Universitat
Pompeu Fabra
Barcelona

VLA-252A: ANALOG MODELING OF THE LANGEVIN 252A UTILIZING WAVE DIGITAL FILTERS

June, 2023

By:
Andrea Sofia Vallejo Budziszewski,
Chris Morse,
Du Huang,
Pablo Rodríguez,
Polina Zvoda,
Tommaso Settimi

Supervisor: Xavier Lizarraga

INDEX

PROJECT PLAN

1. Overview

Our goal for this project is to create a revolutionary Passive Graphic Equalizer (GEQ) prototype that replicates the iconic Langevin EQ252A, which was one of the earliest and most sought-after analog graphic EQs in history. However, instead of relying on costly physical analogs that can set you back around 1400 GBP (around 1600 EUR or 1750 USD), we are taking a modern approach by utilizing Wave Digital Filters to create a digital model that is both affordable and highly efficient. With this innovative solution, we aim to bring the timeless sound of the Langevin EQ252A to a wider audience, while also revolutionizing the way we approach equalization in the digital age.

To achieve our goal, we will be using the Python programming language and the power of Wave Digital Filters. Wave Digital Filters are a modern digital signal processing technique that allows us to simulate analog circuits in a cost-effective and efficient manner. By utilizing this technique, we can accurately replicate the sound of the Langevin EQ252A and create a prototype that is accessible to a wider audience. Our EQ spectrum coverage is thoughtfully designed with a series of bands at precise fixed frequency values, which not only simplifies our workflow but also ensures precision in our equalization process. By strategically choosing fixed frequencies for each band, we can achieve a seamless and effective EQ curve that meets our specific needs. This approach streamlines our work and allows us to focus on achieving optimal results without getting bogged down in unnecessary complexity.

2. Introduction

Equalizers

An equalizer is an essential tool in audio processing that allows you to adjust the balance between different frequency components in an audio signal. It enables you to tailor the sound to your preferences, by adjusting the bass, midrange, and treble frequencies. There are two main types of equalizers: parametric and graphic. In the following paragraphs, we will provide a basic introduction to both types and delve deeper into them in the State of the Art.

Parametric equalizers offer more precise control over the frequency response of your sound system by boosting or cutting specific frequencies. They are called "parametric" because they allow you to control the parameters of the equalization process, such as center frequency, bandwidth, and gain. This allows you to fine-tune your sound to your liking or adjust it to the acoustic properties of your listening environment.

Graphic equalizers, on the other hand, allow users to adjust multiple frequency ranges at once, usually with a set of sliders or knobs arranged in a graphic pattern, where each slider corresponds to a specific frequency range. This type of equalizer is ideal for tailoring the sound to your preferences or compensating for any deficiencies in the original audio signal. Passive graphic equalizers, which we will be developing in this project, are called "passive" because they do not require external power to operate. They rely on passive components, such as resistors, capacitors, and inductors, to perform the equalization function. By using a combination of filters, including high-pass, low-pass, and band pass, the incoming audio signal is split into multiple frequency bands, and each band is sent to a corresponding slider, where its level can be adjusted by a passive potentiometer. The modified signals are then recombined and sent to the output, resulting in a tailored frequency response.

Filters

A filter is a device or process that allows certain frequencies to pass through while blocking or attenuating other frequencies. Think of it like a strainer that lets certain things through while keeping other things out.

Low-pass filter

This type of filter allows low frequencies to pass through while blocking higher frequencies.

High-pass filter

This type of filter allows high frequencies to pass through while blocking lower frequencies.

Band-pass filter

This type of filter only allows a certain range of frequencies to pass through, while blocking frequencies outside of that range.

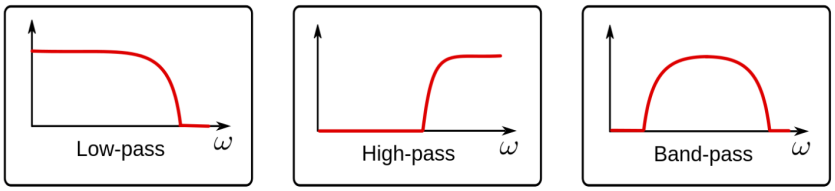


Figure 1: Filters.

Langevin EQ252A

The Langevin EQ252A is a vintage, analog equalizer that was popular in the mid-20th century. It is a white box model, meaning that its internal workings are explicitly represented and understood.

The EQ252A consists of two separate, stereo equalizers that can be used together or independently. Each equalizer has five bands of EQ, with adjustable frequency and gain controls. The EQ bands are centered at 50Hz, 150Hz, 500Hz, 1.5kHz, and 5kHz, with a boost or cut range of +/- 8dB.

The EQ252A uses passive LC filtering, which means that the equalizer circuits use inductors and capacitors to create a frequency-dependent resistance that boosts or cuts certain frequencies. The EQ circuits are designed to have a smooth and musical sound, with a gentle roll-off and a low noise floor.

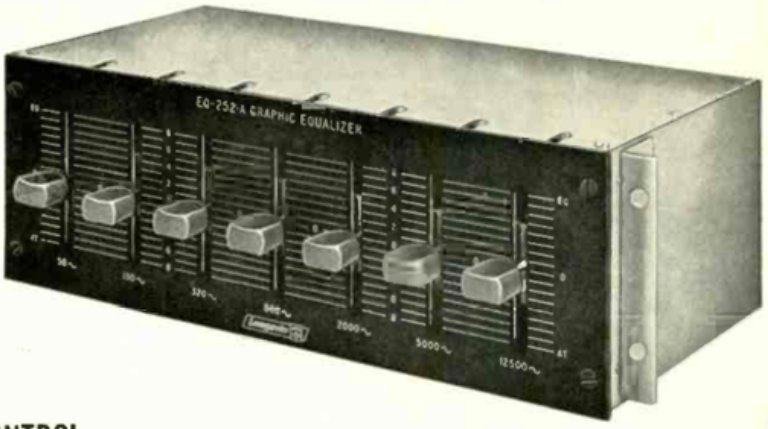
The parameters used in this EQ are the following:

- (a) Band 1: This controls the gain of the 50 Hz band.
- (b) Band 2: This controls the gain of the 130 Hz band.
- (c) Band 3: This controls the gain of the 320 Hz band.
- (d) Band 4: This controls the gain of the 800 Hz band.
- (e) Band 5: This controls the gain of the 2000 Hz band.
- (f) Band 6: This controls the gain of the 5000 Hz band.
- (g) Band 7: This controls the gain of the 12500 Hz band.

Figure 2: Langevin 252A.

Langevin

MODEL EQ-252-A GRAPHIC EQUALIZER



7 POSITIONS FOR ULTIMATE CONTROL OF SPECTRAL QUALITY IN RECORDING, TV-BROADCAST AND MOTION PICTURES

FEATURES

7 Selected Positions of Variable Hi-Lo Equalization and Attenuation.

Gold plated, Noise-free, Switching through ±8 db in 1 db steps during active use.

Hum-free performance through toroid coils from −70 to +24 dbm.

No tubes or power required — all passive Bridge T circuits in one integrated unit.

Small size: 3½" x 10½" x 5¾" deep.

The Langevin Model EQ-252-A Graphic Equalizer fufills the critical need for multiple control at the subjectively important points of the audio range. It employs miniaturized, military quality, gold plated, etched circuitry in each of the 7 plug-in filter units, resulting in a passive assembly requiring no tubes or power supplies. Only input and output connections are required. Sliding Levers permit 8 db of equalization and 8 db of attenuation in 1 db steps at 50, 130, 320, 800, 2000, 5000 and 12,500 cps during the program through noise-free gold-plated switching. Modern controls give quiet operation at −70 up to +24 dbm.

Filter assemblies use sealed toroid coils for hum-free operation. Careful design delivers ±½ db accuracy. Overlap from one filter to the next gives combined flat output when levers are in a straight line in any equalized or attenuated position (see curves). Special frequencies are available to order; overlap may or may not provide combined flat output between adjacent positions as the standard frequencies shown have been calculated for this effect. In zero position each or all filters are flat (resistive only, 16 db loss) from input to output. Because all passive circuitry is used there is no distortion when operated up to plus 24 dbm. Impedance is 600 ohms in and out; for other impedances use Langevin line to line transformers, Model TF-602-C. The model EQ-252-A is limited to 600 ohms impedance for the reason that lower impedances would double the size of the equalizer components every time the impedance is halved.

SPECIFICATIONS

Circuit: Bridged T; **Impedance:** 600/600 ohms; **Insertion Loss:** 16 db; **Operating Level:** −70 to +24 dbm; **Positions:** 7, with 8 db of equalization and 8 db of attenuation at 50, 130, 320, 800, 2000, 5000 and 12,500 cps in 1 db steps; **Distortion:** none; **Coils:** Sealed toroids; **Power Requirements:** none; **Response:** See curves; **Panel Finish:** Black, satin finish, non-halation, anodized aluminum; **Terminals:** solder type, turret; **Filter Sections:** 7 plug-in, printed circuit type; **Size:** 3½" high by 10½" long by 5¾" deep overall.

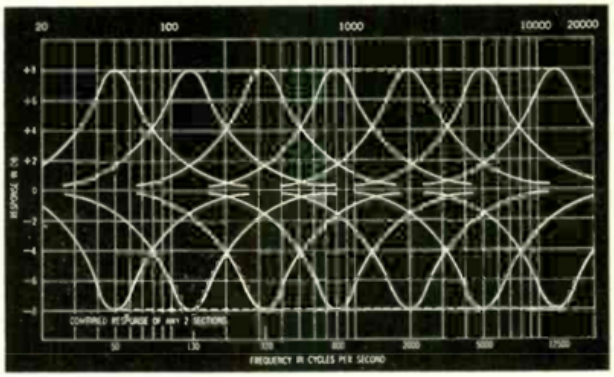
ORDERING INFORMATION

Model EQ-252-A Graphic Equalizer equipped with red knobs, complete with mounting hardware and instructions. Weight, net 9 lbs.; 14 lbs. shipping. Price, Net $475.00.

Recommended Accessories

When lower impedances than 600 ohms are required, use the following matching coils in and out:

Model TF-602-C Line to Line Transformer, Weight, Net, 2¾ lbs.; 3½ lbs. shipping. Price, Net, $25.50.



"Over thirty-five years of audio progress"

Langevin

A Division of Sonotec Incorporated

FREE Write Today! $1.00 Value—84 Page Professional Audio Equipment Catalogue covering Low-Level Amplifiers, Limiters, Power Amplifiers, Variable and Fixed Equalizers, High and Low Pass Filters, VU Meters and Panels, Plugs, Patchcords, Jacks and Jackstrips, Telephone Keys, Transformers, Precision Instrument Switches and complete line of new Langevin Attenuators, Pan-Pots, Straight Line and Rotary Mixers, VU Range Extenders and many others.

503 SOUTH GRAND AVENUE • SANTA ANA, CALIFORNIA

3. Team responsibilities

Assigning responsibilities to team members is a crucial step in ensuring the successful completion of the project. This involves identifying the various tasks required for the project and assigning them to individuals with the necessary skills and expertise. In our project, we have identified several key responsibilities that must be fulfilled in order to meet our objectives. These responsibilities include version control, backend development, reference management, analog circuit modeling, QA testing, user-interface design, human-centric design, and GUI development.
To ensure that each of these responsibilities is fulfilled to the best of our ability, we have identified four key roles that will be responsible for overseeing different aspects of the project. These roles include a code reviewer, an audio specialist, a UI/UX designer, and a full-stack developer. The code reviewer will be responsible for ensuring that all code changes are reviewed and approved before being merged into the main codebase. The audio specialists will be responsible for ensuring that the circuit modeling accurately reflects the desired sound output. The UI/UX designer will be responsible for creating an intuitive and user-friendly interface for the project, while the full-stack developer will be responsible for overseeing the entire project and ensuring that all aspects of development are on track.

Name	Role	Responsibilities	Ambitions	Background
Du	Code reviewer	Version control, backend developing, reference management	Software Lifecycle Management	Git Management, Backend Coding
Tommaso	Audio Specialists	Analog circuit modeling, QA testing	Deepen analog circuit design theory	
Sofia			GUI, backend developing	Audio Engineering, DSP, Machine Learning
Chris	UI/UX Designer	User-interface design, human-centric design, QA testing	Deepen circuit design and front-end design knowledge	Voice-User Experience Design, Usability Testing, User Research
Pablo	Full Stack Developer	Analog circuit modeling, GUI	GUI	DSP & whatever they teach at audiovisual syst engineering
Polina	Project Manager	Github update, idea writing, check documents, code	Steps in the project existence	

- **Code reviewer**: A code reviewer is responsible for examining and evaluating the quality of code written by other team members. They ensure that the code adheres to established coding standards, is efficient, and free of bugs and errors.

- **Audio specialist:** An audio specialist is a professional who has in-depth knowledge of audio technology, production, and processing. They are responsible for ensuring that the project's audio components are of high quality and meet the desired standards.

- **UI/UX designer:** A UI/UX designer is responsible for designing the user interface and user experience of a software application or website. They ensure that the application is easy to use, intuitive, and visually appealing.

- **Full stack developer:** A full-stack developer is responsible for developing both the front-end and back-end of a software application. They must have knowledge of both client-side and server-side technologies.

- **Project manager:** A project manager is responsible for planning, executing, and monitoring a project. They ensure that the project is completed on time, within budget, and to the desired quality standards. They also manage the project team and facilitate communication between team members.

