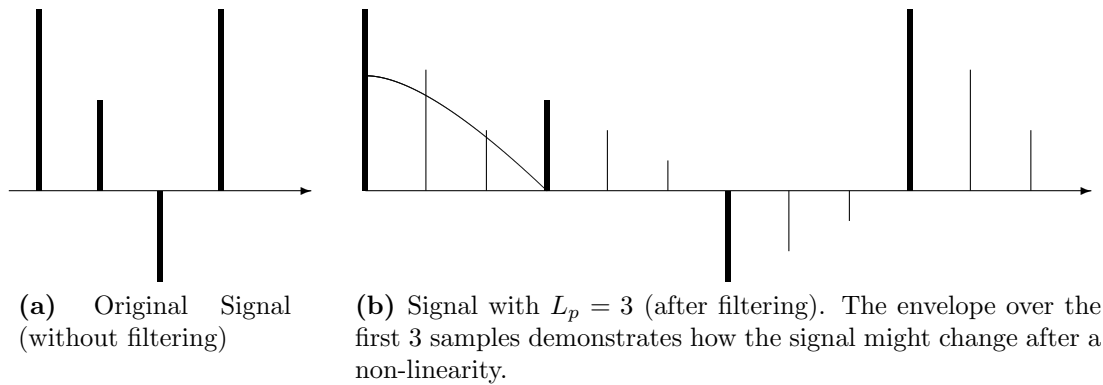


**Figure 10:** Plot of multiple Volterra kernels  $k$  against time. Fig. 10b clearly exhibits an unacceptable level of noise implying instability of the method for WH systems.

The results for composite system are shown in Fig. 10b. We can clearly see that the level of noise in the kernels is significantly higher, given that the noise for a single filter (Fig. 10a) is of order  $10^{-13}$ . Unfortunately this makes it impossible to distinguish the second order kernel  $k_2$ . This means that we can rule out the Hawksford method, at least in its current form, for Wiener-Hammerstein type systems altogether.

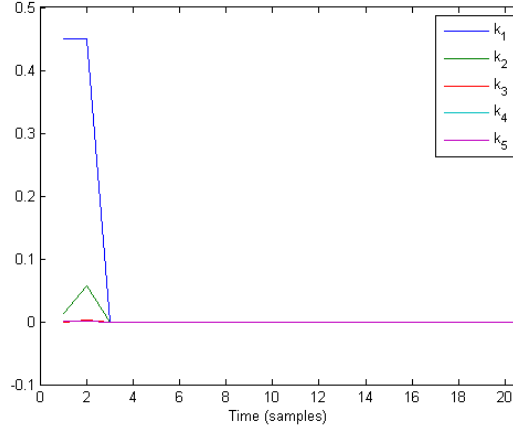
### 2.2.1 Kemp modification: Multiple Wide Noise Sequences

To understand why Hawksford's method fails, we need to think about the effect of a having a linear filter before the non-linearity. Essentially this means that it is not only the current sample that is being fed into the non-linearity, it's also some unknown combination of previous samples. After passing through the non-linearity, it becomes impossible to distinguish the different components of the signal. Kemp argues that by interleaving the input signal with zeros, each point in the sample (and any effects of filtering on that sample) can pass through the non-linearity individually [14]. It is worth noting that the frequency spectrum of a white noise signal remains white after interleaving for any  $L_p$ , and this is important as we wish to test the entire frequency response equally.



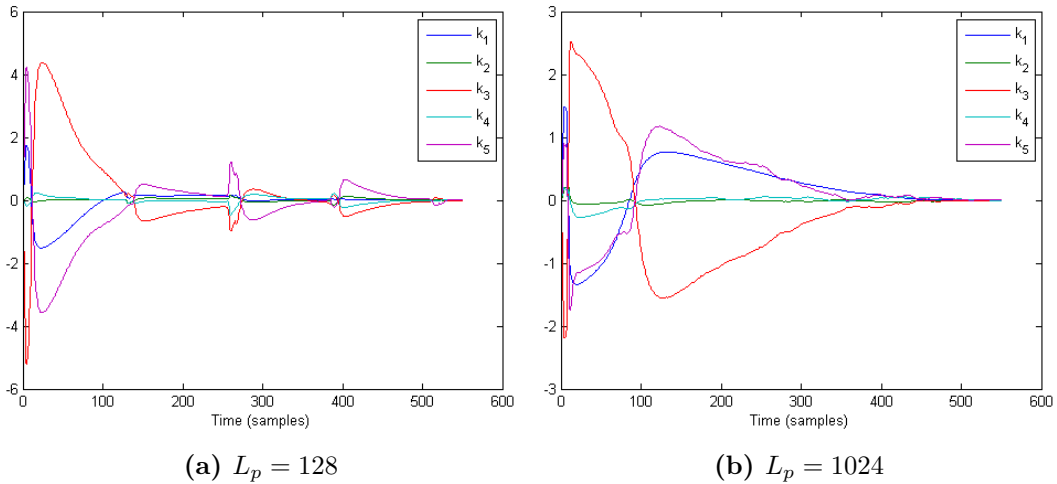
**Figure 11:** Demonstration of how zero-padding can separate out the effect of filtering on each sample.  $L_p$  is defined as the period of the original samples after interleaving.

The concept of interleaving is perhaps best demonstrated with a diagram. In Fig. 11, a simple filter  $[1, 2/3, 1/3]$  is applied to a noise signal. As the length of the filter matches the amount of zero interleaving ( $n_b = L_p = 3$ ) the effects of the filter can be completely isolated. After interleaving we have effectively created composite system comprising of the first filter and the non-linearity, which is then passed through the second filter. The problem is now reduced to the stage for which the original Hawksford method has been shown to be stable for.



**Figure 12:** The zero-padded Hawksford method can be successfully be applied to the multi-filter system.

To test this theory, we again use the two stage system consisting of filters  $k^A$  and  $k^B$  (Eqs. 3 and 4). Fig. 12 shows that with the zero interleaving the method now appears to be stable, with the non-zero kernels clearly distinguishable from any noise. This is particularly obvious after comparison with the vanilla Hawksford method (Fig. 10b) for the same system.



**Figure 13:** Results for the zero-padded Hawksford method for the TD-1.  $L_p$  is the length of the interleaving.

The next logical step then is to test a real system using this method. The results for the TD-1 at a gain setting of 7 are shown in Fig. 13. The system was tested for various amounts of interleaving  $L_p$  and it was found that a significant amount of padding was

required; even a value of  $L_p = 128$  (Fig. 13a) isn't large enough to capture the full set of kernels properly. We can see that the responses are interrupted at every 128 samples with a "blip". Increasing the value of  $L_p$  to 1024 (Fig. 13b), the kernels can be properly observed without interruption.

While the zero-padding method doesn't explicitly provide information about the two filters, it does provide us with a strong estimate for the amount of memory in the initial linear filter, i.e. the system will have a reaction to a signal that can last for up to approximately 1000 samples (or 23 milliseconds).

However noise sources are not the only choice of signal when it comes to Volterra methods, and we will now see how exponential sine sweeps can be used.

## 2.3 Exponential Sine Sweep Method

There have been various reports of successful non-linear system identification using sine sweep methods; in particular Farina et al. advocate the technique for use with Hammerstein systems [13, 15]. By using an exponentially swept sine (ESS), the higher order harmonics caused by the distortion can be precisely localised within the impulse response and separated out (for details see [1] or [13, 15]).

