Modelling the $I-V$ curve behaviour of memristors

Elisa Patrícia da Costa Martins
MSc. in Engineering Physics
Department of Physics and Astronomy
Faculty of Sciences of the University of Porto and Faculty of Engineering of the University of Porto
2024

Modelling the $I - V$ curve behaviour of memristors

Elisa Patrícia da Costa Martins

Dissertation carried out as part of the Master's Degree in Engineering Physics

Department of Physics and Astronomy

2024

Supervisor
Catarina Dias, Initial Level Doctorate, IFIMUP

Co-supervisor
João Oliveira Ventura, Principal Investigator, IFIMUP

IFIMUP
Instituto de Física de Materiais Avançados, Nanotecnologia e Fotónica
Universidade do Porto

Acknowledgements

I would like to thank my supervisor, Doctor Catarina Dias, for the weekly meetings and the data provided for the development of this work. I want to thank my co-supervisor, Doctor João Oliveira Ventura, and the group at IFIMUP for the weekly meetings, where I got to know their work and expand my understanding of the field of memristors.

I would also like to thank my friends for helping me keep my sanity during the challenging experience that was taking this degree.

I want to thank my father for his continuous support, despite not always understanding my struggles.

Finally, I want to thank all my teachers and professors, from pre-school to this last year of my master's degree, because I could never reach this point without the knowledge I learnt from them.

Resumo

Os memristors, tendo a sua teoria sido apresentada pela primeira vez em 1971, são considerados o quarto elemento fundamental dos circuitos. Estes dispositivos comutam entre estados de resistência de uma forma que os torna soluções promissoras para a computação neuromórfica ou aplicações de memória.

Este trabalho explora a modelação do comportamento característico de corrente-tensão $(I - V)$ dos memristores. O objetivo principal deste trabalho é analisar e otimizar diferentes abordagens para representar com precisão as curvas $I - V$ de dispositivos memristivos. Isto permite a definição de um modelo capaz de descrever o comportamento elétrico de dispositivos experimentais, que pode ser posteriormente utilizado em simulações que visam validar a construção de um dispositivo físico ou circuito que o inclua.

O LTSpice é um *software freeware* e amplamente utilizado nesta área e é usado para validar os modelos empíricos, ajudando à compreensão do comportamento de comutação do memristor.

No trabalho desenvolvido para esta dissertação, foram aplicadas técnicas de extração de parâmetros a um conjunto específico de dados experimentais, recolhidos de um dispositivo Si/Ag, usando o modelo generalizado com o seno hiperbólico (GHS), resultando em previsões com um erro percentual absoluto médio de 26%, quando aplicadas a um conjunto de curvas de teste.

Vários modelos baseados em redes neuronais também foram explorados neste trabalho, na tentativa de melhorar a precisão das previsões, em particular, onde as previsões efetuadas com a implementação do modelo GHS demonstraram maiores discrepâncias em relação aos dados experimentais. A introdução de redes neuronais, sozinhas ou em conjunto com o modelo GHS, melhorou o resultado, atingindo erros de cerca de 20%.

Estes resultados mostram que a escolha do modelo deve ser baseada no conjunto de dados disponível, bem como na aplicação pretendida para o modelo. Eles também mostram que, para simulações computacionalmente leves, o uso de redes neuronais como modelo pode ser benéfico, pois não é necessário criar um modelo personalizado para cada dispositivo, alcançando uma precisão semelhante ou até melhor à obtida com modelos generalizados para memristores.

A construção de um modelo de variável de estado para este conjunto de dados específico ou o uso de um conjunto de dados maior para treinar as redes neuronais é sugerida para

melhorar os resultados.

Palavras-chave: memristor, modelos de memristor, redes neuronais, curvas corrente-tensão, comutação de resistência, gráfico de histerese cruzado, modelo de variável de estado

Abstract

Memristors, first theorised in 1971, are considered the fourth fundamental circuit element. These devices switch between resistance states in a way that makes them promising solutions for neuromorphic computation or memory applications.

This work explores the modelling of the characteristic current-voltage $(I - V)$ behaviour of memristors. The main objective of this work is to analyse and optimise different approaches to accurately represent the $I - V$ curve of memristive devices. This allows the definition of a model able to describe the electrical behaviour of experimental devices, which can subsequently be used in simulations that aim to validate the construction of a physical device or circuit.

LTSpice, being highly used in this field and freeware, is used to electrically validate the empirical models, providing insight into the memristor switching behaviour.

In the work developed for this dissertation, parameter extraction techniques are applied for a specific Si/Ag experimental dataset using the General Hyperbolic Sine (GHS) model, which resulted in predictions with a mean absolute percentage error of 26%, when applied to a set of testing curves.

Different neural network-based models were also explored, in this work, in an attempt to improve the accuracy of the predictions, particularly where the predictions achieved by the implemented GHS model presented the most discrepancies to the experimental data. The introduction of neural networks, alone or in conjunction with the GHS model, improved the result, reaching errors of about 20%.

These results show that the choice of model should be based on the available dataset, as well as the intended application for the model. They also show that for computationally light simulation, the use of neural networks as a model can be beneficial, as it is not necessary to create a custom model for each device, achieving an accuracy similar to or even better than the one attained using generalised models for memristors.

The construction of a state-variable model for this specific dataset or the use of a larger dataset to train the neural networks is suggested to further improve the results.

Keywords: memristor, memristor models, neural networks, current-voltage curves, resistive switching, pinched hysteresis loop, state variable model

Table of Contents

List of Tables

List of Figures

List of Code Snippets

List of Abbreviations

ADAM Adaptive Moment Estimation. 35, 43

ANN Artificial Neural Networks. 32

CF conductive filaments. 7, 26

ECCMM Empirical Chalcogenide Compact Memristor Model. v, 11, 14

GHS General Hyperbolic Sine. ii, iv, v, ix, 2, 11, 18, 20, 26, 31, 33

HRS high resistance state. viii, 7, 11, 26

LID Linear Ion Drift. v, 7, 8

LR learning rate. 34–46

LRS low resistance state. viii, 7, 11, 26

MAPE Mean Absolute Percentage Error. v, x, 35–46

MIM metal-insulator-metal. 7, 12

MSE Mean Squared Error. v, x, 35–43, 45, 46

ReLU rectefied linear unit. 32

RS resistive switching. viii, 7, 10, 11, 16, 18

STB Simmons Tunnel Barrier. v, viii, 8, 9

TEAM ThrEshold Adaptive Memristor. 9, 10

VAE Variational AutoEncoder. 42

VTEAM Voltage ThrEshold Adaptive Memristor. v, 9, 10

Introduction

The rapid advancement in technology has continually pushed the boundaries of computational capabilities and memory storage solutions. Traditional computing architectures, primarily based on the von Neumann model, have faced significant challenges in meeting the demands of modern applications, particularly in the era of big data and artificial intelligence. One of the critical bottlenecks in these systems is the separation of memory and processing units, leading to substantial data transfer delays and energy inefficiencies. This has spurred interest in alternative computing paradigms, among which memristive technologies have emerged as a promising solution [1].

Memristors, short for memory resistors, are a class of nonvolatile memory devices that exhibit unique properties, such as variable analog conductance, making them suitable for a wide range of applications, from memory storage to neuromorphic computing [2]. First theorized by Leon Chua in 1971 [3], memristors were later physically realized by researchers at HP Labs in 2008 [4]. These devices are characterized by their ability to retain a history of the electrical charge that has passed through them, effectively 'remembering' their resistance state even when the power is removed. This property is particularly advantageous for creating energy-efficient memory systems and mimicking synaptic functions in artificial neural networks [5].

The potential of memristors extends beyond simple memory storage. Their ability to perform logic operations [6] and their compatibility with existing semiconductor fabrication processes [7] make them ideal candidates for in-memory computing architectures. In such systems, data processing occurs within the memory itself, significantly reducing the need for data transfer between separate memory and processing units. This paradigm shift can lead to substantial improvements in computational speed and energy efficiency, addressing some of the critical limitations of traditional computing systems.

Moreover, memristors have shown great promise in the field of neuromorphic computing, which aims to emulate the neural structures and functionalities of the human brain. Neuromorphic systems leverage the analog nature of memristors to replicate synaptic weights and plasticity observed in biological neural networks. This capability is crucial for developing advanced machine learning algorithms and artificial intelligence applications that require efficient and scalable hardware implementations without von Neumann machines

[5]. Their unique properties and potential applications in neuromorphic computing and in-memory processing make them a focal point of current research efforts. Memristors represent a significant advancement in the field of electronics, offering a versatile and efficient solution for memory storage and computational tasks.

This work focuses on the exploration and development of models for memristors, particularly their current-voltage $(I - V)$ characteristics. The central aim is to analyse and optimise different modelling approaches to accurately capture the electrical behaviour of Si/Ag-based memristor devices in this case.

Overview

This document begins by providing an introduction to memristors, grounded in Chua's circuit theory, and presents the essential theoretical background required to understand the role and behaviour of memristors in electronic circuits in Chapter 1. Following this, various memristor models are explored in depth in Chapter 2. Simulations using LTSpice are conducted, illustrating the behaviour of the device under different circuit configurations in Chapter 3.

The core modelling work, described in Chapter 4, is performed using the GHS model, which is introduced in Chapter 2. Data from multiple cycles are analysed and fitted using this model. The parameter extraction process is key to improving the predictive capabilities of the model.

In addition, this work explores the use of neural network models to approximate the memristor's $I - V$ characteristics, investigating various architectures and training methodologies in Chapter 5.

Finally, conclusions are presented and future work suggested.

1. What is a memristor?

The memristor is a circuit element first postulated by Leon Chua in 1971 [3]. There were three well-known fundamental circuit elements: the resistor, the inductor, and the capacitor. These elements are defined in terms of the relationships between four fundamental circuit variables: voltage (v), current (i), charge (q) and flux-linkage (φ). However, the relationship between charge and flux-linkage was left undefined. So, to fill this gap in circuit theory, a fourth circuit element was proposed. Leon Chua named this circuit element memristor (a contraction of the words memory and resistor) because of the properties it displayed. In 1976, he published a complete circuit theory, introducing a generalisation of his initial memristor [8].

1.1. Chua's circuit theory

A circuit element is a two-terminal device that can be characterised by the set of all its voltage $(v(t))$ and current $(i(t))$ waveform responses to all possible probing circuits (consisting of random interconnections of circuit elements such as resistors, capacitors, inductors, diodes, opamps, transistors, batteries, voltage and current sources) with which we test the circuit element. So, a circuit element is characterized by the list

$$\{(v_1(t), i_1(t)), (v_2(t), i_2(t)), ..., (v_n(t), i_n(t)), ...\} \tag{1.1}$$

where the indexes $1, 2, ..., n, ...$ represent each possible probing circuit [9].

Voltage, v, is a circuit variable that can be measured by a voltmeter, and current, i, is a circuit variable that can be measured with an ammeter. All other circuit variables can be obtained by integrating or differentiating $v(t)$ and $i(t)$ with respect to time. Therefore, we can define flux-linkage and charge as

$$\varphi(t) = v^{-1}(t) = \int_{-\infty}^{t} v(\tau)d\tau \tag{1.2}$$

$$q(t) = i^{-1}(t) = \int_{-\infty}^{t} i(\tau)d\tau \tag{1.3}$$

respectively. The superscript indicates that this is the first time integral of the variable. In the same way, we can define any number of circuit variables [9]

$$..., v^{-n}, ..., v^{-2}, v^{-1}, v, v^1, v^2, ..., v^n, ... \tag{1.4}$$

$$..., i^{-n}, ..., i^{-2}, i^{-1}, i, i^{1}, i^{2}, ..., i^{n}, ... \tag{1.5}$$

where

$$v^{-n} = \int_{-\infty}^{t} \cdots \int_{-\infty}^{\tau_2} v(\tau_1) d\tau_1 \ldots d\tau_n \tag{1.6}$$

$$v^{n} = \frac{d^n v}{dt^n} \tag{1.7}$$

and

$$i^{-n} = \int_{-\infty}^{t} \cdots \int_{-\infty}^{\tau_2} i(\tau_1) d\tau_1 \ldots d\tau_n \tag{1.8}$$

$$i^{n} = \frac{d^n i}{dt^n} \tag{1.9}$$

Every v^k, i^l variable can be obtained from the original v, i variables, therefore, the characterisation of circuit elements can be achieved by a combination of any pairs of these variables [10].

For some circuit elements, we need not know the full list of $v^k(t)$ and $i^l(t)$ waveform responses to characterise them. Basic circuit elements (Figure 1.1) are defined by a relation, f, between the pair of variables. A resistor is defined by

$$f_R\left(v, i\right) = 0 \tag{1.10}$$

(Ohm's Law for linear resistors is an example of such a relation), a capacitor by

$$f_C\left(v, q\right) = 0 \tag{1.11}$$

an inductor by

$$f_L\left(\varphi, i\right) = 0 \tag{1.12}$$

and, finally, an ideal memristor can be defined by

$$f_M\left(\varphi, q\right) = 0 \tag{1.13}$$

Other circuit elements, such as the memductor, $f\left(v^{-2}, q\right) = 0$, and the memcapacitor, $f\left(\varphi, i^{-2}\right) = 0$, can be defined as well.

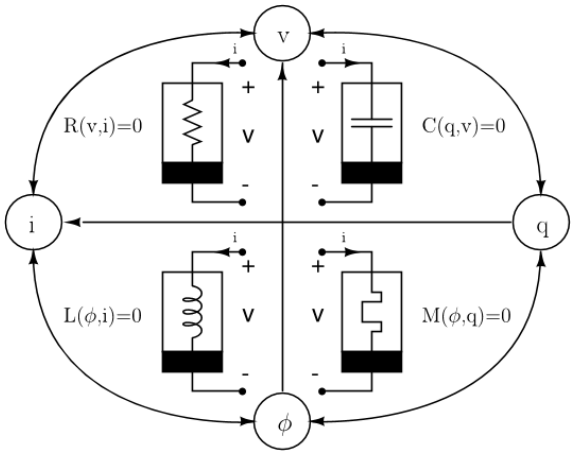


Figure 1.1: The four basic circuit elements and the relations that characterise them (taken from Ref. [11]).

1.2. Generalization of the memristor

In the (v, i) space, the ideal memristor can be perceived as a state-dependent Ohm's Law

$$i = G(\varphi)v \tag{1.14}$$

$$\frac{d\varphi}{dt} = v \tag{1.15}$$

This formulation represents a voltage-controlled memristor, because the state equation is a function of v. The analogous formulation for current-controlled memristors is the following

$$v = R(q)i \tag{1.16}$$

$$\frac{dq}{dt} = i \tag{1.17}$$

In the case of the ideal memristor, the state variable is a circuit variable.

The definition of memristor was generalised in order to encompass a broader set of dynamical systems with similar properties. Based on this generalisation, we can define ideal generic memristors, generic memristors and extended memristors, each of these being a particular case of the following category. The system equations are presented in Table 1.1. From the equations, we can understand that a memristive system always has output 0 when the input is 0. This translates to the characteristic shape of the $I - V$ curve of a memristor: a zero crossing hysteresis loop, often referred to as a pinched hysteresis loop.

Another property that allows us to identify a memristor is its response at high frequencies. When a high enough frequency signal is applied to a memristor, the state variable does not

	Current-controlled	Voltage-controlled
Extended memristor	$v = R(x,i)i, R(x,0) \neq \infty$ $\frac{dx}{dt} = f(x,i)$	$i = G(x,v)v, R(x,0) \neq \infty$ $\frac{dx}{dt} = f(x,v)$
Generic memristor	$v = R(x)i$ $\frac{dx}{dt} = f(x,i)$	$i = G(x)v$ $\frac{dx}{dt} = g(x,v)$
Ideal generic memristor	$v = R(x)i$ $\frac{dx}{dt} = f(x)i$	$i = G(x)v$ $\frac{dx}{dt} = g(x)v$
Ideal memristor	$v = R(q)i$ $\frac{dq}{dt} = i$	$i = G(\varphi)v$ $\frac{d\varphi}{dt} = v$

Table 1.1: Classification of memristors and their equations (adapted from [12]).

have enough time to achieve its limiting states and remains at an intermediate equilibrium state. This state is visible through the $I-V$ curve: at high frequencies, the pinched hysteresis loop collapses to a single-valued function (Figure 1.2). For generic memristors, this function is linear. The shape of the function depends on the initial conditions of the system and the amplitude of the applied signal. When a low-frequency or DC signal is applied, the memristor behaves as a nonlinear resistor with a characteristic DC $I-V$ curve passing through the origin. In the case of an ideal generic memristor, the curve reduces to a single point at the origin [8, 13, 14].

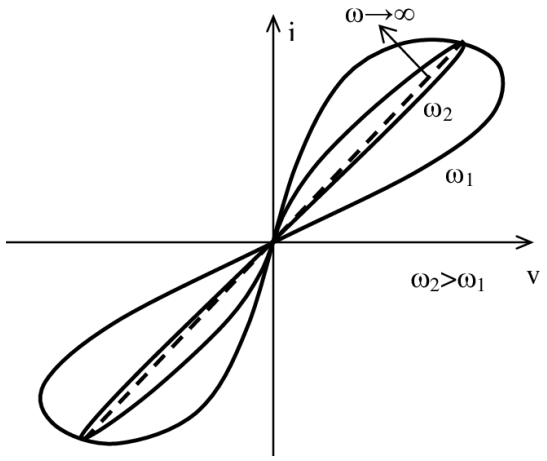


Figure 1.2: Frequency response of a memristor characteristic $I-V$ curve [8].

To accurately describe the behaviour of the memristor device, models adhering to the defined state equations have been developed and will be presented in detail in the following chapter.

2. Memristor models

Memristors present two or more resistance states that are maintained even in the absence of a power source [15]. We will focus predominantly on those that have two resistance states: low resistance state (LRS) (or on state) and high resistance state (HRS) (or off state). The voltage thresholds at which the resistive switching (RS) occurs are generally known as V_{set}, when transitioning from the high resistance state (HRS) to the low resistance state (LRS), and V_{reset}, when transitioning from the LRS to the HRS. Memristors usually have a metal-insulator-metal (MIM) architecture, and the resistive switching (RS) phenomenon occurs due to the formation of conductive filaments (CF) or a phase transition in the insulating layer that lowers its resistivity [16].

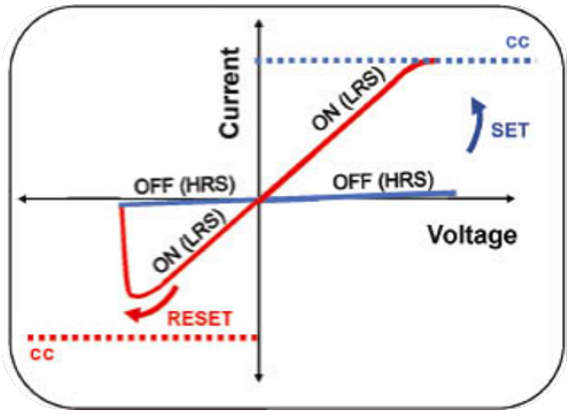


Figure 2.1: Depiction of RS, LRS and HRS. cc represents the current compliance (taken from Ref. [17]).