4. Schedule

The schedule for this project will be continually updated as we progress. We understand that a well-organized schedule is crucial to meet deadlines and deliverables. We will maintain a shared calendar to ensure everyone is aware of upcoming milestones, meetings, and deadlines. Additionally, we will be flexible and agile in our approach to the project, adjusting the schedule as needed to ensure we can deliver high-quality work within the agreed-upon timeframe.

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Project plan		x	x	o						
Software Development Tools				x						
State of the art				x			o			
Software Requirements Specifications						x			o	
Software Development Github							x			
Ethical considerations								x		
Build first python prototype					x					
Building release						x				
Design										o
Final project presentation										o

x → version 1 deliverable
o → final version deadline
color → working time

5. Risk management

Risk management is an essential process in project management that involves identifying, analyzing, and evaluating potential risks that could affect project objectives, scope or timeline. The purpose of risk management is to minimize the impact of risks on the project and increase the chances of successful project completion. The risk management process is iterative and continuous, involving ongoing monitoring and evaluation of risks throughout the project lifecycle. The process includes risk identification and risk response planning. The goal of risk management is not to eliminate all risks, but rather to mitigate or manage them effectively so that they do not have a significant negative impact on the project. In this section, we will discuss the risk management process in detail, including its importance in project management and some of the common risk management strategies used in projects.

4.1 Risk identification.

Risk identification is a critical step in the project management process that involves identifying potential risks that could impact the success of the project. It involves analyzing the project and identifying any potential events, situations or circumstances that could negatively affect the project objectives, scope or timeline. By identifying potential risks early in the project, project managers can develop a risk management plan to mitigate or manage the impact of those risks on the project. In this section, we will discuss the process of identifying risks and some of the common risks that may be encountered in projects.

4.1.1 Technical Risks

Risks associated with analog circuits can present significant challenges in the design and implementation of a project. Analog circuits often have non-linearities and unanticipated sound responses that can be difficult to model accurately. The complexity of these circuits can result in increased project costs and timelines if not appropriately addressed. Additionally, analog circuits are subject to interference and noise, which can further impact their performance and reliability.

4.1.2 Compatibility Risks

Compatibility risks associated with WDFs can impact the performance and functionality of the system in which they are used. These risks can occur due to issues with software or hardware compatibility with the python packages, such as incompatibility with operating systems, programming languages, or hardware platforms. If these issues are not addressed, they can lead to increased project costs and timelines, reduced system performance, and even project failure.

4.1.3 Schedule Risks

Unforeseen project delays can occur due to a variety of factors, such as technical difficulties, scope creep, or inadequate staff availability. These delays can have a significant impact on the project timeline, leading to reduced stakeholder satisfaction and increased project costs.

Technical difficulties may arise during the project lifecycle due to unanticipated issues with technology, hardware or software failures, or compatibility issues. Scope creep, or the gradual expansion of project requirements, can also lead to delays as additional work is added to the project

scope. Inadequate staff availability can be another significant cause of project delays, particularly if key team members are unavailable due to illness, resignations, or other reasons.

4.1.4 User Experience and Workflow:

Control signals from the 7 faders should as closely resemble the experience of using an analog EQ such as the EQ252A. Movement and manipulation of the GUI must result in similar effects, with special consideration to the nonlinearity of certain parameters.

The user-experience should seamlessly integrate into the workflow of the end-user

4.1.5 Accuracy Risks

Accuracy risks can present significant challenges when designing and implementing a project that involves analog circuit modeling and digital filtering algorithms. These risks can arise due to issues related to the precision of analog circuit modeling or the efficacy of digital filtering algorithms.

Inaccurate analog circuit modeling can lead to errors in the system design, resulting in issues with system performance and reliability. The accuracy of analog circuit modeling can be impacted by factors such as the quality of data used, the complexity of the circuit, and the accuracy of measurement devices.

Digital filtering algorithms are used to process digital signals and are crucial in achieving high accuracy in digital signal processing applications. Inaccurate or ineffective digital filtering algorithms can result in poor system performance, signal distortion, and increased noise in the signal.

4.2 Risk response plan

Risk response plan is a critical component of project management that involves identifying potential risks and developing strategies to address them. It is an iterative process that requires ongoing monitoring and evaluation throughout the project lifecycle. The purpose of a risk response plan is to minimize or eliminate the impact of risks on the project objectives, scope or timeline. It includes a set of actions, procedures and guidelines that can be implemented to mitigate or manage identified risks. The risk response plan is developed in consultation with the project team members, stakeholders, and subject matter experts. The plan is a proactive approach to risk management that helps to ensure the success of the project by addressing potential risks before they become major issues. In this section, we will discuss some of the common strategies used to address different types of risks in this project.

4.2.1 Technical Risks

- Conduct a comprehensive technical analysis to identify potential risks impacting the project.
- Develop a contingency plan that includes alternatives and backup solutions to address potential technical issues that may arise.
- Conduct regular testing and quality assurance activities to identify technical issues early in the project cycle.
- Establish a change management process to ensure that the appropriate stakeholder's document, communicate, and approve any technical changes.
- Maintain open communication between project team members to address technical issues promptly and effectively.

4.2.2 Compatibility Risks

- Research and test compatibility with all necessary hardware and software systems before implementation.
- Identify alternative hardware and software solutions that could be used if compatibility issues cannot be resolved.

4.2.3 Schedule Risks

- Create a detailed project schedule with clear deadlines and milestones.
- Establish a change management process to assess the impact of any schedule changes.
- Allocate additional resources if necessary to meet project deadlines.

4.2.4 User Experience and Workflow:

- Conduct user research and usability testing to identify potential user experience issues early in the design process.
- Incorporate feedback from users and stakeholders to refine the user experience and workflow.
- Conduct regular reviews of the user experience and workflow throughout the project to ensure that they meet the needs of the users and the project requirements.
- Allocate adequate time and resources for user experience and workflow design and testing.

4.2.5 Accuracy Risks

- Develop a comprehensive testing plan that includes both unit and integration testing.
- Conduct testing at each stage of the project to identify any accuracy issues early.
- Establish a formal quality assurance process to ensure that all deliverables meet the required level of accuracy.
- Allocate adequate time and resources for testing and quality assurance.

STATE OF THE ART

1. Introduction

This project explores audio processing and software development topics like frequency response, equalization, filters, virtual analog modeling, nonlinear systems, and reverse engineering. We delve into frequency response and its significance, followed by equalization (active, passive, and graphic), different filters (low pass, high pass, band pass, all-band pass, LC, and RC), virtual analog modeling (white-box, black-box, wave digital filters), nonlinear systems, and reverse engineering in software development.

2. Frequency response:

Frequency response is a characteristic of any system or device that processes or produces sound, such as speakers, headphones, microphones, and audio equipment. It refers to the frequency response of a system as the quantitative measure of the magnitude and phase of the output as a function of input frequency [1], [33].

In audio processing, such as during mixing and mastering, frequency response is an essential consideration when adjusting the equalizer (EQ). By altering the frequency response of a track or audio signal, you can emphasize or reduce specific frequencies to achieve the desired sound. [1]

Additionally, it is important to consider how the phase of the audio signal is affected by the EQ adjustment and filters. There are various types of EQs that handle phase differently, including minimum phase EQs and linear phase EQs. Depending on the task at hand, one would choose a different type of EQ, with a preference for EQs that preserve the phase response for mastering tasks and EQs that allow phase shift for mixing tasks. This is because preserving the phase response is crucial for maintaining the overall tonal balance of a track, while allowing phase shift can be useful for creating space and separation between individual elements in a mix.[1], [2]

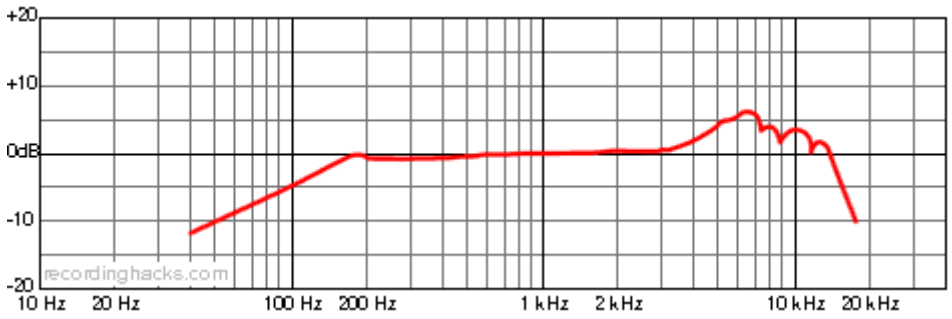


Figure 4. Frequency response of the Shure SM57 microphone.

3. Equalization:

Equalization is a fundamental tool in audio production that enables the modification of the frequency response of a track or audio signal. By adjusting the levels of specific frequencies, you can emphasize or reduce them to achieve the desired sound. [1]

The primary objective of EQ during the mixing process is to ensure that the various instruments in a song blend seamlessly. For instance, if two instruments occupy the same frequency range, they can clash and create a muddy or cluttered sound. EQ can help to separate these instruments by emphasizing different frequencies, making them sound distinct and clear.

There are several types of EQ available, the most common being graphic EQ and parametric EQ, each with its unique characteristics and uses. [2]

Here are some common parameters that can be found in both, graphic and parametric EQs.

- Center frequency: This parameter allows you to select the specific frequency band that you want to modify. It is usually expressed in Hertz (Hz) and can range from low frequencies (e.g., 20 Hz) to high frequencies (e.g.20 kHz).
- Gain/Level: This parameter controls the boost or cut applied to the selected frequency band. It determines how much the volume of that frequency range is increased or decreased. It is usually measured in decibels (dB), with positive values indicating a boost and negative values indicating a cut.
- Q Factor/Bandwidth: This parameter determines the width of the frequency band affected by the EQ. It controls the range of frequencies around the selected frequency that will be modified. A narrower bandwidth (higher Q factor) will affect a smaller range of frequencies, while a wider bandwidth (lower Q factor) will affect a broader range.
- Filter Type: EQs often offer different filter types that affect how the frequencies are modified, in order to serve a particular purpose (e.g.HPF, BPF, notch,...). We will discuss more about the filter types in the next section.
- On/Off/Bypass: This parameter allows you to enable or disable the EQ for a particular track or audio signal. When bypassed, the EQ settings have no effect, allowing you to compare the original sound with the modified sound.

3.1 Parametric equalizer:

A parametric equalizer is a type of equalizer that allows you to select a specific frequency and range of frequencies to adjust, using a Q factor to control the width of the range. The ability to target specific frequencies and adjust them precisely makes parametric equalizers a popular choice for professional audio engineers, particularly in mixing and mastering. [1]

Figure 5: API 5500 equalizer, which features frequency knobs (white) for selecting the center frequency, a gain knob (blue) for adjusting the boost or cut of the selected frequency band, and a range control that allows for micro-adjustments of tonal color. The bypass button removes the equalization, but the electronics and output transformer remain in the signal path, distinguishing it from the out button which applies no equalization. Additionally, the highest and lowest bands offer a shelving filter option for eliminating unwanted noise or adjusting tonal balance towards treble or bass.

The Q is a measure of the width of the range of frequencies affected by the EQ, typically expressed as a ratio. For example, a Q of 1.0 affects a range of frequencies equal to the chosen frequency, while a Q of 0.5 affects a broader range of frequencies. The range of Q values available can vary depending on the EQ software or hardware being used. [3]

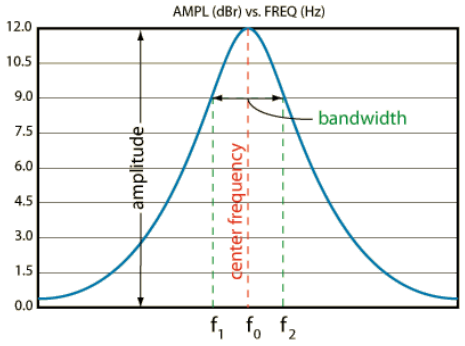


Figure 6: Simple representation of Q. The bandwidth is a term used to describe the range of frequencies that can be transmitted or processed by a filter or signal. In this context, f0 represents the center frequency of the signal or filter, while f1 and f2 represent the frequencies where the signal or filter response increases or decreases by a certain amount from the maximum level. The bandwidth is simply the difference between f1 and f2. So, for example, if the filter has a center frequency of 1000 Hz and the response decreases by 3 decibels at 800 Hz and 1200 Hz, then the bandwidth would be 400 Hz (1200 Hz - 800 Hz).

One of the benefits of a parametric EQ is the ability to select a small band of frequencies and boost or cut them, allowing you to carve out specific parts of the frequency spectrum. This is particularly useful in mixing, as it enables you to adjust the frequency response of each instrument in the mix, helping them fit together more cohesively. [3]

3.2 Graphic equalizer

A graphic equalizer is a type of audio equalizer that allows you to adjust the amplitude of different frequency bands in an audio signal. It typically has a series of sliders, with each slider representing a specific frequency band. By adjusting the slider up or down, you can increase or decrease the volume of that particular frequency range (also meaning that they can be boosted with respect to the input signal gain). [1] [4]

Figure 7: MXR 10 band equalizer guitar pedal. This graphic eq has ten frequency centers. The sliders can boost or attenuate the signal by up to ±18dB. The volume control adjusts the overall effect volume, while the gain slider controls the gain boost.

