



IMDEA 1st Year Project

---

# Real-time Circuit Emulation in an Audio Plugin

---

Eliot DESCHANG

Supervised by  
Mr. Manuel MELON

International Master's Degree in Electroacoustics  
University year : 2023 - 2024

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Context</b>	<b>2</b>
2.1	History of circuit simulation and existing solutions . . . . .	2
2.2	Project objectives . . . . .	2
<b>3</b>	<b>Fundamental theory</b>	<b>3</b>
3.1	Branch constitutive equations of basic linear components . . . . .	3
3.1.1	Passive elements . . . . .	3
3.1.2	Active elements . . . . .	5
<b>4</b>	<b>Linear AC Analysis</b>	<b>5</b>
4.1	Modified Nodal Analysis . . . . .	5
4.1.1	Stamping Method . . . . .	7
4.2	Case of Study - Passive Bridged T Notch filter . . . . .	8
<b>5</b>	<b>Transient Analysis</b>	<b>10</b>
5.1	Integration Methods . . . . .	10
5.1.1	Backward Euler Method . . . . .	10
5.1.2	Trapezoidal Method . . . . .	10
5.2	Reactive Components . . . . .	10
5.2.1	Capacitor . . . . .	11
5.2.2	Inductor . . . . .	12
5.2.3	Companion model used in the plugin . . . . .	12
5.3	Parametric component . . . . .	13
5.3.1	Variable resistor . . . . .	13
5.3.2	Potentiometer . . . . .	13
5.4	Nonlinear Circuits . . . . .	13
5.4.1	Newton-Raphson method . . . . .	14
5.4.2	Diode . . . . .	15
5.4.3	Application of the 1 dimensional Newton-Raphson method . . . . .	15
5.4.4	Diode stamping method - Companion model . . . . .	16
5.5	Summary . . . . .	17
<b>6</b>	<b>Solution of linear systems of equations</b>	<b>18</b>
6.1	Matrix inversion . . . . .	18
6.2	LU Decomposition . . . . .	18
6.3	Matrix library in the frame of the plugin . . . . .	19
<b>7</b>	<b>Simplified flow chart of plugin's transient analysis algorithm</b>	<b>20</b>
<b>8</b>	<b>Final product - Plugin Interface</b>	<b>21</b>
<b>9</b>	<b>Case of study</b>	<b>21</b>
9.1	The baxandall tone . . . . .	22
9.2	Tube Screamer's soft clipping stage . . . . .	23
9.3	Summary . . . . .	25
<b>10</b>	<b>Conclusion</b>	<b>26</b>
<b>11</b>	<b>Future work</b>	<b>26</b>

<b>A</b>	<b>Ideal Linear Components</b>	<b>27</b>
A.1	Ideal Operational Amplifier . . . . .	27
A.2	Transformer . . . . .	27
A.3	Gyrator . . . . .	28
<b>B</b>	<b>Nonlinear components</b>	<b>29</b>
B.1	Diode . . . . .	29
	B.1.1 2-dimensional Newton-Raphson method example . . . . .	29
B.2	Bipolar Junction Transistor (BJT) . . . . .	31
<b>C</b>	<b>Potentiometers</b>	<b>33</b>
C.1	Baxandall Tone Control . . . . .	33
C.2	Tube Screamer soft clipping stage . . . . .	33
<b>D</b>	<b>Newton-Raphson Method</b>	<b>34</b>
D.1	Derivation . . . . .	34
D.2	Numerical Convergence . . . . .	35
	<b>Bibliography</b>	<b>37</b>

## Acknowledgement

I would like to express my deep gratitude to all the professors and administrative staff who, directly or indirectly, contribute to the maintenance and support of the IMDEA master's program.

For this project, I am particularly thankful to Manuel Melon for his availability and feedback on my work. I also wish to extend my thanks to the faculty members and other teachers who were very responsive via email in answering my questions. In particular, I am grateful to Bertrand Lihoreau, research professor, and Antonin Novak, researcher at the acoustics laboratory of the University of Le Mans, as well as Jérôme Tissier and Samuel Poiraud, teachers at ESEO in Angers.

# 1 Introduction

Musicians have always cherished vintage equipment: from violinists dreaming of playing on a Stradivarius to guitarists dreaming of owning an old distortion pedal with germanium diodes from USSR, the phantasms are all crazier than the last. Inside the realm of electronics, with the advent of digital technology and the rise of the internet, the analog sounds and effects of the last century, used by sound engineers and musicians who migrated to a computer, were introduced to thousands of people. From musty synths to obscure multi-band compressors used on iconic 70s rock albums, it wasn't long before everyone was eager to try out all this vintage equipment, with the aim of forging an original sound with character and imperfection.

To meet the growing demand for distinctive tones, companies, software engineers, and researchers have developed sophisticated techniques to replicate the sound of these classic machines within software programs, known as plugin, which can be integrated into Digital Audio Workstations (DAWs), allowing musicians to record, modify, or play music with this microsoftware. Nowadays, these effect plugins can be divided into two categories: plugins that reproduce audio effects from analogue electronic circuits (containing resistors, capacitors, transformers, etc.), and non-electronic-based plugins, ranging from a plugin modeling the reverberation of an existing room to digital-only effects, such as Autotune [1], which allows you to sing with a correct pitch.

Actually there is a main problem with this first type of plugin : there are already many emulations of the same gear on the market: each plug-in manufacturer offers its own version, which has led to market saturation and unnecessary complexity for users. On the other hand, the increasing computational power of modern computers now allows for more efficient performance of tasks that require higher computational costs. From this premise, it is possible to develop modular tools capable of emulating various kinds of electronic circuits, in real time.

This approach is particularly beneficial for non-scientist musicians who want to experiment with different kinds of audio equipment but lack knowledge in electronics, and don't want to invest in a lot of different plugins. Such a solution would enable them to load a netlist, a piece of text symbolizing the circuit components' connections, and experiment by tweaking the component values and knobs, as for traditional plugins. This simplified and intuitive interface would enable musicians to explore and create unique sounds without necessarily understanding the underlying electronics. For those involved in electronics design, it would enable them to modify existing equipment or create new ones.

This project aims to address this problem by developing an audio plugin that utilizes the Modified Nodal Analysis (MNA) [2], for real-time circuit emulation. After explaining with more details the context behind this project and reviewing the fundamental theory about circuit analysis, the MNA algorithm will be presented for linear Alternative Current (AC) analysis to understand the basic mechanics of this algorithm. Then, it will be adapted for transient analysis, demonstrating its capability to solve problems for both linear and non-linear circuits. Two ways for solving linear systems of equations will be evaluated, and one will be chosen for the final implementation in the plugin. At the end, the capabilities and uses of such a tool will be discussed, both in terms of the size of the circuits the plugin can handle, and the virtues it can bring to product design/modification.

## 2 Context

### 2.1 History of circuit simulation and existing solutions

Circuit simulation has been around since the early 60s with the advent of third-generation computers, integrating transistors. This era saw the creation of the CANCER program [3] by Professor Ronald Rohrer of U.C. Berkeley to simulate analog circuits, followed by SPICE in 1972 [4], based on the Modified Nodal Analysis (MNA)[2] which is still the foundation for most modern simulation tools like LTspice, Ngspice, or Tina. However, these programs aren't designed for real-time use. However, two recent solutions, LiveSpice and RTspice, achieve real-time emulation using different technologies to enable real-time emulation (Computer Algebra System and Just-In-time compiler for LiveSpice, GPU calculations for RTspice). This project aimed to explore implementing a similar real-time circuit emulation solution on the CPU.

### 2.2 Project objectives

The main objective of this project is to create a Plugin effect that could be loaded in any digital audio workstation (Ableton, Fl studio...), in order to emulate the behavior of *any* electronic circuit, of relatively modest size in real time: about maximum 40 nodes for Linear Time Invariant (LTI) and Linear Time Variant (LTV) systems, and about 15 nodes for nonlinear circuits (containing diodes, transistors...). The specifications are set out below.

#### Specifications:

- A text editing area for entering circuit information, commonly referred to as a netlist, on which each line represents a component. The first token defines the component type (e.g., voltage source, resistor), the following tokens specify the nodes connected to the component, and the last tokens indicate component values (e.g., resistance, capacitance, etc.). The user must be able to modify the netlist in real-time, updating the signal processing accordingly. An example netlist representing a circuit from Fig. 1 is shown in Tab. 1.
- An area with buttons that can be associated with parameterizable components (potentiometers, variable resistors), referenced in the netlist.
- A zone for controlling input and output gain, and a mix knob enabling the user to mix the processed signal (wet) with the original signal (dry).
- An area for controlling emulation fidelity, enabling the number of iterations to be regulated, and including oversampling functionality (more will be said about that).
- A netlist library with a number of pre-installed circuits, related to audio applications.

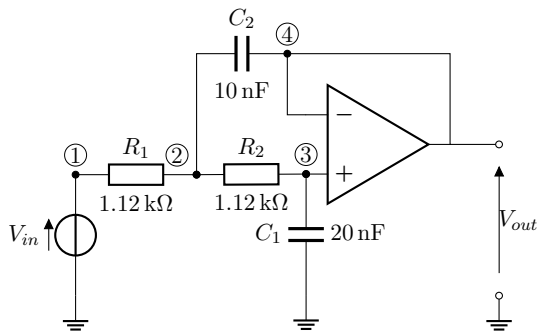


Figure 1: Circuit example - Sallen-Key cell.

Component	Netlist lines
Input voltage source	Vin 1 0
Resistor	R1 1 2 1120
Resistor	R2 2 3 1120
Capacitor	C1 3 0 10e-9
Capacitor	C2 2 4 20e-9
Operational Amplifier	O1 3 4 4
Voltage Probe	Vout 4 0

Table 1: Equivalent netlist representation of the example circuit on the left.

### Application case:

An example of the plugin's use could be a guitarist wishing to play with the effect of his favorite distortion pedal inside a digital audio workstation. If the circuit is already in the plugin's library, all he would need to do is load the netlist into the plugin, which will emulate the pedal in real time. If the circuit isn't in the library, he could look up the circuit diagram on the Internet and type in the netlist describing the system, as this requires only rudimentary knowledge (knowing component values and describing component nodal connections in a *right way*).

## 3 Fundamental theory

In network analysis, various types of network elements are encountered, making it essential to formulate equations for circuits that include as many different types of network elements as possible. There are several methods for equation formulation in circuit analysis, all based on three fundamental types of equations found in circuit theory:

- Equations based on Kirchhoff's Voltage Law (KVL): every circuit node has a unique voltage (with respect to the ground or *datum* node, which is 0 Volts by convention). The voltage drop (potential) across a branch,  $V_B$ , is equal to the difference between the (positive and negative referenced) voltages of the nodes on which it is incident.

$$V_B = v^+ - v^-. \quad (1)$$

The algebraic sum of voltage drops around each closed loop of the network must equal zero at every instant of time.

- Equations based on Kirchhoff's Current Law (KCL): the algebraic sum of all the currents flowing out of (or into) any circuit node is zero. Currents flowing toward a node are defined as positive.
- Branch constitutive equations: define the relationship between voltage and current for different types of circuit elements.

### 3.1 Branch constitutive equations of basic linear components

In this part, the constitutive equations of the main linear dipoles that can be found in a linear circuit are reviewed. Two types of elements can be distinguished: passive and active. Other components are reviewed in Appendix A and B, and for informed readers, this part can be skipped.

#### 3.1.1 Passive elements

##### Resistor

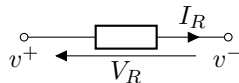


Figure 2: Symbol for a resistor.

The relation between the current and voltage for an ideal resistor (see Fig. 3.1.1) is defined as:

$$V_R(t) = R \cdot I_R(t) \quad (2)$$

where  $V_R$  and  $I_R$  represent respectively the voltage across the resistor and the current passing through the resistor over the time  $t$ .  $R$  is defined as being the resistance value of the resistor in Ohm

( $\Omega$ ). In the frequency domain, by writing  $V_R(t) = V_R e^{st}$  and  $I_R(t) = I_R e^{st}$ , where  $s = j\omega$  is the Laplace variable, with  $\omega$  representing the angular frequency in radian, the relation of equation eq. (2) remains unchanged. This resistance in the frequency domain is called impedance, therefore  $Z_R = R$ .

The inverse of an impedance  $Z = R + sX$ , with  $X$  the reactance, is called an admittance  $Y$ . If this admittance is real, as in the case of a resistor, it is referred to as conductance. In this report, impedances will be denoted by the symbol  $Z$ , admittance by the symbol  $Y$ , and conductance by the symbol  $G$ .

### Capacitor

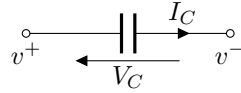


Figure 3: Symbol for a capacitor.

The relation between the current and voltage in terms of integral or as a differential equation for an ideal capacitor (see Fig. 3) is:

$$V_C(t) = \frac{1}{C} \int I_C(t) dt, \quad I_C(t) = C \frac{dV_C}{dt}, \quad (3)$$

where  $I_C(t)$  is the current passing through the capacitor over the time  $t$ ,  $V_C$  is the voltage across the capacitor, and  $C$  is the value of the capacitor in Farad (F). In the frequency domain, by writing  $V_C(t) = V_C e^{st}$  and  $I_C(t) = I_C e^{st}$ , and substituting these expressions into equation (3), the following equation is obtained:

$$V_C = \frac{1}{sC} I_C. \quad (4)$$

Therefore  $Z_C = \frac{1}{Y_C} = \frac{1}{sC}$  represents the impedance of the capacitor. . Here, the impedance depends on the frequency, classifying it as a reactive component.

### Inductor

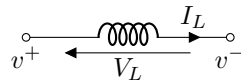


Figure 4: Symbol for an inductor.

The relation between the current and voltage in terms of integral or as a differential equation for an ideal inductor (see Fig. 4) is:

$$I_L(t) = \frac{1}{L} \int V_L(t) dt, \quad V_L(t) = L \frac{dI_L}{dt}, \quad (5)$$

where  $I_L(t)$  is the current passing through the inductor over the time  $t$ , and  $V_C$  is the voltage across the inductor, and  $F$  is the value of the inductance in Farad (F). In the frequency domain, by writing  $V_L(t) = V_L e^{st}$  and  $I_L(t) = I_L e^{st}$ , and substituting these expressions into equation (5), the following equation is obtained:

$$V_L = sL \cdot I_L. \quad (6)$$

Therefore  $Z_L = \frac{1}{Y_L} = sL$  represents the impedance of the inductor.



### 3.1.2 Active elements

#### Voltage source

The voltage of a voltage source is either a constant or a function of time,  $V_s = f(t)$ , that don't depend on the behavior of the other components in the circuit, in which case the element is called an independent voltage source and has the symbols shown in Fig. 3.1.2.



Figure 5: Symbols for a constant (a) and time-varying (b) independent voltage source.

#### Current source

The current of a current source is either a constant or a function of time,  $I_s = f(t)$ , that don't depend on the behavior of the other components in the circuit, in which case the element is called an independent current source and has the symbol shown in Fig. 3.1.2.

