Master Thesis on Sound and Music Computing

Universitat Pompeu Fabra

Pultec EQP-1A Modeling with Wave Digital Filters

Alberto Barrera Herrero

Supervisor: Xavier Lizarraga

August 2023

Master Thesis on Sound and Music Computing

Universitat Pompeu Fabra

Pultec EQP-1A Modeling with Wave Digital Filters

Alberto Barrera Herrero

Supervisor: Xavier Lizarraga

August 2023

Contents

Dedication

To all the beautiful people I've had the pleasure to share this wonderful year with, and to my family that has always supported me.

Abstract

This thesis presents the development of a virtual analog model of the passive equalizer section of the Pultec EQP-1A studio equalizer using Wave Digital Filters (WDF). The aim of the project was to provide an accurate and high performance open-source emulation of the circuitry and sound characteristics of the original hardware unit.

The development process involved compiling the original unit's schematics, generating LTSpice simulations, and implementing the circuit in Python using the pywdf library and R-Type adaptors (a kind of adaptor used for modeling complex circuit junctions that cannot be classified as series or parallel). Since the R-Type adaptors greatly affected the performance of the model, the circuit was slightly modified to maintain its behavior without the need for R-Type adaptors.

The frequency response of the Python prototype was compared to the LTSpice simulation showing that at sufficiently high sampling rates the error between the model and the simulations are minimal.

The Python model was then ported to C++ using the JUCE framework and Chowdsp's wdf library to generate a VST3 plug-in that can be loaded into digital audio workstations. The plug-in has oversampling capabilities to preserve the adequate behavior of the circuit at frequencies close to Nyquist.

The performance and accuracy of the Python model was measured, and the C++ implementation compared against another open-source implementation of the circuit using WDFs and R-Type adaptors (developed in the Faust programming language). The final EQP-1A Python model was 75% faster than our own one that used R-Type adaptors and the C++ implementation was 40% faster than the EQP-1A implementation in Faust and a much more accurate emulation of the original circuit.

Keywords: Virtual Analog Modeling, Wave Digital Filters, R-Type adaptors.

Chapter 1

Introduction

1.1 Motivation

Analog audio gear has been used for music production for almost a century and continues to be used extensively nowadays. These devices, such as compressors, equalizers, and tube amplifiers, have a distinct character, due to the distortion and nonlinearities that they introduce into the processed signal, that many producers and mixing engineers still seek today.

Analog gear, however, can be fragile, needs maintenance, takes up physical space in the studio and is more expensive to acquire than its digital counterparts. It is also less convenient to use since it does not allow to recall configurations or use presets (like plug-ins do). As more of the processes involved in music recording, editing and post-processing are being done in the digital domain, these devices are becoming more and more scarce.

Trying to overcome all these disadvantages, virtual analog modeling techniques have long been of great interest to replicate the sound characteristics of analog gear in a digital environment. There are many different approaches to virtual analog modeling (covered in chapter 2), but the Wave Digital Filters (WDF) approach has gained considerable recognition due to its simplicity and modularity and ease to adapt to real-time applications.

With this increasing scarcity of analog devices, virtual analog modeling techniques (and WDFs in particular) help preserve these analog circuits in a digital form and promote the understanding of both analog electronics and digital signal processing.

The last factor that motivated the project was the development of the `pywdf` [1] library by Gustav Anthon at UPF last year, that makes going from a circuit schematic to a working WDF prototype model exceptionally easy and fast.

1.2 Objectives

The main goal of the project was to provide an accurate and high performance open-source virtual analog emulation of the circuitry and sound characteristics of the Pultec EQP-1A hardware unit using WDFs. The virtual analog model should be able to compile into a VST3 plug-in that can run in real time and replicate the behavior of the original unit accurately.

1.3 The Pultec EQP-1A Equalizer

The Pultec EQP-1A is a classic analog equalizer that has become a favorite among audio professionals and that can be heard on thousands of recordings made over the past 70 years [1].

The original Pultec EQP-1 was the first program equalizer to be introduced to the market in 1951. It was very well received, and some units were installed in recording studios but mainly in radio stations to equalize the broadcast program. According to the original manual it was intended to "add the 'final touch' to the balance of good program material and to improve the quality of program material previously recorded on equipment of inferior quality or differing characteristics".

The device of interest for this thesis is the EQP-1A, a revision of the EQP-1 consisting of the same passive equalizer stage included in the original unit plus a vacuum tube gain stage to restore the insertion loss caused by the equalizer stage.

[1] `gusanthon/pywdf: https://github.com/gusanthon/pywdf`

Figure 1: Front panel of the Pultec EQP-1A hardware unit.

The passive equalizer stage consists of four different filters with selectable cutoff or center frequencies and gain or attenuation controls:

- **Low frequency boost:** shelving filter that amplifies the low frequencies below a frequency point selectable between 20 Hz, 30 Hz, 60 Hz, or 100 Hz.
- **Low frequency cut:** shelving filter that attenuates the frequencies below the same frequency selected for the low-boost filter but slightly shifted up in frequency. This shift allows to boost and cut the same frequency range giving the EQP-1A its characteristic curves.
- **High boost resonant filter:** resonant RLC peak boost centered around 3 kHz, 4 kHz, 5 kHz, 8 kHz, 10 kHz, 12 kHz, or 16 kHz. The width of the filter can be adjusted using the bandwidth knob.
- **High frequency cut:** the high cut shelving control attenuates or cuts the frequencies above the selected frequency between 5 kHz, 10 kHz, and 20 kHz.

The responses of the individual filters from the original manual of the unit are included in figure 2.

Although the EQP-1A manual advises against boosting and attenuating simultaneously on the low frequencies, when both are applied at the same time, the boost and cut curves affect slightly different frequency bands. This allows to boost the low end, adding body to the sound while attenuating the low-mid frequency range to avoid "muddiness". The magnitude response of the equalizer when both boost

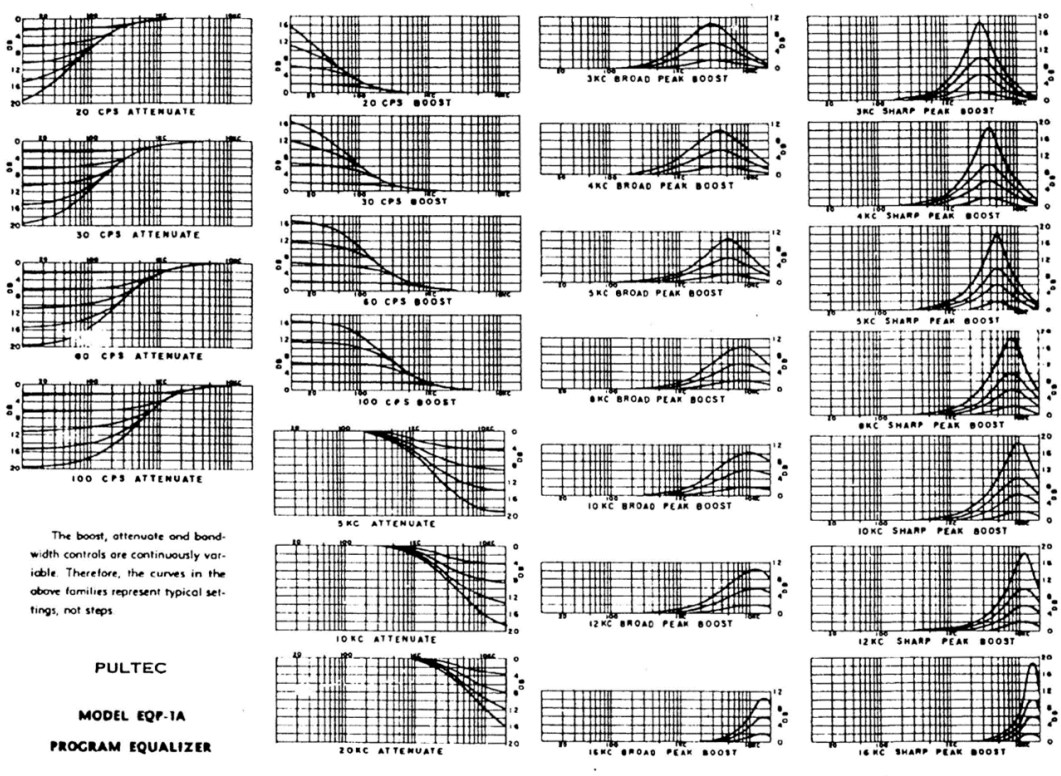


Figure 2: Frequency response curves from the EQP-1A manual.

and attenuation are applied simultaneously at 60 Hz is included in figure 3. This effect is commonly known as the "Pultec low end trick" and it is used extensively to enhance the sound of vocals, drums, guitars, or bass.

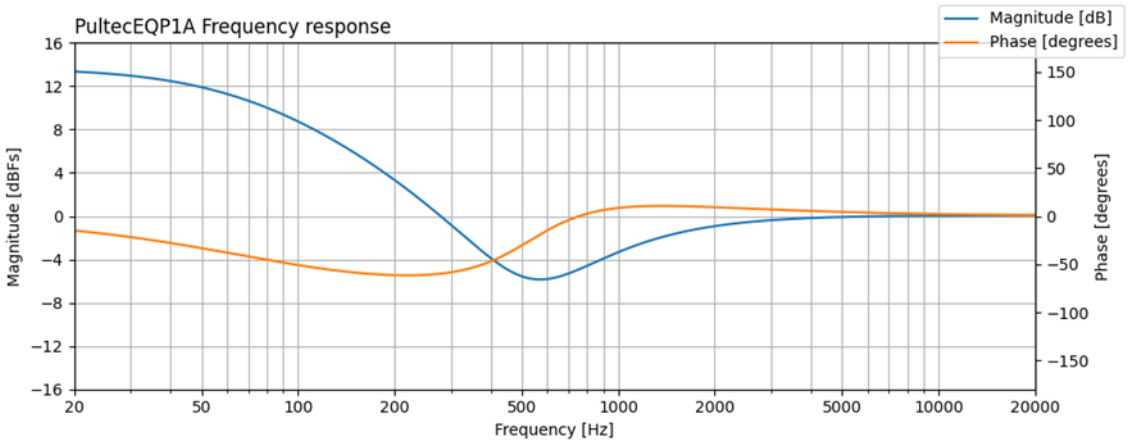


Figure 3: Frequency response when boosting and attenuating at 60 Hz.

Because the equalizer stage is passive, it introduces a loss of level in the signal. To make up for the 16 dB insertion loss, the signal level is restored using an output gain stage based on a 12AX7 and a 12AU7 vacuum tubes.

Chapter 2

State of the Art

2.1 Virtual Analog Modeling

Virtual analog modeling is the task of replicating the behavior of analog circuits in the digital domain. The objective of virtual analog modeling is generally to generate digital emulations that accurately replicate the sound of vintage analog hardware, including studio equipment such as compressors, equalizers, and reverb units, as well as synthesizers, amplifiers, and effect pedals.

This technique has grown in popularity in recent times due to the scarcity, cost, and difficulty in obtaining or building analog gear. The application for a virtual analog model is generally real-time audio processing. This means that for a model to be useful it should be lightweight enough for a consumer-grade computer to run it alongside other audio effects and synthesizers.

2.2 Nonlinear Systems

A nonlinear system is any system for which its output is not directly proportional to its input [2]. Audio equipment typically falls into this category as their electronic circuits often have nonlinearities that introduce distortion into the processed signal. This distortion can result in changes in the harmonic content and dynamic range of

the signal.

Nonlinearities in audio equipment are caused by the physical characteristics or limitations of the components (vacuum tubes, transistors...) or by design (compressors, for example). These nonlinearities can add color and character to the sound, which is often desirable and looked for in music production but are very difficult to replicate in the digital domain [3].

Several techniques can be used to model the behavior of analog nonlinear systems in the digital domain. Included in the following section are the most relevant ones for virtual analog modeling.

2.3 Modeling Techniques

In this section white-box and black-box approaches to virtual analog model are explained and different modeling techniques belonging to each of them are reviewed.

2.3.1 White-box Approach

The white-box approach to virtual analog modeling refers to the group of techniques that aim to model the behavior of a system by modeling the characteristics of each component of the system and the physical interactions between them.

Nodal Analysis

To model a circuit using nodal analysis, the circuit is first transformed into the Laplace domain to obtain its transfer function and then converted to the digital domain [4]. It is a common technique used for virtual modeling of analog circuits due to its simplicity and computational efficiency, but it has some limitations. Specifically, it is limited to modeling linear time-invariant (LTI) systems, making it unsuitable for nonlinear systems unless extended using Modified Nodal Analysis [5]. One disadvantage of this approach is that changes in circuit element values may require recomputation of large portions of the circuit.

Port-Hamiltonian Formulation

The Hamiltonian approach to modeling, coming from analytical mechanics, and the network approach, from electrical engineering, can be combined in port-Hamiltonian systems. It combines both approaches by using a geometric structure, given by a Poisson or Dirac structure [6].

Port-Hamiltonian systems are open dynamical systems, which interact with their environment through ports. This formulation allows to model nonlinear electrical circuits. Each element of the circuit is model as a Port-Hamiltonian system and all of them interconnected using port-based networking models [7].

State-Space Formulation

A state-space representation is a mathematical model of a physical system that can be used to represent a circuit as a set of input, output and variables related by first-order differential equations [8]. It has been successfully applied to modeling of nonlinear systems [9] and it is one of the most common virtual analog modeling techniques, but it is not always suitable for real-time applications [3].

Wave Digital Filters

Following Julius O. Smith definition in [10] a Wave Digital Filter (WDF) is a kind of digital filter based on physical modeling principles. WDF can be used to represent the physical state of an element, for example, the current physical state of a capacitor or an inductor. WDFs can also be considered a particular kind of finite difference scheme with unusually good numerical properties.