3.3 Passive graphic equalizer:

Passive graphic equalizers are a type of graphic equalizer that do not require any external power source to function. Instead, they rely on passive components such as resistors, capacitors, and inductors to adjust the frequency response. Passive graphic equalizers are known for their simplicity and lack of distortion, but they also tend to have a limited range of frequency bands and a reduced level of control compared to active graphic equalizers. They are often used in home audio systems and for tone shaping in musical instruments. [1],[2]

Figure 8: Langevin EQ-252A Graphic EQ.

4. Filters

The theory of linear filters forms the foundation of audio equalization, where input signals are processed over time to produce output signals while following the rule of linearity. These filters can be analyzed using line-time-invariant (LTI) system theory to reveal their frequency domain transfer functions and time domain impulse responses. Real-time implementations of these filters are usually causal, meaning they only depend on past and present input signals. The adjustment and flexibility of audio equalizers depend on the circuitry and controls available to the user.

Shelving controls and parametric equalizers are commonly used in audio processing, allowing users to adjust the gain of high or low frequencies relative to their cutoff frequencies and adjust center frequency, bandwidth, and gain controls, respectively. On the other hand, graphic equalizers feature a bank of fixed second-order filters that cover the audio spectrum, and users can adjust each slider to create the desired frequency response. However, elongating the order of the filter results in a compromise between accuracy, computational power, and real-time audio processing.

4.1 Low-pass filters:

A low-pass filter is an electronic circuit that allows low-frequency components of a signal to pass through while attenuating higher-frequency components. It is characterized by its cutoff frequency, below which most of the input signal is passed through and above which most of the input signal is attenuated. A first-order low-pass filter consists of a resistor in series with the input signal and a capacitor in parallel with the output signal (see figure 6). Its transfer function decreases as frequency increases, with a cutoff frequency of 1/RC. A second-order low-pass filter adds a second RC circuit in series with the first, which further attenuates high-frequency signals (see figure 7). [5], [7]

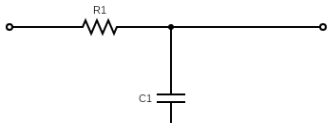


Figure 9. First order low pass filter circuit.

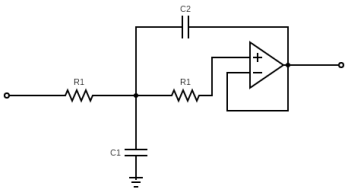


Figure 10. Second order Low pass filter circuit.

In a low-pass filter, the passband is the range of frequencies that are allowed to pass through the filter with minimal attenuation, while the stopband is the range of frequencies that are heavily attenuated or blocked completely. [6]

In figure 8, the cutoff frequency, denoted as fc, is the boundary between the pass band and the stop band. It is the frequency at which the filter begins to attenuate the signal. The stop band also has a boundary, denoted as fs, which is the frequency at which the filter completely blocks the signal.

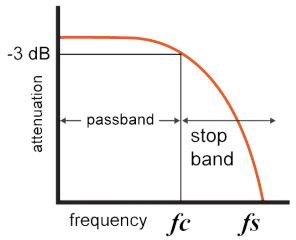


Figure 11. Low pass filter graph.

4.2 High-pass filters:

A high-pass filter is an electronic circuit that allows high-frequency components of a signal to pass through while attenuating lower-frequency components. Like a low-pass filter, it is characterized by its cutoff frequency, above which most of the input signal is passed through and below which most of the input signal is attenuated. A first-order high-pass filter consists of a capacitor in series with the input signal and a resistor in parallel with the output signal (see figure 9). Its transfer function increases as frequency increases, with a cutoff frequency of 1/RC. A second-order high-pass filter adds a second RC circuit in parallel with the first, which further attenuates low-frequency signals (see figure 10). [5],[7]

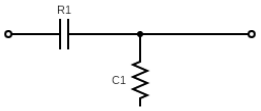


Figure 12. First order high pass filter.

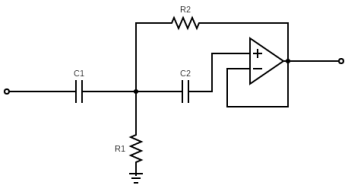


Figure 13. Second order high pass filter.

In a high-pass filter, the passband is the range of frequencies above the cutoff frequency that are allowed to pass through the filter with minimal attenuation, while the stopband is the range of frequencies below the cutoff frequency that are heavily attenuated or blocked completely. In figure 11, the cutoff frequency, denoted as fc, is the boundary between the stop band and the pass band. It is the frequency at which the filter begins to allow the signal to pass through. The stop band also has a boundary, denoted as fs, which is the frequency at which the filter completely blocks the signal.

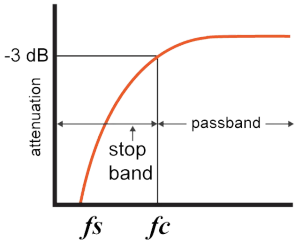


Figure 14. High pass filter graph.

4.3 Band-pass filters

A bandpass filter is an electronic circuit that allows a certain range of frequencies, known as the passband, to pass through while attenuating frequencies outside of this range. It is characterized by its center frequency, which is the frequency in the middle of the passband, and its bandwidth, which is the range of frequencies that are allowed to pass through. A first-order bandpass filter consists of a capacitor in series with the input signal and a resistor in parallel with the output signal (see figure 12). [5], [7]

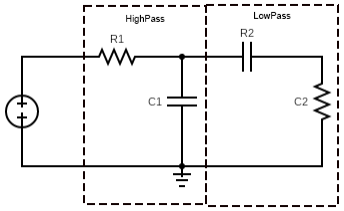


Figure 12. First order band pass filter.

A second-order bandpass filter adds a resonant circuit, such as an inductor and capacitor in parallel, to the first-order bandpass filter configuration. This additional circuit creates a peak in the frequency response, resulting in a sharper roll-off at the edges of the passband and a steeper transition between the passband and stopband. The center frequency and bandwidth of the passband are determined by the values of the resistors, capacitors, and inductor in the circuit. (see figure 13) [7]. [8]

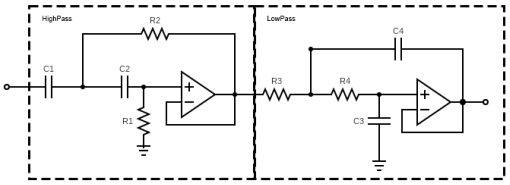


Figure 13. Second order active band pass filter.

In a bandpass filter, the passband is the range of frequencies that are allowed to pass through the filter with minimal attenuation, while the stopband is the range of frequencies that are heavily attenuated or blocked completely. The center frequency, denoted as f0, is the frequency in the middle of the passband, and the bandwidth, denoted as B, is the range of frequencies that are allowed to pass through. The lower and upper cutoff frequencies, denoted as f1 and f2 respectively, are the frequencies at which the filter begins to attenuate the signal.

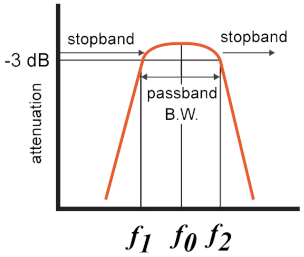


Figure 14. Band pass filter graph.

As shown in figure 14, the frequency response of a bandpass filter has a peak at the center frequency f0, which is the frequency that passes through with the least attenuation. The bandwidth B is the range of frequencies between the lower and upper cutoff frequencies f1 and f2, where the filter starts to attenuate the signal. Frequencies outside of this range are heavily attenuated or blocked completely in the stopband.

4.4 Band-stop filters:

A bandstop filter, also known as a notch filter, is an electronic circuit that attenuates a range of frequencies while allowing others to pass through. It is the opposite of a bandpass filter. A bandstop filter is characterized by its center frequency and its bandwidth, which is the range of frequencies that are attenuated. [5], [6]

A first-order bandstop filter can be created by combining a high-pass filter and a low-pass filter. (see figure 15). Alternatively, it can be created using a single circuit with a capacitor in parallel with a series combination of a resistor and an inductor (see figure 16). [7], [8]

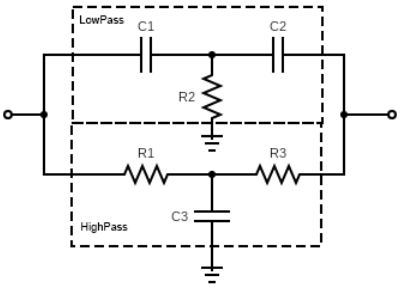


Figure 15. Band-stop filter circuit.

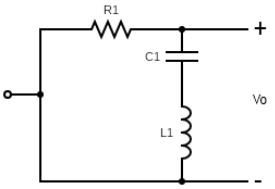


Figure 16. Band-stop filter circuit with an inductor.

A second-order bandstop filter is created using two of the same circuits in series, or a single circuit with an additional capacitor in parallel with the first circuit (see figure 17). [7]

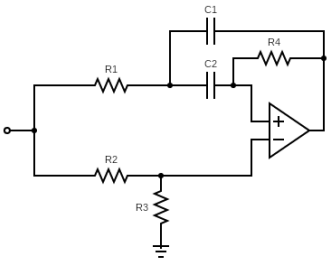


Figure 17. Band-stop filter circuit with an inductor.

In a bandstop filter, the stopband is the range of frequencies that are heavily attenuated or blocked completely, while the passband is the range of frequencies that are allowed to pass through with minimal attenuation. The center frequency of the stopband is the frequency at which the filter provides the maximum attenuation. The bandwidth of the stopband is the range of frequencies around the center frequency that are attenuated. Figure 18 shows the frequency response of a bandstop filter.

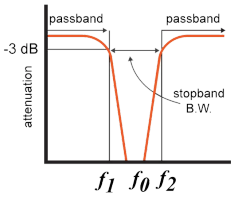


Figure 18. Band-stop filter graph.

The frequency range where the filter operates is called the "stopband" while the frequencies that are allowed to pass through are called the "passband". The width of the stopband is determined by the "Q factor" of the filter, which is a measure of how selective the filter is in rejecting frequencies outside the passband.

4.5 All-pass filters:

An all-pass filter is an electronic circuit that allows all frequencies to pass through, but changes the phase relationship between them. It is created by combining a high-pass filter and a low-pass filter in series or parallel (see figure 19). The output signal of the all-pass filter has the same amplitude as the input signal but with a phase shift that depends on the frequency (see figure 20). The all-pass filter finds applications in audio processing, control systems, and telecommunications, where phase information is important. [8]

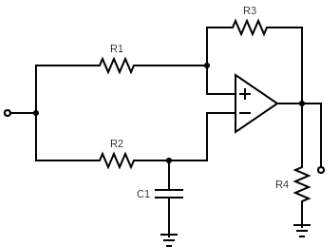


Figure 19. All pass filter circuit.

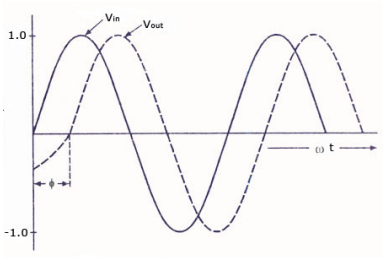


Figure 20. All pass filter graph.

4.6 RC Filters

An RC filter is an electronic filter that uses a combination of resistors and capacitors to pass or attenuate certain frequencies of an input signal. The name RC comes from the use of these two components in the filter circuit. [7], [8]

An RC filter is an electronic filter that uses a combination of resistors and capacitors to pass or attenuate certain frequencies of an input signal (see figure 22). The name RC comes from the use of these two components in the filtA first-order RC filter consists of a single resistor in series with a capacitor. The input signal is applied across the series combination of the resistor and capacitor, and

the output is taken across the capacitor. The capacitor acts as a frequency-dependent resistor, with its impedance decreasing as the frequency increases. [7], [8]

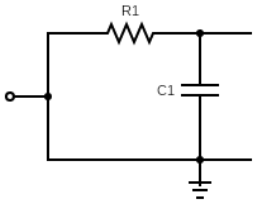


Figure 22. All pass filter graph.

As the frequency of the input signal increases, the impedance of the capacitor decreases, and more of the signal is allowed to pass through to the output. Conversely, as the frequency of the input signal decreases, the impedance of the capacitor increases, and more of the signal is attenuated or blocked. The cutoff frequency of the filter is the frequency at which the output signal has been attenuated to half of its original amplitude. [7], [8]
A second-order RC filter adds another resistor and capacitor to the circuit, creating a more complex transfer function that can further attenuate or pass certain frequency ranges. (see figure 23) [7], [8]

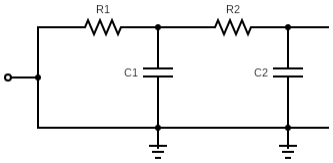


Figure 23. All pass filter graph.

RC filters are commonly used in audio applications, where they can be used to remove unwanted noise or to shape the frequency response of an amplifier or loudspeaker. They are also used in power supply circuits to filter out unwanted ripple or noise.

4.7 LC Filters

An LC filter is an electronic circuit that uses inductors and capacitors to filter out unwanted frequency components from a signal. It is commonly used in radio frequency and audio applications.[7], [8]
In an LC filter, the inductor and capacitor are connected in either series or parallel (see image 24). A series LC filter consists of an inductor and a capacitor in series with the signal, while a parallel LC filter consists of an inductor and capacitor in parallel with the signal. [7]

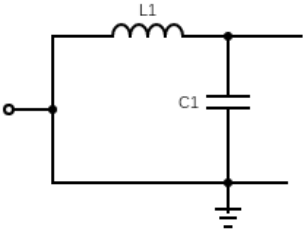


Figure 24. LC filter circuit.