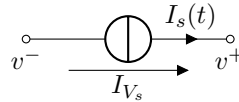


Figure 6: Symbol for a constant or time-varying independent current source.

## 4 Linear AC Analysis

In this section, the goal is to introduce one of the most efficient and common way to automatize the process of writing down the equations governing a circuit, for Alternative Current (AC) Analysis. For the sake of simplicity, only circuit containing linear components and independent voltage/currents sources will be carried out in this part. AC analysis consist in analyzing the AC output variables (currents or voltages) as functions of frequency over a user-specified range, to know for example the gain and/or phase relation between a given input and output (transfert function). In the context of this project, the input can be a voltage source, representing the input audio signal, and the output can be the voltage across a resistor of the circuit.

### 4.1 Modified Nodal Analysis

The equations representing the circuit must be formulated and represented in a computer program automatically, in a simple and comprehensive manner. Once formulated, the system of equations must be solved. Two critical aspects to consider when choosing algorithms for this purpose are accuracy and speed. Modified Nodal Analysis (MNA) originally described by Ho et al. [2] has been proven to effectively accomplish these tasks, and involves the following steps [5]:

1. Write KCL at each node except one, which will be called the datum node.
2. Use the element equations to eliminate as many current variables as possible from KCL, leading to equations in terms of mostly branch voltages (potentials).
3. Use KVL to replace all the branch voltages by nodal voltages ( $v_1, v_2 \dots$ ).
4. Add branch constitutive equations for all the voltage source terms.

## Example

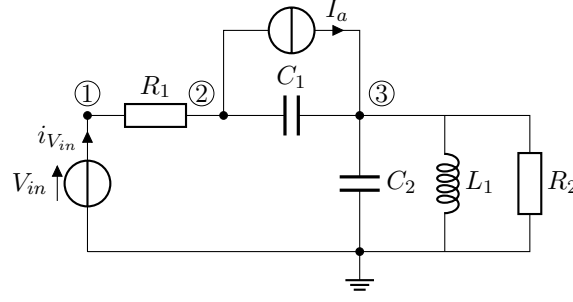


Figure 7: Circuit example.

Consider the following circuit shown in Fig. 7. Application of the rules set out in 4.1 leads to the following equations:

$$\frac{v_1 - v_2}{R_2} - I_{V_{in}} = 0 \quad (\text{KCL at node 1}) \quad (7)$$

$$\frac{v_2 - v_1}{R_1} + (v_2 - v_3) \cdot C_2 s + I_a = 0 \quad (\text{KCL at node 2}) \quad (8)$$

$$v_3 \cdot C_2 s + \frac{v_3}{sL_2} + \frac{v_3}{R_2} + (v_3 - v_2) \cdot C_1 s - I_a = 0 \quad (\text{KCL at node 3}) \quad (9)$$

$$v_1 - 0 = V_{in} \quad (\text{Branch const. eq.}) \quad (10)$$

Finally, this system can be written in a matrix equation of the form

$$\mathbf{Ax} = \mathbf{b}, \quad (11)$$

where  $\mathbf{A}$  is a square Matrix, and where  $\mathbf{x}$  and  $\mathbf{b}$  are 1D column vectors. By bringing all the know voltage and current source terms to the right-hand side (RHS) vector, and all the passive components values in the matrix, it can be stated that:

$$\left[ \begin{array}{ccc|c} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 1 \\ -\frac{1}{R_1} & C_1 s + \frac{1}{R_1} & -C_1 s & 0 \\ 0 & -C_1 s & C_1 s + C_2 s + \frac{1}{R_2} + \frac{1}{L_1 s} & 0 \\ \hline 1 & 0 & 0 & 0 \end{array} \right] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ I_{V_{in}} \end{bmatrix} = \begin{bmatrix} 0 \\ -I_a \\ I_a \\ V_{in} \end{bmatrix}. \quad (12)$$

By inverting the system matrix, the vector of node voltages and branch currents can be solved in terms of the vector containing the input voltage and input current:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}. \quad (13)$$

As it can be seen, the structure of such a matrix system, as for the example case above suggests that this process can be automated. Indeed, MNA sets up the equations governing a circuit in the form [4]:

$$\underbrace{\begin{bmatrix} \mathbf{Y} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{v} \\ \mathbf{j} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{i} \\ \mathbf{e} \end{bmatrix}}_{\mathbf{b}}. \quad (14)$$

Here,  $\mathbf{v}^N$ , with  $N$  representing the number of nodes excluding the datum node, is the vector containing the unknown node voltages. The vector  $\mathbf{i}^N$  represents known contributions from independent current sources, and  $\mathbf{e}^M$ , with  $M$  representing the number of unknown or extra currents,

is a vector containing known voltage sources contributions. The top partition represents KCL at each circuit node, and  $\mathbf{Y}^{N \times N}$  represents the admittances of electrical elements with immittance representations. The bottom partition defines other elements (e.g., voltage sources, controlled sources, nullors) that require defining extra currents referenced in  $\mathbf{j}^M$ . These extra currents influence KCL at each node through  $\mathbf{B}$ , and their relationships to node voltages and each other are defined through  $\mathbf{C}^{M \times N}$  and  $\mathbf{D}^{N \times M}$ .

A crucial aspect of Modified Nodal Analysis is the existence and implication of the datum node. Indeed, including every node in eq. (14), makes  $\mathbf{A}$  singular and noninvertible. This can be problematic if a *naive* approach (such as computing the inverse) is used to solve the system. This issue arises because the number of independent KCL equations in any circuit is always one less than the number of nodes [6]. Additionally, including the ground (or datum node) adds one extra row and column, resulting in a more complex system to solve. To clarify, the ground (node 0) is typically chosen as the datum node, but any other node can be selected.

Luckily the Modified Nodal Analysis framework provides a systematic and even automatic approach to fill the matrix system, called element stamps [2], which are also sometimes called “MNA by inspection” [7]. In the section below, the stamping method for all the linear elements discussed above will be reviewed for AC analysis.

#### 4.1.1 Stamping Method

The stamping method involves inspecting each component of the netlist and adding its stamp contribution to the final system to be analyzed. In the section below, the stamping method for the linear components discussed earlier is demonstrated. Other stamping method for non-linear components will be discussed in Transient Analysis section, as they can’t be directly used in the frequency domain, and other *assumed* ideal linear elements, such as operational amplifiers, transformers or gyrators, are discussed in A.

##### Passive elements

For both resistors, capacitors, and inductors, ones can writes the following branch constitutive equations using admittance notation:

$$I = Y \cdot V_B \quad (15)$$

where  $Y$  is the admittance of the component,  $V_B$  the voltage drop in the branch and  $I$  the current flowing between the terminals 1 and 2. The current contribution from the component at the terminals 1 or 2 (respectively  $i_1$  and  $i_2$ ) can be written from eq. (15) as:

$$\begin{cases} i_1 = Y(v_1 - v_2), \\ i_2 = -Y(v_1 - v_2), \end{cases} \quad (16)$$

consisting to the following stamp of the  $\mathbf{Y}$  matrix:

$$\mathbf{Y} = \begin{matrix} & \overbrace{\begin{matrix} \textcircled{1} & \textcircled{2} \end{matrix}}^v \\ \begin{matrix} \textcircled{1} \\ \textcircled{2} \end{matrix} & \begin{bmatrix} Y & -Y \\ -Y & Y \end{bmatrix} \end{matrix}. \quad (17)$$

##### Voltage source

Considering a voltage source of voltage  $E$ , the branch constitutive equation impose that:

$$v_1 - v_2 = E. \quad (18)$$

In addition, an unknown current will flow between the terminals 1 and 2. Thus, for the voltage source  $i_1 = I_E$  and  $i_2 = -I_E$ . This current equations are taken into account in the KCL as a new variable, and therefore impose to add a supplementary row and column to the initial matrix (conducting to the creation of the **B** and **C** matrix). The **A** matrix and RHS vector will have the following stamp:

$$\mathbf{A} = \begin{array}{c} \textcircled{1} \quad \textcircled{2} \quad \textcircled{A} \\ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & -1 & 0 \end{bmatrix} \end{array}, \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ E \end{bmatrix}. \quad (19)$$

### Current source

For the current source, the stamping method is one of the most simpler, as we just have to place the known current of source on each of the source's nodal connections index in the RHS vector. Therefore, for a current source for which the current  $I_s$  flows from node 1 to node 2, the stamp of **b** is :

$$\mathbf{b} = \begin{array}{c} \textcircled{1} \\ \textcircled{2} \end{array} \begin{bmatrix} -I_s \\ I_s \end{bmatrix} \quad (20)$$

## 4.2 Case of Study - Passive Bridged T Notch filter

To see a practical example of how such an algorithm might be used, let's take a look at a filter that a user of the plugin might wish to model: the bridged T notch filter. This filter can be found in the feedback loop of some operational amplifiers, or simply in series in a circuit, to remove unwanted spectral components from a narrow band (notch filter). The schematic of such a circuit is depicted in Fig. 8. To understand the input/output relationship, it is essential to compute the transfer function for the input audio, which is modeled as the voltage source  $V_{in}$ .

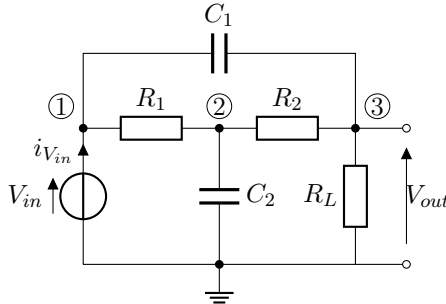


Figure 8: Passive Bridged-T Notch Filter.

Using the standard stamp procedures to write a MNA matrix equation leads to:

$$\begin{array}{c} \textcircled{1} \quad \textcircled{2} \quad \textcircled{3} \quad \textcircled{A} \\ \begin{bmatrix} C_1 s + \frac{1}{R_1} & -\frac{1}{R_1} & -C_1 s & 1 \\ -\frac{1}{R_1} & C_2 s + \frac{1}{R_2} + \frac{1}{R_1} & -\frac{1}{R_2} & 0 \\ -C_1 s & -\frac{1}{R_2} & C_1 s + \frac{1}{R_L} + \frac{1}{R_2} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ I_{V_{in}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ V_{in} \end{bmatrix} \end{array}. \quad (21)$$

After solving such a system, for example by computing the invert of **A**, all the unknown node voltages are obtained. Thus, the transfer function can be derived. Knowing that  $V_{out} = v_3$  and  $V_{in} = v_1$  the transfer function is given by:

$$H_c(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{(R_1 R_2 C_1 C_2) s^2 + (C_1(R_1 + R_2)) s + 1}{(R_1 R_2 C_1 C_2) s^2 + \left( R_1(C_1 + C_2) + R_2 \left( C_1 + \frac{C_2}{R_L} \right) \right) s + \frac{R_1 + R_2}{R_L} + 1}. \quad (22)$$

Here, the transfer function has been suitably factorized to include only polynomials in the numerator and denominator. The problem is that this transfer function is expressed in the continuous frequency domain, and therefore is not suitable to simulate the output signal from a discrete input one, as it will be the case for a musician recording itself on a soundcard. A common solution to digitize continuous-time transfer functions like this one is to use the bilinear transform, a mapping from the  $s$  to the  $z$  plane according to the substitution [8]

$$s \leftarrow \frac{2}{T} \frac{1 - z^{-1}}{1 + z^{-1}}, \quad (23)$$

and factored into biquadratic filters to avoid numerical sensitivity issues [9][10]. The challenge lies in efficiently finding biquadratic filters coefficients, particularly for orders higher than two, which can become complex to derive and even more difficult to adjust on the fly for time-variant filters where coefficients need real-time updates. Using symbolic solvers might reduce computation costs since the symbolic expressions of coefficients are computed once and evaluated as needed. However, solving the system symbolically, computing its transfer function in continuous time, simplifying the expression, substituting  $s$  with the bilinear transform, and then evaluating the poles and zeros in biquadratic form is impractical. This is why, even for a circuit with very few elements as in this study case, the question arises of whether or not to use traditional frequency-domain methods to model such a system.

In the next section, a simpler method for solving both linear time-varying and non-linear systems will be discussed. This algorithm, based on MNA for transient analysis (temporal domain), is the one implemented in the plugin.

## 5 Transient Analysis

In this section, the goal is to introduce the MNA algorithm for transient analysis. Transient analysis computes the output variables, voltages and currents, as functions of time, for either a constant or variable time step. Since this algorithm will be the final one used in the plugin, and as audio is always recorded at a fixed samplerate, It goes without saying that only constant-step numerical methods will be studied.

Most of the components studied have branch constitutive equations that directly link the current to the voltage without any derivative or integral. Therefore, the stamping method for these non-reactive components is the same for both transient and AC analysis. However, reactive components like capacitors and inductors are frequency-dependent and require different modeling for transient analysis compared to AC analysis. This is what will be studied below.

### 5.1 Integration Methods

To solve the differential equations of reactive components (see eq. (3), eq. (5)), several numerical methods such as the Backward Euler Method or the Trapezoidal Method can be used [11, 12]. These ODE solvers employ numerical integration to solve equations of the form:

$$\frac{dx}{dt} = \dot{x}(t) = f(t, x(t)), \quad x(t_0) = x_0,$$

where  $t_0$  is a time at which the value of  $x$  is known (initial condition). Since numerical methods are discrete-time methods, the system variables must be expressed in a discretized form, allowing:

$$\dot{x}^n = f(t^n, x^n), \quad x(t_0) = x_0,$$

with  $x^n = x(t^n)$ ,  $t^n = nT$ , where  $n \in \mathbb{N}$  is the discretization index and  $T = \frac{1}{F_s}$  is the sampling interval, with  $F_s$  the sampling frequency. These methods approximate the exact solution of the function  $x(t)$ . In the literature, the discrete approximation of  $x^n$  is denoted as  $\tilde{x}^n \approx x^n$ , using the tilde symbol to indicate that it is an approximation of the "true" value of  $x$  at sample  $n$ . In the following part, we will make the assumption that  $x^n = \tilde{x}^n$ , to avoid too heavy notation.

#### 5.1.1 Backward Euler Method

In numerical analysis, finite differences are widely used to approximate derivatives, and the term *finite difference* is often used as a shorthand for *finite difference approximation of derivatives*. The Backward Euler (BE) method (also called implicit) can be seen as a *backward* finite-difference approximation of the derivative, leading to the relation :

$$\dot{x}^{n+1} = \frac{x^{n+1} - x^n}{T}, \quad \text{or} \quad x^{n+1} = x^n + T\dot{x}^{n+1}. \quad (24)$$

#### 5.1.2 Trapezoidal Method

The expression for  $\dot{x}^{n+1}$  or  $x^{n+1}$  using the implicit Trapezoidal (TR) method is given by:

$$\dot{x}^{n+1} = \frac{2}{T} \frac{x^{n+1} - x^n}{1} - \dot{x}^n, \quad \text{or} \quad x^{n+1} = x^n + \frac{T}{2}(\dot{x}^n + \dot{x}^{n+1}). \quad (25)$$

## 5.2 Reactive Components

The goal of this part is to describes how the MNA can be used to consider the energy storing effects of reactive components into account, and to derive a way to stamp in the matrix the different reactive element in the MNA matrix system (i.e eq. (14)).