However Farina only provides a solution for the specific case of calculating the first 5 kernels. Here we present a generalisation of the method for finding a Volterra representation of arbitrary order,  $M$ . The key problem is relating the recorded impulse responses (Eq. 5a) to the Volterra kernels (Eq. 5b), and this is simplified significantly by the fact that the signal consists of only a single sine wave of varying frequency,  $\omega_{var}$ :

$$y(t) = h_1 \otimes \sin(\omega_{var}) + h_2 \otimes \sin(2\omega_{var}) + \dots + h_M \otimes \sin(M\omega_{var}) \quad (5a)$$

$$y(t) = k_1 \otimes \sin(\omega_{var}) + k_2 \otimes \sin^2(\omega_{var}) + \dots + k_M \otimes \sin^M(\omega_{var}) \quad (5b)$$

To relate these expressions, we must note that the  $\sin^n(\omega_{var})$  terms can be expanded in terms of sines and cosines. Again, this is best demonstrated with an example, for instance to find  $M = 4$  we expand up to  $\sin^4(x)$ :

$$\begin{aligned} \sin(x) &= \sin(x) \\ \sin^2(x) &= \frac{1}{2}(1 - \cos(2x)) \\ \sin^3(x) &= \frac{1}{4}(3\sin(x) - \sin(3x)) \\ \sin^4(x) &= \frac{1}{8}(3 - 4\cos(2x) + \cos(4x)) \end{aligned}$$

By collecting terms of similar sines and cosines and associating  $K_n$  with  $\sin^n(x)$ , we can rearrange as the following linear superposition where  $H_n = \mathcal{F}(h_n)$ ,  $K_n = \mathcal{F}(k_n)$ <sup>†</sup>:

---

<sup>†</sup>Here  $\mathcal{F}$  is the Fourier Transform.

$$\begin{aligned}
H_1 &= K_1 + \frac{3}{4}K_3 \\
H_2 &= -i\frac{1}{2}K_2 + -i\frac{1}{2}K_4 \\
H_3 &= \frac{-1}{4}K_3 \\
H_4 &= i\frac{1}{8}K_4
\end{aligned}$$

Notice that the cosine terms have been multiplied by  $i$  in the frequency domain: as this corresponds to a phase shift of  $\pi/2$ , this has the effect of transforming them into the sine terms we require. Finally, we can solve this set of simultaneous equations for Volterra kernels  $K$ , remembering that  $i^{-1} = -i$ :

$$\begin{aligned}
K_1 &= H_1 + 3H_3 \\
K_2 &= 2iH_2 + 8iH_4 \\
K_3 &= -4H_3 \\
K_4 &= -8iH_4
\end{aligned}$$

So how can we solve this for the general case? In general we can express the expansions for powers of sines as [16]:

$$\sin^n \theta = \begin{cases} \frac{2}{2^n} \sum_{k=0}^{\frac{n-1}{2}} (-1)^{\binom{n-1}{2}-k} \binom{n}{k} \sin((n-2k)\theta) & \text{for } n \text{ odd} \\ \frac{1}{2^n} \binom{n}{\frac{n}{2}} + \frac{2}{2^n} \sum_{k=0}^{\frac{n}{2}-1} (-1)^{\binom{n}{2}-k} \binom{n}{k} \cos((n-2k)\theta) & \text{for } n \text{ even} \end{cases} \quad (6)$$

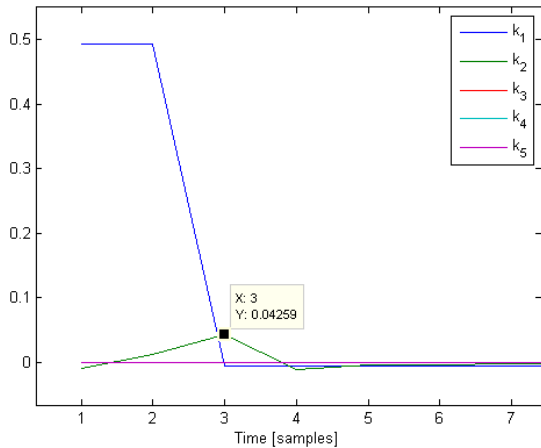
Similarly, we can collect expansions for which the same sine/cos terms appear and using the coefficients found using Eq. 6, we can form a set of simultaneous equations in matrix  $\mathbf{Q}$  such that:

$$\begin{pmatrix} H_1 \\ H_2 \\ H_3 \\ H_4 \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 & 0 & \frac{3}{4} & 0 & \dots \\ 0 & \frac{-1}{2}i & 0 & \frac{-1}{2}i & \dots \\ 0 & 0 & \frac{-1}{4} & 0 & \dots \\ 0 & 0 & 0 & \frac{1}{8}i & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} K_1 \\ K_2 \\ K_3 \\ K_4 \\ \vdots \end{pmatrix} = \mathbf{Q} \begin{pmatrix} K_1 \\ K_2 \\ K_3 \\ K_4 \\ \vdots \end{pmatrix}$$

Given that matrix  $\mathbf{Q}$  is non-singular [17], we can solve for  $K$  simply by:

$$K = \mathbf{Q}^{-1}H$$

Testing correctly recovers the results as presented by Farina et al. for the  $M = 5$  case; furthermore the code matches results for  $M = 7$  as worked out by hand. Details of the practical implementation can be found in Appendix A.2.



**Figure 14:** Results for virtual system  $k^A$  using an exponentially swept sine signal from 20-22050Hz for  $M = 5$ .

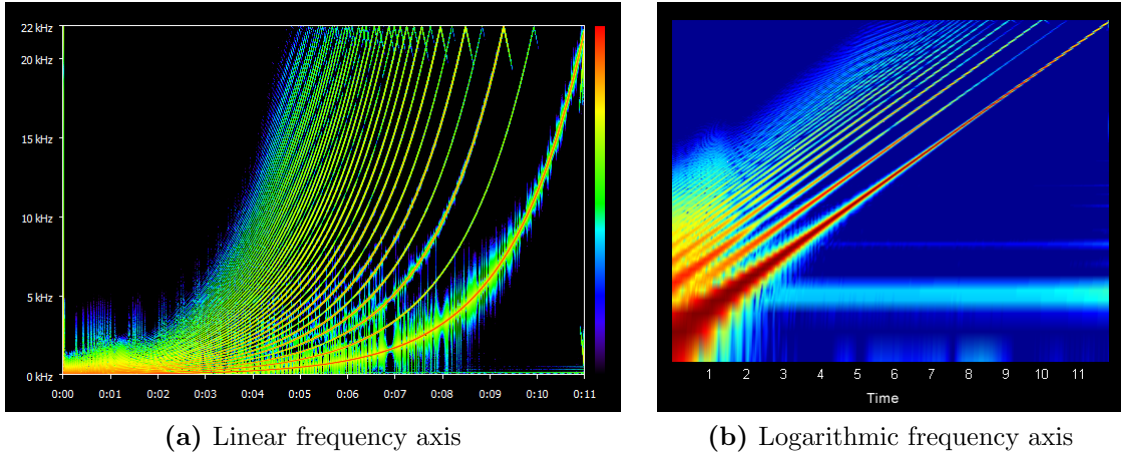
We can see that the lowpass filter  $k_1^A$  is correctly represented to a reasonable degree of accuracy, although as this is just the ordinary linear response this should be expected. The method gets the magnitude of the second order kernel  $k_2^A$  correct, although its position is lagging by one sample; it also doesn't immediately return to zero as expected. This lag may arise from inaccuracies in the way in which the impulses are separated out - rounding errors may cause the portion of the impulse extracted to miss by one sample. It is also worth pointing out that the solution is not exact, e.g. the lowpass filter is returned as  $[0.49 \ 0.49]$  rather than the expected  $[0.5 \ 0.5]$ .