The LC filter works by utilizing the natural resonance frequency of the LC circuit. At this frequency, the inductor and capacitor resonant, and the impedance of the circuit is at its minimum. This means that the circuit passes the input signal with minimal attenuation. At frequencies above or below the

resonance frequency, the impedance of the circuit increases, and the circuit starts to attenuate the input signal. [7], [8]

The resonance frequency of an LC filter is determined by the values of the inductor and capacitor. A higher capacitance value or lower inductance value will result in a lower resonance frequency, while a lower capacitance value or higher inductance value will result in a higher resonance frequency. [7], [8]

LC filters are widely used in electronic circuits to remove unwanted noise or harmonics from a signal or to select a specific frequency range for further processing. [7], [8]

5. Virtual Analog Modeling

Analog modeling is a technique used in digital signal processing to simulate the behavior of real-world systems, such as electrical circuits. It has a rich history dating back to the 1960s, when researchers at Bell Labs developed mathematical models of physical systems to help engineers design and optimize circuits. Analog modeling techniques were later used in digital signal processing, which led to the development of early digital audio effects in the 1970s, such as reverb and delay, based on mathematical algorithms that could be programmed into digital signal processors (DSPs). [10]

As DSP technology improved, so did the sophistication of digital audio effects. Companies such as Eventide and Lexicon began using digital signal processing to create high-quality digital reverbs and other effects that could compete with or even surpass their analog counterparts in the 1980s. In the 1990s, analog modeling techniques emerged with the aim of capturing the sound and feel of classic analog equipment in a digital format. The first commercial analog modeling product was the Line 6 AxSys 212 guitar amplifier, released in 1997, which used digital signal processing to emulate the sound and behavior of classic guitar amplifiers. [10]

Virtual analog modeling is a technique used in digital audio to create software versions of classic analog audio gear, such as synthesizers, guitar amps, and other audio processors. The goal of virtual analog modeling is to create a digital version of an analog device that sounds and behaves like the original. To achieve this, virtual analog modeling uses complex mathematical algorithms to simulate the electrical behavior of the original analog circuitry, creating a digital model of the analog circuit that accurately reproduces its sound and behavior. This allows musicians and audio engineers to access the sound and character of classic analog gear without the high cost and maintenance requirements of physical analog equipment, making it more accessible and affordable for them. [10]

Some popular virtual analog models include the Moog Minimoog, Roland TB-303, and Fender Twin Reverb guitar amplifier. These virtual instruments and effects have become popular among musicians and producers for their ability to create authentic analog sounds in a digital environment. Virtual analog modeling has many advantages over physical analog equipment. It allows musicians and audio engineers to store and recall virtual instruments and effects instantly, use them in combination with other virtual instruments and effects, and have more flexibility in creating their sound. [10]

There are two types of virtual analog modeling: white box modeling and black box modeling.

5.1 White Box Modeling:

White box modeling is a modeling technique that uses knowledge of the internal mechanisms of a system to create a detailed mathematical model that accurately represents the physical interactions of the system's components. In the case of electrical circuits, a white box approach involves modeling the electrical components and their interactions, such as the flow of current, voltage, and resistance. [11]

This approach to modeling is sometimes called "physical modeling" because it attempts to replicate the physical processes that govern the system. For example, a white box model of an electrical circuit

might include models of individual resistors, capacitors, and inductors, as well as models of how these components interact with each other to produce the circuit's behavior.

White box modeling is often used in the design and optimization of complex systems, such as circuits, control systems, and mechanical systems. By understanding the physical interactions of the components, engineers and designers can create models that accurately predict the behavior of the system under different conditions. [11]

One advantage of white box modeling is that it allows engineers and designers to make informed decisions about how to optimize the system for performance, efficiency, or other desired characteristics. For example, a white box model of an electrical circuit might reveal areas where adding or removing components could improve the circuit's performance, or where changes to the component values could reduce power consumption or improve reliability. [11]

However, white box modeling also has some limitations. It requires a detailed understanding of the system's internal mechanisms, which may not always be possible or practical. In addition, the model may become more complex and difficult to understand as more components and interactions are included, making it harder to analyze and optimize. [11]

Some popular methods of white box modeling include nodal analysis, Port-Hamiltonian analysis, non-linear state space analysis and wave digital filters. [11]

5.1.1. Wave Digital Filters:

Wave Digital Filters (WDFs) provide a flexible and modular approach to circuit modeling. Originally introduced by Alfred Fettweis in the 1970s [34], WDFs have gained popularity in modeling audio circuits and have been extended to model a wider class of circuits [35]. In WDF formalism, each circuit element is treated as a port with a characteristic resistance R0, and wave variables are used instead of traditional voltage and current variables. The incident and reflected waves are defined as

$$a = v + R0i$$

and

$$b = v - R0i$$

By using the WDF formalism, circuit elements such as resistors, capacitors, and inductors can be defined in the wave domain and connected through series and parallel adaptors. The WDF formalism is particularly useful in modeling analog circuits that exhibit nonlinear behavior, such as distortion effects in guitar amplifiers. [12]

5.2 Black box modeling:

Black-box modeling is a technique used in circuit modeling that aims to predict the behavior of a circuit based on input and output measurements, without any knowledge of the internal workings of the circuit. This approach is useful when the circuit's internal structure is complex or unknown, and when the focus is on understanding the circuit's behavior rather than its underlying physical mechanisms. [11]

Traditional black-box techniques involve measuring the circuit's impulse response, which is the response of the circuit to a short pulse input. This approach has been extended with methods like the Wiener-Hammerstein method, which uses nonlinear regression techniques to model the system. More recently, machine learning methods have been employed to improve black-box modeling, providing more accurate predictions of the circuit's behavior. [11]

One study by Damskägg et al. [13] used a WaveNet-style architecture to generate an output signal sample-by-sample for several guitar distortion circuits. The approach involved training a neural

network to predict the output of the circuit based on input samples, without any knowledge of the circuit's internal workings. Another study by Parker et al. used deep fully-connected networks to approximate nonlinear state-space solutions for the Korg MS20 filter circuit. This approach represented a "gray-box" method, where some knowledge of the circuit's internal workings is incorporated into the modeling process. [13]

Finally, Wright et al. [13]used a recurrent neural network to model the behavior of guitar distortion circuits with control parameters. The approach was able to predict the output of the circuits with high accuracy, demonstrating the potential of black-box modeling methods for circuit analysis. [13]

Some examples of black box modeling are deep neural networks, volterra series and sweep sine analysis.

6. Reverse engineering:

Reverse engineering is the process of taking apart an object, system, or software program to understand how it works, without access to its original design or documentation. It involves analyzing and examining the product to determine how it was designed and constructed, with the aim of producing a replica or a modified version of the original product.

Reverse engineering is often used to learn from and improve upon existing products, to identify and fix bugs, or to create compatible products that work with a specific system or software. It can also be used for competitive analysis, to understand how a competitor's product works and to create a similar product with improved features.

In virtual analog modeling, reverse engineering techniques can be used to recreate the behavior of analog circuits and devices in digital form. This involves analyzing the internal structure and characteristics of the analog circuit, and then using that information to create a digital model that accurately reproduces its behavior.

For wave digital filters specifically, reverse engineering can be used to extract the parameters and characteristics of the filter from an existing analog device or circuit, such as a physical filter or a schematic diagram. This can involve analyzing the frequency response, phase response, and other characteristics of the analog filter, and then using that information to design a digital model that replicates those characteristics. [14]

7. Nonlinear systems

Nonlinear systems are those that do not follow the principle of superposition, which means that the output is not proportional to the input. Instead, nonlinear systems have a complex relationship between input and output, and the output may include additional frequency components that were not present in the input. [15]

One common example of non-linear systems in audio is distortion, which is intentionally added to many guitar amplifiers and effect pedals to create a desired sound. Distortion is the result of non-linear amplification, where the output signal is not a linear function of the input signal. This causes the input signal to be amplified differently at different levels, and results in the creation of new harmonic frequencies that were not present in the original signal.

Another example of non-linear systems in audio is compression, which is commonly used in music production and live sound reinforcement. Compression is a technique that reduces the dynamic range of an audio signal, by attenuating the louder parts of the signal while leaving the quieter parts unchanged. The compression process is non-linear, as it applies different amounts of attenuation to different parts of the signal based on the input level.

Nonlinear systems can be more complex to analyze than linear systems, as the relationship between input and output is not as simple. However, they are often used in audio systems to achieve specific artistic or technical goals, such as adding warmth or character to a sound, or controlling the dynamics of a recording or live performance. [15]

8. Literature review

There have been numerous published articles related to virtual analog modeling and specifically wave digital filters. These articles range from theoretical papers describing the principles of virtual analog modeling to practical articles detailing the implementation of virtual analog models in software and hardware [12].

One common topic in these articles is the use of wave digital filters (WDFs) for virtual analog modeling. WDFs are a type of digital filter that is well-suited for modeling analog circuits due to their ability to accurately simulate the behavior of passive components such as capacitors, inductors, and resistors. Many articles describe the implementation of WDFs for virtual analog modeling of various types of analog circuits, including filters, amplifiers, and oscillators. The modeling and combination of these analog components has allowed for more realistic emulations of iconic tools such as the Fairchild 670 limiter or the Roland TR-808 drum machine [30][31].

Another common theme in these articles is the use of machine learning techniques for virtual analog modeling. Machine learning algorithms can be trained on large datasets of measured or simulated analog circuit responses to learn the underlying relationships between circuit parameters and circuit behavior. These algorithms can then be used to create virtual analog models that accurately reproduce the behavior of the original analog circuits [32].

If you're interested in learning more about Wave Digital Filters, be sure to check out these informative articles:

- "**Wave Digital Filters: Theory and Practice**" by Alfred Fettweis. This paper provides a comprehensive review of the theory of wave digital filters (WDFs), which are digital filters designed to mimic classical filters in lattice or ladder configurations or their generalizations. WDFs are known for their accuracy requirements, dynamic range, and stability under finite-arithmetic conditions. The primary objectives of this review are to introduce WDF theory to those unfamiliar with the subject, emphasize practical aspects for those interested in designing or applying WDFs, and explore the broad range of relationships between WDF theory and other areas, particularly in signal processing. The review includes mathematical analyses only when necessary for essential insights, with detailed references to existing literature for more specialized aspects. You can read it at: https://ieeexplore.ieee.org/abstract/document/1457726/
- "**Toward a Wave Digital Filter Model of the Fairchild 670 Limiter**" by Peter Raffensperger. This paper presents a digital model of the Fairchild R670 vacuum tube limiter, a highly regarded analog circuit from the 1950s, using a combination of black boxes and wave digital filters as a step towards a fully wave digital filter design. Wave digital filters offer an efficient and modular way to digitally simulate analog circuits. A new model for the 6386 triode is introduced to simulate the active component in a wave digital filter model of the Fairchild 670's signal amplifier. The signal amplifier is integrated with a hybrid wave digital filter/black-box sidechain amplifier model to form a complete model of the Fairchild 670. The paper discusses the model's salient features and their implications for designing dynamics processors, and provides test results for music and pure tones, highlighting the device's static gain characteristics and gain reduction-dependent distortion. You can access it at: https://www.dafx12.york.ac.uk/papers/dafx12_submission_9.pdf.
- "**Toward Nonlinear Wave Digital Filters**" by Augusto Sarti and Giovanni De Poli. This paper talks about an extension of the classic Wave Digital Filter (WDF) principles, which enables modeling of a wider range of nonlinear elements in the wave digital domain. The proposed method is based on a new class of waves that incorporates the intrinsic dynamics of a nonlinear element into dynamic multiport adaptors. This family of junctions represents a generalization of

the concept of "mutator" in the analog nonlinear circuit theory and allows us to treat a nonlinear dynamic element as if it were instantaneous (resistive). The Wave Digital Filter theory provides a systematic methodology for building digital models of analog filters by discretizing individual circuit components, and under mild conditions on its characteristic, WDF principles can also be successfully used for modeling circuits with nonlinear circuit elements. You can find it at: https://ieeexplore.ieee.org/abstract/document/765137/

- "**Digital simulation of nonlinear circuits by wave digital filter principles**" by Klaus Meerkötter and R. Scholz presents the use of wave digital filters to simulate certain nonlinear circuits digitally, with a focus on resistances having a continuous piecewise-linear voltage-current characteristic. The paper demonstrates that a representation of a nonlinear resistance in terms of the wave variables may exist, even if the voltage-current characteristic is neither voltage-controlled nor current-controlled. Chua's circuit is used as an example, and the paper shows that the same double-scroll attractor observed in the corresponding analog realization can be observed in the wave digital model. You can find it at: https://ieeexplore.ieee.org/abstract/document/100452
- **"Parallel Wave Digital Filter Implementations of Audio Circuits with Multiple Nonlinearities"** by Riccardo Giampiccolo, Antonio Natoli, Alberto Bernardini, and Augusto Sarti discusses the development of parallel audio processing algorithms that can exploit multicore processing units in modern audio systems. The paper presents a parallel version of the hierarchical scattering iterative method (HSIM), a technique based on wave digital filter principles recently proposed for the emulation of multiphysics audio circuits containing multiple nonlinear one-ports and nonlinear transformers. The paper proposes three different strategies for the distribution of HSIM workload among threads of execution, showing how to compute the maximum achievable speedup. The emulation of a possible output stage of a vacuum-tube guitar amplifier is considered, and a performance comparison between parallel and serial implementations of HSIM is presented, pointing out a speedup of nearly 30%. The proposed method thus proves to be promising for virtual analog modeling applications, leading the way towards the parallel digital emulation of increasingly complex audio circuits. You can find it at: https://www.aes.org/e-lib/browse.cfm?elib=21797.
- **"An Improved and Generalized Diode Clipper Model for Wave Digital Filters"** by Kurt James Werner, Vaibhav Nangia, Alberto Bernardini, Julius O. Smith III, and Augusto Sarti presents a novel explicit wave-domain model for "diode clipper" circuits with an arbitrary number of diodes in each orientation, applicable to wave digital filter emulation of guitar distortion pedals. The paper improves upon and generalizes the model of Paiva et al. (2012), which approximates reverse-biased diodes as open circuits, by deriving a model with an approximated correction term using two Lambert W functions. The paper studies the energetic properties of each model and demonstrates the model's validity by comparing a modded Tube Screamer clipping stage emulation to SPICE simulation. You can access it at https://www.aes.org/e-lib/browse.cfm?elib=17918.
- **"Virtual Analog Modeling of Audio Circuitry Using Wave Digital Filters"** by Kurt James Werner explores the use of wave digital filters for virtual modeling of audio circuitry. The paper discusses the advantages of wave digital filters, including high numerical stability, energy conservation, and the ability to handle nonlinearities. The paper presents a simple virtual analog modeling framework using wave digital filters and demonstrates its application to various audio circuits. You can read it at https://www.proquest.com/openview/b0b6de66c355e70c612f80319ee3c811/1?cbl=44156&pq-origsite=gscholar&parentSessionId=a4xsVK74Tc2e4RTXRLZGxSWKI2QhY2pyr6ODm56wmWs%3D.