### 5.2.1 Capacitor

The constitutive equations of a capacitor:

$$I_C(t) = C \frac{dV_C}{dt} \quad (26)$$

can be approximated with one of the finite difference scheme seen above, such that at the end, eq. (26) can be rewritten regarding the next integration current:

$$I_C^{n+1} = \frac{C}{T} V_C^{n+1} - \frac{C}{T} V_C^n, \quad (\text{BE}) \quad (27)$$

$$I_C^{n+1} = \underbrace{\frac{2C}{T} V_C^{n+1}}_{G_{eq}} - \underbrace{\frac{2C}{T} V_C^n}_{I_{eq}} - I_C^n. \quad (\text{TR}) \quad (28)$$

This equations can be written as:

$$I_C^{n+1} = G_{eq} V_C^{n+1} + I_{eq}. \quad (29)$$

This equation can be in a sort of way schematize, leading to the following companion model representing a current source with its accompanying internal conductance, commonly referred to as a Norton generator.

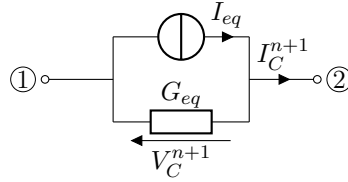


Figure 9: Companion model of the capacitor with a Norton generator, for transient analysis.

This way to represent the capacitor allows it to be stamped into the matrix using the same linear element methods as in AC analysis, which is really convenient. Thus, using the stamping method for a resistor and an independent current source to represent the system in Fig. 9 results in:

$$\begin{matrix} \textcircled{1} & \textcircled{2} \\ \textcircled{1} & \begin{bmatrix} G_{eq} & -G_{eq} \\ -G_{eq} & G_{eq} \end{bmatrix} \end{matrix} \begin{bmatrix} v_1^{n+1} \\ v_2^{n+1} \end{bmatrix} = \begin{bmatrix} -I_{eq} \\ I_{eq} \end{bmatrix}. \quad (30)$$

Alternatively, the eq. (26) can be approximated with the same difference scheme methods, but isolating the next voltage sample such that:

$$V_C^{n+1} = \frac{T}{C} I_C^{n+1} + V_C^n, \quad (\text{BE}) \quad (31)$$

$$V_C^{n+1} = \underbrace{\frac{T}{2C} I_C^{n+1}}_{R_{eq}} + \underbrace{\frac{T}{2C} I_C^n + V_C^n}_{V_{eq}}. \quad (\text{TR}) \quad (32)$$

This leads to another representation of the capacitor for transient analysis, with a resistor in series with a voltage source, commonly referred to as a Thevenin generator. This approach will be further studied in the next part with the inductor to avoid redundancy in the report.

### 5.2.2 Inductor

The constitutive equation of an inductor:

$$V_L(t) = L \frac{dI_L}{dt}, \quad (33)$$

can be approximated with one of the finite difference scheme seen above, such that at the end, eq. (33) can be rewritten regarding the next integration voltage:

$$V_L^{n+1} = \frac{L}{T} I_L^{n+1} - \frac{L}{T} I_L^n, \quad (\text{BE}) \quad (34)$$

$$V_L^{n+1} = \underbrace{\frac{2L}{T} I_L^{n+1}}_{R_{eq}} - \underbrace{\frac{2L}{T} I_L^n + V_L^n}_{V_{eq}}. \quad (\text{TR}) \quad (35)$$

Each of these equations can be rewritten as:

$$V_L^{n+1} = R_{eq} I_L^{n+1} + V_{eq}. \quad (36)$$

This leads to the following companion model representing a voltage source in series with a resistor, commonly referred to as a Thevenin generator.

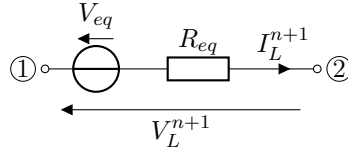


Figure 10: Companion model of the inductor with a Thevenin generator, for transient analysis.

Therefore, the MNA stamping method for an ideal inductor can be written as follows.

$$\begin{array}{c} \textcircled{1} \\ \textcircled{2} \\ \textcircled{A} \end{array} \left[ \begin{array}{cc|c} \textcircled{1} & \textcircled{2} & \textcircled{A} \\ 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & -1 & -R_{eq} \end{array} \right] \left[ \begin{array}{c} v_1^{n+1} \\ v_2^{n+1} \\ I_L^{n+1} \end{array} \right] = \left[ \begin{array}{c} 0 \\ 0 \\ V_{eq} \end{array} \right]. \quad (37)$$

It is also possible to model the ideal inductor as a current source with an internal resistance which would yield a similar equivalent circuit as for the capacitor. To do this, the eq. (33) can be approximated with the same difference scheme method but isolating the next current sample such that:

$$I_L^{n+1} = \frac{T}{L} V_L^{n+1} + I_L^n, \quad (\text{BE}) \quad (38)$$

$$I_L^{n+1} = \underbrace{\frac{T}{2L} V_L^{n+1}}_{G_{eq}} + \underbrace{\frac{T}{2L} V_L^n + I_L^n}_{V_{eq}}. \quad (\text{TR}) \quad (39)$$

The values of  $G_{eq}$  and  $I_{eq}$  could therefore be plugged in the equivalent norton generator (Fig. 9) to model the inductor.

### 5.2.3 Companion model used in the plugin

In the final plugin implementation of the MNA for transient analysis, the Thevenin generator is be used to model both the inductor and capacitor. The Trapezoidal scheme is chosen because it is simple to implement and has good stability [11, 13]. This approach allows for simplifying and sharing the same stamping method for both components in the C++ code.



### 5.3 Parametric component

Since the goal of the plugin is to allow musicians to load or create circuits with controls such as a "Distortion" knob for a guitar pedal, or a "Bass" control knob for a tone stage, we need to include models of variable/parametric components. The most commonly used components in audio circuit design are variable resistors and potentiometers. The following section will quickly explain the stamping method for these two components.

#### 5.3.1 Variable resistor

Variable resistors are resistors whose resistance value  $R(t^n)$  can change over time. Thus, the stamping method for a variable resistor is similar to that of a fixed resistor (refer to part 4.1.1), except that the value of the variable resistor must be updated in the  $\mathbf{Y}$  matrix each time it changes. This requires to invert or decomposed (as it will be shown later) the  $\mathbf{A}$  matrix whenever the resistance value is updated.

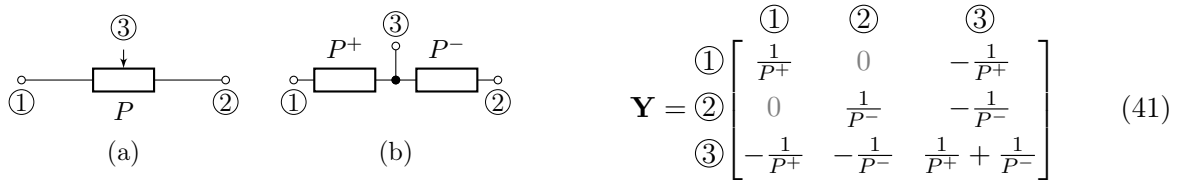


$$\mathbf{Y} = \begin{matrix} & \textcircled{1} & \textcircled{2} \\ \textcircled{1} & G & -G \\ \textcircled{2} & -G & G \end{matrix} \quad (40)$$

Figure 11: Symbol and stamp for a variable resistor.

#### 5.3.2 Potentiometer

Potentiometers are variable resistors with a third terminal, allowing them to function as adjustable voltage dividers. The stamping method for a potentiometer involves treating it as two resistors in series, where the resistance values  $P^+ = P\alpha$  and  $P^- = P(1 - \alpha)$ , with  $P$  the maximal resistance value of the potentiometer, change over time based on the knob position/rotation travel  $\alpha(t^n) \in [0, 1]$ . As for the case of the variable resistor, the  $\mathbf{A}$  matrix must be invert or decomposed each time the knob position change.



$$\mathbf{Y} = \begin{matrix} & \textcircled{1} & \textcircled{2} & \textcircled{3} \\ \textcircled{1} & \frac{1}{P^+} & 0 & -\frac{1}{P^+} \\ \textcircled{2} & 0 & \frac{1}{P^-} & -\frac{1}{P^-} \\ \textcircled{3} & -\frac{1}{P^+} & -\frac{1}{P^-} & \frac{1}{P^+} + \frac{1}{P^-} \end{matrix} \quad (41)$$

Figure 12: Symbol (a), companion model (b) and stamp for a potentiometer.

### 5.4 Nonlinear Circuits

When nonlinear elements are present, network equations become nonlinear and are more complex to solve than linear systems. Indeed, nonlinear systems can have unique, multiple, no or infinite solutions. The effective method for solving nonlinear circuits involves not directly solving their nonlinear equations. Instead, the circuits can be linearized, the resulting linear equations are solved, and this process is repeated until convergence is achieved. This iterative method is essentially the application of Newton's method to the nonlinear MNA equations of the circuit.

By definition, nonlinear elements are those whose constitutive equations are nonlinear functions of the voltage and current, or elements for which the term *in front of* the derivative linking the voltage/current to the current/voltage is not a constant. The most common nonlinear elements are diodes, transistors, and operational amplifiers. In the plugin, operational amplifier are modeled as ideal. After reviewing the Newton-Raphson method, the following parts will explain

how to solve a simple nonlinear circuit containing a diode, in order to understand the purpose of the Newton's algorithm. Then, from this example, the stamping method for the diode will be derive for MNA for transient analysis. Transistor model is covered in Appendix B.2.

#### 5.4.1 Newton-Raphson method

As it will be see in the following parts, transient analysis require a method of solving of nonlinear algebraic equations of the form:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(x_1, \dots, x_L) \\ \vdots \\ f_n(x_1, \dots, x_L) \end{bmatrix} = \mathbf{0}, \quad (42)$$

where  $\mathbf{f}(\mathbf{x})$  is a vector function of  $\mathbf{x}$ , the same vector of  $L = N + M$  unknowns as in eq. (14). Hereafter, superscript letters will be used to designate the location of the element in a vector/matrix, and superscript letters to designate the number associated to an iteration.

Considering the one dimensionnal case, the Newton-Raphson method is an iterative method that can be used to solve  $f(x) = 0$  by linearizing the function  $f(x)$  around an initial guess  $x^{(0)}$  and then solving the linearized equation (more details are provided in Appendix D). When seeking the root of this function, i.e., the point  $x$  where  $f(x) = 0$ , higher-order terms in the Taylor series can be neglected, resulting in the equation:

$$f(x) \approx f(x^{(0)}) + \dot{f}(x^{(0)})(x - x^{(0)}) = 0. \quad (43)$$

Solving this equation for  $x$  yields:

$$x \approx x^{(0)} - \frac{f(x^{(0)})}{\dot{f}(x^{(0)})}, \quad \rightarrow \quad x^{(i+1)} \approx x^{(i)} - \frac{f(x^{(i)})}{\dot{f}(x^{(i)})} \quad (44)$$

where  $x$  is the first estimated guess of the root (therefore  $x^{(1)}$ ), derived from the initial value  $x^{(0)}$ . This process is repeated to produce  $x^{(2)}$ , and so forth. Thus, for the multidimensional case, the Newton-Raphson method can be written as:

$$\mathbf{x}^{(i+1)} \approx \mathbf{x}^{(i)} - \mathbf{J}^{-1}(\mathbf{x}^{(i)})\mathbf{f}(\mathbf{x}^{(i)}), \quad (45)$$

where  $\mathbf{J}$  is the jacobian of the function  $\mathbf{f}$ . In the context of circuit simulation, since the MNA system is always a square one, the jacobian is define such as:

$$\mathbf{J}(\mathbf{x}^{(i)}) = \begin{bmatrix} \left. \frac{\partial f_1}{\partial x_1} \right|_{\mathbf{x}^{(i)}} & \cdots & \left. \frac{\partial f_1}{\partial x_L} \right|_{\mathbf{x}^{(i)}} \\ \vdots & \ddots & \vdots \\ \left. \frac{\partial f_L}{\partial x_1} \right|_{\mathbf{x}^{(i)}} & \cdots & \left. \frac{\partial f_L}{\partial x_L} \right|_{\mathbf{x}^{(i)}} \end{bmatrix}. \quad (46)$$

By repeatedly applying the above “correction” (i.e eq. (45)) to the initial solution  $\mathbf{x}^{(0)}$ , one can iteratively move closer to the “true” solution until convergence is achieved. There exist multiple ways to define convergence criteria, such as the relative error between two consecutive iterations, the absolute error, or the norm of the function, which are detailed in Appendix D.

Restarting from eq. (45), and multiplying it from both side by the jacobian to remove the invert of the jacobian, the Newton-Raphson iteration can be written as :

$$\mathbf{J}(\mathbf{x}^{(i)})\mathbf{x}^{(i+1)} = \mathbf{J}(\mathbf{x}^{(i)})\mathbf{x}^{(i)} - \mathbf{f}(\mathbf{x}^{(i)}). \quad (47)$$

Indeed, it can be shown that at the end, the stamping method apply to nonlinear circuit will end up in writting a system of this form, where the  $\mathbf{A}$  matrix of the MNA system will be  $\mathbf{J}(\mathbf{x}^{(i)})$ ,  $\mathbf{x}^{(i+1)}$  will be the  $i + 1$ -th estimation of the unknowns vector of the MNA system, and where the RHS of this equation will be  $\mathbf{b}$ , the RHS vector of the MNA system. This is demonstrated in the example case treated in Appendix B.1.

### 5.4.2 Diode

The diode, represented in Fig. 13 is a nonlinear element whose constitutive equation is given by the Shockley equation:

$$I_d = g(V_d) = I_s \left( e^{\frac{V_d}{\eta V_T}} - 1 \right), \quad (48)$$

where  $I_s$  is the reverse-bias saturation current,  $V_T \approx 25.852$  mV at 300 K/27 °C is the thermal voltage, and  $\eta$  is the ideality factor.  $I_d$  and  $V_d$  represent the current passing through the diode and the voltage accross it, respectively. For all diodes,  $I_s$  and  $\eta$  are the main parameters to be set by the user to change the behavior of the diode.

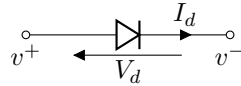


Figure 13: Symbol for a diode.

### 5.4.3 Application of the 1 dimensional Newton-Raphson method

In order to understand the purpose of the Newton-Raphson method and how it works to solve nonlinear circuits, a 1-dimensional application of this algorithm is shown below. For the 2-dimensional case (which is by far the most interesting, but longer to develop), please refer to Appendix B.1. Let's consider the following circuit, represented in Fig. 14, where the current generator  $I_{in}$  is a constant current source, with an internal resistance  $R_s$ . The diode is biased at  $v_1$  and  $I_d$ , and the goal is to find the value of  $v_1$  that solves the system.

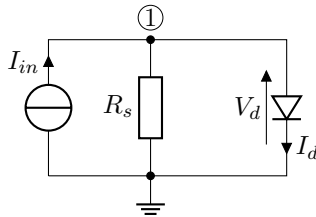


Figure 14: Simple nonlinear circuit example.

