Introduction to the mathematics of Virtual Analog modeling

BSc Thesis in Applied Mathematics

Ték Róbert Máté

Supervisor: Dr. Gergó Lajos

Department of Numerical Analysis

Eötvös Loránd University

Faculty of Science

2025

to Anna

Contents

Introduction

Many different effects pedals and guitar amplifiers exist for electric guitar players to choose from. These devices originate from the mid-20th century, and have had great influence on decades of popular music. They traditionally employ analog circuitry to amplify and process the electric guitar's signal in various ways. Creating digital emulations of such electrical circuits is highly desirable. Analog circuits can be damaged by mechanical impacts, humidity or temperature changes. They degrade over time and require occasional maintenance. Guitar amplifiers are notoriously heavy, and thus their transportation can be a challenge. Some guitar amplifiers need to operate at very high volumes to sound best, which is not always feasible e.g. in an apartment or at night. A handful of digital devices can replace entire guitar rigs and require less maintenance while enabling features that would be impossible or very cumbersome to implement with traditional gear. Every analog circuit is unique and has a unique sound e.g. due to availability of electrical components, manufacturing tolerances or other imperfections. Vintage devices may no longer be in production. Therefore it is important to capture and preserve the tonality of these devices with careful attention to detail for future generations of musicians. The process of creating digital emulations of analog audio circuitry is often referred to as *Virtual Analog* modeling. The goal of this thesis is to offer a glimpse into the mathematical foundation of Virtual Analog modeling through a concrete example; deriving a real-time software emulation of the Ibanez TS808 Tube Screamer distortion pedal. The source codes for the emulation, and other related programs, are available on GitHub [1].

This work builds upon established results from relevant scientific disciplines such as physics or electrical engineering. While these results are rooted in deeper, more fundamental theories, their precise introduction is beyond the scope of this thesis. Instead, simplified yet mathematically coherent definitions will be used. For a more rigorous treatment of the underlying theories, the reader is referred to dedicated textbooks, such as [2, 3].

Though I tried my best to present relevant and up-to-date information on the topic, this thesis is not intended to be a guide to the best practices of digital signal processing.

All product names and trademarks are the property of their respective owners. Their use herein does not imply affiliation or endorsement.

Chapter 1

The Ibanez TS808 Circuit

The Ibanez TS808 is a so-called *overdrive pedal* designed to emulate the soft clipping of overdriven tube amplifiers. An amplifier is said to be overdriven when it is pushed beyond its ideal range of operation. For vacuum tube amplifiers, this produces a smooth, musical sounding distortion, which many listeners—and guitarists—find pleasing. Overdrive pedals, like the TS808, aim to recreate this tonal character at lower volumes, hence the name *Tube Screamer*.

The TS808 circuit can be thought of as the interconnection of several distinct electrical networks:

- input buffer,
- distortion stage,
- tone/volume stage,
- effects bypass switching,
- power supply, and
- output buffer.

Our primary concern will be toward the distortion and the tone/volume stages, which contribute most to the tonality of the pedal, while the other networks are of utilitarian nature and have little effect on the processed signal [4]. The schematic for these stages can be seen in Appendix A.

The clipping stage consists of an operational amplifier—or *op-amp* for short—in a non-inverting configuration, with some passive filtering and two antiparallel clipping diodes in the feedback loop. The resistor R_g and capacitor C_g form a simple high-pass filter, while the resistance R_f and capacitor C_f form a variable-frequency low-pass filter to "shape the amount of clipping and the frequency at which it occurs" [4]. This network is sometimes referred to as a *diode clipper with embedded low-pass filter*. In the real pedal, the resistance R_f actually consists of a 51kΩ resistor and a 0 to 500kΩ variable resistor—or *potentiometer*—in series, but, for the sake of simplicity, we can think of them as a single variable resistance.

The clipping stage feeds directly into the tone/volume stage, which consists of a fixed first-order low-pass filter, followed by an active tone control network centered around the op-amp labeled as 'OpTone'. Here, another potentiometer can be used to adjust the amount of high frequencies passed through to the output. Depending on the position of this potentiometer, the overall effect of the tone circuit can resemble:

- a second-order low-pass filter, attenuating high frequencies even more (potentiometer all the way to the 'bass' side), or
- a band-pass filter, which lets through more of the high frequencies (potentiometer all the way to the 'treble' side), or
- somewhere in-between [5].

The output level control at the end of this stage is just a simple voltage divider.

To create a real-time digital emulation of the TS808 circuit, we need to understand

a) the fundamentals of discrete-time signal processing,

b) how the components of the circuit interact to process an analog input signal, and

c) how to translate these interactions to discrete-time systems.

Circuit theory and *modified nodal analysis* can be used to understand *b*), while *a*) is answered by the theory of *discrete-time signals and systems.* Point *c*), however, is less straightforward. There are many different ways to approach this problem, all of them equally valid, and come with different strengths and weaknesses. The process described by *c*) is called *discretization.* In this thesis, we will look at several different discretization strategies as well as their mathematical background.

The clipping stage will be modeled using *white-box modeling*, that is, we have full knowledge of the reference circuit and its workings. The tone stage will be modeled using *black-box modeling*, where we will pretend that only the input-output relationship of the circuit is known to us.

Chapter 2

Understanding electrical circuits

To derive a software emulation of the Ibanez TS808 pedal, we need a mathematical framework for modeling electrical circuits. To make this thesis self-contained, this chapter outlines the basic theoretical elements of electrical circuits relevant to our digital modeling task. As such, readers not yet familiar with electrical systems shall be able to follow the material presented in this work.

2.1 Methodology

With thermodynamics, one can calculate almost everything crudely; with kinetic theory, one can calculate fewer things, but more accurately; and with statistical mechanics, one can calculate almost nothing exactly.

Wigner Jenő Pál

Electricity at a macroscopic scale is fully described by *classical electrodynamics*, however, direct application of its principles would be quite impractical for our purposes [2]. To bring forth more tractable mathematical models of electrical circuits, we must make certain simplifying assumptions, such as ideal operating conditions and perfectly manufactured eletrical components. Regardless, the resulting models have proven to be sufficiently accurate in predicting the behavior of many real-world circuits [2, 6].

2.2 Circuit Theory

Circuit theory is a special case of electromagnetic field theory with a focus on electrical networks. Circuit theory relies on the so-called *lumped circuit model*, which, together with the laws of electromagnetic field theory, result in a set of equations greatly reduced in complexity.

The following definition should not be considered ground thruth; it is merely my attempt at interpreting and formalizing the notion of lumped electrical networks.

Definition 1. The lumped circuit model assumes the following for an electrical network:

i) the components of the network are physically distinct and are connected only by wires,

ii) all conducting materials outside of components are ideal conductors, i.e. they possess no electrical resistance,

iii) electrical and magnetic energy is stored or converted to other forms of energy only within electrical components, i.e. no electric fields exist outside components and wires,

iv) there is no magnetic coupling between components of the network,

v) electrical effects happen instantaneously throughout the network,

vi) the net electric charge on every component of the network remains zero at all times.

A physical circuit may be considered lumped if said assumptions can be in any way *justified*, i.e. it can be theoretically or experimentally verified that the impact of the described phenomena on the circuit are *negligible.*

The above assumptions can be justified for analog audio circuits. Hence, from now on, we will consider electrical networks and components to be lumped. It follows that the lumped circuit model is concerned only with the *topology* of a network, and not its physical layout.

Definition 2. An *electrical component* is a device with two or more distinct conducting surfaces called *terminals.* Terminals act as connection points. Components may be connected via conducting *wires.* An *electrical network* is the interconnection of electrical components. A maximal set of connected terminals within a network is called a *node.*

The notion of *electrical circuits* is a bit more nuanced, and will be defined shortly. The schematic symbols of components of interest for this thesis can be seen in Appendix A.

It may be tempting to visualize the topology of an electrical network as a graph (V, E), with V corresponding to components, and E corresponding to connections or wires. Indeed, circuit schematics do resemble such a graph, with a single difference; the wires of a circuit node are united into a single hyperedge. Some examples can be found in Appendix A. However, it is more useful to define the topology of a network as the *dual* of said hypergraph, in which graph nodes correspond to circuit nodes and hyperedges correspond to components (or rather, the terminals of a component), as this view more closely resembles *flow networks* known from classical graph theory.

The lumped circuit model allows us to define *voltage* and *current* in the following manner:

Definition 3. We define voltage as the real-valued quantity that can be measured by an ideal voltmeter between two nodes or terminals. The voltage between measurement points a and b is denoted by $v_{a,b}$ and is measured in *Volts*. We define current as the real-valued quantity that can be measured by an ideal ammeter through a terminal. The current through terminal p is denoted by i_p and is measured in *Amperes*.

It is customary to appoint a node of an electrical network to be the so-called *ground node*, which acts as a universal basis for voltage measurements. This node is usually chosen to contain one terminal of a voltage source. A voltage source—e.g. electric guitar pickups—is a 2-terminal electrical device that generates a voltage across its terminals. Hence it is easy to see how the notion of a ground node is useful to us.

Definition 4. Let g denote the ground node within an electrical network. Let a denote a voltage measurement point. The voltage $v_{g,a}$ is simply referred to as the voltage in a.

These practical definitions suffice for the analysis of analog audio circuits without delving into the underlying electromagnetic theory. It should be mentioned that current measurements are *directional*, that is, one should always state the direction of measurement (*in* or *out* of the terminal). In practice, we will assign a measurement direction to all terminals when a network is introduced, thus the direction of measurement need not be indicated in subsequent mathematical formulae.

Axiom 1. *Let a, b, c be measurement points. The following equations hold for voltage measurements:*

$$\begin{aligned} v_{a,b} &= -v_{b,a}, \\ v_{a,a} &= 0, \\ v_{a,c} &= v_{a,b} + v_{b,c}. \end{aligned} \tag{2.1}$$

Axiom 2. *Let i denote the current flowing into terminal p. The current flowing out of terminal p is equal to $-i$.*

Definition 5. Let $p \neq q$ be terminals of an electrical component. If the current flowing into p is equal to the current flowing out of q at all times, then the unordered pair (p, q) is called a *branch*. Suppose that the measurement direction $\overrightarrow{pq}$ was affixed to the branch. The current flowing into terminal p (or equivalently, the current flowing out of terminal q) is called the *branch current*. The voltage $v_{q,p}$ is called the *branch voltage*.

In this work, we will only need to deal with two-terminal components (resistors, capacitors, diodes) that form a branch, and *operational amplifiers*, which we will discuss shortly.

Definition 6. An *electrical circuit* is an electrical network that contains at least one closed loop made of branches, through which electrical current may flow.

It is important to note that we often use the term *circuit* to refer to networks that contain no closed loops, but otherwise form a single *logical unit.* For example, the input network of a guitar effects pedal is an incomplete circuit that becomes complete only when e.g. an electric guitar's circuitry is connected.

In Circuit Theory, if a component has a branch between two of its terminals, then it is customary to define a so-called *branch constitutive equation* that relates the branch voltage to the branch current in the time domain. The graphs of such equations are sometimes referred to as *voltage to current curves* or *I-V curves.* The branch constitutive equations of components of interest can be found in Appendix A.

An ideal operational amplifier is characterized by an infinite input impedance, that is, the current flowing through its input terminals is zero. Furthermore, an ideal operational amplifier in a negative feedback configuration—like the 'OpClip' op-amp in the TS808 diode clipper circuit—will adjust its output voltage in such a way that the resulting voltage at its inverting input terminal is equal to the voltage seen by the operational amplifier at its non-inverting input terminal [2], given that this voltage falls into the operational range of the op-amp, and the feedback network is well-designed. This 'rule' is known as the *virtual short* principle. To achieve this, an ideal op-amp will draw as much current as necessary, which is equivalent to saying that the output impedance of an ideal op-amp is zero. The operational range of an op-amp is determined by the voltages at its power supply terminals. In our case, both op-amps operate in the 0V to 9V range. To be able to process an AC signal that falls within the −4.5V to 4.5V range, the signal is *biased* with 4.5V DC through a 10kΩ resistor. This DC offset will be 'removed' at the end of the tone/volume stage by a 1μF capacitor.

Voltages and currents in a lumped electrical network are governed by the following laws:

Axiom 3. (Kirchhoff's current law, KCL) *In a lumped network, the algebraic sum of branch currents through any node p is equal to 0. With mathematical notation:*

$$\sum_{r \in p} i_r = 0, \tag{2.2}$$

where the direction of current measurement for all $r \in p$ terminals are aligned, that is, into or out of node p.

Axiom 4. (Kirchhoff's voltage law, KVL) *In a lumped network, the algebraic sum of branch voltages around any closed loop C equals 0, that is, all branch voltage measurements are aligned in the same direction along loop C.*

2.3 Mathematical model of the clipping stage

The amount of distortion produced by the clipping stage can be adjusted using the 'OVERDRIVE' knob—or *drive knob* for short—on the physical pedal. The drive knob controls a variable resistrance that is embodied by the resistor R_f in our model.

The input signal enters the clipping stage at the node labeled 'IN' in Appendix A. The signal is then biased with a DC voltage of 4.5V.

Let us denote the input voltage as v_{in}, the voltages at the non-inverting and inverting input terminals of the op-amp as v_+ and v_- respectively, and the output voltage of the op-amp (and the clipping stage) as v_{out}. Using the virtual short principle, we have

$$v_+ = v_{in} + 4.5\text{V}, \tag{2.3}$$

$$v_- = v_+. \tag{2.4}$$

To formulate equations for the output voltage v_{out} we must first express the current through the feedback network using KCL:

$$i_{R_g} = i_{C_g} = i_{R_f} + i_{C_f} + i_{D_1} + i_{D_2} \tag{2.5}$$

Let $I_{D\downarrow\uparrow}(v) : \mathbb{R} \to \mathbb{R}$ denote the instantaneous current through the antiparallel diode configuration as a function of voltage. With the shorthand notation $\Delta := v_{out} - v_-$ and with v_Y denoting the voltage at node Y, we can rewrite (2.5) using the branch constitutive equations of the components as

$$\frac{v_Y}{r_g} = c_g(\dot{v}_- - \dot{v}_Y) = \frac{\Delta}{r_f} + c_f\dot{\Delta} + I_{D\downarrow\uparrow}(\Delta) \tag{2.6}$$

Finally, (2.6) can be reformulated as an ordinary differential equation system with two unknown functions $v_Y(t)$ and $\Delta(t)$:

$$\begin{cases} \dot{v}_Y = \dot{v}_- - \dfrac{v_Y}{r_g c_g}, \\ \dot{\Delta} = \dfrac{v_Y}{r_g c_f} - \dfrac{\Delta}{r_f c_f} - \dfrac{I_{D\downarrow\uparrow}(\Delta)}{c_f}, \end{cases} \tag{2.7}$$

or, to put it in terms of v_{out}:

$$\begin{cases} \dot{v}_Y = \dot{v}_{in} - \dfrac{v_Y}{r_g c_g}, \\ \dot{v}_{out} = \dot{v}_{in} + \dfrac{v_Y}{r_g c_f} - \dfrac{v_{out} - v_-}{r_f c_f} - \dfrac{I_{D\downarrow\uparrow}(v_{out} - v_-)}{c_f}. \end{cases} \tag{2.8}$$

Right now, these two formulations are functionally equivalent in the sense that one can be transformed into the other by a change of variables. However, we must be careful when designing our final solution, as the two forms produce different outcomes when discretized. This is demonstrated in Appendix C. Formulation (2.7) is preferred due to its conciseness, but (2.8) should be used in the final solution.

There are two things worth noting here. Firstly, even though a closed-form solution exists for v_Y, a much simpler formula can be derived in the discrete time domain by considering that the capacitor C_g and resistor R_g form a first-order high-pass filter network.

Second, we have a degree of freedom in defining $I_{D\downarrow\uparrow}$ for our mathematical model. A common approach is to use the *general diode equation* as seen in Appendix A, which is a reasonably accurate description of the behavior of diodes [7]. However, if we take the internal resistance of the diode into account, i.e. $r > 0$, the resulting equation is rendered implicit. Thus, often r is chosen to be 0, which results in a pure exponential relationship between the diode voltage and diode current. The $r = 0$ formulation of the general diode equation is also known as the *Shockley ideal diode equation* or the *explicit diode equation.* In either case, it is clear that the resulting ODE system (2.8) is nonlinear, and, as we will see in a later chapter, the system is *stiff.*

Using the explicit diode equation, $I_{D\downarrow\uparrow}$ may be defined as

$$I_{D\downarrow\uparrow}(v) := I_S\left(e^{\frac{v}{nV_T}} - 1\right) - I_S\left(e^{\frac{-v}{nV_T}} - 1\right) = 2I_S \sinh\left(\frac{v}{nV_T}\right). \tag{2.9}$$

This equation may be further simplified by considering that the reverse current of one diode is dominated by the forward current of the other [8], i.e.

$$2I_S \sinh\left(\frac{v}{nV_T}\right) \approx \operatorname{sgn}(v) I_S\left(e^{\frac{|v|}{nV_T}} - 1\right). \tag{2.10}$$

2.4 Mathematical model of the tone/volume stage

The purpose of this subcircuit is to modify the frequency content of the output signal. The 'LEVEL' knob adjusts the output level of the pedal, while the 'TONE' knob allows the user to adjust the tonal characteristics of their sound. This is a key feature of the TS808 circuit, and is often used in conjunction with the drive knob to shape the overall sound of the pedal.

The tone circuit can be considered a linear, time-invariant filter network. It is an active filter, as it contains a second operational amplifier ('OpTone'). This is a common configuration, as passive filters can introduce significant signal loss, especially at high frequencies. It is important to note that we will be considering the small-signal behavior of this subcircuit. Due to the clipping diodes in the previous stage, the signal entering the tone circuit is limited to a narrow range of voltages, thus the op-amp 'OpTone' is guaranteed to be operating in its linear region.

A *filter* modifies the frequency content of an input signal, i.e. it alters the amplitude and/or phase of the input signal as a function of frequency. Though the term *filter* implies *filtering out* certain frequencies, a filter may also *amplify* or *boost* certain frequencies. Linear and time-invariant filters can be fully described by their *transfer function.*

Lemma 1. *Let $x(t) := e^{st}$ for some $s \in \mathbb{C}$. Let y denote the output signal of a linear, time-invariant system to the input signal x. Then, $\exists!\ b_s \in \mathbb{C}$ such that $y(t) = b_s \cdot e^{st}$.*

Definition 7. Let us define the *transfer function* of a linear, time-invariant system as the $H : \mathbb{C} \to \mathbb{C}$ function $H(s) := b_s$, where $b_s \in \mathbb{C}$ is given by Lemma 1.

Definition 8. Let $f : \mathbb{R} \to \mathbb{C}$. The $F : \mathbb{C} \to \mathbb{C}$ function defined as

$$F(s) = \mathcal{L}\{f\}(s) := \int_{-\infty}^{+\infty} f(t)e^{-st}\, dt \tag{2.11}$$

is the *bilateral Laplace transform* of f.

Lemma 2. *Let $x : \mathbb{R} \to \mathbb{C}$ be a continuous signal, such that $\mathcal{L}\{x\}(s)$ exists for all $s \in \mathbb{C}$. If H denotes the transfer function of a linear, time-invariant system, and y denotes the output signal of the system to the input signal x, then the following equation holds:*

$$\mathcal{L}\{y\}(s) = H(s) \cdot \mathcal{L}\{x\}(s). \tag{2.12}$$

It is common for the transfer function to be expressed as a rational function of the form

$$H(s) = \frac{Y(s)}{X(s)}. \tag{2.13}$$

The transfer function of the TS808 tone stage can be derived using electrical engineering principles, and is given by

$$H_T(s) = \frac{\left[\frac{1-X(s)}{R_{p2}} + \frac{1}{Z_f}\right]}{\left[\frac{1-X(s)}{R_{p1}} + \frac{1}{Z_S(s)} + \frac{1}{Z_{in}}\right]}, \tag{2.14}$$

where

$$X(s) = \frac{Z_p(s)R_p}{Z_p(s)R_p + R_{p1}R_{p2}},$$
$$Z_S(s) = \frac{1}{s \cdot 220 \cdot 10^{-9} + 10^{-3}},$$
$$Z_p(s) = 220 + \frac{1}{s \cdot 220 \cdot 10^{-9}},$$
$$Z_{in} = 10^3,$$
$$Z_f = 10^3,$$
$$R_p = 2 \cdot 10^4,$$
$$R_{p1} = T \cdot R_p,$$
$$R_{p2} = (1 - T) \cdot R_p,$$

and $T \in [0, 1]$ is the position of the tone knob, as calculated and experimentally verified by Paul Darlington [9].