9. Technology review

9.1 Hardware:

In the realm of equalization hardware similar to the Langevin 252A, three notable pieces stand out:

- Motown EQ
- Cinema Engineering 7080
- Altec 9062A.

Of these three, the Cinema Engineering 7080B was the first to be introduced, followed by the Altec/Langevin 252A and the Motown EQ.
Interestingly, there is a shared lineage between these three pieces of equipment. The AM-5116 circuit, which is similar to the Langevin 252A, was produced by a number of different companies, including Collins and General Electric. These similarities can be attributed to Arthur "Art" Davis, who was involved with many of these companies and played a key role in the development and refinement of the circuit. [16]

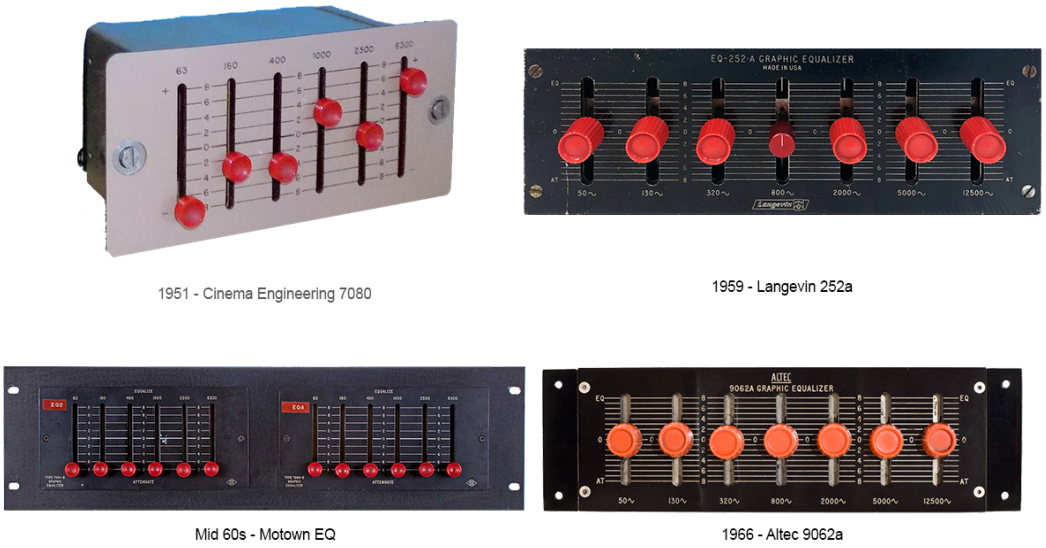


Figure 25: EQ comparison.

The development of the Langevin 252A, Altec 9062A and Motown EQs can be traced back to the pioneering work of the Cinema Engineering. Art Davis, a highly respected audio engineer and designer, played a crucial role in the creation and refinement of these designs, and his contributions to the field of professional audio are still recognized and celebrated to this day. In the early 1950s, he designed the Model 7080B, which became the first-ever passive graphic equalizer. This design would go on to influence countless engineers and designers in the years to come. Later, Davis brought his design expertise to Langevin, where he continued to refine and improve upon his earlier work. It was at Langevin that the groundbreaking 252A equalizer circuit was born, which set the standard for equalizer design for decades to come.[16]
Afterwards, some reeditions of the EQs were made, for example the Cinema Engineering 7080B or the Heritage Audio´s Motorcity EQ, which is based on the Motown EQ. [1]

9.2 Software:

As of the writing of this document, the only known plugin that emulates the Langevin 252A passive graphic equalizer is the **True 252** from Krazrog. This plugin, which was released in late 2022, comes with a range of features, including:

- 7 band graphic equalizer.
- Continuous gain controls.
- Gain compensation.
- Cramping-free equalization.

It also comes with a preset manager with A/B compare, compensated internal bypass, and separate oversampling settings for both real-time and offline processing. Users can choose between two UI themes - black and gray - which are based on the original 252-A and its cosmetic variation, the 270. While there are no other known plugins that offer virtual analog modeling of the Langevin 252A, it is possible that new options may become available in the future as this technology continues to evolve. [17] [18]

Another similar plugin is the Unique Recording Software´s Motorcity EQ, which is an emulation of the Motown EQ from the 1960s.

While there are no other known plugins that offer virtual analog modeling of the Langevin 252A, it is possible that new options may emerge in the future as this technology continues to evolve. In the meantime, audio enthusiasts can also explore other virtual analog modeling plugins from top industry brands such as Universal Audio, Overloud, PluginAlliance, Waves, Chowdhury, Physical Audio and Softube. One notable example is the Unique Recording Software's Motorcity EQ, which emulates the iconic Motown EQ from the 1960s. With virtual analog modeling technology on the rise, we can expect to see even more exciting developments in the world of audio plugin emulation.

Among these brands, the following plugins use wave digital filters:

- Waves Abbey Road REDD Consoles: This plugin uses wave digital filters to model the EQ and filter sections of the classic REDD mixing consoles used at Abbey Road Studios. [19] [36]
- Universal Audio Neve 1073 EQ & Preamp Collection: This collection of plugins uses wave digital filters to accurately model the EQ and preamp sections of the classic Neve 1073 mixing console. [20] [37]
- Chowdhury Chow Tape Model: Chow Tape Model is an open source C++ plugin that emulates the warmth and character of reel-to-reel tape machines using advanced wave digital filters. Originally developed for the Sony TC-260, it now models a wide variety of tape machines. [24]
- Physical Audio Dynamic Plate Reverb: it uses virtual analog modeling to create its reverb effects. Unlike traditional methods, it relies on the Kirchhoff plate equation to accurately calculate the modes of vibration for the system, resulting in a highly realistic and customizable sound. Details can be found in their technical paper from the 2016 International Congress on Acoustics. [25] [26]

10. Applications review

Wave digital filters are a versatile technology that has found applications in various areas, including speech processing and radar/sonar systems.

In speech processing, wave digital filters can be used for speech recognition, speech enhancement, and speech synthesis. For instance, these filters can extract features such as formants, pitch, and spectral features from the speech signal to enable accurate speech recognition. In a paper by Christian Giguère from the University of Ottawa, he proposes a model for simulating the human auditory system using wave digital filters, which are computationally efficient and allow for realistic simulation of nonlinearities and feedback mechanisms. Divided into ascending and descending paths, the model closely mimics the anatomy and physiology of the auditory periphery. The model was applied to the analysis of speech signals, and has potential applications as a front-end preprocessor for automatic speech recognition systems, as well as for simulating the deterioration of peripheral auditory function in hearing-impaired individuals. An exploratory study demonstrated the model's capabilities for speech and hearing research by simulating the communication handicap associated with a complete loss of outer hair cells. [27]

Digital filters are also widely used in radar and sonar systems for processing and filtering incoming signals. A paper from Mya Mya Aye and Thiri Thandar Aung from Ytheangon Technological Universtiy in Myanmar discussed the use of digital filtering in Radar Signal Processing for tasks such as interference reduction and Doppler processing. The proposed digital filter is designed to reject out-of-band interference and improve SNR for target detection. Doppler processing also uses digital filters to cancel signals from fixed and slow targets. The proposed filter is designed to eliminate interference, clutter, and blindness caused by higher speed targets. Matlab simulations are used to observe the features and limitations of the proposed digital filters, with windowing functions used to achieve the best outcome. [28], [29]

SOFTWARE DEVELOPMENT TOOLS

1. Code repository

The repository, which is hosted on GitHub, will serve as the central location for all code related to the project. In there, the user will find a comprehensive collection of Python scripts, libraries, and other artifacts that comprise our solution. As part of our development process, we will be using Git version control to manage changes to the codebase. This will enable us to track changes made by individual developers and manage different branches of the codebase. Additionally, we will be using pull requests to manage changes made by different team members and to ensure that all changes are reviewed and approved before being merged into the main branch. Our code repository will also contain documentation, including a README file that outlines the purpose of the project, the code structure to let the user understand,how to run the code, and any other important information. We will encourage the user to explore the code repository to familiarize themselves with the structure and contents of the codebase. To access the code repository, the user will be able to visit our GitHub page[1].

2. Branches

A branch is a lightweight movable pointer that represents an independent line of development. It allows you to work on different versions of a project simultaneously without interfering with the main codebase. Each branch has its own commit history, containing the changes made to the code over time.

Branches are flexible and can be created, switched, merged, and deleted easily. They serve as a mechanism to manage and organize the development process, providing a structured and controlled way to introduce changes to a codebase while maintaining the stability and integrity of the main branch.

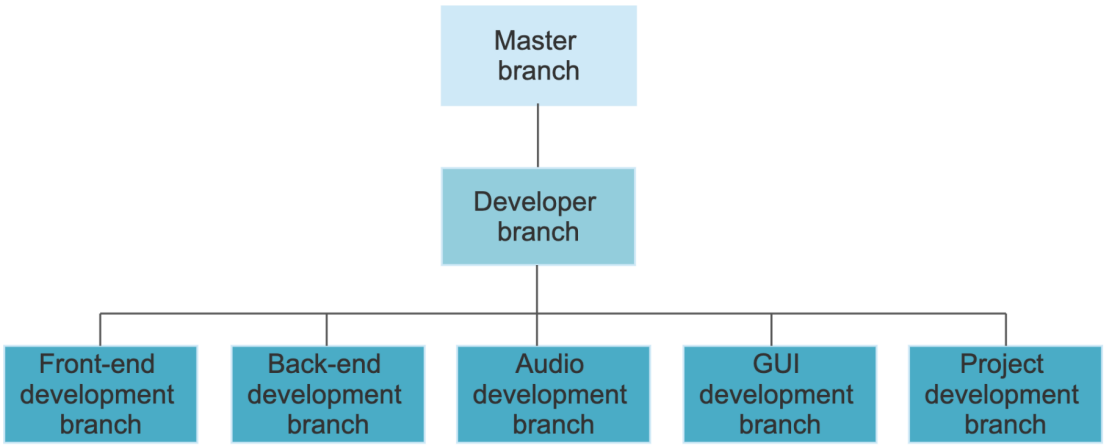


Figure 26: Branches.

- **Master branch:** The master branch is the primary branch in a version control system, such as Git. It is typically considered the stable and production-ready version of the software. It represents the main line of development and is usually used for deployment.
- **Developer branch:** The developer branch is a separate branch created by individual developers to work on specific features or bug fixes. It allows developers to work on their

[1]Github Repository: https://github.com/duduOliver/VLA-252A

changes independently without affecting the main codebase. Once the changes are completed and tested, they can be merged back into the master branch.

- **Front-end development branch:** The front-end development branch is a branch specifically dedicated to working on the user interface and user experience (UI/UX) of a software application or website. It involves writing code in languages like HTML, CSS, and JavaScript to create the visual elements and interactivity that users interact with directly.
- **Back-end development branch:** The back-end development branch is a branch focused on implementing the logic and functionality that powers the server-side of a software application. It involves working with databases, server-side programming languages (e.g., Python, Java, Ruby), and frameworks to handle data processing, business logic, and communication with the front-end.
- **Audio development branch:** The audio development branch refers to a branch dedicated to the development of audio-related features or components in a software application. It could involve tasks like integrating audio playback, recording, or processing functionality, implementing audio effects, or optimizing audio performance.
- **GUI development branch:** The GUI development branch is a branch specifically created for working on the visual interface of a software application. It involves designing and implementing graphical elements, user interactions, and layout management, primarily focusing on creating an intuitive and user-friendly interface.
- **Project development branch:** The project development branch is a general term that can refer to a branch specifically created for the development of an entire software project. It encompasses all aspects of the project's development, including front-end, back-end, and other branches dedicated to specific areas like audio or GUI development. This branch often serves as a staging area for integrating and testing various components before merging them into the master branch.