A more detailed review of this technique is included in section 2.3.3.

Advantages and Disadvantages

The white-box modeling approach offers various **advantages** over the black-box approach:

- By considering physical interactions between all the system components, the mathematical models accurately represent the behavior of the system, even under extreme conditions.

- This approach also allows for easy mapping of control parameters to the model. UI controls can be mapped directly to specific components or variables of the model (like a variable resistor or condenser in a circuit).

- Lastly, these methods can provide an improved understanding of the modeled effect or audio processor. In the modeling process we can gain insight into how the different components of the system interact and how does those interactions affect the system's behavior.

Some **disadvantages** of using this approach to virtual analog modeling can be:

- Running the complex mathematical models required to characterize analog circuits is usually computationally expensive and sometimes not suitable for real-time applications.

- Another drawback of the white-box approach is that developing these models requires domain knowledge of physics and circuit theory as well as digital signal processing.

2.3.2 Black-box Approach

The opposite approach to white-box modeling is the black-box approach, that considers the system as a whole and tries to model it without considering the behavior of the individual components that conform it. It treats the system as an abstract entity with inputs and outputs, without the need to understand how the system works internally. Many different techniques fall into this category. The most relevant ones are highlighted next.

Swept Sine Method

The swept sine method was originally developed to measure the impulse response of rooms, but it was later extended for characterization of nonlinear systems [11]. To characterize a system, its input is excited using a sinewave signal of increasing frequency covering all the frequencies of interest for the modeling [12]. Comparing the output of the system to the introduced input signal, using the inverse filter, the transfer function of the system can be obtained. For nonlinear systems an impulse response is obtained for each of the different harmonics generated given a sinusoidal input signal.

Volterra Series

This method involves the use of the Taylor-like series defined by Vito Volterra in 1887. Using Volterra series it is possible to analyze nonlinear systems to create an explicit input-output relationship of nonlinear differential equations [11]. Unlike the Taylor series, Volterra series are able to model the memory of nonlinear systems. They have been widely used for modeling of audio systems, but the models become very complex for highly nonlinear systems and are not always suitable for real-time use.

Deep Neural Networks

Machine Learning and Deep Learning techniques can be applied to modeling of analog nonlinear audio circuits. In particular, recurrent neural networks (RNN) and WaveNet models have shown great results to model nonlinear effects like the distortion of guitar amplifiers [13], [14]. Using RNN or WaveNet implementations allows for smaller neural networks that can be evaluated much faster, even fast enough for real-time applications [4]. These and other light models are attracting a lot of attention, especially for real-time applications.

Advantages and Disadvantages

Black-box methods are usually better for capturing "unpredicted" behavior that is not reflected in the circuit schematic. It is a good approach for modeling circuits where there is not a complete understanding of the mechanisms involved in the processing.

These kind of methods are also often, but not always, computationally cheaper and faster to simulate than models that include the interaction between every circuit element or physical models.

One disadvantage of using black-box methods is that control parameters do not map to the model's parameters in an obvious way. The relationship between the model's parameters and the parameters available to the user is not always straightforward, making the mapping a complex task.

2.3.3 Wave Digital Filters

Using WDFs each circuit element can be defined by a port with an incident and reflected wave and a free variable [15], the port resistance, R_p. Following the explanation in [16], the incident and reflected waves, a and b respectively, at a certain port are defined in equations 2.1 and 2.2 and R_p can be set to any real value.

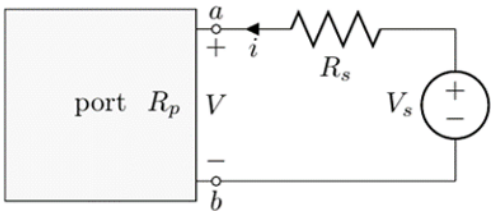


Figure 4: Variables of a WDF element.

$$a = v(t) + R_p \cdot i(t) \tag{2.1}$$

$$b = v(t) - R_p \cdot i(t) \tag{2.2}$$

Where $v(t)$ and $i(t)$ are Kirchhoff's domain variables instantaneous voltage and

current across and through the terminals of an element.

To go back from the WDF's domain to the Kirchhoff's domain variables $v(t)$ and $i(t)$ are defined as:

$$v(t) = \frac{a+b}{2} \tag{2.3}$$

$$i(t) = \frac{a-b}{2R_p} \tag{2.4}$$

Derivation of WDF Elements

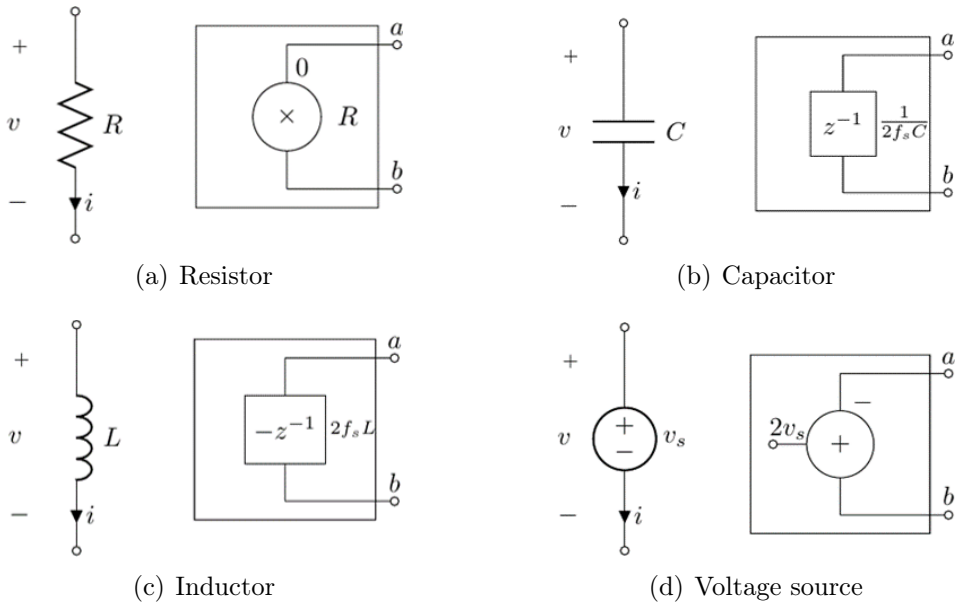
In order to make the WDF structure, first it is necessary to discretize each of the components of the circuit. The usual method for the discretization is the bi-linear transform but other methods can be employed.

One important thing to have into consideration when deriving the WDF elements is that delay-free loops (that would cause instability) must be avoided. This means that the instantaneous value of b, $b[n]$ cannot be proportional to $a[n]$. It can be proportional to a delayed version of it, $a[n-\tau]$, though. Usually R_p can be set to a value that avoids delay-free loops.

Table 1: Expression for the WDF's domain variables of the different circuit elements.

	$\boldsymbol{R_p}$	$\boldsymbol{b}$
Resistor	R	0
Capacitor	$1/(2f_sC)$	$z^{-1}a$
Inductor	$2f_sL$	$-z^{-1}a$
Voltage Source	Matches the rest of the circuit	$2v_s - a$

Using the bi-linear transform and setting R_p to avoid delay-free loops we can obtain the definitions included in table 1 for the different circuit elements in the WDF domain. The schematic representation of each element is included in figure 5.



(a) Resistor (b) Capacitor

(c) Inductor (d) Voltage source

Figure 5: Derivation of the Wave Digital Filter elements.

Series and Parallel Adaptors

Once all the necessary components have been individually discretized they can be connected using adaptors. In traditional WDF theory there is only two types of adaptors: series and parallel. Adaptors are two-port elements characterized by a reflection coefficient that determines what amount of each of the incident waves gets reflected into each of the connected components.

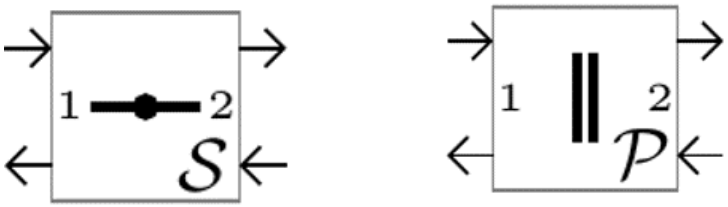
This limits the circuits that can be modeled using traditional WDF to those where all of the components are connected either in series or in parallel, leaving outside the circuits with more complex topologies. The expression for all the reflected waves on each type of adaptor is included in table 2 and the schematic representation used in WDF for series and parallel adaptors is included in figure 6.

Table 2: Definition of the reflected waves for the WDF adaptors.

	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$
Series	$-a_2 + \gamma \cdot (a_1 + a_2)$	$-a_1 + \gamma \cdot (a_1 + a_2)$
Parallel	$a_2 + \gamma \cdot (a_1 - a_2)$	$a_1 + \gamma \cdot (a_1 - a_2)$

Where γ is the reflection coefficient, defined as $\gamma = (R_1 - R_2)/(R_1 + R_2)$ and R_1

and R_2 are the port resistances of both elements connected to the adaptor.



(a) Series adaptor (b) Capacitor

Figure 6: Wave Digital Filter adaptors.

Computation

With all the circuit elements and adaptors defined, they are connected together in a structure called an SPQR tree [15]. At each computational cycle of a WDF algorithm it is necessary to first, recalculate and propagate the reflected waves towards the root of the tree. Then, the resulting incident wave is used to calculate the reflected wave at the root of the tree, which is then propagated downwards. Finally, the voltage values are calculated at the output of the circuit and values in reactive components are stored to be used in the next computational cycle [17].

Using classic WDF modeling only one nonlinear element can be introduced in the circuit. To do so, the nonlinear element is placed at the root of the SPQR tree and all the branches are connected to it. To model circuits with other than series or parallel configurations or circuits containing multiple nonlinear elements it is necessary to introduce special adaptors such as N-port or R-type adaptors [15].

Advantages and Limitations

About the advantages of using WDFs, the main one is its modularity [4]. It allows every circuit component to be discretized independently, even using different transformations for each one. This modularity also means that when a component of the circuit changes, only components with behavior that depends on the impedance of the changed component need to be recomputed and not the entire circuit.

One limitation of WDFs is their difficulty to model circuits with complex topologies

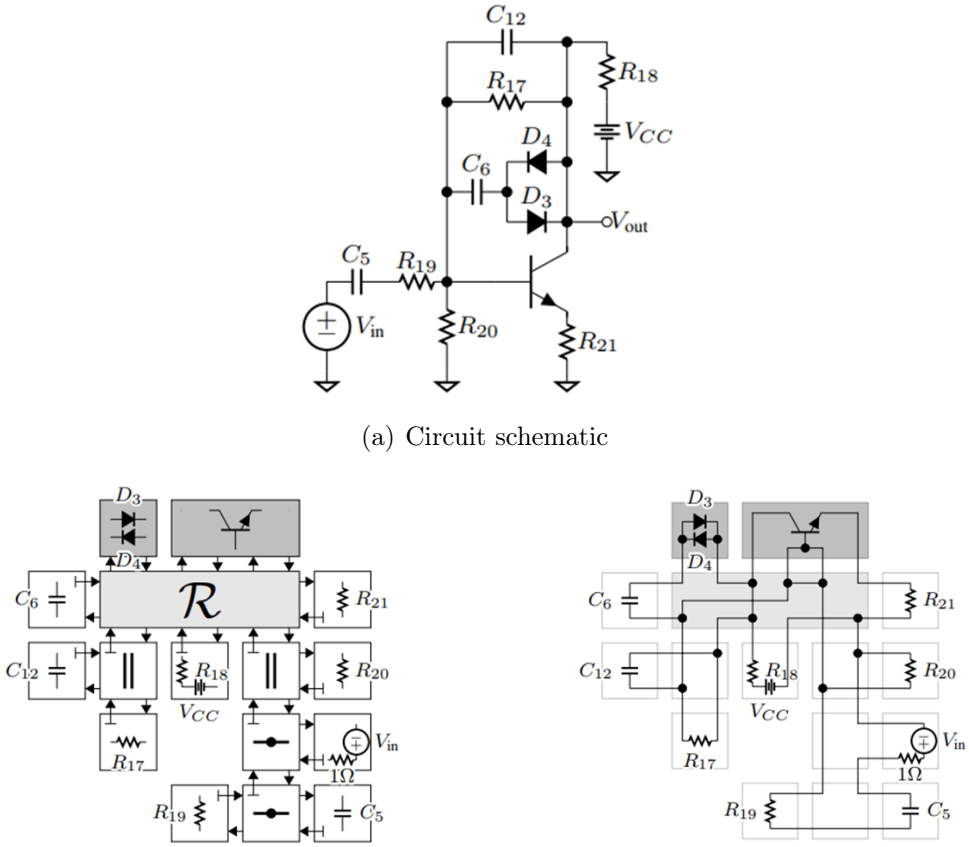
or multiple nonlinearities. While it is possible to model them using R-type adaptors, the resulting models are much more computationally heavy, and they are not easily adaptable to real-time applications.

R-Type Adaptors

Not every circuit can be decomposed into series and parallel connections. Circuits with more complex topologies or with multiple nonlinear elements are modeled using more general junctions called R (rigid) type adaptors characterized by its scattering matrix [15]. R-type adaptors include pure junctions as well as junctions that have absorbed multi-port linear elements [18].

Unlike series an parallel adaptors, that can be reused for every circuit, R-type adaptors are characterized by a scattering matrix that needs to be re-derived for each circuit model. It defines how waves are reflected to the elements connected to the adaptor. To work with R-type adaptors it is necessary to have a method for deriving their scattering matrices and adapting it to the rest of the circuit. There exist software tools like R-solver [19] to calculate the scattering matrices for this kind of junctions.

