REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

MEMRISTOR BASED SYNAPSE DESIGN

Seda GÜNAKIN

MASTER OF SCIENCE THESIS
Department of Electronics and Communications Engineering
Program of Electronics

Supervisor
Asst. Prof. Zehra Gülru ÇAM TAŞKIRAN

July, 2023

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

MEMRISTOR BASED SYNAPSE DESIGN

A thesis submitted by Seda GÜNAKIN in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 31.07.2023 in Department of Electronics and Communications Engineering, Program of Electronics.

Asst. Prof. Zehra Gülru ÇAM TAŞKIRAN
Yildiz Technical University
Supervisor

Approved By the Examining Committee

Asst. Prof. Zehra Gülru ÇAM TAŞKIRAN, Supervisor
Yildiz Technical University _____

Prof. Dr. Burcu ERKMEN, Member
Yildiz Technical University _____

Asst. Prof. Nurullah ÇALIK, Member
Istanbul Medeniyet University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Memristor Based Synapse Design supervised by my supervisor, Asst. Prof. Zehra Gülru ÇAM TAŞKIRAN. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Seda GÜNAKIN

Signature

To my beloved family

ACKNOWLEDGEMENTS

I would like to thank my esteemed advisor Asst. Prof. Zehra Gülru Çam Taşkıran, who helped me in every way, shared her valuable information with me and did not spare her support.

I would also like to thank my dear friends who have always been supportive and have been with me throughout this process.

And lastly, I would like to thank my dear mother, father and sisters who have always stood by me and supported me throughout my life.

Seda GÜNAKIN

TABLE OF CONTENTS

LIST OF SYMBOLS vii

LIST OF ABBREVIATIONS viii

LIST OF FIGURES ix

LIST OF TABLES xi

ABSTRACT xii

ÖZET xiv

1 INTRODUCTION 1

1.1 Literature Review 1

1.2 Objective of the Thesis 2

1.3 Hypothesis 3

2 MEMRISTOR 4

3 NEURAL NETWORK STRUCTURE 9

3.1 Software Designed Network Structure 10

3.2 Hardware Designed Network Structure 11

3.3 Mathematical Analysis of Network Structure 15

3.4 Network Simulation Results 18

4 JOGLEKAR AND WOLF MEMRISTOR MODEL APPLICATION 24

4.1 Characteristic Analysis with Inputs 25

5 MEMRISTOR EMULATOR CIRCUIT APPLICATION 32

5.1 Characteristic Analysis with Inputs 32

6 RESULTS AND DISCUSSION 39

REFERENCES 41

A JOGLEKAR AND WOLF PSPICE MODEL 44

PUBLICATIONS FROM THE THESIS **46**

LIST OF SYMBOLS

y_o	Algorithmic Output
I_b	Bias Current
C	Capacitor
q	Charge
I-i	Current
Φ	Flux
δ_i	Gradient Coefficient from the Hidden Layer
δ_o	Gradient Coefficient from the Output Layer
M	Memristance
ΔR_{ij}	Memristance Change
t	Pulse Duration
R	Resistance
W	Synapse Weight
V-v	Voltage
ΔW	Weight Change

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
CMOS	Complementary Metal Oxide Semiconductor
CNN	Convolutional Neural Network
FPGA	Field Programmable Gate Array
HP Labs	Hewlett-Packard Laboratory
LSTM	Long-short Term Memory
MATLAB	Matrix Laboratory
MNN	Multilayer Neural Network
OTA	Operational Transconductance Amplifier
PSPICE	Personel Simulation Program with Integrated Circuit Emphasis
TEAM	Threshold Adaptive Memristor
TiO_2	Titanyum Dioksit
VLSI	Very Large Scale Integration

LIST OF FIGURES

Figure 2.1 Relationship between fundamental variables **4**
Figure 2.2 q-Φ curves of different type of memristors [1] **5**
Figure 2.3 Memristor realized by HP in nano scale [5] **6**
Figure 2.4 HP memristor model [5] **7**
Figure 2.5 An example of hysteresis curve [32] **8**
Figure 3.1 Relationship between fundamental variables **9**
Figure 3.2 Designed network structure **10**
Figure 3.3 Circuit design for feed forward structure **12**
Figure 3.4 Circuit design for backpropagation **13**
Figure 3.5 Circuit design of the synapse **14**
Figure 3.6 Simulation results for network outputs **18**
Figure 3.7 Activation function results **18**
Figure 3.8 Backpropagation error results of output **19**
Figure 3.9 Backpropagation error results for hidden layer **19**
Figure 3.10 Flow chart of the digital process **22**
Figure 4.1 Joglekar and Wolf window function for different p values **25**
Figure 4.2 SPICE circuit design with Joglekar and Wolf memristor model . **26**
Figure 4.3 Voltage graph of the sinusoidal input with high frequency **26**
Figure 4.4 Current graph of the sinusoidal input with high frequency **27**
Figure 4.5 Hysteresis curve of Joglekar and Wolf memristor model at 10 Hz **27**
Figure 4.6 Hysteresis curve of Joglekar and Wolf memristor model at 20 Hz **27**
Figure 4.7 Hysteresis curve of Joglekar and Wolf memristor model at 3 kHz **28**
Figure 4.8 Zoomed-in view of the pulse train created in MATLAB **29**
Figure 4.9 SPICE result of the voltage of the pulse train **29**
Figure 4.10 SPICE result of the current of the pulse train **30**
Figure 4.11 SPICE result showing change of memristance with pulse train . **30**
Figure 4.12 Zoomed-in memristance change **30**
Figure 5.1 Proposed memristance simulator circuit [1] **33**
Figure 5.2 Hysteresis lobes for different frequencies [1] **33**
Figure 5.3 Voltage graph of the sinusoidal input at 3kHz **34**
Figure 5.4 Current graph of the sinusoidal input at 3kHz **34**
Figure 5.5 Hysteresis curve of memristor emulator circuit at 100 Hz **35**

Figure 5.6 Hysteresis curve of memristor emulator circuit at 500 Hz **35**
Figure 5.7 Hysteresis curve of memristor emulator circuit at 3 kHz **36**
Figure 5.8 Voltage graph for applied pulse train **36**
Figure 5.9 Current graph for applied pulse train **37**
Figure 5.10 Memristance change with square wave **37**
Figure 5.11 Zoomed-in memristance change with square wave **38**

LIST OF TABLES

Table 3.1 Weights for minimum error values **11**
Table 3.2 Circuit parameters **13**
Table 3.3 Normalized memristor values **16**
Table 3.4 Algorithm and Simulation Results **20**
Table 3.5 Algorithm and Simulation Results(continued) **21**

ABSTRACT

Memristor Based Synapse Design

Seda GÜNAKIN

Department of Electronics and Communications Engineering
Master of Science Thesis

Supervisor: Asst. Prof. Zehra Gülru ÇAM TAŞKIRAN

Artificial neural networks have been used in many applications such as computer vision, natural language processing, voice or character recognition until today. As these tasks given to the network become more difficult, the structure of the network becomes more complex and the processing time increases as the parameters increase. For this reason, networks have started to be implemented as hardware. The memristor is the most frequently used element in the literature for synapse implementations due to its non-volatile memory feature and the convenience it provides in weighted summation process due to its definition relation.

In this study, a memristor based perceptron design to be used for classification has been created. The network proposed for this design has a multi-layered structure. The iris data set, which is widely used in the literature, was used for classification. The network with input layer, hidden layer and output layer is trained with backpropagation learning algorithm and the threshold function is used as the transfer function.

The proposed circuit design consists of feed forward and backpropagation blocks. The feed forward block performs the weighted multiplication operation using a memristor crossbar array, and the activation function using comparator blocks in a completely analog circuit. By finding the error values of the calculated outputs in the feed forward process, the training process of the network was carried out by backpropagation of these errors. The process is continued until the minimum error is found, and the weight updates required at the end of this process are carried out without the need for user intervention, thanks to an analog-digital hybrid circuit.

Applications were made with the Joglekar and Wolf memristor model, and an emulator circuit with memristive properties. The range in which these systems can achieve the desired weights has been determined by examining their *i*-*v* characteristics. It has been shown that their control can be achieved with pulse trains applied for a certain period of time. By examining the obtained characteristics, the memristor intervals required for weight updates were determined and the sensitivities of the memristors were calculated. The operability of the designed analog circuits has been verified by PSPICE simulations and it has been seen that the algorithmic results obtained with MATLAB and the simulation results are compatible.

Keywords: Memristor, emulator circuit, neural networks, backpropagation, multilayer perceptron

ÖZET

Memristör Temelli Sinaps Tasarımı

Seda GÜNAKIN

Eletronik ve Haberleşme Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Dr. Öğr. Üyesi Zehra Gülru ÇAM TAŞKIRAN

Yapay sinir ağları günümüze kadar bilgisayarlı görü, doğal dil işleme, ses veya karakter tanıma gibi pek çok uygulamada kullanılmıştır. Ağa verilen bu görevler zorlaştıkça ağın yapısı da karmaşıklaşmakta ve parametreler arttıkça işlem süresi uzamaktadır. Bu sebeple de ağlar donanımsal olarak gerçeklenmeye başlanmıştır. Memristör, uçucu olmayan hafıza özelliği ve tanım bağıntısı gereği ağırlıklı toplam işleminde sağladığı kolaylık sebebiyle sinaps gerçeklemeleri için literatürde en sık kullanılan eleman olarak yer almıştır.

Bu çalışmada da memristör temelli, sınıflama amacıyla kullanılacak bir algılayıcı tasarımı oluşturulmuştur. Bu tasarım için oluşturulan ağ çok katmanlı bir yapıya sahiptir. Sınıflandırma için literatürde yaygın olarak kullanılan iris veri seti kullanılmıştır. Giriş katmanı, gizli katman ve çıkış katmanı bulunan ağ geriye yayılımlı öğrenme algoritması ile eğitilmiş ve transfer fonksiyonu olarak eşik fonksiyonu kullanılmıştır.

Önerilen devre tasarımı ileri yol ve geri yayılım bloklarından oluşmaktadır. İleri yol bloğu ağırlıklı çarpım işlemini memristör çapraz çubuk dizisi kullanarak, aktivasyon fonksiyonunu ise karşılaştırıcı bloklarını kullanarak tamamen analog olarak gerçeklemektedir. İleri yolda hesaplanan çıkışların hata değerleri bulunarak bu hataların geri yayılımı ile ağın eğitim işlemi gerçekleştirilmiştir. Minimum hata bulunana kadar işleme devam edilerek bu sürecin sonunda gerekli olan ağırlık güncellemeleri analog-dijital hibrit bir devre sayesinde kullanıcı müdahalesine ihtiyaç duyulmadan gerçekleştirilmiş olur.

Joglekar ve Wolf memristör modeli ve memristif özellik gösteren bir emülatör devresi ile uygulamalar gerçekleştirilmiştir. Bu sistemlerin istenen ağırlıkları gerçekleyebilecekleri aralık *i-v* karakteristikleri incelenerek belirlenmiş ve kontrollerinin belirli bir süre uygulanan darbe katarları ile sağlanabileceği gösterilmiştir. Elde edilen karakteristikler incelenerek ağırlık güncellemeleri için gereken memristör aralıkları belirlenmiş ve memristörlere ait hassasiyetler hesaplanmıştır. Tasarlanan analog devrelerin çalışabilirliği PSPICE simülasyonları ile doğrulanmıştır ve MATLAB ile elde edilen algoritmik sonuçlar ile simülasyon sonuçlarının uyumlu olduğu görülmüştür.

Anahtar Kelimeler: Memristör, emülatör devresi, sinir ağları, geriye yayılım, çok katmanlı algılayıcı

1
INTRODUCTION

1.1 Literature Review

Passive and active electronic components are being used in the electronic circuits. Resistor, capacitor and inductor, which gives the relationship between basic quantities, are the most used passive circuit elements in electronic circuit designs. Leon O. Chua presented a two terminal passive electronic component that gives relationship between electric charge and magnetic flux as a fourth basic circuit element in 1971 [1]. Although the relationship between flux and charge of the memristor is linear, the *i*-*v* graph has a nonlinear characteristic on the contrary to resistor, capacitor and inductor which are linear. It is named memristor which stands for memory resistor. Depending on the magnetic flux applied to the memristor, the memristor resistance changes and even if the source is removed, the memristor maintains its recent resistance value. This shows that the memristor has permanent memory which called as non-volatility. The memristive behavior displayed by the memristor is actually a feature inherent in some materials in nature. However, since the memristive behavior is very weak in macro-scale structures, it has not been noticed until now. In studies where it was detected, the observed events were considered as abnormal behavior since they could not be explained [2–4]. In fact, memristive behavior is a behavior that has begun to be seen as effective at the micron scale. Chua also proposed the mathematical model of the memristor in his article and designed the circuit that gives the charge-flux relationship that may belong to the memristor [1]. Afterwards, Williams and his colleagues from HP Labs realized the memristor in 2008 [5]. After the physical memristor, many studies have taken place in the literature. As a recent device, memristor has advantages such as non-volatility, non-linearity, less power consumption, good scalability and CMOS integrity [6, 7] which makes it an important component. Therefore, it is used in many research areas such as memory systems [8, 9], neuromorphic applications [10], and chaotic circuits [11].

Because of the similarity of the memristor to the synapse, it is frequently used in

artificial neural network implementations. Firstly, Hudgkin and Huxley proposed a circuit to model axon membrane in 1952 [12]. The electrical current that occurs on the cell membrane is formed by the flow between ionic structures. In the presented model, the currents of potassium, sodium and chloride ions formed in the cell membrane are shown with an equivalent circuit using batteries and parallel conductivities. Also, Chua explained that sodium and potassium ion channels in an axon are actually time-varying conductivity, which is an example of memristive devices based on Hudgkin-Huxley neuron model. CMOS structures were generally used before memristor was included in synapse designs [13–15]. VLSI use began to participate in studies with Mead in 1980 [16]. Although VLSI systems are suitable for the parallel structure of ANN, they cannot be compatible with different ANNs. Also, they are not efficient because they are costly and take a lot of time. Therefore, more successful results have been obtained by using FPGA in synapse designs [17, 18]. With the increase in theoretical models mimicking the memristor, the use of memristor in synapse designs has increased. In the network structures created in these studies, memristors can be used together with CMOS or can only be composed of memristors [19–21].