In order to model the $I-V$ behaviour of memristors, several models obbeying the memristive system equations presented in Table 1.1 have been developed. The models are based on the physical phenomena occurring in the devices, the experimental results, or both. Some models will be presented in the following sections.

2.1. Linear Ion Drift (LID)

The first system to be announced as a memristor was a TiO_x thin film between two Pt electrodes developed by HP Labs. A model obeying the memristor equations that accurately describes the device's behaviour was presented at the same time [4].

In the device presented, the TiO_2 region has oxygen vacancies whose drift controls the resistance of the device (Figure 2.2). The proposed model assumes two states of ohmic

conduction, R_{on} and R_{off}. The conductivity of the device is controlled by a state variable, w, which represents the width of the doped region, in this case, the TiO_{2-x} higher conductivity region. The equations that represent this system are:

$$v(t) = \left(R_{on}\frac{w(t)}{D} + R_{off}\left(1 - \frac{w(t)}{D}\right)\right) i\,(t) \tag{2.1}$$

$$\frac{dw(t)}{dt} = \mu_V \frac{R_{on}}{D} i(t) \tag{2.2}$$

where D is the full length of the TiO_x region and μ_V is the dopant average mobility (in this case, the mobility of oxygen vacancies).

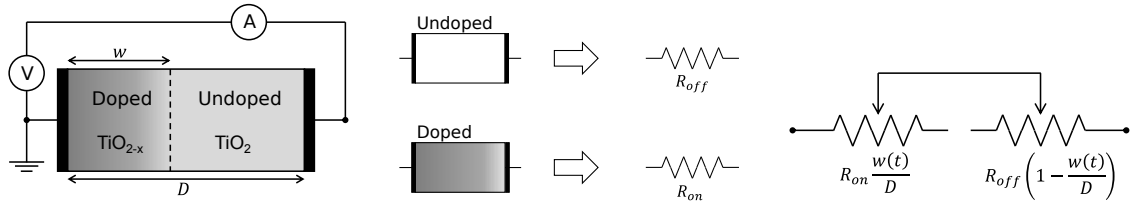


Figure 2.2: HP Labs device model (adapted from [4]).

2.2. Simmons Tunnel Barrier (STB)

STB model was also developed with TiO_x memristors in mind. Unlike the LID model, which consists of two resistors connected in series, STB models a resistor connected in series with an electron tunnel barrier, which is a more accurate physical description of the phenomena happening in the device [18, 19].

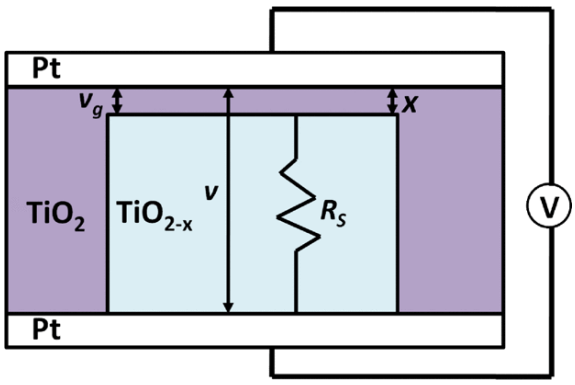


Figure 2.3: Illustration of Simmons Tunnel Barrier model. The state variable, x is the width of the tunnel barrier, v_g is the voltage across the barrier, R_s is the resistance across the TiO_{2-x} region, v is the internal voltage in the device and V is the voltage applied to the device (taken from [18]).

The dynamical equations that describe this model are

$$\frac{dx}{dt} = \begin{cases} f_{off} \sinh\left(\frac{i}{i_{off}}\right) \exp\left[-\exp\left(\frac{x-a_{off}}{w_c} - \frac{|i|}{b}\right) - \frac{x}{w_c}\right] & i > 0 \quad (2.3a) \\ f_{on} \sinh\left(\frac{i}{i_{on}}\right) \exp\left[-\exp\left(-\frac{x-a_{on}}{w_c} - \frac{|i|}{b}\right) - \frac{x}{w_c}\right] & i < 0 \quad (2.3b) \end{cases}$$

where f_{off}, f_{on}, i_{on}, i_{off}, i_{on}, a_{off}, a_{on}, w_c, and b are fitting parameters. The current-voltage relationship is defined in terms of v_g and R_s, by the coupled equations

$$i(t) = \begin{cases} A_g J_0 \left\{ \varphi_I \exp\left(-A\sqrt{\varphi_I}\right) - \left(\varphi_I + ev_g\right) \exp\left[-A\sqrt{\left(\varphi_I + ev_g\right)}\right] \right\} & v_g > 0 \quad (2.4a) \\ A_g J_L \sqrt{\varphi_L} \exp\left(-A\sqrt{\varphi_L}\right) & v_g \simeq 0 \quad (2.4b) \end{cases}$$

$$v_g = v - i(t) R_s \quad (2.5)$$

where A_g is the area of the cross-section of the tunnel barrier, e the charge of the electron,

$$J_0 = \frac{e}{2\pi h (\Delta x)^2} \quad (2.6)$$

$$J_L = \frac{\sqrt{2m}e^2}{\Delta x h^2} \quad (2.7)$$

$$A = \frac{\sqrt{2m}4\pi\Delta x}{h} \quad (2.8)$$

$$\varphi_I = \varphi_0 - \frac{ev_g}{2x}(x_1 + x_2) - \frac{1.15\lambda x}{x_2 - x_1} \cdot \ln\left[\frac{x_2(x - x_1)}{x_1(x - x_2)}\right] \quad (2.9)$$

$$\varphi_L = \varphi_0 - \frac{1.15\lambda x}{x_2 - x_1} \cdot \ln\left[\frac{x_2(x - x_1)}{x_1(x - x_2)}\right] \quad (2.10)$$

$$\lambda = \frac{e^2 \ln 2}{8\pi\epsilon x} \quad (2.11)$$

$$\Delta x = x_2 - x_1 \quad (2.12)$$

where m the mass of the electron, h the Planck's constant, x_1 and x_2 the limits of the barrier at the Fermi level, φ_0 the barrier height in electron volts and ϵ the permitivity of the TiO_2 insulating region.

2.3. Voltage ThrEshold Adaptive Memristor (VTEAM)

VTEAM is the voltage-controlled version of the original ThrEshold Adaptive Memristor (TEAM) model. TEAM came as an improvement of the STB model. In the latter, the

$I-V$ relationship is implicit and the model is computationally less efficient than its successor, which was improved by partially replacing the exponential and hyperbolic sine dependencies with polynomials in the state equation [18].

In the TEAM model, the switching dynamics occur only above or below certain current thresholds (Equation 2.13).

$$\frac{dx}{dt} = \begin{cases} k_{off} \cdot \left(\frac{i(t)}{i_{off}} - 1 \right)^{\alpha_{off}} \cdot f_{off}(x) & 0 < i_{off} < i \quad (2.13a) \\ 0 & i_{on} < i < i_{off} \quad (2.13b) \\ k_{on} \cdot \left(\frac{i(t)}{i_{on}} - 1 \right)^{\alpha_{on}} \cdot f_{on}(x) & i < i_{on} < 0 \quad (2.13c) \end{cases}$$

However, some devices presented a voltage threshold for RS [4, 20, 21] instead of a current threshold. Additionally, a threshold voltage is more suitable than a threshold current for specific logic and memory applications [22, 23]. Similarly to its current-controlled counterpart, the VTEAM state equation is:

$$\frac{dw}{dt} = \begin{cases} k_{off} \cdot \left(\frac{v(t)}{v_{off}} - 1 \right)^{\alpha_{off}} \cdot f_{off}(w) & 0 < v_{off} < v \quad (2.14a) \\ 0 & v_{on} < v < v_{off} \quad (2.14b) \\ k_{on} \cdot \left(\frac{v(t)}{v_{on}} - 1 \right)^{\alpha_{on}} \cdot f_{on}(w) & v < v_{on} < 0 \quad (2.14c) \end{cases}$$

In Equation 2.13 and Equation 2.14, f_{off} and f_{on} represent the dependence of the state equation on the state variable, w, and behave as window functions, limiting the state variable to the interval $\left[w_{on}, w_{off}\right]$. These window functions are chosen for our better convenience, be it the closer fit or the simplest model, and can be different from each other, in case there is an asymmetry to the curve [18, 20]. In these models, the current-voltage relationship is not inherently defined. A linear relationship with w can be assumed, but an exponential dependence on the state variable has been reported [24]. Keeping this in mind, the current-voltage relationship becomes:

$$i(t) = \frac{\exp\left[-\frac{\lambda}{w_{off}-w_{on}} \cdot (w - w_{on})\right]}{R_{on}} \cdot v(t) \quad (2.15)$$

where $\exp(\lambda) = R_{on}/R_{off}$.

2.4. Empirical Chalcogenide Compact Memristor Model (ECCMM)

Pino et al. developed a Ge-Se-Ag-based memristor (Figure 2.4) whose $I-V$ curve was not well represented by the existing and modified ion drift models. They designed a compact empirical model, that does not portray information regarding the physical mechanisms that give rise to the RS phenomenon, but accurately models the behaviour of the device [25].

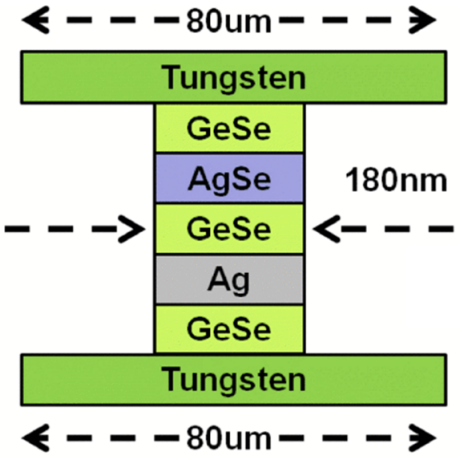


Figure 2.4: Ge-Se-Ag-based memristor (taken from [25])

In this model, the state variable is the memristance, M, which is ruled by the state equation:

$$\frac{dM\left[v\left(t\right)\right]}{dt}=\begin{cases}\begin{cases}K_{l1}\exp\left[K_{l2}\left(v\left(t\right)-T_{l}\right)\right] & M\left[v\left(t\right)\right]<R_{off}\\ 0 & M\left[v\left(t\right)\right]=R_{off}\end{cases} & v\left(t\right)<T_{l}\\ 0 & T_{l}\leq v\left(t\right)\leq T_{h}\\ \begin{cases}-K_{h1}\exp\left[K_{h2}\left(v\left(t\right)-T_{h}\right)\right] & M\left[v\left(t\right)\right]>R_{on}\\ 0 & M\left[v\left(t\right)\right]=R_{on}\end{cases} & v\left(t\right)>T_{h}\end{cases} \tag{2.16}$$

where T_l and T_h are the voltage thresholds to enter the LRS and the HRS respectively, R_{on} and R_{off} are the minimum and maximum resistances achieved by the device, and K_{l1}, K_{l2}, K_{h1} and K_{h2} are fitting parameters that capture the nonlinear effects.

2.5. General Hyperbolic Sine (GHS)

The GHS is another empirical model. The model was first presented in 2011 [26] in its exclusive hyperbolic sinusoidal form which was tested against several devices [27] and

was later updated to include a linear region and tested against a tantalum oxide memristor [28].

The first model's $I-V$ curve consists of two hyperbolic sine branches

$$i(t) = \begin{cases} a_1 x(t) \sinh\left(bv(t)\right) & v(t) \geq 0 \quad (2.17a) \\ a_2 x(t) \sinh\left(bv(t)\right) & v(t) < 0 \quad (2.17b) \end{cases}$$

The distinction between branches exists to account for the $I-V$ curve asymmetry with respect to the voltage polarity. This is modelled by a hyperbolic sine function because the memristor is considered as a metal-insulator-metal (MIM) junction [27, 29].

In the updated model, there is no voltage polarity dependence; however, it is more similar to the previously presented models, in that it is a mixture of two states of resistance. The $I-V$ curve relationship is given by

$$i(t) = h_1\left(v(t)\right) x(t) + h_2\left(v(t)\right)\left(1 - x(t)\right) \quad (2.18)$$

where h_1 and h_2 are appropriately chosen functions that model the two types of resistive behaviour. In this work, we will consider ohmic and MIM conduction, therefore

$$h_1 = \sigma v(t) \quad (2.19)$$

and

$$h_2 = \gamma \sinh\left(bv(t)\right) \quad (2.20)$$

The dynamical equation that rules the state variable evolution is

$$\frac{dx}{dt} = \eta g\left(v(t)\right) f\left(x(t)\right) \quad (2.21)$$

This is a threshold model, where a voltage threshold must be surpassed for the conductance state to change. The function

$$g\left(v(t)\right) = \begin{cases} A_p\left(e^{v(t)} - e^{V_p}\right) & v(t) > V_p \quad (2.22a) \\ -A_n\left(e^{v(t)} - e^{Vn}\right) & v(t) < -V_n \quad (2.22b) \\ 0 & -V_n \leq v(t) \leq V_p \quad (2.22c) \end{cases}$$

implements this mechanism. The parameters A_p and A_n, which describe the rate at which the state shifts once the threshold is crossed, can be tuned. The function

$$f\left(x\left(t\right)\right)=\begin{cases}\begin{cases}e^{\left(x_p-x\right)}w_p\left(x,x_p\right) & x\geq x_p\\ 1 & x<x_p\end{cases} & \eta v\left(t\right)>0\\ \begin{cases}e^{\left(x+x_n-1\right)}w_n\left(x,x_n\right) & x\leq 1-x_n\\ 1 & x>1-x_n\end{cases} & \eta v\left(t\right)<0\end{cases} \tag{2.23}$$

captures the nonlinear ion motion. Moreover, it models the resistance to change of the state variable as it approaches its boundaries $\left(x\in\left[0,1\right]\right)$. This function enables the modelling of the motion of the state variable to vary according to the polarity of the input voltage. The state of the device according to the voltage polarity is controlled by η. The windowing functions

$$w_p\left(x\left(t\right),x_p\right)=\frac{x_p-x\left(t\right)}{1-x_p}+1 \tag{2.24}$$

$$w_n\left(x\left(t\right),x_n\right)=\frac{x\left(t\right)}{1-x_n} \tag{2.25}$$

guarantee that dx/dt equals 0 when $x\left(t\right)=1$ and prevent $x\left(t\right)$ from taking negative values when the current flow changes direction, respectively.

3. LTspice simulations

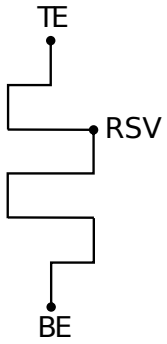
Before delving into the modelling of $I - V$ provided in Chapter 4, this chapter introduces LTspice simulations based on the ECCMM to offer an initial practical understanding of memristive behaviour providing easy parameter variation. This model was selected for its straightforward behavior under voltage control, allowing the memristor to switch from an off state (1200 Ω) to an on state (160 Ω) when the applied voltage exceeds a certain threshold.

The memristor was described as a SPICE subcircuit (Code Snippet 3.1) and simulated using parameters $R_{on} = 160\,\Omega$, $R_{off} = 1200\,\Omega$, $T_h = 0.2\,\mathrm{V}$, $T_l = -0.35\,\mathrm{V}$, $K_{h1} = 5.5 \times 10^6$ $\Omega\,\mathrm{s}^{-1}$, $K_{h2} = -20\,\mathrm{V}^{-1}$, $K_{l1} = 4 \times 10^6\,\Omega\,\mathrm{s}^{-1}$ and $K_{l2} = 20\,\mathrm{V}^{-1}$ for the ECCMM. The memristor (Figure 3.1a), in this SPICE implementation, is represented by a current source that connects the top and bottom electrodes. The generated current equals the current obtained by Ohm's law for the voltage applied between the two terminals and the calculated present resistance value (Figure 3.1b). The resistance, which is the state variable in the ECCMM, is represented by the voltage in a third terminal (RSV). This third terminal does not exist in real memristors and is only added in the simulation so we can access the state variable, which is always hidden in the real world. The function representing the evolution of the resistance (right-hand side of Equation 2.16) is implemented and generated as a current, resorting to a current source, and subsequently integrated using a capacitor of capacitance 1 F. This subcircuit returns a voltage, which is equal, in value, to the resistance at each given time (Figure 3.1c). The initial resistance state is R_{off} [30, 31].

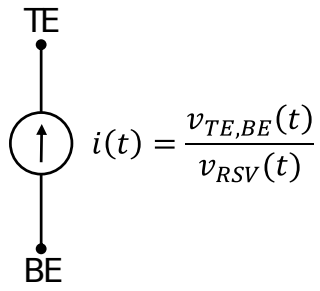
```
* Memristor Model proposed by Pino et al

* Connections:
* TE: Top electrode
* BE: Bottom electrode
* RSV: External connection to plot state variable that is not used otherwise

.SUBCKT MEM_PINO TE BE RSV

* Ron
* Roff
* Th: Positive voltage threshold
* Tl: Negative voltage threshold
* Kh1, Kh2: Fitting params for pos voltage
* Kl1, Kl2: Fitting params for neg voltage
```

```
.params Ron=160 Roff=1200 Th=.2 Tl=-0.35 Kh1=5.5e6 Kh2=-20 Kl1=4e6 Kl2=20

* Fits the change in resistance to characterization data
.func Rt(V1,V2) = IF( V1 <= Th, IF(V1 >= Tl,0,IF(V2 < Roff, Kl1*exp(Kl2*(V1-Tl)),0)), IF
    (V2 > Ron, -Kh1*exp(Kh2*(V1-Th)),0))

* Circuit to integrate to find resistance
Gx 0 RSV value={Rt(V(TE,BE),V(RSV))}
Cx RSV 0 {1}
.ic V(RSV) = Roff

* Current source representing memristor
Gmem TE BE value={V(TE,BE)/V(RSV)}

.ENDS MEM_PINO
```

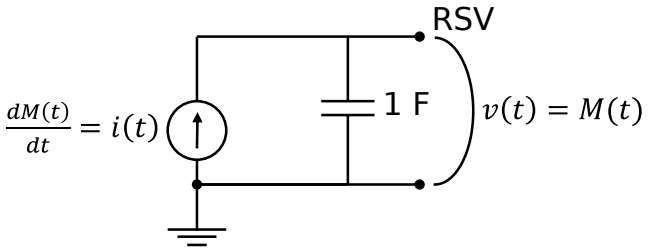
Code Snippet 3.1: Subcircuit definition of memristor used in the simulations



(a) Memristor with top electrode (TE), bottom electrode (BE) and auxiliary resistance state variable terminal (RSV)



(b) Current source that represents the memristor's response



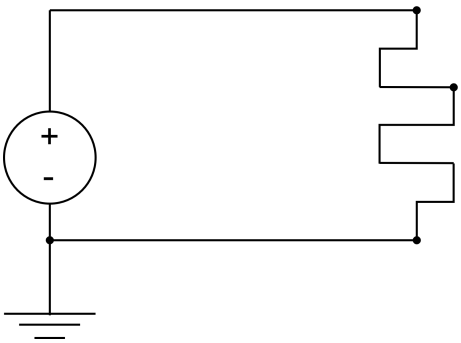
(c) Auxiliary integrating circuit

Figure 3.1: Representation of the memristor modelled in LTspice

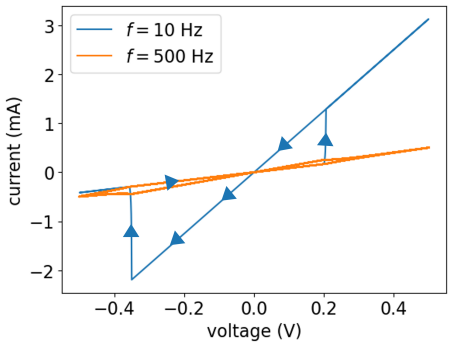
3.1. Voltage source and memristor circuit

The first simulation was performed with the previously described memristor connected to a voltage source. The source generated a sinusoidal voltage with an amplitude of 0.5 V and a frequency of 10 Hz, for the first simulation, and 500 Hz, for the second. The circuit

was simulated for 0.1 s. We can see the characteristic pinched hysteresis loop and its collapse at high frequencies (Figure 3.2).