An example of the circuit of the *Big Muff Pi* clipping stage is included in figure 7, taken from [20]. It has a complex topology and multiple nonlinearities so it requires the use of R-type adaptors to modeled the circuit using WDFs.



(a) Circuit schematic

(b) WDF diagram

(c) Mixed diagram

Figure 7: WDF derivation of the *Big Muff Pi* clipping stage.

Chapter 3

Methodology

3.1 References for the Circuit

Since the original schematics for the circuit of the EQP-1A are not publicly available and we did not have access to a real hardware unit, we took as reference a few documents that can be found online. With all of them we can get a pretty good idea about the topology and components of the original device and what its frequency response is supposed to be like. All of the documents that we ended up using as reference are included in appendix A and were:

- The circuit diagram for the EQP-1R; another variant of the EQP-1A built by Pulse Techniques. We assume it has the same topology as the EQP-1A but different component values since the selectable frequencies for low and high frequencies are different in both devices. The schematic was taken from [21] and included in figure 20.
- The open-source circuit diagram for the EQP-1A clone by Gyraf Audio [22], included in figure 21. This circuit shares almost exactly the same topology as the EQP-1R with some minor differences.
- Frequency response curves from the EQP-1A user manual [23], included in figure 2 in section 1.3. They were very useful to get an idea of what the

behavior of the individual filters should be like but the low resolution did not allow to use it as a real reference to adjust the components of the circuit.

- Frequency response curves from the EQP-1A 2019 reissue made by the original manufacturer taken from this article from Sound on Sound [24] where they were measured using laboratory equipment. These ones have a better resolution than the ones from the original manual and were the ones used for reference after making sure that they were otherwise identical. The frequency response graphs from the article are included in figures 23, 24 and 25.

3.2 Circuit Simulations

3.2.1 LTspice

Since for this work we did not have access to a real EQP-1A hardware unit, LTspice was used and considered as the ground truth.

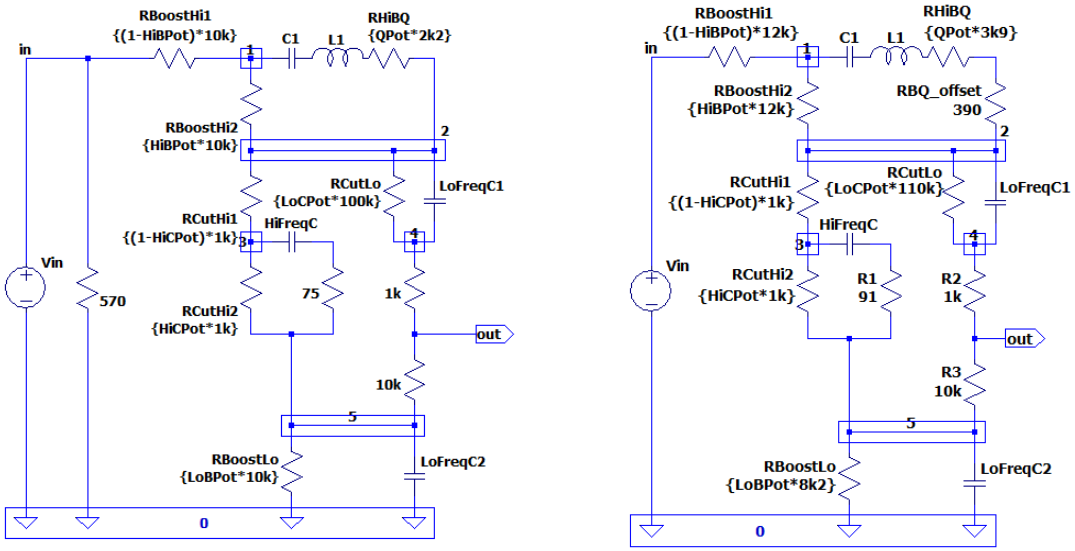
SPICE (Simulation Program with Integrated Circuit Emphasis) is a general purpose, open-source analog circuit simulator created in 1973 by L. W. Nagel and D. O. Pederson [25].

Based on SPICE, Linear Technology (now part of Analog Devices Inc.) developed LTspice [26]. It is a free-to-use circuit simulation software widely adopted in both academic and industrial settings as a tool for circuit analysis and design. Apart from using SPICE as its simulation engine, LTspice includes an extensive library of accurate models of passive components, transistors, operational amplifiers, integrated circuits, etc.

3.2.2 Simulations

Based on the circuit schematics of the EQP-1R [21] and Gyraf Audio's schematic [22] a first version of the circuit was simulated in LTspice. The LTspice circuit is included in figure 3.8(a). After a frequency response analysis of the circuit some discrepancies were found between the simulations and the measurements taken from

the 2019 reissue so the value of some component was adjusted to closely match the reference curves.



(a) First version

(b) Final version

Figure 8: Different versions of the circuit used for simulation.

Since there is no need to adapt the input and output impedances in the digital domain, the input and output resistors were removed, adjusting the necessary components to maintain the circuit's behavior. This allowed us to later model the circuit in the WDF domain without the need of R-Type adaptors and obtain better performance while keeping the circuit's response the same. The final version of the circuit after adjusting all the component values is included in figure 3.8(b).

The results of the simulations carried out on this modified version of the circuit are included for: the maximum low frequency boost and cut configurations for all of the selectable frequencies (figure 9), different bandwidth values for the resonant high frequency filter at maximum boost at 5 kHz (figure 10) and maximum cut for all of the selectable frequencies of the high shelving filter (figure 11).

All of the simulations were carried out with the same configuration as the measurements taken from the hardware unit (included in appendix A) so they could be directly compared against them. Although there is no way to numerically compare them, under visual inspection the response curves accomplished are identical.

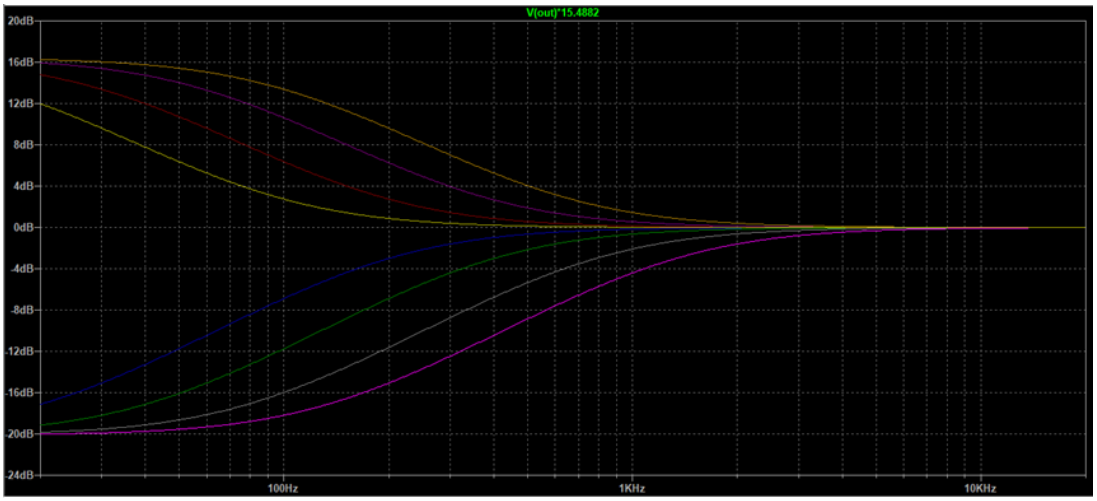


Figure 9: LTspice. Low frequency filters maximum boost and cut.

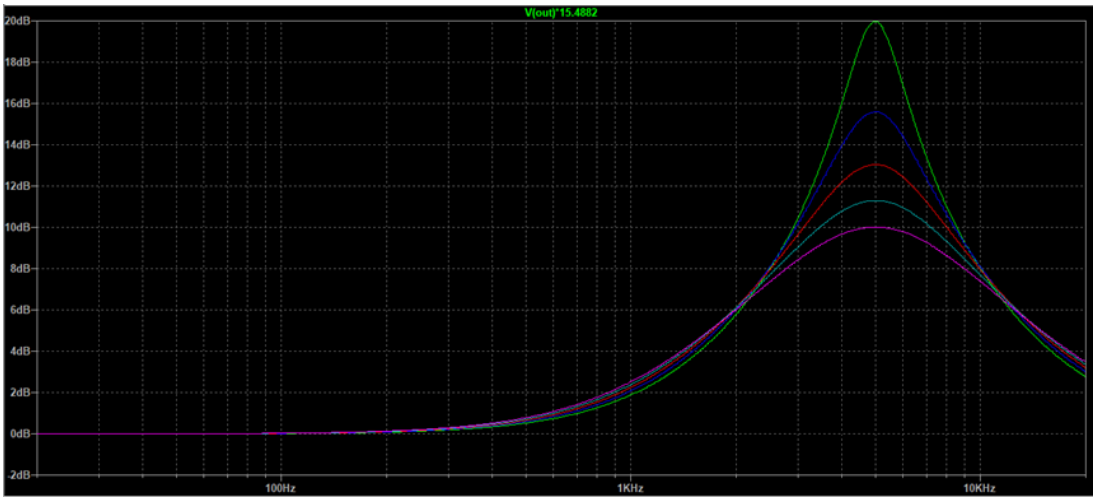


Figure 10: LTspice. Maximum boost with different bandwidths at 5kHz.

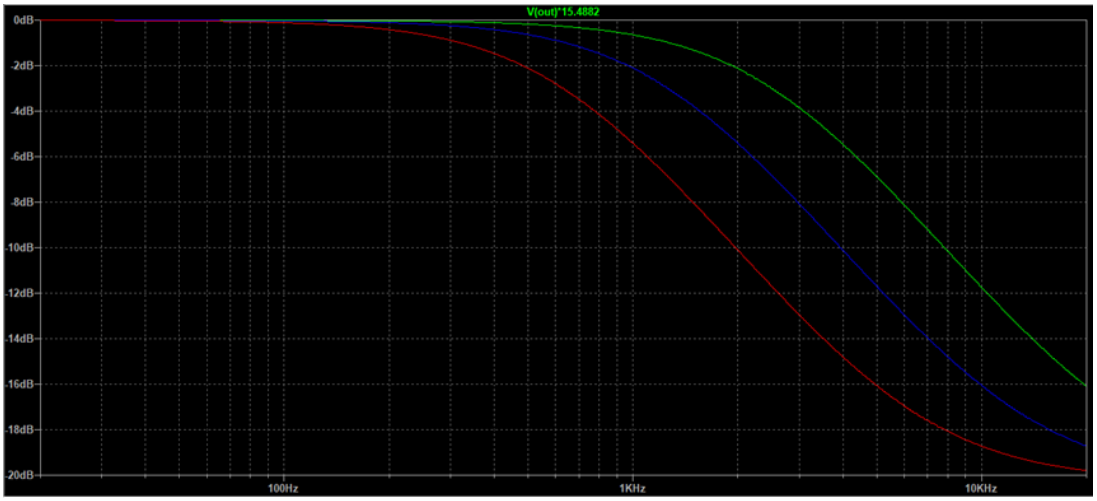


Figure 11: LTspice. Maximum attenuation for the high shelving filter.

3.3 Wave Digital Filter Implementation

With the frequency responses of the simulations in LTspice closely matching those measured on the hardware unit, the circuit is converted into a WDF structure using Python and the `pywdf` library. `pywdf` includes WDF models for resistors, capacitors and inductors as well as series, parallel and R-Type adaptors.

A first implementation of the circuit, that used R-Type adaptors, was developed following the circuit of figure 3.8(a). To derive the R-Type adaptor, the circuit's nodes and ports are labeled to create a net-list that describes the circuit. With this net-list the scattering matrix can be computed using R-Solver and SageMath. The process of obtaining the scattering matrix from the circuit is explained in this Medium article by Jatin Chowdhury [27].

This first model, that uses R-Type adaptors, had the same response as the final version but worse performance. The performance of both models is compared in section 4.2.

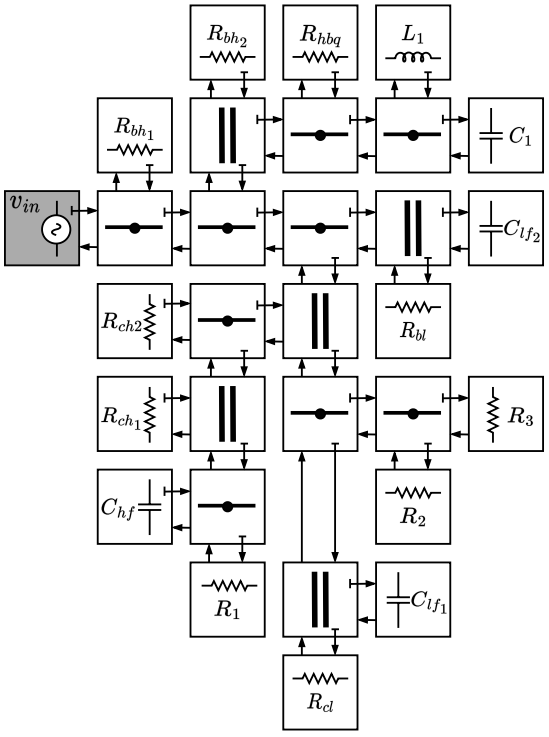


Figure 12: WDF diagram of the passive equalizer stage.

The final WDF implementation of the circuit followed the circuit on figure 3.8(b)

resulting in the WDF structure represented in figure 12. The voltage source is connected at the root of the circuit and all of the components are connected through series and parallel adaptors to it.

Using the `pywdf` model and the functions included in the library we can easily obtain the frequency response of the circuit. The frequency responses of the circuit for the same configurations as in the graphs computed in LTspice and the ones measured from the hardware unit were computed at a sampling rate of 192 kHz and are included in figures 13, 14 and 15.