3. Permissions

Permissions in the context of software development typically refer to access rights and privileges granted to different users or entities within a version control system like Git. These permissions define what actions each user or group can perform on the repository and its associated branches.

Permission	Description	Role
Developers	A developer is a user who has been granted permission to contribute to a repository. They can create new branches, push changes to existing branches, create pull requests, and perform other tasks related to contributing to the repository.	Polina and Du
Read	A user with read access can view the repository, but they cannot make any changes to the code or files in the repository.	Tommaso, Pablo, Chris, Sofia

4. License

License: A software license is a legal document that outlines the terms and conditions under which a software program is distributed and used. It defines the rights and permissions granted to users, developers, and other stakeholders regarding the usage, modification, distribution, and redistribution of the software. Licenses play a crucial role in determining whether the software is open-source or proprietary and help protect the intellectual property rights of the software's creators. They also promote collaboration, ensure legal compliance, and provide clarity on how the software can be used.

The Affero GPL (AGPL) is a free software license that is similar to the GNU GPL, but with an additional requirement that software using the licensed code must also be released under the same license. This means that any modifications or extensions made to the original code must also be released under the AGPL, even if the modified code is not distributed to others.
We have chosen to use the Affero GPL license for several compelling reasons:

- Strong Copyleft Provision: The Affero GPL license includes a strong copyleft provision, which ensures that any modifications or extensions made to the licensed code must be released under the same license. This provision helps safeguard the principles of free and open-source software by preventing the code from being incorporated into proprietary software without contributing back to the community.
- Encourages Community Collaboration: By requiring that any modifications or enhancements to the code be released under the same license, the AGPL promotes a collaborative environment. This provision encourages developers to share their improvements with the community, fostering innovation and allowing others to benefit from their contributions. It helps build a rich ecosystem of shared knowledge and advancements.
- Protection Against SaaS Exploitation: The AGPL addresses a specific concern related to the software-as-a-service (SaaS) model. Without the AGPL's requirements, organizations could potentially use AGPL-licensed software to power their services without disclosing the modified source code to users. The AGPL ensures that if the software is used over a network to provide a service, the source code must be made available, thus preventing exploitation of the code in a closed, proprietary manner.
- Alignment with GNU GPL Compatibility: The AGPL is designed to be compatible with the GNU GPL. This compatibility allows developers to combine AGPL-licensed code with other GPL-licensed code, creating a broader pool of available software components. It enables the integration of AGPL-licensed software with existing GPL-licensed projects, promoting code reuse and collaboration.
- Community and Legal Support: The AGPL is a widely recognized and respected license in the open-source community. Its legal terms and conditions have been tested and upheld in various jurisdictions, providing a strong legal foundation for projects using the license. Additionally, because the AGPL is part of the GNU project, it benefits from the support and expertise of the Free Software Foundation (FSF) and the larger GNU community.

5. Folder structure

Folder structure refers to the organization and hierarchy of directories (folders) within a software project. A well-designed folder structure helps developers and collaborators navigate and manage the project's files and resources effectively. It provides a logical and consistent framework for organizing different types of files, such as source code, documentation, configuration files, assets, and test files. A good folder structure improves code maintainability, facilitates collaboration, and makes it easier to locate specific files or modules within the project. Common practices include organizing files by functional modules, layers (e.g., presentation, business logic, data access), or file types (e.g., source code, assets, tests). Establishing a clear and intuitive folder structure from the beginning of a project can contribute to its long-term scalability, ease of development, and overall project organization.
The folder structure of our Github is the following:

```
./VLA-252A/
├── LC_BPF.py
├── LCSecOrd_BPF.py
```

```
├── LCThirdOrd_BPF.py
├── main.py
├── RC_hpf.py
├── RC_lpf.py
├── test
│   ├── test_LC_BPF.py
│   ├── test_LCSecOrd_BPF.py
│   ├── test_LCThirdOrd_BPF.py
│   └── test_VLA251.py
├── utils
│   ├── new_plot_freqz.py
│   └── octave_distance.py
├── VLA251_GUI.py
└── VLA251.py
```

6. Software tools

The success of any software development project depends on the tools and technologies that are used to develop it. In our project, we will be leveraging a combination of Python libraries and development tools to create a robust and scalable solution. Specifically, we will be using the following Python libraries: NumPy, SciPy, Matplotlib, PyQT5, PySoundDevice, and PyWDF. These libraries will enable us to model and simulate wave digital filter circuits, as well as provide a range of Application Programming Interfaces (APIs) for accessing desktop and mobile system features. Additionally, we will be using development tools such as Visual Studio Code and Git, which will enable us to manage version control, perform code reviews, and collaborate effectively as a team. By utilizing these tools and technologies, we aim to create a high-quality software solution that meets the project requirements and delivers value to our stakeholders.

6.1 Software Development Tools

a. **Python:** Python is a high-level, interpreted programming language that is widely used for general-purpose programming, web development, data analysis, and artificial intelligence. It has a simple and easy-to-learn syntax. Python is free and open-source software and is supported by a large community of developers.
b. **Git**: *Git* is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
c. **Real time processing:** Real-time processing refers to the processing of data as it is generated or received, without any delay. This is important in applications where data needs to be processed in real-time, such as audio and video processing, financial trading, and monitoring systems.
d. **Algorithm implementation:** Algorithm implementation refers to the process of writing code to execute an algorithm. Algorithms are a set of instructions that are used to solve a particular problem or perform a specific task. Implementing an algorithm involves translating the algorithmic solution into a programming language, such as Python, and writing code that can execute the algorithm. Algorithm implementation requires a deep understanding of the problem domain and programming concepts such as data structures and control flow.

6.2 Python Libraries:

e. **Pywdf**[2]: *Pywdf* is designed to work with digital audio signals and can help users to create and test audio circuits. *Pywdf* provides a convenient way to create digital audio circuits, and it is an excellent tool for audio engineers, researchers, and hobbyists.

f. **Sounddevice**[3]: Play and Record Sound with Python. This library provides bindings for the PortAudio library and some convenience functions that allow users to play and record audio signals stored as *NumPy* arrays. Python-sounddevice is an easy-to-use library that can be useful for creating audio applications or for testing audio algorithms.

g. **PyQt5**[4]: *PyQt5* is a library that provides Python bindings for the Qt application toolkit. Qt is a set of cross-platform C++ libraries that implement high-level APIs for accessing various aspects of modern desktop and mobile systems. With *PyQt5*, Python developers can create desktop applications that run on different platforms, including Windows, Linux, and macOS. *PyQt5* is a powerful library that can be used for creating graphical user interfaces (GUIs), multimedia applications, and more.

h. **Numpy**[5]: *NumPy* is a powerful Python library for scientific computing. It provides an efficient interface for working with multi-dimensional arrays and matrices. *NumPy* is widely used in scientific research, data analysis, and machine learning. It provides various functions for numerical computations, such as mathematical functions, linear algebra, random number generation, and more. *NumPy* is a fundamental library for many Python-based scientific computing applications.

i. **Scipy**[6]**:** *SciPy* is a library that is built on top of *NumPy* and provides additional functionality for scientific computing. It provides modules for optimization, integration, interpolation, linear algebra, and more. *SciPy* is a powerful library that can be used for scientific research, data analysis, and machine learning.

j. **Tkinter**[7]**:** Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

6.3 Applications

k. **LTspice**[8]: *LTspice* is a free circuit simulation software designed by Linear Technology Corporation. It allows users to design and simulate analog circuits quickly and easily. LTSpice can be used to simulate a variety of circuits, including filters, amplifiers, and power supplies. It is widely used in the electronics industry and is an essential tool for circuit design and analysis.

l. **Visual Studio Code**[9]**:** *Visual Studio Code* is a popular open-source code editor developed by Microsoft. It provides a wide range of features to enhance the coding experience, such as code highlighting, code completion, and debugging tools. *Visual Studio Code* is highly extensible, and it can be customized with various extensions to enhance its functionality.

---

[2] https://github.com/gusanthon/pywdf
[3] https://python-sounddevice.readthedocs.io/en/0.4.6/
[4] https://pypi.org/project/PyQt5/
[5] https://github.com/numpy/numpy
[6] https://github.com/scipy/scipy
[7] https://wiki.python.org/moin/TkInter
[8] https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html
[9] https://code.visualstudio.com/

7. Collaborative coding strategy

Collaborative coding practices are critical for the success of any coding project, as they enable team members to work together seamlessly, resulting in a more efficient and effective project development process. In our project, we will be implementing several key coding practices to ensure that we are working together in the most effective way possible. These practices will include version control with Git, pull requests, code reviews, live sharing, and issue tracking. By leveraging these practices, we aim to produce high-quality code that meets the project's requirements and exceeds our stakeholders' expectations.

1. **Version Control**: Git is a version control system that allows developers to keep track of changes made to code files. With Git, developers can create branches for different features or bugs, and merge changes made by different team members into the main branch. Git is also integrated with GitHub, allowing developers to easily collaborate on code.
2. **Pull Requests:** Pull requests are a feature of GitHub that allows developers to propose changes to the main codebase. Developers can create a pull request for a specific branch and ask for feedback and approval from other team members before merging it into the main branch.
3. **Code Reviews:** Code reviews are an important part of the development process that ensures the quality and consistency of code. With Visual Studio Code, developers can use extensions like CodeStream or GitHub Pull Requests to review code changes and provide feedback to other team members.
4. **Live Share**: Visual Studio Code also provides a Live Share extension that allows team members to collaborate in real-time on the same codebase. This feature is especially useful for pair programming or when working on complex features that require multiple team members to work together.
5. **Issue Tracking:** GitHub provides an issue tracking system that allows team members to create and assign tasks, track progress, and communicate about specific issues or bugs. This feature can help keep the project organized and ensure that everyone is working towards the same goals.

8. Team responsibilities

Responsibility	Description	Person in charge
Code reviewer	A person responsible for examining and evaluating code written by developers to ensure its quality, adherence to coding standards, and identify any potential bugs or issues.	Du
Algorithms logic reviewer	An individual who assesses and verifies the correctness, efficiency, and effectiveness of algorithms used in software development, ensuring they meet the required logic and perform as expected.	Du, Sofia, Tommaso
Developers	Professionals who write, modify, and maintain software programs or applications using programming languages and tools. They are responsible for implementing the design and functionality of software based on project requirements.	Sofia and Tommaso
QA	The team or process responsible for ensuring that software meets specified quality standards. QA professionals conduct various testing activities, identify	Pablo and Chris

	bugs or defects, and provide feedback to improve the software's reliability, usability, and overall quality.	

SOFTWARE REQUIREMENT SPECIFICATIONS

1. Overview

Introduction:
The purpose of this document is to outline the requirements for the development of a software version of the GEQ Langevin 252A, a high-quality equalizer for professional audio applications. The software will utilize Wave Digital Filters to achieve the desired sound characteristics and will be compatible with Windows and Mac operating systems.
To provide additional clarity, the standalone version of the GEQ Langevin 252A will be prototyped in Python, using Wave Digital Filters to achieve the desired sound characteristics. This approach will allow for offline development, ensuring the software can be refined and optimized before being released to users.

System overview:
The system will consist of a digital signal processing engine that simulates the analog circuitry of the Langevin 252A equalizer. The engine will utilize Wave Digital Filters to accurately reproduce the frequency response and phase shift characteristics of the original hardware. The software will be compatible with Windows and Mac operating systems. The prototype will be developed using Wave Digital Filters in Python and will include a graphical user interface for parameter adjustment and visualization as Figure 1 shows.

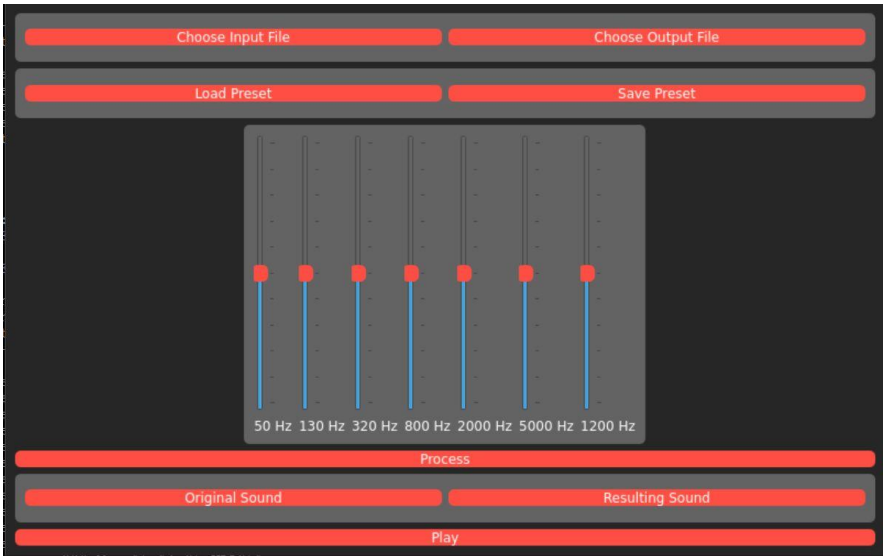


Figure 27: Graphical interface

2. Requirements

Functional Requirements:
The functional requirements of a software system are an essential component of its success. They represent the specific functionalities and capabilities that the software must possess to perform its intended purpose. In essence, they define the core specifications that the software must meet to fulfill its objectives. These requirements are critical to the overall success of the system, as they outline the key functionalities that the software must perform to meet the needs of its users and stakeholders. By clearly defining the functional

requirements of the software, developers and stakeholders can ensure that the system is designed to meet the needs of its users and perform at a high level of efficiency and effectiveness. The functional requirements of the software system includes:

- To create the user interface for the Langevin 252A equalizer software, several requirements must be met. By this we mean that It should support at least seven equalization bands with fixed frequency and variable gain control. Furthermore, the front-end should include a display using PyQT.
- The back-end will require several libraries, including soundfile, essentia, matpltlib, sns, numpy, and sounddevice.
- The software must accurately replicate the frequency response of the Langevin 252A equalizer.
- The interface must allow for loading an audio file from the system and feature a play/stop button.
- Users should be able to save and recall presets and manage them, including the ability to save, recall, and delete presets, which can be managed locally once the application is run.
- A bypass function should allow for A/B comparison of processed and unprocessed audio signals.
- A graphical representation of the equalizer bands with adjustable parameters should be included.