(a) Memristor with the top electrode (TE), bottom electrode (BE) and auxiliary resistance state variable terminal (RSV)



(b) Current source that represents the memristor's response

Figure 3.2: Voltage source and memristor simulation.

3.2. Memristor network

A 4×4 memristor network was simulated (Figure 3.3) [32]. This circuit was simulated to demonstrate the network's working principle. The state of each memristor in this network is determined by the signals applied by voltage sources $V_{\alpha m}$ and $V_{\beta n}$ as well as the state of other memristors. Each memristor can be identified by a pair (m, n). In the simulation, two pulses of 150 mV were sent by voltage source $V_{\alpha 4}$. The pulses were sent at 2 ms and lasted 1 ms each, with an interval of 1 ms between them. The voltage source $V_{\beta 1}$ sent one 1 ms pulse at 2 ms (Figure 3.4). The remaining voltage sources stayed at 0 V. 50 ms were simulated. The combination of the pulses put a voltage of 300 mV across the terminals of the memristor $(4, 1)$, which is above the 200 mV threshold for it to switch to the on state. The second pulse sent by $V_{\alpha 4}$, as it was unmatched, acted as a reading pulse. Through it we can verify that the memristor is in a low resistance state, since the current going through it is higher, without changing the state of the memristor, given that the voltage across its terminals is below the positive threshold, T_h, and above the negative threshold, T_l, for RS. The memristor $(4, 2)$ only has a voltage of 150 mV across its terminal, which is below the T_h threshold; therefore, both pulses sent by $V_{\alpha 4}$ were felt as reading pulses by this memristor. In both instances, we understand that the memristor is in the off state.

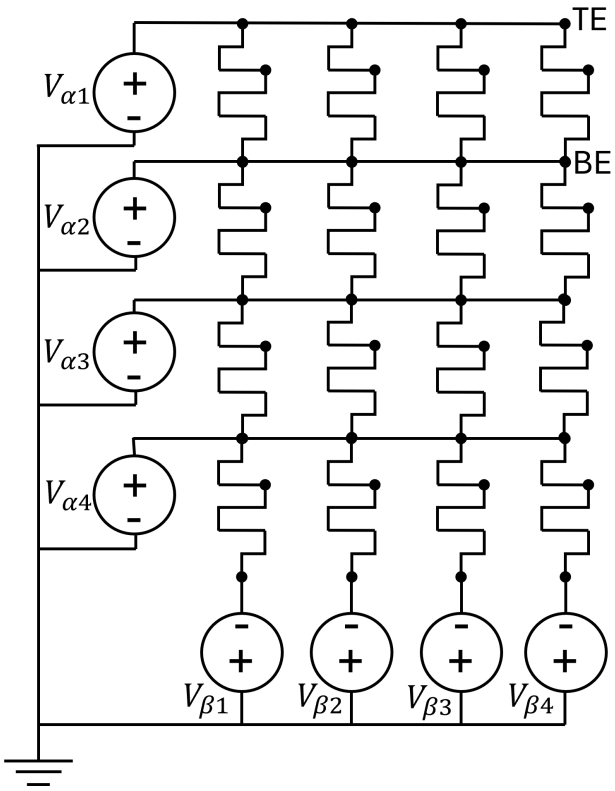


Figure 3.3: Memristor network simulated in LTspice. All memristor are represented with the top electrode up and the bottom electrode down.

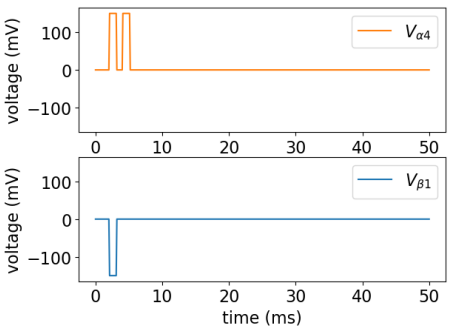


Figure 3.4: Signals applied by voltage sources $V_{\alpha 4}$ and $V_{\beta 1}$

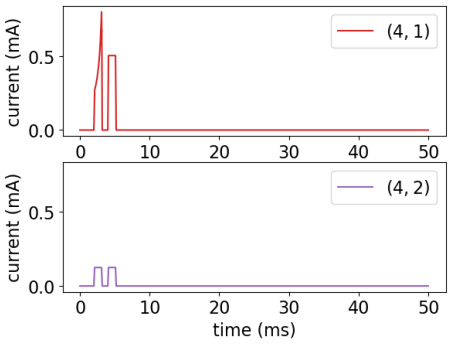


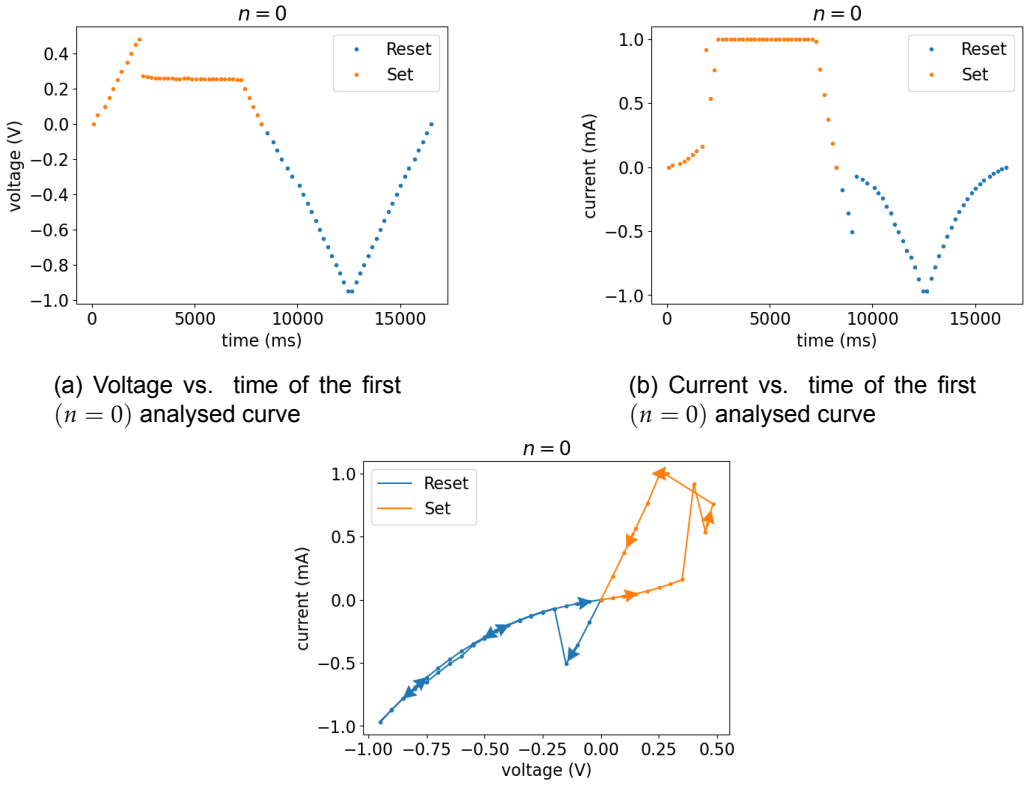
Figure 3.5: Response of memristors $(4, 1)$ and $(4, 2)$

4. Modelling memristor's $I - V$ curves using the GHS model

In this work, previously acquired data regarding cyclic $I - V$ characterisation of Pt (150) / Si (20) / Ag (5) / TiW (100) (nm) stack memristor devices [33] were used and fit on the GHS updated model. Data from 100 consecutive cycles of a single memristor device upon voltage sweeps of $0\ \mathrm{V} \rightarrow 1\ \mathrm{V} \rightarrow -0.95\ \mathrm{V} \rightarrow 0\ \mathrm{V}$ were used.

4.1. Data representation

Current and voltage measurements in time were represented, as well as the $I - V$ curve (Figure 4.1).



(a) Voltage vs. time of the first $(n = 0)$ analysed curve

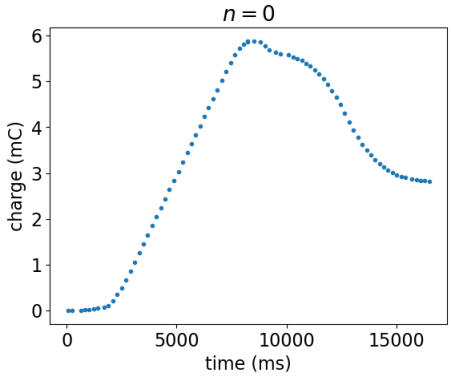
(b) Current vs. time of the first $(n = 0)$ analysed curve

(c) Current vs. voltage of the first $(n = 0)$ analysed curve

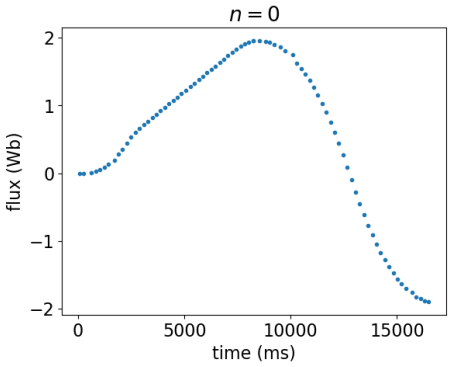
Figure 4.1: Representation of the data for $n = 0$. Set corresponds to a positive voltage applied and Reset to a negative voltage.

The saturation of the current and the voltage drop happen due to the application of a current compliance of $1\ \mathrm{mA}$. The voltage drop occurs at the RS set point, where the resistance decreases from $635.4\ \Omega$ to $273.8\ \Omega$, so the limitation in current forces the reduction.

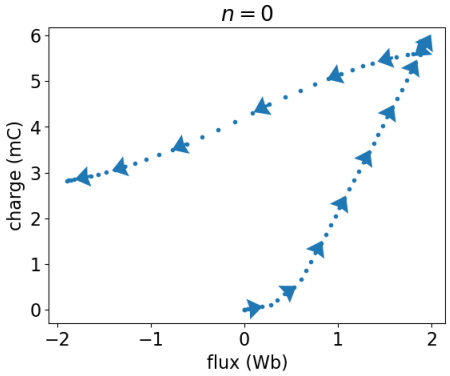
$i(t)$ and $v(t)$ were integrated, using the trapezoidal rule, to represent $q(t)$ and $\varphi(t)$ (Figure 4.2). In this representation, we can verify that this device does not obey $d\varphi = Mdq$ [4], i.e., the relationship between q and φ is not linear; there are two different branches, one for each resistance state. We can also see that the $q-\varphi$ space is not the most appropriate to describe the state of the device, as it does not display a closed trajectory, contrary to the device, that returns approximately to its original state in terms of resistive behaviour. We can observe that the open trajectory derives from the application of a current compliance, as this restriction translates to asymmetrical signals between the positive and negative regions, both in current and voltage. This causes an accumulation of charge in the device and a decrease in flux.



(a) Charge vs. time of the first $(n=0)$ analysed curve



(b) Flux-linkage vs. time of the first $(n=0)$ analysed curve

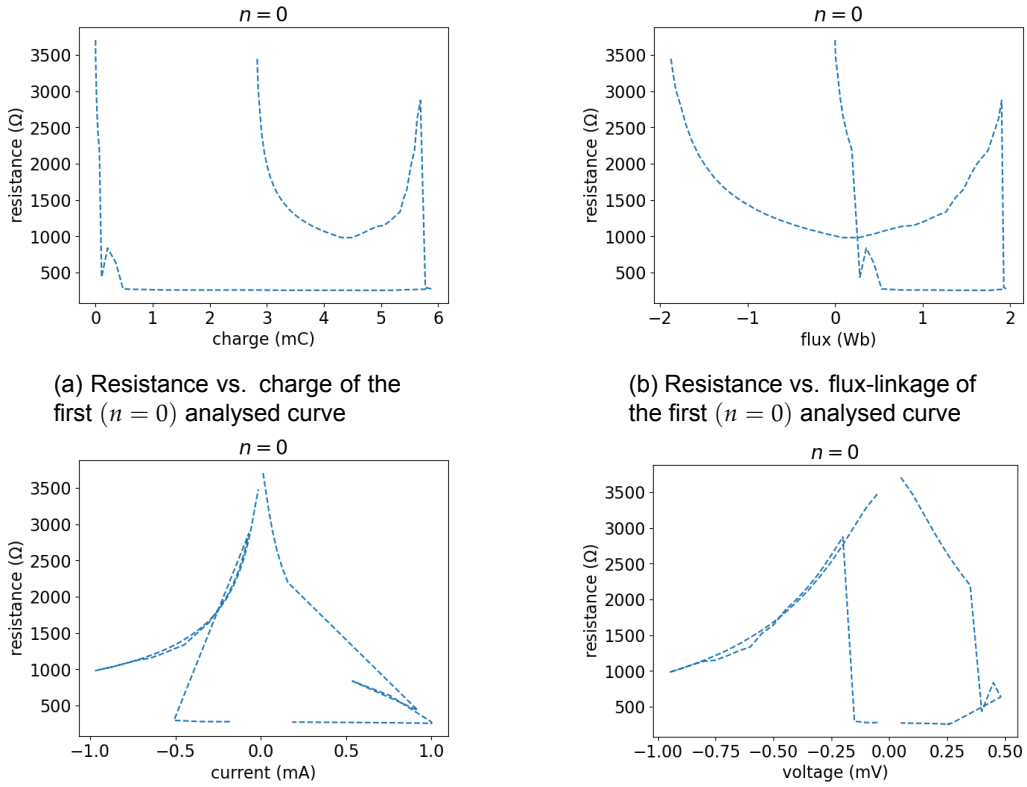


(c) Charge vs. flux-linkage of the first $(n=0)$ analysed curve

Figure 4.2: Representation of the integral of the original data for $n=0$.

The resistance dependencies with i, v, q and φ were also plotted to verify any relationship between resistance and any of these variables. For an ideal memristor, it is expected to find resistance as a function of charge, $R = f\left(q(t)\right)$, for current-controlled memristors, or resistance as a function of flux-linkage, $R = g\left(\varphi(t)\right)$, for voltage-controlled memristors

[12]. No relationship was found (Figure 4.3), which leads to the conclusion that this device cannot be modelled by the ideal memristor equations.



(a) Resistance vs. charge of the first $(n = 0)$ analysed curve

(b) Resistance vs. flux-linkage of the first $(n = 0)$ analysed curve

(c) Resistance vs. current of the first $(n = 0)$ analysed curve

(d) Resistance vs. voltage of the first $(n = 0)$ analysed curve

Figure 4.3: Representation of the resistance as a function of the circuit variables q, φ, i and v for $n = 0$.

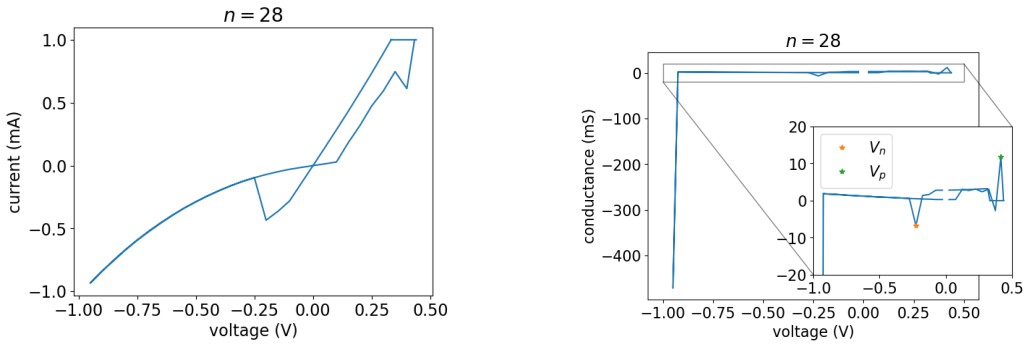
4.2. Parameter extraction

The updated GHS model was fitted to the data. This model was chosen because it details algorithms for parameter extraction [28]. Furthermore, the data present a linear ohmic conduction region as can be seen in Figure 4.1c, when the voltage descends from $0.20\ \mathrm{V}$ to $-0.15\ \mathrm{V}$, which lines with one of the functions that can be used in the model.

This requires the extraction of nine parameters from the experimental curves: $V_p = V_{set}$, $V_n = V_{reset}$, $g_{pk,p}$, $g_{pk,n}$, g_{max}, g_{min}, $g_{slow,p}$, $g_{slow,n}$ and b. Concurrently, it needs two supplementary parameters that inform of the position where the state transition stops.

The first parameters to be extracted were V_p and V_n. For that, the maximum and minimum of the di/dv derivative, which can be interpreted as conductance, needed to be found. This model assumes jumps between conductance states, so, during the set operation, it would be expected that $di/dv = +\infty$ and during the reset operation $di/dv = -\infty$.

An approximation of the conductance, di/dv, was computed. The maximum and minimum of this approximation correspond to the discontinuity points that would be expected if this was an ideal, continuous curve that behaved accordingly to the model. The approximation had invalid values (*inf* and *NaN*) coming from consecutive points with equal voltage and current, so the array containing it was masked before operations were performed on it. Points with close $I-V$ coordinates gave rise to outliers of the derivative: they were not the extremes of the conductance we were trying to find (Figure 4.4, the desired extremes, which correspond to V_{set} and V_{reset} easily identifiable in the experimental data $I-V$ curve, are visible on the inset). Therefore, any points higher (lower) than 100 times the interquartile range of the third (first) quantile were removed from the di/dv curve.