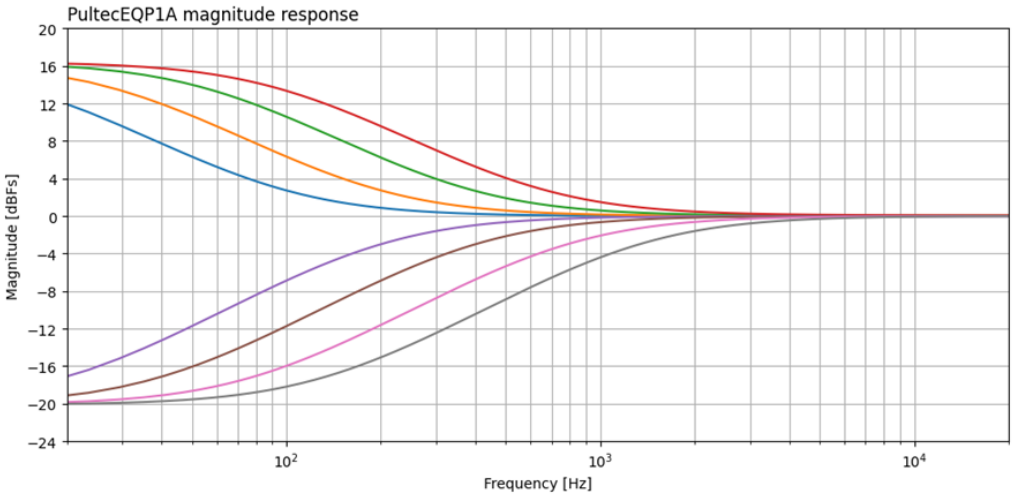


Figure 13: pywdf. Low frequency filters maximum boost and cut.

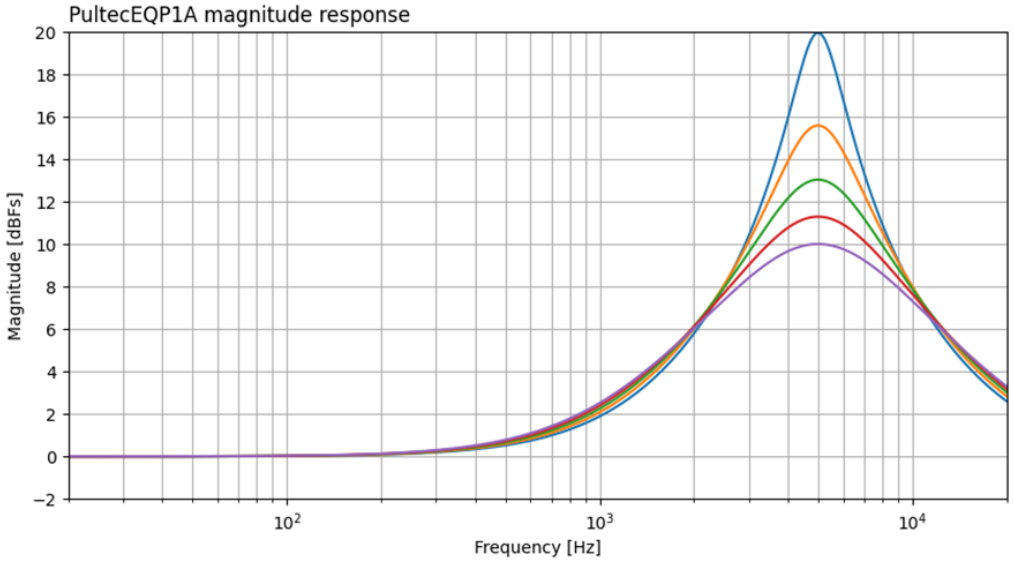


Figure 14: pywdf. Maximum boost with different bandwidths at 5kHz.

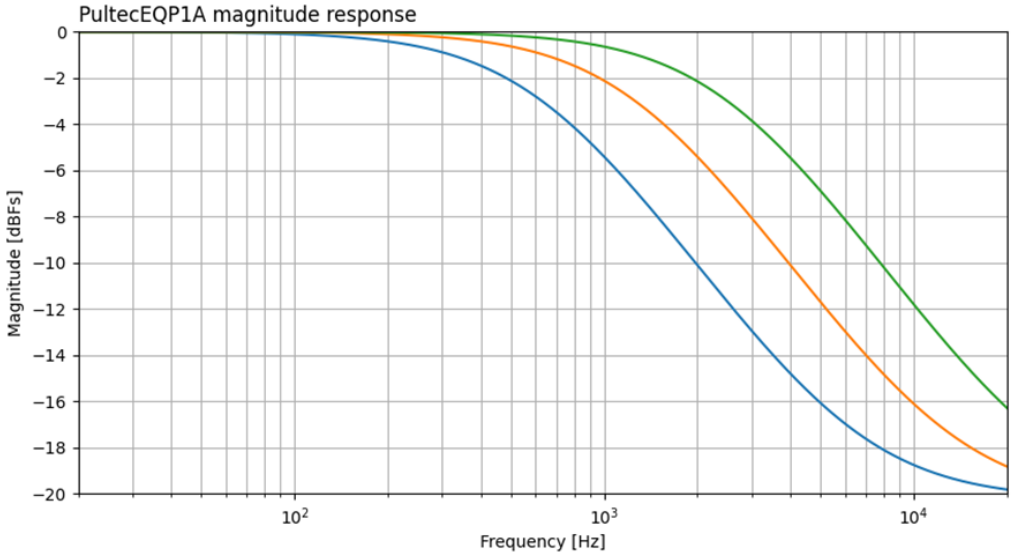


Figure 15: pywdf. Maximum attenuation for the high shelving filter.

While the frequency responses obtained are very accurate at high sampling rates (error measurements are included in appendix B) the Python models are slow and only able to process audio offline.

3.4 Real-time VST3 Plug-in

After achieving accurate results of the WDF model in Python, it is necessary to translate the WDF model to C++ to enable real-time operation. To achieve this, `chowdsp_wdf` library [1][28] was used for the WDF modeling and the JUCE framework[2] was used to compile the model into a VST3 plug-in.

One big advantage of using `pywdf` library for prototyping in Python with `chowdsp_wdf` library for the real-time implementation is that they share the same foundation: all the processes are equivalent and numerically they shed the same results. This ensures that a model prototyped in Python can be seamlessly ported to C++ without any alterations, preserving identically the model's behavior.

Getting good accuracy out of the WDF model (as quantified in section 4.1) requires the use of high sampling rates above the conventional 48 kHz typically used in audio

[1] `Chowdhury-DSP/chowdsp_wdf: https://github.com/Chowdhury-DSP/chowdsp_wdf`
[2] `JUCE framework: https://juce.com`

production. However, running an entire project at high sampling rates is not always efficient. To address this, our plug-in performs a series of operations: it internally up-samples the input signal, processes the over-sampled signal using the WDF model, and then down-samples the processed signal back to the original host sampling frequency. To enable oversampling, we used the `JUCE::dsp::Oversampling` class [3], and provide options for x2, x4, or no oversampling.

Regarding the user interface, the background and the GUI elements (knobs, selectors and switches) were designed and rendered individually. Then, they were integrated into the plug-in, and their behavior was programmed in JUCE. Figure 16 shows the GUI of the plug-in running on Cubase 11.

Figure 16: Graphical user interface of the VST3 plug-in.

The result of this process is a fully operational audio plug-in that effectively incorporates the WDF model within a real-time environment. It is compiled into VST3 format for integration into any Digital Audio Workstation (DAW) and it is available to download on GitHub[4].

[3] `JUCE dsp::Oversampling Class Reference: https://docs.juce.com/master/classdsp_1_1Oversampling.html`

[4] `ABSounds/EQP-WDF-1A: https://github.com/ABSounds/EQP-WDF-1A`

Chapter 4

Results

4.1 Frequency Response

The frequency response curves computed from the Python model are compared against the ones obtained from the LTspice simulations and the error between the two is calculated.

For each graph, both the frequency response obtained from LTspice and the one computed from `pywdf` are displayed. The background color indicates the squared error between the two, $(|H_{LTspice}(f)|-|H_{pywdf}(f)|)^2$, and the $Error_{RMS}$, that follows the definition of equation 4.1, is included at the bottom left corner.

$$Error_{RMS} = \sqrt{\frac{1}{N}\sum_{k}(|H_{LTspice}(k\Delta f)| - |H_{pywdf}(k\Delta f)|)^2} \tag{4.1}$$

The error graphs for a typical EQP-1A configuration are included in figures 17, 18 and 19 for sampling frequencies of 48 kHz, 96 kHz and 192 kHz respectively. The error graphs for all of the individual filters at the the most adverse configurations are included in appendix B and the numerical results are gathered in table 3.

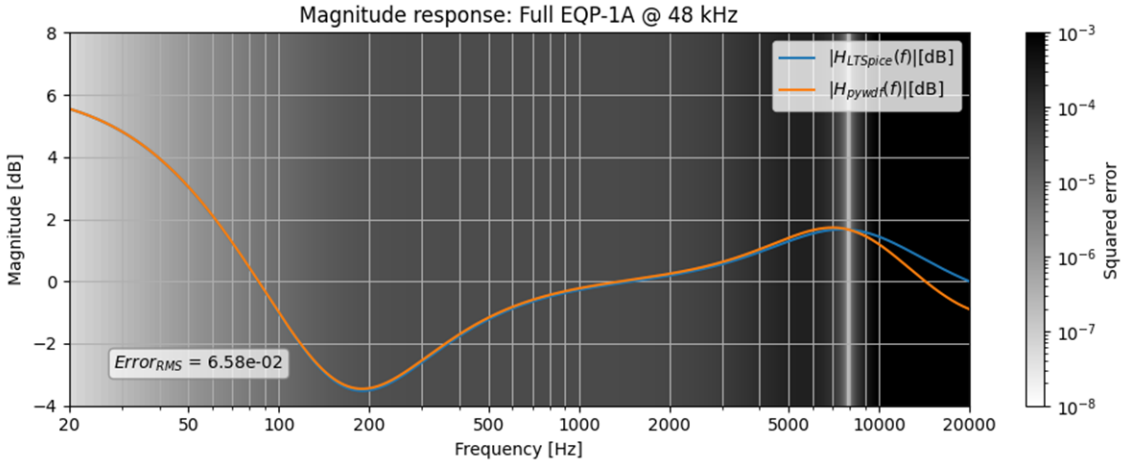


Figure 17: Example setup frequency response error at 48 kHz.

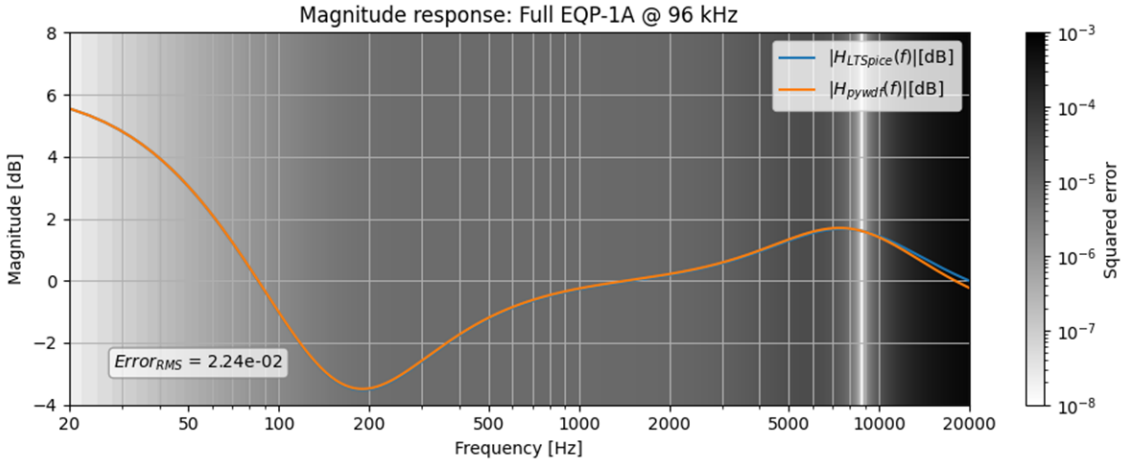


Figure 18: Example setup frequency response error at 96 kHz.

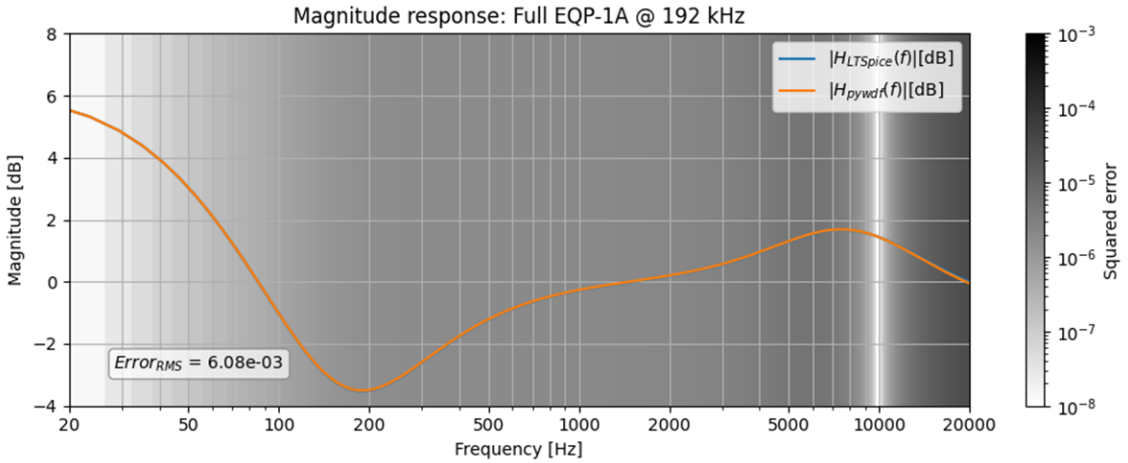


Figure 19: Example setup frequency response error at 192 kHz.