Non-Functional Requirements:
Non-functional requirements are a critical aspect of any software system, as they define the key parameters that govern how the system should operate beyond its basic functionalities. These requirements cover a wide range of important aspects, including performance, usability, security, compatibility, and more. By defining and prioritizing non-functional requirements, developers and stakeholders can ensure that the software system is not only functional but also optimized to deliver an exceptional user experience across a wide range of important parameters. The non-functional requirements of the software system includes:

- The software will have an efficient latency to minimize the delay between processing the audio and hearing the result.
- The software shall be stable and reliable, with no crashes or unexpected behavior.
- The software shall be optimized for efficient CPU usage, to minimize the impact on system performance.

3. Structure

System Architecture:
System architecture refers to the overall design and organization of a software system, including its components, modules, and the relationships between them. The system architecture plays a crucial role in determining the overall performance, scalability, and maintainability of the software system, and it must be carefully planned and implemented to ensure that the system operates as intended. The system architecture of the software system includes:

- The digital signal processing engine, which will utilize Wave Digital Filters to simulate the analog circuitry of the Langevin 252A equalizer.
- The graphical user interface, which will allow users to adjust parameters and visualize the frequency response.

Data Model:
The data model of a software system refers to the way in which data is organized, stored, and managed within the system. The data model plays a crucial role in determining the overall performance and scalability of the software system, and it must be carefully planned and implemented to ensure that the system can handle the volume and complexity of data that it is required to manage. The data model of the software system includes:

- The equalizer settings, which will be stored in preset files and recalled by the user.
- The audio input and output signals, which will be processed by the digital signal processing engine.
- The GUI parameters, which will be adjusted by the user through the graphical user interface in line with the abilities and behavior of the Langevin 252A equalizer.

Error handling:
Error handling refers to the processes and procedures that are used to detect, report, and manage errors or issues that may occur within a software system. The error handling is a critical aspect of the software system, as it directly impacts the reliability and robustness of the system under different conditions and scenarios. Effective error handling can help to minimize downtime and disruptions to the system's operation, while also ensuring that users can continue to work effectively and efficiently within the system.
The error handling of the software system includes:

- The software shall display error messages in the event of unexpected behavior or errors, explaining in a pop-up-error-style message window a few details of the error using a user-friendly language.
- The software shall provide a mechanism for reporting errors to the development team for investigation and resolution, an instant approach to tackle this issue would be to provide some kind of email to the development team..
- The software shall include robust error checking and input validation to prevent crashes or unexpected behavior, such

Constraints
Constraints refer to the limitations or restrictions that are imposed on a software system due to factors such as technical or resource limitations, regulatory requirements, or other external factors. Effective management of constraints can help to optimize system performance and ensure that the system can operate reliably and consistently under a wide range of conditions and scenarios.

- The software must operate with low latency, to ensure real-time processing of audio signals.
- The software must be stable and reliable, with no crashes or unexpected behavior.
- The software must be compatible with multiple operating systems.
- The software must be optimized for efficient CPU usage to minimize the impact on system performance.
- The software must adhere to industry-standard security practices to prevent unauthorized access to user data.
- The software must meet all relevant industry standards for audio quality and processing accuracy.
- The software must be developed within the given timeline.
- The software must be easily maintainable and upgradable in the future to support new features or bug fixes

4. UI/UX and GUI

User experience (UX)
User experience (UX) encompasses all aspects of a user's interaction with a software system, including its visual design, usability, accessibility, and overall satisfaction. A positive user experience is essential to the success of a software system, as it drives user engagement, retention, and advocacy.
Our system's UX should be intuitive, efficient, and enjoyable for users to use. This means providing clear and concise instructions, easy-to-use controls, and a responsive design that adapts to different screen sizes and resolutions. We should also consider factors such as color contrast, font size, and keyboard accessibility to ensure that our system is usable by as many people as possible.

User Interface (UI):
The user interface (UI) of a software system refers to the visual and interactive components that enable users to interact with the system. A well-designed user interface is essential to the success of a software system, as it directly impacts the usability and overall user experience of the system.

- Our system's UI should be clear, simple, and intuitive, providing users with easy-to-use controls and a visually appealing design. We should consider factors such as color scheme, typography, and layout to create a cohesive and aesthetically pleasing user interface.
- In terms of functionality, our system's UI should allow users to easily adjust equalizer settings using faders and buttons. We should ensure that the controls are responsive and accurately reflect the changes made to the audio signal in real-time. We should also provide clear instructions on how to save and recall presets, load and save audio files, and perform A/B comparisons.
- To enhance the user's experience, we could include additional features such as a spectrum analyzer or frequency response curve that visually display the effect of the equalizer settings.
- We could also include tooltips or contextual help to assist users in understanding the different controls and how to use them effectively.

Our system has 7 faders, each dedicated to one of the 7 bands of frequencies (50Hz, 130Hz, 320Hz, 800Hz, 2000Hz, 5000Hz, and 12500Hz). With the ability to attenuate or boost frequencies from -8dBs to +8dBs. Our system also includes buttons for saving and loading audio files, as well as a preset selection, A/B comparison, and On/Off button for easy bypassing. We might also need to add a settings button to allow the user to easily customize the sample rate, bit depth, audio i/o, etc.

User interaction:
User interaction refers to the way in which users engage with a software system to perform specific tasks or functions. The user interaction is a critical aspect of the software system, as it directly impacts the usability and overall user experience of the system. The user interaction of the software system includes:

- Adjusting equalizer settings through the graphical user interface.
- Saving and recalling presets.
- Loading and saving audio files.
- Bypassing the equalizer.
- A/B comparison.
- Monitoring the frequency response of the equalizer.
- Choose between working on mono & stereo.

Use Cases:
Use cases refer to specific scenarios or situations in which a software system is intended to be used to achieve a specific goal or objective. The use cases are a critical aspect of the software system, as they directly impact the relevance and usefulness of the system for its intended users. The use cases of the software system include:

- Audio engineers and musicians can use the software to apply precise and musical equalization to their audio recordings.
- Mixing and mastering engineers can use the software to enhance the tonal balance and clarity of their mixes.
- Post-production professionals can use the software to match the tonal characteristics of audio recordings captured in different environments.
- Live sound engineers can use the software to apply EQ to their live sound mixes, improving the sound quality of the performance.

Assumptions and Dependencies:

Assumptions and dependencies refer to the external factors and conditions that are required for a software system to function as intended. The assumptions and dependencies are a critical aspect of the software system, as they directly impact the system's ability to deliver the expected functionality and performance to its users. The following assumptions and dependencies have been identified for the software system:

- The software system assumes that the user has a basic understanding of equalization principles.
- The software system is dependent on the operating system to provide audio input and output.
- The software system is dependent on third-party libraries for implementing Wave Digital Filters and for GUI development.

ETHICAL CONSIDERATIONS

1. Introduction

Designing audio software intended to emulate the Langevin EQ 252A Equalizer involves a range of ethical considerations. Ensuring accuracy and transparency in the representation of the original device with respect to intellectual property rights and promoting accessibility are among the key considerations. Additionally, prioritizing user experience and inclusivity are essential aspects to address. With consideration to these aspects of the design and development process the software can offer a high-quality experience while upholding integrity, user satisfaction and responsible research and development practices.

2. Accuracy and Transparency

It is important to ensure that the digital version of an analog device should accurately represent the original device.

It is important to acknowledge though that sometimes compromises may arise due to time constraints or other limitations. These compromises may result in deviations from the original design, and It is crucial to address such deviations and clearly document them to avoid misleading users.

3. Intellectual Property

The Langevin EQ 252A, and many analog devices with similar circuitry already exist as products in the digital realm. As such, important considerations should be made to ensure that intellectual property rights are respected in order to avoid infringing on copyrights of existing devices.

Also, It is crucial to distinguish between patents and inventions within our specific context; while patents protect the specific implementation and design of a device, inventions encompass broader concepts and ideas. Even if a patent has expired, the underlying invention may still be protected by intellectual property rights.

Furthermore, when utilizing code repositories like pywdf, it is essential to review the licensing and usage terms associated with the code and any accompanying documentation. Some repositories may provide open-source code, allowing for unrestricted use and modification, while others may have specific licensing requirements that must be adhered to.

4. Accessibility

4.1 Software Accessibility

Software should be designed with consideration of users with disabilities. The graphical user interface should be easy to read and interact with. The software and its consequent updates should be able to be obtained at a cost-free, such as open-source software or free versions with upgrade options.

Additionally, ensuring compatibility with major operating systems and supporting different bit architectures will help reach a larger user base.

4.2 Hardware Accessibility

The software in question will be fully digital. This may create issues of accessibility for users who do not own a computer. Additionally the software requires the use of an audio input device such as a microphone. Microphones can be costly, and as such, options for digital audio inputs, such as pre-existing audio files should be included.

5. User Experience and Inclusivity

The user experience should be designed in such a way that accommodates different skill levels and preferences. The diverse needs of uses should be considered including those with visual, hearing or cognitive impairments.

6. User Data Privacy/Protection

All relevant personal information should be stored in a secure way. When possible, personal identifiers should be removed. User information should be stored in a secure database with protections against data attacks. The software developer should only collect and store user information when it is vital to allow for successful running of the program.

While data privacy remains crucial, It is still important to mention that in applications like our one, which are not directly related to machine learning models, the focus is typically on providing specific functionalities or services to users rather than analyzing and leveraging user data. This means that our application may not require extensive personalization or customization based on individual user data, instead, the main focus is on delivering predefined features, executing specific tasks, or facilitating certain interactions.

SOFTWARE EVALUATION

1. Introduction:

The purpose of this evaluation document is to conduct a comprehensive assessment of the performance, features, and overall suitability of the VLA-252A software. This evaluation aims to provide an objective analysis that will assist in the decision-making process regarding the adoption, usage, and potential integration of the software in a professional audio production environment.
The VLA-252A software is a digital audio processing tool designed to replicate the characteristics and behavior of the vintage Langevin 252A hardware unit through the use of wave digital filters. By employing advanced analog modeling techniques, specifically wave digital filters, the software strives to faithfully recreate the unique sound and response of the original hardware.

The evaluation of the VLA-252A aims to achieve the following objectives and goals:

- Evaluate the accuracy and fidelity of the software's analog modeling capabilities.
- Assess the software's ability to reproduce the sonic characteristics and tonal qualities of the original Langevin 252A hardware unit.
- Measure the computational efficiency and real-time processing capabilities of the software.
- Explore and analyze the available features and functionalities of the software, such as parameter controls, signal routing options, and user interface design.
- Compare the VLA-252A to other similar analog modeling plugins or hardware alternatives available in the market.

By conducting this evaluation, we aim to provide a comprehensive assessment of the VLA-252A software, enabling informed decision-making regarding its potential adoption, usage, and integration within professional audio production environments.

2. Evaluation Criteria:

The evaluation of the VLA-252A software involves a meticulous assessment of various criteria to determine its performance, features, and overall suitability for professional audio production. The following evaluation criteria have been identified as key factors in analyzing the software:

- Accuracy and Fidelity: This criterion focuses on the software's ability to accurately emulate the vintage Langevin 252A hardware unit. It involves evaluating the precision and faithfulness of the software's analog modeling capabilities in reproducing the unique sonic characteristics, tonal qualities, and dynamic behavior of the original hardware.
- Sonic Reproduction: Under this criterion, the software's effectiveness in reproducing the sonic qualities of the original hardware is evaluated. This includes assessing its ability to capture the warmth, coloration, harmonic content, and overall sonic signature that made the Langevin 252A highly regarded in professional audio circles.
- Computational Efficiency and Real-Time Processing: This criterion addresses the software's efficiency in utilizing computational resources and its ability to perform real-time processing tasks. The evaluation focuses on measuring the software's resource demands, such as CPU usage and memory requirements, while also assessing its ability to handle complex audio processing tasks without significant latency or performance issues.

- Feature Set and Functionality: Under this criterion, the evaluation examines the features and functionalities offered by the VLA-252A software. This includes an analysis of its parameter controls, signal routing options, user interface design, and any additional tools or capabilities that enhance the user experience and facilitate creative audio processing workflows.