Writing the KCL at node 1 allows the formulation of the only equation constituting the system:

$$\frac{v_1 - 0}{R_s} + I_d = I_{in}, \quad (49)$$

where  $I_d = g(v_1) = I_s \left( e^{\frac{v_1}{\eta V_T}} - 1 \right)$  is the Shockley equation. Rewriting Eq. (49) as follows:

$$f(v_1) = \frac{v_1}{R_s} + g(v_1) - I_{in}, \quad (50)$$

leads to solving a 1-dimensional nonlinear equation  $f(v_1)$ . To find the value  $v_1$  that solves the system  $f(v_1) = 0$ , the Newton-Raphson method can be applied. Therefore, the  $(i + 1)$ -th estimation of  $v_1$  can be expressed as:

$$v_1^{(i+1)} = v_1^{(i)} - \frac{f(v_1^{(i)})}{\dot{f}(v_1^{(i)})}, \quad (51)$$

where:

$$f(v_1^{(i)}) = \frac{v_1^{(i)}}{R_s} + g(v_1^{(i)}) - I_{in}, \quad \text{and} \quad \dot{f}(v_1^{(i)}) = \frac{1}{R_s} + \dot{g}(v_1^{(i)}). \quad (52)$$

This relationship can be used to solve the nonlinear equation  $f(v_1) = 0$  iteratively, and the process is repeated until convergence is achieved. This example illustrates the method, though from the view perspective of the plugin, the current source would vary with the audio input signal. If the current were varying, the system would need to be solved for each different value of  $I_{in}$ .

From this example, it is possible to develop an automatized approach to linearize the diode, by developing a companion model (as for the inductor and capacitor) that could be stamp into the MNA system for transient analysis.

#### 5.4.4 Diode stamping method - Companion model

In this part, it will be seen that the diode can be represented as a combination of a conductance  $G_{eq}$  and a current source  $I_{eq}$  in the MNA system. This model is updated at each iteration of the Newton-Raphson method, allowing the circuit to be analyzed using linear techniques iteratively. Consider a diode as shown in Fig. 13 and 14, that has the element equation:

$$I_d = g(V_d) = I_s \left( e^{V_d/\eta V_T} - 1 \right). \quad (53)$$

Assuming the diode is biased at  $V_d^{(i)}$  and  $I_d^{(i)}$ , a conductance  $G_{eq}$  can be define such that:

$$G_{eq}^{(i)} \triangleq g' \left( V_d^{(i)} \right) = \left. \frac{dI_d}{dV_d} \right|_{V_d=V_d^{(i)}} = \frac{I_s}{\eta V_T} e^{V_d^{(i)}/\eta V_T}, \quad (54)$$

so that the affine approximation at  $V_d^{(i)}$  of the current is:

$$\begin{aligned} I &= \hat{g}_{V_d^{(i)}}(v) = G_{eq}^{(i)} (v - V_d^{(i)}) + I_d^{(i)} \\ &= G_{eq}^{(i)} v + I_{eq}^{(i)}, \end{aligned} \quad (55)$$

where:

$$I_{eq}^{(i)} \triangleq I_d^{(i)} - G_{eq}^{(i)} V_d^{(i)} = I_s \left( e^{V_d^{(i)}/\eta V_T} - 1 \right) - \frac{I_s}{\eta V_T} e^{V_d^{(i)}/\eta V_T}. \quad (56)$$

Then, the companion model of the diode is as shown in Fig. 15.

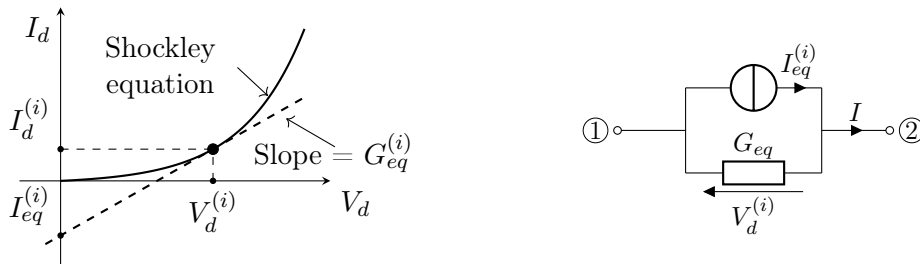


Figure 15: Affine approximation of the diode and companion model associated.

The eq. (55) for this parallel connection of  $G_{eq}$  and  $I_{eq}$  allows at each iteration to approximate the diode nonlinearity by a linear equivalent relation that is tangential to the Shockley equation. In general, this nonlinear iteration procedure is attempted with multiple nonlinear elements in the circuit, all being linearized simultaneously.

From this equation, the companion model can be deduced (it's just a graphical way to represent this equation), and therefore, the stamp method for the diode can be derived for the MNA system for transient analysis:

$$\begin{array}{c} \textcircled{1} \\ \textcircled{2} \end{array} \begin{bmatrix} \textcircled{1} & \textcircled{2} \\ G_{eq}^{(i)} & -G_{eq}^{(i)} \\ -G_{eq}^{(i)} & G_{eq}^{(i)} \end{bmatrix} \begin{bmatrix} v_1^{(i+1)} \\ v_2^{(i+1)} \end{bmatrix} = \begin{bmatrix} -I_{eq}^{(i)} \\ I_{eq}^{(i)} \end{bmatrix}. \quad (57)$$

Returning to the 1-dimensional example, and now using the affine approximation of the diode at  $V_d^{(i)} = v_1^{(i)}$ , the equation (52) becomes:

$$\dot{f}(v_1^{(i)}) = \frac{1}{R_s} + G_{eq}^{(i)}. \quad (58)$$

Thus, the  $i$ -th Newton iteration can be written as:

$$\left( \frac{1}{R_s} + G_{eq}^{(i)} \right) (v_1^{(i+1)} - v_1^{(i)}) + \frac{v_1^{(i)}}{R_s} + g(v_1^{(i)}) - I_{in} = 0, \quad (59)$$

$$\frac{v_1^{(i+1)}}{R_s} + \left[ G_{eq}^{(i)} (v_1^{(i+1)} - v_1^{(i)}) + g(v_1^{(i)}) \right] = I_{in}, \quad (60)$$

which, using the affine approximation of the diode, can be written as:

$$\left( G_{eq}^{(i)} + \frac{1}{R_s} \right) v_1^{(i+1)} + I_{eq}^{(i)} = I_{in}. \quad (61)$$

Thus, solving the linear equation resulting from the  $i$ -th Newton iteration is equivalent to solving the equivalent linear circuit, where the diode is replaced by its companion model, depicted in Fig. 15.

## 5.5 Summary

The MNA algorithm for transient analysis was introduced, in order to model reactive components like capacitors and inductors with constant time-step integration method due to fixed sample rate of audio recordings. When discretizing and approximating the derivatives of these reactive components, this element can be represented by a companion model, which are composed of linear components. The advantage is that these models have a time-invariant stamping of the  $\mathbf{A}$  matrix, constructed similarly to linear elements, meaning that  $\mathbf{A}$  needs to be stamped only once for all circuit elements, eliminating the need for inversion or LU decomposition of  $\mathbf{A}$  at each sample.

The Newton-Raphson method was then introduced for solving nonlinear circuits, focusing on linearizing the nonlinear elements iteratively. Specifically, the diode's Shockley equation was used to derive its companion model, which is updated at each Newton iteration, allowing the circuit to be analyzed using linear techniques.

At the end, the goal was to conduct approximations (integration and Newton-Raphson method) to always end up with a linear system to solve. In the next section, the process of solving linear systems within the plugin will be explained.

## 6 Solution of linear systems of equations

Because the topic of solving linear systems is vast and extensively covered in numerous works, and since it is an area of active research in science, particularly in physics simulation (FEM, BEM, graphics engine), this section will focus on presenting two methods for solving a linear system. Additionally, it will delve into the motivations for choosing the latter method discussed. Apologies for the mention/name dropping of specific method names; due to page constraints, providing more detailed explanations is challenging.

### 6.1 Matrix inversion

As we have seen this above, the goal of the MNA algorithm for transient analysis is to end up with a resistive network equation, so with a linear algebra system to solve of the form:

$$\mathbf{Ax} = \mathbf{b}. \quad (62)$$

Of course, the obvious solution to the problem  $\mathbf{Ax} = \mathbf{b}$  is  $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ , so that one way to find  $\mathbf{x}$  is to first explicitly compute  $\mathbf{A}^{-1}$  and then multiply that by  $\mathbf{b}$ , but in practice, this is very inefficient. Indeed, the matrix inversion is a very expensive operation : it can be done with Cramer's rule (which is extremely inneficient, requiring  $2(n+1)!$  multiplication to the solution, where  $n$  is the number of row/lenght of the square matrix), or Gaussian elimination (which is more efficient), but if  $\mathbf{A}$  is singular, and therefore, non-invertible, the method will fail.

### 6.2 LU Decomposition

LU decomposition (decomposition into a lower and upper triangular matrix) is recommended when dealing with equation systems where the  $\mathbf{A}$  matrix do not change but the right-hand side (the vector  $\mathbf{b}$ ) does. The LU decomposition splits a matrix  $\mathbf{A}$  into a product of a lower triangular matrix  $\mathbf{L}$  with an upper triangular matrix  $\mathbf{U}$ .

$$\mathbf{A} = \mathbf{LU} \quad \text{with} \quad \mathbf{L} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ l_{n1} & \cdots & l_{nn} \end{bmatrix} \quad \text{and} \quad \mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{bmatrix}.$$

The algorithm for solving the linear equation system  $\mathbf{Ax} = \mathbf{b}$  with LU factorisation/decomposition involves three steps:

1. Decompose the  $A$  matrix coefficient in a LU form, such that:

$$\mathbf{Ax} = \mathbf{LUx} = \mathbf{b}.$$

2. Introduction of an arbitrary vector  $\mathbf{y}$  and solving the equation system  $\mathbf{Ly} = \mathbf{b}$  by forward substitution:

$$\mathbf{y} = \mathbf{L}^{-1}\mathbf{b}.$$

3. Solving the equation system  $\mathbf{Ux} = \mathbf{y}$  by backward substitution:

$$\mathbf{x} = \mathbf{U}^{-1}\mathbf{y} = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{b}.$$

Due to the triangular form of the matrices  $\mathbf{L}$  and  $\mathbf{U}$ , the forward and backward substitution steps are straightforward to compute, and don't require an inversion of the  $\mathbf{L}$  and  $\mathbf{U}$  matrices. The forward and backward substitution steps are respectively given by:

**Forward substitution:**

$$y_k = \left( b_k - \sum_{j=1}^{k-1} l_{kj} y_j \right) \frac{1}{l_{kk}} \quad \text{for } k = 1, 2, \dots, n. \quad (63)$$

**Backward substitution:**

$$x_k = \left( y_k - \sum_{j=k+1}^n u_{kj} x_j \right) \frac{1}{u_{kk}} \quad \text{for } k = n, n-1, \dots, 1. \quad (64)$$

They exist multiple way to decompose the  $\mathbf{A}$  matrix into a lower and upper triangular matrix, such as the Doolittle, the Crout and the Cholesky decomposition, all based on Gaussian elimination [4, 5]. Depending on the algorithm used to make the decomposition, the LU decomposition approximately requires  $n^3/3 + n^2$  operations to solve a linear equation system, where  $n$  is the size of the unknown vector [4]. For  $M$  consecutive solutions, the method demands  $n^3/3 + Mn^2$  operations, because the LU decomposition is only performed once and the forward and backward substitution steps are repeated  $M$  times. But this is for the case of dense matrices.

Indeed, in the case of MNA, the matrix can be quite sparse (a  $n \times n$  matrix is said to be sparse if it has  $\mathcal{O}(n)$  non-zero elements). Most of the time, not all components are connected to the same nodes. Additionally, when introducing extra unknowns (such as in the Thevenin companion model, voltage sources, etc.), it's necessary to add rows and columns, which increases the number of zero elements. With *sparse matrix techniques*, zero-valued matrix entries are not stored and, to the extent possible, not manipulated. It is therefore advisable to take advantage of these sparsity pattern to improve the calculation time required for factoring and forward/backward substitution. In the part below, the focus is not on the specific formats for storing a sparse matrix but rather on the libraries/methods used to solve the MNA system in the plugin case.

### 6.3 Matrix library in the frame of the plugin

For the plugin algorithm, and because of the sparsity pattern of MNA systems, the design of the stamping method and the linear algebra system-solving method is tailored for sparse matrices. The code, entirely written in C/C++, required a library capable of dynamic memory allocation to adjust matrix size in real-time as the user modifies or changes the circuit. For these reasons, the EIGEN [14] library was chosen, offering a good documentation, friendly usage, and various LU decomposition algorithms, allowing performance testing on different circuits of modest size (40/50 nodes for linear circuits, 20 for nonlinear).

Various algorithms were tested, including the default sparseLU solver and two external ones supported by EIGEN: SuperLU [15] and KLU [16]. These tests were conducted on a pure C++ solution different from the plugin, which integrates the JUCE framework [17] for audio workstation compatibility. Among the algorithms, KLU proved to be the most efficient. Previous studies have also highlighted KLU's performance [18, 19]. Unfortunately, integrating KLU into the plugin was not feasible due to a dynamic library dependency issue during the plugin build phase (not compilation), limiting the implementation to the default sparseLU solver. With more time this problem could have been solved.

After detailing how to populate the MNA system for LTI, LTV, and nonlinear systems and solving them, it's essential to understand how these processes are implemented within the plugin. The next section presents a flowchart that outlines the main steps involved in emulating a real-time circuit within the plugin.

## 7 Simplified flow chart of plugin's transient analysis algorithm

A simplified flow chart of the plugin algorithm is shown in Fig. 16. For clarity and comprehension purposes, this structure is not exactly as implemented in the plugin. Key points are explained here.