(a) Current vs. voltage $(n = 28)$

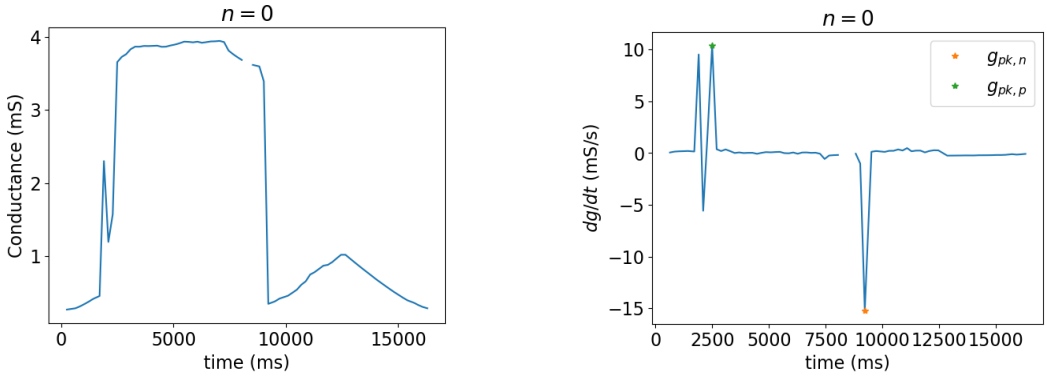
(b) Conductance vs. voltage $(n = 28)$

Figure 4.4: Example of curve with conductance outliers $(n = 28)$.

The maximum conductance was found for the positive voltage half, and the minimum for the negative voltage half, which were considered the V_{set} and V_{reset} points. To assess the quality of the automatically chosen values, 10 out of the 100 curves were selected at random. The selected curves corresponded to $n = 8, 15, 28, 31, 56, 62, 63, 73, 78, 98$. From here on, those curves will be designated as *testing curves*. From those curves, V_{set} and V_{reset} were manually extracted and chosen to be the voltage point at which the system seems to be starting the state transition. Using the manually extracted values as the true values and comparing them against the automatically extracted ones, we get a percentage error of 75.02% for V_{set} and of 48.48% for V_{reset}. To improve this, additional criteria were added. di/dv did, sometimes, present a maximum after V_{set}. As is visible in Figure 4.4, the conductance suffers a fluctuation, which leads to a greater change in conductance at a voltage higher than V_{set}. To address this problem, V_{set} was considered the voltage at which the first peak with a prominence of a fraction of the peak-to-peak value (excluding the outliers) of di/dv. To choose the fraction value, several values were

tried and tested with the *testing curves*. Upon visual inspection of the results, the value of $1/8$ was chosen, only failing at *testing curve* $n = 28$. The values of $1/4$, $1/8$, $1/16$ and $1/32$ were tested. V_{reset} marks the end of the linear region. At the linear region, the current drop for each voltage step is, in general, greater than 0.1 mA. V_{reset} was then considered to be the voltage at which the current decrease was less than 0.1 mA for two consecutive points. The use of two consecutive points serves to account for fluctuations that should be disregarded. Using these new criteria, the percentage error on the *testing curves*, considering the manually extracted values as true values, becomes 16.07% for V_{set} and 22.42% for V_{reset}.

An approximation of the time derivative of conductance, dg/dt was calculated. The maximum and minimum of dg/dt were taken to get the $g_{pk,p}$ and $g_{pk,n}$ parameters. It is worth mentioning that, for this model, these parameters take units of $\mathrm{mS\,s^{-1}}$.



(a) Conductance vs. time $(n = 0)$

(b) dg/dt vs. time $(n = 0)$

Figure 4.5: Conductance and variation in conductance over time $(n = 0)$.

The extraction of parameters $g_{slow,p}$ and $g_{slow,n}$ was not as straightforward. These correspond to more stable conductance values after the abrupt change signalled by $g_{pk,p}$ and $g_{pk,n}$. The criterion used to find these values was the presence of two consecutive points in time that distanced in conductance less than $1/40$ of the peak-to-peak value of conductance. This criterion was used because it was the most straightforward way to extract $g_{slow,p}$ and $g_{slow,n}$ given the present dataset.

For this model, an adjustment for a linear and a hyperbolic sine function needs to be made. For that, the linear and hyperbolic sine sections of the $I-V$ curve need to be delimited. V_{set} and V_{reset} parameters mark the ending of the hyperbolic sinusoidal and linear region, respectively. There is still a necessity for other two parameters that indicate the beginning of said regions.

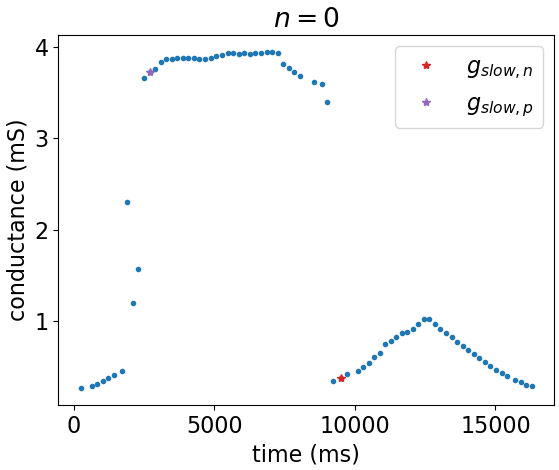
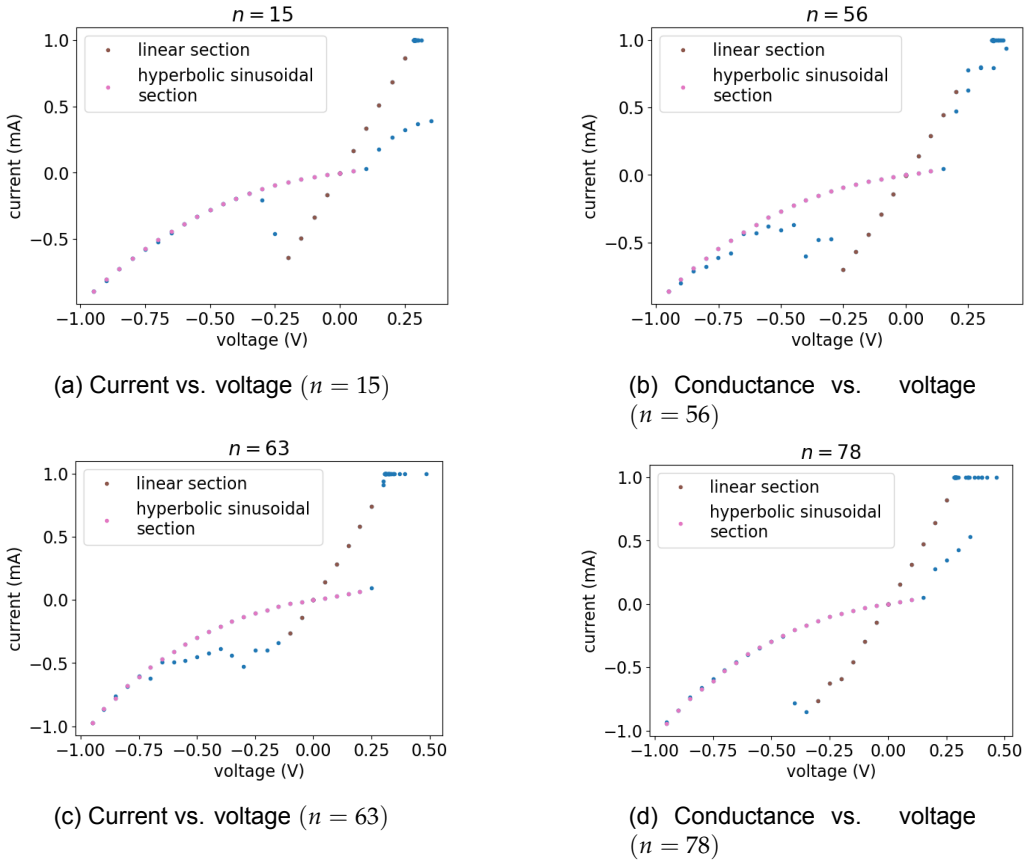


Figure 4.6: Conductance vs. time with illustration of parameters $g_{slow,p}$ and $g_{slow,n}$

An appropriate criterion to identify the beginning of the linear region was searched. The first one used was to find the voltage corresponding to the minimum of di/dv in the positive voltage region. Upon visual inspection of the *testing curves*, it was found that this criterion was not appropriate, as it is sensitive to fluctuations, in addition to considering the points corresponding to the current compliance part of the linear region. The presence of the current compliance helped define a second criterion: there should be a current drop greater than 0.1 mA, approximately $1/20$ of the peak-to-peak value of the $I-V$ curve. Using these two criteria, it fails on *testing curves* $n = 28, 56$. An additional criterion is needed. The direction of the voltage sweep is taken into account: in addition to the previous criteria, the voltage needs to be decreasing. This still fails for *testing curve* $n = 56$, which asks for a fluctuation-proof criterion, so we choose to have two consecutive points of descending voltage instead of only one.

The hyperbolic sinusoidal region was taken to start at the voltage minimum. Other criteria were tried, they were, however, less accurate. An approach similar to that used to extract V_{reset}, V_{set} and the beginning of the linear region was tried: to use a maximum of the derivative in the negative voltage region. Another criterion experimented with was using the voltage at which minimum current was achieved, however, this method failed for *testing curve* $n = 62$. The voltage minimum was the most expedited and reliable criterion.

With the sections isolated, it is possible to proceed with extracting g_{max}, g_{min} and b. Equation 2.19 was fitted to the linear section, extracting $g_{max} = \sigma$ and Equation 2.20 was fitted to the hyperbolic sinusoidal section, extracting $g_{min} = \gamma$ and b for each of the 100 curves.



(a) Current vs. voltage $(n = 15)$

(b) Conductance vs. voltage $(n = 56)$

(c) Current vs. voltage $(n = 63)$

(d) Conductance vs. voltage $(n = 78)$

Figure 4.7: Linear and hyperbolic sinusoidal sections automatically extracted from *testing curves* $(n = 15, 56, 63, 78)$.

g_{max} and g_{min} are needed to calculate x_p, x_n, A_p and A_n

$$x_p = \frac{g_{slow,p} - g_{min}}{g_{max} - g_{min}} \tag{4.1}$$

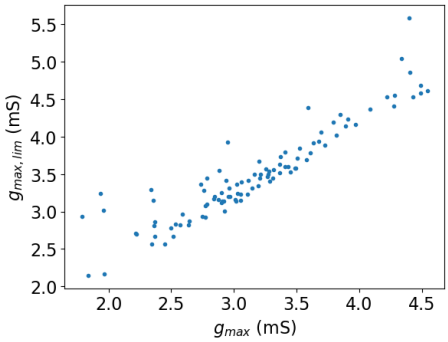
$$x_n = \frac{g_{slow,n} - g_{min}}{g_{max} - g_{min}} \tag{4.2}$$

$$A_p = \frac{g_{pk,p}}{g_{max} - g_{min}} \tag{4.3}$$

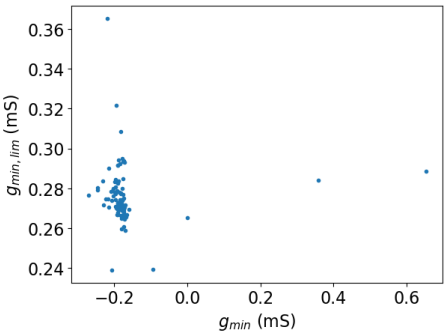
$$A_n = \frac{g_{pk,n}}{g_{max} - g_{min}} \tag{4.4}$$

In this case, g_{max} and g_{min} are used as a normalisation, so that x_p, x_n, A_p and A_n take a value $\in [0, 1]$. However, the g_{max} and g_{min} values obtained through the fit do not exactly correspond to the maximum and minimum conductance values. For g_{max}, the values obtained through the fit closely match the maximum value of conductivity (Figure 4.8a), while for g_{min} the same does not happen (Figure 4.8b). To achieve correct normalisation, the maximum and minimum conductivity values, $g_{max,lim}$ and $g_{min,lim}$, were used, and replaced g_{max} and g_{min} in equations 4.1-4.4. For the adjustments, the originally determined g_{max} and g_{min} were used, as they were specially optimised for that end. This results in

two additional parameters, raising the total number of parameters necessary to implement this model to 13.



(a) Comparison between maximum conductance and g_{max} extracted form the linear fit



(b) Comparison between minimum conductance and g_{min} extracted form the hyperbolic sinusoidal fit

Figure 4.8: Comparison of parameters g_{max} and g_{min} extracted in different ways.

Parameter	Average	Standard Deviation	Pearson correlation coefficients of $Q - Q$ plot
V_{set} (V)	0.26	0.08	0.9791
V_{reset} (V)	-0.25	0.10	0.9631
$g_{pk,p}$ ($\mathrm{mS\,s^{-1}}$)	7.0	2.8	0.9870
$g_{pk,n}$ ($\mathrm{mS\,s^{-1}}$)	-6.9	3.5	0.9738
$g_{slow,p}$ (mS)	3.0	0.6	0.9703
$g_{slow,n}$ (mS)	0.76	0.36	0.8226
g_{max} (mS)	3.2	0.6	0.9925
g_{min} (mS)	-0.17	0.10	0.5205
b ($\mathrm{V^{-1}}$)	-2.4	0.6	0.5501
$g_{max,lim}$ (mS)	3.5	0.6	0.9805
$g_{min,lim}$ (mS)	0.28	0.01	0.8583
x_p	0.84	0.14	0.9256
x_n	0.16	0.12	0.8600
A_p ($\mathrm{s^{-1}}$)	2.2	0.8	0.9924
A_n ($\mathrm{s^{-1}}$)	-2.1	0.9	0.9797

Table 4.1: Extracted parameters statistics for the 100 cycles.

The evolution of the parameters over time (more specifically, the number of cycles to which the device was subjected) was plotted in order to evaluate whether there was a correlation. No correlation was found for any of the parameters: their cycle variation presented random. The variation could be described by a normal distribution, for V_{set}, V_{reset}, $g_{pk,p}$, $g_{pk,n}$, $g_{slow,p}$, g_{max}, $g_{max,lim}$, A_p andA_n. This assumption was supported by the $Q - Q$ plots and their Pearson correlation coefficients (Table 4.1). The probability distribution of V_{set} and V_{reset} displays a discretisation because of the applied voltage step

in the experimental acquisition of the data. $g_{slow,n}$, g_{min}, b, $g_{min,lim}$, x_p and x_n could not be assumed to have a normal distribution (Figure 4.9-Figure 4.21) [34].

The predictions of this application of the GHS model present a mean absolute percentage error of 26% in relation to the experimental data for the *testing curves*. The most significant discrepancy can be observed in the state transition regions, about V_{set} and V_{reset} (Figure 4.22). The GHS model assumes a much more abrupt transition. This aligns well with the data presented in [28], however, the experimental data used in this work presents a smoother transition. The smoother transition presented by the experimental data is due to the gradual formation and dissolution of the conductive filaments (CF). As the filaments form, the device enters an intermediate state of conductance and only achieves the LRS when the filament is completely formed. The same happens with the dissolution of the filament: it is a gradual process that gives rise to an intermediate state. The GHS model only presents smooth transitions when the HRS and LRS are not fully reached and the state of the device is a combination of both. This model is equipped to deal with faster transitions than the ones presented by the data. If we try to tune the parameter which modulates the transition velocity to make it more compatible with the transition displayed by the experimental data, we do not reach the fully ohmic state of conduction in the model and therefore the ohmic region is not correctly modelled. Furthermore, the intermediate state of the device is not necessarily a mixture of the two conduction states used in the model, which comprises an additional difficulty to predict this state.

The algorithm for extracting the parameters for the GHS model is neither optimised for real curves nor fully automated. It is not prepared to handle the presence of a current compliance. Moreover, it is highly sensitive to noise in the experimental data, particularly concerning the extraction of $g_{pk,p}$ and $g_{pk,n}$, where noise translates to extreme values of the conductance derivative that do not correspond to the values intended to be extracted. The noise cannot be filtered using traditional signal processing techniques, as a low-pass filter, because of the nature of the signal: it has a steep transition that is needed for the model and using such a filter would eliminate the points corresponding to the intermediate state during the transition. Furthermore, the adjustment of functions $h_1(v)$ and $h_2(v)$ (Equation 2.19-2.20) demands the isolation of the points used for each adjustment. The isolation can be performed manually or by resorting to additional parameters.

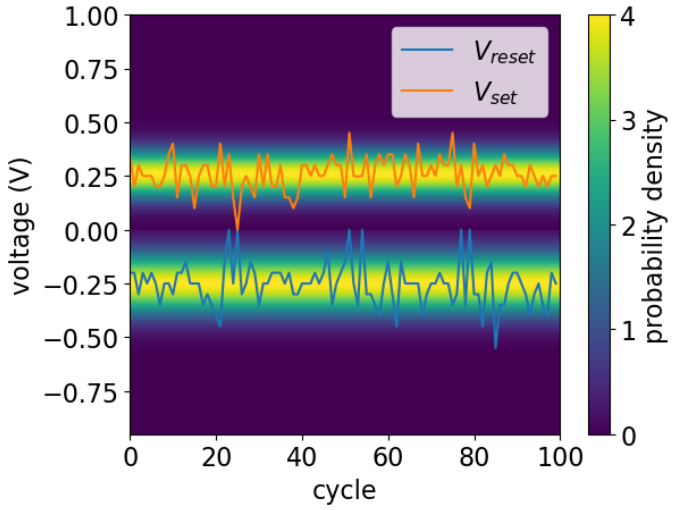
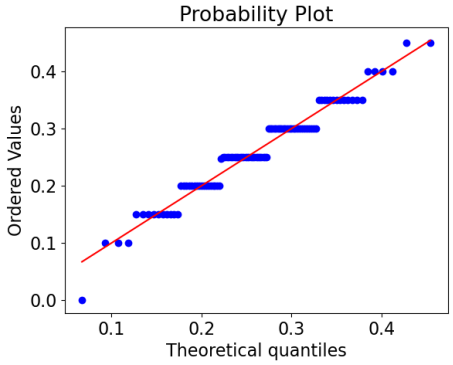
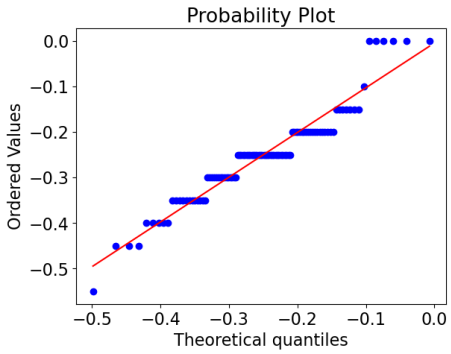


Figure 4.9: V_{set} and V_{reset} distributions.