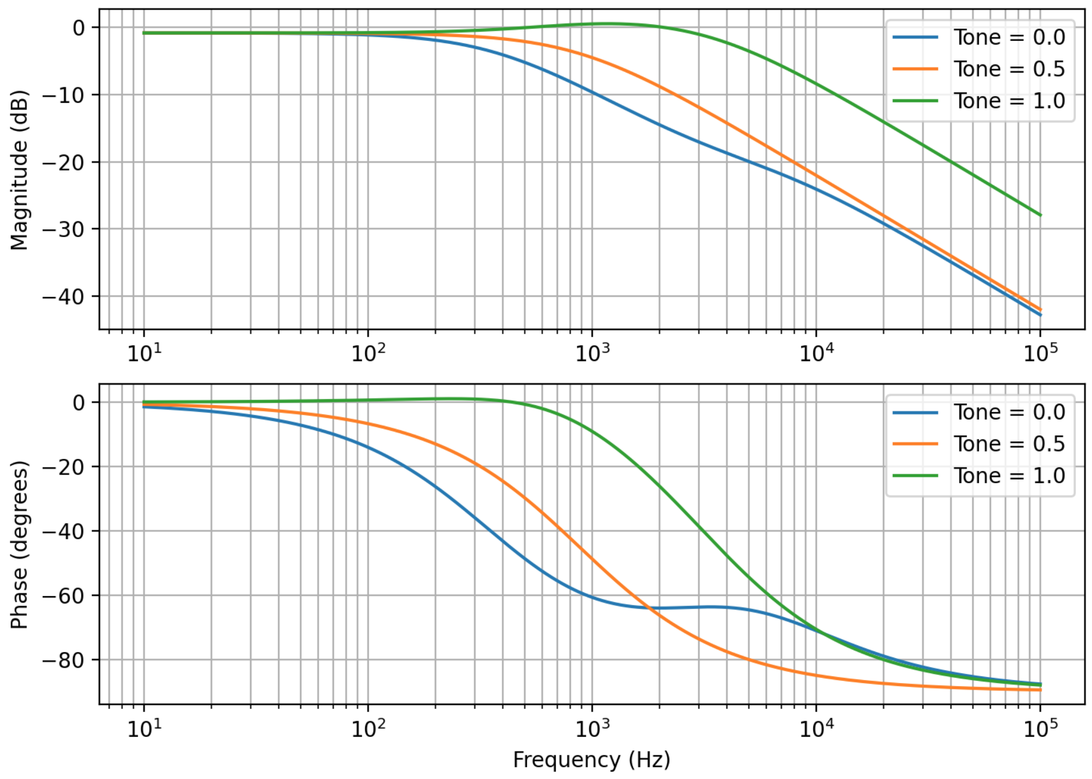


Figure 2.1. Frequency response of the TS808 tone stage.

Chapter 3

Introduction to digital signal processing

In this chapter, we will introduce the fundamentals of digital signal processing (DSP). We will take a look at two aspects of DSP, representing analog signals in the digital domain and discrete-time systems.

3.1 Understanding signals

Throughout this work, the letter μ will denote the one-dimensional Lebesgue measure, the letter j will be used to denote the imaginary unit instead of i.

Definition 9. An *analog signal* or *continuous-time signal* is an $x_c : \mathbb{R} \to \mathbb{R}$ continuous function of time. This is sometimes referred to as the *time-domain* representation of the signal. The *duration* of the signal is given by $\mu(\text{conv supp}(x_c))$. An analog signal is *finite-length* or *time-limited* if its duration is finite. For simplicity, we can assume that $x_c \neq 0$ is centered at time $t = 0$, i.e. $\text{conv supp}(x_c) = [-T, T]$ for some $T \in \mathbb{R}$. Let $\mathcal{A}$ denote the *set of all continuous-time signals.*

The voltage output of guitar pickups, or the input and ouput signals to an analog guitar effect pedal are examples of finite-length analog signals.

The Fourier transform, named after French mathematician and physicist Jean-Baptiste Joseph Fourier, is an essential tool for studying signals.

Definition 10. Let $f \in L^1(\mu)$. Let $e_\psi : \mathbb{R} \to \mathbb{C}$ be defined as $e_\psi(t) := exp(j\psi t)$. The *Fourier transform* of f is the $\widehat{f} : \mathbb{R} \to \mathbb{C}$ function

$$\widehat{f}(\psi) := \int_{\mathbb{R}} f e_\psi \, d\mu. \tag{3.1}$$

Theorem 1. *The Fourier transform is well-defined for all $f \in L^1(\mu)$, moreover,*

i) $\widehat{f}$ *is uniformly continuous,*

ii) $\widehat{f}(-\psi) = \overline{\widehat{f}(\psi)}$,

iii) $\lim_{|\psi| \to +\infty} \widehat{f}(\psi) = 0$,

iv) $f \neq g$ *w.r.t.* $\mu \implies \widehat{f} \neq \widehat{g}$ *w.r.t.* $\mu \quad (g \in L^1(\mu))$,

v) $g \in L^1(\mu) \implies \widehat{f * g} = \widehat{f} \cdot \widehat{g}$, *where* $(*)$ *is the convolution operator,*

vi) $g \in L^1(\mu) \implies \int \widehat{f} g \, d\mu = \int f \widehat{g} \, d\mu$

vii) (Inverse Fourier transform) *if* $\widehat{f} \in L^1(\mu)$, *then*

$$f(t) = \frac{1}{2\pi} \int_{\mathbb{R}} \widehat{f} e_{-t} \, d\mu \tag{3.2}$$

almost everywhere w.r.t. μ,

viii) *if* $f, \widehat{f} \in L^1(\mu) \implies f, \widehat{f} \in L^2(\mu)$.

This formula may be familiar; indeed, the Fourier transform can be viewed as a special case of the bilateral Laplace transform (2.11), where $s = j\psi$.

Intuitively, the Fourier transform decomposes a signal into a continuous spectrum of sinusoidal components. With DSP jargon, the Fourier transform is called the *frequency-domain* representation of a signal.

While the Fourier transform characterizes the signal as a whole, the *windowed Fourier transform* analyzes the signal within a given time frame.

Definition 11. Let $f \in L^1(\mu)$. Let $w : \mathbb{R} \to \mathbb{R}$, $w \in L^1(\mu)$ called a *window function.* The *windowed Fourier transform* of f at time $t = \tau$ is defined as

$$\widehat{f^w}(\tau, \psi) := \int_{\mathbb{R}} w_\tau f e_\psi \, d\mu, \tag{3.3}$$

where $w_\tau(t) := w(t - \tau)$.

Most common window functions are bell-shaped, even functions with support on $[-T, T]$, where the duration $2T$ is significantly shorter than the duration of the signal. Hence the windowed Fourier transform is sometimes referred to as *short-time Fourier transform* or *STFT.*

In the context of music, we are often concerned with the instantaneous frequency content or *harmonic content* of a signal. So-called *spectrum analyzers* usually plot the *magnitude spectrum*, that is, the value—or rather, the approximation—of $|\widehat{f^w}|$ with an appropriate window function w. We note that $|\widehat{f^w}(\psi)| = |\widehat{f^w}(-\psi)|$, thus it is sufficient to plot $|\widehat{f^w}|$ for positive frequencies only.

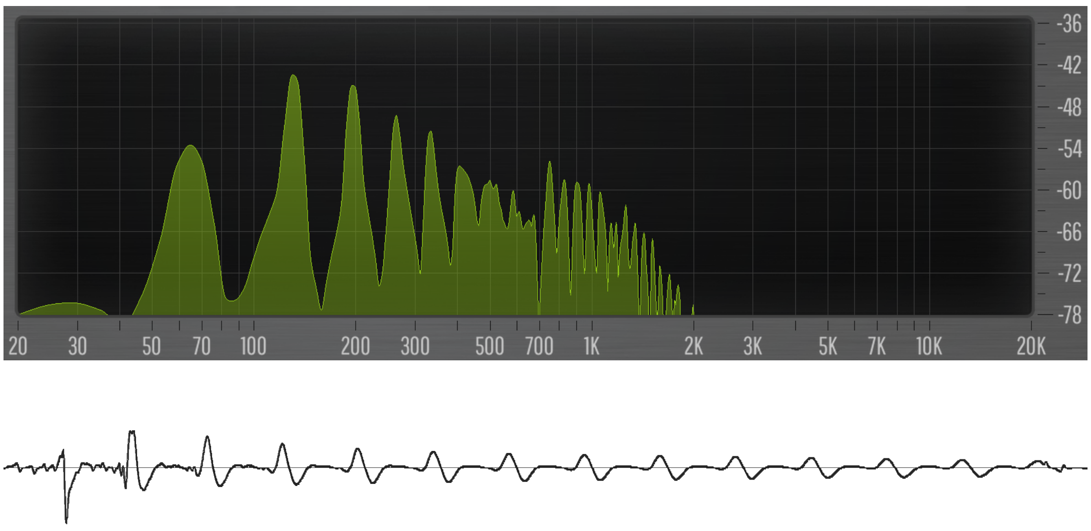


Figure 3.1. Magnitude spectrum of an electric guitar signal. Signal length approx. 210ms. Created with Voxengo SPAN [10].

To be able to use computer software to process signals, we need a way of representing them digitally.

Definition 12. A *discrete-time signal* or *digital signal* is a sequence of numbers $x_d : \mathbb{Z} \to \mathbb{R}$. The elements of the sequence are called *samples*. The n-th sample is usually written as $x_d[n]$. In practice, analog signals are converted to digital signals via *periodic sampling*, that is, $x_d[n] := x_c(nT)$ for some $T > 0$ called *sampling time* or *sampling period*. The ratio $\frac{1}{T}$ is called the *sampling frequency*. Let $\mathcal{D}$ denote the *set of all digital signals.*

Naturally the question arises: is it possible to restore the original signal from its samples without loss? The following theorem provides insight.

Definition 13. Let $x_c \in L^1(\mu)$. x_c is said to be *bandlimited* if $\exists\ \Psi > 0$ such that

$$\forall\ |\psi| \geq \Psi : \widehat{x}_c(2\pi\psi) = 0. \tag{3.4}$$

Let us call the minimum of such Ψ values the *bandwidth* of x_c.

We note that, due to the continuity of $\widehat{x}_c$, the set of values satisfying (3.4) takes the form $[a, +\infty)$, and thus the definition of bandwidth is well-posed.

Theorem 2. (Whittaker-Kotelnikov-Shannon sampling theorem) *Let x_c be a bandlimited analog signal. Let $T > 0$, $f_T := \frac{1}{T}$. x_c is uniquely determined by the uniform sampling $x_d[n] := x_c(nT)$ $(n \in \mathbb{Z})$, if and only if the bandwidth of x_c is less than $\frac{1}{2}f_T$, in which case*

$$x_c(t) = \sum_{n\in\mathbb{Z}} x_d[n] s_T(t - nT), \tag{3.5}$$

where

$$s_T(t) := \begin{cases} \frac{sin(\pi t f_T)}{\pi t f_T} & \text{if } t \neq 0, \\ 1 & \text{if } t = 0, \end{cases} \tag{3.6}$$

also known as the sinc function. In other words, the system $(s_{T,n}(t) := s_T(t-nT), n \in \mathbb{Z})$ *forms an orthogonal base w.r.t.* $\|.\|_2$ *in the space* $\{x_c \in L^2(\mu) : supp(\widehat{x}_c) \subset [-\frac{1}{2}f_T, \frac{1}{2}f_T]\}$*. The frequency* $\frac{1}{2}f_T$ *is called the Nyquist frequency. The frequency* f_T *is called the Nyquist rate.*

However, a truly bandlimited signal must be infinite in length, or conversely, a time-limited signal must have infinite bandwidth [11]. Hence it is impossible to perfectly reconstruct a time-limited signal from its samples. Suppose that x_c is such a signal, with $x_d[n] := x_c(nT)$, $n \in \mathbb{Z}$, $T > 0$. The reconstructed signal $x_c' \neq x_c$ produces the same sequence of samples x_d when sampled with period T. This effect is called *aliasing.*

Definition 14. Let $\mathcal{D}_T : \mathcal{A} \to \mathcal{D}$ denote the *sampling operator* with sampling period $T > 0$, that is, $(\mathcal{D}_T x_c)[n] := x_c(nT)$ for $n \in \mathbb{Z}$. Let $x_d \in \mathcal{D}$. Let X_c^T denote the set of analog signals whose image with respect to $\mathcal{D}_T$ is x_d, i.e. $\mathcal{D}_T[X_c^T] = \{x_d\}$. We say that the signals in X_c^T are *aliases* of one another.

Let us look at a concrete example of aliasing. Let $x_c(t) := cos(2\pi f t)$, where $\frac{1}{2}f_T < f < f_T$. With the substitution $f = \frac{1}{2}f_T + \delta$, the samples of x_c can be written as

$$x_d[n] := cos\big(2\pi(\tfrac{1}{2}f_T + \delta)nT\big) = cos\big(2\pi(\delta - \tfrac{1}{2}f_T)nT\big) = cos\big(2\pi(\tfrac{1}{2}f_T - \delta)nT\big), \tag{3.7}$$

thus resembling a $\frac{1}{2}f_T - \delta$ frequency oscillation. This effect is called *frequency folding.* In the case of musical signals, high frequency oscillations that would normally be inaudible, may 'fold back' into the audible frequency range. This form of distortion usually sounds unpleasant, therefore we often strive to minimize aliasing. The next theorem can aid us in quantifying aliasing.

Definition 15. Let $x_c \in L^2(\mu)$. The *energy* of x_c is defined as

$$E_{x_c} := \|x_c\|_2^2 = \int_{\mathbb{R}} |x_c|^2 d\mu < +\infty. \tag{3.8}$$

We note that time-limited signals as defined in Definition 1 are members of $L^2(\mu)$, and therefore possess finite energy.

Theorem 3. (Plancherel's theorem) *If the energy of a signal* x_c *is finite, i.e.* $x_c \in L^2(\mu)$*, then* $\widehat{x}_c \in L^2(\mu)$ *and the following equation holds:*

$$\|x_c\|_2^2 = \frac{1}{2\pi}\|\widehat{x}_c\|_2^2. \tag{3.9}$$

Definition 16. The $\mathbb{R} \to \mathbb{R}$ function $\overline{S}_{x_c x_c}(\psi) := |\widehat{x}_c(\psi)|^2$ is called the *energy spectral density* of x_c.

Let us use Plancherel's theorem to express the energy of a signal as

$$E_{x_c} = \frac{1}{2\pi}\|\widehat{x}_c\|_2^2 = \frac{1}{2\pi}\int_{-\infty}^{+\infty} |\widehat{x}_c(\psi)|^2 \, d\psi = \frac{1}{\pi}\int_{0}^{+\infty} |\widehat{x}_c(\psi)|^2 \, d\psi, \tag{3.10}$$

therefore, to minimize aliasing is to minimize the spectral energy beyond the Nyquist frequency. This can be done by increasing the sampling rate, or by attenuating high-frequency oscillations via carefully designed *low-pass filters*. Such filters are often called *anti-aliasing filters*.

It is important to note that aliasing may happen not only during sampling. Certain digital signal processing algorithms, such as distortion, can generate upper harmonics, thus increasing the bandwidth of a signal.

While aliasing cannot be completely eliminated when sampling time-limited signals, the methods described above allow us to reduce it to the point where it is rarely a concern with modern devices.

So far we looked at the frequency-domain representation of analog signals, but not digital signals.

Definition 17. Let $x_d \in \mathcal{D}$. Let $e_\xi : \mathbb{R} \to \mathbb{C}$ be defined as $e_\xi(t) := exp(j\xi t)$. The *discrete-time Fourier transform* (DTFT) of the digital signal x_d is defined as

$$\widehat{x}_d(\xi) := \sum_{n\in\mathbb{Z}} x_d[n] e_{-n}(\xi)\Big|_{[-\pi,\pi]}. \tag{3.11}$$

Let us note that the DTFT is defined over the normalized frequency range $[-\pi, \pi]$. Given a sampling rate $T > 0$, we may reinterpret the DTFT on the range $[-\frac{1}{2}f_T, \frac{1}{2}f_T]$ by the substitution $\xi = 2\pi\psi T$. This allows us to analyze bandwidth and spectral content in physically meaningful units. The restriction $\xi \in [-\pi, \pi]$ is necessary as $e_{-n}(\xi)$ is 2π-periodic $\forall\ n \in \mathbb{Z}$, though any other interval of length 2π would suffice, this choice is in harmony with the continuous-time Fourier transform.

Lemma 3. *The DTFT of the digital signal $x_d \in \mathcal{D}$ exists if and only if*

$$\sum_{k\in\mathbb{Z}} \Big| x_d[k] \Big| < +\infty. \tag{3.12}$$

It can be shown [12] that the DTFT is equivalent to the Fourier transform of the reconstructed signal, i.e. if $x_c \in L^2(\mu)$ is bandlimited, $x_d \in \mathcal{D}$ is an alias-free sampling of

x_c with some sampling period $T > 0$, and given the DTFT $\widehat{x}_d$ exists, then

$$\left\| \widehat{x}_c^T - \sum_{n=-k}^{k} x_d[n] e_{-n} \Big|_{[-\pi,\pi]} \right\|_2 \to 0 \qquad (k \to \infty), \tag{3.13}$$

where $\widehat{x}_c^T(\xi) := \widehat{x}_c(2\pi\xi T)$.

In practice, however, the DTFT is hard to calculate numerically, instead, the *discrete Fourier transform* is used to get an approximation of the DTFT.

Definition 18. Let $x_d \in \mathcal{D}$ be a finite-length digital signal, with $x_d[0]$ and $x_d[N-1]$ denoting the first and last non-zero sample respectively. The *discrete Fourier transform* or DFT of x_d is the N-tuple $(X_d[0], \ldots, X_d[N-1])$ defined by

$$X_d[n] := \sum_{k=0}^{N-1} x_d[k] e^{-2\pi jkn/N}. \tag{3.14}$$

Essentially, the DFT is the uniform sampling of the DTFT at N equidistant 'frequencies'. For ease of notation, however, the sample points span the interval $[0, 2\pi]$ instead of $[-\pi, \pi]$.

The DFT can be computed in $\mathcal{O}(N \log N)$ time using the family of *fast Fourier transform* (FFT) algorithms. Hence, many digital audio processing software depend on FFT for frequency analysis. One example is the spectrum analyzer plugin in Figure 3.1.

3.2 Discrete-time systems

Before delving into discrete-time systems, it is helpful to briefly introduce computational block diagrams. These diagrams provide an intuitive and compact way to represent and analyze discrete-time systems, with the basic building blocks being adders, multipliers and delays.

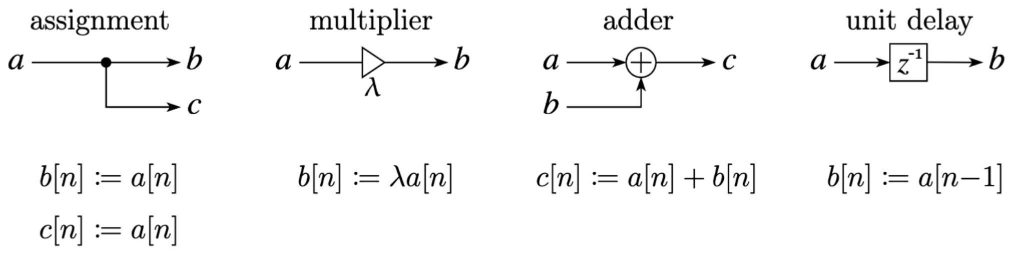


Figure 3.2. The basic symbols of computational block diagrams.