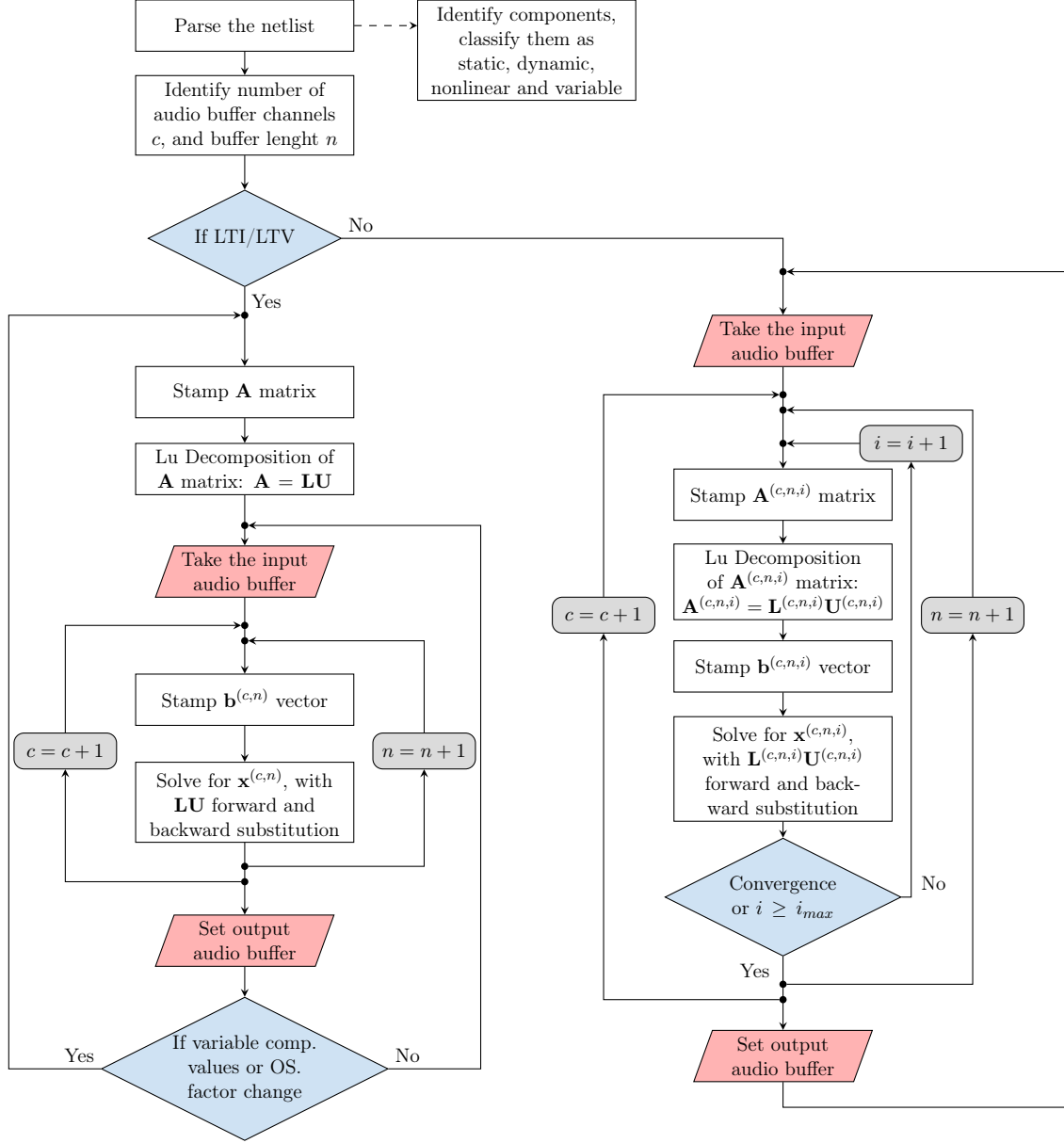


Figure 16: Simplified flow chart of the MNA based solution procedure for real-time transient analysis, where  $c$  is the number of audio channels,  $n$  is the buffer length (length of one audio channel), and  $i$  is the number of Newton-Raphson iterations.

Firstly, note the distinction between LTI/LTV and nonlinear circuits. The main difference is that LU decomposition is performed only once for LTI/LTV systems during initialization. The matrix  $\mathbf{A}$  is reused in the nested loop as long as variable component values or oversampling do not change. This is possible because resistive and reactive elements are represented by companion models, which have a stamping that does not depend on Newton iterations as in the case of nonlinear components. However, the resistances in the Thevenin model used for inductors and capacitors depend on  $F_s = 1/T_s$ . Hence, since the oversampling factor modifies  $F_s$ ,  $\mathbf{A}$  must be stamped again each time  $F_s$  changes.



The check for changes in oversampling or variable component values in LTI/LTV systems is done outside the nested loop. In real-time audio applications, this is feasible because the buffer length ranges from 64 to 256 samples per audio channel, which corresponds to maximum 6 ms at  $F_s = 44100$  Hz. Thus, if a plugin knob, linked to a potentiometer, is turned and its value changes over time, any latency or discontinuity will not be noticeable by the user between the moment the knob is turned, the value is checked, and the sound is changed.

For nonlinear systems, the matrix  $\mathbf{A}$  must be placed inside the nested loop, and it depends on  $i$  (and consequently on  $c$  and  $n$ ) due to the Newton-Raphson iterations where the equivalent conductances of diodes/transistors are updated at each iteration.

During the parsing procedure, components are categorized as static (constant stamp of  $\mathbf{A}$  and  $\mathbf{b}$ ), dynamic (stamp of  $\mathbf{A}$  and  $\mathbf{b}$  that might change depending on  $F_s$ ), nonlinear, and variable. This classification allows to efficiently populate the system, so in the case of an LTV circuit, when the resistive value of a varistor changes, the entire system does not need to be reset and re-stamped. Instead, the matrix with contributions from static and dynamic components is updated by re-stamping only the contributions from variable elements. The same mechanic is applied to nonlinear systems.

## 8 Final product - Plugin Interface

The graphical user interface of the plugin is shown in Fig. 17. All the features intended, as mentioned in section 2, have been successfully integrated. The software was coded entirely in C++ using the JUCE Framework. The source code is available on GitHub [20].

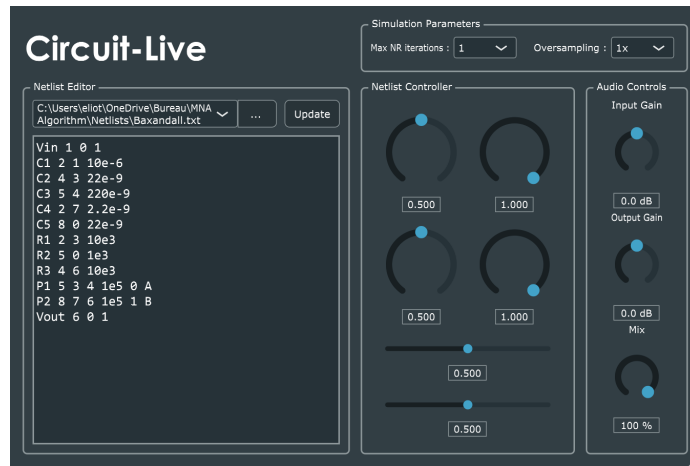


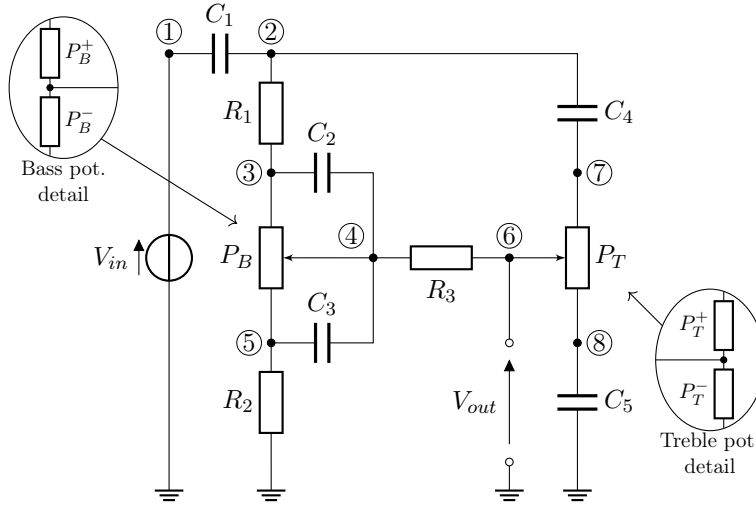
Figure 17: Graphical user interface of the plugin.

## 9 Case of study

The goal of this part is to see some circuit emulation results for two circuit configurations: a LTI/LTV system and a non-linear system. The LTI system studied is a baxandall tone, whereas the non-circuit will be a soft clipping stage commonly found in guitar pedals. For both case of study, LTspice performed simulations are used as the ground truth to compare the results of the MNA transient analysis algorithm implemented in the plugin. The audio processed by the plugin is evaluated by loading the VST3 plugin into a DAW, setting the sample rate to  $F_s = 44100$  Hz, processing the input signal through the plugin, and exporting the output signal. The resulting audio is then analyzed and compared to LTspice simulations using Python for data processing.

## 9.1 The baxandall tone

The Baxandall tone control circuit, represented in the Fig. 18, is a classic circuit used in guitar amplifiers [21]. It is a two-band equalizer that allows the user to adjust the bass and treble frequencies thanks to two potentiometers,  $P_B$  and  $P_T$ , respectively. The values of the components are given in the Table 2.



Comp.	Value
$R_1$	10 k $\Omega$
$R_2$	1 k $\Omega$
$R_3$	10 k $\Omega$
$C_1$	10 $\mu$ F
$C_2$	22 nF
$C_3$	220 nF
$C_4$	2.2 nF
$C_5$	22 nF
$P_B$	100 k $\Omega$
$P_B^+$	$P_B \cdot B$
$P_B^-$	$P_B \cdot (1 - B)$
$P_T$	100 k $\Omega$
$P_T^+$	$P_B \cdot T$
$P_T^-$	$P_B \cdot (1 - T)$

Figure 18: Baxandall Tone circuit schematic.

Table 2: Values of the components.

To compare LTspice results with those from the plugin, a linear sweep from 20 Hz to  $F_s/2$  is used as input to the plugin, and the output audio is recorded to determine the circuit's transfer function via Fourier transform. Meanwhile, LTspice performs an AC analysis of the circuit to directly obtain its transfer function. The objective is to compare the transfer functions obtained for different settings of the bass and treble control potentiometers.

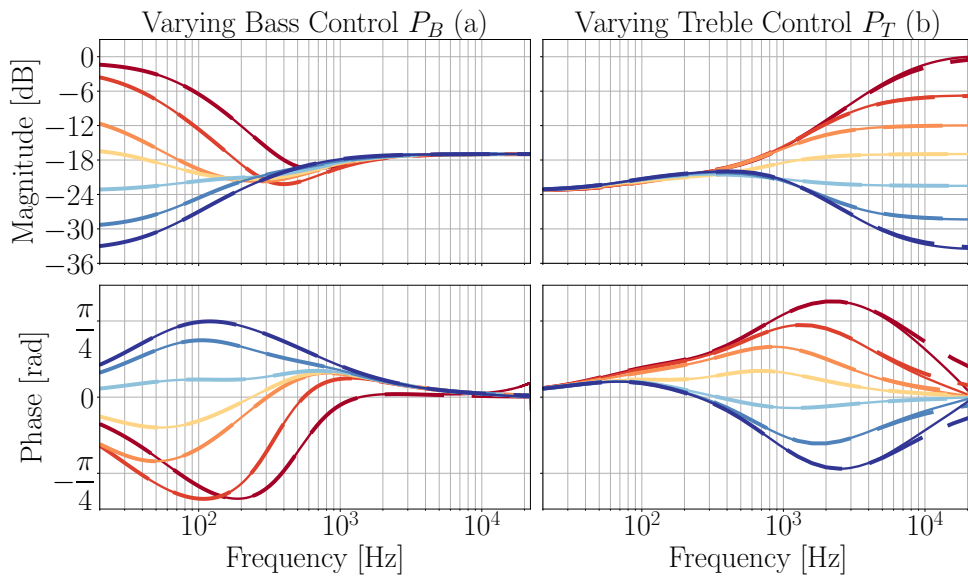


Figure 19: Transfert function of the Baxandall tone control circuit for different potentiometer settings. Dashed lines: LTspice analog response; solid lines: plugin response.

The Fig. 19 shows the transfer function  $H = V_{out}/V_{in}$  of the Baxandall tone control circuit for different potentiometer settings. On the left, the treble control is fixed while the bass control varies, and on the right, the bass control is fixed while the treble control varies (details about the potentiometer coefficient values used are provided in Appendix C). Dashed lines showing the analog response calculated using LTspice and solid lines showing the plugin response have excellent agreement: bass and treble control looks like a type of shelf filtering [22]. Distortion occurs only at high frequencies near  $F_s/2$ , as expected due to the Trapezoidal method used for numerical integration in the MNA system (in fact, the trapezoidal method is similar to the bilinear transform in the frequency domain(i.e (23)), which create frequency wrapping).

## 9.2 Tube Screamer's soft clipping stage

The soft clipping stage is a non-linear circuit that is used in guitar pedals to create a distortion effect. In the case of this study, the focus is on the Ibanez Tube Screamer soft clipping stage [23, 24, 25]. As seen Fig. 20, this circuit consists of an operational amplifier in a non-inverting configuration, where two antiparallel diodes are in the feedback loop of the operational amplifier: this kind of configuration allows the pedal to be categorized as an overdrive pedal for guitarists purists. 1N4148 diodes are used as substitute of the original MA150 diodes (as LTspice don't provide any model of the original ones). The potentiometer that modulate the amount of distortion/clipping is replace by a variable resistor  $R_d$  since one of the potentiometer extremity pin is unused. The component values are listed in Table 3.

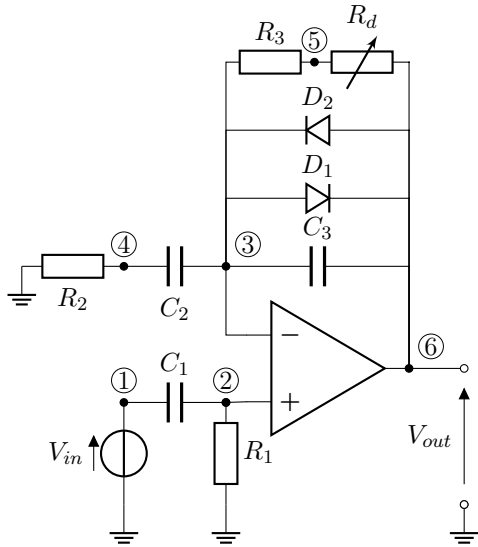


Figure 20: Ibanez Tube Screamer soft clipping stage schematic.

Comp.	Value	Notes
$R_1$	10 k $\Omega$	
$R_2$	4.7 k $\Omega$	
$R_3$	51 k $\Omega$	
$C_1$	1 $\mu$ F	
$C_2$	47 nF	
$C_3$	51 pF	
$R_d$	500 k $\Omega$	max. value
Diodes	1N4148	sub. for MA150
$I_s$	2.52 nA	saturation current
$V_T^a$	25.85 mV	thermal voltage
$\eta$	1.752	quality factor

<sup>a</sup> evaluated at 300 K.

Table 3: Ibanez Tube Screamer component.

First, let's examine the effect of the variable resistor  $R_d$ . To do this, it's important to estimate the input signal level. Electric guitars typically have nominal output levels between 60-200 mV for single-coil pickups and 200-600 mV for humbuckers and hot pickups (though the latter is less common). An input level of 80 mV is chosen for this study. This level remains unamplified or attenuated before the soft clipping stage due to the presence of an input buffer for impedance matching. Considering an 500 Hz sinusoidal input signal at 80 mV, Fig. 21 shows the output signal from the clipping stage for various  $R_d$  values, obtained with the plugin, with  $F_s = 48000$  Hz,  $i_{max} = 64$  and  $e_{abs} = 1e-6$  (absolute error parameter for Newton-Raphson convergence criteria).