(a) $Q - Q$ plot of V_{set} against normal distribution $(y = mx + b; m = 0.9828; b = -0.0043)$



(b) $Q - Q$ plot of V_{reset} against normal distribution $(y = mx + b; m = 1.0009; b = -0.0003)$

Figure 4.10: $Q - Q$ plots of V_{set} and V_{reset}.

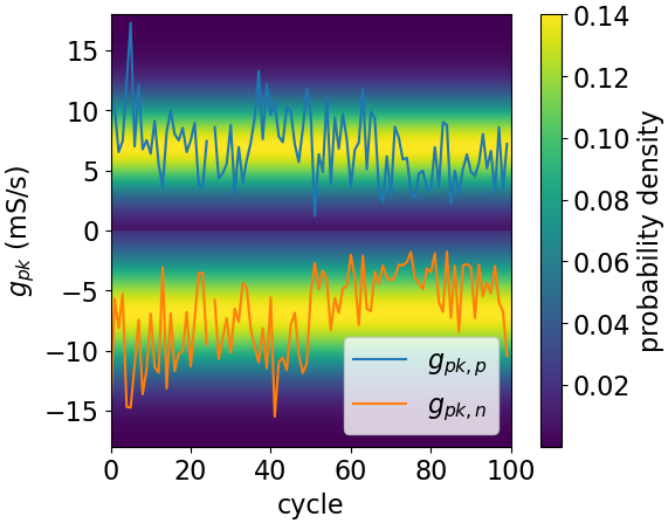
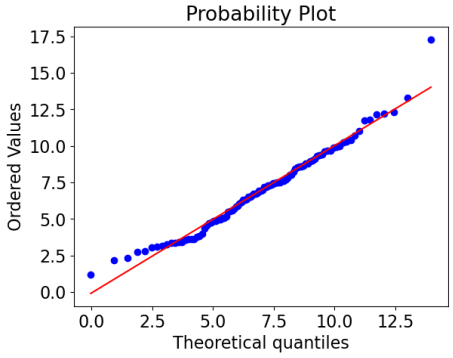
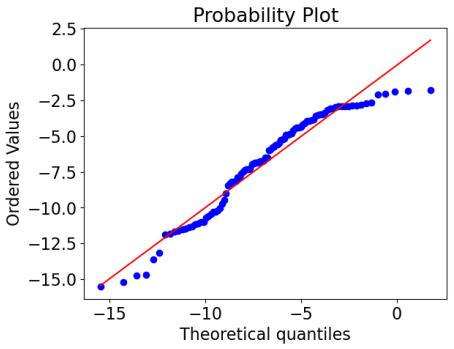


Figure 4.11: $g_{pk,p}$ and $g_{pk,n}$ distributions.

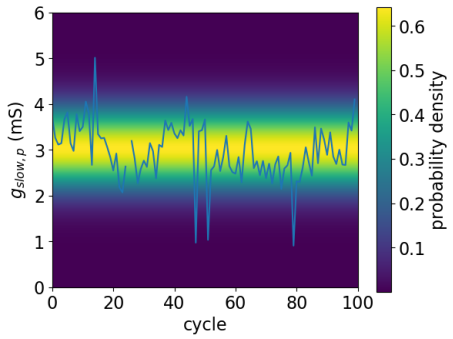


(a) $Q - Q$ plot of $g_{pk,p}$ against normal distribution $(y = mx + b; m = 1.0073; b = -0.0508)$

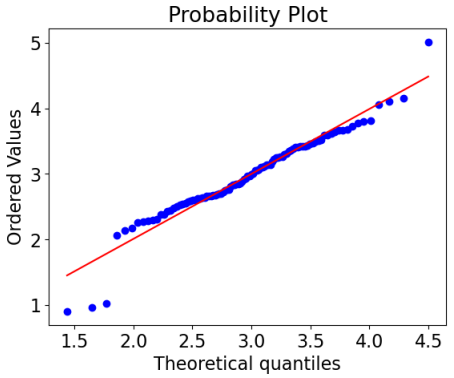


(b) $Q - Q$ plot of $g_{pk,n}$ against normal distribution $(y = mx + b; m = 0.9938; b = -0.0423)$

Figure 4.12: $Q - Q$ plots of $g_{pk,p}$ and $g_{pk,n}$.

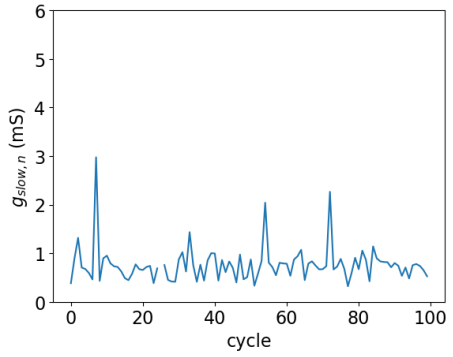


(a) $g_{slow,p}$ over the 100 cycles

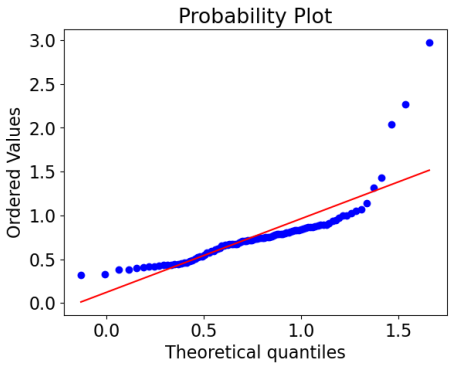


(b) $Q - Q$ plot of $g_{slow,p}$ against normal distribution $(y = mx + b; m = 0.9903; b = 0.0289)$

Figure 4.13: $g_{slow,p}$ distribution.

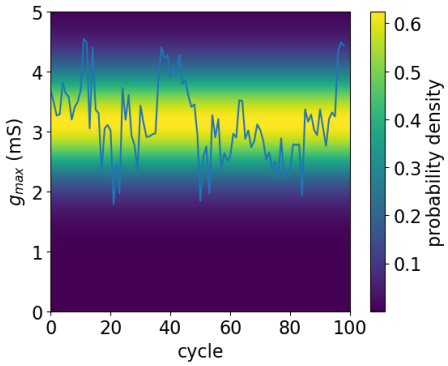


(a) $g_{slow,n}$ over the 100 cycles

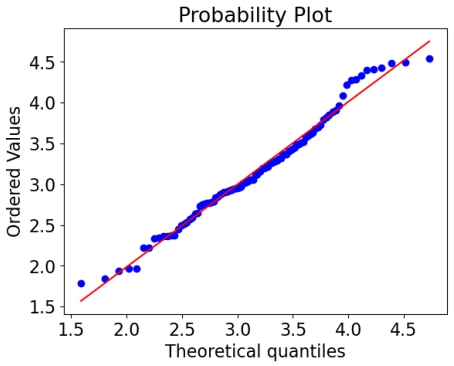


(b) $Q - Q$ plot of $g_{slow,n}$ against normal distribution $(y = mx + b; m = 0.8395; b = 0.1225)$

Figure 4.14: $g_{slow,n}$ distribution.

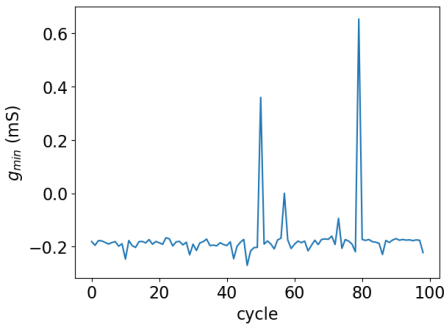


(a) g_{max} over the 100 cycles

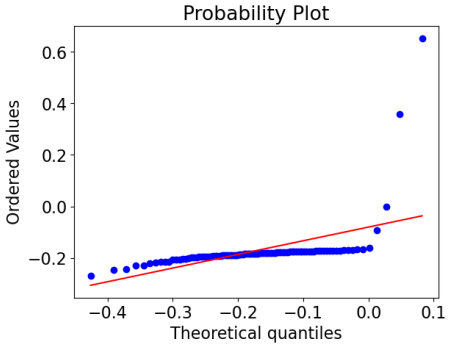


(b) $Q-Q$ plot of g_{max} against normal distribution $(y = mx + b; m = 1.0129; b = -0.0409)$

Figure 4.15: g_{max} distribution.

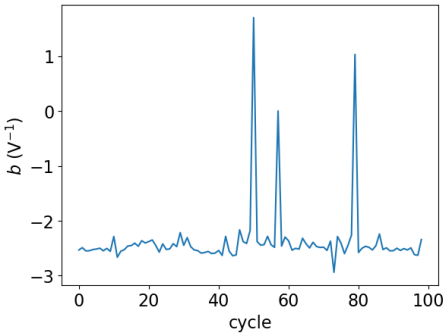


(a) g_{min} over the 100 cycles

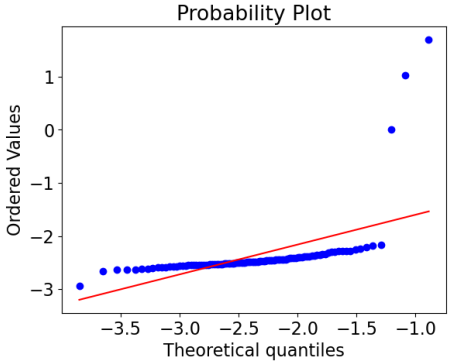


(b) $Q-Q$ plot of g_{min} against normal distribution $(y = mx + b; m = 0.5312; b = -0.0806)$

Figure 4.16: g_{min} distribution.

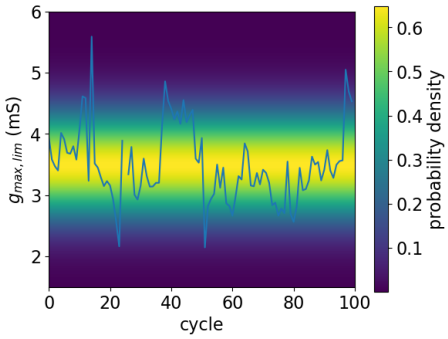


(a) b over the 100 cycles

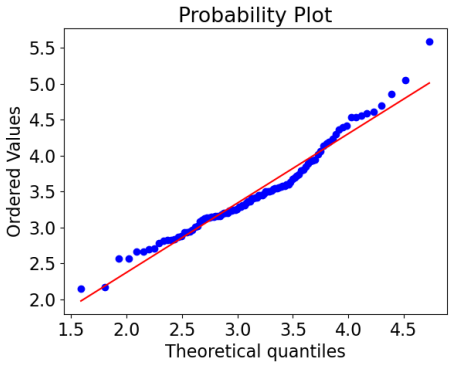


(b) $Q-Q$ plot of b against normal distribution $(y = mx + b; m = 0.5615; b = -1.0389)$

Figure 4.17: b distribution.

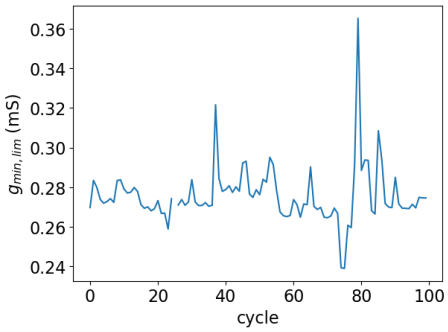


(a) $g_{max,lim}$ over the 100 cycles

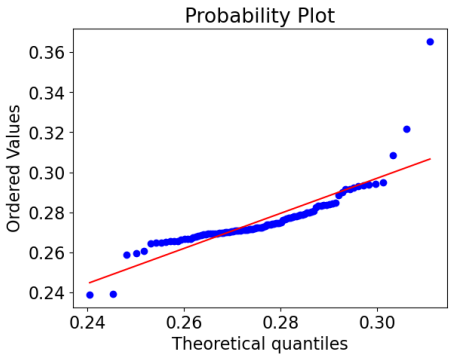


(b) $Q-Q$ plot of $g_{max,lim}$ against normal distribution $(y = mx + b; m = 0.9652; b = 0.4423)$

Figure 4.18: $g_{max,lim}$ distribution.



(a) $g_{min,lim}$ over the 100 cycles



(b) $Q-Q$ plot of $g_{min,lim}$ against normal distribution $(y = mx + b; m = 0.8760; b = 0.0342)$

Figure 4.19: $g_{min,lim}$ distribution.

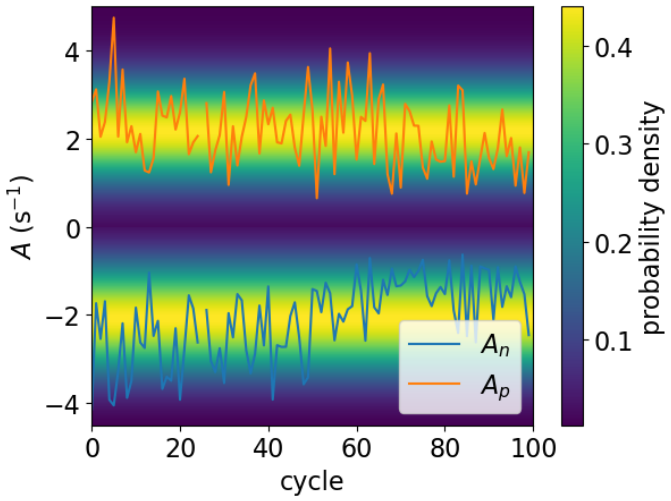
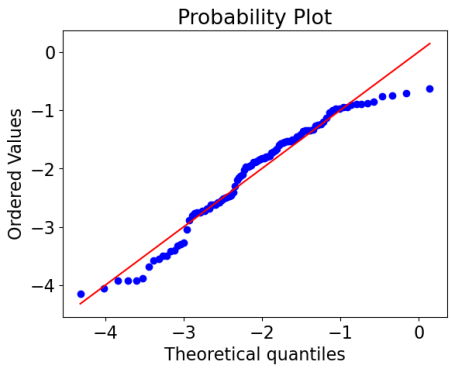
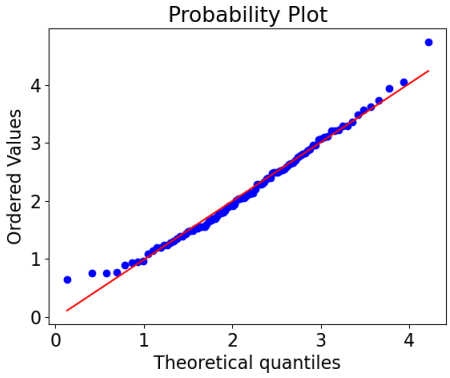


Figure 4.20: A_n and A_p distributions.

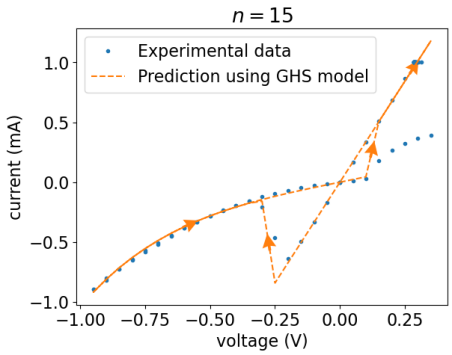


(a) $Q-Q$ plot of A_n against normal distribution $(y = mx + b; m = 0.9999; b = -0.0001)$

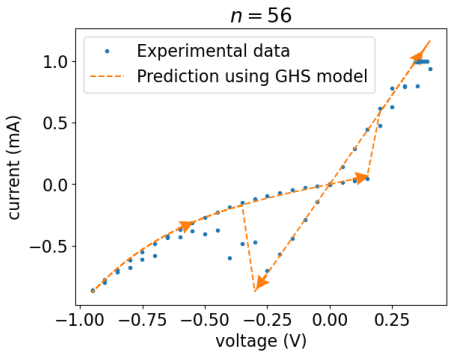


(b) $Q-Q$ plot of A_p against normal distribution $(y = mx + b; m = 1.0128; b = -0.0279)$

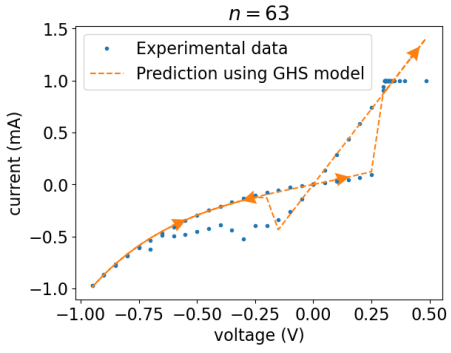
Figure 4.21: $Q-Q$ plots of A_n and A_n.



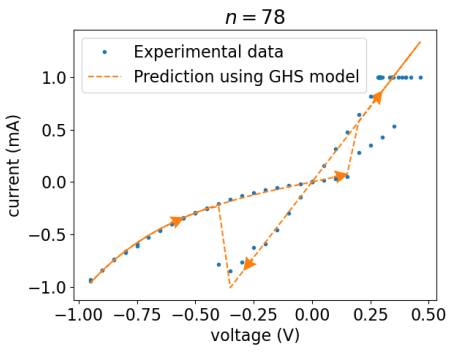
(a) $n = 15$



(b) $n = 56$



(c) $n = 63$



(d) $n = 78$

Figure 4.22: Predictions made by GHS model using the extracted parameters for *testing curves* $n = 15, 56, 63, 78$.

5. Neural network model

Artificial Neural Networks (ANN) work by taking the inputs and progressively transforming them as they move through the network's layers. At each layer, the network performs a linear transformation of the input data, multiplying the inputs by weights and adding biases. These weights are the key parameters that the network learns during training. After this linear transformation, the result is passed through a non-linear activation function at each neuron, such as the rectefied linear unit (ReLU), the logistic sigmoid, or the hyperbolic tangent. This activation function is crucial because it introduces non-linearity, enabling the network to learn complex patterns and representations from the data. Without it, the network would only be able to approximate linear functions, no matter how many layers it had. As this process of linear transformation and activation continues through the layers of the network, the final output is generated at the output layer. This output is then compared to the true values using a loss function. The loss function calculates the error or difference between the network's predictions and the actual target values. Next, the network uses back-propagation to adjust the weights. The key to back-propagation is computing the gradient of the loss with respect to the weights. This gradient indicates how the loss will change if the weights are adjusted slightly. The gradients are used to update the weights in a way that minimizes the loss. However, these updates are controlled by a factor called the learning rate. The learning rate plays a crucial role in determining how quickly or slowly the network adjusts its weights. It controls the size of the updates made to the weights during each iteration of optimisation. If the learning rate is too high, the network might overshoot the optimal solution, leading to instability. Conversely, if the learning rate is too low, the network may converge very slowly, potentially getting stuck in a local minimum and failing to find the optimal solution. Batch size refers to the number of samples the network processes before updating its weights. It can significantly influence the training process. Smaller batch sizes can lead to more precise updates, as each update is based on fewer samples, but this approach is computationally more expensive. Larger batch sizes, on the other hand, smooth out the updates, making the training faster, but may result in less accurate generalisation. Finally, the number of epochs indicates how many times the network will go through the entire dataset during training. More epochs allow the model to refine its understanding of the data, improving performance. However, too

many epochs can lead to over-fitting, where the network becomes too specialized in the training data and performs poorly on unseen data [35, 36].