Lemma 4. *The computational structure defined by a computational block diagram is realizable if and only if the block diagram contains no delay-free loops.*

3.2.1 Linear and time-invariant systems

Definition 19. A $P : \mathcal{D} \to \mathcal{D}$ operator is called a *discrete-time system*, and is usually denoted as $y[n] = P\{x\}[n]$. Let $x_1, x_2 \in \mathcal{D}$, $n_0 \in \mathbb{Z}$. P is

- *causal* if $(\forall\, n \leq n_0 : x_1[n] = x_2[n]) \implies (\forall\, n \leq n_0 : y_1[n] = y_2[n])$,
- *linear* if for all $a, b \in \mathbb{R}$, $n \in \mathbb{Z}$:

$$T\{ax_1 + bx_2\}[n] = ay_1[n] + by_2[n] \tag{3.15}$$

- *time-invariant* if $(\forall\, n \in \mathbb{Z} : x_1[n] = x_2[n_0{+}n]) \implies (\forall\, n \in \mathbb{Z} : y_1[n] = y_2[n_0{+}n])$,
- *stable* in the bounded-input bounded-output (BIBO) sense if for every bounded input signal, the output of the system is also bounded.

Definition 20. The digital signal

$$\delta[n] := \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

is called the *unit sample sequence.*

It is clear that any digital signal x can be expressed as the weighted sum of infinitely many time-shifted unit sample sequences:

$$x = \sum_{k \in \mathbb{Z}} x[k]\delta_k, \tag{3.17}$$

where $\delta_k[n] := \delta[n - k]$.

Consider a linear, time-invariant system P. The output of this system to the input signal x can be expressed using equations (3.17) and (3.15) as

$$y = P\{x\} = \sum_{k \in \mathbb{Z}} x[k]P\{\delta_k\}. \tag{3.18}$$

Since P is time-invariant, $P\{\delta_k\}$ is simply the time-shifted version of $P\{\delta\}$.

Definition 21. The sequence $h := P\{\delta\}$ for a linear, time-invariant system P is called its *impulse response.*

Linear and time-invariant—or LTI for short—systems are of particular importance in DSP due to their usefulness to model many real-world processes. From (3.18) it can be said that a LTI system is fully described by its impulse response. The summation in (3.18) is sometimes called a *convolution sum*, and shall be written as $y = x * h = x * P\{\delta\}$

for simplicity. The general properties of LTI systems can be found by considering the properties of discrete-time convolution. More in-depth analysis of LTI systems can be found in [13].

Lemma 5. *A LTI system with impulse response h is causal if and only if $h[n] = 0$ for all $n < 0$.*

We will only deal with causal LTI systems in this work. Therefore, from now on, we will assume that LTI systems are causal, and if $h \not\equiv 0$ then $h[0] \neq 0$.

Lemma 6. *A LTI system P is BIBO-stable if and only if*

$$\sum_{k\in\mathbb{Z}} |h[k]| < +\infty. \tag{3.19}$$

Many analog audio circuits can be discretized using discrete-time LTI systems. For this purpose, let us look at the frequency-domain representation of LTI systems.

Let $x[n] := e^{j\omega n}$ for some $\omega > 0$. The output of a LTI system with impulse response $h[n]$, to the input signal x, can be written as

$$y[n] = H(e^{j\omega})e^{j\omega n}, \tag{3.20}$$

where

$$H(e^{j\omega}) = \sum_{k\in\mathbb{Z}} h[k]e^{-j\omega k}. \tag{3.21}$$

Thus $e^{j\omega n}$ is an eigenfunction of the system, with $H(e^{j\omega})$ being the associated eigenvalue.

We duly note the similarity of the above equation to the DTFT of the impulse response h of the system:

$$\widehat{h}_d = H(e^{j\omega})\Big|_{\omega\in[-\pi,\pi]}. \tag{3.22}$$

If h is real-valued, then $H(e^{-j\omega}) = \overline{H(e^{j\omega})}$, similar to the Fourier transform of real-valued signals.

In general, we wish to find the $H : \mathbb{C} \to \mathbb{C}$ function, called the *transfer function* of the system, such that equations (3.20) and (3.21) hold. We also hope that the transfer function of a system—if found—provides further information about the system's behavior. We will later see that this is indeed the case for FIR and IIR systems. The power series on the right hand side of (3.21) is also known as the *z-transform* of the impulse response h.

Definition 22. The $\mathbb{R} \to \mathbb{C}$ function $\overset{\circ}{H}(\omega) := H(e^{j\omega})$ is called the *frequency response* of a LTI system. The $\mathbb{R} \to \mathbb{R}$ function $|\overset{\circ}{H}|$ is called the *magnitude response* of the system, while the $\mathbb{R} \to \mathbb{R}$ function $\theta := \arg \overset{\circ}{H}$ is called the *phase response* of the system.

The notation I use for the frequency response $\overset{\circ}{H}$ is non-standard, but it helps to avoid confusion with the transfer function H. Essentially, the frequency response is the transfer function evaluated on the unit circle of the complex plane, hence the notation.

The frequency response characterizes the steady-state reponse of the system. LTI systems can also be characterized by their behavior to so-called *suddenly applied* complex exponential inputs, the result of which is called the *transient response* of the system. In this work, we will only be concerned with the frequency response of a system for the sake of simplicity.

If we apply the convolution theorem to (3.18), we can express the DTFT of the output signal as

$$\widehat{y}_d = \widehat{x}_d \cdot \widehat{h}_d = \widehat{x}_d \cdot \overset{\circ}{H}(\omega)\Big|_{\omega \in [-\pi,\pi]}. \tag{3.23}$$

Thus the magnitude and phase response of the system can be interpreted as the gain (amplification or attenuation) and phase shift of the system at a given frequency. With a concrete example, let

$$x[n] = 2\cos(\omega n) = e^{j\omega n} + e^{-j\omega n}$$

be the input signal to the system. The output signal can be expressed as

$$y[n] = \overset{\circ}{H}(\omega)e^{j\omega n} + \overline{\overset{\circ}{H}(\omega)}e^{-j\omega n} = 2|\overset{\circ}{H}(\omega)|\cos(\omega n + \theta(\omega)).$$

In the context of audio and digital signal processing, the phase behavior of an LTI system plays a critical role in preserving perceptual audio quality.

Definition 23. The $\mathbb{R} \to \mathbb{R}$ function

$$\tau_{\mathrm{p}}(\omega) := -\frac{\theta(\omega)}{\omega} \tag{3.24}$$

is called the *phase delay* of the system. The $\mathbb{R} \to \mathbb{R}$ function

$$\tau_{\mathrm{g}}(\omega) := -\frac{d\theta(\omega)}{d\omega} \tag{3.25}$$

is called the *group delay* of the system. A LTI system is *linear phase* if its frequency response can be expressed as

$$\overset{\circ}{H}(\omega) = A(\omega) \cdot e^{j\omega\phi}, \tag{3.26}$$

where A is a $\mathbb{R} \to \mathbb{R}$ function, and $\phi \in \mathbb{R}$.

In a linear phase system, all frequency components are delayed equally, preserving the waveform shape of the input signal. The group delay of such systems is therefore constant and equal to ϕ.

Theorem 4. *A causal LTI system is linear phase if and only if its impulse response h is of finite length and is symmetric or anti-symmetric, that is,*

$$h[n] = h[N-1-n] \quad \textit{(symmetric)} \quad \textit{or} \quad h[n] = -h[N-1-n] \quad \textit{(anti-symmetric)},$$

where N is the length of the impulse response, i.e.

$$N \coloneqq \max\big(\{0\} \cup \{n \in \mathbb{Z} \mid h[n-1] \neq 0\}\big).$$

Proof can be found in [14].

In practice, it is often impossible to achieve a perfectly linear phase response due to other constraints and considerations. However, LTI systems can be designed to 'approximate' linear phase behavior to a degree. Such systems are called *minimum phase* systems, and will be discussed in more detail in the context of IIR systems.

3.2.2 FIR and IIR systems

Let us look more closely at two particular group of LTI systems, namely, *finite impulse response* (FIR) and *infinite impulse response* (IIR) systems. We will uncover the 'true' transfer function of these systems by considering the desired relationship between the transfer function and the impulse response in equation (3.21). We can view the above equation as a Laurent series expansion of the transfer function around the origin of the complex plane. Since this series consists of terms with negative powers only, we may also consider the 'dual' series, given by

$$S'(z) \coloneqq \sum_{k \in \mathbb{Z}} h[k] z^k, \tag{3.27}$$

in our reasoning. We will see that the transfer function of FIR and IIR systems can be expressed as rational functions, with the zeros of the denominator being the poles of the system. The poles determine the system's stability, while the arrangement of both poles and zeros determine the frequency response.

FIR systems

Definition 24. A *finite impulse response* system is a LTI system whose impulse response h is finite-length, i.e. $h[n] = 0$ for all but finitely many $n \in \mathbb{Z}$.

All non-causal FIR systems can be made causal by adding a finite number of unit delays to the system. The impulse response coefficients are often referred to as taps, as in "a FIR filter with 92 taps". Often a FIR system is identified with its impulse response, and may be referred to as a *kernel*, e.g. "convolving the signal with a FIR kernel".

FIR systems are, by definition, BIBO-stable, and can be implemented using a finite number of multipliers and adders. Furthermore, FIR systems can be designed to have a linear phase response by enforcing symmetry or anti-symmetry in their impulse response, as per Theorem 4. Thus, FIR systems can be employed when a perfectly linear phase behavior is desired. However, this comes at a cost, as the group delay of a linear phase FIR system is equal to $\frac{N-1}{2}$, which means that the system introduces a delay of $\frac{N-1}{2}$ samples. This can be problematic in real-time applications if the length of the filter is too large, as it may introduce a noticeable latency to the system.

We have an easy way of calculating the transfer function of a FIR system using equation (3.21), as the impulse response is of finite length.

Definition 25. The *transfer function* of a FIR system is defined as

$$H(z) := \sum_{k=0}^{N-1} h[k]z^{-k} = z^{-N+1}[\ h_0 z^{N-1} + h_1 z^{N-2} \cdots + h_{N-1}\ ] = \frac{D(z)}{z^{N-1}}, \tag{3.28}$$

where D is a polynomial of degree $N-1$ that is multiplied by an $(N-1)$th-order pole located at the origin of the complex plane.

The zeros of the transfer function are the roots of the polynomial D, which alone determine the frequency response, as the poles of the system are fixed at the origin. A real-valued impulse response implies that the zeros are either real or occur in complex conjugate pairs. If the system is also linear phase, then it can be shown [14] that the zeros of the transfer function are symmetric with respect to the unit circle in the complex plane, i.e. if z_0 is a zero, then $\overline{z_0}^{-1}$ is also a zero. Based on these properties, it can be shown [14] that, for every linear phase FIR system, the numerator D can be factored into a product of the following five basic forms:

$$\begin{aligned}
&\text{(I/II)} \quad D(z) = (z \pm 1)K && (K \in \mathbb{R}),\\
&\text{(III)} \quad D(z) = (z^2 + (r + r^{-1})z + 1)K && (K \in \mathbb{R},\ r \in \mathbb{R}\setminus\{0\}),\\
&\text{(IV)} \quad D(z) = (z^2 + (2\cos x)z + 1)K && (K, x \in \mathbb{R}),\\
&\text{(V)} \quad D(z) = \Bigg\{ z^4 + \left[\left(2\frac{r^2+1}{r}\right)\cos x\right] z^3 + \left[r^2 + \frac{1}{r^2} + 4\cos^2 x\right] z^2 + \\
&\qquad + \left[\left(2\frac{r^2+1}{r}\right)\cos x\right] z + 1 \Bigg\} K && (K, x \in \mathbb{R},\ r \in \mathbb{R}\setminus\{0\}).
\end{aligned}$$

This factored form can be used in implementing a filter by cascading short filters to realize a long filter.

IIR systems

Definition 26. An *infinite impulse response* system is a LTI system whose impulse response h is infinite-length, i.e. $h[n] \neq 0$ for infinitely many $n \in \mathbb{Z}$.

We will limit our discussion to causal IIR systems of the form

$$y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k], \tag{3.29}$$

where $a_k, b_k \in \mathbb{R}$ and $N, M \in \mathbb{N}$. We will also assume that at least one of the coefficients a_k is non-zero, otherwise the system is a FIR system. Such systems are also called *recursive* systems, as the output of the system depends on not just the input sequence x, but also on the previous N output values. Equations of the form (3.29) are called *linear, constant-coefficient difference equations* (LCCDE). While IIR systems are usually defined by LCCDEs, the output of the system can also be expressed as a convolution sum $y = h * x$, similar to FIR systems, albeit with an infinite-term summation.

By defining $a_0 := 1$, equation (3.29) can also be written as

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k], \qquad (n \in \mathbb{Z}). \tag{3.30}$$

Deriving the transfer function of an IIR system is a tad more complicated than that of a FIR system.

Definition 27. The *transfer function* of an IIR system given by (3.29) is defined as

$$H(z) := \frac{B(z)}{A(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}, \qquad (A(z) \neq 0). \tag{3.31}$$

Lemma 7.

$$H(z) = \sum_{n=0}^{\infty} h[n] z^{-n}, \tag{3.32}$$

where the region of convergence is given by

$$|z| > R := \max\{|z_0| \bigm| z_0 \textit{ is a pole of } H\}. \tag{3.33}$$

Proof. Let us rearrange (3.31) to obtain

$$0 = B(z) - A(z)H(z), \tag{3.34}$$

which is equivalent to

$$H(z) = B(z) + H(z)(1 - A(z)). \tag{3.35}$$

On the other hand, if the lemma was true, then by using equation (3.29), we may also express H in the region of convergence as

$$H(z) = \sum_{n=0}^{\infty} h[n]z^{-n} = \sum_{n=0}^{\infty} \left(-\sum_{k=1}^{N} a_k h[n-k]z^{-n} + \sum_{k=0}^{M} b_k \delta[n-k]z^{-n} \right). \tag{3.36}$$

Since $\delta[n-k] = 1$ only if $n = k$, we have

$$H(z) = B(z) + \sum_{n=0}^{\infty} \sum_{k=1}^{N} -a_k h[n-k]z^{-n}. \tag{3.37}$$

Let us visualize the terms of the summation in (3.37) in the following table:

	$k=1$	2	3	$\cdots$	N
$n=0$	0	0	0	$\cdots$	0
1	$-a_1h[0]z^{-1}$	0	0	$\cdots$	0
2	$-a_1h[1]z^{-2}$	$-a_2h[0]z^{-2}$	0	$\cdots$	0
3	$-a_1h[2]z^{-3}$	$-a_2h[1]z^{-3}$	$-a_3h[0]z^{-3}$	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
N	$-a_1h[N-1]z^{-N}$	$-a_2h[N-2]z^{-N}$	$-a_3h[N-3]z^{-N}$	$\cdots$	$-a_Nh[0]z^{-N}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$

We observe that, by grouping the terms in the table diagonally—and given that the series converges—we can rewrite the summation as

$$\sum_{n=0}^{\infty} \sum_{k=1}^{N} -a_k h[n-k]z^{-n} = \left(\sum_{m=0}^{\infty} h[m]z^{-m} \right) \left(\sum_{l=1}^{N} -a_l z^{-l} \right) = H(z)(1 - A(z)), \tag{3.38}$$

thus giving us (3.35).

The region of convergence can be found by considering the 'dual' of the transfer function,

$$H'(z) \coloneqq H(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{\sum_{k=0}^{M} b_k z^k}{\sum_{k=0}^{N} a_k z^k} = \frac{b_0 + b_1 z + \cdots + b_M z^M}{1 + a_1 z + \cdots + a_N z^N}. \tag{3.39}$$

It is clear that z_0 is an n-th order pole of H' $\iff$ z_0^{-1} is an n-th order pole of H $\iff$ z_0 is a root of the denominator of H'. However, $z_0 \neq 0$, as the denominator has a constant term of 1. We can also see that H' is *analytic* for all $z \in \mathbb{C} \setminus \{z_0 \mid z_0 \text{ is a pole of } H'\}$, as it is a 'nice' function. Therefore, H' is analytic in 0, and S'—as defined in (3.27)—is the power series expansion of H' around 0. The radius of convergence for S' is therefore the distance from 0 to the nearest pole of H' [15]. From this result, the statement of the lemma can be easily concluded. □

From the above lemma, we can see that in order for an IIR system to be stable, the region of convergence for (3.32) must include the unit circle of the complex plane. We are now ready to characterize the stability of IIR systems in the following theorem.

Theorem 5. *A causal IIR system is BIBO-stable if and only if all poles of the transfer function H are located strictly inside the unit circle of the complex plane, i.e. if z_0 is a pole, then $|z_0| < 1$.*

Unlike FIR systems, IIR systems can be unstable. This fact must be taken into account when designing and working with IIR systems.

Let us now look at the phase behavior of IIR systems. IIR systems can not be designed to have a linear phase response due to Theorem 4. However, it is possible to design IIR systems to have the *minimum phase* shift possible for a given magnitude response. It can be shown that, if $z_0 \neq 0$ is a zero of the transfer function H, then the variation of the system, where the zero z_0 is 'replaced' by $\overline{z_0}^{-1}$, has the same magnitude response, but a different phase response [14].

Definition 28. A causal and stable IIR system is *minimum-phase* if all zeroes of its trasfer function H lie within or on the unit circle of the complex plane, i.e. if z_0 is a zero, then $|z_0| \leq 1$.

Minimum-phase IIR systems are valued for combining efficient magnitude response shaping with minimal phase distortion, which is especially useful in real-time applications.

For more information on FIR and IIR systems, the reader is referred to the excellent textbooks [13] and [14].

3.2.3 Resampling digital signals

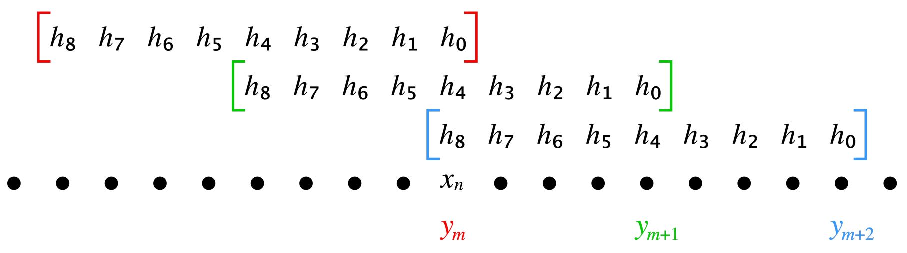
Sometimes we need to change the sampling rate of a digital signal within the digital domain, for example, to match the sampling rate of a digital audio effect. This process is called *resampling* or *sample rate conversion.* Two important cases of resampling are *downsampling* or *decimation*, and *upsampling* or *oversampling*, by an integer factor M.

Decimation by an integer factor

Decimation by an integer factor can be easily achieved by keeping only every M-th sample from the signal. However, to avoid aliasing, we must first ensure that the signal is sufficiently bandlimited with respect to the new sampling rate. This can be done using an appropriate digital low-pass filter—usually implemented as linear-phase FIR filters—before decimation. Such filters are called *decimation filters*. This is analogous to the anti-aliasing filter applied before sampling an analog signal, with the difference that, in the case of sampling, the filter is an analog filter, usually implemented using analog circuitry.

However, since we are 'throwing away' every non-M-th sample after the decimation filter, we would be wasting computational resources if we calculated every post-filter sample. We can take advantage of this fact, and design much more efficient decimation algorithms. One obvious optimization is to just not calculate the unneeded samples. We can do this, because FIR filters don't depend on previous output values. This alone reduces the operation count by a factor of $\frac{M-1}{M}$.

We can optimize this algorithm even further by restructuring the calculations in a way that benefits modern computer architectures. We can visualize the decimation filter as a 'sliding window' over the input signal, where the window is the vector of the impulse response coefficients. The output samples of the filter are given by the dot product of the input signal and the impulse response. The illustration below shows a theoretical 4x decimation filter with a 9-tap impulse response.



h_8 h_7 h_6 h_5 h_4 h_3 h_2 h_1 h_0
h_8 h_7 h_6 h_5 h_4 h_3 h_2 h_1 h_0
h_8 h_7 h_6 h_5 h_4 h_3 h_2 h_1 h_0
x_n
y_m
y_{m+1}
y_{m+2}

We can see that, when forming the dot products, the input sample x_n will only ever be multiplied by the coefficients h_0, h_4 and h_8. Similarly, the input sample x_{n+1} will only ever be multiplied by the coefficients h_3 and h_7, and so on. This motivates the following approach. We split the input signal into M sub-signals, each containing every M-th sample of the original signal. We also split the impulse response into M sub-impulse responses with a maximum length of $\lceil \frac{N}{M} \rceil$ in a similar fashion, where N is the length of the full impulse response. We can then apply these much shorter filters to each sub-signal in parallel, and then combine the results to obtain the final output signal. This is called *polyphase decimation*.