As shown, the variable resistor  $R_d$  controls the amount of distortion by increasing or reducing the feedback resistance. The higher the  $R_d$ , the greater the distortion (details about the coefficients applied to  $R_d$  are in Appendix C). Due to the antiparallel diode configuration, the distortion

produced is symmetrical, resulting in only odd harmonics in the output signal. This will be verified below.

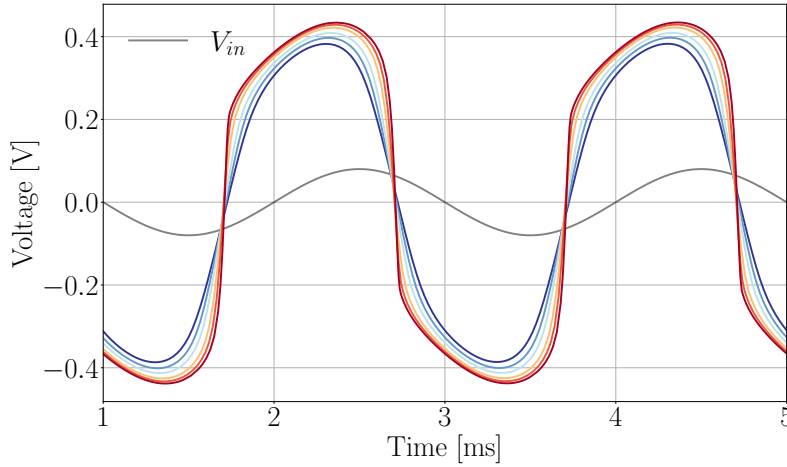


Figure 21: Output signal  $V_{out}$  of the soft clipping stage for different  $R_d$  values (from blue to red curves, the value of  $R_d$  increases).

The aim is now to observe how harmonics are generated by the clipping stage and to compare LTspice data with the plugin qualitatively. This is done by suppressing the temporal traces and their respective power spectral density. Fig. 22 shows this for two different excitation frequencies of the same amplitude (200 mV) and for different values of  $R_d$ . The spectral lines associated with the LTspice plots are slightly offset to facilitate visualization.

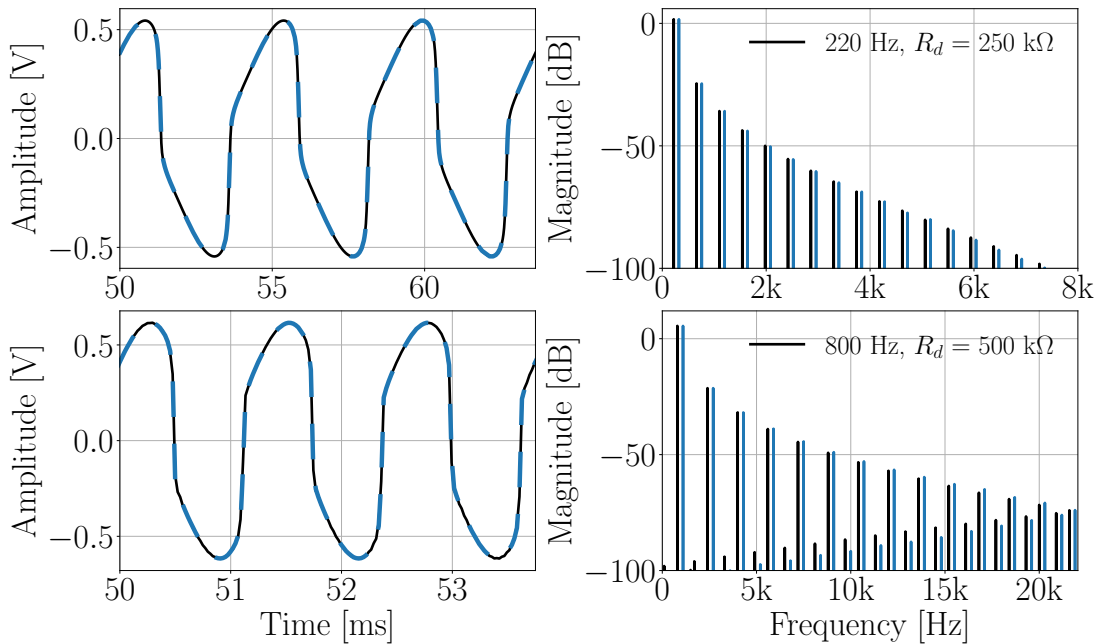


Figure 22: Time-domain plots of  $V_{out}$  and their associated power spectral densities for a sinusoidal input  $V_{in}$  with an amplitude of 200 mV, shown for two different values of  $R_d$  and two different frequencies. Blue lines: LTspice transient analysis; black lines: plugin output.

Generally, for both  $f = 220$  Hz and  $f = 800$  Hz, the time-domain plots and their associated power spectral densities overlap satisfactorily. At 220 Hz, a harmonic cascade with only odd-order harmonics is clearly visible. However, for the 800 Hz trace, the harmonic cascade is different, showing a step-like pattern rising as it approaches  $F_s/2$ . This artifact is due to frequency folding,

as the numerical resolution method is applied in the time domain, making it impossible to apply an anti-aliasing filter on the simulated signal, which already contains frequency folding artifacts. A solution to overcome this problem is to oversample the input signal and perform the simulation on this new signal, thereby pushing  $F_s/2$  further and causing the folding to occur later in the frequency scale. This folding also explains why small roughnesses are present on the 800 Hz time trace. This is the main reason why oversampling functionality has been implemented in the plugin.

At 800 Hz, we can also see more convincingly that the amplitudes of the harmonics are not exactly identical as their rank increases. One hypothesis to explain this is that the LTspice diode model is more complete: in fact, it includes a resistor in series with the Norton generator, and it is this that could cause these minor differences.

To further compare the plugin's results with LTspice, a real guitar input signal was used. The sound from a Fender Stratocaster was recorded with a sound card, and the resulting .wav file was imported into both LTspice and the DAW for output simulation. The time-domain plots are compared in Fig. 23, and the absolute error between the LTspice and plugin outputs is shown.

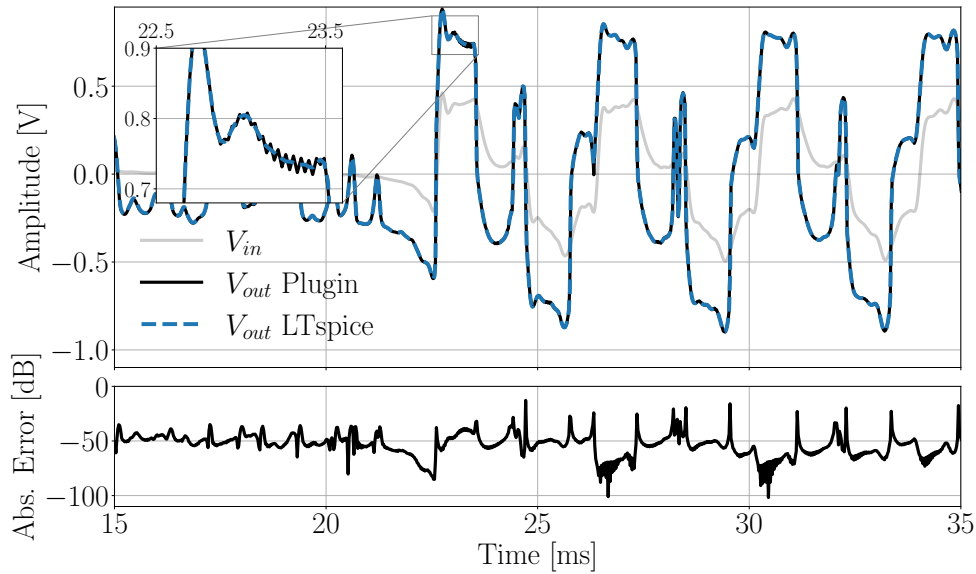


Figure 23: Time-domain plots of the real guitar input and simulated output. Blue lines: LTspice transient analysis; black lines: plugin output.

Overall, the time plots coincide satisfactorily, with the absolute error averaging around -50 dB. Interestingly, the error decreases during very marked attacks/transients: abrupt changes in input data make it more challenging to accurately estimate the output voltage. This highlights the value of the oversampling option, as it allows for less abrupt variations in the input signal.

### 9.3 Summary

After comparison with LTspice, the plugin seems to emulate satisfactorily the circuits screened in this section, namely the baxandall tone control, and the soft clipping stage of the Ibanez tube screamer. The oversampling feature was justified, especially for non-linear circuits, which generate a harmonic cascade that can lead to significant frequency aliasing.

## 10 Conclusion

The project aimed to create a real-time audio plugin capable of emulating the behavior of any electronic circuit in real time. To do so, the Modified Nodal Analysis (MNA) algorithm was chosen as the basis for the project, as it is the foundation for most modern simulation tools like LTspice, or NGspice.

First, the Modified Nodal Analysis (MNA) algorithm was studied in the frequency domain (AC analysis). In this case, only linear elements (resistors, capacitors, inductors, etc.) can be used. Moreover, it was demonstrated with an example that this method proves impractical for real-time applications. This is because the analog transfer functions derived from even small circuits (approximately 3-4 nodes) must be adapted for the discrete frequency domain. This adaptation can be complex to automate in an algorithm (using bilinear transformation for mapping  $s$  to  $z$ , biquad factorization to avoid numerical sensitivity issues, etc.).

In a second phase, the MNA algorithm was adapted for transient analysis, which is more suitable for real-time applications. In this configuration, the system is solved in the time domain and accepts all types of components. Transient analysis relies on numerical methods, specifically finite difference approximation schemes, and Newton-Raphson to obtain a system of equations containing only resistive elements, therefore a linear system. Reactive elements, through finite difference approximation of derivatives, can be represented by equivalent models called companion models, composed of source and resistance/conductance terms. For nonlinear elements, the effective method involves linearizing the nonlinear equations and iteratively finding a more precise solution at the operating point using the Newton-Raphson method, ultimately deriving a companion model for nonlinear elements.

The project's final phase involved implementing and testing the MNA algorithm for transient analysis in C++ and integrating it into a VST3 plugin using the JUCE framework. The plugin's interface was designed to allow users to enter a netlist, modify component values, and observe the circuit's behavior in real time. The plugin has been tested with various circuits, and two case studies, including a LTV/LTI and a nonlinear system, have been conducted to demonstrate its emulation accuracy by comparing plugin simulations with LTspice. Overall, the comparisons between LTspice and the plugin simulations have shown a highly satisfactory match. Minor discrepancies occur at high frequencies due to frequency wrapping, caused by the finite difference approximation scheme. For nonlinear circuits, significant frequency folding can occur if the circuit generates high-frequency and high-amplitude harmonics. To address these issues, an oversampling option was integrated, effectively pushing frequency folding and frequency wrapping to higher ranges.

## 11 Future work

The plugin holds immense potential for further enhancement, with the efficiency of linear system resolution being a critical focus. For example, integrate the KLU method - which was not included in the final solution due to time constraints - is a logical next step. Additionally, exploring alternative methods for formulating electronic circuit equations would be highly beneficial: newer approaches like the nodal-DK method [26] and wave-digital filtering[27], although more complex to adapt to various systems, show great promise. Embracing these advancements inside the realm of this audio plugin could improve the real-time circuit simulation. However the best improvement might not lie in the methods used to write the linear system, but rather in leveraging the GPU instead of limiting the plugin to run solely on the CPU [28], as is common in the current market. This shift could significantly enhance performance and open up new possibilities for real-time circuit simulation and audio processing.

## A Ideal Linear Components

### A.1 Ideal Operational Amplifier

Operational amplifiers (OPA) are widely used in electronic circuits. This model is a simplification of the real component. The assumptions made for an ideal OPA are: infinite input impedance, zero output impedance, zero offset voltage, zero input bias current, and zero input offset current. The symbol for an ideal OPA is shown in Fig. 24.

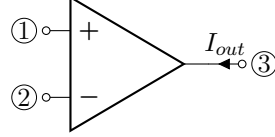


Figure 24: Symbol for an ideal operational amplifier.

The assumptions made conducted to the following equations:

$$v_1 - v_2 = 0 \qquad i_1 = i_2 = 0, \qquad i_3 = I_{out}. \quad (65)$$

Here  $I_{out}$  represent the unknow output current of the OPA. This conduce to the following stamp for the OPA, where the A indexed column and row is added to take into account the unknow output current:

$$\mathbf{A} = \begin{array}{c} \textcircled{1} \quad \textcircled{2} \quad \textcircled{3} \quad \textcircled{A} \\ \begin{array}{c} \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{A} \end{array} \left[ \begin{array}{ccc|c} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & -1 & 0 & 0 \end{array} \right] \end{array}. \quad (66)$$

### A.2 Transformer

The transformer is a passive component that is used to increase or decrease the voltage level of an DC/AC signal. The ideal transformer is a linear component that has the following assumptions: the transformer's core of an has infinite permeability, and there is no energy losses. The symbol for an ideal transformer is shown in Fig. 25.

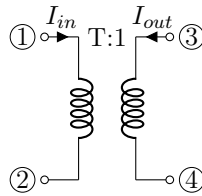


Figure 25: Symbol for an ideal transformer.

The assumptions made conducted to the following equation linking all the node voltages by the turning ratio  $T$ :

$$v_1 - v_2 = T(v_3 - v_4) \quad \rightarrow \quad v_1 - v_2 - Tv_3 + Tv_4 = 0. \quad (67)$$

The unknowns are the currents  $I_{in}$  and  $I_{out}$  that are linked by the following equation:

$$I_{out} = T \cdot I_{in}, \quad (68)$$

which result in writing the current at each node as:

$$i_1 = I_{in}, \quad i_2 = -I_{in}, \quad i_3 = T \cdot I_{in}, \quad i_4 = -T \cdot I_{in}. \quad (69)$$

Thus, the stamp for the transformer is:

$$\mathbf{A} = \begin{array}{c} \begin{array}{c} \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \\ \textcircled{A} \end{array} \left[ \begin{array}{cccc|c} \textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} & \textcircled{A} \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & T \\ 0 & 0 & 0 & 0 & -T \\ 1 & -1 & -T & T & 0 \end{array} \right] \end{array}. \quad (70)$$

### A.3 Gyrator

The ideal gyrator is a quadropole that has the property that it "gyrates" a current into a voltage and vice versa [29]. The symbol for an ideal gyrator is shown in Fig. 26.

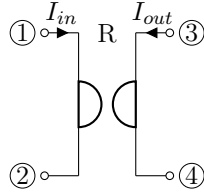


Figure 26: Symbol for an ideal gyrator.