Table 3: Frequency response error measured on each filter.

	Low Boost		**Low Cut**		**High Boost**		**High Cut**	
fs (kHz)	**Fig.**	**RMSE**	**Fig.**	**RMSE**	**Fig.**	**RMSE**	**Fig.**	**RMSE**
48	26	3.65e-2	29	2.07e-3	32	1.40	35	3.60e-2
96	27	1.86e-2	30	5.43e-4	33	6.44e-1	36	1.28e-2
192	28	9.37e-3	31	1.39e-4	34	2.20e-1	37	3.96e-3

4.2 Performance

To assess the performance of the different models, a random audio spanning 20 seconds is generated and processed using each model. The time taken to process all samples is measured using Python's `time.monotonic()` function from the standard library `time`.

The Python implementations are executed directly on a Jupyter Notebook, while the VST plug-ins are loaded into Python using Spotify's Pedalboard library [1] . All the processing is executed locally on a computer with an i7-8550U processor running at 1.8 GHz and 16 GB of RAM operating on Windows 11.

The evaluation includes the processing time for the 20 seconds of audio, as well as a real-time ratio. This ratio indicates how many times faster than real-time the model can operate. For instance, a ratio of 2 implies that the model would be able to process 2 seconds of audio in 1 second. Any ratio below 1 means that the model is not suitable for real-time use.

Table 4: Time taken to process 20 s of audio with each of the Python models.

	pywdf		**pywdf w/ R-Type**	
fs (kHz)	**Time (s)**	**Ratio**	**Time (s)**	**Ratio**
48	**26.969**	**0.74**	120.775	0.17
96	**53.931**	**0.37**	229.953	0.09
192	**108.984**	**0.18**	451.212	0.04

[1] `Spotify/pedalboard: https://github.com/spotify/pedalboard`

Table 4 includes the comparison between the performance of our two Python models developed using the `pywdf` library. The first one uses traditional WDF with the modified version of the circuit, while the other one uses R-Type adaptors.

The results show that the implementation that used R-Type adaptors was 75% slower than the final one. Working at typical audio rates (48, 96 and 192 kHz), none of them was able to run in real-time. This was expected and caused by the nature of the Python programming language.

Table 5 includes the performance assessment of two versions of our VST plug-ins (EQP-WDF-1A) with and without internal oversampling and compares it to another implementation of the EQP-1A described on this paper [29] developed in the Faust programming language that uses R-Type adaptors.

Table 5: Time taken to process 20 s of audio with each of the VST plug-ins.

	EQP-WDF-1A w/o OS		**EQP-WDF-1A w/ x2 OS**		**Faust EQP-1A (R-Type)**	
fs (kHz)	**Time (s)**	**Ratio**	**Time (s)**	**Ratio**	**Time (s)**	**Ratio**
48	**0.092**	**217**	0.099	202	0.175	114
96	**0.182**	**110**	0.195	102	0.326	61
192	**0.360**	**56**	0.379	53	0.638	31

The results in the second column, corresponding to the plug-in with x2 internal oversampling, are measured with the plug-in working internally at the sampling frequency stated in the row's header. It is slightly slower than the other one because of the up-sampling and down-sampling processes that need to be performed before and after processing the signal through the WDF model.

Overall, our plug-in performed between 40% to 47% faster than the Faust implementation on every configuration.

Chapter 5

Conclusions

5.1 Discussion

The primary goal of this project was to develop a functional WDF model of the Pultec EQP-1A equalizer stage. As an additional objective, we aimed to adapt the model for real-time use. We successfully achieved both goals by developing the model and then wrapping it in a VST3 plug-in. The plug-in can operate in real-time within any DAW and closely matches the frequency response of the original unit and will be freely available.

A first evaluation of the model involved comparing the LTspice simulations of the circuit against the WDF model. We accomplished good accuracy at high sampling rates of 96 and 192 kHz, which conditioned the introduction of oversampling capabilities into the VST plug-in.

As part of our evaluation, we also measured the model's performance and compared our real-time model with the only other available open-source implementation of the Pultec EQP-1A, that was developed using the Faust programming language. The results of this comparison show that our model outperforms the Faust-based implementation in terms of processing time. However, both models were able to handle real-time audio processing without any issues.

Upon inspection of the frequency response of the Faust implementation, some discrepancies were found with the references taken for our model (measurements of the 2019 EQP-1A reissue and graphs from the original user's manual). These observations lead us to consider our model as a more faithful representation of the original unit's equalizer stage behavior.

This project was a first step in the direction to a complete open-source model of the Pultec EQP-1A. Although the frequency response achieved is faithful to the original unit, other sections of the circuit would need to be modeled to recreate its harmonic properties and the distortion it introduces into the processed signal. The next steps necessary to model the gain stage of the EQP-1A are detailed in the next section.

5.2 Future Work

Having modeled the equalizer stage of the EQP-1A, the other section of the unit that has the most impact on the character of the processed sound is the vacuum tube gain stage. Some efforts were put towards modeling the gain stage for this thesis but it finally was not possible to get any relevant results in time for its delivery.

Multiple approaches could be taken to model the distortion caused by the 12AX7 and 12AU7 vacuum tubes gain stage. A review of different methods to model vacuum tube amplifiers distortion can be found in this article [30] and this talk from 2017's Audio Developer Conference [31].

Probably the easiest approach to model it would be to use a filtering block followed by a static waveshaper as proposed on this article [32] or in the ADC'17 talk [31]. Although this method is fairly easy to implement, it does not account for any non-symmetrical distortion caused to the waveform and will result in bigger errors when applied to circuits with pairs of triodes, like the circuit of the gain stage of EQP-1A, that usually produce this kind of distortion.

A more accurate option, but also a more involved one, would be to include models of the vacuum tubes into a WDF implementation of the gain stage circuit. This has been proposed in these articles [33, 34] and would give a very accurate representation

of the circuit but due to the complexity of the circuit of the gain stage of the EQP-1A might not be easy to accomplish in the WDF domain.

Instead of the WDF models of the vacuum tubes, neural networks can be introduced into the WDF domain model as proposed in [35]. This has shown great results in modeling guitar amplifiers and would likely work as well for the gain stage of the EQP-1A but it would not reduce the complexity of the WDF model so the issue to implement the circuit would remain the same as with the WDF model of the vacuum tubes.

Apart from the efforts to model the gain stage of the unit, some contributions to the `pywdf` library are planned. The methods used for comparing the LTspice simulations to the WDF implementation in Python can be introduced to the library as well as improving the plot functions included in the library by making them a bit more flexible and clear. The inclusion of the model of the EQP-1A as an example in the library is also a possibility.

List of Figures

List of Tables

Bibliography

[1] Demystifying the 'magic' of the Pultec EQ. URL https://abbeyroadinstitute.nl/blog/demystifying-the-pultec/.

[2] Nonlinear system (2023). URL https://en.wikipedia.org/w/index.php?title=Nonlinear_system. Page Version ID: 1144950707.

[3] Anthon, G. Evaluation of Nonlinearities in a Diode Clipper Circuit based on Wave Digital Filters (2022). URL https://zenodo.org/record/7116075.

[4] Chowdhury, J. A Comparison of Virtual Analog Modelling Techniques for Desktop and Embedded Implementations (2020). URL http://arxiv.org/abs/2009.02833. ArXiv:2009.02833 [cs, eess].

[5] Ho, C.-W., Ruehli, A. & Brennan, P. The modified nodal approach to network analysis. *IEEE Transactions on Circuits and Systems* **22**, 504–509 (1975). Conference Name: IEEE Transactions on Circuits and Systems.

[6] van der Schaft, A. Port-Hamiltonian systems: an introductory survey. In Sanz-Solé, M., Soria, J., Varona, J. L. & Verdera, J. (eds.) *Proceedings of the International Congress of Mathematicians Madrid, August 22–30, 2006*, 1339–1365 (European Mathematical Society Publishing House, Zuerich, Switzerland, 2007). URL https://ems.press/doi/10.4171/022-3/65.

[7] Gernandt, H., Haller, F. & van der Schaft, T. R. A. Port-Hamiltonian formulation of nonlinear electrical circuits. *Journal of Geometry and Physics* **159**, 103959 (2021). URL http://arxiv.org/abs/2004.10821. ArXiv:2004.10821 [math].

[8] State-space representation (2023). URL `https://en.wikipedia.org/w/index.php?title=State-space_representation`. Page Version ID: 1144232598.

[9] Holters, M. & Zölzer, U. A generalized method for the derivation of non-linear state-space models from circuit schematics. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, 1073–1077 (2015). ISSN: 2076-1465.

[10] O. Smith, J. Wave Digital Filters. URL `https://ccrma.stanford.edu/~jos/pasp/Wave_Digital_Filters_I.html`.

[11] Rollo, C. Black-box modelling of nonlinear audio circuits (2018). URL `https://aaltodoc.aalto.fi/handle/123456789/30521`.

[12] Chirp (2023). URL `https://en.wikipedia.org/w/index.php?title=Chirp&oldid=1141754848`. Page Version ID: 1141754848.

[13] Wright, A., Damskägg, E.-P. & Välimäki, V. Real-Time Black-Box Modelling with Recurrent Neural Networks (2019).

[14] Wright, A., Damskägg, E.-P., Juvela, L. & Välimäki, V. Real-Time Guitar Amplifier Emulation with Deep Learning. *Applied Sciences* **10**, 766 (2020). URL `https://www.mdpi.com/2076-3417/10/3/766`. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[15] Fettweis, A. Wave digital filters: Theory and practice. *Proceedings of the IEEE* **74**, 270–327 (1986). Conference Name: Proceedings of the IEEE.

[16] multivac61. multivac61/wave_digital_notebook (2023). URL `https://github.com/multivac61/wave_digital_notebook`. Original-date: 2016-10-23.

[17] Werner, K. J. *Virtual Analog Modeling of Audio Circuitry Using Wave Digital Filters*. Ph.D. thesis, Stanford University, Stanford (2016). Pages: 261 Volume: Ph.D. in Computer-Based Music Theory and Acoustics.

[18] O. Smith, J. MUS420 Lecture: Wave Digital Filters (2022). URL https://ccrma.stanford.edu/~jos/WaveDigitalFilters/WaveDigitalFilters_4up.pdf.

[19] Chowdhury, J. R-Solver (2023). URL https://github.com/jatinchowdhury18/R-Solver. Original-date: 2021-12-13T18:52:19Z.

[20] Werner, K. J., Nangia, V., Iii, J. O. S. & Abel, J. S. Resolving Wave Digital Filters with Multiple/Multiport Nonlinearities (2015).

[21] Dave Library. URL http://www.davegroupjapan.com/ekoukoku9.html.

[22] Gyraf Audio. The G-Pultec. URL https://www.gyraf.dk/gy_pd/pultec/pultec.htm.

[23] Pulse Techniques. Pultec EQP-1A User Manual. URL https://www.thehistoryofrecording.com/Manuals/Pultec/Pultec_EQP-1A_Manual.pdf.

[24] Robjohns, H. Pulse Techniques EQP-1A. *Sound on Sound* (2019). URL https://www.soundonsound.com/reviews/pulse-techniques-eqp-1a.

[25] Nagel, L. W. & Pederson, D. Spice (simulation program with integrated circuit emphasis). Tech. Rep. UCB/ERL M382, EECS Department, University of California, Berkeley (1973). URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html.

[26] LTspice Information Center | Analog Devices. URL https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html.

[27] Chowdhury, J. Wave Digital Circuit Models with R-Type Adaptors (2022). URL https://jatinchowdhury18.medium.com/wave-digital-filter-circuit-models-with-r-type-adaptors-39ad0ad658ce.

[28] Chowdhury, J. chowdsp_wdf: An Advanced C++ Library for Wave Digital Circuit Modelling (2022). URL `http://arxiv.org/abs/2210.12554`. ArXiv:2210.12554 [eess].

[29] Roosenburg, D., Stine, E., Michon, R. & Chowdhury, J. A wave digital filter modeling library for the faust programming language. In *Proc. 18th Sound Music Comput. Conf., Virtual*, 24–30 (2021).

[30] Pakarinen, J. & Yeh, D. T. A review of digital techniques for modeling vacuum-tube guitar amplifiers. *Computer Music Journal* **33**, 85–100 (2009).

[31] Cohen, I. Fifty shades of distortion (London, UK, 2017). URL `https://www.youtube.com/watch?v=oIChUOV_0w4`.

[32] Digital Modeling of Guitar Amplifier Preamp Distortion. URL `https://www.ampbooks.com/mobile/dsp/preamp/`.

[33] Karjalainen, M. & Pakarinen, J. Wave digital simulation of a vacuum-tube amplifier. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, V–V (IEEE, 2006).

[34] Pakarinen, J. & Karjalainen, M. Enhanced wave digital triode model for real-time tube amplifier emulation. *IEEE Transactions on Audio, Speech, and Language Processing* **18**, 738–746 (2009).

[35] Darabundit, C. C., Roosenburg, D. & Iii, J. O. S. Neural Net Tube Models for Wave Digital Filters (2022).

Appendix A

References for the Circuit

Included in this appendix you can find all of the documents we used as references to develop the model of the Pultec EQP-1A: schematics of the original unit, similar models and clones and measurements carried out on the hardware unit.