3. Methodology:

The evaluation of the VLA-252A software follows a structured and systematic methodology to ensure a thorough and reliable assessment of its performance, features, and overall suitability for professional audio production. The following methodology outlines the approach taken to conduct the evaluation:

- Testing and Analysis: Extensive testing was conducted to evaluate the software's accuracy, fidelity, sonic reproduction, and computational efficiency. This involved running the software in different scenarios, applying various audio sources, and assessing its performance based on predefined criteria and metrics. Real-time processing capabilities were measured, and the software's response to different parameter adjustments and signal routing options was analyzed.
- Comparative Analysis: The VLA-252A software was compared with other similar analog modeling plugins or hardware alternatives available in the market. This involved benchmarking against established industry standards and conducting side-by-side comparisons to identify the unique strengths, advantages, and potential areas for improvement of the VLA-252A software.
- Documentation Review: The software's technical specifications, user guides, and any relevant documentation were thoroughly reviewed to gain insights into its features, functionalities, and intended usage. This allowed for a comprehensive understanding of the software's capabilities and provided context for the evaluation.

4. Data collection methods:

The data collection process included a combination of methods to gather comprehensive insights and feedback on the VLA-252A software. The following methods were employed:

- Testing and Measurements: Quantitative data was collected through rigorous testing procedures. Measurements, such as computational efficiency, latency, and processing accuracy, were recorded to assess the software's performance objectively. User

5. Evaluation Results:

The evaluation of the VLA-252A software has generated comprehensive and insightful results that provide a detailed analysis of its performance, features, and overall suitability for professional audio production. The following sections present a summary of the key findings and outcomes derived from the evaluation process.}

	Accuracy and Fidelity	Sonic Reproduction	Computational Efficiency and Real-Time Processing	Feature Set and Functionality
Python Prototype	9	9	1	4
Supercollider	9	9	10	10

Python Prototype:

The evaluation results of the python prototype showcased exceptional accuracy and fidelity in terms of sonic reproduction. We were able to achieve a remarkably close sound to our desired outcome, highlighting the prototype's potential in delivering high-quality audio.

However, one significant drawback surfaced during the evaluation process - the computational efficiency of the prototype was insufficient for real-time performance. This limitation impacted the overall feature set and functionality of the prototype, falling short of our initial expectations. While the prototype could potentially serve as a standalone version of the plugin, it remains less convenient due to its inability to operate in real time.

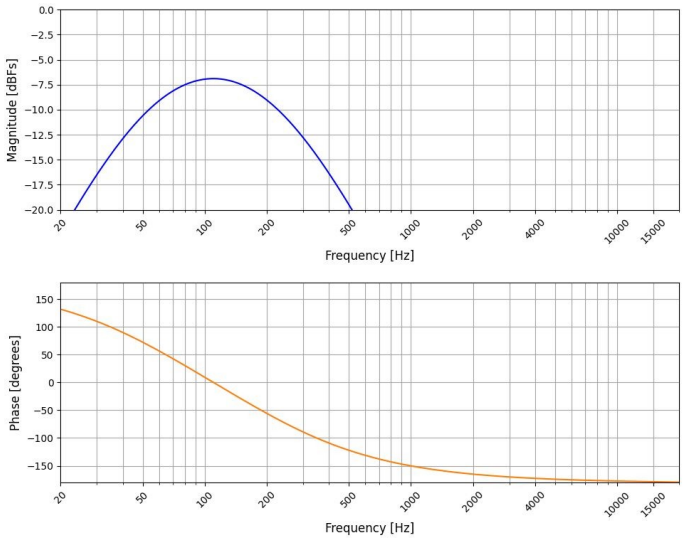


Figure 28: Example of the band-pass filter working at 230 hz.

Real-time with Supercollider

The evaluation results showed that it may not be able to perfectly reproduce the distinctive sound of the Langevin 252A due to the unavailability of its schematics. However, it does possess a commendable ability to accurately reproduce the sonic characteristics of a bandpass filter. While it may not replicate the exact sound signature of the Langevin 252A, it still offers a satisfying audio experience.

Moreover, this plugin has the potential to deliver high-quality audio output. It is designed to ensure that the audio signals passing through it are faithfully reproduced, maintaining their clarity, dynamics, and fidelity. In addition, it offers sufficient computational efficiency to enable real-time performance. It can handle audio processing tasks without significant latency or lag. This capability is especially important for applications where immediate audio feedback or live performances are involved.

6. Strengths and Weaknesses:

The evaluation of the VLA-252A software has identified several strengths and weaknesses that provide a comprehensive understanding of its capabilities and limitations. This section presents an overview of the key strengths and weaknesses derived from the evaluation process, shedding light on the software's positive aspects and areas that may require improvement.

One of the main weaknesses is that basic SuperCollider programming knowledge is required from the users. Moreover it is rare to find analogue models of classic audio processors in SC since the use of

the program that users perform emphasizes the algorithmical aspect of control, rather than mixing methods where the 252A is mainly used.

7. Recommendations:

Based on the analysis conducted, we would like to make the following recommendations for the prototyping and implementation of the project.

First and foremost, it is advisable to conduct the prototyping phase using the C++ programming language. While the PyWDF library provides a convenient way to work with Python, it is important to consider the real-time processing requirements of the specific circuit being utilized. In this case, the circuit employs an R-Type adaptor which significantly hampers the processing speed when using Python. Processing just one second of audio can take up to 13 seconds, indicating a substantial performance bottleneck. Therefore, opting for C++ would be more suitable, as it offers better performance and real-time capabilities.

Furthermore, I recommend exploring the use of a dedicated framework like JUCE for implementing the real-time version of the project. JUCE is specifically designed for audio and DSP applications, making it an excellent choice for this type of project. Leveraging JUCE's features and functionalities will likely result in more efficient and streamlined development, ultimately improving the real-time performance and overall user experience of the final product.

By following these recommendations, you can ensure that the prototyping is conducted in a language (C++) that is better suited for real-time processing, and that the implementation phase benefits from the capabilities of a specialized framework like JUCE, leading to improved performance and a more optimized end product.

8. Conclusion:

In conclusion, the evaluation of the VLA-252A software has provided us with valuable insights into its performance, features, and suitability for professional audio production. Through this project, we have identified both strengths and areas for improvement, shaping our final conclusions.

The utilization of R-type adaptors has proven to be an invaluable tool for modeling analog circuits. Their flexibility allows for the implementation of diverse circuit topologies and the use of wave digital filters. However, it is important to note that the drawback of R-type adaptors lies in their sluggish processing speed, as highlighted in our earlier recommendations. To leverage the power of Python in projects of this nature, it is imperative to explore and implement faster alternatives that can meet the real-time processing demands more effectively. By doing so, we can unlock the potential of Python for future endeavors.

Furthermore, our exploration of wave digital filters has showcased their effectiveness in achieving realistic analog circuit modeling. When correctly implemented and utilized in real time, these filters can deliver remarkably similar analog sound. It is essential, however, to

acknowledge that while digital sound can strive for authenticity, it will never fully replicate the nuances of analog sound.

Looking ahead, we recognize the scope for future improvements in the circuit used. Given more time, we could have sought out a circuit that closely matched our requirements, ultimately yielding better results. Continual refinement and fine-tuning of the circuit will contribute to the overall enhancement of the project.

In summary, the evaluation of the VLA-252A software has provided us with valuable insights and conclusions. By addressing the challenges posed by R-type adaptors, exploring the potential of wave digital filters, and refining the circuit in future iterations, we can further optimize the performance and realism of our projects in professional audio production.

APPENDICES

The appendices section of this evaluation document provides supplementary information and supporting materials that complement the main findings and analysis. This section offers additional details and resources that may be beneficial for users, decision-makers, and stakeholders seeking a deeper understanding of the evaluation process and its outcomes.

- Manual
- Source code
- Prototype & Real-Time Demos

REFERENCES

[1] G. Louie and G. White, "The Audio Dictionary," University of Washington Press, 2005, p. 140. [Online]. Available: https://books.google.com/books?id=DulVm8t88QkC&q=the+audio+dictionary. ISBN 9780295984988.

[2] Mackie, "Extraordinary EQ from Extraordinary Engineers," [Online]. Available: https://web.archive.org/web/20131202221704/http://www.mackie.com/technology/perkinseq.html

[3] G. Massenburg, "Parametric Equalization," 1972, [Online]. Available: https://web.archive.org/web/20110714044154/http://www.massenburg.com/gml/downloads/parametric_paper.pdf.

[4] G. Massenburg, "A Short History of Graphic and Parametric Equalization," [Online]. Available: https://intelligentsoundengineering.wordpress.com/2016/02/22/a-short-history-of-graphic-and-parametric-equalization/.

[5] B. Shenoi, "Introduction to digital signal processing and filter design," Choice Reviews Online, vol. 43, no. 08, pp. 43–4691, 2006. https://doi.org/10.5860/choice.43-4691.

[6] R. Raut and M. K. Swamy, "Modern Analog Filter Analysis and Design," Wiley-VCH Verlag GmbH & Co. KGaA eBooks, Wiley, 2010. https://doi.org/10.1002/9783527631506.

[7] A. R. Williams and F. R. Taylor, "Electronic filter design handbook," 1981.

[8] L. J. Jackson, "Digital filters and signal processing," 1985.

[9] L. D. Paarmann, "Design and Analysis of Analog Filters: A Signal Processing Perspective," 2001.

[10] K. J. Werner, "Virtual Analog Modeling of Audio Circuitry Using Wave Digital Filters," Stanford University, Stanford, 2016.

[11] O. Loyola-González, "Black-Box vs. White-Box: Understanding Their Advantages and Weaknesses From a Practical Point of View," IEEE Access, vol. 7, pp. 154096–154113, 2019. https://doi.org/10.1109/access.2019.2949286.

[12] "Wave Digital Filters," [Online]. Available: https://ccrma.stanford.edu/~jos/pasp/Wave_Digital_Filters_I.html.

[13] E. Damskägg, L. Juvela, and V. Välimäki, "Real-Time Modeling of Audio Distortion Circuits with Deep Learning," in Sound and Music Computing Conference, 2019, pp. 332–339

[14] K. Thayer, "How Does Reverse Engineering Work?," IEEE Global Spec, [Online]. Available: https://insights.globalspec.com/article/7367/how-does-reverse-engineering-work.

[15] S. W. Smith, "Nonlinear Audio Processing," DSPGuide, [Online]. Available: https://www.dspguide.com/ch22/7.htm.

[16] "In Memoriam," Audio Engineering Society Journal, [Online]. Available: https://www.aes.org/aeshc/docs/jaes.obit/JAES_V18_6_PG730.pdf.

[17] "True 252A," Kazrog, [Online]. Available: https://kazrog.com/products/true-252.

[18] "URS M Series Motor City Equalizer," Unique Recording Software, [Online]. Available: https://www.ursplugins.com/ursMEQ.php.

[19] "Abbey Road REDD Consoles," Waves, [Online]. Available: https://www.waves.com/plugins/redd.

[20] "Neve 1073 Collection," Universal Audio, [Online]. Available: https://www.uaudio.com/uad-plugins/channel-strips/neve-1073-collection.html.

[24] "Chow Tape Model," ChowDSP, [Online]. Available: https://chowdsp.com/products.html.

[25] "Dynamic Plate Reverb," Physical Audio, [Online]. Available: https://physicalaudio.co.uk/products/dynamic-plate-reverb/.

[26] M. Ducceschi and C. J. Webb, "Plate reverberation: Towards the development of a real-time physical model for the working musician," in International Congress on Acoustics ICA 2016, 2016. [Online]. Available: https://www.researchgate.net/publication/306936091_Plate_reverberation_Towards_the_development_of_a_real-time_physical_model_for_the_working_musician.

[27] K. Renner and S. C. Gupta, "Digital formant speech synthesis utilizing wave digital filters," Computers & Electrical Engineering, vol. 1, no. 2, pp. 211–225, 1973. https://doi.org/10.1016/0045-7906(73)90015-3.

[28] K. Mostafa and M. Sobhy, "Lumped/distributed single-pulse matched filter for radar applications based on wave digital filters," in International Symposium on Circuits and Systems, 1999. https://doi.org/10.1109/iscas.1999.778847.

[29] C. Giguère, "Speech Processing using a Wave Digital Filter Model of the Auditory Periphery," International Research Journal of Engineering and Technology IRJET, 1993. [Online]. Available: https://www.academia.edu/34468256/Digital_Filters_for_Radar_Signal_Processing.

[30] P. Raffensperger, "Toward a wave digital filter model of the Fairchild 670 limiter," 2012.

[31] "Virtual Analog Modeling of Audio Circuitry Using Wave Digital Filters - ProQuest," 2016

[32] J. Chowdbury and Stanford University, "A comparison of virtual analog modelling techniques for desktop and embedded implementations," 2020.

[33] S. W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing," 1997.

[34] A. Fettweis, "Canonic realization of ladder wave digital filters," in IEEE Transactions on Circuit Theory, vol. 22, no. 4, pp. 385-388, 1975. https://doi.org/10.1109/TCT.1975.1083732.

[35] A. Bernandini, "Toward the Wave Digital Real-Time Emulation of Audio Circuits with Multiple Nonlinearities," [Online]. Available: https://www.researchgate.net/figure/WD-structure-corresponding-to-the-circuit-in-Fig-1_fig2_344628770, 2020.

[36] “Analog modeled bundles: Waves,” waves.com, [Online]. Available: https://www.waves.com/bundles/analog-models (accessed May 23, 2023)

[37] “Circuit emulation in software,” Universal Audio, [Online]. Available: https://www.uaudio.com/blog/circuit-emulation-software/(accessed May 23, 2023).