After HP's discovery of the physical memristor, Snider performed spike timing dependent learning using a crossbar array [22]. Later in 2010, CNN was realized using the crossbar array by Carrasco et al. and model recognition studies were carried out [23]. Rose et al. realized artificial synapse implementation with a hybrid design of CMOS and memristor [24]. Querlioz et al. performed unsupervised learning by constructing a memristor-based neural network [25]. In 2016, Rosenthal et al. performed online gradient training by designing a fully analog network with 2 CMOS structures and 1 memristor and obtained results with close accuracy with the software implemented network [26]. Yakopcic et al. carried out on-chip training of crossbar-based multilayer neural networks. During the training, weight updates were performed using the backpropagation algorithm [27]. In 2018, Zhang et al. also performed backpropagation learning by designing a multi-layer network with memristors only [28]. As the use of memristor has become widespread in the studies, besides the implementation of different network structures and learning algorithms such as LSTM, Hopfield, application areas such as emotion classification, image recognition and classification have also varied [29–31].

1.2 Objective of the Thesis

With the developing technology, there is a need for hardware implementation of artificial neural networks designed as software. As the applications made with

artificial intelligence increase, the search for solutions to much more difficult problems has begun. For this reason, the tasks assigned to the network have become more difficult. As these tasks become more difficult, the structure of the network becomes more complex and the processing time increases as the parameters increase. For this reason, networks have started to be implemented as hardware. In this study, it is aimed to design a hardware memristor-based multilayer artificial neural network for classification. Learning in the network consisting of input layer, hidden layer and output layer is performed by backpropagation algorithm. The proposed circuit design consists of feed forward and backpropagation circuit blocks. In the feed forward block, a weighted sum was created by forming a crossbar array with memristors, and the activation function was performed using a comparator. The training of the network was carried out by updating the weights with the backpropagation of the errors obtained in the feed forward process. The intervals at which the necessary weight updates can be made were determined by examining the SPICE models of the memristors and the memristive emulator circuit in the literature by applying appropriate pulse train. While performing feed forward operations by making use of the memristors' ability to act as a resistor at high frequencies, weight updates are carried out with the pulse train with its non-volatility feature at low frequencies. The simulations were performed with PSPICE and the compatibility with the algorithmic results was observed.

1.3 Hypothesis

Hardware circuit design of an algorithmic network structure was made with the designed memristor based synapse. There are different processing blocks in this circuit structure, which performs the learning process with the backpropagation algorithm. With these operation blocks, circuit parts such as the activation function can be changed and adapted to the algorithmic design. They can be implemented with a commerical memristor , as well as with emulator circuits in the literature. Memristor models or emulators may have different resistance ranges according to their characteristics. By providing the appropriate pulse trains, the weight updates required for learning can be performed by using the approximate resistance ranges. These updates can be updated to obtain the closest values to the algorithmic results at different sensitivities according to the memristor type.

2
MEMRISTOR

Chua stated that the relationship between the 3 basic circuit elements is defined by the relations between current, voltage, charge and flux. Only 5 combinations are known of the 6 possible combinations between these four fundamental circuit variables. Based on this, Chua claimed that the sixth of these combinations would reveal the missing relationship in the circuit theory of electronics. The relationship between voltage and current gives the resistor, the relationship between charge and voltage gives the capacitor, the relationship between flux and current gives the inductor and the relationship between charge and flux gives the memristor as shown in Figure 2.1.

Resistor, inductor and capacitor, which are the three basic passive circuit elements, exhibit linear behavior according to relationship in Figure 2.1. Although the memristor is a memory resistor, the *i-v* graph is nonlinear unlike the resistor. In addition to this, unlike other circuit elements, the value of the memristor increases with the current flowing in one direction, while its value decreases when the current flows in the other direction. When the current flowing through the memristor is interrupted, the memristor retains its last value and exhibits non-volatile memory characteristics.

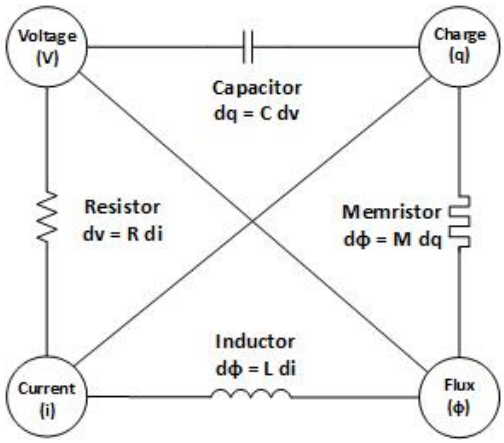

Figure 2.1 Relationship between fundamental variables

The q-Φ curve has a monotonically increasing property. The slope of this curve gives the memristance value. Hence, the memristance is always $M(q) \geq 0$ positive. Therefore, the memristor is a passive circuit element.

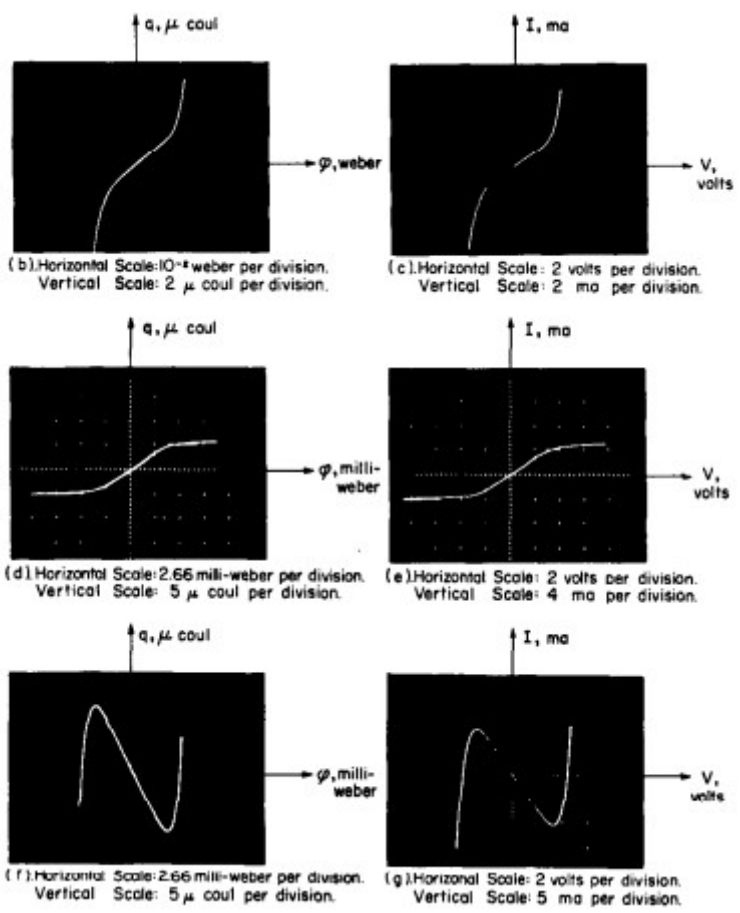

Figure 2.2 q-Φ curves of different type of memristors [1]

The memristor can be expressed in two ways according to the charge and flux. If the relationship between charge and flux is represented as a function of charge, it is expressed as a charge controlled memristor and defined as 2.1.

$$M(q) = \frac{df(q)}{dq} \tag{2.1}$$

If it is defined as a function of the flux, it is expressed as 2.2.

$$M(\Phi) = \frac{df(\Phi)}{d\Phi} \tag{2.2}$$

It is called a current controlled memristor, as in the HP memristor which defined as 2.3.

$$V(t) = R(t)i(t)$$
$$R(t) = \mathrm{R}_{ON}\frac{\omega(t)}{D} + \mathrm{R}_{OFF}(1 - \frac{\omega(t)}{D}) \tag{2.3}$$

where $\omega(t)$ is the width of the doped region, ***R(t)*** is the memristance, ***D*** is the thickness of the TiO_2, R_{ON} is internal low memristance when the memristor is completely doped and R_{OFF} is internal high memristance when the memristor is completely undoped [5].

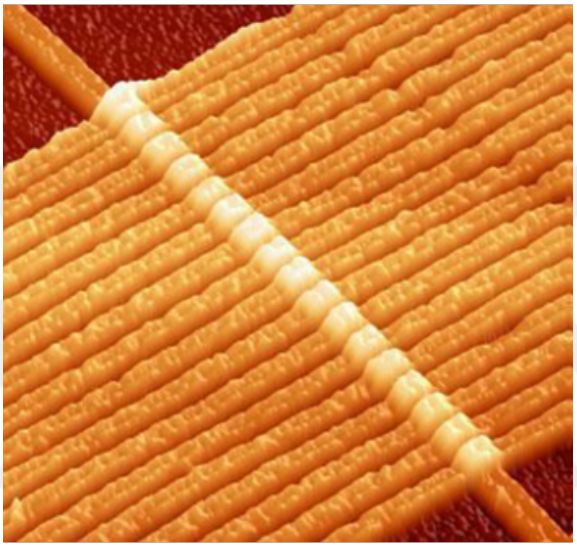

Figure 2.3 Memristor realized by HP in nano scale [5]

Williams et al. used TiO_2 to obtain the memristor in HP Labs. TiO_2 is a semiconductor like silicon with a high resistance which can become a conductor with other elements. The additives used to make TiO_2 conductive are not stable under a strong electric field and tend to drift in the current direction. Although this mobility is actually detrimental to transistors, it is also what drives the memristor. Applying an initial voltage to a thin layer of TiO_2 with additive elements on one side will cause these elements to move towards the other side with pure TiO_2 which lower the resistance. The elements will return to their places by applying a current in the opposite direction which increase the resistance again. When the power supply is interrupted, the elements stop moving and the memristor remains in a high or low resistance state [5].

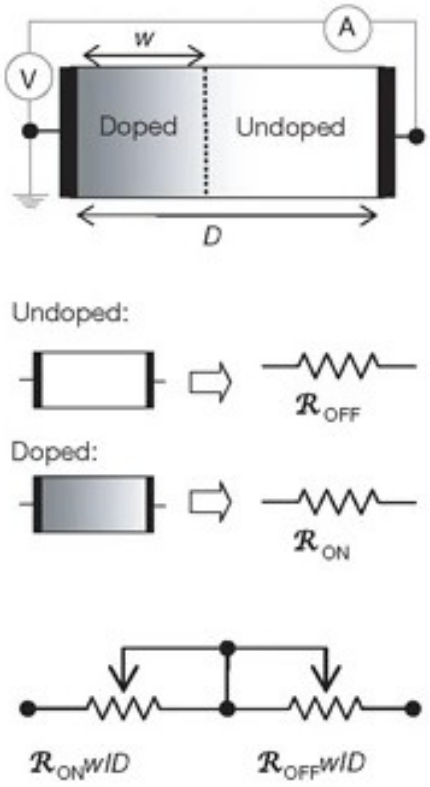

Figure 2.4 HP memristor model [5]

After the discovery of the memristor, studies on the memristor have increased. Different models have been proposed by adding window functions to the linear drift model used by the HP. Also, models such as nonlinear drift model, Simmons tunnel barrier model and TEAM model have also been proposed to eliminate the deficiencies in these models. In addition to these studies, memristor emulator circuits with memristive properties have also emerged. However, there has been some controversy because the exact limits for an element or system to be called a memristor are not clear. Therefore, Adhikari et al. stated that in order for a device to be called a memristor, it must have some features [32].

Fingerprints of memristor

- When the memristor is driven with sinusoidal current source, it must have a hysteresis loop in the *i-v* plane and this curve must pass through the origin. That is, both the current and voltage of the memristor must be zero at the same time.
- The hysteresis lobe area depends on the frequency of the bipolar periodic input signal. As the input frequency increases, the area of the hysteresis lobe decreases.

- At low frequency, the memristor value should be variable. As the frequency value increases, the hysteresis curve should turn into a straight line and the memristor value should be constant.

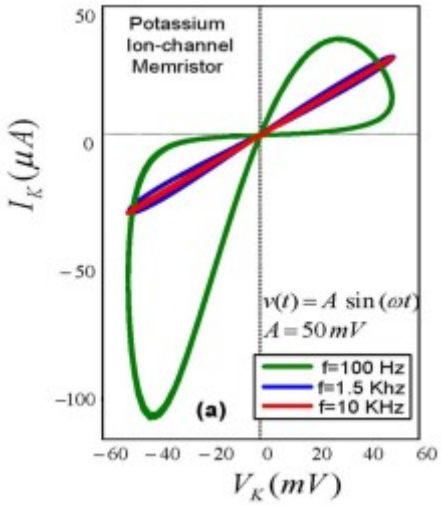

Figure 2.5 An example of hysteresis curve [32]

3
NEURAL NETWORK STRUCTURE

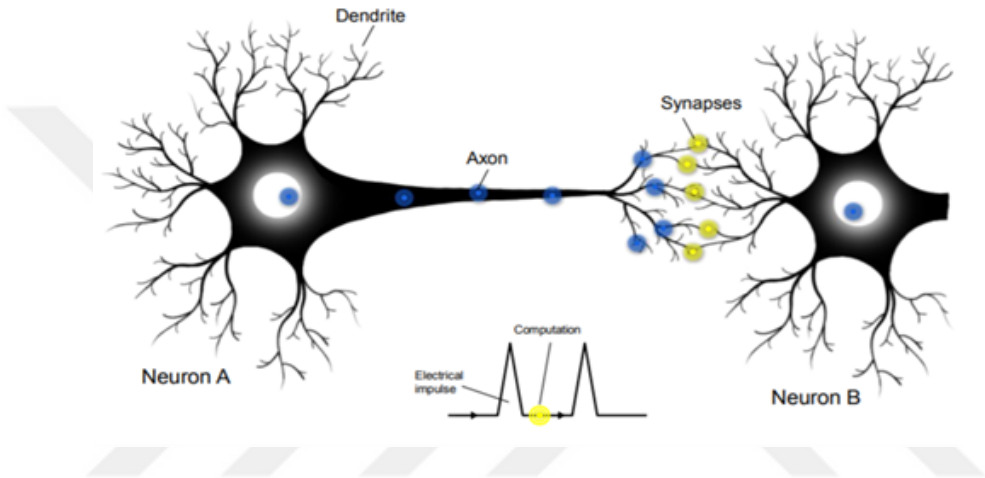

Figure 3.1 Relationship between fundamental variables

As it is known, there are nerve cells called neurons in our neural system and there are more than 100 billion neurons in the human brain and an average of 1000 synapses in each neuron. In order to model a single synapse, it is necessary to know how the synapse works. Neurons consist of parts such as axon, dendrite and synapse.

Axon : It is the electrically active body where the output pulses are produced and the transmission on the body is unidirectional. It is the system output.

Dendrites: They are electrically passive arms that collect signals from other cells. It is the system input.

Synapse: It provides the connection of axons of cells with dendrites of other cells.