The Hawksford result for the same system (Fig. 10a) returns the exact filter values to floating point precision: it is clear that the sine sweep method is failing to reproduce this level of accuracy. This may be due to the inherent unsuitability of sine sweep methods for systems with memory before a non-linearity, and to understand this we should think about what practical effect filtering may have. For example, a simple lowpass filter  $[0.5, 0.5]$  effectively creates a second copy of the original sine sweep signal delayed by a sample; more complex filtering could create more copies with larger delays. However when multiple frequencies are present in a non-linear system, intermodulation distortion can become a problem (see **Analysis of Tube Distortion** on page 8). Because the delay between copies of the original is so small, difference tones can appear at *any* of the adjacent frequency bands. This can lead to a certain amount of *smoothing out* of both linear and higher order responses which is clearly undesirable.

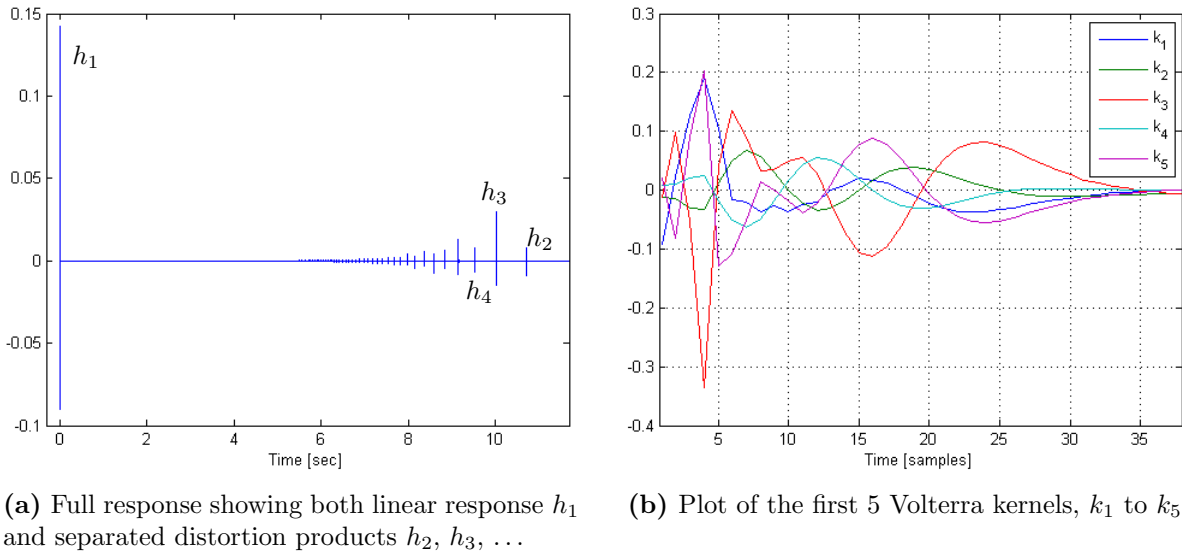
It is unclear whether there is significant filtering occurring before the non-linearity in valve distortion, although it is certainly plausible that at least some part of the circuitry introduces some memory effects. To examine this we now look at how well the sine sweep method works for tube distortion using the TD-1 pedal.

Figure 15 demonstrates some of the beauty that arises from the additional harmonics produced by the pedal. The lowest curve in Fig. 15a corresponds to the original signal, and as expected this appears most predominately; the lighter curves are the higher order harmonics. Closer inspection reveals that both odd and even harmonics are present, implying that the non-linearity is asymmetric. Note that the lines appear curved as the frequency axis is linear: on a logarithmic frequency scale the higher order harmonics

The method outlined by Farina, including the above generalisation, was successfully implemented in MATLAB, and testing was initially performed using a virtual system of the same parameters as the filter  $k^A$  in Sect. III 2.2. Initially the sweep was carried out between 20Hz to 20kHz and these results suffered from significant interference at the Nyquist frequency. This was due to the fact that the method was recording harmonic content from outside the sweep range but within the capabilities of the sampling rate,  $F_s$ . This issue was avoided by simply sweeping up to the Nyquist frequency,  $F_s/2 = 22050\text{Hz}$ , the results of which can be seen in Fig. 14.



**Figure 15:** Spectrogram plots of frequency against time for the response of the Guyatone TD-1 pedal to a swept sine signal.



**Figure 16:** Results for the TD-1 pedal, as found by exponential sine sweep methods.

appear as lines parallel to the original signal (Fig. 15b).

Fig. 16a shows the full impulse response for the pedal output. As hoped, we can see that the linear response (far left) and the higher order responses (increasing order as you move from the RHS) have been successfully separated out. We should also note that the odd order harmonic responses are generally larger in magnitude than their even counterparts.

Once isolated these responses are used to calculate the Volterra kernels  $k_n$ , and these are plotted in Fig. 16b. Encouragingly as with the Hawksford results, the set of kernels appear to be essentially of the same shape, albeit with varying scale factors. The linear term  $k_1$  is the exception to this and appears to have its own distinct shape.

As with the Hawksford results, the odd order  $3^{rd}$  and  $5^{th}$  responses dominate, even compared to the first order kernel (which represents the original component of the signal). However this isn't necessarily an issue as these have inverse shapes: the responses are harmonically related so their effects may cancel out to a certain extent.

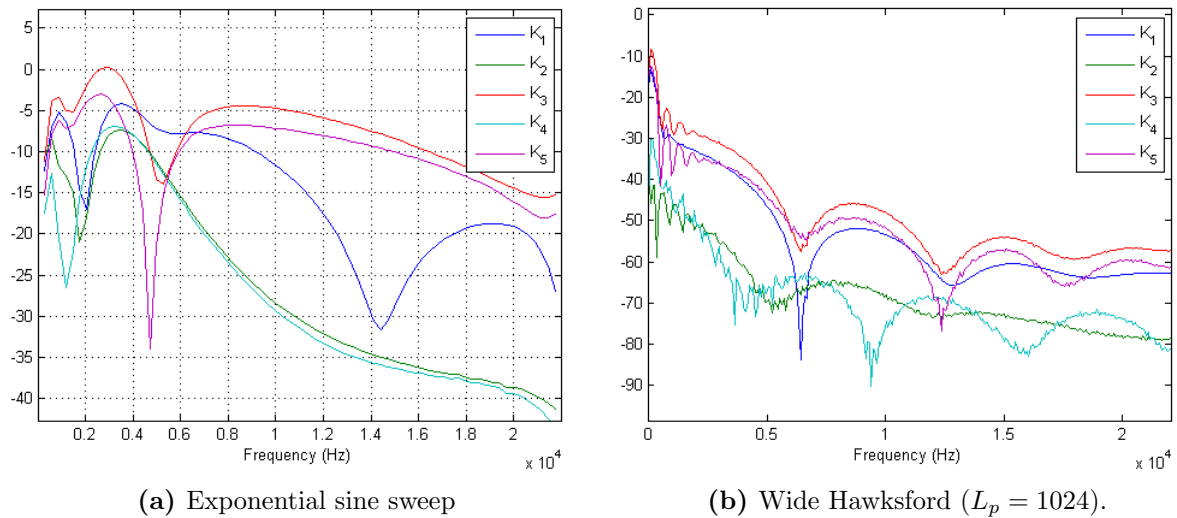
## 2.4 Comparison of Volterra Methods

So how else do the ESS results compare with those obtained by the Wide Hawksford method? Interestingly the characteristic length of the responses is significantly shorter than the Hawksford results by a factor of around 10. As a result, the peaks of the Hawksford kernels are much smoother whereas in the sine sweep method these are only a couple of samples wide.