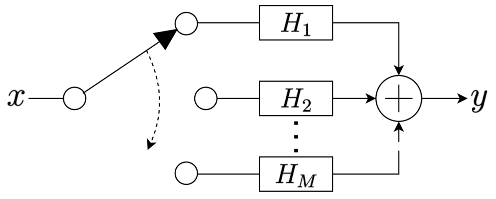


Figure 3.3. Block diagram of a polyphase decimation filter.

Again, this method does *not* reduce the number of required theoretical calculations. Instead, it improves performance by enabling more effective compiler optimizations—such as vectorization and loop unrolling—which are particularly effective on SIMD (Single Instruction, Multiple Data) architectures, where multiple data points can be processed in parallel. Additional benefits, such as improved cache locality, may also help in reducing the number of CPU cycles required. Even on single-threaded, non-SIMD architectures, the polyphase decimation algorithm can be more efficient than the direct approach.

Another optimization technique is *multistage decimation*. Suppose that $M = M_1 \cdot M_2$, where $M_1, M_2 \in \mathbb{N}$. We can decimate the signal by a factor of M_1, and then decimate the result by a factor of M_2, to achieve an overall decimation by a factor of M. This technique is useful when M is large, and hence the decimation filter would need to have a large number of taps. Though the overall number of calculations may be even higher than that of a single-stage decimation, the shorter sub-filters allow more efficient calculations, similar to polyphase decimation.

Upsampling by an integer factor

Upsampling by an integer factor M is essentially an interpolation problem, where $M-1$ new sample points need to be calculated between two existing adjacent samples. This can be done using the formula for the reconstructed signal (3.5) from the previous section. Resampling methods based on this formula are called *sinc interpolation* methods. However, we must realize that the sinc function (3.6) 'extends infinitely far' in time, and, theoretically, we would need to consider *all* existing samples for the upsampling process. This method is not very efficient, and considering the limitations of floating point arithmetic and the inherent background noise in real-world signals, the results are only marginally better, if at all, than that of windowed sinc interpolation. Still, as the sinc function decays with a rate of $1/t$, we need to consider a relatively large number of nearby points to achieve good results. The Digital Audio Workstation (DAW) software REAPER [16] lists a 64-point sinc interpolation as 'medium' quality, the default option uses 192 points, and the highest quality built-in option uses 768 points. Due to the large number of lookahead samples required, sinc interpolators are not ideal for real-time applications. Instead, other interpolation methods, such as polynomial interpolation [17] can be employed.

To understand the effects of upsampling on audio signals, consider the following.

Definition 29. Let $x \in \mathcal{D}$. The process of inserting $0 < k$ zero-valued samples between the samples of x is called *zero-stuffing*.

Lemma 8. *Any M-factor LTI upsampling method is equivalent to convolving the $M-1$-zero-stuffed input signal with a FIR kernel that is uniquely determined by the method.*

Let us now calculate the DTFT of a zero-stuffed signal y:

$$\widehat{y}(\xi) := \sum_{n\in\mathbb{Z}} y[n]e_{-n}(\xi)\Big|_{[-\pi,\pi]} = \sum_{n\in\mathbb{Z}} x[n]e_{-Mn}(\xi)\Big|_{[-\pi,\pi]}. \tag{3.40}$$

Suppose that $\xi = \frac{\pi}{M}(2k+\delta)$, where $k \in \mathbb{N}_0^+$ and $\delta \in [0,1]$, then

$$\widehat{y}(\xi) = \sum_{n\in\mathbb{Z}} x[n]e_{-Mn}\left(\tfrac{\pi}{M}(2k+\delta)\right)\Big|_{[-\pi,\pi]} = \sum_{n\in\mathbb{Z}} x[n]e_{-n}\left(\delta\pi\right)\Big|_{[-\pi,\pi]} = \widehat{x}(\delta\pi). \tag{3.41}$$

In the case of $\xi = \frac{\pi}{M}(2k+1+\delta)$:

$$\widehat{y}(\xi) = \sum_{n\in\mathbb{Z}} x[n]e_{-Mn}\left(\tfrac{\pi}{M}(2k+1+\delta)\right)\Big|_{[-\pi,\pi]} = \sum_{n\in\mathbb{Z}} x[n]\left(e_{-n}\left(\pi-\delta\pi\right)\Big|_{[-\pi,\pi]}\right)^{*} = \left(\widehat{x}(\pi-\delta\pi)\right)^{*}. \tag{3.42}$$

Therefore the frequency spectrum of the zero-stuffed signal consists of M alternating mirrored replicas of the original signal's, that span the entire frequency range up to Nyquist.

Based on these results, we can say that the quality of a LTI upsampling method depends on how well its equivalent FIR filter can suppress these unwanted spectral images. As we will see in a later chapter, polynomial interpolation methods often turn out to be LTI upsampling methods, therefore, they can be studied as we have just demonstrated.

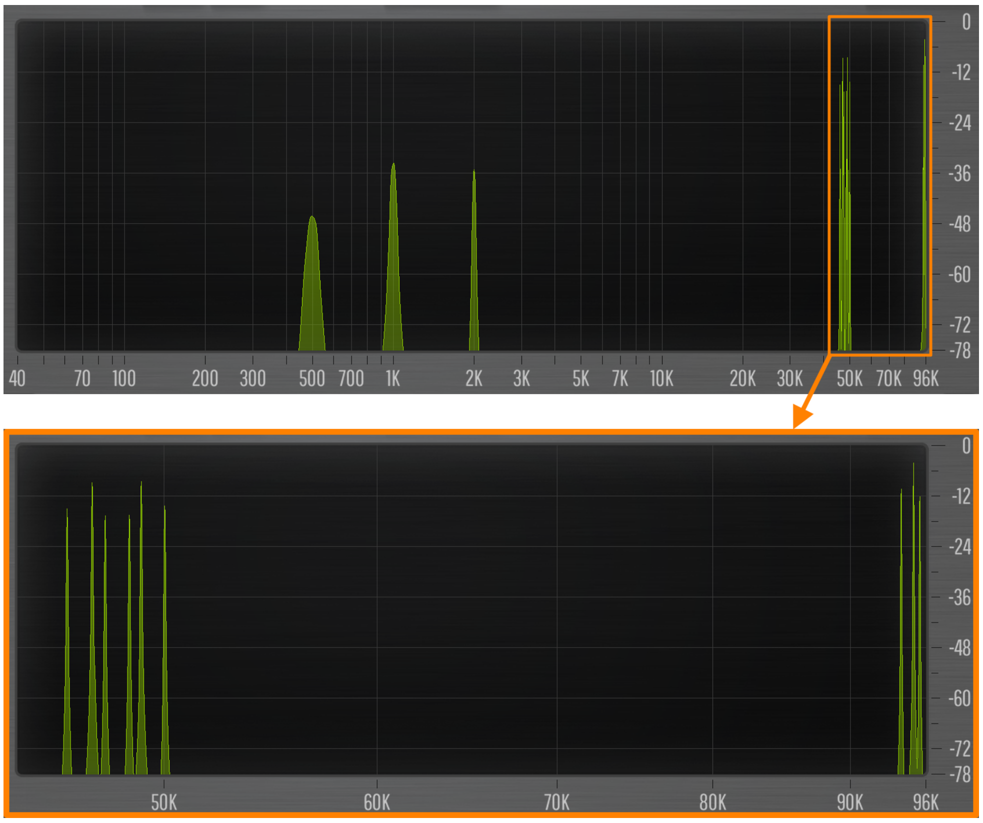


Figure 3.4. Magnitude spectrum of a 3-zero-stuffed signal made of three overlaid sine waves of frequencies 500Hz, 1000Hz and 2000Hz.

Chapter 4

Discretization methods

So far we have looked at the continuous-time modeling of audio circuits and introduced the basic principles of discrete-time systems and signals. In this chapter, we will take a look at some discretization methods that we can use to convert continuous-time systems into discrete-time systems.

4.1 Numerical differentiation of musical signals

In the ODE system (2.8) the first derivative of v_{in} appears on the right-hand side. v_{in} denotes the continuous-time input signal of the TS808 pedal. This signal will be fed to our final software emulation in a digital form, thus we need a way of approximating the derivative of v_{in} from its samples. This is a common problem in digital signal processing.

The most common way of approximating derivatives in discrete-time is via finite differences. Let $x[n]$ ($n \in \mathbb{Z}$) denote the uniform sampling of signal $x_c(t)$ with sampling period $T > 0$. The first derivative of x_c can be approximated using the 7-point stencil central-difference formula:

$$\dot{x}_c(nT) \approx \frac{-x[n{-}3] + 9x[n{-}2] - 45x[n{-}1] + 45x[n{+}1] - 9x[n{+}2] + x[n{+}3]}{60T} + \mathcal{O}(T^6) \quad (4.1)$$

A superior method for approximating the first and second derivatives of discrete musical signals was proposed by Marcin Lewandowski [18]. By considering the round-off and truncation error of floating-point arithmetic and using the step size of numerical differentiation as a regularization parameter, the proposed method is able keep a consistent error ratio of approximately 10^{-13} across the full audio spectrum. However, this method is not suitable for real-time applications without modification, as it requires a relatively large number of lookahead samples, thus increasing the latency of the system.

Still, the empirical data presented in [18] is valuable to us, as it helps in understanding the limitations of numerical differentiation methods. Figure 3 shows the maximum relative error of the 7-point stencil central-difference formula (4.1) for simple sinusoidal signals for different step sizes. The examined step sizes are $h = 0.001T$, $h = 0.01T$, $h = 0.1T$, $h = T$, $h = 10T$, and $h = 100T$, where $T = \frac{1}{44100}$. We can see that the maximum relative error for step size $h = T$ is approximatly 10^{-10} in the worst case over the entire audio spectrum, with an ideal error of approximately 10^{-13} at 6000Hz. This is a sufficient approximation for our purposes, especially when considering the low computational cost of this method, and that only 3 lookahead samples are needed, thus minimizing the latency of the system.

4.2 Lookup tables and interpolation

The use of lookup tables is a common technique in digital signal processing to approximate nonlinear functions that may be too expensive computationally for real-time applications. A lookup table is a precomputed array of values that can be used to approximate a continuous function by interpolating between the values in the table. Depending on the application, the lookup table can be one-dimensional or multi-dimensional, while the most common interpolation methods are linear, polynomial, or spline-based.

Besides the computational efficiency, lookup tables can also be used to more closely capture the behavior of circuit elements whose mathematical model may be incomplete or otherwise cumbersome to work with. A good example of this is the implicit diode equation, which realistically takes into account the internal resistance of the diode at the cost of increased complexity.

It is important to note that the implicit diode equation describes the steady state I-Vrelationship of a diode, but it does not account for the transient behavior of the diode, which is outside the scope of this thesis.

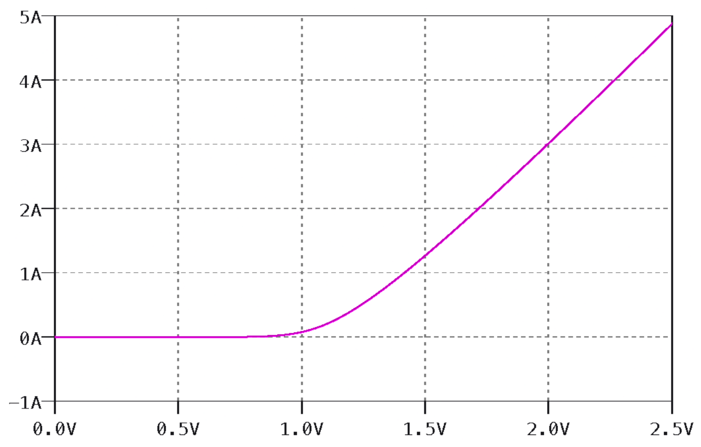


Figure 4.1. Steady state I-Vcharacteristics of the 1N4148 diode found in the TS808 pedal.

Data points for a lookup table can come from physical measurements, or may be generated by computer programs. In lieu of measurement instruments and hands-on experience, I used LTspice, a SPICE (Simulation Program with Integrated Circuit Emphasis) software, to calculate the steady-state I-V curve of the 1N4148 diode from -5V to 5V, in 0.001V increments. We are now free to transform this table as we please.

Depending on the requirements, we must decide what interpolation method we want to use for the final lookup table. A common method is to approximate the data points piecewise using interpolating polynomials, whose coefficients are precalculated. But how do we split the function's domain into intervals? In other words, how do we find a handful of basepoints, such that the piecewise interpolating polynomials are optimal, in a certain sense? We may have a size constraint on the lookup table, which would raise the question; how to choose N basepoints, so that the maximum error of interpolation is minimized? Maybe we are given a desired maximum error, and the question becomes; what is the least amount of basepoints needed to attain the desired maximum error of interpolation? This is an interesting optimization problem in itself, but sadly, I did not have the time to look into the topic.

4.3 Digital filter design

Digital filter design is the process of determining the coefficients of a discrete-time system that satisfies a set of frequency-domain specifications. These specifications typically include constraints on the passband (the range of frequencies to be preserved), the stopband (the range of frequencies to be attenuated), and the transition band (the region between passband and stopband). Additional criteria may include maximum allowed ripple within the passband, minimum required attenuation in the stopband, or constraints on the phase response—such as linear phase or minimum phase. To meet these criteria, various design methods can be employed, most of which follow the same four basic steps:

1. specify the desired filter response,
2. choose an allowed class of digital filters, e.g. N-th order minimum phase IIR filters,
3. establish a measure of goodness for the response of an allowed filter compared to the desired response,
4. develop a method to find the best member of the allowed class of filters.

This process may be repeated iteratively to refine the design criteria and thus the resulting optimal filter as well. Classical techniques include windowing the ideal impulse response for FIR filters, or transforming analog prototype filters e.g. via the bilinear transform for IIR filters. More flexible, optimization-based methods—such as least-squares fitting, equiripple design, Parks-McClellan algorithm—allow for precise control over error distribution across the frequency range.

4.4 Wave Digital filters

Wave Digital filters—or WDF for short—is an emerging, if not already well-established, framework in DSP, and thus I feel obligated to at least introduce the basic concepts in this work. Classical WDF theory originates from the 1970s [19], and with recent extensions and advancements, such as [20–23], it has gained popularity in Virtual Analog modeling as well. I would like to express my gratitude and respect to Kurt James Werner, whose doctoral dissertation on Wave Digital filters [24] was of great help in understanding the subject.

One powerful aspect of Wave Digital filters is their modularity and reusability as well as their quick prototyping ability. Electrical components are discretized in advance, turning them into WDF 'building blocks' which can be 'assembled' into a model of a physical electrical circuit. Once assembled, these building blocks define a well-ordered set of calculations that can be directly turned into computer programs to simulate the reference circuit. WDFs also possess excellent stability and other numerical properties [24].

In WDF theory, electrical components are described with the notion of *ports*. A port is defined by two terminals of an electrical component, such as the terminals of a resistor. More precisely, a port is an abstraction used to describe energy flow within the system, and in the context of WDFs, it provides the foundation for defining so-called *wave variables* and deriving the system's scattering relations, i.e. the mutual interaction between the components of the circuit. The state of a port at any given instance in time can described by two Kirchhoff variables, voltage (v) and current (i). However, in WDF theory, Kirchhoff variables are replaced by the wave variables

$$\begin{aligned} a(t) &:= R^{\rho-1}v(t) + R^{\rho}i(t), \\ b(t) &:= R^{\rho-1}v(t) - R^{\rho}i(t), \end{aligned}$$

where $R > 0$ is a free parameter, called *port resistance*, and ρ is usually chosen from the set $\{1, \frac{1}{2}, 0\}$ based on convention. In matrix form:

$$\begin{pmatrix} 1 & R \\ 1 & -R \end{pmatrix} \ (\rho = 1) \ \sim \ \begin{pmatrix} \frac{1}{\sqrt{R}} & \sqrt{R} \\ \frac{1}{\sqrt{R}} & -\sqrt{R} \end{pmatrix} \ (\rho = \tfrac{1}{2}) \ \sim \ \begin{pmatrix} \frac{1}{R} & 1 \\ \frac{1}{R} & -1 \end{pmatrix} \ (\rho = 0)$$

a is called the *incident wave*, and b is called the *reflected wave*. It is clear that this basis transformation is invertible due to the requirement $R > 0$. Thus we introduced a degree of freedom for every port in the system, which we can—and will—take advantage of. A crucial piece of information is that, in many cases, the two variables—be it Kirchhoff or wave variables—are not independent; they are 'tied' by the I-Vrelationship of the electrical components. This may be demonstrated as follows. Consider a resistor, whose

I-Vcharacteristics can be found in Appendix A. Let us define the three-dimensional 'I-V-curve' of the resistor based on Kirchhoff variables as the $\gamma : [0, +\infty) \to \mathbb{R}^3$ function

$$\gamma(t) := \begin{bmatrix} t \\ v(t) \\ i(t) \end{bmatrix} = \begin{bmatrix} t \\ v(t) \\ r^{-1}v(t) \end{bmatrix}.$$

Thus the graph of voltage over time may be found as the projection of the curve γ onto the xy-plane along the z axis. The curve γ however, can be unambiguously reconstructed from this projection, as the z coordinate is given by the I-Vequation of the resistor as $r^{-1}v$. Thus we can also see that, for all t, the point $\gamma(t)$ lies on the $y = zr$ plane. Using wave variables, the I-Vequation of the resistor can be written as

$$b(t) = \frac{r - R}{r + R} a(t). \tag{4.2}$$

By setting the port resistance $R = r$, the above equation reduces to

$$b(t) = 0. \tag{4.3}$$

In three dimensions, this may be viewed as the basis transformation

$$A := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & r \\ 0 & 1 & -r \end{pmatrix}$$

if $\rho = 1$, thus γ becomes

$$\gamma_A(t) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & r \\ 0 & 1 & -r \end{pmatrix} \begin{bmatrix} t \\ v(t) \\ r^{-1}v(t) \end{bmatrix} = \begin{bmatrix} t \\ 2v(t) \\ 0 \end{bmatrix}.$$

Thus we saw how the port resistance—an added degree of freedom—can be used to choose a new basis which is advantageous to us. This is a key concept in WDF theory. Equation (4.2) is called the *unadapted* continuous-time wave-domain resistor equation, and equation (4.3) is called the *adapted* continuous-time wave-domain resistor equation.

To arrive at the WDF 'building block' for a resistor, we must discretize these equations. In the case of the resistor, this is easily done:

$$b[n] = \frac{r - R}{r + R} a[n], \tag{4.4}$$

where $a[n] := a(nT)$ and $b[n] := b(nT)$ for $n \in \mathbb{N}$ and $T > 0$.

However, this discretization process may not be so easy for other electrical components. Let us look at the capacitor for example. The Kirchhoff-domain capacitor I-Vequation is defined as

$$i(t) = c\dot{v}(t). \tag{4.5}$$

The usual approach [24] is to transform equation (4.5) to the Laplace-domain:

$$I(s) = csV(s), \tag{4.6}$$

then plug the parametric wave definition into (4.6) to get

$$\frac{1}{2}R^{-\rho}(A(s) - B(s)) = \frac{1}{2}cR^{1-\rho}s(A(s) + B(s)), \tag{4.7}$$

then rearrange to get the Laplace- and wave domain transfer function

$$H(s) = \frac{B(s)}{A(s)} = \frac{1 - Rcs}{1 + Rcs}. \tag{4.8}$$

This equation can then be used to find a discrete-time system with a sufficiently similar transfer function. In the case of the capacitor, this is usually an IIR system, given by a LCCDE (3.29).

Other electrical components can be turned into building blocks in a similar manner. There are other types of building blocks as well, which are called *adaptors*. These are usually multi-port blocks, and can serve various purposes, such as modeling the interconnection of individual electrical components (e.g. parallel or series connections).

As the building blocks are assembled, a WDF *tree* is formed. The choice of the root of the tree is somewhat arbitrary, though if a non-adaptable and/or nonlinear electrical component—or subcircuit—is involved, then it must be chosen as the root, and all other components must be adaptable. We will return to the precise definition of adaptable blocks shortly.

Each building block 'contributes' to the computational block diagram of the tree; the diagrams of the individually discretized electrical devices are 'sewn together' to form a whole. However, we must be careful not to introduce delay-free loops, or else the resulting system is non-computable due to Lemma 4. The key notion to underline here is that we can use the port resistances of—most, but not all—WDF blocks to eliminate delay-free loops that may arise in the assembled WDF tree block diagram. This is what we mean by *adapting* a WDF building block.