When the signal transmission from the neuron A to the neuron B is examined in Figure 3.1, the signal from the neuron A is sent to the synapses via the axon and reaches the dendrites of the neuron B. Neuron B, which has a signal that has already arrived, compares the both stimuli. Then, these signals are sent to the main part of the neuron B over the dendrites for the evaluation of both the signal from the neuron

A and the signal in the neuron B. During this information transfer, calculations are made and a single value is received by the neuron B. This working structure of synapses was examined and imitated and artificial neural networks were created.

3.1 Software Designed Network Structure

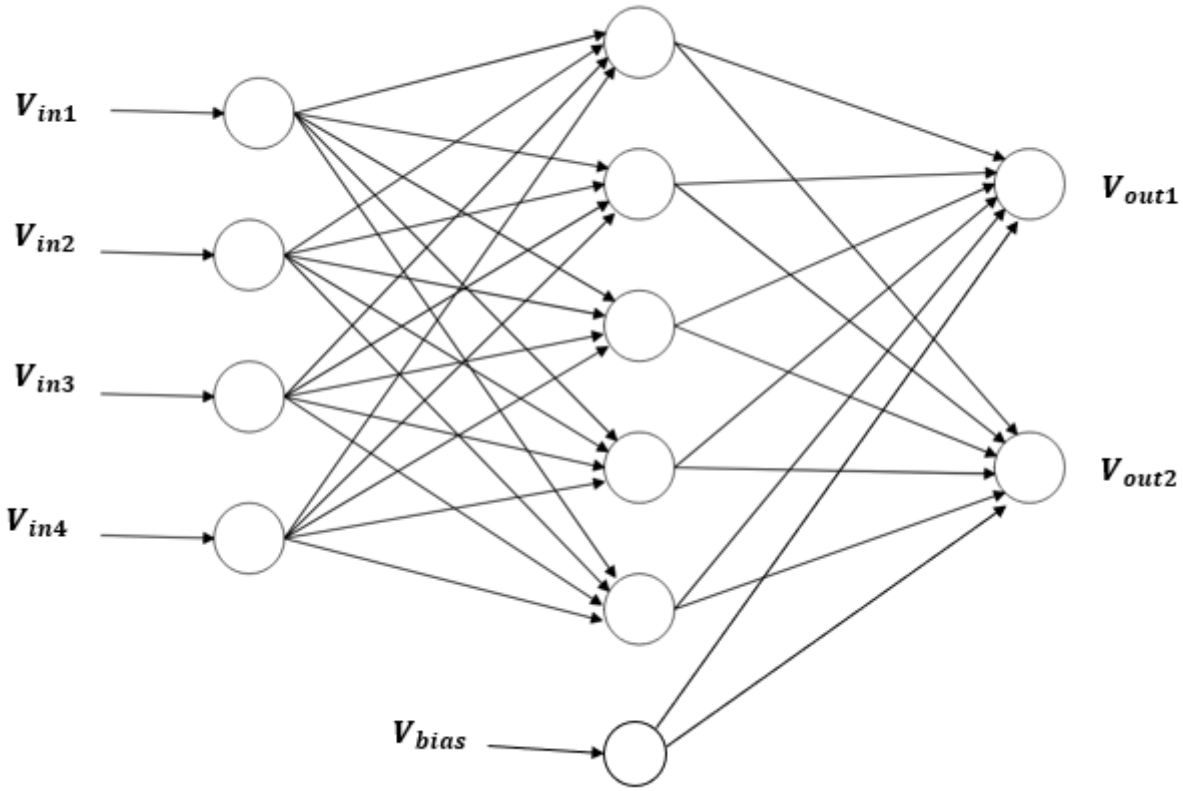

Figure 3.2 Designed network structure

A network was designed as shown in Figure 3.2 to classify the iris dataset, which is widely used in the literature. The designed network has 2 layers. In the input layer, 4 neurons are used to represent the input number of the dataset, and 2 neurons are used in the output layer to represent the 3 classes in binary order. The number of hidden layer neurons was determined as 5 by trial and error method. A bias input has also been added to the hidden layer. The inputs of the circuit are voltage inputs with normalized values. The software normalization process compresses the input values to the range of [0, +1]. Backpropagation learning algorithm was used to perform the learning process and a linear function was used as the activation function. Half of the data set was used for training and the other half for testing purposes, and the weights obtained for the network providing 74.67% test success are given in Table 3.1.

The test performance, which is low compared to the studies in the literature, can be improved depending on the design of the network. However, the aim of this thesis is not to design a high performance artificial neural network, but to implement a designed neural network as hardware. For this reason, the network designed as software is accepted as it is, and only a close accuracy hardware implementation is aimed.

Table 3.1 Weights for minimum error values

Layer 1	$\begin{pmatrix} -0.060 & -0,192 & -0,150 & 0,444 \\ -0,486 & -0,688 & 0,578 & -0,187 \\ 0,182 & 0,210 & 0,522 & 0,533 \\ -0,048 & 0,380 & 0,491 & 0,087 \\ -0,030 & -0,280 & 0,103 & -0,316 \end{pmatrix}$
Layer 2	$\begin{pmatrix} 0,374 & -0,495 \\ 0,459 & -0,214 \\ 0,367 & 0,733 \\ 0,420 & -0,426 \\ 0,482 & -0,013 \\ 0,549 & 0,398 \end{pmatrix}$

3.2 Hardware Designed Network Structure

In this study, it is aimed to realize the hardware of the network structure designed as software. A crossbar array structure is formed with memristors in [28]. In order to realize the network structure given above as hardware, memristors, resistors and operational amplifiers are used in the circuit [28]. The circuit consists of 2 blocks. Since the backpropagation learning algorithm is used for the learning process, the first block performs feed forward operations. In order to perform the weighted sum operation in this block, MNN was obtained by creating a two-layer structure using the crossbar array. Also, one bias is added in the second layer of MNN. In order to determine the output in binary, the obtained output is passed through a threshold function. An opamp comparator circuit is used to generate logic '0' for activation function output $y_0 \leq 0.5$ and logic '1' for $y_0 > 0.5$. Since input is a sinusoidal signal, the activation function output is obtained by comparing the output signal with the threshold value by using the amplitude detector. Feed forward block is designed as in Figure 3.3.

The second block performs backpropagation. Backpropagation algorithm is based on the principle of updating the weights by reflecting the calculated error value to the influencing weights. The comparator's output is used to estimate the target

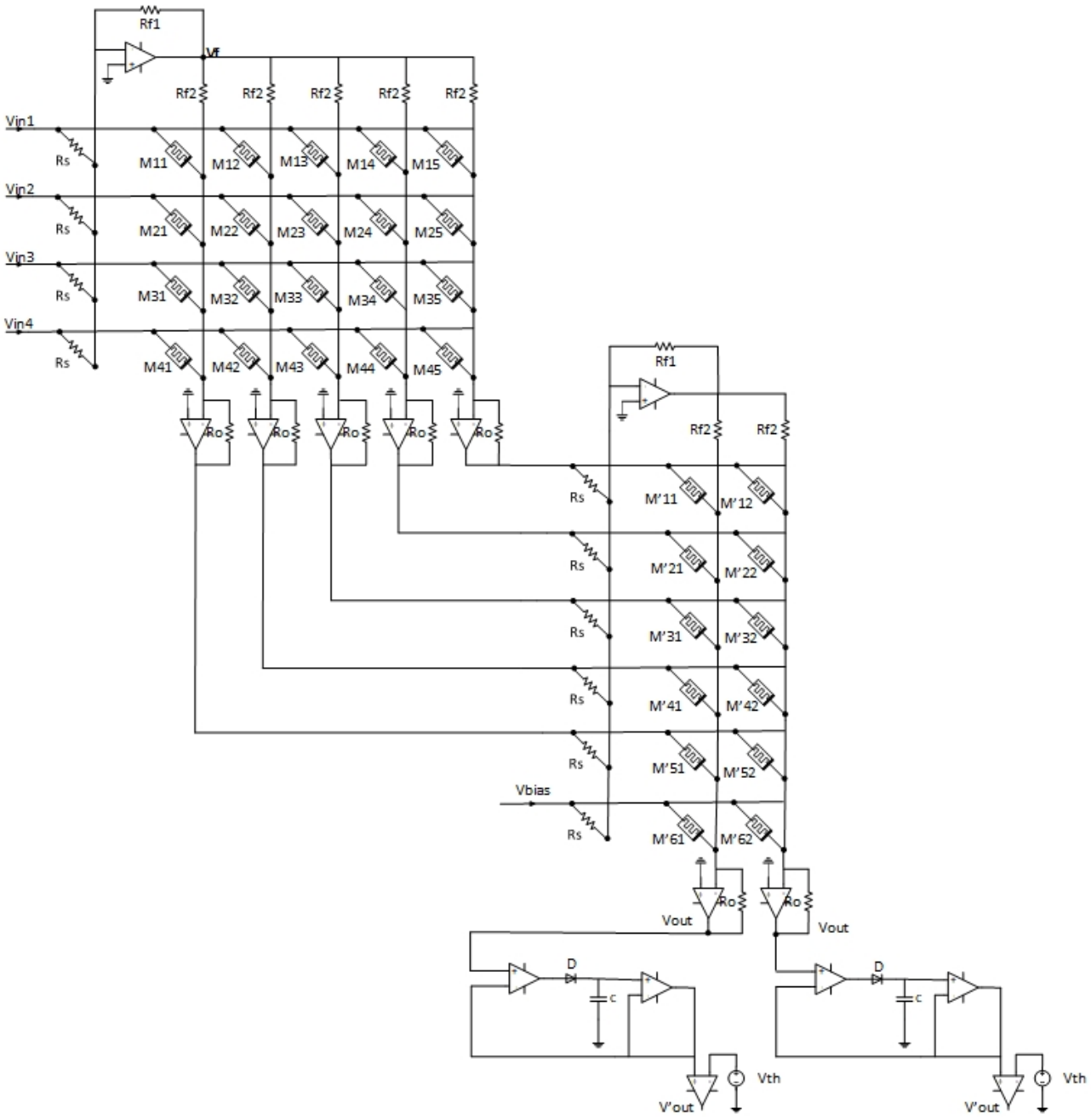

Figure 3.3 Circuit design for feed forward structure

output needed for the next input, so the weights are updated to minimize the error at the output. Thus, the error must be calculated in the second block. Therefore, a differential circuit is designed by using the output of the feed forward process as an input. Error values are obtained with the output of this circuit. However, these errors need to be reflected back. Since the derivative of the activation function for the errors reflected on the output layer is 1, the δ_o values here are equal to the error values. A crossbar array is used again to reflect the δ_o values in the output layer back to obtain δ_i values. In this array, δ_o creates inputs for previous layer and the number of neurons in the layer where the error is reflected creates the outputs. For this reason, a crossbar array was created by taking the transpose of the last layer with same memristor values. These memristors are used for weight updates in the synapse. Second block design is given in Figure 3.4.

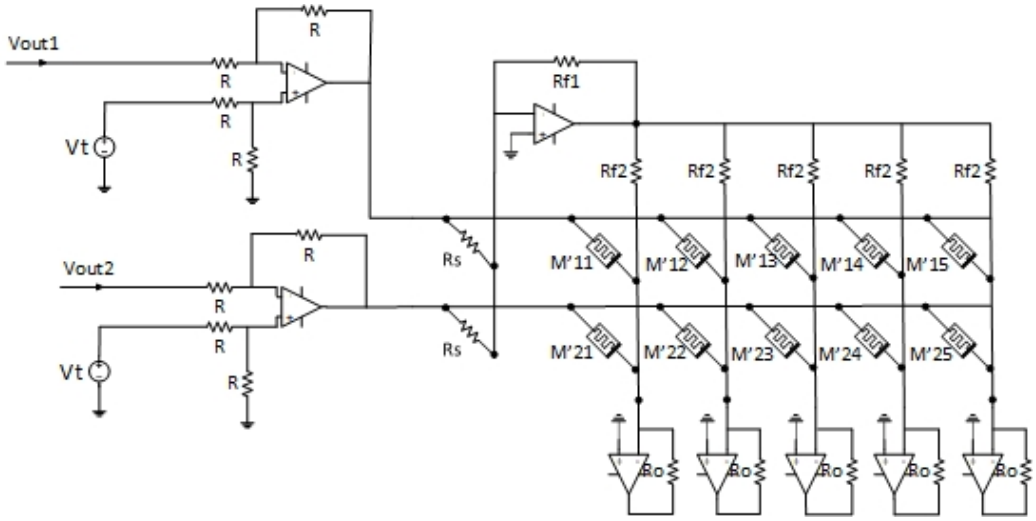

Figure 3.4 Circuit design for backpropagation

Required component values are given in Table 3.2

Table 3.2 Circuit parameters

Parameters for MNN	Values
$R_o[k\Omega]$	2
$R_s[k\Omega]$	1
$R_{f1}[k\Omega]$	2
$R_{f2}[k\Omega]$	2
$R[k\Omega]$	2
V_H	1
V_L	0
$V_{th}[V]$	0.5
$V[nF]$	100

Whole circuit design belongs to the synapse is given in Figure 3.5.