5.1. The hybrid approach

To improve the quality of the approximation achieved using state variable models, the creation of a hybrid state variable/neural network was implemented. The approach presented in Ref. [37] consists of such a model, in which the $I-V$ curve approximation obtained by a state variable model is further improved by the use of neural networks along with that model. First, the state variable is calculated through

$$\frac{\partial w(t)}{\partial t} = a \cdot v(t)^s \begin{cases} 0 & -V_{thr} < v(t) \leq V_{thr} \quad (5.1a) \\ 1-\left(1-w(t)\right)^{2\cdot \text{round}\left(\frac{b}{|v(t)|+c}\right)} & v(t) \leq -V_{thr} \quad (5.1b) \\ 1-w(t)^{2\cdot \text{round}\left(\frac{b}{|v(t)|+c}\right)} & v(t) > V_{thr} \quad (5.1c) \end{cases}$$

where V_{thr} is the threshold activation voltage, a is constant, s is an odd integer, round is the function for obtaining an integer result and b and c are fitting coefficients.

In the state variable model, the current would be given by

$$i(t) = w(t)^n \beta \sinh\left(\alpha_M v(t)\right) + \chi\left[\exp\left(\gamma v(t)\right) - 1\right] \quad (5.2)$$

where n, β, α_M, χ and γ are fitting parameters. However, for a more accurate approximation, the current was given by

$$i(w,v) = a + \sum_{i=1}^{N} \frac{b_i}{1+\exp\left(-c_i w - d_i v - e_i\right)} \quad (5.3)$$

which corresponds to the output of a neural network with a single hidden layer with $N = 120$ neurons with logistic sigmoids as activation functions and (w, v), the state variable and voltage at a given time, as the input. A schematic representation of the neural network architecture is depicted in Figure 5.1.

In the presented example, the hybrid model constitutes a more accurate representation of the experimental data than the state variable model (Figure 5.2).

5.1.1. Testing neural network architectures and parameters

The value of the state variable calculated in the previous chapter was used to train neural networks, along with the voltage value, using the hybrid approach described. In this case, the model for the state variable was the GHS, not the one described by Equations 5.1-5.2. Different neural network architectures were explored as well as various parameters

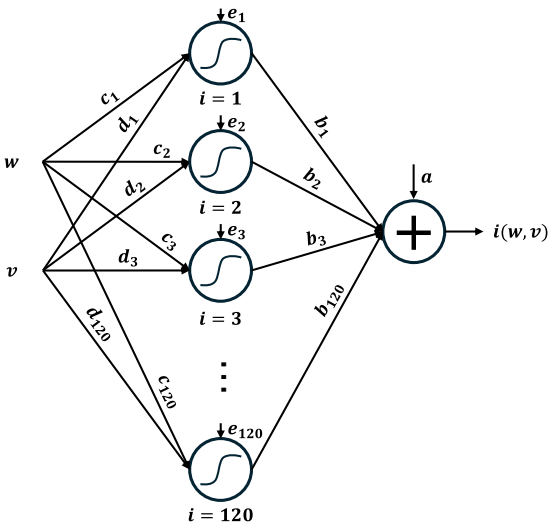
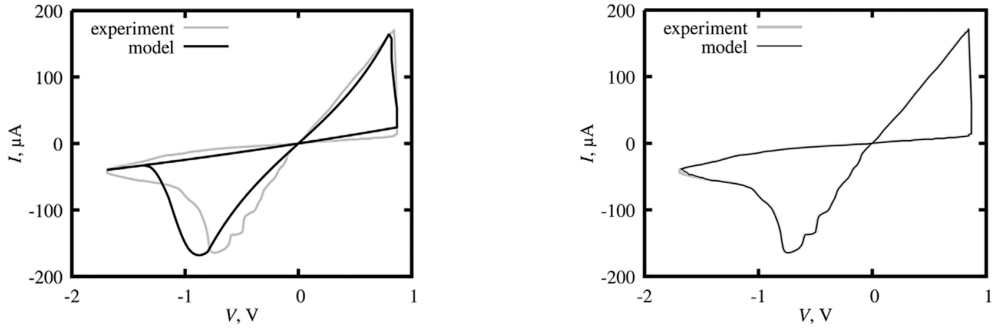


Figure 5.1: Depiction of the neural network architecture used in [37]



(a) State variable model described by Equations 5.1-5.2

(b) Hybrid state variable/neural network model

Figure 5.2: Comparison between state variable model and hybrid model (taken from Ref. [37])

(learning rate (LR), loss function, activation function, batch size, and number of epochs). 80% of the data were used for training and the remaining 20% were used for validation. The data were normalised as follows

$$\hat{v} = \frac{v - \overline{v}}{v_{max} - v_{min}} \tag{5.4}$$

$$\hat{x} = \frac{x - \bar{x}}{x_{max} - x_{min}} \tag{5.5}$$

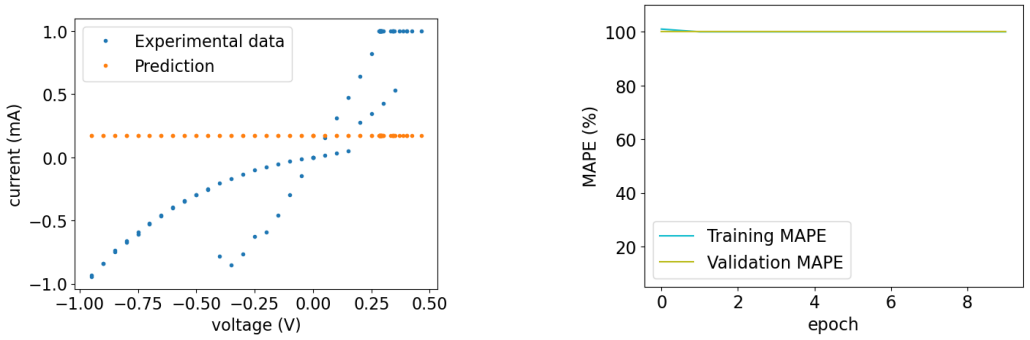
$$\hat{i} = \frac{i - \bar{i}}{i_{max} - i_{min}} \tag{5.6}$$

where $\overline{v}$ is the average of v, $\bar{x}$ is the average of x and $\bar{i}$ is the average of i. This normalization is important because leaving data with non-zero mean can lead to large differences in gradient along different directions leading to irregular progress in the convergence of the algorithm. Additionally, normalizing the variances of the input variables is important

because it helps improve the convergence speed of the training process by reducing the eccentricity of the error surface. If the input variables are normalized to have similar variances, it helps make the error surface more spherical, meaning the gradient descent algorithm can move more efficiently in all directions [36]. The models were trained using $\hat{v}$, $\hat{x}$ and $\hat{i}$. All training was performed using the open software library TensorFlow. [38]

5.1.1.1. Single-layer, 120 neurons, sigmoid activation at output neural network

The first neural network to be tested had a single hidden layer with 120 neurons using the logistic sigmoid as the activation function and a single neuron in the output layer, also using the sigmoid as the activation function. The optimizer used was Adaptive Moment Estimation (ADAM) and the loss function was Mean Absolute Percentage Error (MAPE). This loss function was chosen due to its easy interpretability. At the same time, the most commonly used loss function Mean Squared Error (MSE) presents small values and a more difficult convergence, as will be demonstrated by the results. A learning rate of 0.1 and a batch size of 10 were used and the model was trained for 10 epochs. At the second epoch, the training and validation losses get stuck at 100%. The predicted current is always $0.17\ \mathrm{mA}$. This indicates that the LR is too large, so the algorithm skips over the minimum of the loss function and gets trapped at a local minimum.



(a) Predictions

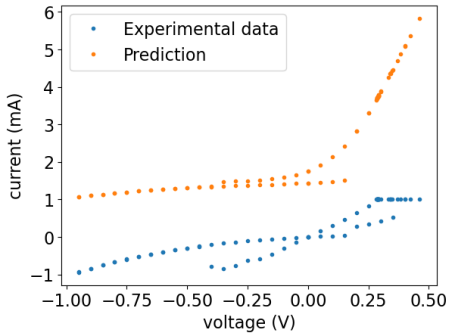
(b) Evolution of the MAPE

Figure 5.3: Results for 5.1.1.1 Single-layer, 120 neurons, sigmoid activation at output neural network for *testing curve* $n = 78$.

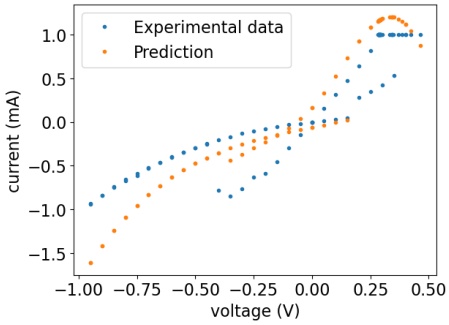
5.1.1.2. Single-layer, 120 neurons, no activation at output neural network

Another model was trained, which had the same architecture as the previous one and had no activation function at the output layer. The model was trained with the MAPE loss function, a batch size of 10 and a LR of 0.1. The model was trained for 10 epochs, resulting in a training loss of 179% and a validation loss of 571%. The training continued, this time with LR of 0.01, for 20 epochs. The training loss evolves to 40% and the validation loss to

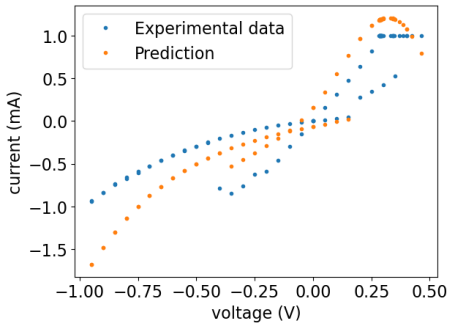
18%. Training it with a LR of 0.001 for another 10 epochs results in a training loss of 25% and a validation loss of 18%. Training for 10 more epochs with a LR of 0.0001 results in a training loss of 23% and a validation loss of 16%.



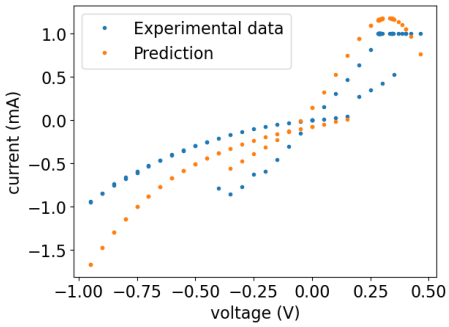
(a) Predictions using a LR of 0.1 for 10 epochs with a batch size of 10 and MAPE loss function



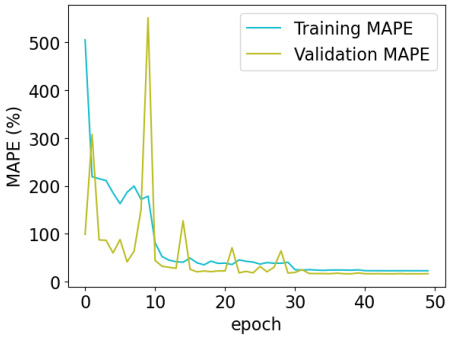
(b) Predictions using a LR of 0.01 for 20 epochs with a batch size of 10 and MAPE loss function



(c) Predictions using a LR of 0.001 for 10 epochs with a batch size of 10 and MAPE loss function



(d) Predictions using a LR of 0.0001 for 10 epochs with a batch size of 10 and MAPE loss function



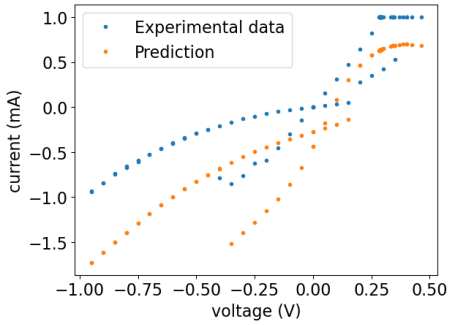
(e) Evolution of the MAPE

Figure 5.4: Results for 5.1.1.2 Single-layer, 120 neurons, no activation at output neural network for *testing curve* $n = 78$.

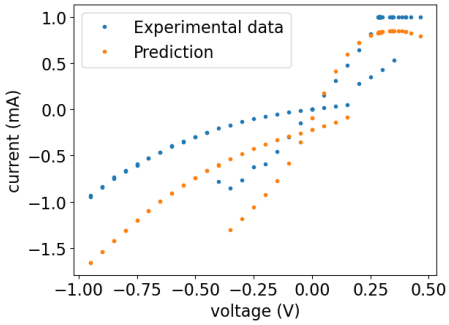
5.1.1.3. Trying the MSE loss function

What was described in 5.1.1.2 was repeated, this time using MSE as the loss function. For the first training, the MAPE at the end of the training was 47% for the training data and 43% for the validation data. In this case, the MSE loss function is revealed to be

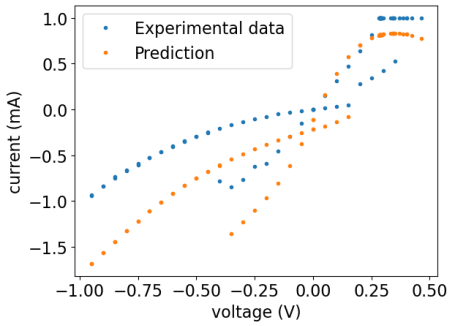
more adequate than the MAPE. The MAPE is still used as a metric because of its easier interpretability. At the second training, the training MAPE is of 29% and the validation MAPE of 18%. In this training round, MSE still presents better results, however, they are approaching the results achieved using MAPE as the loss function. At the third training, the training and validation MAPE are of 27% and 18%, respectively. Using MAPE as a measure of the quality of the approximation, here this function becomes more suitable than MSE. This can be simply because we are using it as measure. At the final training step, the training MAPE reaches the value of 27% and the validation MAPE the value of 16%.



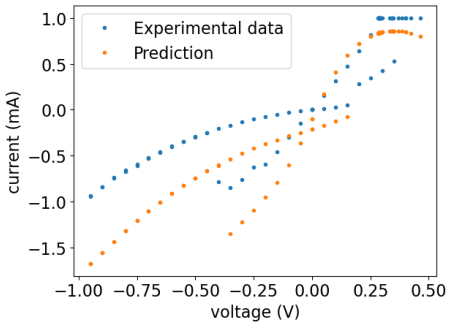
(a) Predictions using a LR of 0.1 for 10 epochs with a batch size of 10 and MSE loss function



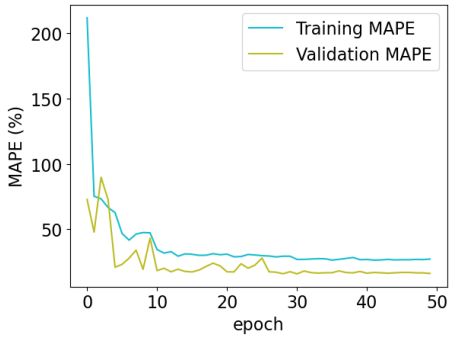
(b) Predictions using a LR of 0.01 for 20 epochs with a batch size of 10 and MSE loss function



(c) Predictions using a LR of 0.001 for 10 epochs with a batch size of 10 and MSE loss function



(d) Predictions using a LR of 0.0001 for 10 epochs with a batch size of 10 and MSE loss function

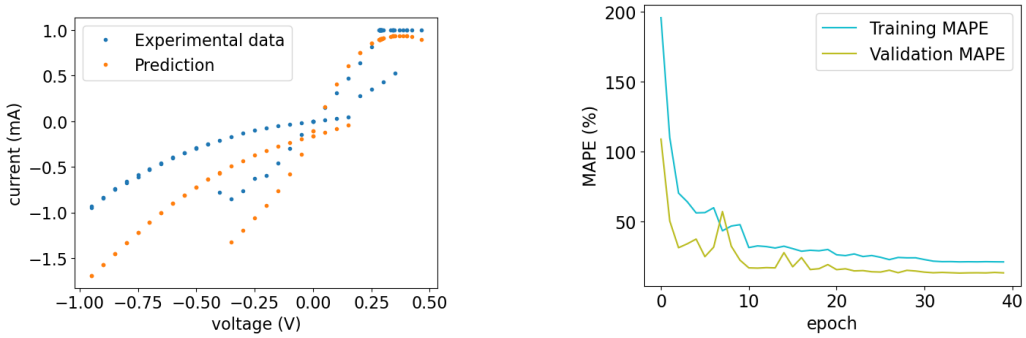


(e) Evolution of the MAPE

Figure 5.5: Results for 5.1.1.3 Trying the MSE loss function for *testing curve* $n = 78$.

5.1.1.4. Using both MSE and MAPE losses

A mixed approach using both the MSE and MAPE loss functions was taken. A first round of training using a LR of 0.1, a batch size of 10 and the MSE loss function was performed for 10 epochs. This was followed by a second round of training still using the same batch size and loss function (MSE) and a LR of 0.01 for 10 epochs. The third round of training used MAPE as the loss function, with LR of 0.001 for 10 epochs. The final round of training used MAPE as the loss function and LR of 0.0001. The training MAPE after all four rounds of training was of 21% and the validation MAPE of 13%.



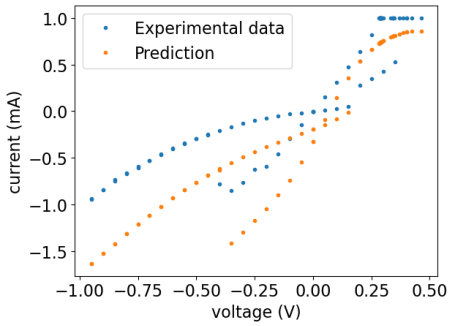
(a) Predictions

(b) Evolution of the MAPE

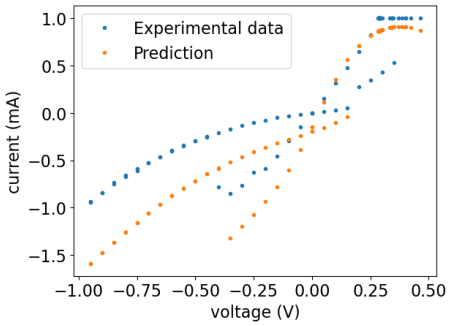
Figure 5.6: Results for 5.1.1.4 Using both MSE and MAPE losses for *testing curve* $n = 78$.