Interestingly for both methods there appears to be a pattern of similar shapes for kernels of the same parity. For example, odd numbered responses  $k_3$  and  $k_5$  have very similar shapes with peaks and zero crossings at generally at the same positions. Furthermore, the polarity<sup>‡</sup> appears to alternate as:

$4n$	$(k_4, k_8, \dots)$	<b>Shape A</b>
$4n - 2$	$(k_2, k_6, \dots)$	<b>Shape A (inverted)</b>
$4n - 1$	$(k_3, k_7, \dots)$	<b>Shape B</b>
$4n - 3$	$(k_1, k_5, \dots)$	<b>Shape B (inverted)</b>

It is also plausible that there is in fact only one shape, but that either lag or extreme scaling creates the appearance of two.



**Figure 17:** Frequency responses (in dB) for Volterra series methods.

Further insight into the two methods may be gained by looking at the frequency responses of the kernels (Fig. 17). However the lack of agreement between the two plots makes it difficult to make any general statement about how ideal these Volterra representations are. The Hawksford kernels show a strong lowpass filtering effect (Fig. 17b) and this can be heard in the output (see Appendix A.1). We can also observe notches at multiples of 3000Hz, although the origin of this is unclear.

An inherent weakness of the sine sweep method is that it simply doesn't test the system with any impulsive signals (as in the Wide Hawksford method) and as a result, signals

<sup>‡</sup>Here we mean polarity in the sense of inversion of the signal about the x-axis.

with the short attacks may well be poorly represented. However for sustained periods of a signal, the method copes well and the explicit excitation of the full frequency range means that a solid frequency response is produced.

Both audio output and analysis of the results suggest that the Volterra models as presented here do not represent the system accurately. However, they do provide us with insight into the mechanism of tube distortion. Both models reveal that both odd and even harmonics are present, implying that the non-linearity is asymmetric: there is also agreement that the odd order harmonics are more dominant.

## 3 Parameter Estimation: Least Squares

### 3.1 Introduction

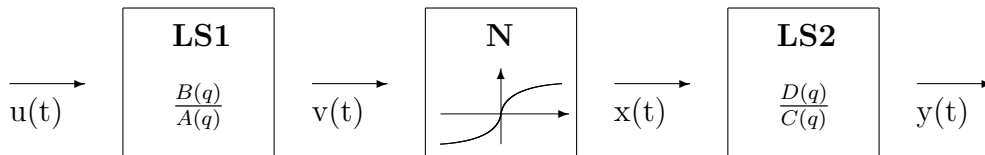
Departing altogether from Volterra series deconstruction, we now explore a new class of system identification technique. Parameter estimation methods take a different approach. The idea is that we can parametrise the system in terms of sets of filter coefficients for the linear sections; similarly the coefficients of polynomial descriptions of non-linearities are included. In the most general case, parameter estimation methods aim to find the set of values for these parameters that minimise the error in the model.

To appreciate the complexity of these algorithms, it is worth considering a naïve brute force implementation. First let's consider a simple system with a linear filter (at a low order of 2) and a quadratic non-linearity. Assuming that the parameters are bounded between -1 and 1, we can attempt to brute force a solution by iterating through values between these limits and checking the error. Even this simple system contains 7 parameters already, so the search is large and takes around 10 minutes on a modern machine (see Appendix A.6 for details). The main problem is that the size of the problem grows exponentially with the number of parameters, and clearly a more subtle solution is required.

Looking at it in a more abstract sense, we can think of the system being represented as a point in N dimensional parameter space (where N is the number of parameters). Brute force systematically checks each point which is an incredibly inefficient process. Instead the following method hones in on the solution using a gradient descent method, analogous to finding the low point of a valley where height is given by the error.

### 3.2 Vörös Technique

In a 2007 paper [18], Vörös outlines a method for identifying Wiener-Hammerstein systems, that is systems with a filter, non-linearity, and finally a second filter. This seems a sensible model to try to fit tube behaviour to as it accounts for both memory and non-linearity. Figure 18 illustrates a typical Wiener-Hammerstein system. It is worth noting that we have designated “internal variables”  $v(t)$  and  $x(t)$  to represent the signal in the intermediary stages of the system.



**Figure 18:** Schematic of a Wiener-Hammerstein system: Linear System (LS1), Non-linearity (N), Linear System (LS2).

The model begins by constructing a vector containing the full set of parameters needed to describe the system. For a Wiener Hammerstein system, this vector needs to include

parameters for two sets of filters, each with both feedforward ( $\{a\}$  and  $\{c\}$ ) and feedback ( $\{b\}$  and  $\{d\}$ ) coefficients. The vector must also include polynomial coefficients  $\{f\}$  for describing the non-linearity. Thus the parameter vector is:

$$\theta = [b_0, \dots, b_{n_b}, a_1, \dots, a_{n_a}, f_2, \dots, f_r, d_1, \dots, d_{n_d}, c_1, \dots, c_{n_c}]^T$$

where  $n_b$ ,  $n_a$ ,  $n_d$ , and  $n_c$  are the lengths of each part of the filters and  $r$  is the order of the polynomial describing the non-linearity. We can similarly define a matching “data vector” as follows:

$$\Phi(t, \theta) = [u(t), \dots, u(t - n_b), -v(t - 1), \dots, -v(t - n_a), v^2(t), \dots, v^r(t), \\ x(t - 1), \dots, x(t - n_d), -y(t), \dots, y(t - n_c)]^T$$

If the parameter vector is exactly known then for any point in time, we can find the output  $y$  by:

$$y(t) = \Phi^T(t, \theta)\theta \quad (7)$$

However  $\theta$  is generally not known, but can be numerically found by iterative methods. By defining the error  $e(t)$  as:

$$e(t) = y(t) - \Phi(t, \theta^s)\theta^{s+1}$$

where  $\theta^s$  is the parameter vector at iteration  $s$ , the problem now becomes one of error minimisation: what values of  $\theta$  can we choose such as to minimise error  $e(t)$ . Vörös does not explicitly mention how this is achieved in the paper, but he implies that a *recursive Least Squares* method is used. However in a similar 2010 paper [5], he does describe the method in more detail.

The key point of this technique is that Eq. 7 is *linear* in parameters [18]. This means that the problem can be solved by a method loosely related to linear regression analysis - basically it becomes similar to a curve fitting problem. The system is overdetermined meaning that there are many more data points (or samples) than there are parameters to the fit. In solving, the method works through the sample recursively, improving its “best fit” of the parameter vector using the set of nearby samples. A full derivation of the method is beyond the scope of this report, but can be found in “System Identification: Theory for the User” by Ljung [19], where the method appears to have originated.

Despite the several inconsistencies throughout the paper, it was possible to produce a working implementation of the method in MATLAB. For testing, we stick to the Wiener-Hammerstein model proposed in [18] with the initial input  $u$  consisting of Gaussian white noise:

$$\begin{aligned}
v(t) &= u(t-1) + 0.5u(t-2) + 0.2v(t-1) - 0.35(t-2) \\
x(t) &= v(t) + 1.3v^2(t) - 1.1v^3(t) \\
y(t) &= x(t-1) + 0.25x(t-2) - 0.2y(t-1) + 0.3y(t-2) \\
\theta_{exact} &= [1.0, 0.5, -0.2, 0.35, 1.3, -1.1, 0.25, 0.2, -0.3]^T
\end{aligned}$$

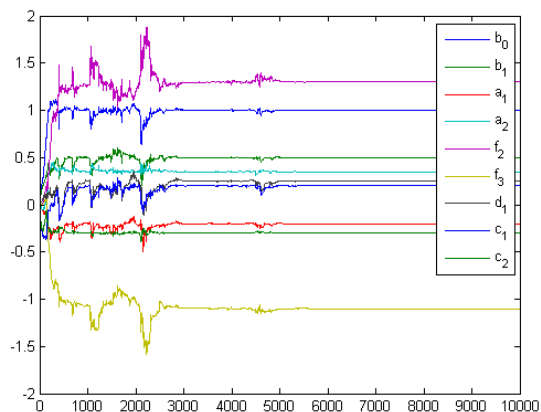
where  $u$ ,  $v$ ,  $x$ , and  $y$  are defined as in Fig. 9. The first thing to notice about this is that the non-linearity is very strong; the squared component of the signal is larger than the original, which we would not typically expect to see in a valve non-linearity. However in the interests of validating the code, we shall initially stick to this model for testing.