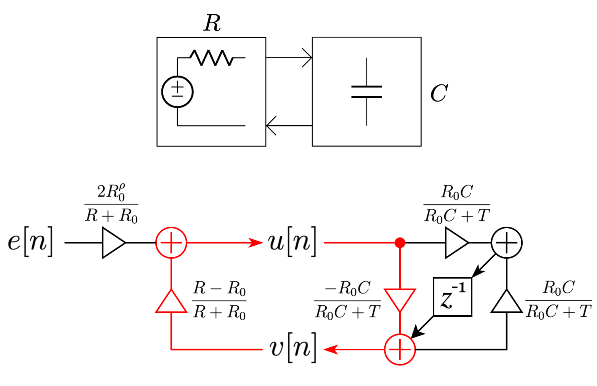
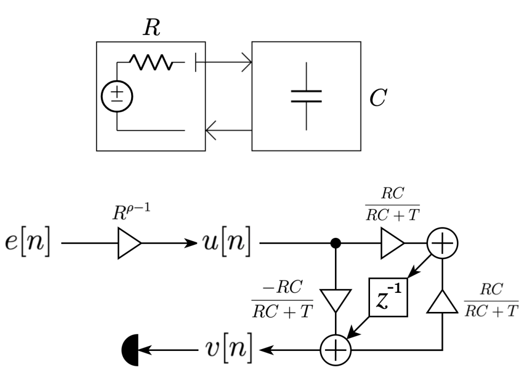
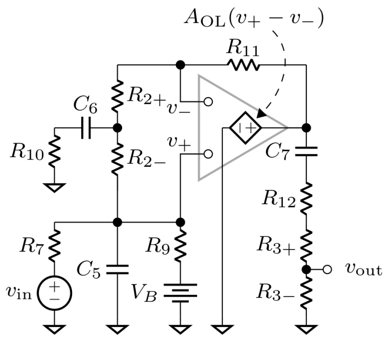
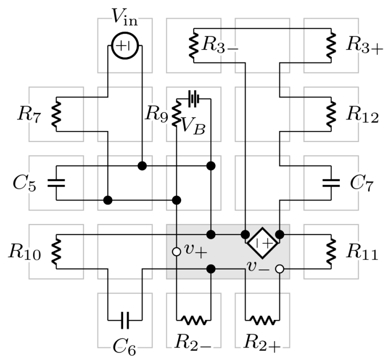



Figure 4.2. WDF formulation of an RC low-pass filter, with unadapted and adapted resistive voltage source.


(a) TS808 tone/volume circuit


(b) Rearranged to highlight structure

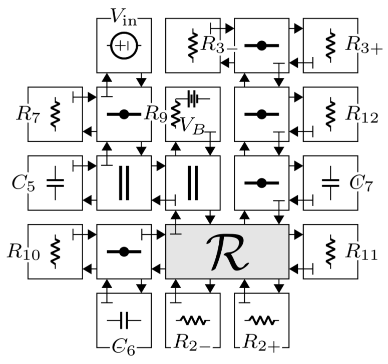

(c) Corresponding WDF structure

Figure 4.3. WDF formulation of the Ibanez TS808 tone/volume circuit. Figures courtesy of Kurt James Werner, used with permission.

4.5 Numerical methods for ODEs

Continuous-time differential equations can be discretized using various numerical methods. For real-time applications, however, we must be careful to choose a method that is both computationally efficient and stable. *Predictability* is key. We must be able to prove theoretically that the computational cost and the stability of our solution meet certain requirements. Therefore, adaptive step size methods are not suitable for our purposes. Implicit methods are preferred over explicit methods, as they are more stable and less prone to numerical errors, e.g. accumulation of floating-point errors. The step size is usually dictated by the sampling rate of the system, which may be too large for explicit methods to be stable. This is especially true for nonlinear ODEs, as we will see in the next chapter. The most common implicit methods are the trapezoidal rule and the backward Euler method.

Explicit methods can still be useful when designing real-time systems though. For example, they can be used to precompute the parameters of a system, or for high-precision offline (slower than real-time) simulations.

Chapter 5

Putting it all together

So far, we've looked at many mathematical models and methods that appear frequently in digital signal processing. Let us now apply this knowledge to construct a real-time emulation of the TS808 circuit. All benchmarks and experiments were run on a 2014 Apple MacBook Pro machine powered by a quad-core Intel Core i7 CPU running at a base frequency of 2.2GHz.

5.1 Emulating the diode clipper circuit

5.1.1 Verifying the theory

The mathematical model of the diode clipper circuit is given by equation (2.8). Firstly, we should verify that this equation is, in fact, correct. To do this, the clipping stage was modeled in LTspice, which can be seen in Appendix B. The SPICE model of the 1N4148 diode was downloaded from the manufacturer's website [25]. A 10-second long electric guitar signal was passed through the clipping stage in a SPICE transient simulation. The direct guitar signal was recorded through a Line 6 Helix LT. It is very important to know, what kind of analog to digital conversion took place when recording an analog signal, as we need to 'reconstruct' the original amplitude of the signal. In this case, the conversion was done within the Helix device, with an input sensitivity of 11dBu FS, which amounts to about 7.773V peak-to-peak. Hence the input signal in the SPICE simulation is scaled with a multiplier of $\frac{7.773}{2} \approx 3.88$. The instantaneous voltages at different nodes of the circuit were exported to 1536kHz wave files, that were postprocessed with a Python script to eliminate any DC offset and decimate the signals to 48kHz. When LTspice exports wave files, 1V amplitude constitutes a full-scale sample, thus, to avoid clipping the output, these voltages were scaled down by a factor of 10. A C++ application was created, which calculated branch currents for the components in the feedback network, namely, capacitor C_f, resistors R_g and R_f, and the two diodes. The for the diode currents, the 10001-length lookup table was used from the previous chapter. The resistor currents were directly calculated from their branch constitutive equations. For the capacitor current calculation, I used the 7-point stencil central finite-difference formula for the estimation of the first derivatives. From these values, the current through R_g ($I(R_g)$) and through the feedback network ($I(R_f) + I(D_1) + I(D_2) + I(C_f)$) were calculated and compared. The results can be seen in Appendix B. We can see from the graphs that the two currents are, in fact, in very close agreement. A constant, but very small—compared to the signal's magnitude—difference can be seen from beginning to end. This is probably due to the high precision of the SPICE simulation, which treats the operational amplifier with more detail. We can also see small deviations near transient events. This is to be expected, the SPICE simulation handles transient events in much more detail, as it uses adaptive time steps, whereas the C++ application runs with a constant $h = \frac{1}{48000}$ time step. Nevertheless, these results look promising, and indicate that we are on the right track.

5.1.2 Requirements

The next step is to discretize the ODE system defined by (2.8). In other words, we are looking for a $P_{\text{clip}} : \mathcal{D} \to \mathcal{D}$ discrete-time system, that is

i) causal,

ii) BIBO-stable,

iii) if $x[n]$ is the uniform sampling of the input signal x_c, then $P_{\text{clip}}\{x\}[n]$ approximates the uniform sampling of the 'OpClip' op-amp output voltage to the given input signal x_c.

Causality is not much of a design choice. Classical computers are, by definition, causal. Any theoretical discrete-time system that uses a finite number N of lookahead samples, can be turned into a causal system by adding a length-N delay to the system. In other words, if our algorithm depends on N future input samples to calculate the current output, we have no choice, but to wait until those samples become available, before we can continue the computation. This is called the *latency* of the system. One example of added latency is the group delay of FIR filters. Since we are looking for a real-time emulation, we can only afford so many lookahead samples, before our software becomes unusable for guitarists. My personal experience is that a 1-2 millisecond latency is unnoticable, while ≥ 10 milliseconds is very noticeable and quickly becomes distracting for playing music.

Digital signal processing happens in so-called *blocks*, due to the architecture of digital computers. These blocks usually contain 2^B samples ($B \in \mathbb{N}^+$), which become available for processing at regular intervals, depending on the block size and the sampling frequency settings. This is usually configurable on a desktop PC, while dedicated devices, such as digital guitar effects pedals, use fixed settings optimized for real-time audio. One example of such a setting for a desktop PC is perhaps a block size of 32 samples with a sampling frequency of 48kHz. Even without any additional processing, this introduces a base latency of $\frac{32}{48} \approx 0.67$ milliseconds. The total *roundtrip latency* is at least *input latency* plus *processing latency*. Therefore it is imperative that we design clever algorithms that introduce as little latency as possible. It is safe to assume that the software emulation will be run in a $T_{\text{FS}} = 48000\text{Hz}$ environment.

5.1.3 Calculating the voltage at node Y

Let us start by solving for $\dot{v}_Y$ in (2.8). One solution would be to discretize the ODE

$$\dot{v}_Y = \dot{v}_{in} - \frac{v_Y}{r_g c_g}, \tag{5.1}$$

perhaps via the implicit Euler method:

$$v_Y[n] = K v_Y[n-1] + hK\dot{v}_{in}[n], \tag{5.2}$$

where $h := \frac{1}{T_{\text{FS}}}$, and $K := (1 + \frac{h}{r_g c_g})^{-1} \approx 0.914$. $\dot{v}_{in}$ can be approximated using numerical differentiation.

However, we can get a better solution if we consider that the capacitor C_g and resistor R_g form a passive first order RC high-pass filter network. The transfer function of this network is given by

$$H(s) = \frac{r_g}{(c_g s)^{-1} + r_g}, \tag{5.3}$$

and the cutoff frequency—the frequency at which 3db attenuation occurs—is given by

$$\omega_c = \frac{1}{2\pi r_g c_g} \approx 720.48\text{Hz}. \tag{5.4}$$

While the derivation and methodology in greater detail can be found in [14], article [26] provides a shorthand formula for turning RC filter networks into digital IIR filters. This solution requires no lookahead samples, and can be implemented using 2 multiplications and 2 additions per output sample, regardless of the sampling rate. It is also more accurate than the previous solution, when compared to the SPICE simulated voltage at node Y.

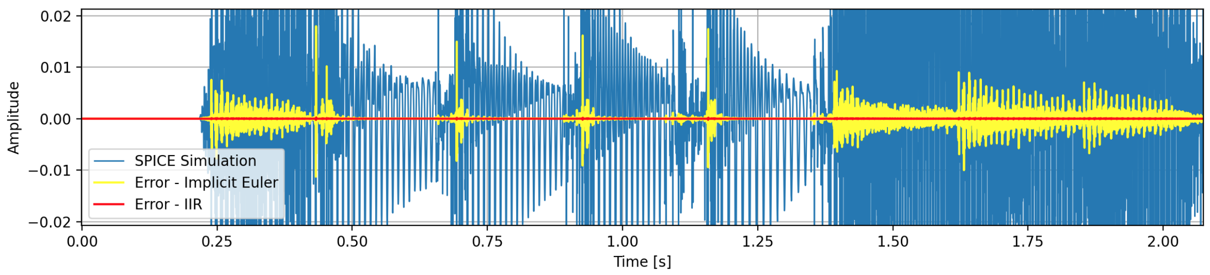


Figure 5.1. Error of implcit Euler method vs. IIR for calculating v_Y.

5.1.4 Solving the diode clipper equation

Using the previous result, we can focus on the diode clipper equation,

$$\dot{v}_{out} = \dot{v}_{in} + \frac{v_Y}{r_g c_f} - \frac{\Delta}{r_f c_f} - \frac{I_{D\downharpoonleft\upharpoonright}(\Delta)}{c_f}, \tag{5.5}$$

where

$$\Delta(t) \coloneqq v_{out} - v_{-}. \tag{5.6}$$

We will use the 7-point stencil central finite difference method (4.1) to estimate $\dot{v}_{in}$. This requires 3 lookahead samples, which is a good tradeoff given the observed accuracy of the method [18].

As for the diode currents, we have two viable options. We can either use the explicit formulation (2.9), thus equation (5.5) becomes

$$\dot{v}_{out} = \dot{v}_{in} + \frac{v_Y}{r_g c_f} - \frac{\Delta}{r_f c_f} - \frac{2 I_S \mathrm{sinh}\big(\Delta(nV_T)^{-1}\big)}{c_f}, \tag{5.7}$$

or we can opt for a lookup table, and 'worry about it later'.

Out of curiosity, I tried both options. First, as a naive approach, I implemented the fourth-order explicit Runge-Kutta method—also known as 'RK4'—in C++, to solve equation (5.7). For this experimentation, to be able to evaluate the right-hand side at 'half time steps', I oversampled the input signal beforehand, using the 768-point sinc interpolation method of REAPER [16], as a high-quality oversampling method. The solution quickly diverged to infinity. Could it be due to the hyperbolic sine function on the right-hand side? Possibly. The simulated diode I-V curve is very different to the explicit formulation, which describes a pure exponential relation between current and voltage.

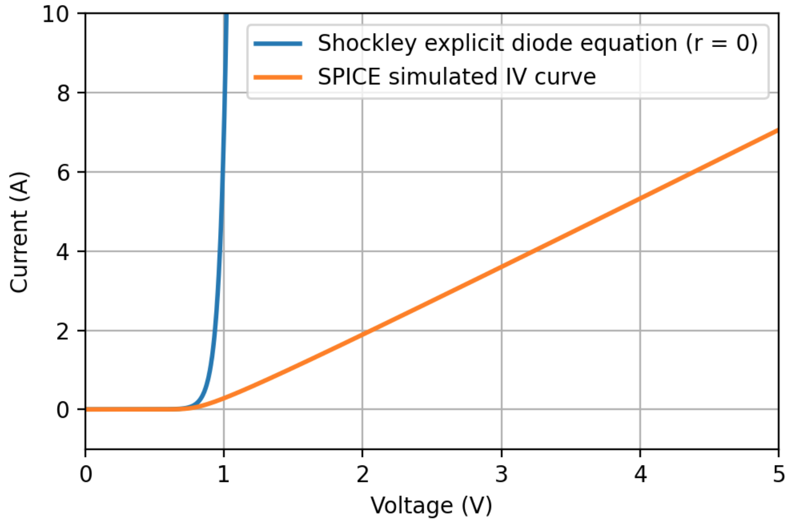


Figure 5.2. Shockley explicit diode equation with $V_T = 26.77\text{mV}$, $n = 1.92$ vs. SPICE simulated I-V curve.

I modified the C++ code to use the precomputed lookup table instead, but the results were the same. Therefore we can say that equation (5.5) is *stiff*.

Despite the indication that explicit methods may not be the best fit for this problem, I experimented with more capable explicit methods. [27] describes a new, fifth-order rational method with promising capabilities. Since this method is rather complicated, I set out to reproduce one of the experiments in the article to test my C++ implementation. Keeping the notations similar to the article, the method was reformulated as:

$$y_{n+1} = \frac{\begin{array}{c}\omega y^2 + \big(2\beta + (\gamma + 480fc^2)h + \delta\big)y - \\ 72h\big((f^3e - 5f^2bd - \frac{10}{3}f^2c^2 + 15fb^2c - \frac{15}{2}b^4)h - 5f^3d + 20f^2bc - 15fb^3\big)\end{array}}{\omega y + (60bcd - 18b^2e - \alpha f - 40c^3)h^3 + \beta + (\gamma + 180fbd + 240fc^2)h + \delta}, \tag{5.8}$$

where

$$\begin{aligned}
\alpha &:= 15d^2 - 12ce, \\
\omega &:= \alpha h^2 + (36be - 60cd)h - 180bd + 240c^2, \\
\beta &:= \big(-90b^2d + (36fe + 120c^2)b - 60fcd\big)h^2, \\
\gamma &:= -72f^2e - 360b^2c, \\
\delta &:= 360f^2d - 1440fbc + 1080b^3.
\end{aligned}$$

Problem 3 describes the following nonlinear logistic growth model:

$$\begin{cases} \dot{y}(t) &= \frac{y(t)}{4}\left(1 - \frac{y(t)}{20}\right), \\ y(0) &= 1, \end{cases} \tag{5.9}$$

with true solution

$$y(t) = \frac{20\exp(t/4)}{19 + \exp(t/4)}. \tag{5.10}$$

I calculated the required formulae for the method as:

$$\begin{cases} f^{(0)} &:= \frac{y}{4}\left(1 - \frac{y}{20}\right), \\ f^{(1)} &:= \left(\frac{1}{4} - \frac{y}{40}\right) f^{(0)}, \\ f^{(2)} &:= \left(\frac{1}{4} - \frac{y}{40}\right) f^{(1)} + \frac{(f^{(0)})^2}{40}, \\ f^{(3)} &:= \left(\frac{1}{4} - \frac{y}{40}\right) f^{(2)} + \frac{f^{(1)}f^{(0)}}{40}, \\ f^{(4)} &:= \left(\frac{1}{4} - \frac{y}{40}\right) f^{(3)} + \frac{(f^{(1)})^2}{40}. \end{cases} \tag{5.11}$$

I successfully reproduced the results for Problem 3, which can be seen in Figure 5.3.

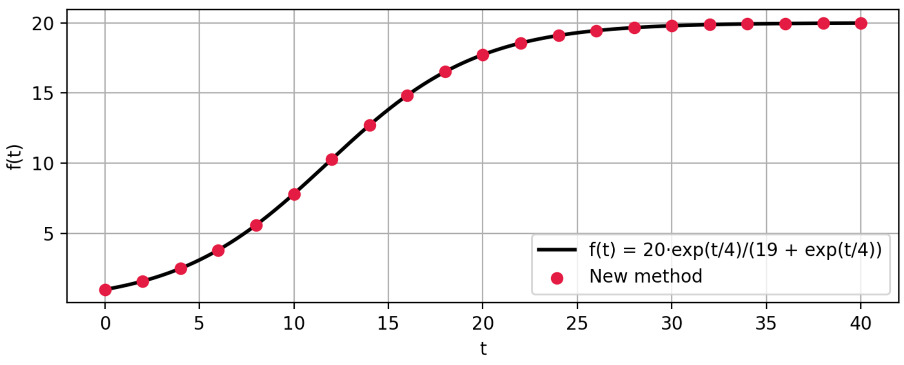


Figure 5.3. True solution vs. numerical solution of (5.9) using the new numerical method [27].

I then implemented this method for the diode clipper equation (5.7). Using the short-hand notations

$$\begin{aligned} y(t) &:= \frac{v_Y(t)}{r_g c_f}, \\ b &:= \frac{-1}{r_f c_f}, \\ d &:= \frac{1}{nV_T}, \\ S(t) &:= \frac{-2I_S \sinh(\Delta d)}{c_f}, \\ C(t) &:= \frac{-2I_S \cosh(\Delta d)}{c_f}, \end{aligned}$$

the derivatives of the right-hand side were calculated as follows.

$$\begin{cases} f^{(0)} & := y + b\Delta + S, \\ f^{(1)} & := y^{(1)} + (b + Cd)f^{(0)}, \\ f^{(2)} & := y^{(2)} + (b + Cd)f^{(1)} + (f^{(0)}d)^2 S, \\ f^{(3)} & := y^{(3)} + (b + Cd)f^{(2)} + 3f^{(1)}f^{(0)}d^2 S + (f^{(0)}d)^3 C, \\ f^{(4)} & := y^{(4)} + (b + Cd)f^{(3)} + 4f^{(0)}f^{(2)}d^2 S + \\ & \qquad 6(f^{(0)})^2 f^{(1)} d^3 C + 3(f^{(1)}d)^2 S + (f^{(0)}d)^4 S. \end{cases} \tag{5.12}$$

The results were similar, the method diverges rather quickly.

Both explicit methods may have been convergent using shorter time steps, which I did not try. Though the tested signal may have been processed successfully, the stability of the resulting algorithms would be quite difficult to reason about.

This result seems to be not uncommon in Virtual Analog modeling, when dealing with nonlinear equations. Article [8] describes a possible solution to a similar diode clipper circuit equation using an implicit numerical method instead. In their work, the simplified explicit diode model (2.10) is used. First, the diode clipper equation is discretized by the generalized 1-step linear implicit method. Then, to solve for the diode current, the resulting implicit equation is reformulated using the Lambert W function

$$W(z) := f^{-1}(z), \tag{5.13}$$

where $f : \mathbb{C} \to \mathbb{C}$ is

$$f(z) := ze^{z}. \tag{5.14}$$

Among many other applications of the W function [28], it can be used to solve equations of the form

$$e^{ax+b} = cx + d, \tag{5.15}$$

with the solution—if exists—being

$$x = \frac{W\left(\frac{-a}{c}e^{b-\frac{ad}{c}}\right)}{-a} - \frac{d}{c} \qquad (a, c \neq 0). \tag{5.16}$$

In their work, D'Angelo et al. proposed several methods to approximate the values of W efficiently using polynomial approximation and Newton-Raphson iteration.