5.1.1.5. Using more neurons

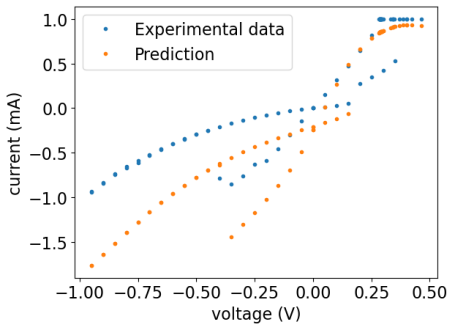
As the current can be approximated by Equation 5.3, one can assume that a higher value of N will result in a more accurate approximation. To test this, a neural network with a single hidden layer with 200 neurons was implemented. The first training happened with MSE as the loss function and a LR of 0.1 for 10 epochs, resulting in training and validation MAPE of 53% and 19%.The second, still with MSE as the loss function, and a LR of 0.01 for 10 epochs, resulting in training MAPE of 30% and validation MAPE of 17%. The model was recompiled with MAPE as the loss function and a LR of 0.001 and trained for 10 epochs. The training MAPE was of 23% and the validation MAPE of 19%. Finally, the model was compiled with LR of 0.0001 and MAPE as the loss function and trained for 10 epochs. The final training MAPE was of 21% and the validation MAPE of 13%. The increase in number of neurons only slightly improves the approximation, so it can be concluded that it is not a good approach to increase the number of neurons, as it increases the number of parameters to be trained and does not result in a significant improvement of the approximation.



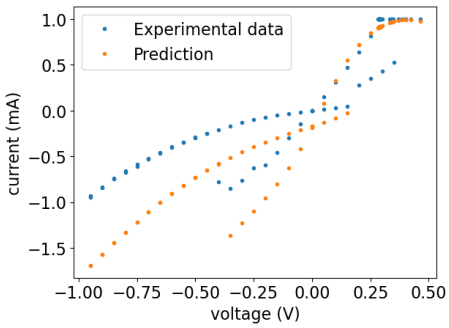
(a) Predictions using a LR of 0.1 for 10 epochs with a batch size of 10 and MSE loss function



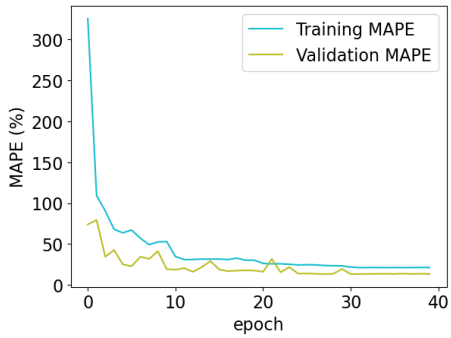
(b) Predictions using a LR of 0.01 for 10 epochs with a batch size of 10 and MSE loss function



(c) Predictions using a LR of 0.001 for 10 epochs with a batch size of 10 and MAPE loss function



(d) Predictions using a LR of 0.0001 for 10 epochs with a batch size of 10 and MAPE loss function



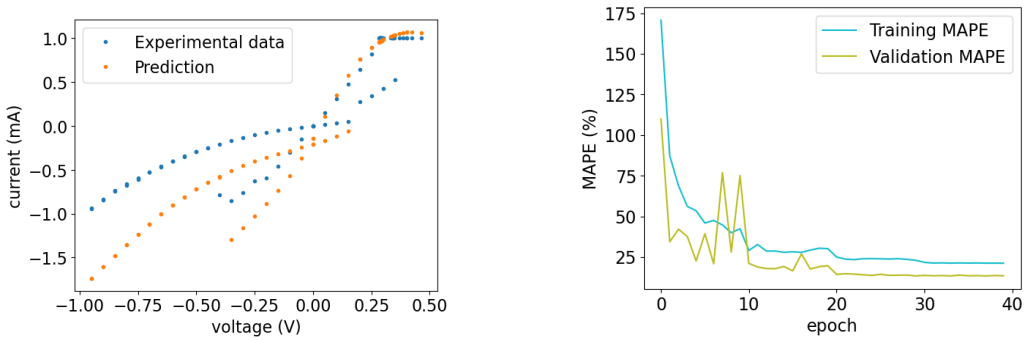
(e) Evolution of the MAPE

Figure 5.7: Results for 5.1.1.5 Using more neurons for *testing curve* $n = 78$.

5.1.1.6. Using fewer neurons

Keeping in mind that using 200 neurons did not markedly improve the model, the question arises of whether 120 are needed. Decreasing the number of neurons reduces the number of trainable parameters, thus making the model more simple and requiring fewer operations to make predictions. A model with a single hidden layer with 80 neurons was tested. Training with a LR of 0.1 with MSE loss function for 10 epochs results in training MAPE of 42% and validation MAPE of 75%. The subsequent training with LR of 0.01 and the same loss function for 10 epochs results in training and validation MAPE of 30%

and 20%. The next training with LR of 0.001 and MAPE loss function results in a training MAPE of 23% and a validation MAPE of 13%. The fourth training step, with LR of 0.0001 with MAPE loss function for 10 epochs resulted in training MAPE of 21% and validation MAPE of 13%. The quality of the approximation does not decrease, so it is advantageous to use a lower number of neurons.



(a) Predictions

(b) Evolution of the MAPE

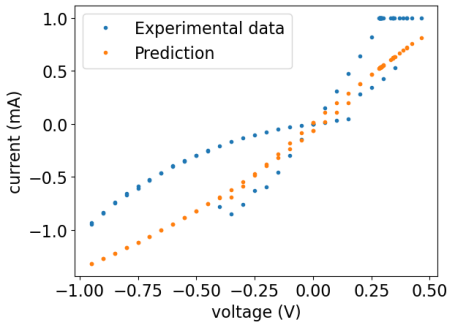
Figure 5.8: Results for 5.1.1.6 Using fewer neurons for *testing curve* $n = 78$.

5.1.1.7. Using batch size of 100

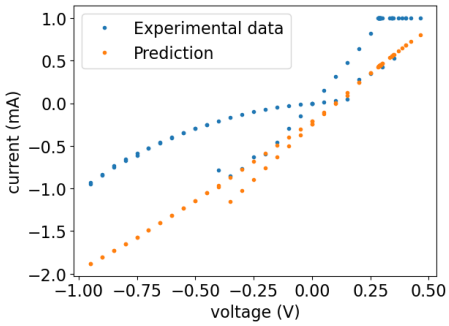
Testing of a different batch size was performed. The same neural network architecture using 80 neurons in a single hidden layer was used. The first round of training used a LR of 0.1, the MSE loss function and a batch size of 100 for 10 epochs. The resulting MAPE was 87% for training data and 50% for validation data. Then, training was performed with a LR of 0.01, the MSE loss function and, again, a batch size of 100. The training MAPE was 47% and the validation MAPE of 29%. As previously, a third round of training was performed, with LR of 0.001, MAPE loss function and a batch size of 100 for 10 epochs. Training MAPE was 35% and validation MAPE 31%. The last training round, with LR of 0.0001, MAPE loss function and batch size of 100 for 10 epochs presented a training MAPE of 29% and a validation MAPE of 25%. We can conclude that the smaller batch size of 10 is preferable.

5.1.1.8. Using a batch size of 1

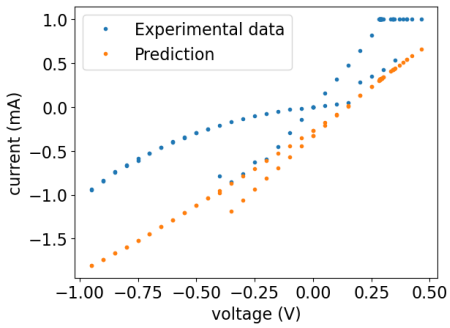
Performing the same test as in 5.1.1.6 and 5.1.1.7, this time with a batch size of 1, leads to a training MAPE of 25% and validation MAPE of 19%, which is a poorer approximation than the one attained in 5.1.1.6 with a batch size of 10.



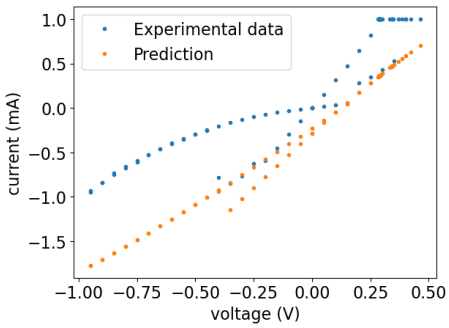
(a) Predictions using a LR of 0.1 for 10 epochs with a batch size of 100 and MSE loss function



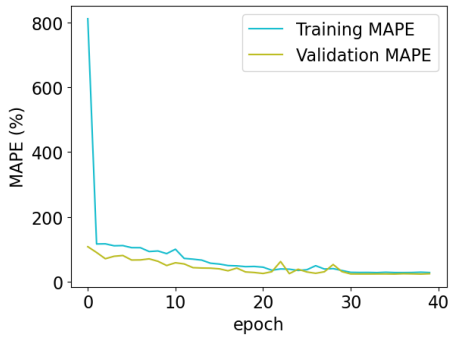
(b) Predictions using a LR of 0.01 for 10 epochs with a batch size of 100 and MSE loss function



(c) Predictions using a LR of 0.001 for 10 epochs with a batch size of 100 and MAPE loss function



(d) Predictions using a LR of 0.0001 for 10 epochs with a batch size of 100 and MAPE loss function

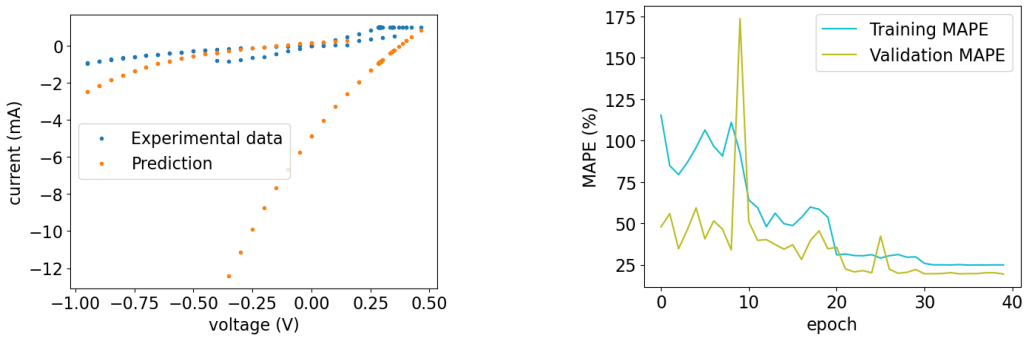


(e) Evolution of the MAPE

Figure 5.9: Results for 5.1.1.7 Using batch size of 100 for *testing curve* $n = 78$.

5.1.1.9. Three-layer neural network

In classification problems, a three-layer neural network can group points where the classes correspond to any union of polyhedra [35]. Although this is not a classification problem, an experiment was conducted to describe a model with 3 hidden layers and an output layer without an activation function. The hidden layers used the logistic sigmoid as an activation function. Besides having more capabilities (at least in classification problems), implementing a deeper network provides a solution with fewer training parameters that



(a) Predictions

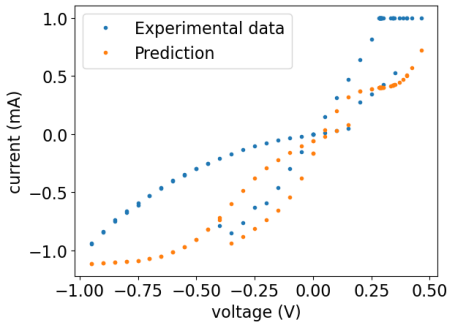
(b) Evolution of the MAPE

Figure 5.10: Results for 5.1.1.8 Using a batch size of 1 for *testing curve* $n = 78$.

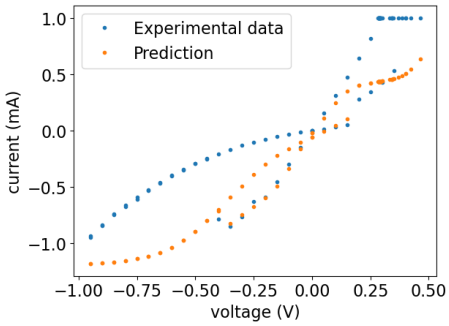
could have the same or better quality than the approximation achieved using a single-layer neural network. The first hidden layer had 4 neurons, the second 16 neurons and the third 4 neurons. The output layer remained with no activation function. The first training was performed with a LR of 0.1, batch size of 10 and MSE loss function for 10 epochs. The training MAPE was 45% and the validation MAPE of 14%. The second training was carried out with LR of 0.01, batch size of 10 and MSE loss function for 10 epochs. The obtained training MAPE was 26% and the validation MAPE 14%. The subsequent training was conducted with LR of 0.001, MAPE loss function and batch size of 10 for 10 epochs. This led to a training MAPE of 21% and a validation MAPE of 12%. The next training, with LR of 0.0001, MAPE loss function and batch size of 10 for 10 epochs resulted in a training MAPE of 19% and a validation MAPE of 12%. A final round of training with LR of 0.00001 was performed. The final training and validation MAPE were 19% and 12%, revealing that this final round was unnecessary. This deep network provided a better approximation than the single-layer network, employing fewer parameters, so it is a better choice for this hybrid model.

5.1.1.10. Stochastic neural network

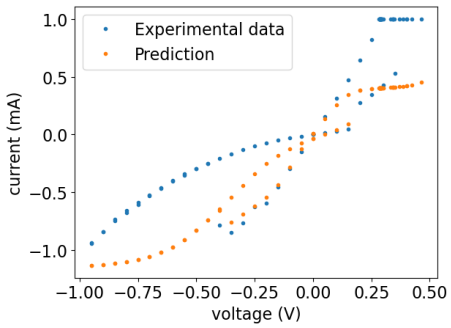
In an attempt to model the device's intrinsic variability, a neural network with a stochastic layer was tested. The network had three initial layers, as in the preceding model. The output of the third layer, however, was not directly connected to the output but was instead mapped onto a two-dimensional space meant to represent an average and standard deviation for the output variable. These values were used to create a new variable with added noise, which was then linearly transformed. This final variable corresponded to the current i predicted by the network for a pair (w, v). The architecture is inspired by the Variational



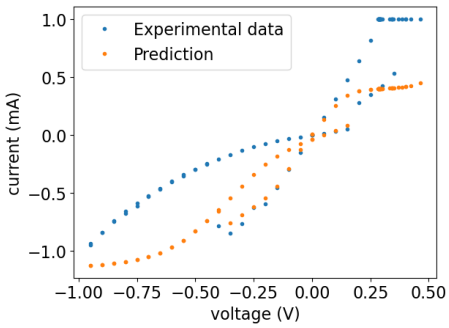
(a) Predictions using a LR of 0.1 for 10 epochs with a batch size of 10 and MSE loss function



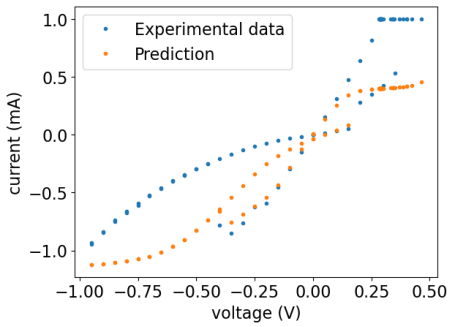
(b) Predictions using a LR of 0.01 for 10 epochs with a batch size of 10 and MSE loss function



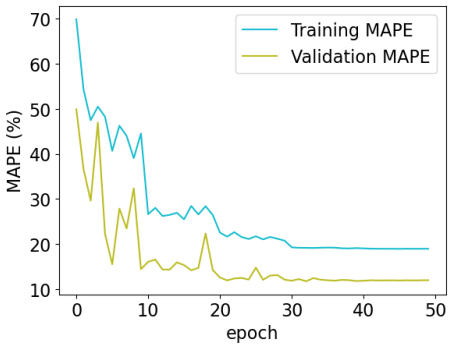
(c) Predictions using a LR of 0.001 for 10 epochs with a batch size of 10 and MAPE loss function



(d) Predictions using a LR of 0.0001 for 10 epochs with a batch size of 10 and MAPE loss function



(e) Predictions using a LR of 0.00001 for 10 epochs with a batch size of 10 and MAPE loss function



(f) Evolution of the MAPE

Figure 5.11: Results for 5.1.1.9 Three-layer neural network for *testing curve* $n = 78$.

AutoEncoder (VAE) architecture [39, 40], but rather than reconstructing the original input, it predicts the output. This architecture introduces variability to the model that was believed to improve the predictions, as the original data inherently exhibit variability.

The model used the ADAM optimizer, a batch size of 10 and was trained for three rounds. The first round used the MSE loss function and a LR of 0.001, for 30 epochs. It resulted in training MAPE of 36% and validation MAPE of 27%. The second round used MAPE as

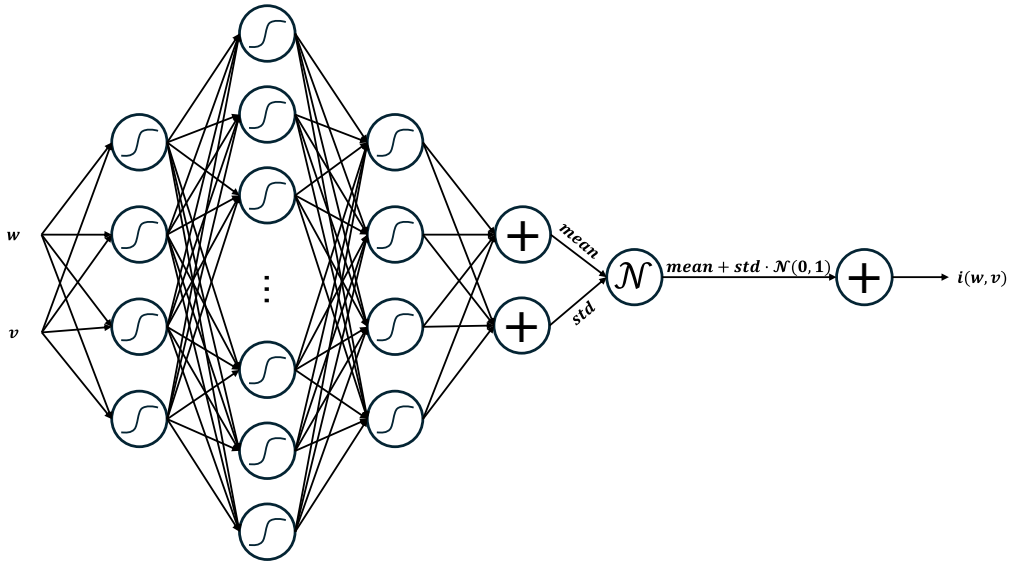


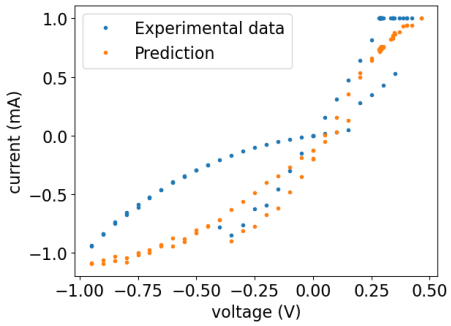
Figure 5.12: Stochastic neural network architecture (with bias omitted for the first, second, third, fourth and output layer)

the loss function and LR of 0.0001, for 20 epochs, resulting in training MAPE of 25% and validation MAPE of 18%. The last training round, with MAPE loss and 0.00001 LR, carried out over 10 epochs, was not fruitful, as the training and validation MAPE did not decrease. The addition of stochasticity did not improve the model.