The implementation does recover the parameters of the above system, albeit with a few caveats. Firstly, Vörös predicts that convergence occurs after only 8 iterations whereas we see (Fig. 19a) that in reality this takes much longer at around 6000. This may be due to differences in the definition of an “iteration”, but given the conventions in similar literature this seems unlikely. Furthermore, later work by Vörös [5] finds that convergence times are of the order of at least 1000 iterations, suggesting that the value of 8 may be an error.

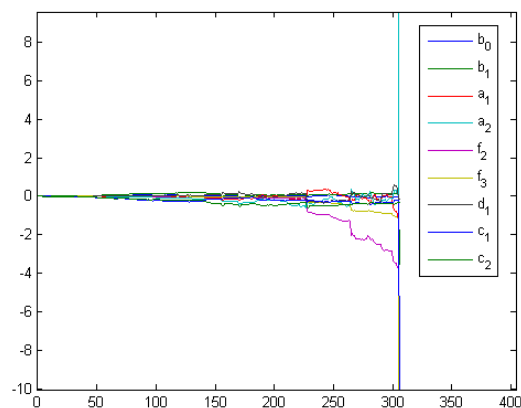
Secondly, the method appears to be highly unstable. Even tiny perturbations to the model system can result in divergences: Fig. 19b demonstrates what happens when the quadratic term in the non-linearity is changed from 1.3 to 0.3, the parameter estimates explode over only a couple of iterations. It is unlikely that this is due to a specific peculiarity in the input as this uses the same signal as in Fig. 19a. It is with some pessimism then that this method is applied to the guitar pedal, and as expected the results are disappointing (Fig. 19c), even when searching for a large number of parameters (Fig. 19d).

It is certainly possible that this instability is due to an error in the implementation: recursive least squares methods are very well established so the underlying method is unlikely to be incorrect. It is possible however that the extension to Wiener-Hammerstein systems as proposed by Vörös is unstable, or at least not completely general.

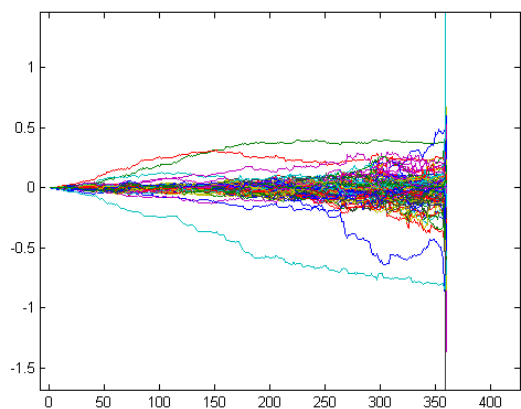
On the other hand, when working, the model does manage to separate out and parametrise both linear filters *and* the non-linearity which would be very useful in understanding the different stages of the valve. By comparison, Volterra methods may be able to reproduce system behaviour but don’t necessarily give as much physical insight into the individual components. Either way, the method certainly appears unsuitable for use in this project, at least in its currently form.



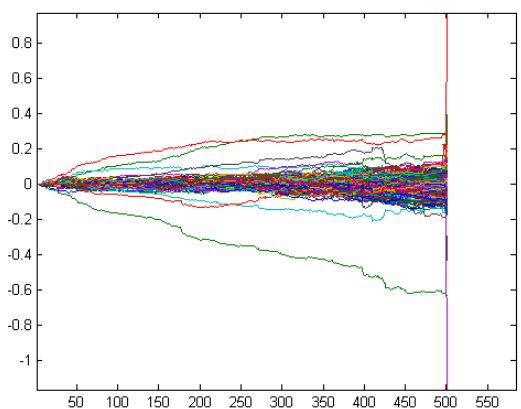
(a) Results for a system exactly as described in the paper. Stability is achieved at around sample 6000.



(b) Results for a system differing only slightly from that described in the paper. Note the divergence at around sample 300



(c) Results for Guyatone TD-1, with  $n_a, n_b$  etc all equal to 20. Notice that divergence occurs at about sample 350



(d) Results for TD-1 repeated with  $n_a, n_b$  etc all equal to 100 instead. The system still diverges, this time at around sample 500.

**Figure 19:** Plots of how parameter estimates evolve with time using the Vörös model for identifying Wiener-Hammerstein systems.

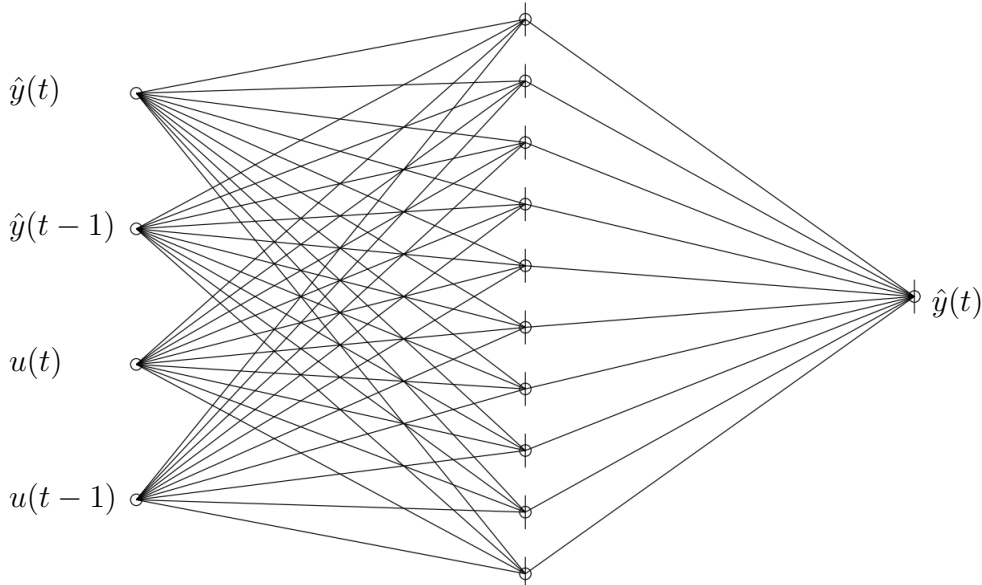
## 4 Neural Networks

### 4.1 Introduction to Neural Networks

One of the promising areas of the project has been in the use of artificial neural networks (or ANNs). In this section we will see how this quite general computational model can prove to be a very effective black box technique for valve amplifier identification.

The neural network paradigm started to gain popularity in the 1960s. Advances in neuroscience meant that scientists, for the first time, were starting to built up a picture of how neural activity in the brain was linked to learning and the computational model took inspiration from this, albeit in a greatly simplified form. Since then it has fluctuated in and out of favour, often the idea was oversold with some proclaiming this to be the holy grail of machine learning, while others strongly doubted its usefulness as a generalised solving tool [20]. The reality is somewhere in between, and for the simplistic cases used in this project it is more than adequate.

### 4.2 Practical Implementation



**Figure 20:** Example second order neural network with  $H = 10$  hidden *tanh* units, for input  $u$  and predicted output  $\hat{y}$ . Figure adapted from [21].

The ANN model mimics neural pathways in the brain by connecting together a series of nodes (neurons in this analogy). By application of an appropriate algorithm, the system can be trained with a input and output sequence which produces a set of weights for each connection in the network. The model itself is quite general; here we are specifically interested in its signal processing capabilities, particularly in non-linear system identification.