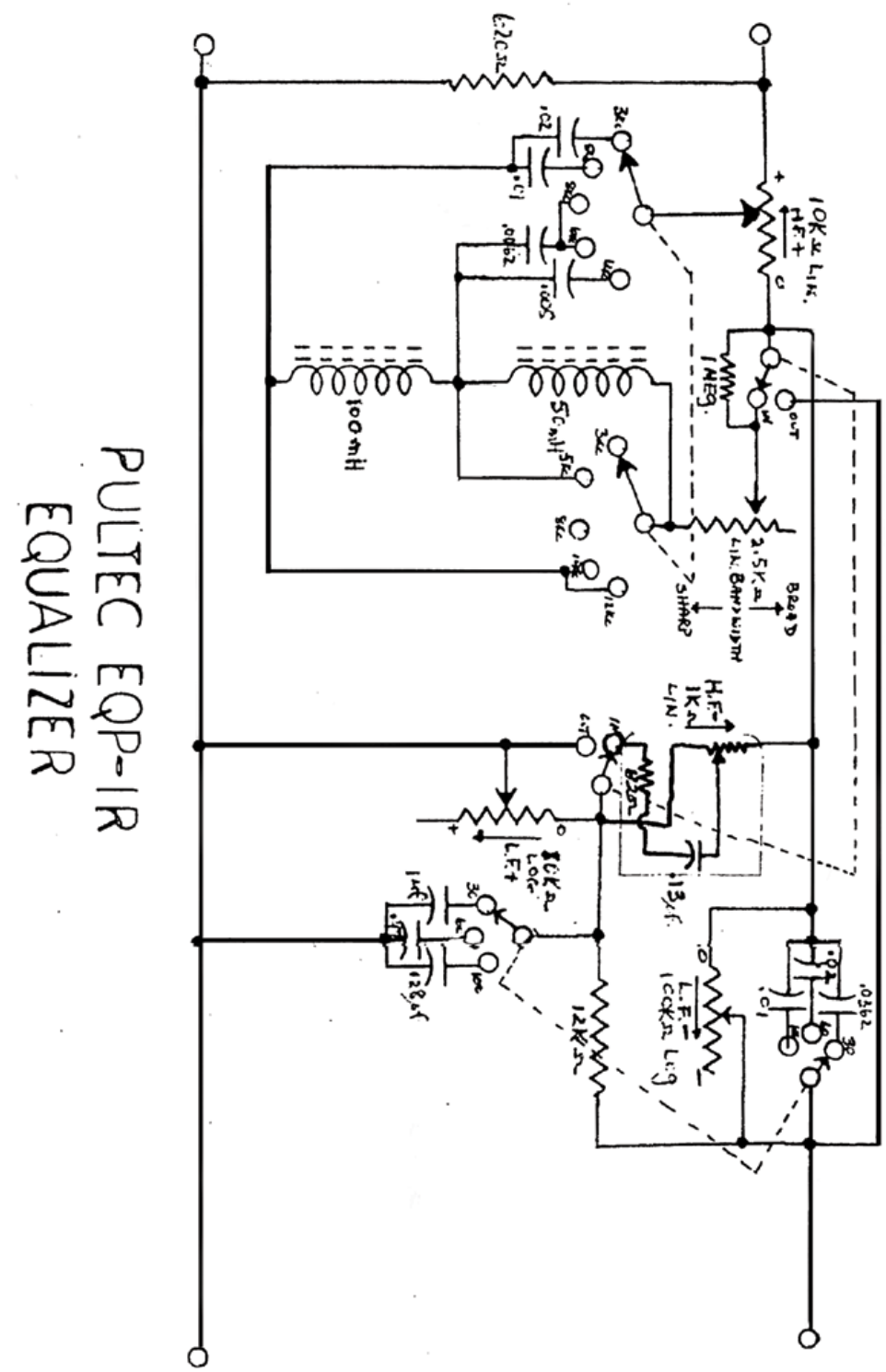


Figure 20: Circuit schematic for the Pultec EQP-1R.

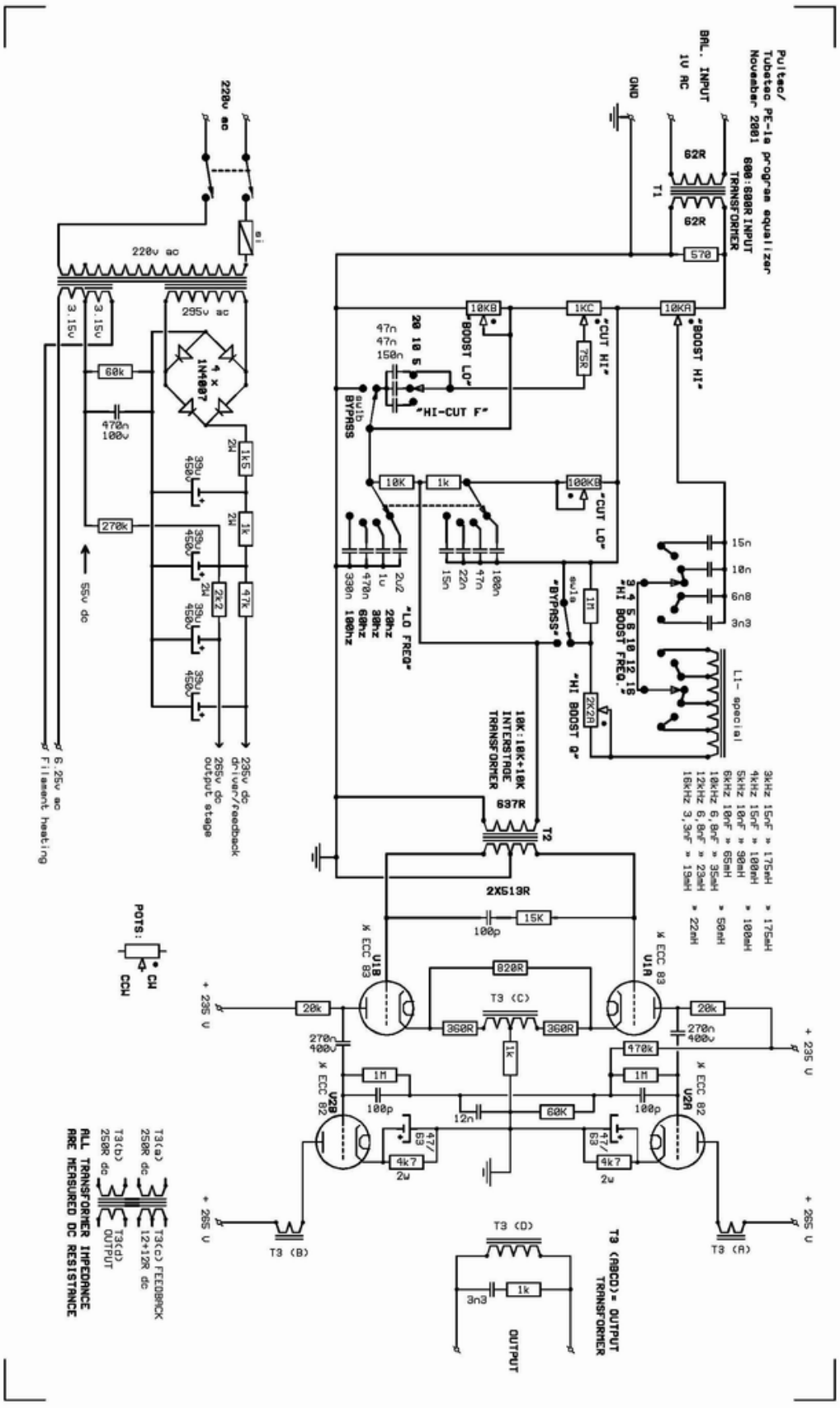


Figure 21: Circuit schematic for the EQP-1A clone by Gyraf Audio.

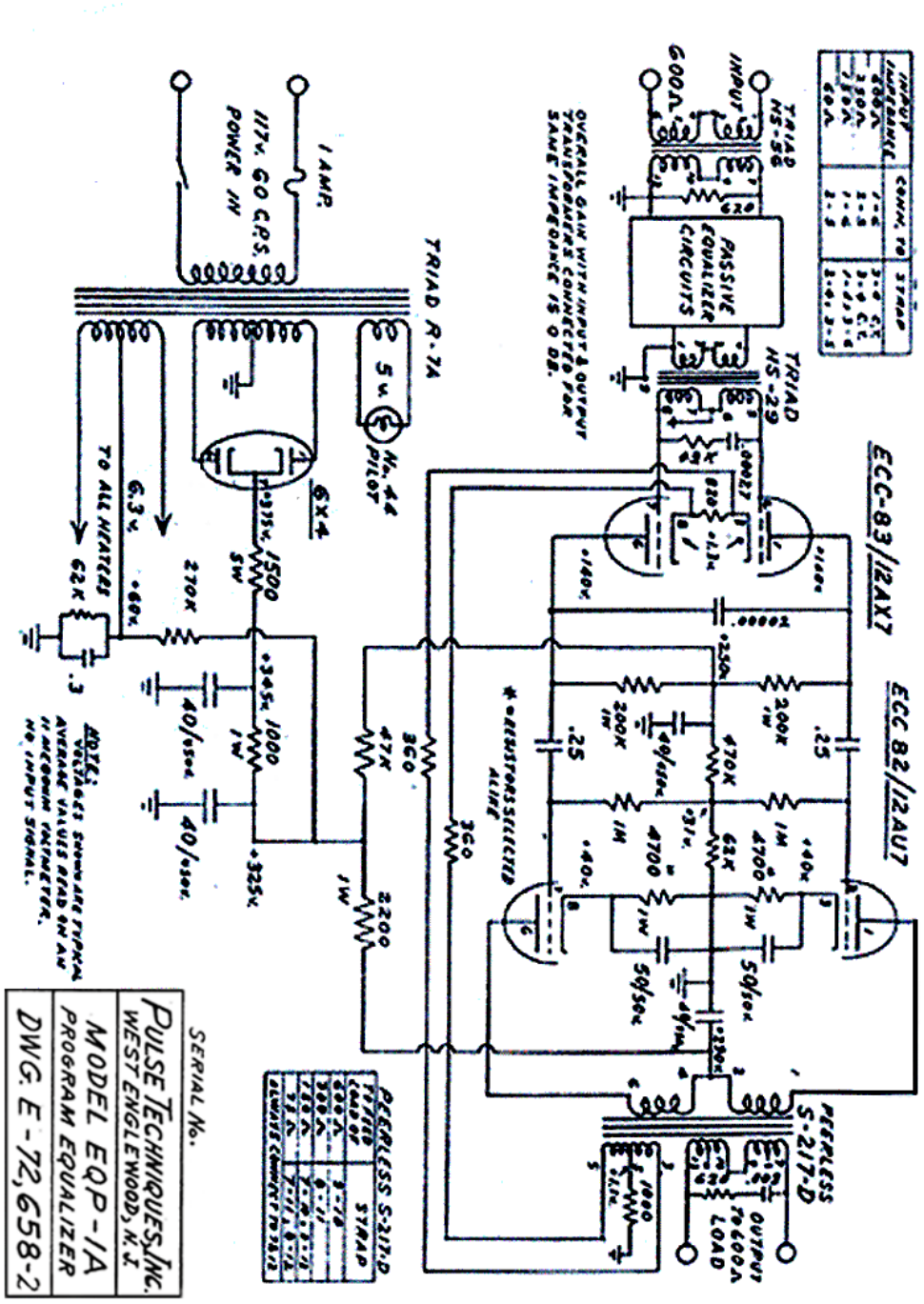


Figure 22: Gain and power stages schematic from the EQP1-A user manual.

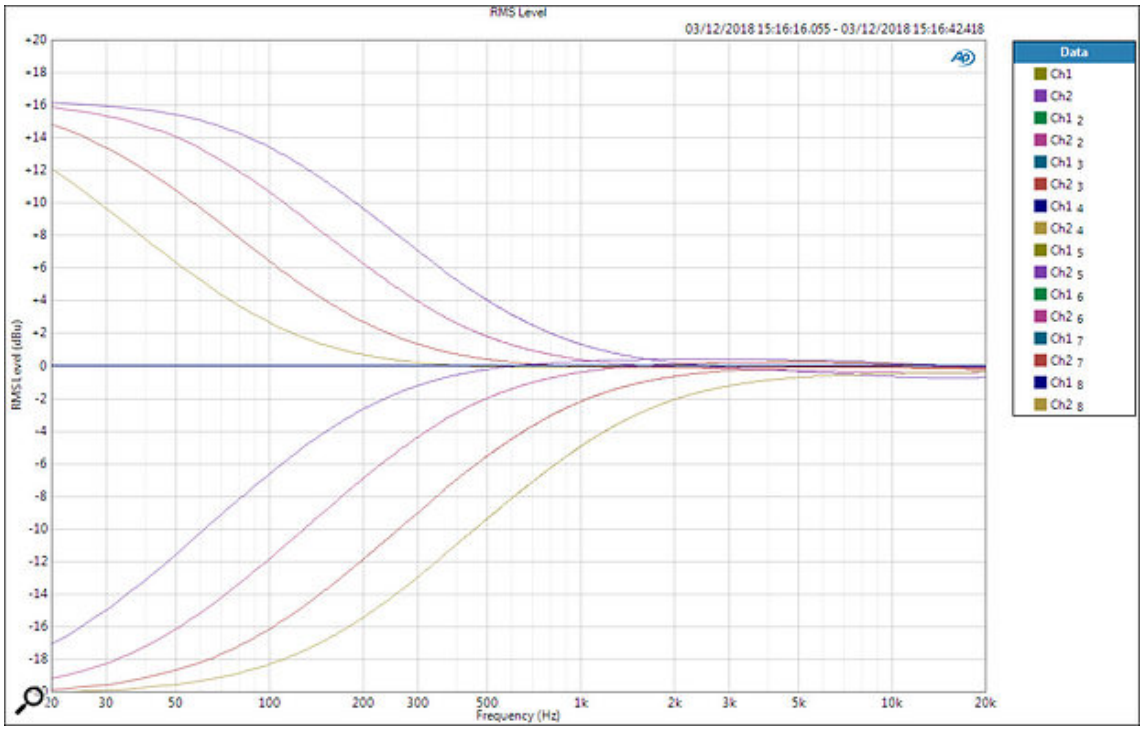


Figure 23: Low frequency filters maximum boost and cut measured on EQP-1A hardware unit.