The constitutive equations for the gyrator are:

$$(v_3 - v_4) = R \cdot I_{in}, \quad (v_1 - v_2) = -R \cdot I_{out}, \quad (71)$$

where  $R = \frac{1}{G}$  is the resistance of the gyrator. The unknown variables  $I_{out}$  and  $I_{in}$  must be expressed considering the node currents:

$$i_1 = I_{in}, \quad i_2 = -I_{in}, \quad i_3 = I_{out}, \quad i_4 = -I_{out}. \quad (72)$$

Thus, the stamp for the gyrator can be expressed with just an admittance matrix as:

$$\mathbf{Y} = \begin{array}{c} \begin{array}{c} \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \end{array} \left[ \begin{array}{cccc} \textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} \\ 0 & 0 & G & -G \\ 0 & 0 & -G & G \\ -G & G & 0 & 0 \\ G & -G & 0 & 0 \end{array} \right] \end{array}. \quad (73)$$

Otherwise, if the currents  $I_{in}$  and  $I_{out}$  are considered as unknowns, the stamp for the gyrator is:

$$\mathbf{A} = \begin{array}{c} \begin{array}{c} \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \\ \textcircled{A} \\ \textcircled{B} \end{array} \left[ \begin{array}{cccc|cc} \textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} & \textcircled{A} & \textcircled{B} \\ 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & R \\ 0 & 0 & 1 & -1 & -R & 0 \end{array} \right] \end{array}. \quad (74)$$

This last stamp method can be suitable if the linear algebraic system solver is optimized for sparse matrices.



## B Nonlinear components

### B.1 Diode

#### B.1.1 2-dimensional Newton-Raphson method example

The goal of this section is to explore the 2-dimensional application of the Newton method on a nonlinear circuit containing a diode. Similar to the 1-dimensional example discussed in the main report, this example involves a circuit operating in DC (constant current source). This approach is comparable to solving a system with a varying current source for transient analysis, were the system needs to be solved for each different value of  $I_{in}$ , if it is not constant. The circuit is shown in Fig. 27.

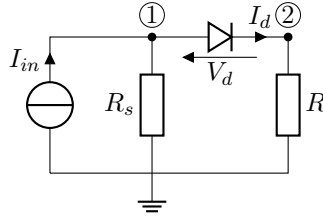


Figure 27: Example circuit with a diode.

Using the KCL at each node to write down the equations of the circuit, the following equations are obtained:

$$\frac{v_1}{R_s} + I_d = I_{in} \quad (\text{KCL at node 1}) \quad (75)$$

$$\frac{v_2}{R} = I_d \quad (\text{KCL at node 2}) \quad (76)$$

Therefore, the MNA system can be written as being:

$$\begin{bmatrix} \frac{1}{R_s} & 0 \\ 0 & \frac{1}{R} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} +I_d \\ -I_d \end{bmatrix} = \begin{bmatrix} I_{in} \\ 0 \end{bmatrix}, \quad (77)$$

or as:

$$\begin{bmatrix} \frac{1}{R_s} & 0 \\ 0 & \frac{1}{R} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} +1 \\ -1 \end{bmatrix} g(v_1, v_2) = \begin{bmatrix} I_{in} \\ 0 \end{bmatrix}, \quad (78)$$

where  $g(v_1, v_2) = I_d = I_s \left( e^{\frac{v_1 - v_2}{\eta V_T}} - 1 \right)$  is the diode current. Taking the same vectorial notation as the one used for the MNA system (i.e eq. (14)), this last formulation allow to write the system in the form:

$$\mathbf{Y}\mathbf{x} + \mathbf{h}g(\mathbf{x}) = \mathbf{b}, \quad (79)$$

where  $\mathbf{x}$  is the vector of unknowns,  $\mathbf{Y}$  is the admittance matrix,  $\mathbf{b}$  is the RHS vector containing the source terms, and where  $\mathbf{h}$  is a vector containing the coefficients of the nonlinear terms. To facilitate comprehension, the vector  $\mathbf{x}$  will be rename as  $\mathbf{x} = \mathbf{v}$ , since it only contains the unknown nodal voltages.

In this case, the goal is to solve the 2-dimensional system  $\mathbf{f}(\mathbf{v}) = 0$ , with:

$$\mathbf{f}(\mathbf{v}) = \mathbf{Y}\mathbf{v} + \mathbf{h}g(\mathbf{v}) - \mathbf{b}, \quad (80)$$

using the Newton-Raphson method. The  $i$ -th iteration of the Newton-Raphson method is given by:

$$\mathbf{v}^{(i+1)} = \mathbf{v}^{(i)} - \mathbf{J}^{-1}(\mathbf{v}^{(i)})\mathbf{f}(\mathbf{v}^{(i)}), \quad (81)$$

where  $\mathbf{J}$  is the Jacobian matrix of  $\mathbf{f}(\mathbf{x})$ . Multiplying this last equation by the jacobian, to avoid the inverse leads to:

$$\mathbf{J}(\mathbf{v}^{(i)})(\mathbf{v}^{(i+1)} - \mathbf{v}^{(i)}) + \mathbf{f}(\mathbf{v}^{(i)}) = 0 \quad (82)$$

so that we seek a zero of the affine approximation of  $f(\mathbf{v})$  around  $\mathbf{v}^{(i)}$ . The Jacobian matrix is given by:

$$\mathbf{J}(\mathbf{v}^{(i)}) = \begin{bmatrix} \left. \frac{\partial f_1}{\partial v_1} \right|_{\mathbf{v}^{(i)}} & \left. \frac{\partial f_1}{\partial v_2} \right|_{\mathbf{v}^{(i)}} \\ \left. \frac{\partial f_2}{\partial v_1} \right|_{\mathbf{v}^{(i)}} & \left. \frac{\partial f_2}{\partial v_2} \right|_{\mathbf{v}^{(i)}} \end{bmatrix} = \begin{bmatrix} \frac{1}{R_s} + \frac{I_s}{\eta V_T} e^{(v_1^{(i)} - v_2^{(i)})/(\eta V_T)} & -\frac{I_s}{\eta V_T} e^{(v_1^{(i)} - v_2^{(i)})/(\eta V_T)} \\ -\frac{I_s}{\eta V_T} e^{(v_1^{(i)} - v_2^{(i)})/(\eta V_T)} & \frac{1}{R} + \frac{I_s}{\eta V_T} e^{(v_1^{(i)} - v_2^{(i)})/(\eta V_T)} \end{bmatrix}. \quad (83)$$

By writting the following definition (i.e eq. (54)):

$$G_{eq}^{(i)} = \left. \frac{dI_d}{dV_d} \right|_{V_d^{(i)}} = \frac{I_s}{\eta V_T} e^{V_d^{(i)}/(\eta V_T)} = \frac{I_s}{\eta V_T} e^{(v_1^{(i)} - v_2^{(i)})/(\eta V_T)}, \quad (84)$$

the Jacobian matrix can be expressed as:

$$\mathbf{J}(\mathbf{v}^{(i)}) = \begin{bmatrix} \frac{1}{R_s} + G_{eq}^{(i)} & -G_{eq}^{(i)} \\ -G_{eq}^{(i)} & \frac{1}{R} + G_{eq}^{(i)} \end{bmatrix}. \quad (85)$$

Therefore, restarting from eq. (82), the Newton-Raphson iteration can be written as:

$$\mathbf{J}(\mathbf{v}^{(i)})\mathbf{v}^{(i+1)} = \mathbf{J}(\mathbf{v}^{(i)})\mathbf{v}^{(i)} - \mathbf{f}(\mathbf{v}^{(i)}), \quad (86)$$

$$\begin{bmatrix} \frac{1}{R_s} + G_{eq}^{(i)} & -G_{eq}^{(i)} \\ -G_{eq}^{(i)} & \frac{1}{R} + G_{eq}^{(i)} \end{bmatrix} \begin{bmatrix} v_1^{(i+1)} \\ v_2^{(i+1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{R_s} + G_{eq}^{(i)} & -G_{eq}^{(i)} \\ -G_{eq}^{(i)} & \frac{1}{R} + G_{eq}^{(i)} \end{bmatrix} \begin{bmatrix} v_1^{(i)} \\ v_2^{(i)} \end{bmatrix} - \begin{bmatrix} \frac{v_1^{(i)}}{R_s} - I_d^{(i)} + I_{in} \\ \frac{v_2^{(i)}}{R} + I_d^{(i)} \end{bmatrix}. \quad (87)$$

Then, by writting (i.e eq. (56)):

$$I_{eq}^{(i)} = I_d^{(i)} - G_{eq}^{(i)} V_d^{(i)} = I_d^{(i)} - G_{eq}^{(i)} (v_1^{(i)} - v_2^{(i)}), \quad (88)$$

the Newton-Raphson iteration becomes:

$$\begin{bmatrix} \frac{1}{R_s} + G_{eq}^{(i)} & -G_{eq}^{(i)} \\ -G_{eq}^{(i)} & \frac{1}{R} + G_{eq}^{(i)} \end{bmatrix} \begin{bmatrix} v_1^{(i+1)} \\ v_2^{(i+1)} \end{bmatrix} = \begin{bmatrix} I_{in} - I_{eq}^{(i)} \\ I_{eq}^{(i)} \end{bmatrix}. \quad (89)$$

If the stamp method was used to write the MNA system directly, the system would look exactly the same.

**Remark:** When writing the MNA system for nonlinear circuits using the stamping method, it always results in a system of the form shown in Eq. (86), and the right and side of this equation can be expressed as a column vector. This system is then solved using methods such as LU factorization.

## B.2 Bipolar Junction Transistor (BJT)

The Bipolar Junction Transistor (BJT) is a 3-port element, with the base, collector and emitter as its ports. It's a polarized element, which means that its behavior is dependent on the biasing of the element (like the diode). The BJT is a nonlinear element, and the model chosen to describe its behavior is the Ebers-Moll model. There exist other models, such as the Gummel-Poon model, but the Ebers-Moll model allows to describe the transistor in a more simple way, which is sufficient for most applications, and in our case for real-time audio processing. The following section employs the notation introduced in Pillage et al. [13].

Consider an NPN BJT with its Ebers-Moll model, shown in Fig. 28. The model equations of this BJT are given by the following equations:

$$i_e = -I_{es} \left( e^{v_{be}/V_{Te}} - 1 \right) + a_R I_{cs} \left( e^{v_{bc}/V_{Tc}} - 1 \right) \quad (4.154)$$

$$i_c = a_F I_{es} \left( e^{v_{be}/V_{Te}} - 1 \right) - I_{cs} \left( e^{v_{bc}/V_{Tc}} - 1 \right) \quad (4.155)$$

$$i_b = -(i_e + i_c) \quad (4.156)$$

where  $I_{es}$  and  $I_{cs}$  are the emitter and collector junction saturation currents respectively,  $V_{Te}$  and  $V_{Tc}$  are the thermal voltages,  $kT/q$ , scaled by the appropriate nonideality factor for each junction, and  $\alpha_F$  and  $\alpha_R$  are the forward and reverse current gains.

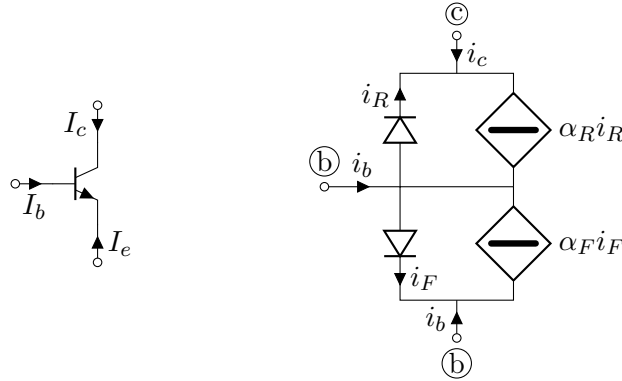


Figure 28: NPN type BJT and NPN type BJT Ebers-Moll model.

To find the affine approximation, these equations will be linearized directly, starting by finding the Jacobian  $\mathbf{J}(\mathbf{x})$ :

$$\frac{\partial i_e}{\partial v_{be}} = -\frac{I_{es}}{V_{Te}} e^{v_{be}/V_{Te}} \triangleq -g_{ee}, \quad (90)$$

$$\frac{\partial i_e}{\partial v_{bc}} = a_R \frac{I_{cs}}{V_{Tc}} e^{v_{bc}/V_{Tc}} \triangleq g_{ec}, \quad (91)$$

$$\frac{\partial i_c}{\partial v_{be}} = a_F \frac{I_{es}}{V_{Te}} e^{v_{be}/V_{Te}} \triangleq g_{ce}, \quad (92)$$

$$\frac{\partial i_c}{\partial v_{bc}} = -\frac{I_{cs}}{V_{Tc}} e^{v_{bc}/V_{Tc}} \triangleq -g_{cc}, \quad (93)$$

$$\frac{\partial i_b}{\partial v_{be}} = g_{ee} - g_{ce}, \quad (94)$$

$$\frac{\partial i_b}{\partial v_{bc}} = g_{cc} - g_{ec}. \quad (95)$$

Therefore, the affine approximation for each terminal signal can be written as:

$$\hat{i}_e = g_{ee}\hat{v}_{be} + g_{ec}\hat{v}_{bc}, \quad (96)$$

$$\hat{i}_c = g_{ce}\hat{v}_{be} + g_{cc}\hat{v}_{bc}, \quad (97)$$

or

$$\hat{i}_e = -g_{ee}\hat{v}_e - g_{ec}\hat{v}_c + (g_{ee} + g_{ec})\hat{v}_b, \quad (98)$$

$$\hat{i}_c = -g_{ce}\hat{v}_e - g_{cc}\hat{v}_c + (g_{ce} + g_{cc})\hat{v}_b, \quad (99)$$

$$\hat{i}_b = -(\hat{i}_e + \hat{i}_c), \quad (100)$$

where:

$$I_e^{(k)} \triangleq i_e^{(k)} + g_{ee}^{(k)}v_{be}^{(k)} - g_{ec}^{(k)}v_{bc}^{(k)}, \quad (101)$$

$$I_c^{(k)} \triangleq i_c^{(k)} - g_{ce}^{(k)}v_{be}^{(k)} + g_{cc}^{(k)}v_{bc}^{(k)}, \quad (102)$$

and where:

$$i_e^{(k)} = -I_{es} \left( e^{v_{be}^{(k)}/V_{Te}} - 1 \right) + a_R I_{cs} \left( e^{v_{bc}^{(k)}/V_{Tc}} - 1 \right), \quad (103)$$

$$i_c^{(k)} = a_F I_{es} \left( e^{v_{be}^{(k)}/V_{Te}} - 1 \right) - I_{cs} \left( e^{v_{bc}^{(k)}/V_{Tc}} - 1 \right). \quad (104)$$

This last set of equation can be schematized by the companion model represented in Fig. 29. From this companion model, it's possible to write the stamp procedure for the BJT, and to solve the system using the Newton-Raphson method.

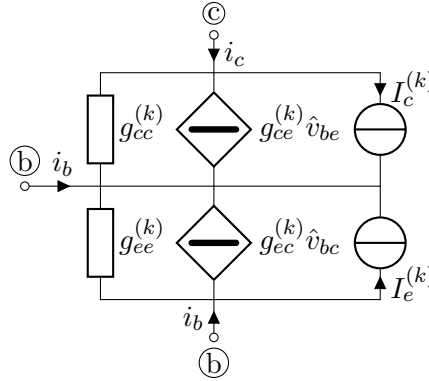


Figure 29: Companion model for the NPN type BJT Ebers-Moll model.