The NNSYSID toolbox, available to download for MATLAB from [21], provides one such

implementation. Here, the middle layer of “hidden” neurons are used to model the non-linearity. This is achieved by using a hyperbolic tangent function as a basis unit, although in practice this can be computationally expensive and more efficient sigmoidal shapes may be used instead<sup>§</sup>. On the left hand side of Fig. 20, we have the input signal,  $u$ . For a given point in the signal  $u(t)$ , the inputs to the network consist of this along with the  $n_b$  previous samples, i.e.  $u(t-1), u(t-2), \dots, u(t-n_b)$ . Feedback may also be included by looping the predicted output  $\hat{y}$  back into the network, similarly with  $n_a$  previous samples,  $\hat{y}(t-1), \hat{y}(t-2), \dots, \hat{y}(t-n_a)$ . Now the system is capable of modelling both non-linear and memory effects.

The procedure then requires two stages: first the initial estimate for the network weights is made using a training signal (e.g. noise signal), then the accuracy of the model is evaluated using a different signal. As an optional middle stage, the network can be optimised by removing unnecessary connections in a process called *Optimal Brain Surgeon*, or OBS [22]. While OBS is computationally very heavy, often requiring many hours to run, it need only be performed once, and only after a satisfactory model has been found: the results can then be saved to disc.

### 4.3 Results

White noise has proven to be successful for other methods in the project, and with no suggestions from the toolbox documentation as to which signals to use for training, it seems like a sensible choice to begin with: the wide range of frequency content should help to properly explore any frequency-dependent behaviour of the system. Although the training algorithm only scales linearly with the length of the input  $N$ , runtimes even for relatively short signals (of the order of a second) are measured in hours, and we cannot train with much longer signals within a computationally sensible time frame.

For initial testing the TD-1 was used; again a gain setting of 7 out of 10 was chosen as this allows us to examine both the interesting non-linear behaviour (which would be negligible at very low gains) without the full complexities of the valve running at maximum capacity. The toolbox offers a number of models, but it is the recommended model NNARX<sup>¶</sup> that appears to perform best. Some trial and error was required to find the minimal number of hidden *tanh* units  $H$ ,  $n_a$ ,  $n_b$  that produced sensible results, and we find that  $H = 30$ ,  $n_a = n_b = 50$  works reasonably well.

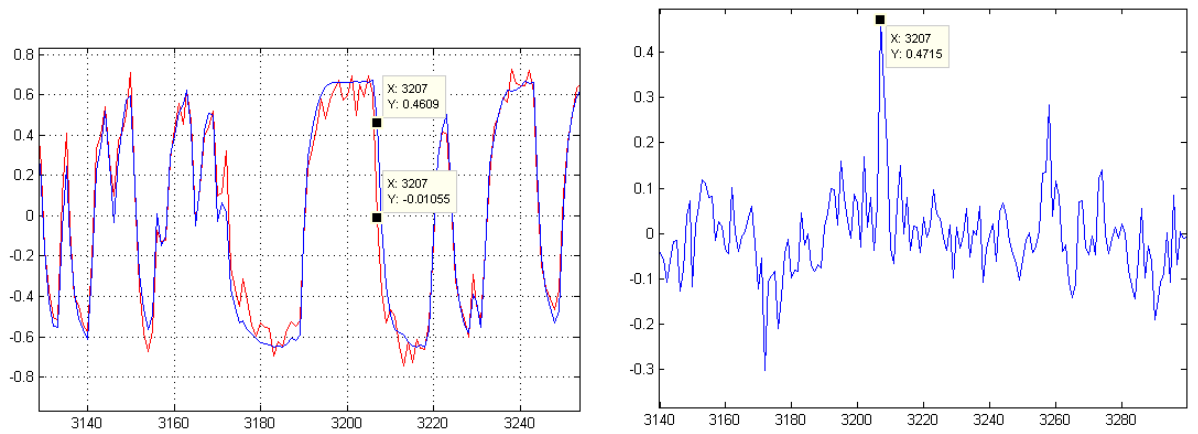
Validating the system with a new noise signal produces the plot in Fig. 21a; we can clearly see that the simulated output matches the measured results fairly closely. The main difference appears to be that the model fails to pick up some of the Nyquist type frequencies in the peaks and troughs, though this shouldn't be too important as it is unlikely that human hearing would be sensitive enough to make out differences of this magnitude.

The plot of prediction error shown in Fig. 21b (defined as the difference between simulated output  $\hat{y}$  and actual output  $y$ ) is also encouraging as we see that error is generally within 0.1 of the original signal. In fact, if we examine the regions where the error is greatest

---

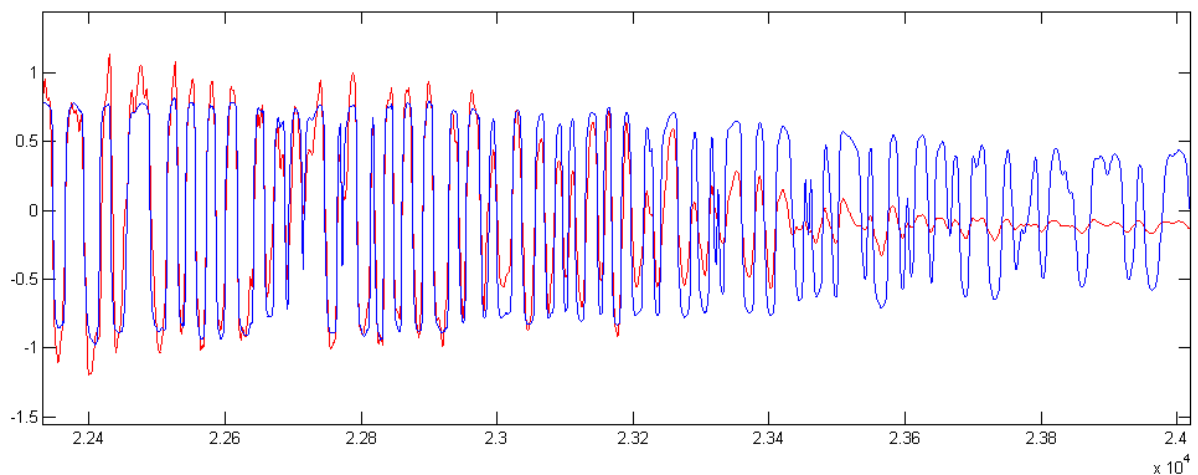
<sup>§</sup>A fast *tanh* method is included in the toolbox using the approximation:  $\tanh(x) = 1 - 2/(e^{2x} + 1)$

<sup>¶</sup>Neural Non-linear Auto Regressive Exogenous model



(a) Comparison of simulated (blue) and actual (red) outputs. Sample 3207 is marked on both signals displaying a large vertical gap despite similarity of signals. (b) Plot of the prediction error ( $e = y - \hat{y}$ ) without prior knowledge of the output.

**Figure 21:** Output from the neural network model trained and validated using noise. The plots demonstrate how the prediction error isn't always a good measure of accuracy, with particular reference to sample 3207 in this example.



**Figure 22:** Plot of simulated (blue) and actual (red) outputs for a guitar signal. Notice how the model copes well with high amplitude signals on the left but doesn't recreate low amplitude behaviour on the right (although the frequency is at least correct).