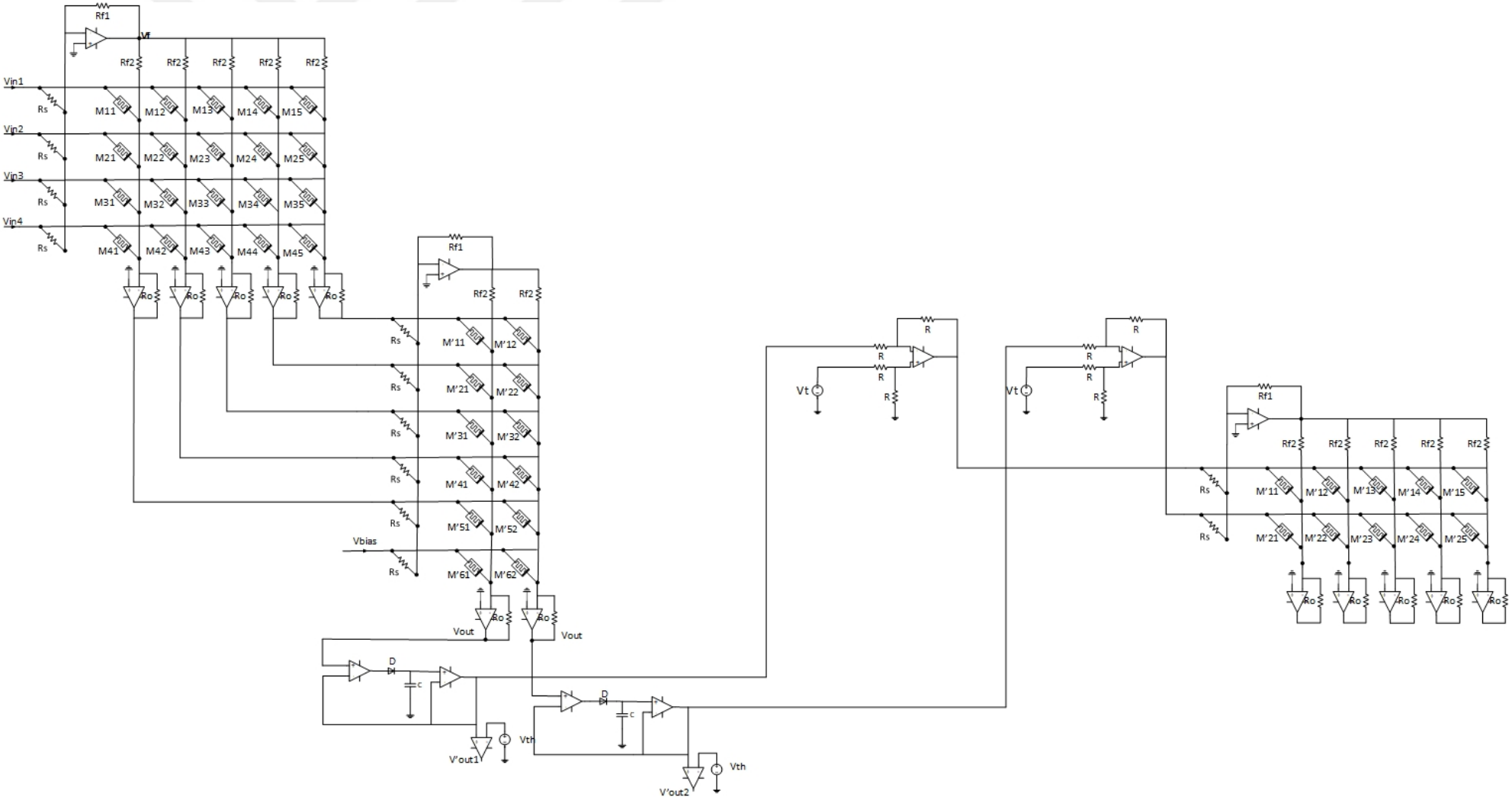

Figure 3.5 Circuit design of the synapse

3.3 Mathematical Analysis of Network Structure

3.3.1 Calculations for Feedforward Structure

The feed forward circuit of the hardware implemented network is given in Figure 3.3. The inputs of the circuit are voltage inputs with normalized values. The weights needed for the weighted summation process were created with the conductivity values of the memristor elements. This operation performed by the feed forward block is provided with the crossbar arrays.

In a learning system with k iterations, the inputs and outputs are 2 vectors with dimensions M and N, named $V_{in}^{(k)}$ and $V_{out}^{(k)}$. The estimator is when the weights are $M \times N$ matrix

$$\mathrm{V}_{out}^{(k)} = \mathrm{W}^{(k)}\mathrm{V}_{in}^{(k)} \tag{3.1}$$

or

$$\mathrm{V}_{outj}^{(k)} = \sum_{i=1}^{M} \mathrm{W}_{ij}^{(k)}\mathrm{V}_{ini}^{(k)} \tag{3.2}$$

where i = 1,2,. . . ,M and j = 1,2,. . . ,N.

The presented synapse model is implemented using a single crossbar structure consisting of memristors and a constant term circuit consisting of R_s resistor. R_{ij} shows the conductivity of the memristor between the ith row and j^{th} column. V_{ini} represents the input voltage applied to the ith row. According to Kirchoff's laws at V_f

$$\mathrm{V}_f = -\sum_{i=1}^{M} \frac{\mathrm{R}_{f1}}{\mathrm{R}_s}\mathrm{V}_{ini}. \tag{3.3}$$

The voltages are summed with an operational amplifier to obtain the output voltage of the ith column as in 3.4.

$$\mathrm{V}_{outj} = -\sum_{i=1}^{M} \frac{\mathrm{R}_o}{\mathrm{R}_{ij}}\mathrm{V}_{ini} \cdot \frac{\mathrm{R}_o}{\mathrm{R}_{f2}}\mathrm{V}_f \tag{3.4}$$

Combining 3.3 with 3.4, V_{outj} can be obtained as 3.5

$$\mathrm{V}_{outj} = \sum_{i=1}^{M} \mathrm{R}_o \cdot \left(\frac{\mathrm{R}_{f1}}{\mathrm{R}_s \mathrm{R}_{f2}} - \frac{1}{\mathrm{R}_{ij}} \right) \mathrm{V}_{ini} \tag{3.5}$$

Since $R_{f1} = R_{f2}$, the weight of the synapse is

$$\mathrm{W}_{ij} = \mathrm{R}_o \cdot (\mathrm{G}_s - \mathrm{G}_{ij}) \,. \tag{3.6}$$

A threshold function is used given in 3.7. The comparator decides the output of the jth column should be activated or not.

$$\mathrm{V'}_{outj} = \begin{cases} \mathrm{V}_H & , \mathrm{V}_{outj} > 0.5 \\ \mathrm{V}_L & , \mathrm{V}_{outj} \leq 0.5 \end{cases} \tag{3.7}$$

By using 3.5 and Table 3.1 memristor values can be calculated. In the continuation of the thesis, 2 different memristor applications were carried out in order to make weight updates. Normalized memristor values obtained to realize different applications as in Table 3.3 for the designed network.

Table 3.3 Normalized memristor values

Layer 1[$k\Omega$]	$\begin{pmatrix} 0,97 & 0,91 & 0,93 & 1,29 \\ 0,81 & 0,74 & 1,41 & 0,92 \\ 1,10 & 1,12 & 1,35 & 1,36 \\ 0,98 & 1,24 & 1,33 & 1,05 \\ 0,99 & 0,88 & 1,05 & 0,86 \end{pmatrix}$
Layer 2[$k\Omega$]	$\begin{pmatrix} 1,23 & 0,80 \\ 1,30 & 0,90 \\ 1,23 & 1,58 \\ 1,27 & 0,82 \\ 1,32 & 0,99 \\ 1,38 & 1,25 \end{pmatrix}$

3.3.2 Calculations for Backpropagation

During network training, the weights of the network are randomly initialized in forward propagation. However, it is important to find the right weights to reach the target output. For this reason, it is desired to obtain optimum results by updating the weights to the smallest error value. In this study, the weight update was performed with the backpropagation learning algorithm. After the forward training, the error

value at the output is calculated. It is tried to reduce the error by backpropagation of this error value. The weights are updated until the minimum error value is reached by forward propagation with the new weights obtained.

The output of the estimator $V_{out} = WV_{in}$ estimates the target output V_T for the new inputs of V_{in}. W values, which are the synapse weights, are updated with a training set of length of K_0 to reduce the error between outputs and target outputs. Error vector is

$$\mathrm{E}_j^{(k)} = \mathrm{V}_{Tj}^{(k)} - \mathrm{V'}_{outj}^{(k)} \tag{3.8}$$

Backpropagation algorithm is based on the principle of updating the weights by reflecting the calculated error value on the influencing weights. Therefore, the derivative of the output errors is passed from the output layer to the hidden layer. As a result of this process, if the jth neuron is in the output layer, δ values are calculated as

$$\delta_{ij} = \mathrm{E}_j^{(k)} \frac{\partial \mathrm{V'}_{outj}^{(k)}}{\partial \mathrm{V}_{outj}^{(k)}} \tag{3.9}$$

If the error is reflected from the hidden layer,

$$\delta_{ij} = \frac{\partial \mathrm{V'}_{outj}^{(k)}}{\partial \mathrm{V}_{outj}^{(k)}} \sum_p \delta_{jp} \mathrm{W}_{jp}^{(k)} \tag{3.10}$$

where p is the hidden layer neuron number. To update the weights from *W(n)* to *W(n+1)*,

$$\Delta \mathrm{W}_{ij}(n+1) = \eta \delta_{ij} \partial \mathrm{V'}_{outp} + \Delta \mathrm{W}_{ij}(n) \tag{3.11}$$

where η is learning rate.

3.4 Network Simulation Results

The simulations of the outputs of the multilayer network structure were performed with PSPICE. MATLAB was used for network design to perform the learning process in software. As an example test data, [0.23V, 0.39V, 0.14V, 0.13V] values were applied to the inputs. The values to be taken at the output algorithmically are calculated as [0.475V, 0.691V]. In response to these algorithmic calculations, the outputs taken in the proposed circuit are shown in Figure 3.6.

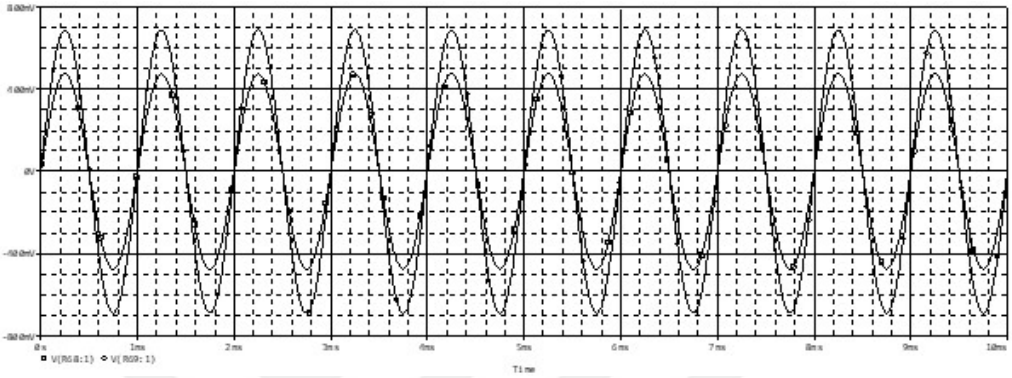

Figure 3.6 Simulation results for network outputs

The obtained outputs were passed through the threshold function and the target outputs obtained as seen in Figure 3.7. When the data set is examined, the example of these inputs belongs to the logic [0, 1] class. The simulation results in Figure 3.7 are also compatible with these target values.

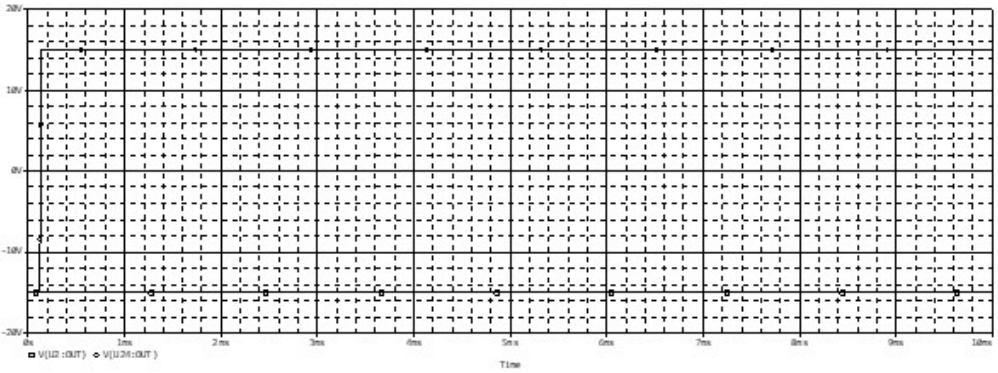

Figure 3.7 Activation function results

Backpropagation operations are performed using the outputs obtained in the feed forward operations. Error values are calculated using network outputs and target outputs. These calculated error values are equal to the δ_o values of the output layer since the derivative of the activation function is 1. Graphics of calculated error values are shown in Figure 3.8.

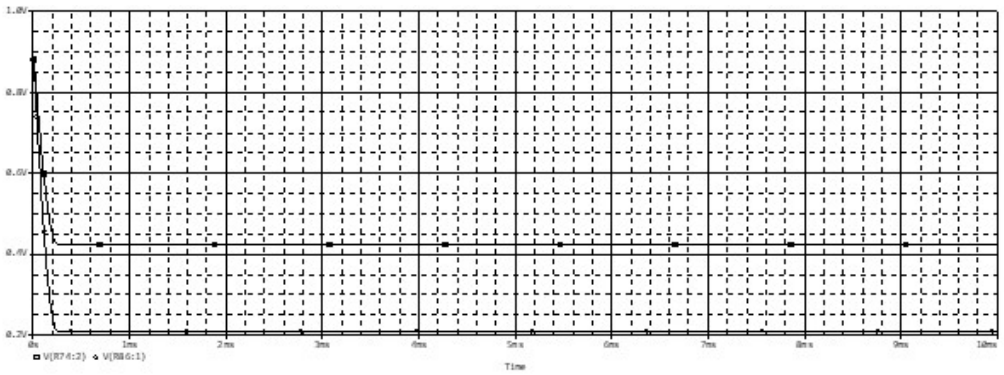

Figure 3.8 Backpropagation error results of output

In order to calculate the error values in the hidden layer, the δ_o values in the output layer must be used in backpropagation. The graph of the δ_i values showing the 5 neuron cells in the hidden layer while passing from the output layer to the hidden layer is shown in Figure 3.9.

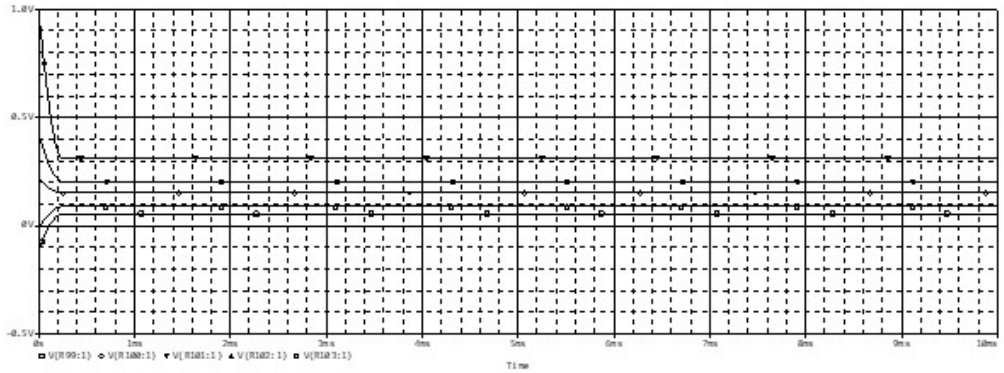

Figure 3.9 Backpropagation error results for hidden layer

Example graphics of network outputs are given above. Algorithmic and simulation outputs of the test set applied to the circuit are compared in Table 3.4 and Table 3.5. 0.66% mean square error was calculated using the data obtained as a result of this comparison.