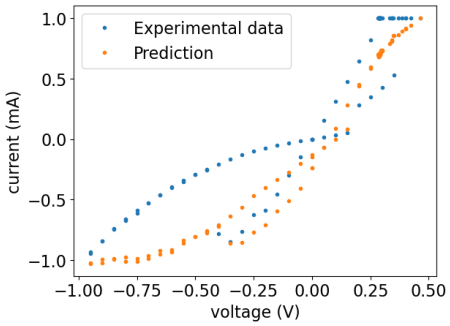
5.1.2. Summary

In this exploration of neural network architectures for hybrid state variable modelling, we aimed to optimize the approximation of complex systems by systematically varying key parameters. Initially, a single-layer neural network with 120 neurons was tested, revealing the challenges of finding the right balance between learning rate (LR) and loss function. Early trials demonstrated that a high LR could cause the model to converge poorly, while adjustments in the LR improved the model's performance, particularly when switching between MAPE and MSE loss functions. These findings underscored the importance of fine-tuning hyperparameters to achieve more accurate approximations.

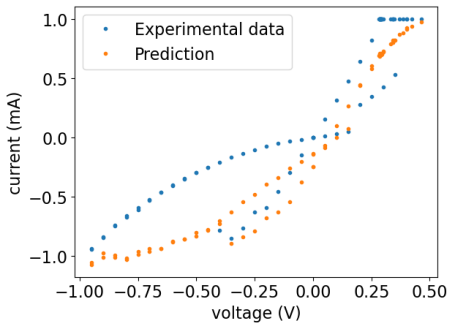
Subsequent experiments focused on modifying the architecture itself. The experimentation with increasing and decreasing the number of neurons demonstrated that simply adding more neurons does not necessarily lead to better performance. A model with 200 neurons in a single hidden layer showed marginal improvement, whereas reducing the number of neurons to 80 maintained model quality while simplifying the architecture. The investigation also examined the effects of varying batch sizes, concluding that smaller



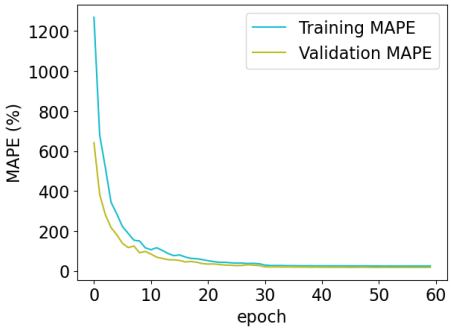
(a) Predictions using a LR of 0.001 for 30 epochs with a batch size of 10 and MSE loss function



(b) Predictions using a LR of 0.0001 for 20 epochs with a batch size of 10 and MAPE loss function



(c) Predictions using a LR of 0.00001 for 10 epochs with a batch size of 10 and MAPE loss function



(d) Evolution of the MAPE

Figure 5.13: Results for 5.1.1.10 Stochastic neural network for *testing curve* $n = 78$.

batches (specifically, a batch size of 10) provided better results compared to larger ones, which led to poorer approximations.

The transition to a deeper neural network with three layers offered significant improvements over the single-layer models. This deep network not only provided a more accurate approximation, but also required fewer parameters, making it a more efficient and effective choice for hybrid modelling.

Finally, a stochastic neural network inspired by the Variational Autoencoder (VAE) was tested to account for the intrinsic variability in the data. The addition of stochasticity did not improve the model's performance, indicating that this complexity might not be necessary for the specific problem at hand.

The deep three-layer network emerged as the most effective, balancing complexity and performance, whereas the stochastic approach, though theoretically promising, did not yield the expected improvements.

5.2. The memory approach

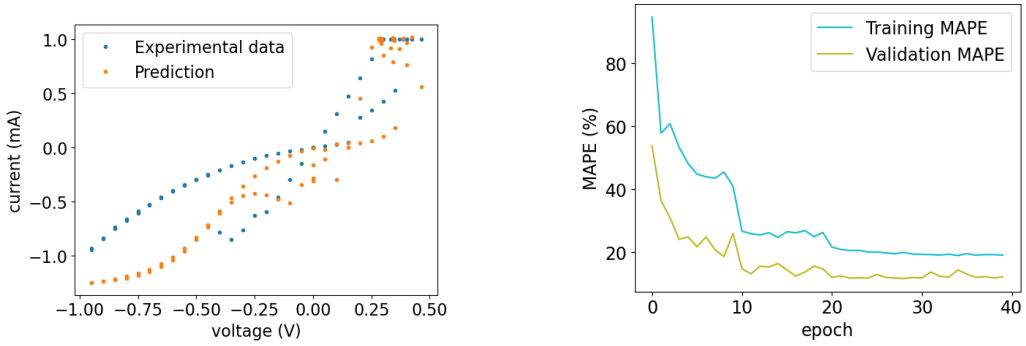
Rather than using the state variable and voltage as inputs to the neural network, the present voltage value, along with its previous values, was utilized this time. This approach was based on work that has already been done to model ferromagnetic hysteresis loops using neural networks [41]. This type of model has the advantage of not needing an additional model to find the state variable and is instead entirely neural network based: it is only necessary to train the neural network, and no additional parameters or variables need to be calculated.

It was chosen to use the present voltage value along with the two previous ones. The neural network is trained using the pairs

$$\left[\left(v\left(t\right), v\left(t-\Delta t_1\right), v\left(t-\Delta t_2\right)\right), i\left(t\right)\right] \tag{5.7}$$

where $\Delta t_2 > \Delta t_1$ and the consecutive measurements occur at $t-\Delta t_2$, $t-\Delta t_1$ and t. The network receives as input $\left(v\left(t\right), v\left(t-\Delta t_1\right), v\left(t-\Delta t_2\right)\right)$ and returns a prediction for $i\left(t\right)$ as the output.

The neural network used had three hidden layers, with 4, 16 and 4 neurons each with sigmoid activation functions, like the one used in 5.1.1.9. The model was trained with a batch size of 1, a LR 0.01 and MSE loss for 10 epochs. It was trained for 10 additional epochs with a LR of 0.001. It was then trained for 20 more epochs, this time with MAPE loss and a LR of 0.0001. After training, the resulting training MAPE was 19% and validation MAPE 12%. The model shows results comparable to the ones obtained by the hybrid mode (in this case, even better), with the advantage of not needing an underlying model to get the results.



(a) Predictions

(b) Evolution of the MAPE

Figure 5.14: Results for the neural network with past memory for *testing curve* $n = 78$.

Conclusion

This work explored the development and optimisation of hybrid state variable/neural network models to enhance the accuracy of modelling complex systems, specifically memristor devices. The research aimed to address the limitations of traditional state variable models, which often struggle with non-linearities and the intrinsic variability found in experimental data. However, the findings indicate that while neural networks offer some benefits, they are not necessarily superior to state variable models, especially given the current dataset and computational constraints.

The initial implementation of a single-layer neural network with 120 neurons highlighted the challenges in optimizing learning rates and loss functions. While adjustments led to improved accuracy, the neural network models still required a significantly higher number of parameters compared to state variable models. This complexity, combined with the large number of linear transformations in neural networks, poses a substantial computational burden. In scenarios involving extended simulations or large arrays of memristors, the computational inefficiency of neural networks becomes apparent, potentially rendering them impractical for simulating projected circuits before their physical implementation.

Further experimentation with different neural network architectures, including deeper models and variations in neuron count, reinforced this conclusion. Deeper networks provided slightly better approximations utilizing fewer parameters while still managing to capture the essential dynamics of the system. The stochastic neural network, inspired by Variational Autoencoders (VAE), was tested to introduce variability into the model, but it failed to deliver the anticipated improvements in accuracy. These results suggest that while neural networks can model complex behaviours, they may not be the most effective tool for all types of simulations, particularly with the limited dataset available for this device.

On the other hand, the state variable model, despite its limitations, particularly in accurately capturing transitions, remains a robust option. Its simplicity and lower computational demands make it suitable for simulations that require large-scale memristor arrays or extended simulation times. The memory model, which was also investigated, appears to offer better accuracy than the hybrid approach, suggesting that neural networks can be used separately and independently from state variable models and can be used when the tailoring of a state variable model for the specific device is not beneficial.

It would be beneficial to gather data without the application of current compliance. The use of current compliance in this study, while necessary to protect the device, likely obscured certain behaviours, leaving some aspects of the memristor response unknown. Data without current compliance would provide a fuller picture of the device's behaviour, potentially revealing important dynamics that are crucial for developing more accurate models.

In conclusion, while neural network models hold promising modelling complex systems, their high computational costs and the significant number of parameters required limit their practicality in certain contexts. State variable models, particularly those refined to address specific device characteristics, offer a more balanced approach that combines simplicity with sufficient accuracy. The research underscores the importance of selecting the modelling approach based on the specific requirements of the simulation, whether it be accuracy, computational efficiency, or the ability to handle large-scale systems. Future exploration in developing more specialized state variable models could further advance the field, providing robust tools for simulating complex devices like memristors with greater precision and efficiency. Such a model could address the current model's shortcomings in transition regions and potentially incorporate stochasticity by averaging parameters and including standard deviations. This approach would retain the simplicity and computational efficiency of state variable models while improving accuracy and broadening their applicability in simulations. Furthermore, the addition of stochasticity could better model real devices which exhibit intra-device variability (throughout the cycles) as well as inter-device variability, some of it introduced by random defects during the fabrication process. Future work could also explore the potential of memory neural networks with larger datasets. The current study was somewhat limited by the size of the dataset and the number of cycles performed on the device, which restricted the amount of data available for training. By increasing the number of cycles and expanding the dataset, a memory neural network could be better trained to capture the complex dynamics and variability of memristors more effectively. With a richer dataset, the model's ability to generalise across different operating conditions could be significantly improved, potentially leading to better performance and greater applicability in practical scenarios.

References

[1] H. Bao, H. Zhou, J. Li, H. Pei, J. Tian, L. Yang, S. Ren, S. Tong, Y. Li, Y. He *et al.*, "Toward memristive in-memory computing: principles and applications," *Frontiers of Optoelectronics*, vol. 15, no. 1, p. 23, 2022. [Cited on page 1.]

[2] I. Barraj, H. Mestiri, and M. Masmoudi, "Overview of memristor-based design for analog applications," *Micromachines*, vol. 15, no. 4, 2024. [Cited on page 1.]

[3] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971. [Cited on pages 1 and 3.]

[4] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008. [Cited on pages viii, 1, 7, 8, 10, and 19.]

[5] P. Sheridan and W. Lu, *Memristors and Memristive Devices for Neuromorphic Computing*. Cham: Springer International Publishing, 2019, pp. 369–389. [Cited on pages 1 and 2.]

[6] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Magic—memristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014. [Cited on page 1.]

[7] Q. Xia, W. Robinett, M. W. Cumbie, N. Banerjee, T. J. Cardinali, J. J. Yang, W. Wu, X. Li, W. M. Tong, D. B. Strukov, G. S. Snider, G. Medeiros-Ribeiro, and R. S. Williams, "Memristor−cmos hybrid integrated circuits for reconfigurable logic," *Nano Letters*, vol. 9, no. 10, pp. 3640–3645, 2009, pMID: 19722537. [Cited on page 1.]

[8] L. Chua and S. M. Kang, "Memristive devices and systems," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976. [Cited on pages viii, 3, and 6.]

[9] L. Chua, *The Fourth Element*. Cham: Springer International Publishing, 2019, pp. 1–14. [Cited on page 3.]

[10] B. Muthuswamy and S. Banerjee, *Two-Terminal Network Elements*. Cham: Springer International Publishing, 2019, pp. 1–62. [Cited on page 4.]

[11] B. Muthuswamy, J. Jevtic, H. H. C. Iu, C. K. Subramaniam, K. Ganesan, V. Sankaranarayanan, K. Sethupathi, H. Kim, M. P. Shah, and L. O. Chua, "Memristor modelling," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 490–493. [Cited on pages viii and 5.]

[12] L. Chua, *Everything You Wish to Know About Memristors but Are Afraid to Ask*. Cham: Springer International Publishing, 2019, pp. 89–157. [Cited on pages vii, 6, and 20.]

[13] ——, *Resistance Switching Memories are Memristors*. Cham: Springer International Publishing, 2019, pp. 197–230. [Cited on page 6.]

[14] ——, *Memristor, Hodgkin-Huxley, and Edge of Chaos*. Cham: Springer International Publishing, 2019, pp. 287–313. [Cited on page 6.]

[15] F. Qin, Y. Zhang, H. W. Song, and S. Lee, "Enhancing memristor fundamentals through instrumental characterization and understanding reliability issues," *Materials Advances*, vol. 4, no. 8, pp. 1850–1875, 2023. [Cited on page 7.]

[16] L. Gao, Q. Ren, J. Sun, S.-T. Han, and Y. Zhou, "Memristor modeling: challenges in theories, simulations, and device variability," *Journal of Materials Chemistry C*, vol. 9, no. 47, pp. 16 859–16 884, 2021. [Cited on page 7.]

[17] B. Mohammad, M. A. Jaoude, V. Kumar, D. M. A. Homouz, H. A. Nahla, M. Al-Qutayri, and N. Christoforou, "State of the art of metal oxide memristor devices," *Nanotechnology Reviews*, vol. 5, no. 3, pp. 311–329, 2016. [Online]. Available: https://doi.org/10.1515/ntrev-2015-0029 [Cited on pages viii and 7.]

[18] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Team: Threshold adaptive memristor model," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 211–221, 2013. [Cited on pages viii, 8, and 10.]

[19] M. D. Pickett, D. B. Strukov, J. L. Borghetti, J. J. Yang, G. S. Snider, D. R. Stewart, and R. S. Williams, "Switching dynamics in titanium dioxide memristive devices," *Journal of Applied Physics*, vol. 106, no. 7, p. 074508, 10 2009. [Cited on page 8.]

[20] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "Vteam: A general model for voltage-controlled memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 786–790, 2015. [Cited on page 10.]

[21] D. Liu, H. Cheng, X. Zhu, G. Wang, and N. Wang, "Analog memristors based on thickening/thinning of ag nanofilaments in amorphous manganite thin films," *ACS Applied Materials & Interfaces*, vol. 5, no. 21, pp. 11 258–11 264, 2013, pMID: 24083960. [Cited on page 10.]

[22] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Information-theoretic sneak-path mitigation in memristor crossbar arrays," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 4801–4813, 2016. [Cited on page 10.]

[23] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Magic—memristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014. [Cited on page 10.]

[24] M. D. Pickett, D. B. Strukov, J. L. Borghetti, J. J. Yang, G. S. Snider, D. R. Stewart, and R. S. Williams, "Switching dynamics in titanium dioxide memristive devices," *Journal of Applied Physics*, vol. 106, no. 7, p. 074508, 10 2009. [Cited on page 10.]

[25] R. E. Pino, J. W. Bohl, N. McDonald, B. Wysocki, P. Rozwood, K. A. Campbell, A. Oblea, and A. Timilsina, "Compact method for modeling and simulation of memristor devices: Ion conductor chalcogenide-based memristor devices," in *2010 IEEE/ACM International Symposium on Nanoscale Architectures*, 2010, pp. 1–4. [Cited on pages viii and 11.]

[26] C. Yakopcic, T. M. Taha, G. Subramanyam, R. E. Pino, and S. Rogers, "A memristor device model," *IEEE Electron Device Letters*, vol. 32, no. 10, pp. 1436–1438, 2011. [Cited on page 11.]

[27] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, "Generalized memristive device spice model and its application in circuit design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 8, pp. 1201–1214, 2013. [Cited on pages 11 and 12.]

[28] C. Yakopcic, T. M. Taha, D. J. Mountain, T. Salter, M. J. Marinella, and M. McLean, "Memristor model optimization based on parameter extraction from device characterization data," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1084–1095, 2020. [Cited on pages 12, 20, and 26.]

[29] T. Chang, S.-H. Jo, K.-H. Kim, P. Sheridan, S. Gaba, and W. Lu, "Synaptic behaviors and modeling of a metal oxide memristive device," *Applied physics A*, vol. 102, pp. 857–863, 2011. [Cited on page 12.]

[30] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, *Memristor SPICE Modeling*. Dordrecht: Springer Netherlands, 2012, pp. 211–244. [Cited on page 14.]

[31] "LTspice® XVII," accessed: 2024-09-04. [Online]. Available: https://agnd.net/valpo/341/LTspiceHelpXVII.pdf [Cited on page 14.]

[32] C. Dias, L. M. Guerra, J. Ventura, and P. Aguiar, "Memristor-based Willshaw network: Capacity and robustness to noise in the presence of defects," *Applied Physics Letters*, vol. 106, no. 22, p. 223505, 06 2015. [Cited on page 16.]

[33] C. J. L. da Silva Dias, "Resistive Switching in MgO and Si/Ag Metal-Insulator-Metal structures," Ph.D. dissertation, Faculty of Sciences of the University of Porto, 2019. [Cited on page 18.]

[34] H. Thode, *Testing For Normality*, ser. Statistics, textbooks and monographs. CRC Press, 2002. [Cited on page 26.]

[35] S. Theodoridis, "Chapter 18 - neural networks and deep learning," in *Machine Learning (Second Edition)*, second edition ed., S. Theodoridis, Ed. Academic Press, 2020, pp. 901–1038. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128188033000301 [Cited on pages 33 and 41.]

[36] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, *Efficient BackProp*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 9–50. [Cited on pages 33 and 35.]

[37] K. K. Abgaryan, A. Y. Morozov, and D. L. Reviznikov, "Hybrid approach for modeling memristive elements," *physica status solidi (b)*, p. 2400058, 2024. [Cited on pages ix, 33, and 34.]

[38] Accessed: 2024-03-01. [Online]. Available: https://www.tensorflow.org/?hl=pt [Cited on page 35.]

[39] D. Foster and K. Friston, *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*. O'Reilly, 2023. [Cited on page 43.]

[40] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, p. 307–392, 2019. [Online]. Available: http://dx.doi.org/10.1561/2200000056 [Cited on page 43.]

[41] S. Quondam Antonio, V. Bonaiuto, F. Sargeni, and A. Salvini, “Neural network modeling of arbitrary hysteresis processes: Application to go ferromagnetic steel,” *Magnetochemistry*, vol. 8, no. 2, 2022. [Cited on page 46.]