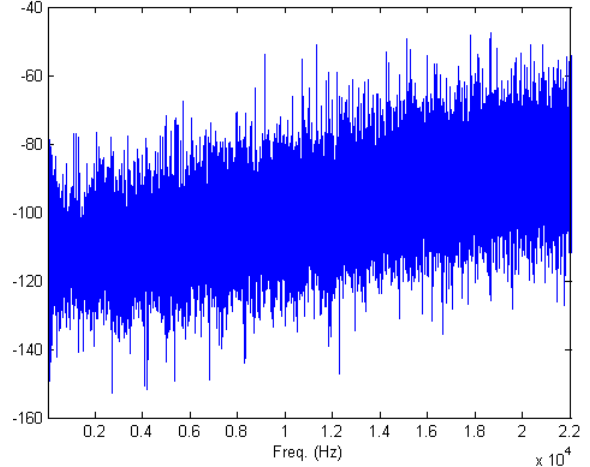
(for example the spike at sample 3207) we see that the model still maintains a very close fit to the original signal, but is lagging behind by 1 sample. As the slope of the signal is very steep (i.e. we have some high frequency content) a delay of 1 sample actually results in quite a large vertical gap, hence the large error. Again this should not be of any concern as we are dealing with an occasional delay of a  $10000^{\text{th}}$  of a second - well beyond the discernibility of the human ear!

So how does this model fare when applied to guitar signals instead? Fig. 22 shows the a comparison of the model (blue) and the actual output (red). For high amplitudes, the model appears to cope reasonably well and manages to recreate the basic shape of the signal (as seen on the left of the diagram). However as the amplitude decreases,

we see that the model fails to produce the desired output, though with close inspection we can see that at least the correct frequency is retained. One reason for this may be that the noise signal used for training has a fairly consistent amplitude that doesn't fully explore the range that real guitar signals may possess. To try to overcome this, various ramping amplitude envelopes were applied to the input signal but these failed to have any significant effect. Listening to the audio results (Appendix A.4), we can hear that the actual output has much high frequency content than the model. We can create an impulse response filter describing the difference between the two outputs by:

$$h(t) = \mathcal{IFT} \left\{ \frac{\mathcal{FT}\{y\}}{\mathcal{FT}\{\hat{y}\}} \right\} \quad (8)$$

The  $\mathcal{FT}$  of this filter is plotted in Fig. 23 and this confirms the disparity of high frequency content between the two. As seen in Sect. II.3, more extreme distortion will produce more high frequency products (as the signal becomes more square-like) and from this we can infer that the neural model may be on the right track, but that the strength of distortion isn't quite enough. Multiple applications of the model to a signal may rectify this somewhat. Alternatively, it may be possible to use some corrective filtering to bring the model output in line with real recordings. The filter in Eq. 8 will be specifically tailored for a specific signal only, but a similar design could be recreated in MATLAB's filter design toolbox for example.



**Figure 23:** Frequency response (in dB) of the filter relating predicted and actual outputs. This shows that the model output is lacking high frequency content.

An alternative method is to use a guitar signal for both training and validation, although some creativity is required to create a sample that is short enough to keep the computation times down yet contains enough information to be representative of typical input signals. For example, a sequence was prepared that consisted of many different short samples of the just the attack stage of various guitar chords, but this caused the model to be wildly inaccurate when reapplied to alternative signals. Additionally, several long running<sup>||</sup> simulations were also set up using a 4 second guitar sample for training, but these also failed to produce sensible output. Interestingly, in both these cases when the model was applied to a different validation signal, the output appeared to contain traces of the training sequence which suggests that the neural network may be just trying to replicate a specific signal rather than to produce a working model of the system. This is in line with the results from using noise as a training signal: in that case there is no periodicity or specific signal characteristics to corrupt the training stage and a reasonably good model is produced.

The neural methods have shown a lot of potential, particularly whilst using noise as a training signal. However more work is required in the design and study of training

<sup>||</sup>10-12 hours long

sequences, in particular with respect to probing the varying amplitudes, and other shapes of amplitude envelopes (such as exponential or sinusoidal) may improve this. The zero-interleaving technique mentioned in Sect. III 2.2.1 may also have applications here.

## Part IV

# Evaluation of Practical Implementation

An important consideration for all of these methods is how well a practical realisation on current hardware is. Methods that provide an exact model of the system but require computing power that is generations away are of limited interest outside an academic environment. With this in mind, we now evaluate the computational requirements of the main methods.

The only computationally intensive stage in the Hawksford method is the inversion of the input matrix. The cost to both memory and processing time goes as  $M^2$  for an  $M$ th order Volterra series which may make exploring very high order systems prohibitively expensive. However the results of this inversion can be stored and used for testing multiple systems so the cost can be seen as one off, given that enough memory is available. In comparison, sine sweep calculations are essentially linear in  $M$  and although there is one-off matrix inversion that goes as  $M^2$ , the effect of this is negligible.

Both Volterra methods eventually produce a set of kernels which can be written to disk. Convolution in the time domain is a particularly inefficient way to apply these kernels, instead it's likely a practical implementation would perform this as multiplication in the frequency domain. The problem with the Volterra methods in general is that an  $M^{\text{th}}$  order system requires  $M$  separate Fourier Transforms, and copies of the signal at  $M$  different powers. The computing power required for this type of model is probably just outside the reach of current dedicated DSP consumer hardware, but commercial VST/AU\*\* implementations are appearing [23] for higher end machines.

The parametrisation described by Vörös would possibly fare best of all the methods on current consumer hardware. Digital methods for applying linear filters are well within the capabilities of modern hardware; the non-linearity may require some creativity to be applied efficiently, for example lookup tables have been employed with reasonable degrees of success [24].

The normal neural network model has much heavier computational requirements. Both the storage and application of the large matrix of weights, combined with the computational cost of the using trigonometric basis units for the non-linearity make this one of the least attractive for DSP chips. While aggressive “brain surgery” may reduce matrix sizes somewhat, there still remains a prohibitively large number of operations to be performed. This is in line with the fact that the majority of existing neural network applications are non-realtime. However it may be technically possible to create a hardware representation of the network in circuit form although any comment on the effectiveness of such a circuit would be pure speculation.

---

\*\*Windows/Mac formats of plugins for DAWs.

## Part V

# Conclusions

Throughout this project we have explored a wide range of different techniques, some serve better as pure black box models for only recreating system output while others provide a more physical explanation for what is happening.

While the neither of the Volterra methods have yet managed to produce a sensible black box simulation of pedal distortion, their results do provide some interesting details about the tube mechanics. By varying the length of zero-interleaving, the wide Hawksford method has shown that there is an extensive amount of memory *before* the non-linearity, of the order of 1000 samples. This also tells us that there is a strong possibility that the pedal at least conforms to some type of Wiener-Hammerstein model, although the possibility of further non-linearities and linear filters cannot be entirely ruled out.

It is encouraging that the results from the exponential sine sweep agree with those from the Hawksford method, if only very loosely. The dominance of the odd order impulse responses  $h$  and Volterra kernels  $k$  is in line with spectral analysis of tube output. However the presence of even order responses proves that the tube involves an asymmetric non-linearity: this result should be taken into consideration when designing any tube inspired waveshaper. Despite the disappointing audio output results, others have shown sine sweep methods to be effective in non-linear system identification; the generalisation presented in Sect. III.2.3 should allow others to explore much higher order Volterra systems even if the method currently appears to fail for tube distortion.

The parameter estimation method as outlined by Vörös has great potential for increasing our understanding of valve distortion. The ability to fully parametrise the valve would allow for a very efficient hardware implementation. DSP chips are easily capable of applying linear filters quickly, and low order polynomial distortion shouldn't pose too much of an issue either. However the method appears to currently be unstable, at least for the pedals tested in this project so further work would be required in this area.

At the other end of the spectrum, neural network methods have managed to produce a fairly accurate model of the pedal, although some work is required to guarantee correct operation at different amplitudes. The simulated audio output is encouraging, and while it is clearly distinguishable from the actual output, this can be fixed to a certain extent by creative corrective filtering. However the downside of this rather abstract method of modelling is that it fails to give any real insight into the physical mechanism of the pedal, though as this can be provided by other methods in the project this is not much of an issue.