Table 3.4 Algorithm and Simulation Results

Inputs	**Algorithmic Outputs**	**Simulation Outputs**
V_{in1} V_{in2} V_{in3} V_{in4}	y_{o1} y_{o2}	V_{out1} V_{out2}
0.23V 0.39V 0.14V 0.13V	0.475V 0.691V	0.475V 0.689V
0.17V 0.43V 0.16V 0.13V	0.474V 0.740V	0.473V 0.739V
0.35V 0.72V 0.19V 0.20V	0.389V 0.977V	0.391V 0.975V
0.26V 0.54V 0.16V 0.13V	0.420V 0.817V	0.419V 0.816V
0.23V 0.43V 0.16V 0.10V	0.454V 0.716V	0.455V 0.717V
0.21V 0.54V 0.17V 0.13V	0.432V 0.828V	0.428V 0.828V
0.10V 0.39V 0.10V 0.10V	0.434V 0.713V	0.435V 0.711V
0.41V 0.90V 0.16V 0.20V	0.281V 1.124V	0.284V 1.121V
0.28V 0.57V 0.14V 0.17V	0.410V 0.853V	0.411V 0.849V
0.28V 0.68V 0.16V 0.17V	0.370V 0.951V	0.375V 0.947
0.28V 0.65V 0.16V 0.20V	0.404V 0.927V	0.404V 0.928V
0.28V 0.50V 0.19V 0.23V	0.509V 0.806V	0.510V 0.802V
0.26V 0.39V 0.17V 0.13V	0.498V 0.687V	0.496V 0.683V
0.30V 0.57V 0.16V 0.13V	0.402V 0.840V	0.404V 0.834V
0.19V 0.46V 0.17V 0.13V	0.466V 0.768V	0.465V 0.761V
0.35V 0.54V 0.16V 0.20V	0.452V 0.814V	0.453V 0.814V
0.37V 0.83V 0.14V 0.13V	0.272V 1.052V	0.267V 1.050V
0.26V 0.46V 0.11V 0.13V	0.423V 0.751V	0.420V 0.744V
0.23V 0.43V 0.16V 0.10V	0.454V 0.716V	0.455V 0.718V
0.28V 0.54V 0.16V 0.13V	0.419V 0.812V	0.418V 0.812
0.15V 0.14V 0.13V 0.17V	0.606V 0.493V	0.607V 0.495
0.26V 0.57V 0.17V 0.27V	0.483V 0.885V	0.486V 0.880V
0.21V 0.39V 0.14V 0.17V	0.496V 0.706V	0.496V 0.704V
0.17V 0.46V 0.14V 0.13V	0.447V 0.772V	0.445V 0.766V
0.26V 0.50V 0.14V 0.13V	0.427V 0.784V	0.423V 0.781V
0.57V 0.46V 0.59V 0.57V	0.978V 0.802V	0.983V 0.794V
0.37V 0.14V 0.51V 0.50V	1.05V 0.536V	1.04V 0.536V
0.41V 0.32V 0.59V 0.50V	1.016V 0.690V	1.020V 0.689V
0.23V 0.17V 0.41V 0.40V	0.913V 0.571V	0.912V 0.567V
0.30V 0 .28V 0.50V 0.53V	0.992V 0.690V	0.992V 0.685
0.46V 0.39V 0.54V 0.57V	0.984V 0.761V	0.985V 0.758V
0.50V 0.35V 0.61V 0.53V	1.034V 0.711V	1.031V 0.704V
0.63V 0.43V 0.57V 0.53V	0.965V 0.745V	0.960V 0.743V
0.43V 0.28V 0.53V 0.40V	0.939V 0.625V	0.941V 0.622V
0.39V 0.21V 0.50V 0.43V	0.971V 0.578V	0.967V 0.575V
0.50V 0.32V 0.51V 0.50V	0.961V 0.668V	0.958V 0.667V
0.50V 0.32V 0.61V 0.47V	1.016V 0.661V	0.015V 0.662V
0.61V 0.39V 0.57V 0.53V	0.982V 0.718V	0.980V 0.713V
0.63V 0.39V 0.66V 0.63V	1.093V 0.737V	1.096V 0.736V
0.41V 0.25V 0.44V 0.40V	0.894V 0.596V	0.890V 0.598V
0.37V 0.17V 0.47V 0.40V	0.950V 0.542V	0.951V 0.537V
0.48V 0.28V 0.67V 0.60V	1.142V 0.670V	1.144V 0.665V

Table 3.5 Algorithm and Simulation Results(continued)

Inputs	**Algorithmic Outputs**	**Simulation Outputs**
V_{in1} V_{in2} V_{in3} V_{in4}	y_{o1} y_{o2}	V_{out1} V_{out2}
0.48V 0.54V 0.59V 0.60V	0.965V 0.896V	0.965V 0.894V
0.54V 0.14V 0.57V 0.50V	1.091V 0.497V	1.089V 0.495V
0.37V 0.21V 0.51V 0.50V	1.016V 0.601V	1.015V 0.595V
0.50V 0.39V 0.60V 0.53V	1.007V 0.743V	1.007V 0.745V
0.26V 0.14V 0.41V 0.40V	0.930V 0.533V	0.927V 0.533V
0.41V 0.39V 0.54V 0.47V	0.934V 0.745V	0.934V 0.739V
0.52V 0.35V 0.56V 0.50V	0.975V 0.696V	0.980V 0.691V
0.41V 0.32V 0.53V 0.50V	0.975V 0.689V	0.976V 0.689V
0.43V 0.28V 0.67V 0.70V	1.198V 0.706V	1.197V 0.703V
0.54V 0.35V 0.74V 0.67V	1.195V 0.738V	1.196V 0.733V
0.83V 0.39V 0.89V 0.77V	1.323V 0.734V	1.320V 0.735V
0.77V 0.35V 0.84V 0.67V	1.259V 0.689V	1.255V 0.691V
0.75V 0.61V 0.81V 0.90V	1.243V 0.982V	1.242V 0.983V
0.57V 0.28V 0.70V 0.70V	1.214V 0.676V	1.213V 0.671V
0.41V 0.21V 0.66V 0.73V	1.239V 0.654V	1.239V 0.656V
0.57V 0.46V 0.70V 0.83V	1.199V 0.873V	1.197V 0.869V
0.86V 0.68V 0.90V 0.80V	1.216V 0.998V	1.217V 0.993V
0.48V 0.10V 0.66V 0.57V	1.201V 0.498V	1.204V 0.496V
0.39V 0.32V 0.64V 0.73V	1.179V 0.757V	1.181V 0.755V
0.54V 0.28V 0.64V 0.67V	1.156V 0.671V	1.157V 0.667V
0.75V 0.46V 0.80V 0.67V	1.179V 0.791V	1.175V 0.793V
0.50V 0.39V 0.64V 0.67V	1.107V 0.779V	1.107V 0.771V
0.75V 0.39V 0.77V 0.60V	1.157V 0.708V	1.156V 0.703V
0.90V 0.68V 0.86V 0.73V	1.148V 0.969V	1.145V 0.971V
0.54V 0.32V 0.67V 0.57V	1.108V 0.678V	1.105V 0.678V
0.86V 0.39V 0.81V 0.83V	1.305V 0.745V	1.306V 0.744V
0.57V 0.43V 0.73V 0.67V	1.150V 0.798V	1.152V 0.798V
0.68V 0.43V 0.71V 0.77V	1.188V 0.798V	1.186V 0.798V
0.68V 0.43V 0.67V 0.83V	1.192V 0.815V	1.188V 0.813V
0.66V 0.46V 0.79V 0.83V	1.258V 0.855V	1.259V 0.845V
0.63V 0.39V 0.69V 0.83V	1.220V 0.793V	1.223V 0.789V
0.59V 0.39V 0.69V 0.73V	1.170V 0.777V	1.171V 0.772V
0.46V 0.39V 0.67V 0.67V	1.129V 0.790V	1.131V 0.786V

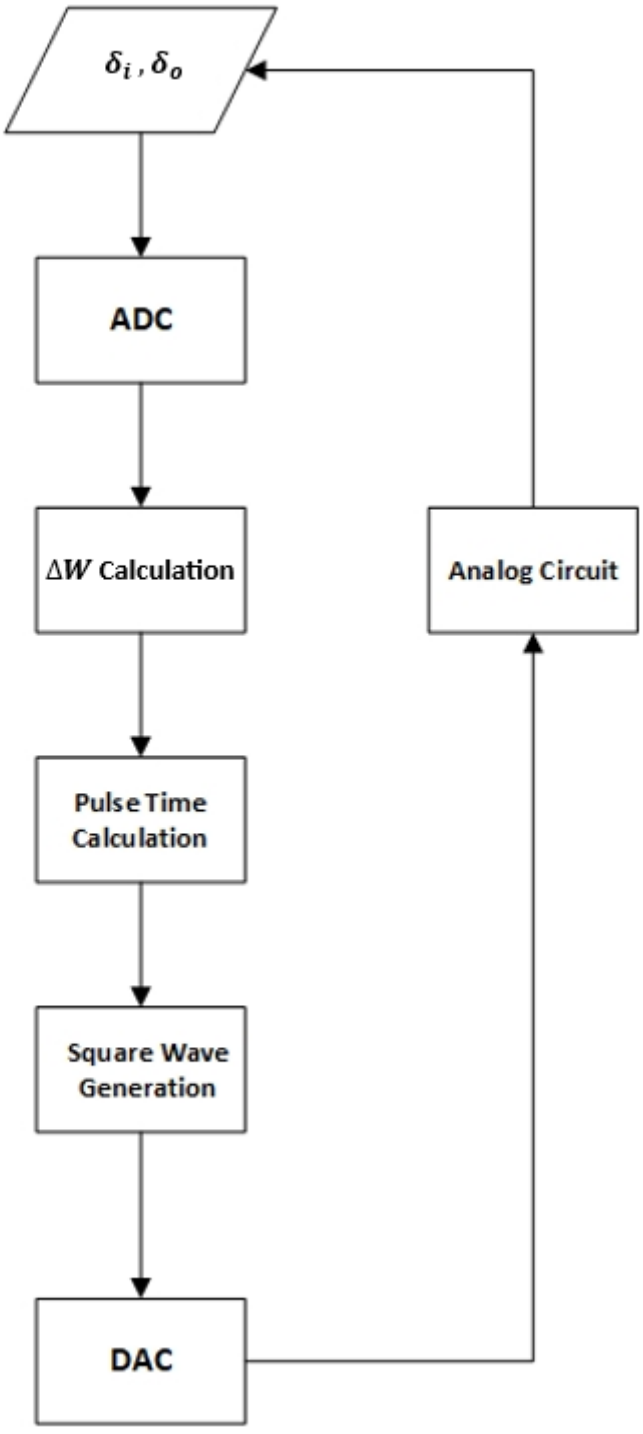

Figure 3.10 Flow chart of the digital process

Up to this point, the operations have been performed as analog and the calculations can be performed digitally have been obtained with MATLAB. However, to obtain the digital part the δ values taken from the analog circuit can be transferred to a microprocessor using ADC. Weight changes can be calculated using 3.12 and 3.13.

$$\Delta \mathrm{W}_i = \eta \delta_i \mathrm{V}_{ini} \tag{3.12}$$

$$\Delta \mathrm{W}_o = \eta \delta_o \mathrm{y}_i \tag{3.13}$$

Using these change values, the pulse durations to be applied to the memristor can be calculated with 4.2 and 5.1. Using these durations, square waves with appropriate amplitude and frequency can be generated and these pulses can be applied to the analog circuit with the DAC. The flow chart for the digital process of the pulses to be applied to the memristor is given in Figure 3.10.

4
JOGLEKAR AND WOLF MEMRISTOR MODEL APPLICATION

With the discovery of the HP team in 2008, memristors started to show up in the studies more frequently. The HP memristor model given by Williams et al. has been made with the assumption that the ion movements are linear. It has been claimed that the oxygen ions move with equal velocity as the doped region expands or contracts, regardless of the current direction, until the memristor saturates during charge flow [33]. In reality, even under a uniform electric field, the ion movements do not show a homogeneous distribution. Since the homogeneous dispersion of the ions mentioned in the linear ion drift velocity model does not match the reality, the HP memristor model is considered invalid and new memristor models are created with window functions with nonlinear ion drift velocity are developed to solve this problem. These functions model the nonlinear motion or drift of ions or regions within the memristor in memristor models. These window functions are multiplied by the memristor current and some memristor parameters and added to the memristor model to determine the drift rate. In 2009, Joglekar and Wolf proposed a new memristor model by adding a window function to the linear drift model [34]. The function of the limit values they determined as f(0) = f(1) = 0 is given as

$$\mathrm{f(x)} = 1 - (2\mathrm{x} - 1)^{2p} \tag{4.1}$$

Figure 4.1 shows the graph of the proposed window function for different p values. It looks like a rectangle with rounded edges on both sides for large values of the constant p. That is, the function starts from zero at the left boundary and rapidly rises to an almost constant value, and then decreases rapidly to zero as it approaches the right boundary. In other words, the nonlinear drift velocity becomes zero at the boundaries.