However, their formulation of the diode clipper equation does not account for the internal resistance of the diodes. Trying to solve for the implicit diode equation analytically would add another layer of difficulty, and would possibly require the use of iterative methods as a subroutine for every time step. Instead, we carry on with our solution, keeping the function symbol $I_{D\downharpoonleft\upharpoonright}$ inside the equations, and 'see where we end up'.

Let us discretize (5.5) using the implicit Euler method:

$$v_{out}[n] = v_{out}[n-1] + h\left(\dot{v}_{in}[n] + \frac{v_Y[n]}{r_g c_f} - \frac{\Delta[n]}{r_f c_f} - \frac{I_{D\downharpoonleft\upharpoonright}(\Delta[n])}{c_f}\right), \tag{5.17}$$

where

$$\Delta[k] := v_{out}[k] - v_{-}[k]. \tag{5.18}$$

Rearrange to get

$$I_{D\downharpoonleft\upharpoonright}\big(\Delta[n]\big) = -\Delta[n]\left(\frac{c_f}{h} + \frac{1}{r_f}\right) + \frac{c_f}{h}\big(v_{out}[n-1] - v_{in}[n]\big) + c_f\dot{v}_{in}[n] + \frac{v_Y[n]}{r_g}, \tag{5.19}$$

or, in a more elegant form,

$$I_{D\downharpoonleft\upharpoonright}\big(\Delta[n]\big) = -A\Delta[n] + C, \tag{5.20}$$

where

$$A := \left(\frac{c_f}{h} + \frac{1}{r_f}\right),$$
$$C := \frac{c_f}{h}\big(v_{out}[n-1] - v_{in}[n]\big) + c_f \dot{v}_{in}[n] + \frac{v_Y[n]}{r_g}.$$

Let us inspect equation (5.20) in more detail. The solution of this equation is exactly the x coordinate of the 2-dimensional point where the line $y = -Ax + C$ intersects the antiparallel diode I-V curve. This point exists and is unique, since $\frac{d}{dx} I_{D\downarrow\uparrow} \geq 0$, while $\frac{d}{dx}(-Ax + C) = -A < 0$. This is a very favorable outcome, as this equation can be solved easily and efficiently by using a precomputed lookup table for the antiparallel diode current. If $I_{D\downarrow\uparrow}$ is approximated piecewise with first-order polynomials, i.e. with a polyline P, the intersecting line segment of P can be found via binary search, and then the exact point of intersection can be calculated, with a single floating-point division being the 'most demanding' computation.

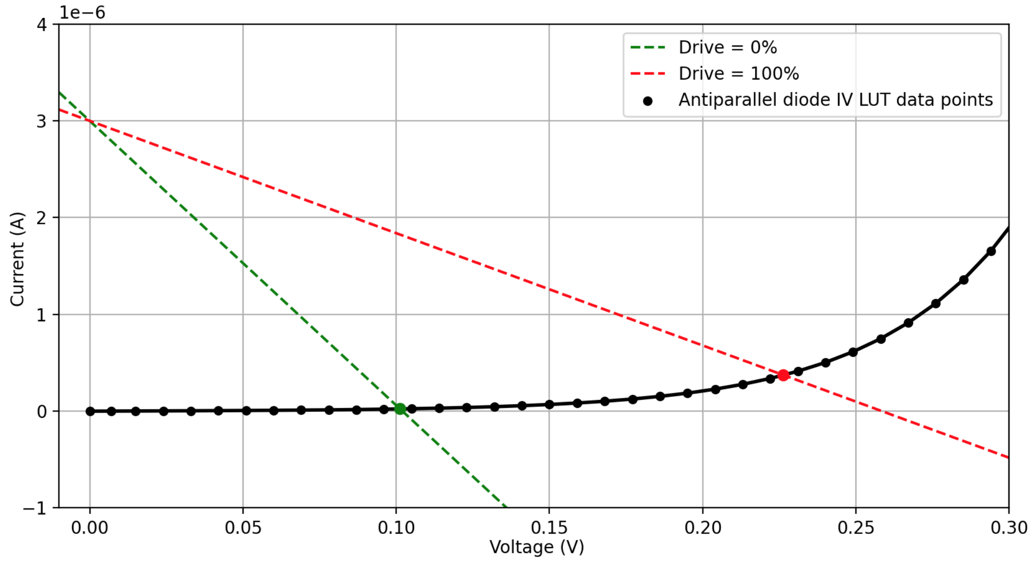


Figure 5.4. Typical example of an I-V intersection calculation during the execution of the diode clipper emulation.

Thus we have found the final solution for the diode clipper equation.

5.2 Stability investigation

Let us now inspect the stability of our solution. For the following calculations we will suppose that our algorithm runs at $T_{\text{FS}} = 192000\text{Hz}$—despite specifying earlier that the target sample rate is 48000Hz, this choice however will be justified shorty. Since the left-hand side of (5.20) is static, we focus on the right-hand side. We will suppose that $C > 0$.

If $\Delta[n]$ is a solution of (5.20), and $\Delta'[n]$ is a solution of

$$I_{D\Downarrow\Uparrow}(x) = C, \tag{5.21}$$

then

$$|\Delta[n]| \leq |\Delta'[n]|. \tag{5.22}$$

Let us suppose that the input signal v_{in} is bounded, and is in the range of -5V to 5V. Since the IIR system used to calculate v_Y is stable, and its magnitude response is not greater than 1, the following upper bound for C can be easily seen:

$$|C| \leq 4 \cdot 10^{-9} |\Delta[n{-}1]| + 28 \cdot 10^{-9} + \frac{1}{940}. \tag{5.23}$$

We can 'cheat' a little bit here, and suppose that, as the last operation in our implementation of the diode clipper simulation, we clamp the output signal to be in the range of -5V to 5V—which is actually quite sensible when considering the working principles of the operational amplifier in the clipping stage, and would obviously ensure that the algorithm is BIBO-stable—the final estimate becomes

$$|C| \leq 68 \cdot 10^{-9} + \frac{1}{940}. \tag{5.24}$$

From this, we can conclude that

$$|\Delta[n]| \leq I^{-1}_{D\Downarrow\Uparrow}(68 \cdot 10^{-9} + \frac{1}{940}) \approx 0.6. \tag{5.25}$$

This result tells us that, as long as we formulate our software so as to avoid a division by a near-zero value when working with the lookup table, i.e. maintain numerical stability, the output waveform should, in theory, follow the input waveform rather closely. This is indeed the case, and the algorithm shows no signs of instability when tested with various sensible input signals.

We can also see now that using the implicit diode equation brings little improvement compared to the explicit formulation, as $\Delta[n] \leq 0.6$ implies that only a very small amount of current is flowing through the diodes, and thus we can see from Figure 5.2 that the two formulations should perform similarly. This also explains why the implicit diode formulation did not help with the stability of the tested explicit methods.

5.3 Multirate signal processing to mitigate aliasing

During my experimentation, I realized that significant aliasing occurs when the diode clipper emulation runs at a rate of 48kHz. This is not surprising, as distortion algorithms generate strong upper harmonics, thus increasing the bandwidth of the signal. The solution to this problem is to oversample the input signal on-the-fly, then feed the oversampled signal to the distortion algorithm, which runs at a higher rate, and then decimate the output signal. This technique is called *multirate signal processing*, and is very common in modern signal processing algorithms with nonlinear behavior.

For the sake of simplicity, we will only cover the case when our emulation runs in a 48kHz environment. We will aim for an emulation rate of 192kHz, therefore we need to oversample the input signal by a factor of 4, and then decimate the output by a factor of 4 as well.

One possible solution is to estimate the derivative of the unoversampled signal—we will need it for the diode clipper emulation anyway—and then incorporate this estimate into the oversampling polynomial. Let us construct a 7-th order Hermite-Fejér interpolating polynomial—named after French and Hungarian mathematicians Charles Hermite and Lipót Fejér—with 8 constraints, using four nearby points, so that

$$\begin{aligned} p(-h) &= x[n_0-1], & p'(-h) &= dx[n_0-1], \\ p(0) &= x[n_0], & p'(0) &= dx[n_0], \\ p(h) &= x[n_0+1], & p'(h) &= dx[n_0+1], \\ p(2h) &= x[n_0+2], & p'(2h) &= dx[n_0+2], \end{aligned} \tag{5.26}$$

where $x[k]$ denotes the unoversampled input signal, and $dx[k]$ denotes the estimated derivatives. We can use p to generate 3-3 new sample points at $t = \frac{1}{4}h$, $t = \frac{1}{2}h$ and $t = \frac{3}{4}h$ to achieve a 4x oversampling of both the input signal $x[k]$ and its derivative $dx[k]$. To do so, we must solve the following system of linear equations:

$$\begin{array}{c} \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & u_1 & u_2 & u_3 & v_1 & v_2 & v_3 \end{bmatrix} \\ \begin{bmatrix} 1 & -h & (-h)^2 & (-h)^3 & (-h)^4 & (-h)^5 & (-h)^6 & (-h)^7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & h & h^2 & h^3 & h^4 & h^5 & h^6 & h^7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2h & (2h)^2 & (2h)^3 & (2h)^4 & (2h)^5 & (2h)^6 & (2h)^7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2(-h) & 3(-h)^2 & 4(-h)^3 & 5(-h)^4 & 6(-h)^5 & 7(-h)^6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2h & 3h^2 & 4h^3 & 5h^4 & 6h^5 & 7h^6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2(2h) & 3(2h)^2 & 4(2h)^3 & 5(2h)^4 & 6(2h)^5 & 7(2h)^6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \frac{1}{4}h & (\frac{1}{4}h)^2 & (\frac{1}{4}h)^3 & (\frac{1}{4}h)^4 & (\frac{1}{4}h)^5 & (\frac{1}{4}h)^6 & (\frac{1}{4}h)^7 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & \frac{1}{2}h & (\frac{1}{2}h)^2 & (\frac{1}{2}h)^3 & (\frac{1}{2}h)^4 & (\frac{1}{2}h)^5 & (\frac{1}{2}h)^6 & (\frac{1}{2}h)^7 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & \frac{3}{4}h & (\frac{3}{4}h)^2 & (\frac{3}{4}h)^3 & (\frac{3}{4}h)^4 & (\frac{3}{4}h)^5 & (\frac{3}{4}h)^6 & (\frac{3}{4}h)^7 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 2(\frac{1}{4}h) & 3(\frac{1}{4}h)^2 & 4(\frac{1}{4}h)^3 & 5(\frac{1}{4}h)^4 & 6(\frac{1}{4}h)^5 & 7(\frac{1}{4}h)^6 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 2(\frac{1}{2}h) & 3(\frac{1}{2}h)^2 & 4(\frac{1}{2}h)^3 & 5(\frac{1}{2}h)^4 & 6(\frac{1}{2}h)^5 & 7(\frac{1}{2}h)^6 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 2(\frac{3}{4}h) & 3(\frac{3}{4}h)^2 & 4(\frac{3}{4}h)^3 & 5(\frac{3}{4}h)^4 & 6(\frac{3}{4}h)^5 & 7(\frac{3}{4}h)^6 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \end{array} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ d_0 \\ d_1 \\ d_2 \\ d_3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.27}$$

where

$$\begin{aligned} x_0 &:= x[n_0-1], & d_0 &:= dx[n_0-1], \\ x_1 &:= x[n_0], & d_1 &:= dx[n_0], \\ x_2 &:= x[n_0+1], & d_2 &:= dx[n_0+1], \\ x_3 &:= x[n_0+2], & d_3 &:= dx[n_0+2], \end{aligned} \tag{5.28}$$

and a_k denote the coefficients of p. The solution of (5.27) calculated symbolically is

$$\begin{aligned} u_1 &= \frac{3283x_0 + 165375x_1 + 25725x_2 + 2225x_3 + h(735d_0 + 33075d_1 - 11025d_2 - 525d_3)}{196608}, \\ u_2 &= \frac{13x_0 + 243x_1 + 243x_2 + 13x_3 + h(3d_0 + 81d_1 - 81d_2 - 3d_3)}{512}, \\ u_3 &= \frac{2225x_0 + 25725x_1 + 165375x_2 + 3283x_3 + h(525d_0 + 11025d_1 - 33075d_2 - 735d_3)}{196608}, \\ v_1 &= \frac{11935x_0 - 174825x_1 + 152145x_2 + 10745x_3 + h(2751d_0 + 44415d_1 - 58905d_2 - 2505d_3)}{147456h}, \\ v_2 &= \frac{-5x_0 - 405x_1 + 405x_2 + 5x_3 + h(-d_0 - 81d_1 - 81d_2 - d_3)}{256h}, \\ v_3 &= \frac{-10745x_0 - 152145x_1 + 174825x_2 - 11935x_3 + h(-2505d_0 - 58905d_1 + 44415d_2 + 2751d_3)}{147456h}. \end{aligned} \tag{5.29}$$

Not too surprisingly, this method is equivalent to convolving the 3-zero-stuffed input signal with the following 39-tap FIR kernel:

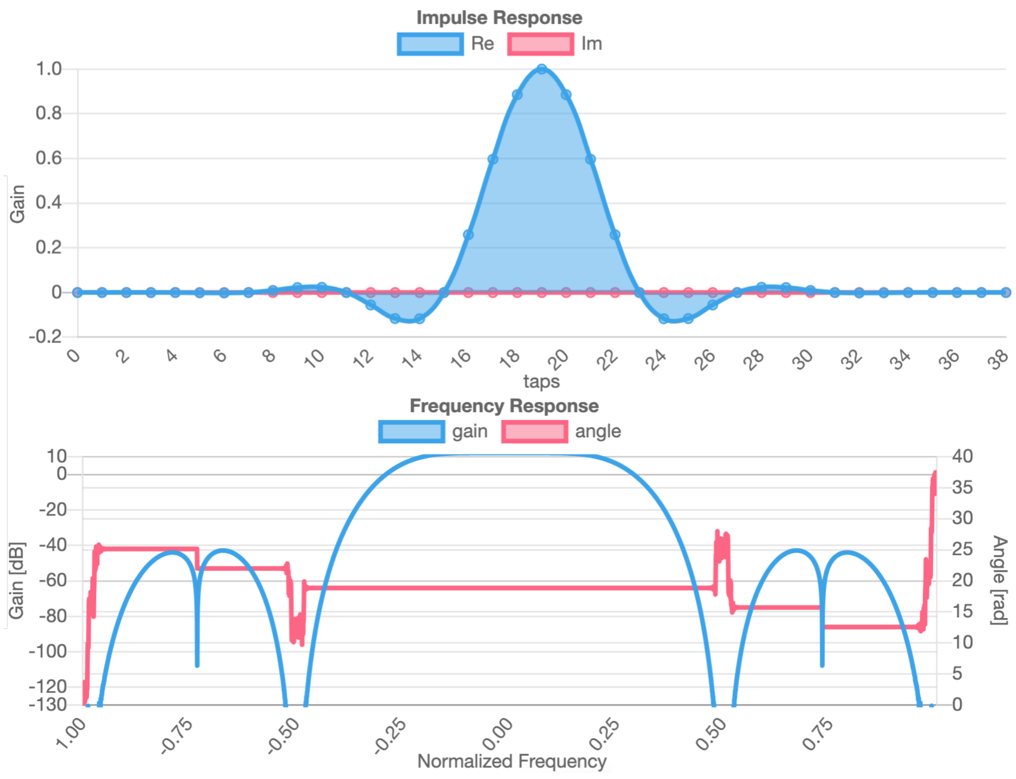


Figure 5.5. Created with `https://cho45.stfuawsc.com/fir-filter-vis/`.

The frequency response plot shows desirable qualities for the proposed method; a flat passband in the audible frequency range, and a strong attenuation near $\pm 0.5\pi$ and $\pm\pi$ to suppress the zero-stuffed spectral images. The passband gain is about 12dB, which describes a 4x amplification. This is logical if we consider that the equivalent FIR interpolator must 'work against' the 3-zero-stuffed input signal. To oversample a block of 32 input samples—resulting in 128-128 samples for the signal and its derivative—this method requires a total of 2240 multiplications and 1952 additions.

The decimation of the processed signal is more straightforward. For this, I used a FIR design tool [29] to design a 4x linear-phase decimation filter, which runs at 192kHz and has 105 taps, and therefore has a group delay of roughly 0.27 milliseconds. It has an excellent 0.13dB passband ripple and -75dB stopband attenuation. In my C++ code, I implemented it using the polyphase decimation technique described earlier in Chapter 3. I also created a small C++ benchmark to test the performance of the decimation algorithm. A 240 seconds long 192kHz audio track was decimated in about 315 milliseconds using the single-phase approach. This was reduced to about 270 milliseconds with the polyphase approach.

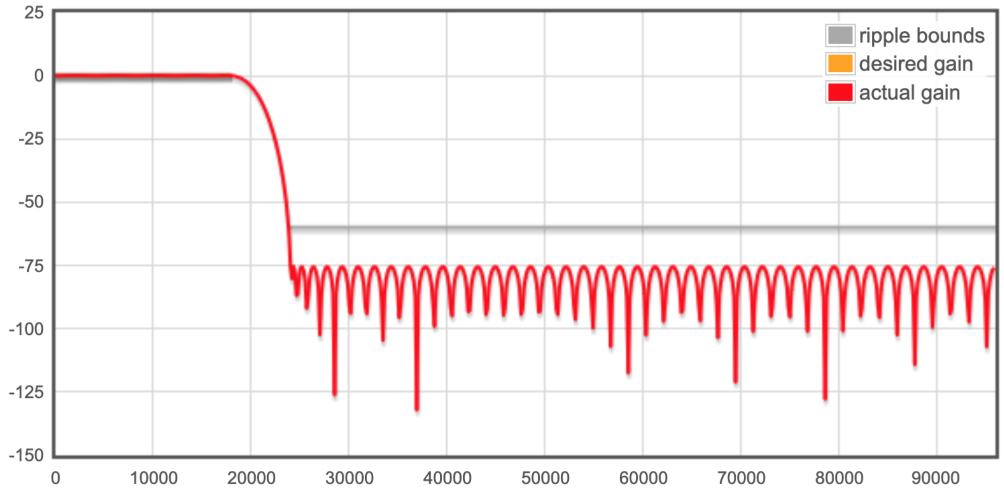


Figure 5.6. The designed 4x decimation filter.

5.4 Emulating the tone/volume section

So far, we have been doing white-box modeling. The same approach would work for the tone/volume section as well; we could use our knowledge of this subcircuit to decompose it into a cascade of LTI filters, perhaps a low-pass filter, followed by a high-frequency shelving filter. These filters may be derived using the 'more conventional' digital filter design approaches presented in [14]. However, I opted for a more general black-box modeling approach out of curiosity.

It should be mentioned that we have the choice of running the tone circuit emulation before or after the decimation process. Our first instict might say that we should run it afterwards to cut down on the amount of required calculations. However, as we will see shortly, the tone circuit emulation is rather lightweight, and running it before decimation can in fact help with aliasing suppression.

Suppose that we only have knowledge about the frequency response of this subcircuit for a set of tone knob positions that span the $[0, 1]$ interval. Let $p_0, \ldots, p_N$ denote this set of positions in ascending order, with $p_0 = 0$ and $p_N = 1$. More specifically, we only have a finite set of samples of the frequency response in the 60Hz to $\frac{192000}{2} = 96000$Hz range, with 60Hz being the lowest frequency component that electric guitars typically produce. Let $f_0, \ldots, f_M$ denote this set of frequencies in ascending order. Let $\overset{\circ}{H}_k \in \mathbb{C}^{N+1}$ denote the samples of the target frequency response at the k-th tone knob position, where $k \in \{0, \ldots, N\}$. For all positions of the tone knob, the frequency response of the tone stage closely resembles that of a low-pass filter with some additional resonance (Figure 2.1). Therefore we will try to find a second-order IIR system of the form

$$y[n] = -a_{k,1}y[n-1] - a_{k,2}y[n-2] + b_{k,0}x[n] + b_{k,1}x[n-1] + b_{k,2}x[n-2], \tag{5.30}$$

whose frequency response $\overset{\circ}{H}'_k$ approximates $\overset{\circ}{H}_k$, and the unkown coefficients to be determined are real numbers. This may be formulated as an optimization problem, where the cost function to be minimized is the error of the IIR system's frequency response at frequencies $f_0, \ldots, f_M$, i.e. $\|\overset{\circ}{H} - \overset{\circ}{H}'\|_2^2$. This problem can be solved e.g. via gradient descent methods. For my experimentation, I used the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) algorithm from the SciPy Python library.