The majority of the measurements in this project have been taken from the 12AX7 tube, a design which is fairly common in guitar amplifiers. Multi-tube stages in more complex hardware could be simulated by multiple applications of the model. Additional tone controls may also be introduced using some sort of hybrid model, for example it could be used in conjunction with the Fender tone stack as found by Yeh and Smith [25].

## Acknowledgements

I would like to thank Dr. Jonathan Kemp for his guidance and enthusiasm throughout the project, and in particular for the loan of the Gytone guitar pedal which was used extensively throughout the project.

# Appendices

## A Instructions for Code

All MATLAB code used in this project can be found in the code folder which should be supplied with this report (can also be downloaded at [2]). Instructions for use are as follows.

### A.1 Wide Hawksford

The important file here is `run_it_all.m`. This file calls methods to produce input sequences, then looks for output in the folder `audio/gain7` matching the pattern `y_lpX_mY_nZ`, where  $X$  is the amount of interleaving,  $Y$  is the number of Volterra kernels, and  $Z$  is the order defining the length of the sequence. The rest of the code should be well commented and should be self explanatory. Figures for the plots used throughout the project can be found in the `results` folder. Files `drumin.wav` and `drumout.wav` give an example of poor audio output of the method.

### A.2 Exponential Sine Sweeps

Again the key file here is `run_it_all.m`. The number of kernels  $M$  (on line 7) can be varied. To apply the trial system instead of real output, lines 38–50 should be uncommented and line 58 should be commented. When run, the method produces time and frequency domain plots of the resulting kernels as well as the full impulse response. Figures for the plots used throughout the project can be found in the `results` folder. The implementation of the generalised Volterra series can be found in `volterra_coefficients.m`.

### A.3 Voros Parameter Estimation

To run the Vörös code for a test system, the file `testmodel.m` should be executed. This will produce a plot of the evolution of parameter estimates against time. Varying the parameters in this file may or may not produce stable results. To test TD-1, the file `testrealsystem.m` should be run.

### A.4 Neural

To run the neural code for the pedal, the file `eh_example_neuralnetwork.m` should be executed. To test the different setups, the type variable (lines 25–27) should be varied. The order of the filters `order` and number of hidden units,  $H$  can be varied on the lines below. Note that runtimes can be several hours long depending on the configuration. With this in mind, the plots and audio results from each of these runs can be found in their respective folders. For example to hear the results of the model trained by noise and

validated with guitar signals, you should look at the `gain7guitarvalidate` folder. Files `u1.wav` and `y1.wav` are the training sequence inputs and outputs; `u2.wav` and `y2.wav` are used for validation. **Note:** The neural network toolbox (included in `Toolboxes/Neural Networks`) is also needed. You may need to modify the path (line 10) to get this to work.

## A.5 Background Noise Analysis

Simple script to demonstrate the 50Hz hum, run `show_50Hz.m` for graphs.

## A.6 Brute Force

Another simple script to demonstrate the impracticalities of brute force parameter estimation methods, can run `show_50Hz.m` for results (quite slow).

# References

## References

- [1] Ewan Hemingway. MPhys Project Literature Review, 2010.
- [2] Ewan Hemingway. MPhys Project Files Download. <http://www.ewanhemingway.co.uk/sites/default/files/downloads/MPhys.zip>, Retrieved 2011. If unavailable contact `s0673758@sms.ed.ac.uk` for source files.
- [3] R. Linde. *Build your own AF valve amplifiers: circuits for hi-fi and musical instruments*. Elektor Electronics, 1995. pg. 3.
- [4] D. Brosnac. *The Amp Book: A Guitarist's Introductory Guide to Tube Amplifiers*. Bold Strummer Ltd, 1987.
- [5] J. Vörös. Recursive Identification of Systems with Noninvertible Output Nonlinearities. *Informatica*, 21:139–148, January 2010.
- [6] Malcolm J. Hawksford. System Measurement and Identification Using Pseudorandom Filtered Noise and Music Sequences. *J. Audio Eng. Soc*, 53(4):275–296, 2005.
- [7] Wikipedia: Clipping compared to limiting. [http://en.wikipedia.org/wiki/File:Clipping\\_compared\\_to\\_limiting.svg](http://en.wikipedia.org/wiki/File:Clipping_compared_to_limiting.svg), Retrieved 2011.
- [8] The Common Cathode, Triode Gain Stage [pdf]. [http://www.freewebs.com/valvewizard1/Common\\_Gain\\_Stage.pdf](http://www.freewebs.com/valvewizard1/Common_Gain_Stage.pdf), Retrieved 2011.
- [9] VintageAmps.com forum discussion on Guyatone TD-1. <http://vintageamps.com/plexiboard/viewtopic.php?f=6&t=86235>, Retrieved 2011.

- [10] TheGearPage.net forum discussion on Guyatone TD-1. <http://www.thegearpage.net/board/showthread.php?t=69096>, Retrieved 2011.
- [11] M.L. Meade and C.R. Dillon. *Signals and systems: models and behaviour*. Tutorial guides in electronic engineering. Chapman & Hall, 1991. pg. 61.
- [12] S.A. Billings. Identification of nonlinear systems - a survey. *Control Theory and Applications, IEE Proceedings D*, 127(6):272–285, November 1980.
- [13] E. Armelloni, A. Bellini, A. Farina. Not-Linear Convolution: A New Approach For The Auralization Of Distorting Systems. In *Audio Engineering Society Convention 110*, 5 2001.
- [14] J. Kemp, H. Primack. Characterisation of non-linear systems using impulse response techniques. *Journal of the Audio Engineering Society*, 2011.
- [15] A. Farina. Simultaneous Measurement of Impulse Response and Distortion with a Swept-Sine Technique. In *Audio Engineering Society Convention 108*, 2 2000.
- [16] G. Chrystal. *Algebra, an elementary text-book for the higher classes of secondary schools and for colleges*. Chelsea Publishing Series. Chelsea Pub. Co., 1964. pg. 278.
- [17] A.C. Aitken. *Determinants and matrices*. University Mathematical Texts. Oliver and Boyd, 4th edition, 1944. pg. 55.
- [18] J. Vörös. An Iterative Method for Wiener-Hammerstein Systems Parameter Identification. *Journal of ELECTRICAL ENGINEERING*, 58(2):114–117, 2007.
- [19] L. Ljung. *System Identification: Theory for the User*. Prentice Hall PTR, 1987.
- [20] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the theory of neural computation*. Santa Fe Institute studies in the sciences of complexity: Lecture notes. Addison-Wesley Pub. Co., 1991.
- [21] Neural Network System Identification Toolbox (NNSYSID). <http://www.iau.dtu.dk/research/control/nnsysid.html>, Retrieved 2011. Magnus Nørgaard.
- [22] B. Hassibi, D. G. Stork, and G. J. Wolff. Optimal Brain Surgeon and General Network Pruning. Technical report, RICOH California Research Centre, 1992.
- [23] Acustica: Nebula 3 Pro VST. [http://www.acustica-audio.com/index.php?option=com\\_content&view=article&id=43&Itemid=144](http://www.acustica-audio.com/index.php?option=com_content&view=article&id=43&Itemid=144), Retrieved 2011.
- [24] G Kramer. Digital signal processor for providing timbral change in arbitrary audio and dynamically controlled stored digital audio signals. *US Patent No. 4991218*, Issued 1991.
- [25] D. T. Yeh, J. O. Smith. Discretization of the '59 Fender Bassman Tone Stack, in Proc. of the 9th Int. Conference on Digital Audio Effects. Sept 2006.