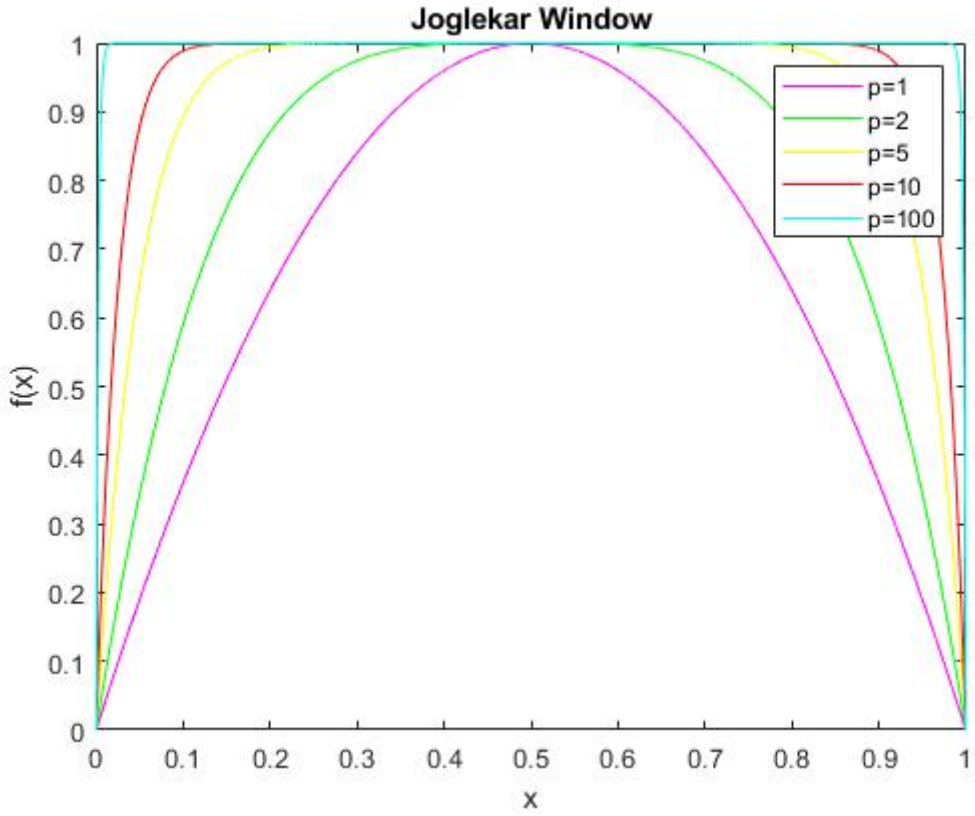

Figure 4.1 Joglekar and Wolf window function for different p values

4.1 Characteristic Analysis with Inputs

The aforementioned Joglekar and Wolf memristor models were used to implement the designed synapse model. As mentioned before, there are 3 fingerprints for memristor or memristive systems to be called memristors. The *i-v* graph creates a hysteresis curve as zero at the origin when a periodic voltage source is connected and the hysteresis lobe depends on the frequency which makes the memristor a constant value at the higher frequencies. For feed forward operations, the weights must remain constant. Taking advantage of the memristor's stability at high frequencies, sinusoidal signal with high frequency is applied to the memristor for feed forward processing. Taking advantage of the memristor's stability at high frequencies, a high frequency sinusoidal signal is applied to the memristor for feed forward processing. These applications are done in the SPICE environment with Joglekar and Wolf memristor model. Circuit is given in Figure 4.2 and the design of the SPICE model properties can be found in Appendix A.

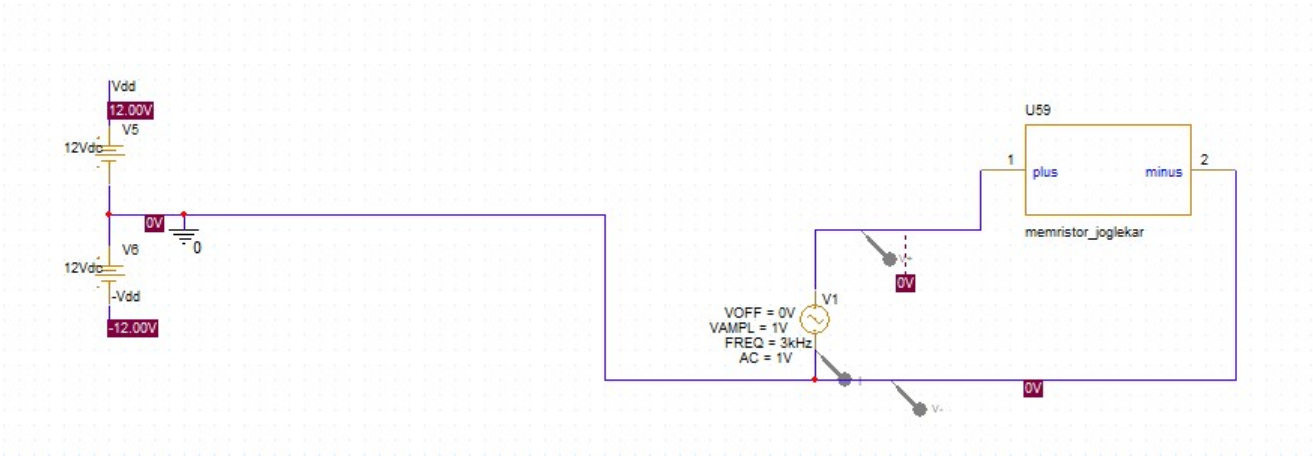

Figure 4.2 SPICE circuit design with Joglekar and Wolf memristor model

4.1.1 Memristor Simulations with Sinusoidal Input

In order to apply the input signals for feed forward operations and learning process, characteristic features can be observed of the memristor. Sinusoidal inputs at low and high frequencies are applied to the circuit created in the SPICE environment. Current and voltage graphs of an example input with high frequency applied in the SPICE environment can be seen in Figure 4.3 and Figure 4.4.

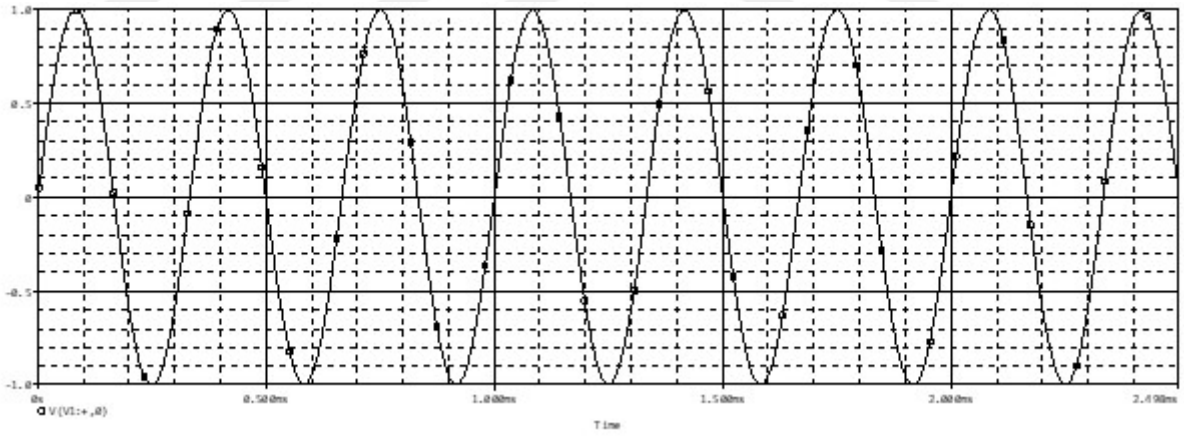

Figure 4.3 Voltage graph of the sinusoidal input with high frequency

For the feed forward operation a sinusoidal input is applied to the memristor. It has been observed that the memristor gives the hysteresis curve at low frequencies and behaves like a linear resistor at high frequencies. The behavior of the memristor at low frequencies is shown in Figure 4.5.

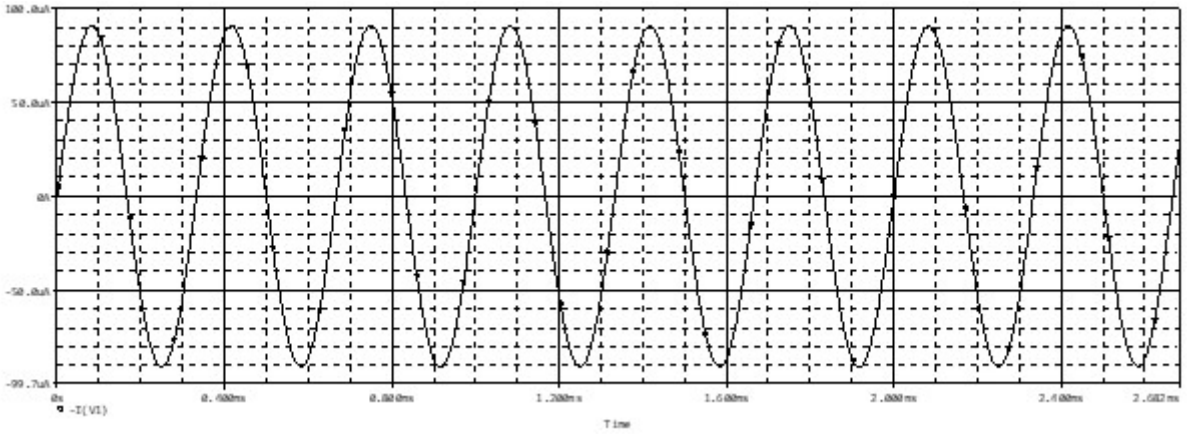

Figure 4.4 Current graph of the sinusoidal input with high frequency

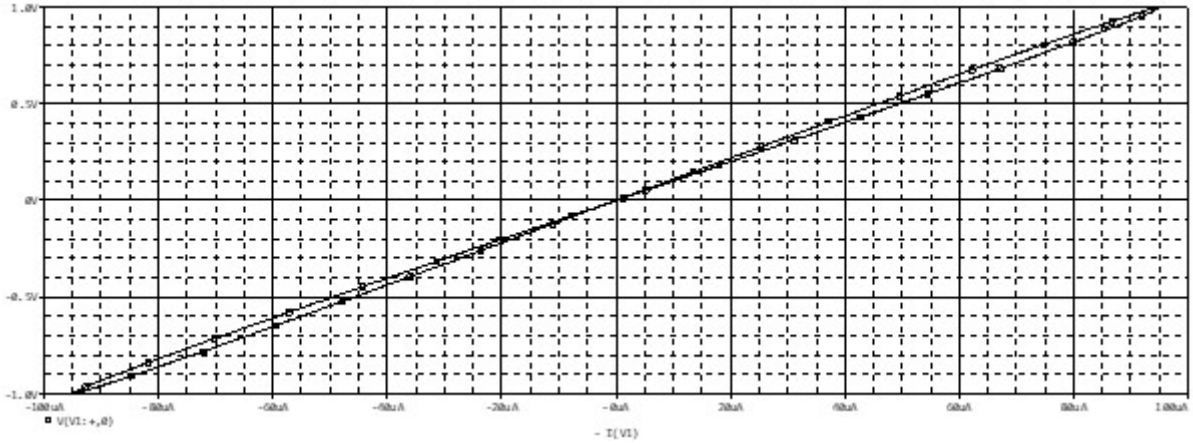

Figure 4.5 Hysteresis curve of Joglekar and Wolf memristor model at 10 Hz

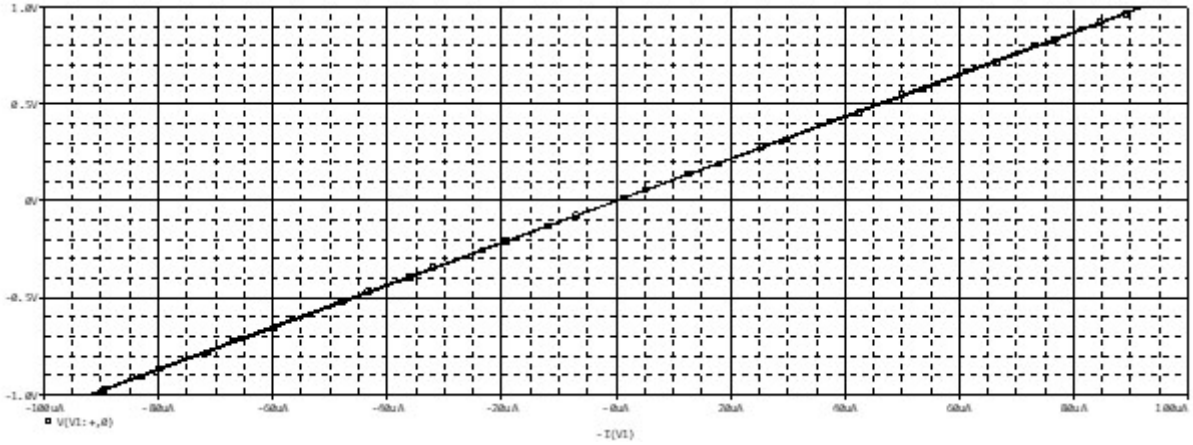

Figure 4.6 Hysteresis curve of Joglekar and Wolf memristor model at 20 Hz

When Figure 4.5, Figure 4.6 and Figure 4.7 are examined, it can be observed that the hysteresis curve becomes linear as the frequency of the signal applied to the memristor increases.

Observing that it behaves like a linear resistor at a frequency of 3 kHz as in Figure 4.7, this value was chosen as the feed forward operating frequency. The software normalization process compresses the input values to the range of [0, +1]. At this amplitude level, 3 kHz frequency sinusoidal signals form the input values.

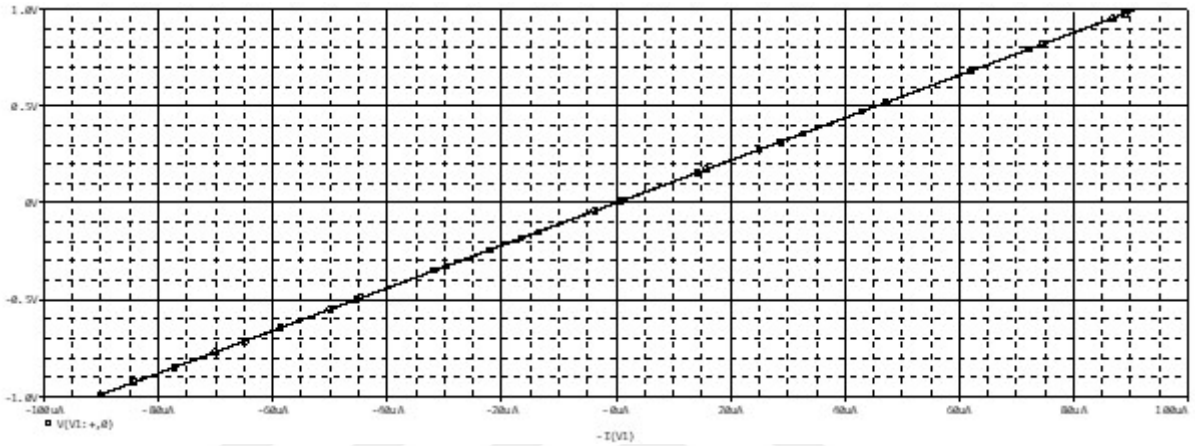

Figure 4.7 Hysteresis curve of Joglekar and Wolf memristor model at 3 kHz

4.1.2 Memristor Simulations with Pulse Train

Backpropagation algorithm is based on the principle of updating the weights by reflecting the calculated error value to the influencing weights. The algorithm generates the information by how much the current weight needs to change. That is, the current weight value must be changed to a certain value without the current memristance value information. The memristor element works as a nonlinear resistor at low frequencies and has a non-volatile memory feature. In this way, it allows to control the value with the control voltage applied at low frequencies.

In order to control the memristor used, a pulse train with 10 mV amplitude and 20 ms period were created in MATLAB as in Figure 4.8.