However, to make sure the resulting IIR systems are stable, we need to extend the cost function to include an error term that penalizes poles outside and on the complex unit circle. To keep the poles safely within the unit circle, I found the penalty function

$$(1.01 \cdot (|z| + 0.01))^{64} \tag{5.31}$$

to be very effective, where $z \in \mathbb{C}$ denotes a pole of the system. We can use the same formula to keep the zeros of the system within the unit circle, thus resulting in a minimum-phase IIR system, if desired. This optimization problem may be further refined by assigning different error weights to the $f_0, \ldots, f_M$ frequencies. In my experiment, I assigned a weight of 1 to audible frequencies in the range of 60Hz to 20kHz, and a weight of 0.2 to frequencies above 20kHz, thus resulting in a slightly better fit in the audible frequency range.

This approach gave me good results throughout the entire range of the tone knob parameter. Thus we have a way of precalculating optimal IIR coefficients for the set of measurements corresponding to the tone knob positions $p_0, \ldots, p_N$. However, to be able to incorporate a continuously variable parameter into the final emulation, we need a way to 'interpolate between IIR filters'. At first I tried linear interpolation between the neighboring filter coefficients, which turned out to be subpar. The interpolated filters were not always stable or sounded badly. I needed a way to guarantee the stability of the interpolated filters. A much better approach is to interpolate between the poles and zeros of the system. It is easy to see that, given a 1-1 mapping between the poles and zeros of two IIR systems, and given that the poles of both systems lie within the unit circle, the interpolated filters will also be stable. Furthermore, we can see from (3.31) that varying the poles and zeros of an IIR system continuously results in a continuous 'morphing' of the frequency response, that is, the frequency response of the system will converge pointwise. I devised a new optimization problem, that includes all $5(N+1)$ coefficients, and besides the individual cost terms, also incorporated a new penalty term based on the distances of poles and zeros of neighboring IIR filters. The end result was a set of IIR filters corresponding to the tone knob positions $p_0, \ldots, p_N$ that I was able to interpolate between, and sounded good throughout the entire parameter range, with no signs of instability. The final set of IIR filters can be found in Appendix D.

5.5 Evaluating the final emulation

We can conduct various tests on the final software emulation to evaluate its performance. We can compare the output signal to that of other software emulations or a SPICE simulated output signal. I had access to two other software emulations, the Line 6 Helix Native [30] and the STL Tonality: Andy James [31] VST plugins. For a fair comparison, I set both the drive and tone knobs on all platforms to neutral, that is, to a 'noon' knob position. All output signals were phase-aligned and normalized to -18 LUFS (integrated). I found a significant discrepancy between the input level handling of the mentioned software, therefore I adjusted the level of the input signals individually to achieve a similar level of distortion from all software emulations.

The simplest method is to listen to the output signal and see if we can identify sonic differences compared to the reference signals. My personal experience was that our emulation sounds reasonably good, and captures the overall character of the real circuit. The amount of distortion and the frequency content of the output signal is not exactly spot-on, but this is to be expected. These aspects can and should be adjusted manually to fine-tune the final emulation and compensate for numerical or other inaccuracies. I am certain that the mentioned reference emulations are designed and engineered much more thorougly, therefore they sounded more faithful to the SPICE simulation. I did not identify any clear issues with our solution—e.g. popping or clicking artifacts, aliasing, instability—with sensible input signals, the sound was clear and consistent with any drive and tone control values, even when adjusting them continuously in real-time.

A slightly more scientific comparison is the so-called *null test*, where the two signals to be compared are phase- and level-matched and then played simultaneously, but with one signal phase-inverted. The frequency content and overall loudness of the resulting signal can be used to measure the similarity of the compared signals. The resulting 'difference signals' with respect to our own emulation had the following loudness values: SPICE simulation −37.6 LUFS (momentary), Helix Native −32.5 LUFS (momentary), and STL Tonality −43 LUFS (momentary). This result confirms my subjective experience, as I found our model to sound most similar to the STL Tonality plugin, both of which sounded slightly dissimilar to the SPICE simulation and Helix Native.

By inspecting the phase-aligned and normalized output waveforms visually, we can arrive at the same conclusions. We can also clearly see the effects of soft clipping, that is, the peaks of the output waveforms are rounded off in a smooth manner.

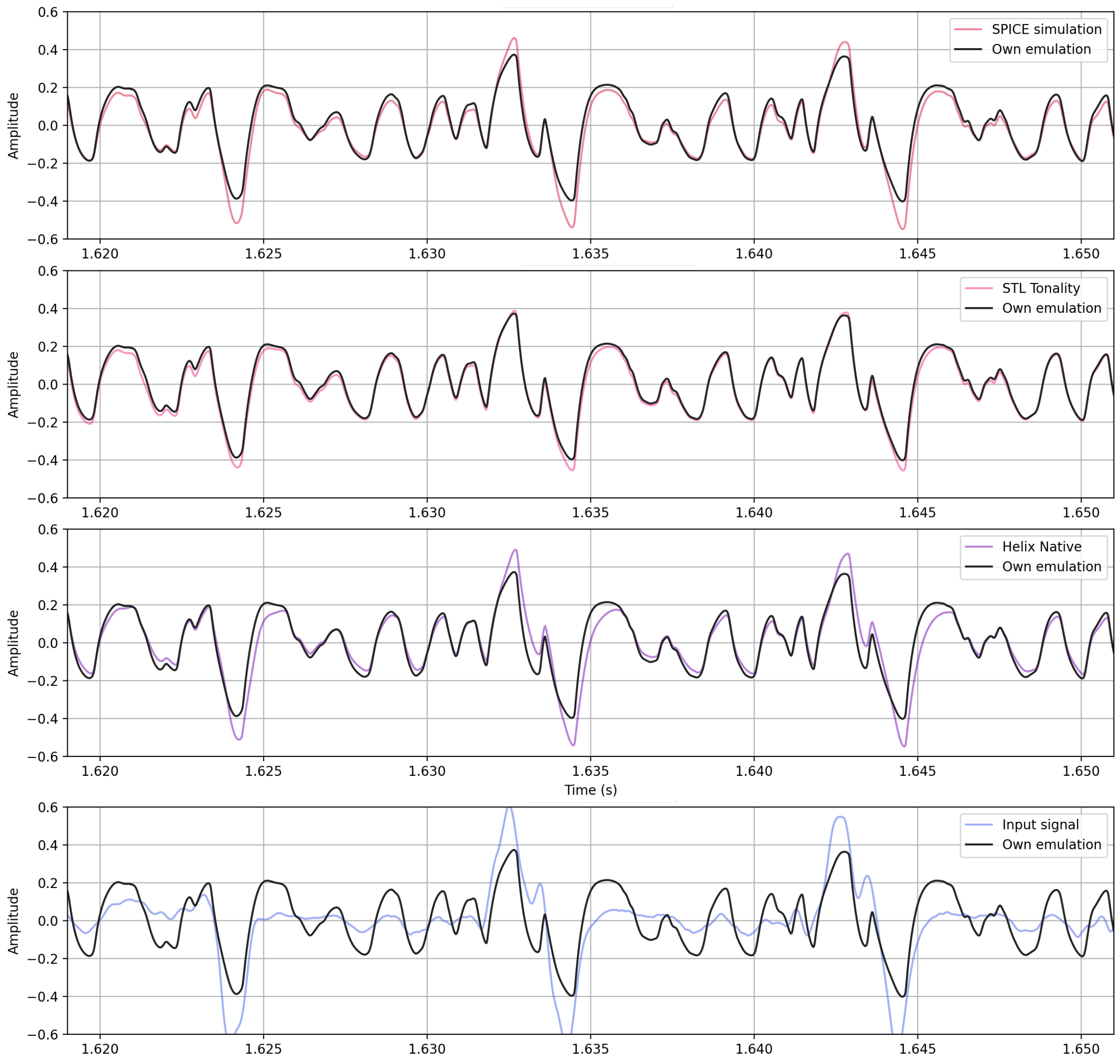
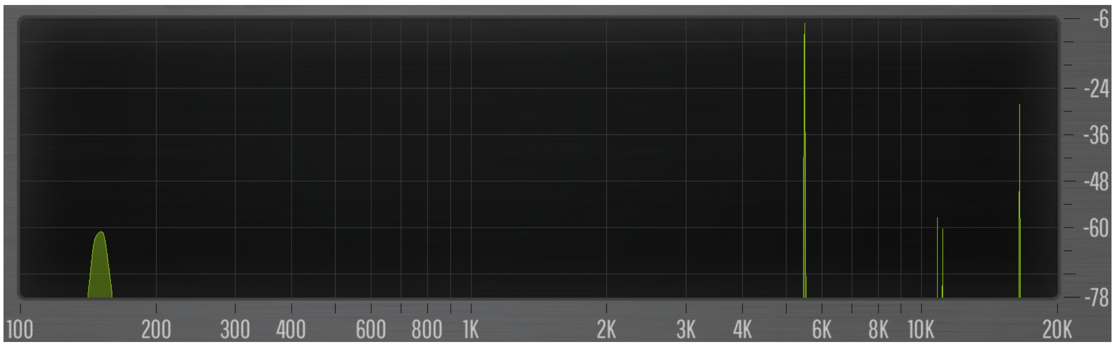
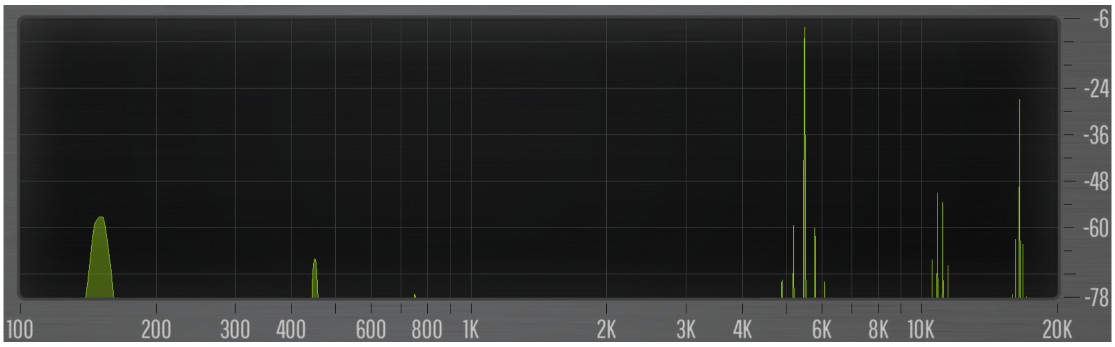


Figure 5.7. Comparison of output waveforms near a transient event (plucked guitar string) at ∼1.624s. The drive and tone controls were set to neutral (0.5) for all software.

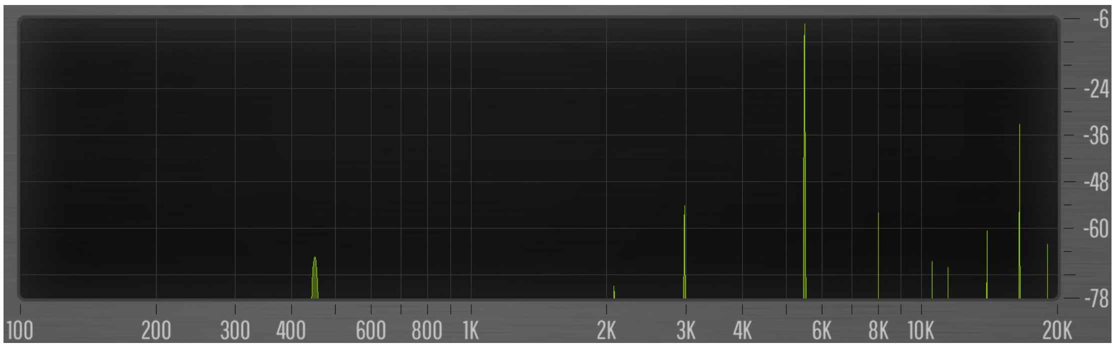
To test aliasing suppression, a pure sinewave signal with a frequency of 5490Hz was processed using all software emulations. I did not encounter obvious signs of aliasing below about 5000Hz with any of the tested software. The output signal was then analyzed with a FFT spectrum analyzer, the result of which can be seen in Figure 5.8. Our own emulation showed comparable aliasing suppression to the reference emulations.



(a) Own emulation



(b) STL Tonality



(c) Helix Native

Figure 5.8. Comparison of output magnitude spectra to test aliasing suppression.

To reduce the size of the diode LUT, I used a primitive algorithm that greedily removed data points from the 10001-long LUT in such a way that the maximum relative error of the resulting LUT—when compared to the full table—was no more than 0.5%. Furthermore, since $I_{D\downarrow\uparrow}$ is odd, it is sufficient to store only the positive half of the table. This resulted in a 96-long LUT, thus a single lookup operation requires at most $\lceil \log_2 96 \rceil = 7$ floating-point comparisons. The introduced latency and amount of required calculations to process a block of 32 input samples are summarized in the following table.

	Required calculations	Required lookahead samples	Latency (ms)
Oversampling + derivative estimation	2240 multiplications 1952 additions	4 (48kHz)	0.083
Clipping circuit emulation	128 multiplications 129 divisions 513 additions 1024 comparisons 2049 std::fma invocations 128 std::lerp invocations	0	0
Tone circuit emulation	640 multiplications 512 additions	0	0
Decimation	3360 multiplications 3328 additions	52 (192kHz)	0.271

Table 5.1. Computational requirements of the final emulation, implemented in C++.

The total latency of the emulation is precisely 17 samples at 48kHz, which I could manually verify in REAPER. For comparison, Helix Native has a self-reported latency of 16 samples at 48kHz.

I used the built-in performance meter of REAPER to compare the CPU utilization of the three emulations. The single-core utilization for Helix Native fluctuated around 4.2%, STL Tonality around 3.8%, and our own emulation around 3%.

Based on the presented results, we can conclude that the final software is adequate for real-time use, and should be able to run on less powerful devices, such as portable audio DSP chips.

5.6 Final thoughts

We successfully created a real-time emulation of a guitar distortion pedal using various branches of mathematics. Yet there are numerous topics we didn't have time to explore, but can be found within the referenced works. For those interested, I highly recommend [2,13,14,24] to guide them deeper into the world of digital signal processing.

I really enjoyed working on this project, and am very satisfied with the results. That said, the final emulation isn't perfect. There is plenty of room for improvement. The diode clipper emulation could be enhanced by using higher order numerical methods as discretization tools. The diode LUT could be optimized further based on the calculated upper bound for the range of output voltages, and so on. The current formulation works as-is, and makes for a good starting point, but wasn't designed to be optimal e.g. in terms of operation count or accuracy. Trying to improve upon the presented solutions to adhere to new, additional requirements would make for a good exercise, if not more.

References

[1] Accompanying GitHub repository for this thesis. [Online]. Available: https://github.com/Szapi/va-modeling

[2] J. W. Nilsson and S. A. Riedel, *Electric Circuits*, 10th ed. Pearson Education, 2015.

[3] J. D. Jackson, *Classical Electrodynamics*, 2nd ed. John Wiley & Sons, Inc., 1975.

[4] R. G. Keen, "The Technology of the Tube Screamer," 1998, last accessed 17 March 2025. [Online]. Available: http://www.geofex.com/article_folders/tstech/tsxtech.htm

[5] "ElectroSmash - Tube Screamer Analysis," last accessed 9 May 2025. [Online]. Available: https://www.electrosmash.com/tube-screamer-analysis

[6] D. E. Richards, *Basic Engineering Science - A Systems, Accounting and Modeling Approach.* Rose-Hulman Institute of Technology, 2001.

[7] P. Chakravorty, "A Modified General Diode Equation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[8] S. D'Angelo, L. Gabrielli, and L. Turchet, "Fast Approximation of the Lambert W Function for Virtual Analog Modelling," *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19)*, 2019.

[9] P. Darlington, "Analysis of the Tube Screamer," 2012, last accessed 14 April 2025. [Online]. Available: https://m0xpd.blogspot.com/2012/11/analysis-of-tube-screamer.html

[10] Voxengo SPAN real-time audio frequency spectrum analyzer. Last accessed 18 May 2025. [Online]. Available: https://www.voxengo.com/product/span/

[11] B. Boashash, *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*, 2003.

[12] P. Simon, "Fourier transform - University textbook," 2019, last accessed 27 April 2025. [Online]. Available: https://www.inf.elte.hu/dstore/document/300/Simon-Peter-Fourier-transzformacio.pdf

[13] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Pearson Education, 2010.

[14] T. W. Parks and C. S. Burrus, *Digital Filter Design.* John Wiley & Sons, Inc., 1987.

[15] M. Nagy, "Complex Function Theory (introduction) - Draft," p. 80, 2021, last accessed 27 April 2025. [Online]. Available: https://nmarci.web.elte.hu/dirs/jegyzetek/Complex.pdf

[16] REAPER Digital Audio Workstation. Last accessed 30 April 2025. [Online]. Available: https://www.reaper.fm/

[17] O. Niemitalo, "Polynomial Interpolators for High-Quality Resampling of Oversampled Audio," last accessed 30 April 2025. [Online]. Available: https://yehar.com/blog/wp-content/uploads/2009/08/deip.pdf

[18] M. Lewandowski, "Estimating the first and second derivatives of discrete audio data," *EURASIP Journal on Audio, Speech, and Music Processing*, 2024.

[19] A. Fettweis, "Wave Digital Filters: Theory and Practice," *Proceedings of the IEEE*, vol. 74, 1986.

[20] A. Sarti and G. D. Poli, "Toward Nonlinear Wave Digital Filters," *IEEE Transactions on Signal Processing*, vol. 47, 1999.

[21] K. J. Werner, J. O. S. III, and J. S. Abel, "Wave Digital Filter Adaptors for Arbitrary Topologies and Multiport Linear Elements," *Proceedings of the 18th International Conference on Digital Audio Effects (DAFx-15)*, 2015.

[22] S. Petrausch and R. Rabenstein, "Wave digital filters with multiple nonlinearities," *12th European Signal Processing Conference*, 2004.

[23] K. Werner, V. Nangia, A. Bernardini, J. Smith, and A. Sarti, "An Improved and Generalized Diode Clipper Model for Wave Digital Filters," 2015.

[24] K. J. Werner, "Virtual analog modeling of audio circuitry using wave digital filters," PhD thesis, Stanford University, Department of Music, 2016.

[25] 1N4148 diode SPICE model. Last accessed 2 May 2025. [Online]. Available: https://diotec.com/request/spice/1n4148.zip

[26] D. Horváth, Z. Červeňanská, and J. Kotianová, "Digital implementation of Butterworth first-order filter type IIR," *Research Papers Faculty of Materials Science and Technology Slovak University of Technology*, vol. 27, 2019.

[27] S. Qureshi, M. A. Akanbi, A. A. Shaikh, A. S. Wusu, O. M. Ogunlaran, W. Mahmoud, and M. Osman, “A new adaptive nonlinear numerical method for singular and stiff differential problems,” *Alexandria Engineering Journal*, vol. 74, pp. 585–597, 2023.

[28] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, “On the Lambert W Function,” *Advances in Computational Mathematics*, vol. 5, pp. 329–359, 01 1996.