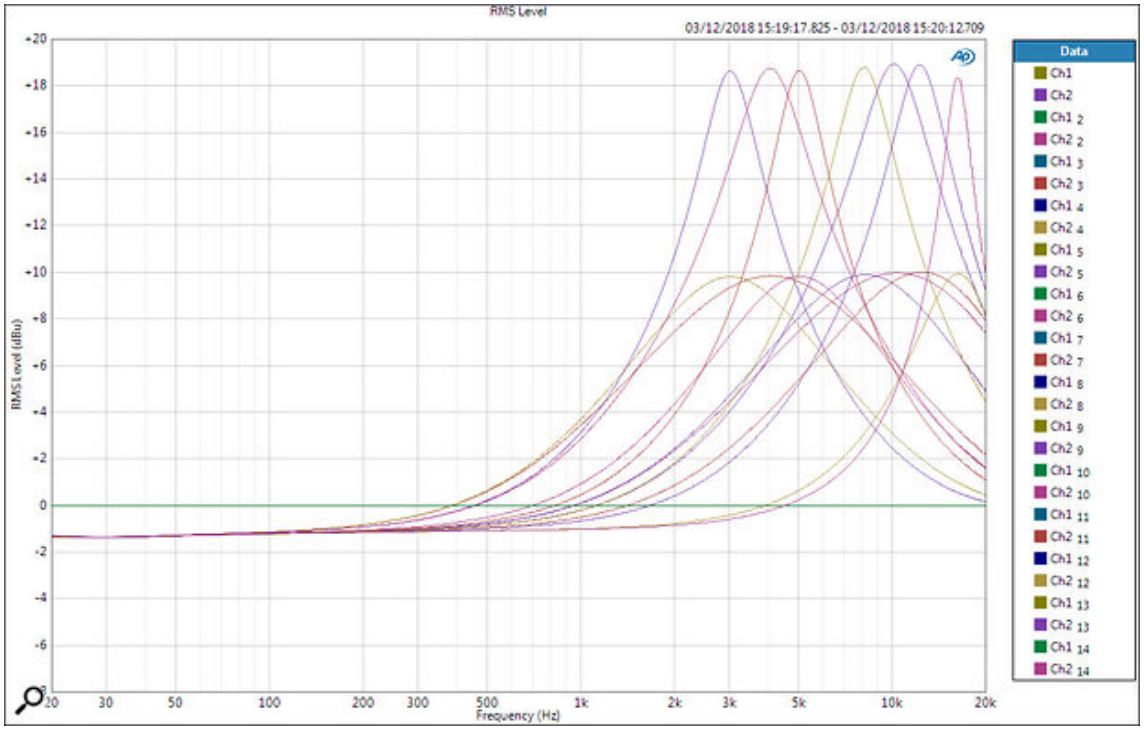


Figure 24: Maximum boost with different bandwidths measured on EQP-1A hardware unit.

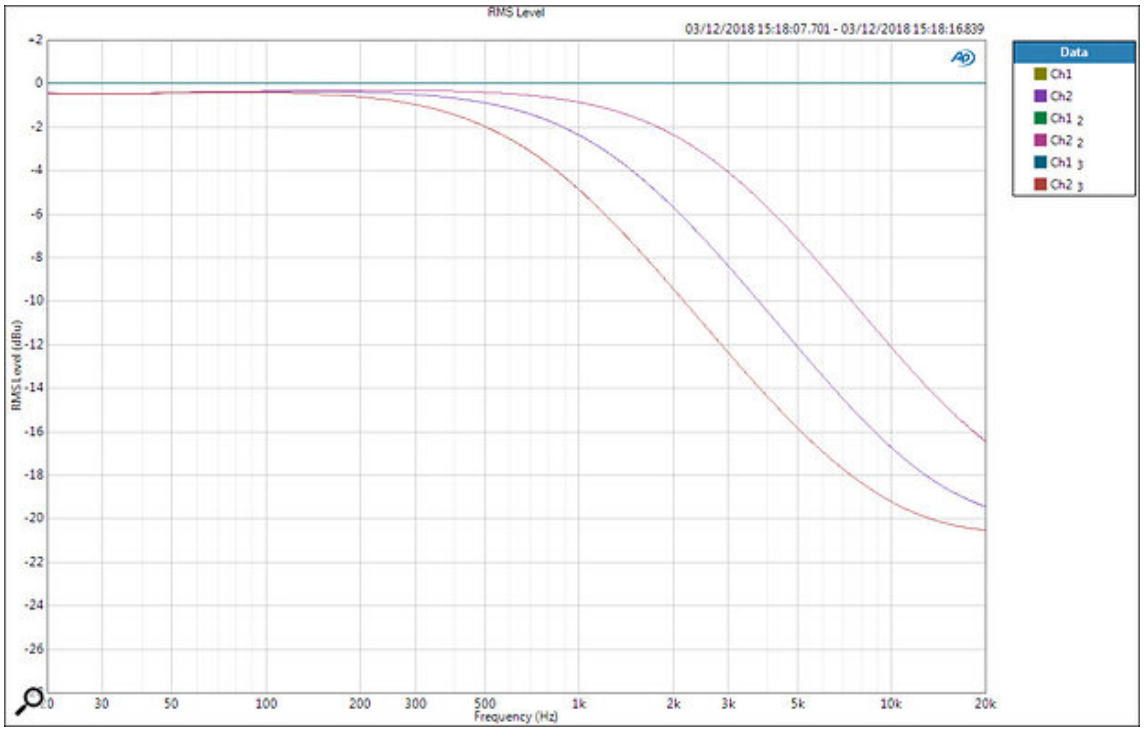


Figure 25: Maximum attenuation for the high shelving filter measured on EQP-1A hardware unit.

Appendix B

Frequency Response Error Graphs

This appendix contains the graphs for the frequency response error between the Python WDF implementation and the simulations conducted on LTspice. These comparisons are done for each of the separate filters within the passive equalizer stage of the EQP-1A. For the frequency response error graph of a typical EQP-1A configuration, refer to section 4.1.

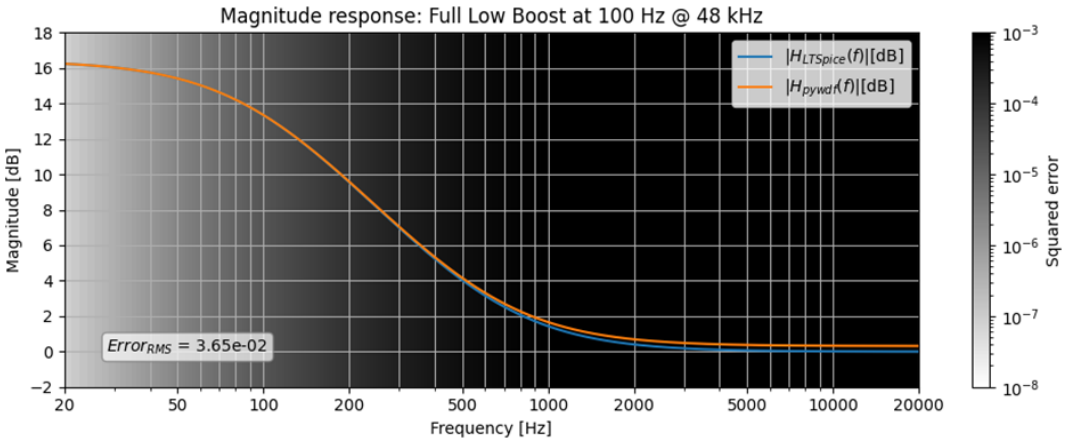


Figure 26: Low frequency boost error at 48 kHz.

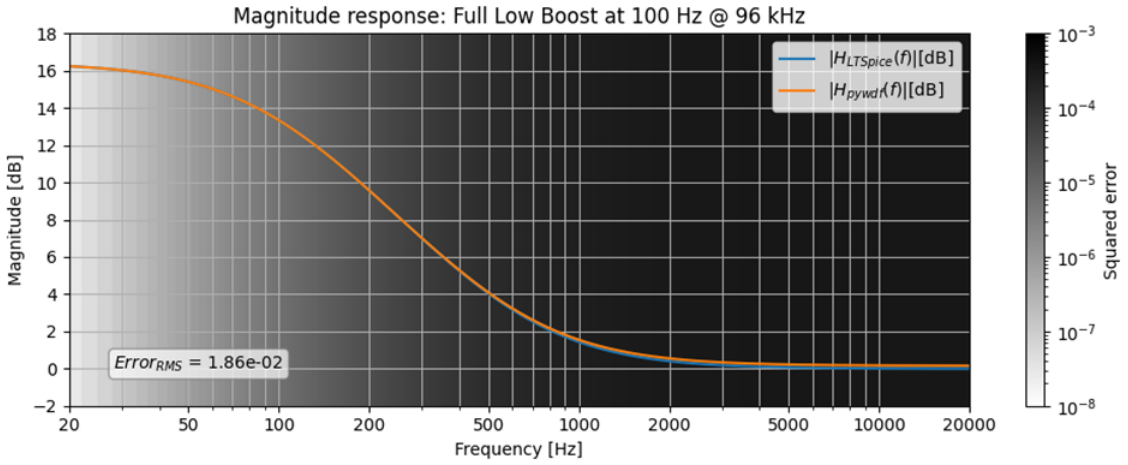


Figure 27: Low frequency boost error at 96 kHz.

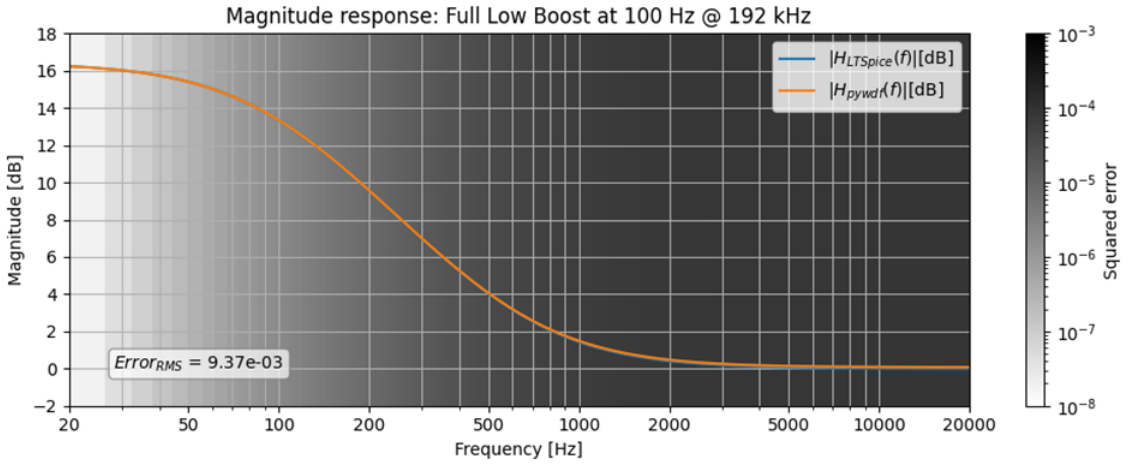


Figure 28: Low frequency boost error at 192 kHz.

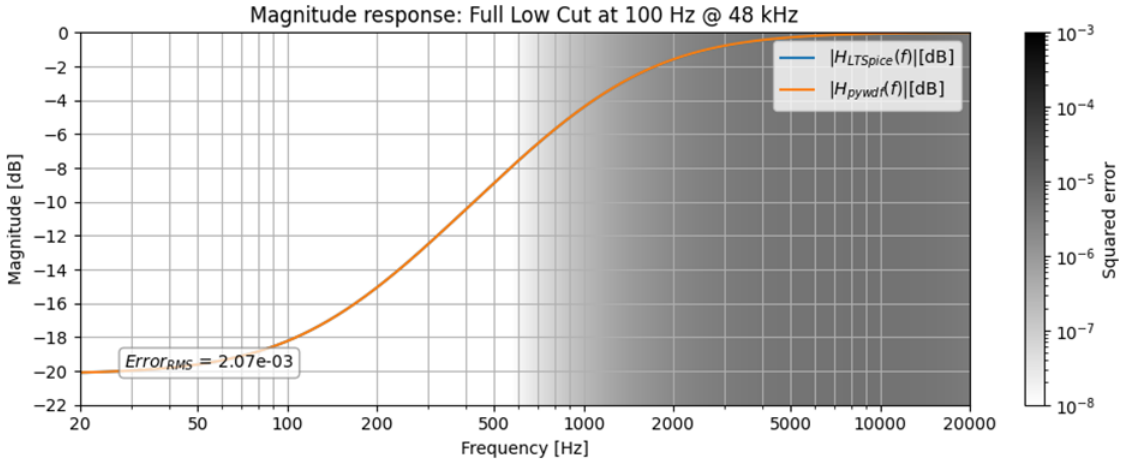


Figure 29: Low frequency attenuation error at 48 kHz.