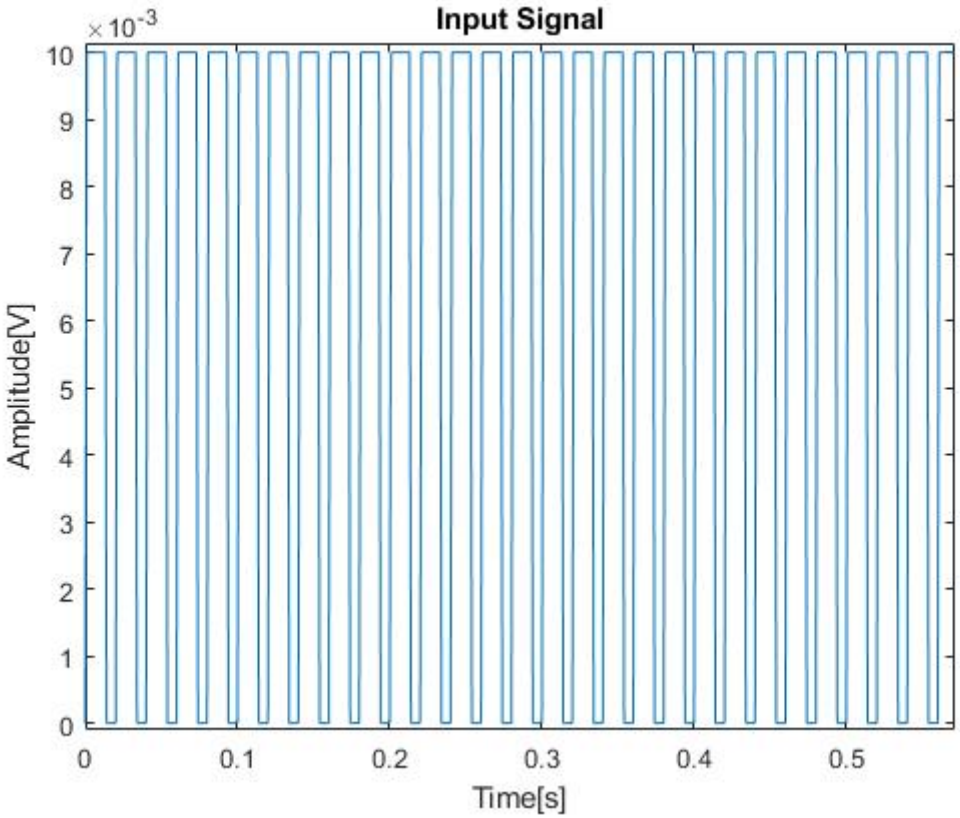

Figure 4.8 Zoomed-in view of the pulse train created in MATLAB

This signal applied and the change in memristance was examined. Current and voltage graphs of this input can be seen in Figure 4.9 and Figure 4.10.

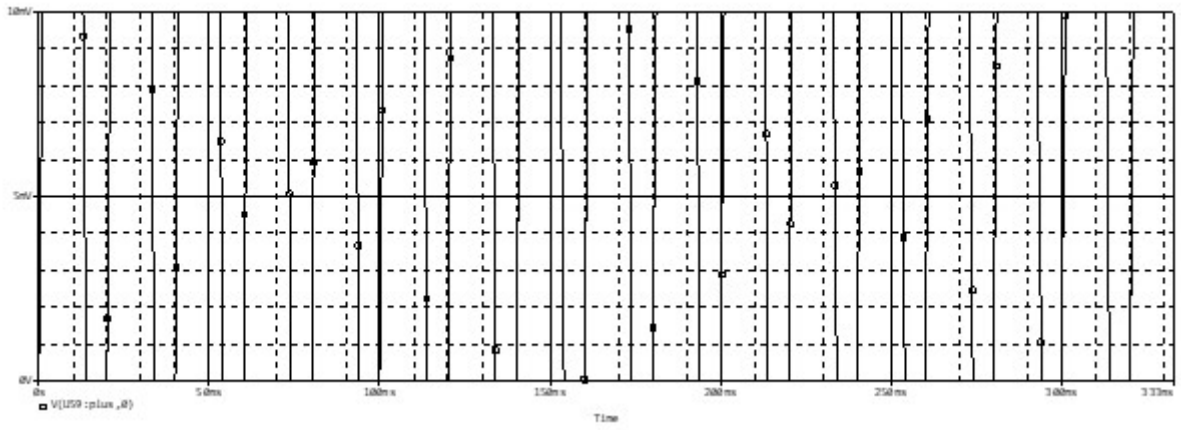

Figure 4.9 SPICE result of the voltage of the pulse train

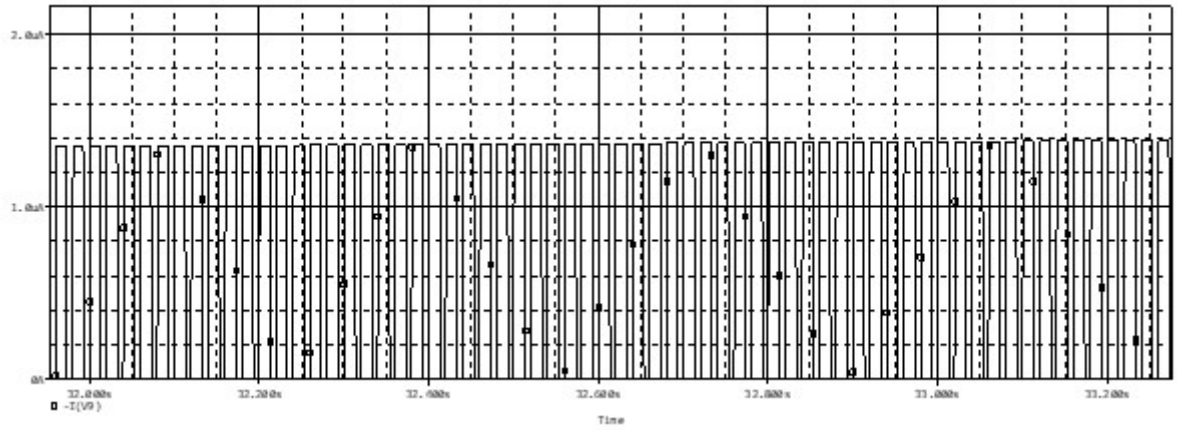

Figure 4.10 SPICE result of the current of the pulse train

According to the characteristic given in Figure 4.11, the memristor has an approximately linearly controllable resistance range in the range of 6-10.5 kΩ.

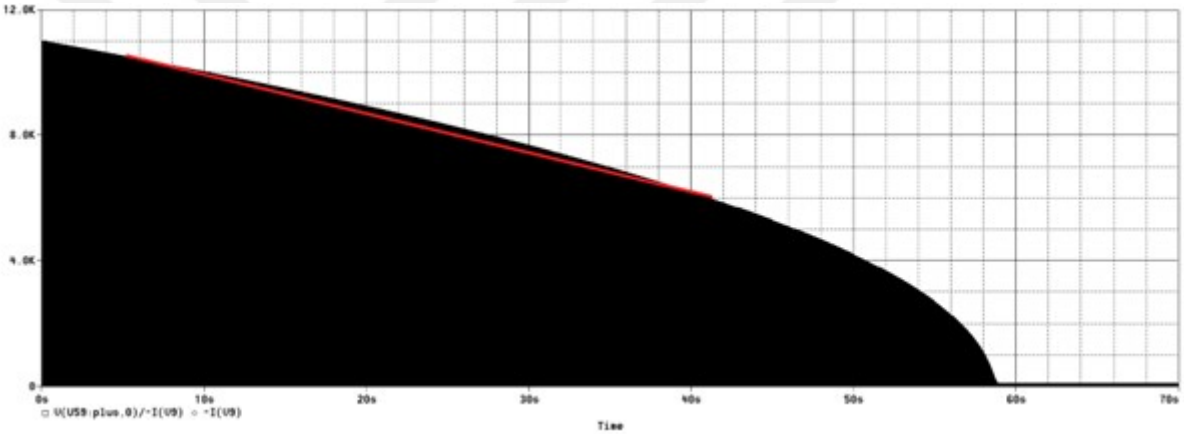

Figure 4.11 SPICE result showing change of memristance with pulse train

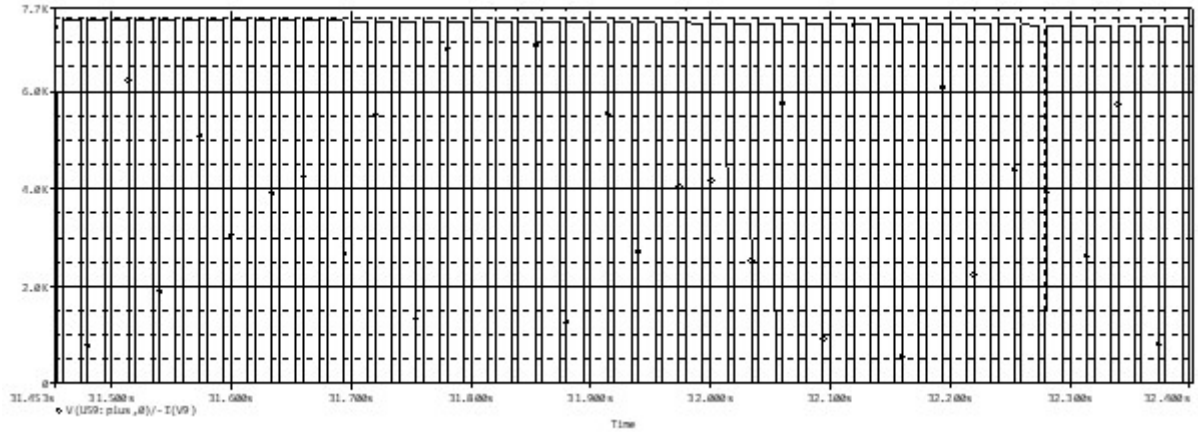

Figure 4.12 Zoomed-in memristance change

According to the graph given in Figure 4.11, approximately 2.5 Ω of memristance change is obtained with 1 pulse applied. By backpropagation, the required weight

changes are calculated. It can be seen that approximately 10 Ω $\Delta \mathrm{R}_{ij}$ will be obtained for each 0.014 ΔW by using the coefficient in Equation 3.5. Using the obtained ΔW value, the pulse duration required to be applied is calculated according to Equation 4.2.

$$t = 80\frac{\Delta \mathrm{R}_{ij}}{10}ms = 80\frac{\Delta W}{0.014}ms \tag{4.2}$$

5
MEMRISTOR EMULATOR CIRCUIT APPLICATION

After the discovery of the memristor, the absence of an ideal memristor element led scientists to design memristive circuits. These circuits are basically based on the idea of designing a circuit that displays the *i-v* characteristics of the memristor and exhibits memristive behavior with the existing circuit elements. It can be thought of as the physical hardware version of memristor models. The first known memristor emulator circuit was developed by Pershin and Di Ventra [35]. This circuit is built with a digital potentiometer, an analog-to-digital converter and a microcontroller to model the memristor. The development of emulator circuits for the physical application of the memristor to the circuits is an important development. This also allowed us to see how the memristor element behaves in physical circuits. After this development, a great increase has been observed in the studies on imitative circuits, as in the same memristor models. Different memristor emulator circuits obtained with different elements are available in the literature. A memristor emulator is also a memristor system and displays memristor-like properties. The memristive system design that Chua gave in his discovery of the memristor complies with the definitions of the memristor [1].

5.1 Characteristic Analysis with Inputs

A circuit that has been studied in the literature is used as the emulator circuit [36]. The schematic of the presented memristance emulator circuit with $\mathrm{R} = 15\mathrm{k}\Omega$, $\mathrm{C} = 100\mathrm{nF}$ and $\mathrm{I_b} = 200\mu\mathrm{A}$ values is given in Figure 5.1.

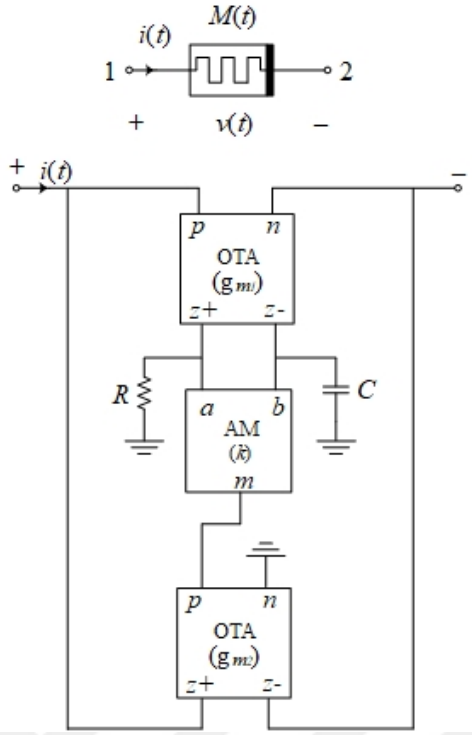

Figure 5.1 Proposed memristance simulator circuit [1]

CMOS transistors are used in this designed emulator circuit and the OTA circuit using 0.18μm level-7 TSMC CMOS technology parameters. Also, the hysteresis curves obtained with the input signals applied at different frequencies to the emulator circuit that can operate between 5Hz and 180kHz are given in Figure 5.2.

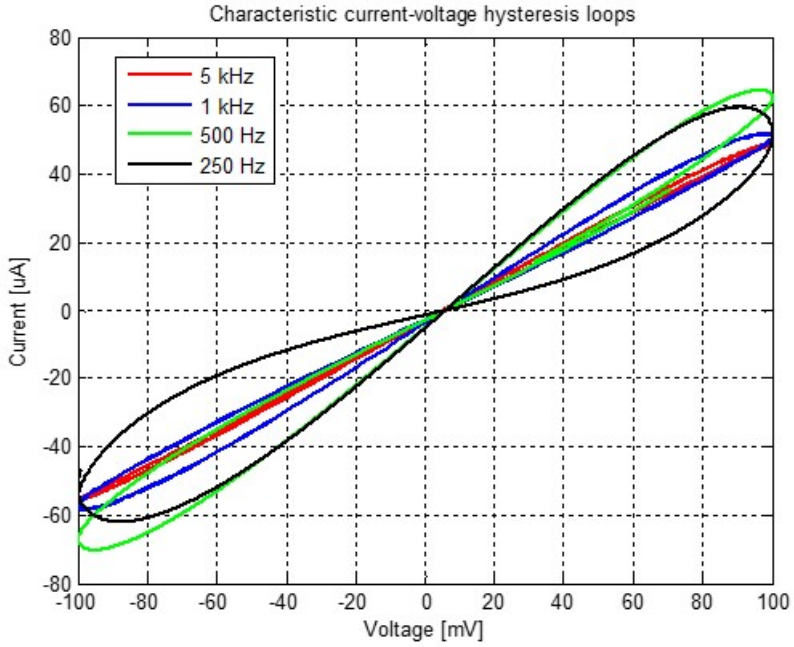

Figure 5.2 Hysteresis lobes for different frequencies [1]

Since memristor emulators act as memristors, high frequency sinusoidal input is applied for the feed forward in this application. With this input applied, only error calculation is performed without weight updates. Since the learning process takes place in the backpropagation stage, weight change is carried out by applying a pulse. By providing the characteristic features of the memristor, it is necessary to determine the appropriate inputs for synapse implementation.