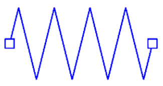
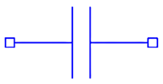
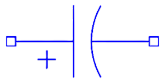
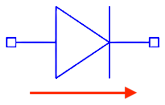
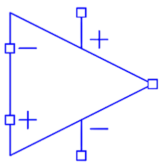
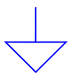
[29] TFilter filter design tool. Last accessed 7 May 2025. [Online]. Available: http://t-filter.engineerjs.com/

[30] Line 6 Helix Native Amp & Effects Plugin. Last accessed 23 May 2025. [Online]. Available: https://line6.com/helix/helixnative.html

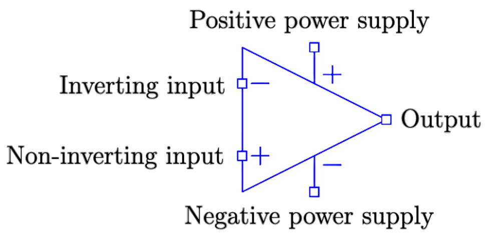
[31] STL Tonality: Andy James Guitar Plug-In Suite. Last accessed 23 May 2025. [Online]. Available: https://www.stltones.com/products/tonality-andy-james-guitar-plug-in-suite

Appendix A

Electrical components

Name	Symbol	Parameters	Branch Constitutive Equation
Resistor		$r > 0$ (resistance)	$i = \frac{v}{r}$
Capacitor		$c > 0$ (capacitance)	$i = c\dot{v}$
Polarized Capacitor		$c > 0$ (capacitance)	$i = c\dot{v}$
Diode		$I_S > 0$ (reverse saturation current) V_T (thermal voltage) $n \geq 1$ (ideality factor) $r \geq 0$ (internal resistance)	$i = I_S\left(e^{\frac{v-ir}{nV_T}} - 1\right)$
Operational Amplifier			
Ground Node			

Operational amplifier nomenclature



TS808 schematic

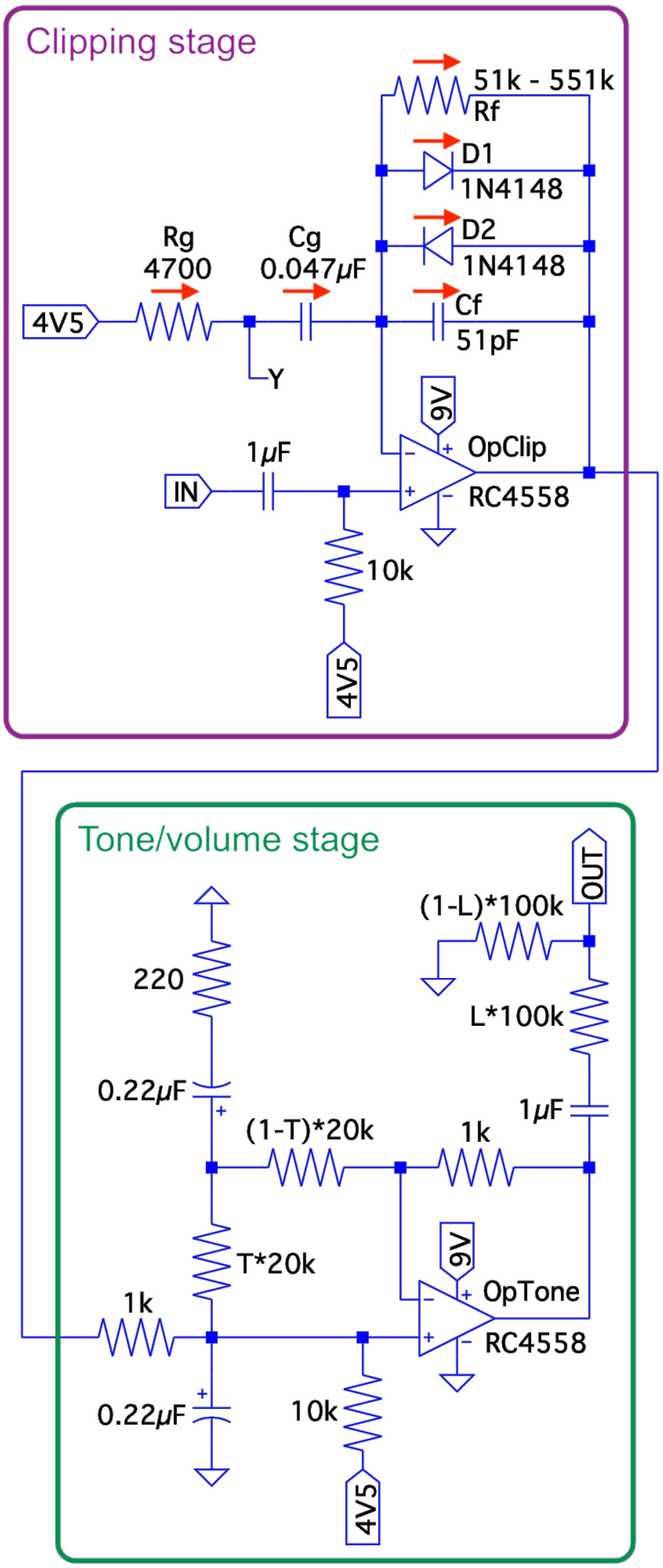
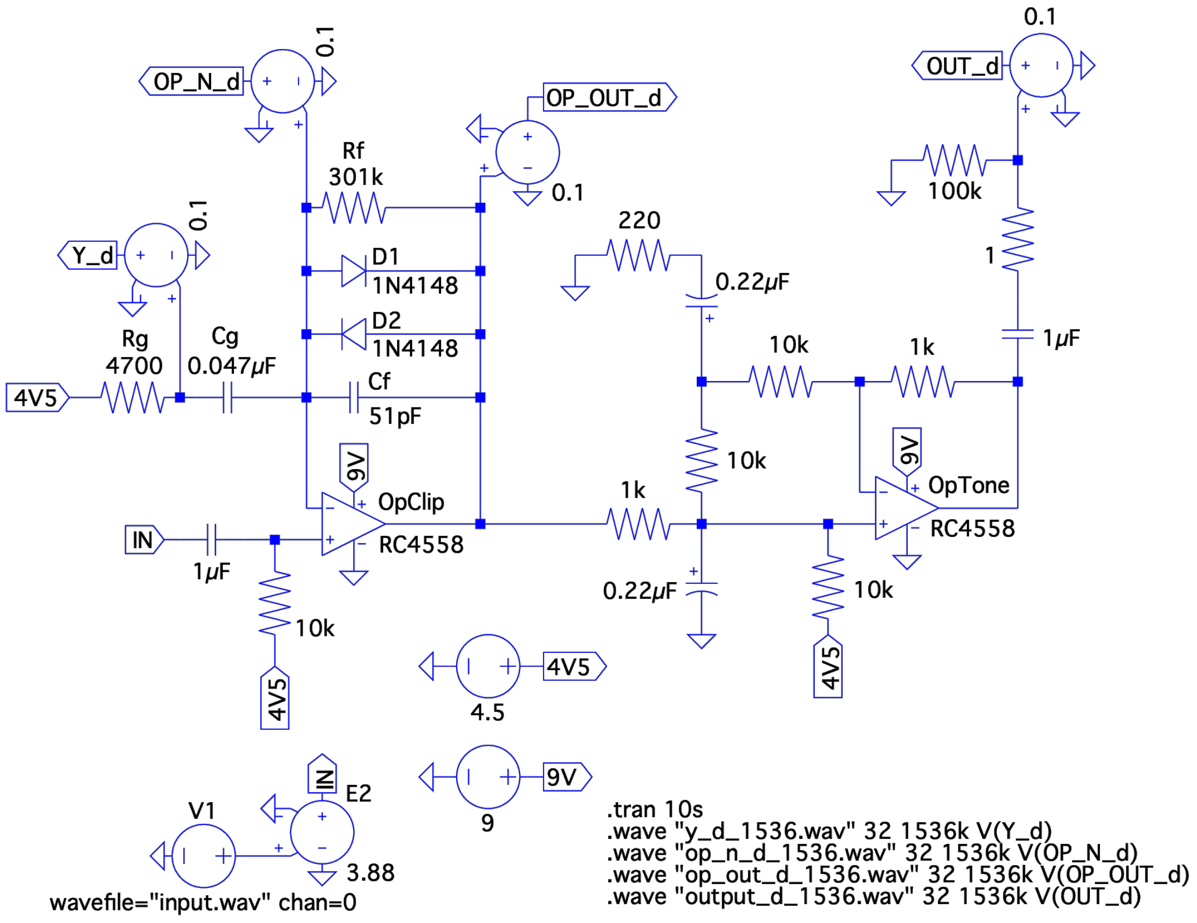


Figure 1. Schematics of the TS808 circuit, with indicated current measurements for the diode clipper network.

Appendix B

SPICE simulation of the TS808 circuit



```
.tran 10s
.wave "y_d_1536.wav" 32 1536k V(Y_d)
.wave "op_n_d_1536.wav" 32 1536k V(OP_N_d)
.wave "op_out_d_1536.wav" 32 1536k V(OP_OUT_d)
.wave "output_d_1536.wav" 32 1536k V(OUT_d)
```

Figure 2. SPICE simulation of the TS808 circuit with both the drive and tone controls at 0.5.

Test program output

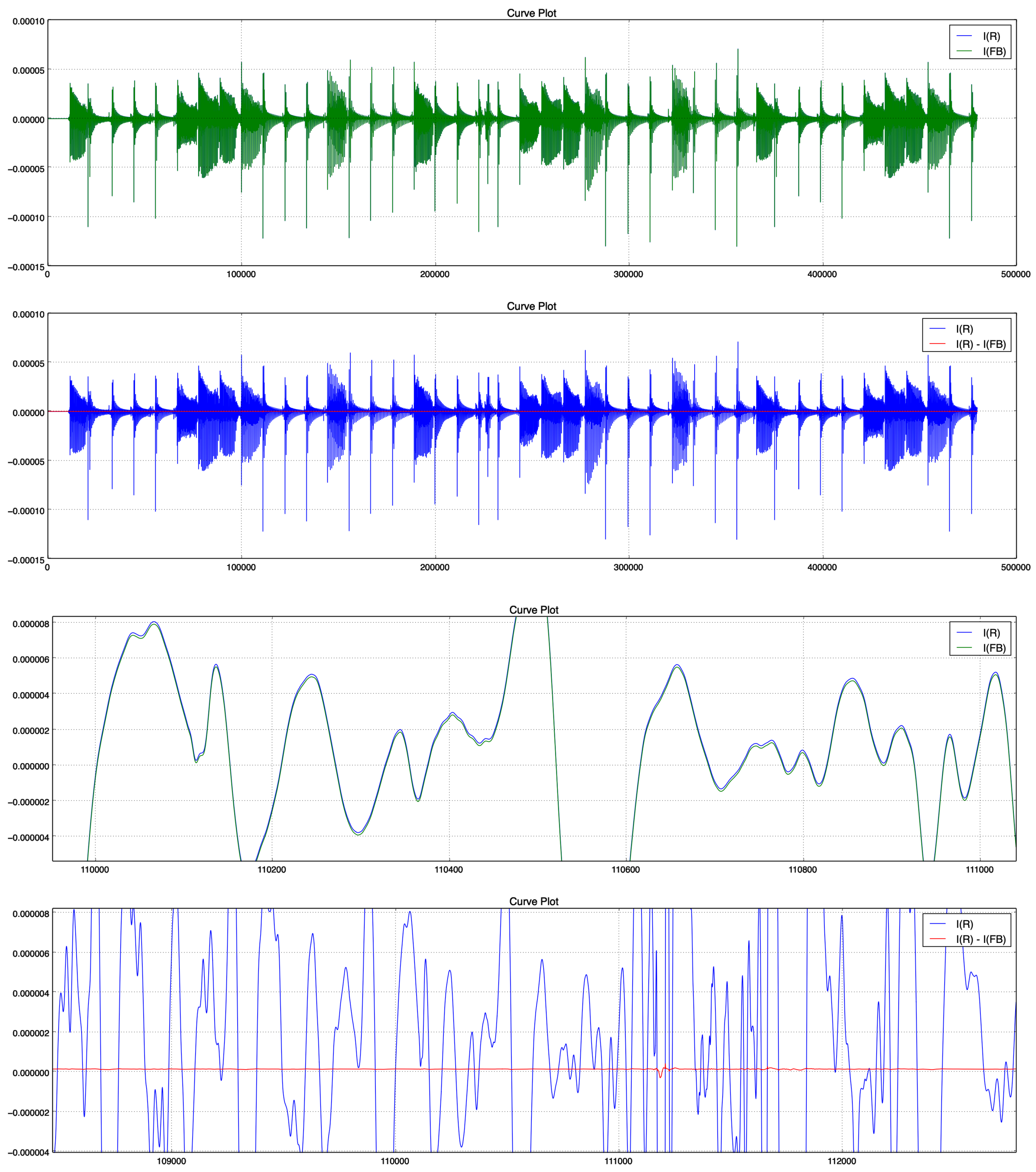


Figure 3. Output of a test program to verify equation (2.8). $I(R)$ denotes the current through resistor R_g. $I(FB)$ denotes the current through the feedback network (R_f, C_f, D_1, D_2).

Appendix C

Which diode clipper equation to discretize?

Let us discretize both formulations of the diode clipper equation and compare the results. Discretizing equation (2.7) yields

$$\Delta[n] = \Delta[n-1] + h\left(\frac{v_Y[n]}{r_g c_f} - \frac{\Delta[n]}{r_f c_f} - \frac{I_{D\downharpoonleft\upharpoonright}(\Delta[n])}{c_f}\right), \tag{1}$$

where

$$\Delta[k] \coloneqq v_{out}[k] - v_-[k], \tag{2}$$

whereas discretizing (2.8) yields

$$v_{out}[n] = v_{out}[n-1] + h\left(\dot{v}_{in}[n] + \frac{v_Y[n]}{r_g c_f} - \frac{\Delta[n]}{r_f c_f} - \frac{I_{D\downharpoonleft\upharpoonright}(\Delta[n])}{c_f}\right). \tag{3}$$

Let us subtract (1) from (3):

$$v_-[n] = v_-[n-1] + h\dot{v}_{in}[n], \tag{4}$$

which rearranges to

$$\frac{v_-[n] - v_-[n-1]}{h} = \dot{v}_{in}[n]. \tag{5}$$

This tells us that we can derive equation (1) from (3) by substituting the above approximation formula for $\dot{v}_{in}[n]$. We can get a much better approximation for the derivative of the input signal using the 7-point stencil formula, therefore equation (3) should result in a more accurate emulation.

Appendix D

Optimal tone circuit filters and the pole-zero-gain IIR parameterization

The optimal set of IIR filters for the tone/volume section turned out to have real-valued poles and zeros. This makes intuitive sense, as 2-2 real-valued poles and zeros allow for more flexible frequency response shaping than if they were complex conjugate pairs, as they can 'cover' more of the frequency spectrum. It would have been interesting to examine this phenomenon more closely, however, due to time constraints, I just accepted this result as the designed filters worked and sounded fine. These second-order IIR filters of the form (5.30) can be equivalently represented by 5 real numbers; the locations of the poles and zeros, and a 'gain' parameter. The following formula can be used to calculate the coefficients of such filters efficiently:

$$\begin{aligned} b_0 &= g, & a_1 &= -(p_1 + p_2), \\ b_1 &= -g \cdot (z_1 + z_2), & a_2 &= p_1 \cdot p_2, \\ b_2 &= g \cdot z_1 \cdot z_2, \end{aligned}$$

where $z_{1,2}$ and $p_{1,2}$ denote the zeros and poles respectively, and g denotes the gain parameter value.

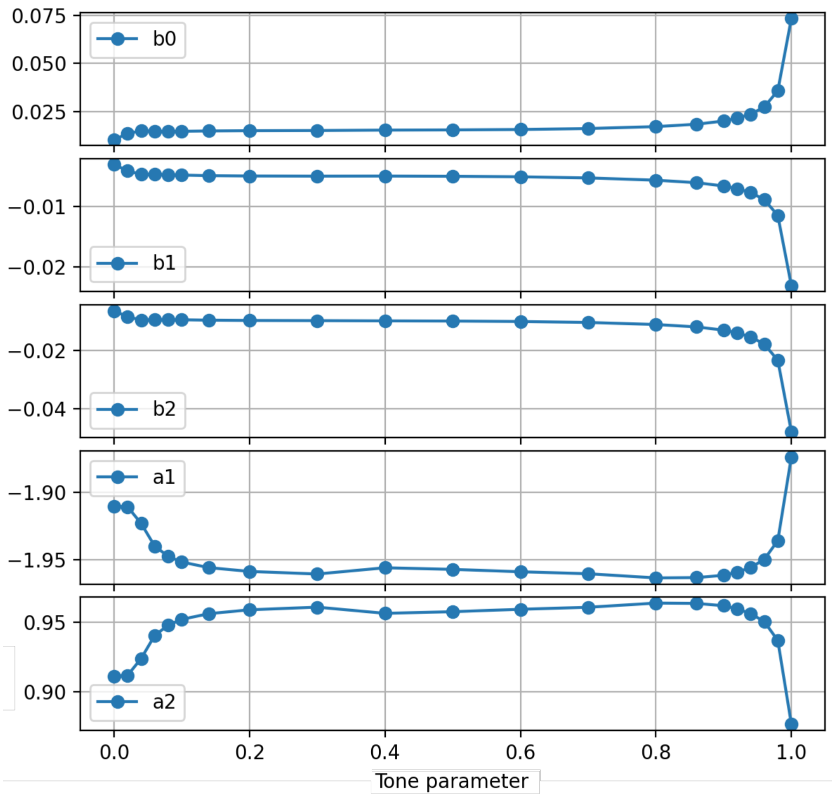


Figure 4. Optimal tone circuit IIR filter coefficients.

Tone param.	Pole 1	Pole 2	Zero 1	Zero 2	Gain
0.00	0.988573733	0.921770981	0.951611485	-0.665514294	0.010646047
0.02	0.989119823	0.921769510	0.965697699	-0.665495445	0.014008906
0.04	0.989765732	0.933399619	0.974874521	-0.665407386	0.015159218
0.06	0.990861110	0.949007324	0.982561684	-0.665383379	0.014826614
0.08	0.991688219	0.955808458	0.986273820	-0.665406549	0.014780225
0.1	0.992250912	0.959391745	0.988284028	-0.665421679	0.014831721
0.14	0.992628706	0.963154015	0.990052848	-0.665495656	0.014945623
0.2	0.992621601	0.965949156	0.990918706	-0.665387912	0.015088762
0.3	0.991903486	0.968582514	0.990890535	-0.665368746	0.015201411
0.4	0.986395586	0.969489563	0.985260394	-0.665473894	0.015432830
0.5	0.985550111	0.971522586	0.985362676	-0.665491359	0.015540709
0.6	0.985543241	0.973280182	0.986373992	-0.665484903	0.015809152
0.7	0.985634468	0.974657299	0.987416491	-0.665494284	0.016267243
0.8	0.988710323	0.974654237	0.990720330	-0.665514799	0.017239386
0.86	0.988603602	0.974529597	0.991276136	-0.665526396	0.018457541
0.9	0.987795197	0.973609324	0.991227246	-0.665517694	0.020203693
0.92	0.985548707	0.973621711	0.990256791	-0.665487693	0.021634968
0.94	0.982172595	0.973496962	0.989027403	-0.665325959	0.023827269
0.96	0.976187375	0.973494590	0.987349712	-0.665306245	0.027655391
0.98	0.973286563	0.962515944	0.984689069	-0.665302451	0.036337655
1.0	0.973297472	0.900786387	0.980303797	-0.665322887	0.074949153

Table 2. Optimal tone circuit IIR filters.

Alulírott Ték Róbert Máté nyilatkozom, hogy szakdolgozatom elkészítése során az alább felsorolt feladatok elvégzésére a megadott MI alapú eszközöket alkalmaztam:

Feladat	Felhasznált eszköz	Felhasználás helye	Megjegyzés
Szövegvázlat készítés	OpenAI GPT-4o	2. fejezet (bevezetés) 2.4 paragrafus 3.2 fejezet (bevezetés) 3.2.2 paragrafus (bevezetés) 4.2 paragrafus 4.3 paragrafus 4.5 paragrafus 5.4 paragrafus	Többmondatos vázlat készítése inspirációs céllal, egy-egy mondat/mondatrész/kifejezés szebb megfogalmazására javaslat.
Segítség új szakterület megismeréséhez	OpenAI GPT-4o	3. fejezet	Pl. legfontosabb szakkifejezések, definíciók, tételek, módszerek megismerése, amik alapján a nagyobb terjedelmű forrásokat (pl. [1], [2], [11], [12], [13]) már célzott módon tudtam feldolgozni.
Python kód generálás	OpenAI GPT-4o, GitHub Copilot	2. fejezet 5. fejezet	Ábrák készítéséhez program. Gyors prototipizálás. Numerikus és szimbolikus matematikai könyvtárak használatához segítség, pl. egyedi gradient descent IIR design program, 5.4 paragrafus.

A felsoroltakon túl más MI alapú eszközt nem használtam.

[illegible] 2025.05.25.