The stamping method for the NPN BJT is therefore:

$$\mathbf{Y} = \begin{matrix} & \textcircled{c} & & \textcircled{c} & & \textcircled{b} \\ \textcircled{c} & g_{ee}^{(k)} & & -g_{ec}^{(k)} & & g_{ec}^{(k)} - g_{ee}^{(k)} \\ \textcircled{c} & -g_{ec}^{(k)} & & g_{ee}^{(k)} & & g_{ce}^{(k)} - g_{cc}^{(k)} \\ \textcircled{b} & g_{ce}^{(k)} - g_{ee}^{(k)} & & g_{ec}^{(k)} - g_{cc}^{(k)} & & g_{ee}^{(k)} + g_{cc}^{(k)} - g_{ce}^{(k)} - g_{ec}^{(k)} \end{matrix}, \quad \mathbf{b} = \begin{bmatrix} -I_e^{(k)} \\ -I_c^{(k)} \\ I_e^{(k)} + I_c^{(k)} \end{bmatrix} \quad (105)$$

In the same fashion, the companion model for the PNP BJT can be derived.

## C Potentiometers

The potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor. Most audio equipment potentiometers do not have a linear "alpha" coefficient (which controls resistance on both sides of the wiper, such that  $P^+ = \alpha \times R_{max}$ ,  $P^- = (1 - \alpha) \times R_{max}$ ) with rotation travel. The most common type is the "logarithmic" potentiometer, also known as an audio taper potentiometer. Its alpha coefficient varies logarithmically with the angle of rotation, providing a linear change in volume for a given angle of rotation, which compensates for the human ear's logarithmic perception of sound intensity.

The "log" potentiometer isn't truly logarithmic but approximates a logarithmic function. The curve linking the alpha coefficient to the rotation travel is called the "taper," and not all tapers have a logarithmic shape. Various tapers exist, each with a different resistance-to-rotation travel relationship, but the most common are the "15A" and "15C" tapers. Figure 30 shows the resistance-to-travel relationship for the "15A" and "15C" tapers. The functions in this figure are piecewise linear functions constructed based on resistance taper definitions from the ALPS datasheet [30].

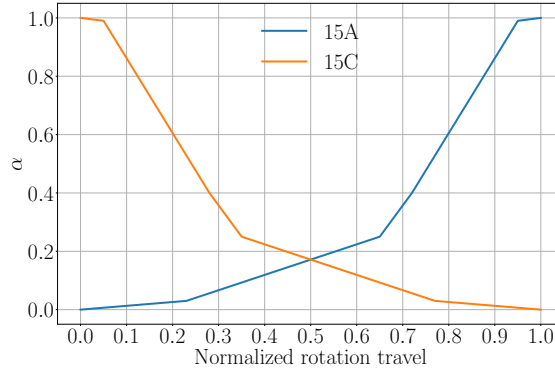


Figure 30: Resistance-to-travel relationship for the "15A" and "15C" tapers.

For the plugin, the parsing algorithm allows adding an "A" or "C" suffix to the potentiometer value in the netlist to specify the taper. The plugin then uses the corresponding alpha coefficient to compute the potentiometer value.

### C.1 Baxandall Tone Control

In the case of the Baxandall tone control circuit, the potentiometer that control the bass is a "15A" taper, and the one that controls the treble is a "15C" taper.

- For the varying bass control subfigure (Fig. 19a), the normalized rotation travel of the treble knob have been set to 0.6 and the normalized rotation travel of the bass knob have been setted to [0.1, 0.2, 0.3, 0.5, 0.7, 0.9, 1], from the blue curve to the red curve.
- For the varying treble control subfigure (Fig. 19b), the normalized rotation travel of the bass knob have been set to 0.3 and the normalized rotation travel for the treble knob have been succesively setted to [0.1, 0.2, 0.3, 0.6, 0.8, 0.9], from the red curve to the blue curve.

### C.2 Tube Screamer soft clipping stage

In the case of the Tube Screamer circuit (i.e Fig. 21), the potentiometer that controls the distortion is a "15A" taper. The values that have been set for the normalized rotation travel are [0, 0.2, 0.3, 0.5, 0.7, 1], from the blue curve to the red curve.

## D Newton-Raphson Method

The Newton-Raphson algorithm is an iterative method for finding the roots of a function using its derivative. It generates a sequence of values that converge to the desired root from an initial guess. This method approximates the root by using the tangent to the function's curve. The initial root estimate is refined iteratively until the desired precision is achieved.

### D.1 Derivation

The Newton-Raphson method is derived using the Taylor series expansion of the function  $f(x)$  around the initial value  $x^0$ . If  $f(x)$  is sufficiently smooth and differentiable at  $x^0$ , the Taylor series of  $f$  at this point is:

$$f(x) = \sum_{k=0}^{\infty} \frac{1}{k!} \frac{\partial^k f(x)}{\partial x^k} \Big|_{x=x^0} (x - x^0)^k. \quad (106)$$

When seeking the root of this function, i.e., the point  $x$  where  $f(x) = 0$ , higher-order terms in the Taylor series can be neglected, resulting in the equation:

$$f(x) \approx f(x^0) + \dot{f}(x^0)(x - x^0) = 0. \quad (107)$$

Solving this equation for  $x$  yields:

$$x \approx x^0 - \frac{f(x^0)}{\dot{f}(x^0)}, \quad (108)$$

where  $x$  is the first estimate  $x^1$  of the root, derived from the initial value  $x^0$ . This process is repeated to produce  $x^2$ , and so forth. By induction, the Newton-Raphson algorithm is:

$$\begin{cases} x^0 = \text{initial value}, \\ x^{i+1} = x^i - \frac{f(x^i)}{\dot{f}(x^i)}, \end{cases} \quad (109)$$

where  $x^{i+1}$  is the  $(i + 1)$ -th approximation of the root.

Since the exact solution is unknown, stopping conditions are based on the previous approximation of the root, using either a relative or absolute error:

$$\text{Exit the loop if } \begin{cases} \left| \frac{x^i - x^{i-1}}{x^i} \right| < \text{relative error}, \\ |x^i - x^{i-1}| < \text{absolute error}. \end{cases}$$

Using a relative error criterion poses issues when  $x^i = 0$ , resulting in undefined mathematical expressions (division by 0), which appear as "NaN" (Not a Number) or "Inf" (Infinity) in computing. Therefore, an "hybrid" error criterion is often used, combining both relative and absolute errors, such that  $x^{i+1}$  is computed until:

$$|x^{i+1} - x^i| < e_{abs} + e_{rel}|x^{i+1}|, \quad (110)$$

where  $e_{abs}$  and  $e_{rel}$  are the absolute and relative errors, respectively.

## D.2 Numerical Convergence

Let  $e^i = |r - x^i|$  be the error after step  $i$  of an iterative method where  $r$  is the exact solution. The iteration converges quadratically if:

$$M = \lim_{i \rightarrow \infty} \frac{e^{i+1}}{(e^i)^2} < \infty. \quad (111)$$

Theorem:

Let  $f$  be twice continuously differentiable and  $f(r) = 0$ . If  $\dot{f}(r) \neq 0$ , then the Newton method converges locally and quadratically to  $r$ . The error  $e^i$  at step  $i$  satisfies:

$$\lim_{i \rightarrow \infty} \frac{e^{i+1}}{(e^i)^2} = M, \quad (112)$$

where:

$$M = \frac{\ddot{f}(r)}{2\dot{f}(r)}. \quad (113)$$

Quadratic convergence of the Newton-Raphson method means that the error between the approximated solution at each iteration and the exact solution decreases quadratically. For an estimate  $x^i$  of the root, the error between the next estimate  $x^{i+1}$  and the exact solution will be approximately the square of the error between  $x^i$  and the exact solution  $r$ .

### Proof of Quadratic Convergence

To prove quadratic convergence, the Newton method is derived again, keeping a close watch on the error  $e^i = |r - x^i|$  at each iteration. Taylor's formula indicates the difference between the values of a function at a given point and a nearby point. For two points, the root  $r$  and the current estimate  $x^i$  after  $i$  steps:

$$f(r) = f(x^i) + (r - x^i)\dot{f}(x^i) + \frac{(r - x^i)^2}{2}\ddot{f}(\xi^i). \quad (114)$$

Here,  $\xi^i$  lies between  $x^i$  and  $r$ . Since  $r$  is the root,  $f(r) = 0$ , thus:

$$\begin{aligned} 0 &= f(x^i) + (r - x^i)\dot{f}(x^i) + \frac{(r - x^i)^2}{2}\ddot{f}(\xi^i), \\ -\frac{f(x^i)}{\dot{f}(x^i)} &= r - x^i + \frac{(r - x^i)^2}{2}\frac{\ddot{f}(\xi^i)}{\dot{f}(x^i)}, \end{aligned} \quad (115)$$

assuming  $\dot{f}(x^i) \neq 0$ . Rearranging this expression, the next Newton-Raphson iteration is compared to the root:

$$x^i - \frac{f(x^i)}{\dot{f}(x^i)} - r = \frac{(r - x^i)^2}{2}\frac{\ddot{f}(\xi^i)}{\dot{f}(x^i)}, \quad (116)$$

$$x^{i+1} - r = (e^i)^2 \frac{\ddot{f}(\xi^i)}{2\dot{f}(x^i)}, \quad (117)$$

$$e^{i+1} = (e^i)^2 \left| \frac{\ddot{f}(\xi^i)}{2\dot{f}(x^i)} \right|. \quad (118)$$

In this equation, the error at step  $i$  is  $e^i = |x^i - r|$ . As  $\xi^i$  lies between  $r$  and  $x^i$ , it converges to  $r$  as  $x^i$  does, and:

$$\lim_{i \rightarrow \infty} \frac{e^{i+1}}{(e^i)^2} = \left| \frac{\ddot{f}(r)}{2\dot{f}(r)} \right|, \quad (119)$$

which defines quadratic convergence.

The developed error formula can be considered as:

$$e^{i+1} \approx M(e^i)^2, \tag{120}$$

where  $M = \left| \ddot{f}(r)/2\dot{f}(r) \right|$ , assuming  $\dot{f}(r) \neq 0$ . The approximation improves as the Newton method converges, since the estimates  $x^i$  approach  $r$  and  $\xi^i$  is between  $x^i$  and  $r$ .



## Bibliography

- [1] H. Hildebrand. *Patent, Pitch detection and intonation correction apparatus and method*. Antares Audio Technologies LLC, 1998.
- [2] Chung-Wen Ho, Albert Ruehli, and Pierce Brennan. “The Modified Nodal Approach to Network Analysis”. In: *Circuits and Systems, IEEE Transactions on* 22 (1975), pp. 504–509.
- [3] R. Rohrer et al. “CANCER: Computer Analysis of Nonlinear Circuits Excluding Radiation”. In: *IEEE Solid-State Circuits Magazine* 3 (2011), pp. 40–42.
- [4] L.W. Nagel. *SPICE2: A Computer Program to Simulate Semiconductor Circuits*. Electronics Research Laboratory Berkeley, College of Engineering, University of California, 1975.
- [5] F.N. Najm. *Circuit Simulation*. IEEE Press. Wiley, 2010, p. 23.
- [6] S. Seshu and M.B. Reed. *Linear Graphs and Electrical Networks*. Addison-Wesley series in behavioral science: Quantitative methods. Addison-Wesley Publishing Company, 1961.
- [7] J. Vlach and K. Singhal. *Computer Methods for Circuit Analysis and Design*. Van Nostrand Reinhold Electrical/Computer Science and Engineering Series. Springer, 1994.
- [8] U. Zölzer et al. *DAFX - Digital Audio Effects*. John Wiley & Sons, 2002.
- [9] A. Gray and J. Markel. “Digital lattice and ladder filter synthesis”. In: *IEEE Transactions on Audio and Electroacoustics* 21.6 (1973), pp. 491–500.
- [10] Aaron Wishnick. “Time-Varying Filters for Musical Applications”. In: *International Conference on Digital Audio Effects*. 2014.
- [11] Kendall E. Atkinson, Weimin Han, and David Stewart. *Numerical Solution of Ordinary Differential Equations*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. John Wiley & Sons, 2011.
- [12] Timothy Sauer. *Numerical Analysis, 2<sup>nd</sup> edition*. Pearson Addison Wesley, 2006.
- [13] Lawrence Pillage. *Electronic Circuit & System Simulation Methods (SRE)*. McGraw-Hill, Inc., 1998.
- [14] Gaël Guennebaud et al. *Eigen*. <https://gitlab.com/libeigen/eigen>. 2010.
- [15] Xiaoye S. Li. “An Overview of SuperLU: Algorithms, Implementation, and User Interface”. In: 31.3 (Oct. 2005), pp. 302–325.
- [16] Timothy A. Davis and Ekanathan Palamadai Natarajan. “Algorithm 907: KLU, A Direct Sparse Solver for Circuit Simulation Problems”. In: *ACM Trans. Math. Softw.* 37.3 (Sept. 2010).
- [17] *JUCE: Jules’ Utility Class Extensions*. <https://github.com/juce-framework>.
- [18] Lukas Razik et al. “A comparative analysis of LU decomposition methods for power system simulations”. In: June 2019, pp. 1–6.
- [19] Jan Dinkelbach et al. “Factorisation Path Based Refactorisation for High-Performance LU Decomposition in Real-Time Power System Simulation”. In: *Energies* 14 (Nov. 2021), p. 7989.
- [20] Eliot Deschang. *CircuitLive source code*. <https://github.com/eliot-des/CircuitLive>. Accessed: 2024-05-30.
- [21] Peter J Baxandall. “Negative-feedback tone control”. In: *Wireless World* 58.10 (1952), pp. 402–405.
- [22] Vesa Välimäki and Joshua Reiss. “All About Audio Equalization: Solutions and Frontiers”. In: *Applied Sciences* 6 (May 2016), p. 129.

- [23] ElectroSmash. *Tube Screamer Circuit Analysis*. <https://www.electrosmash.com/tube-screamer-analysis>. Accessed: 2024-05-30.
- [24] GeoFex. *The Technology of the Tube Screamer*. [http://www.geofex.com/Article\\_Folders/TStech/tsexfram.htm](http://www.geofex.com/Article_Folders/TStech/tsexfram.htm). Accessed: 2024-05-30. n.d.
- [25] David Yeh, Jonathan Abel, and Julius Smith. “Simplified, physically-informed models of distortion and Overdrive guitar effects pedals”. In: *Proceedings of the 10th International Conference on Digital Audio Effects, DAFx 2007* (Jan. 2007).
- [26] Martin Holters and Udo Zölzer. “Physical Modelling of a Wah-wah Effect Pedal as a Case Study for Application of the Nodal DK Method to Circuits with Variable Parts”. In: Sept. 2011.
- [27] Kurt Werner et al. “Resolving Wave Digital Filters with Multiple/Multiport Nonlinearities”. In: Nov. 2015.
- [28] GPU audio. *GPU audio SDK*. <https://www.gpu.audio/>. Accessed: 2024-05-30.
- [29] Bernard DH Tellegen. “The gyrator, a new electric network element”. In: *Philips Res. Rep* 3.2 (1948), pp. 81–101.
- [30] ALPS Alpine. *Resistance Taper Definitions*. <https://www.onlinecomponents.com/productfiles/mf-ale/alps%20-%20resistance%20taper%20definitions.pdf>. Accessed: 2024-05-30.