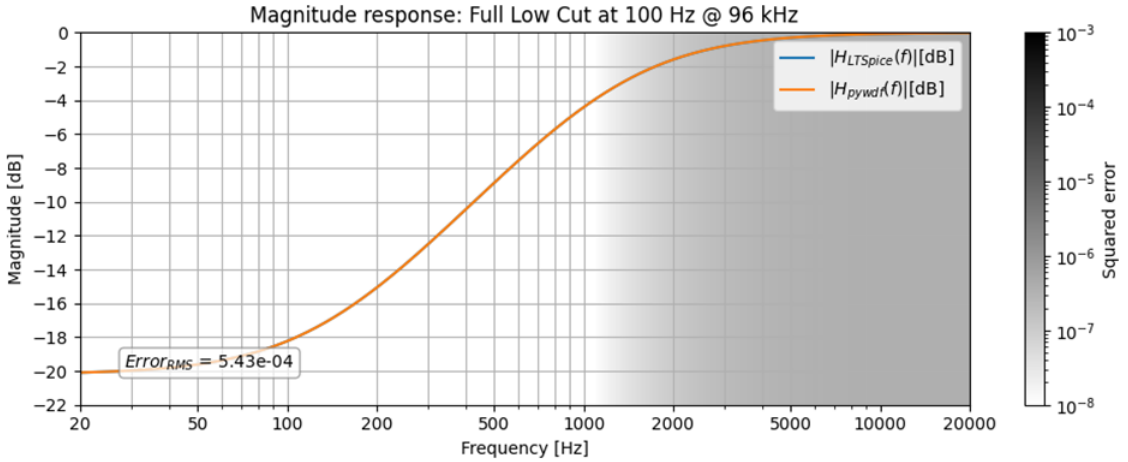


Figure 30: Low frequency attenuation error at 96 kHz.

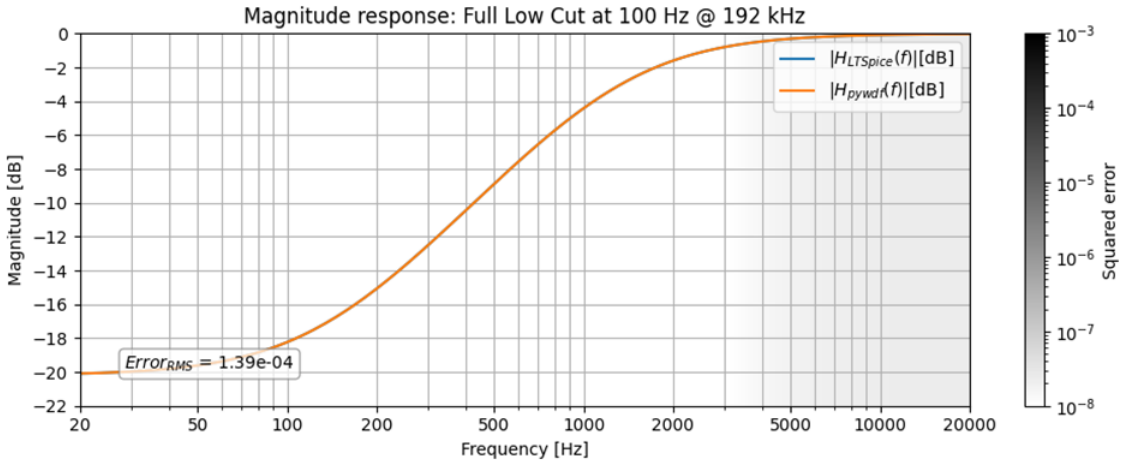


Figure 31: Low frequency attenuation error at 192 kHz.

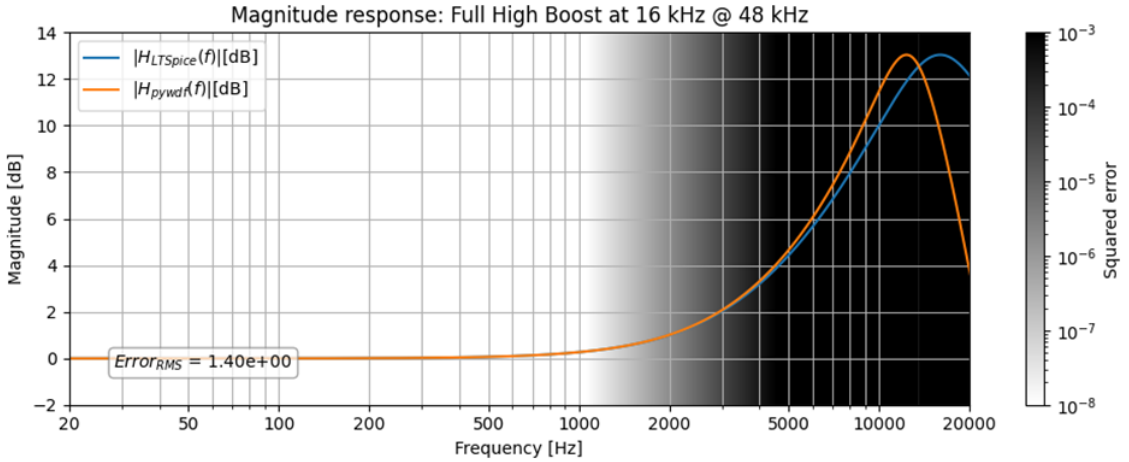


Figure 32: High frequency boost error at 48 kHz.

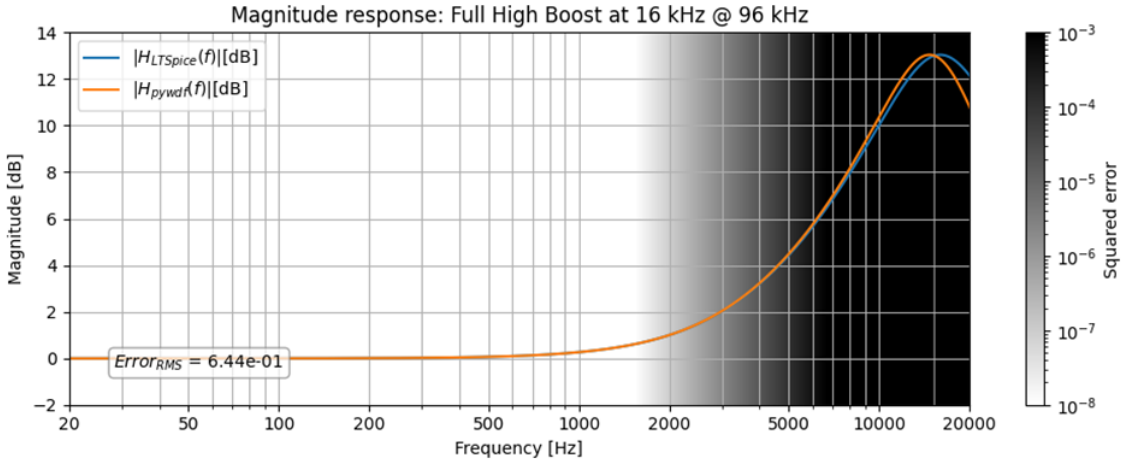


Figure 33: High frequency boost error at 96 kHz.

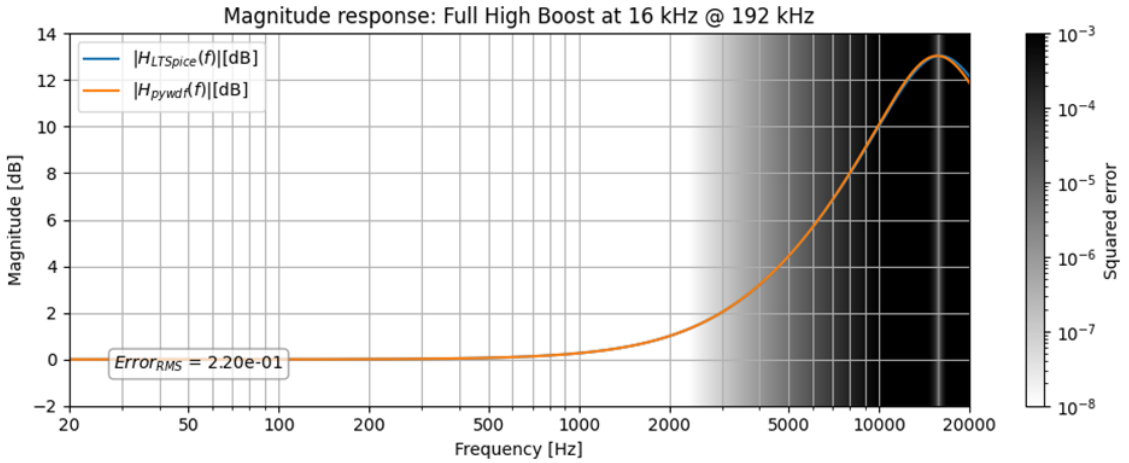


Figure 34: High frequency boost error at 192 kHz.

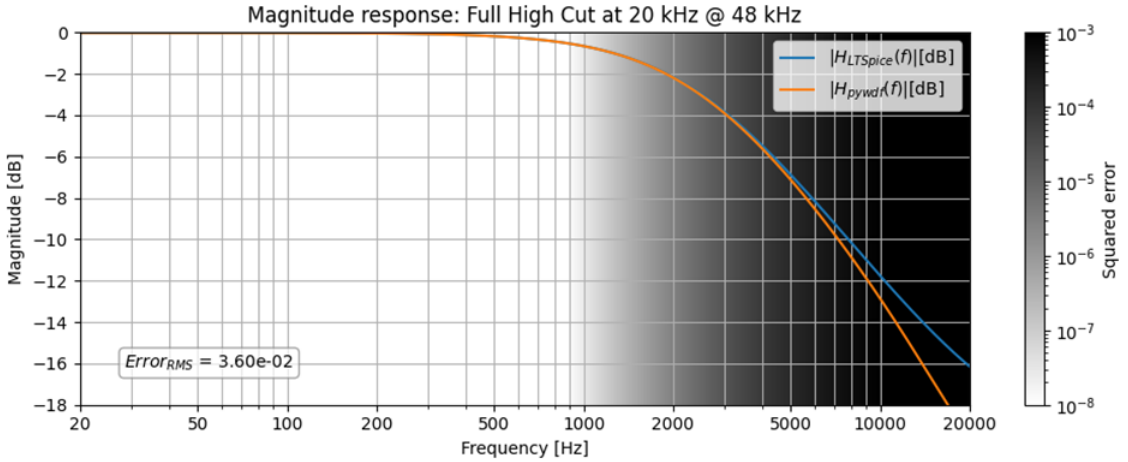


Figure 35: High frequency attenuation error at 48 kHz.

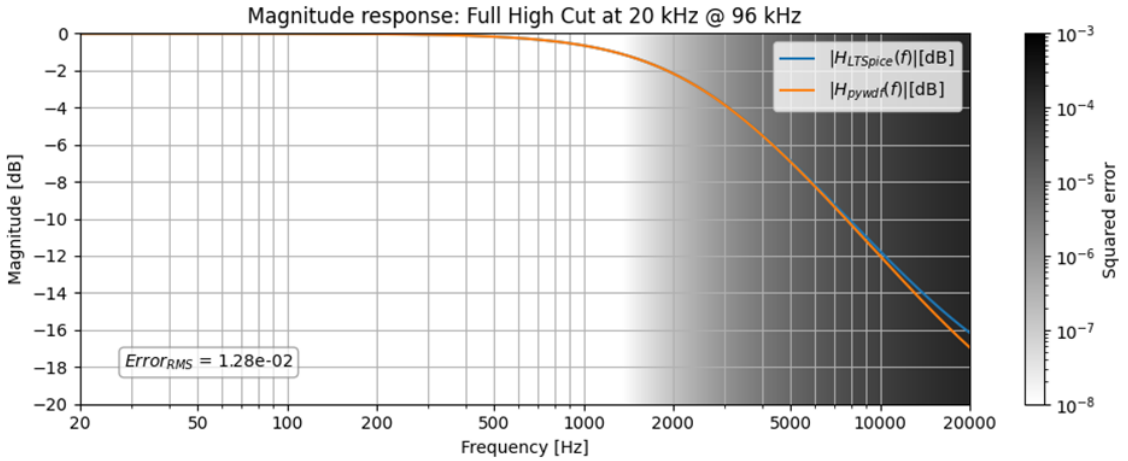


Figure 36: High frequency attenuation error at 96 kHz.

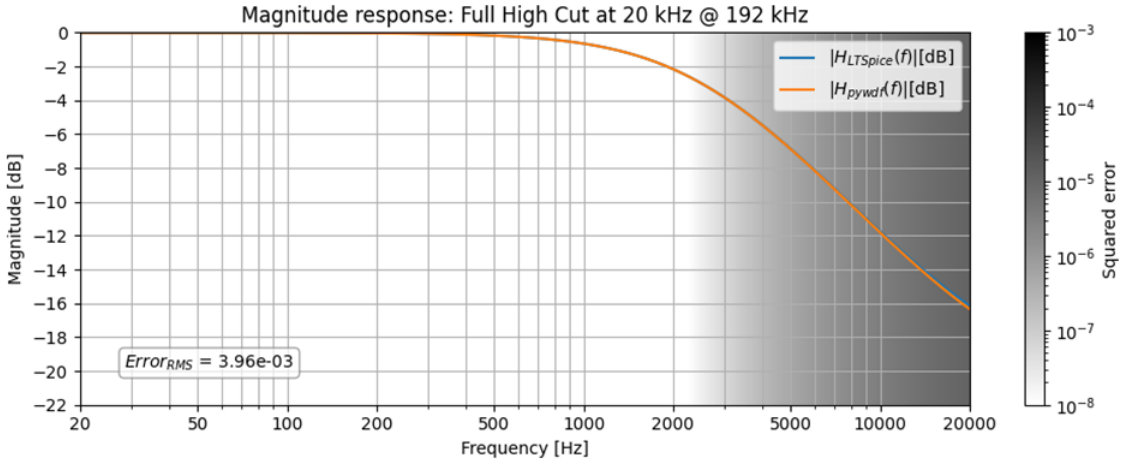


Figure 37: High frequency attenuation error at 192 kHz.