5.1.1 Emulator Circuit Simulations with Sinusoidal Input

Simulations of the memristor emulator, which is expected to be linear at high frequencies for the feed forward operation, were carried out on PSPICE. The 3 kHz frequency is chosen for the feed forward. The voltage and current graphics of this sinusoidal signal applied to the emulator circuit are shown in Figure 5.3 and Figure 5.4.



Figure 5.3 Voltage graph of the sinusoidal input at 3kHz

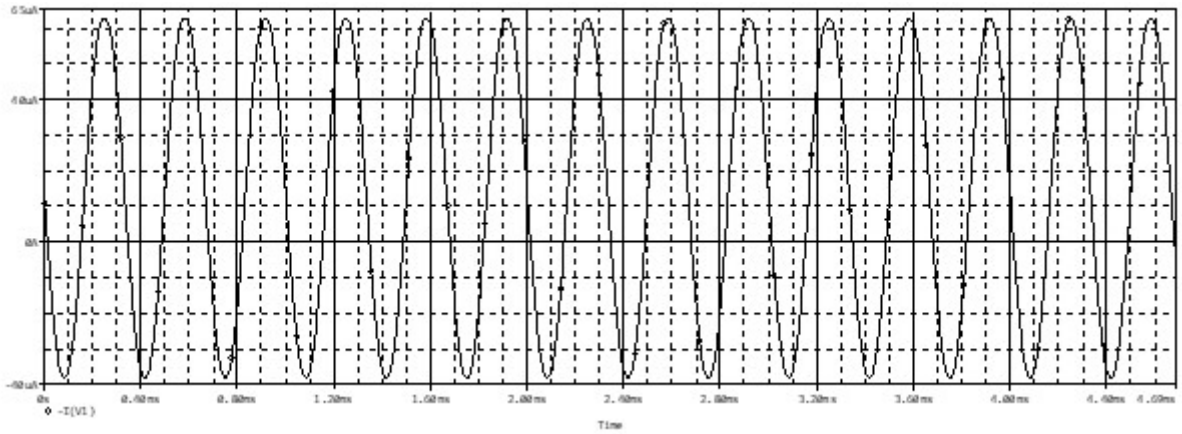

Figure 5.4 Current graph of the sinusoidal input at 3kHz

The hysteresis curve structure is observed by applying a sinusoidal voltage source to the emulator circuit used. While the other parameters are kept constant in the circuit, the results are obtained by changing only the frequency of the applied voltage source. A sinusoidal signal with 50 mV amplitude at different frequencies is applied for the input. The lobe area decreased as the frequency increased when the PSPICE graphs given in Figure 5.5, Figure 5.6 and Figure 5.7 are compared. The i-v characteristic turned into a linear line and became a constant resistance after a certain frequency value. In other words, it has been seen that this emulator circuit provides three fingerprints of the memristor.

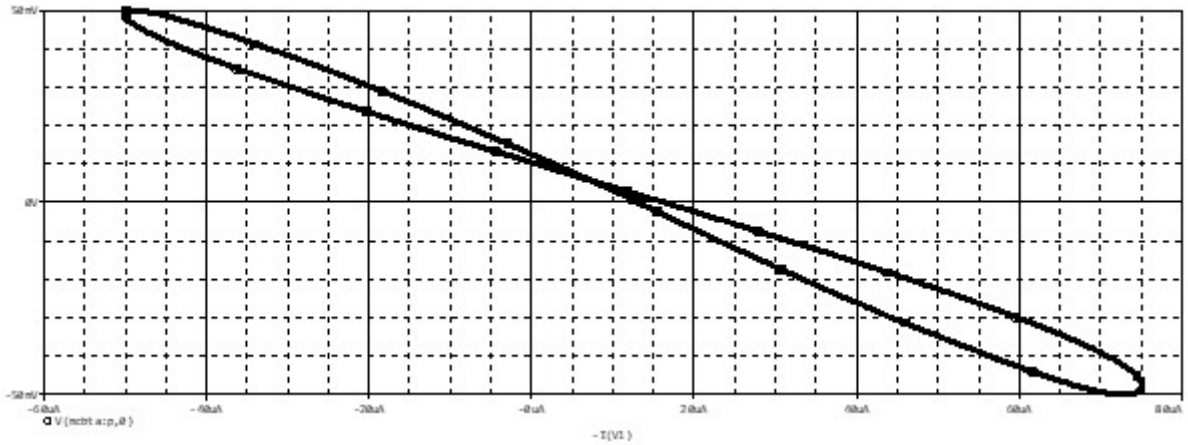

Figure 5.5 Hysteresis curve of memristor emulator circuit at 100 Hz

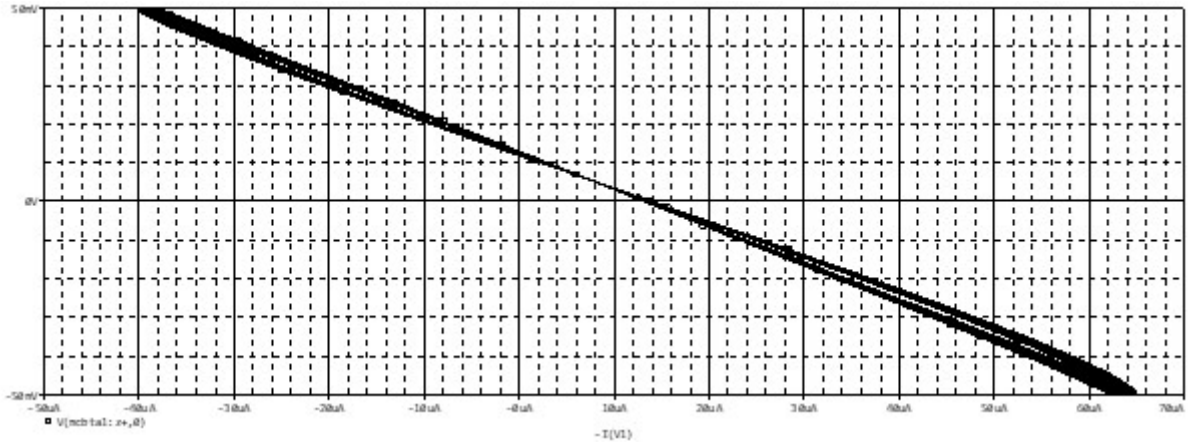

Figure 5.6 Hysteresis curve of memristor emulator circuit at 500 Hz

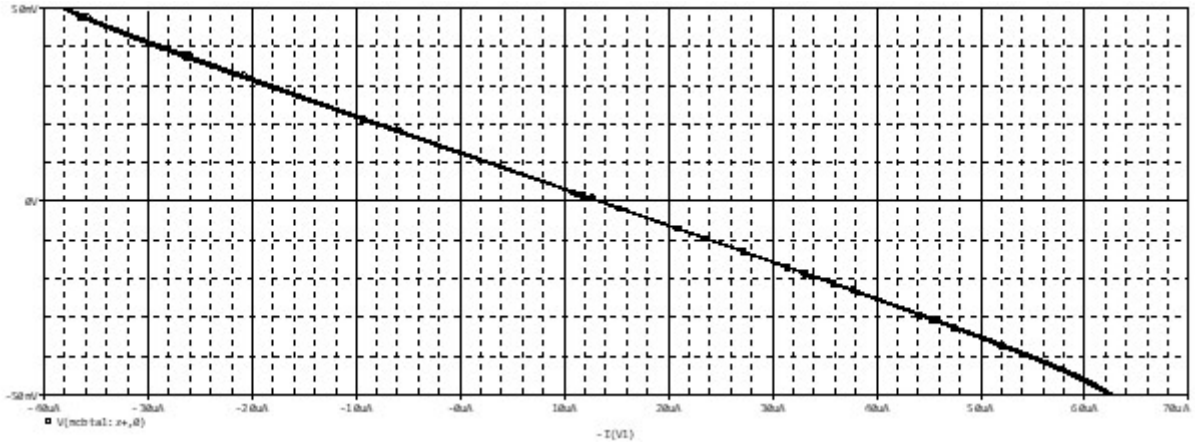

Figure 5.7 Hysteresis curve of memristor emulator circuit at 3 kHz

5.1.2 Emulator Circuit Simulations with Pulse Train

In order to perform weight updates during the learning process, pulses must be applied to the circuit at certain intervals. A pulse signal with 0.15 ms period and 50 mV amplitude was generated in MATLAB and applied. The non-volatility of the memory in the memristance circuit was tested by applying this signal. Voltage and current graphics of this input applied on SPICE are given in Figure 5.8 and Figure 5.9.

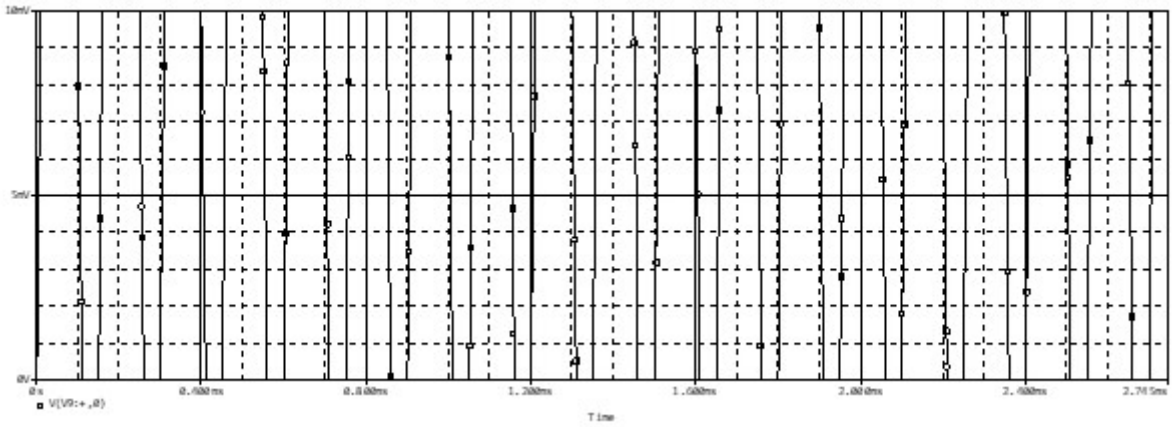

Figure 5.8 Voltage graph for applied pulse train

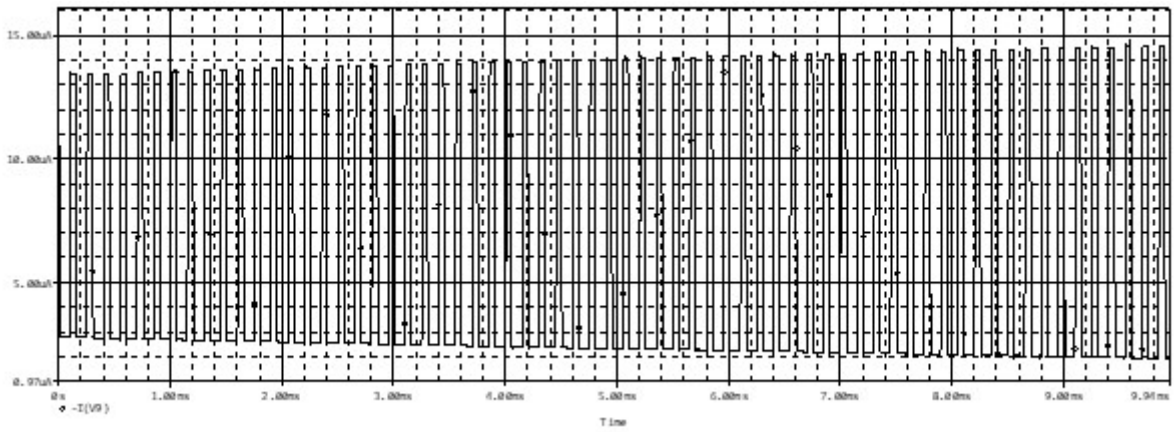

Figure 5.9 Current graph for applied pulse train

It has been observed that the memristance value changes with an applied square wave and the memristance value remains constant as long as no voltage is applied. The simulation results are given in Figure 5.10 and 5.11. Memristance change and non-volatility feature are confirmed with these graphics.

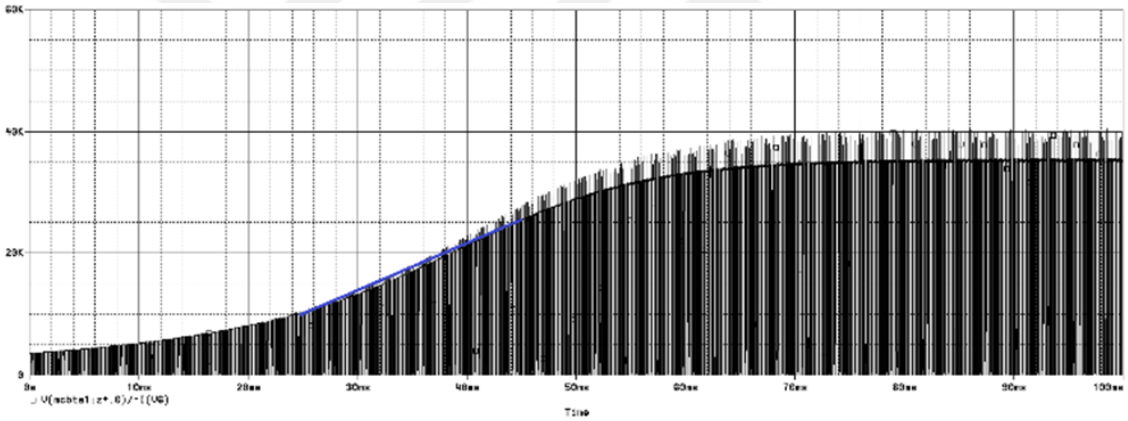

Figure 5.10 Memristance change with square wave

According to the characteristic given in Figure 5.10, the memristor has an approximately linearly controllable resistance range in the range of 10-25 kΩ.

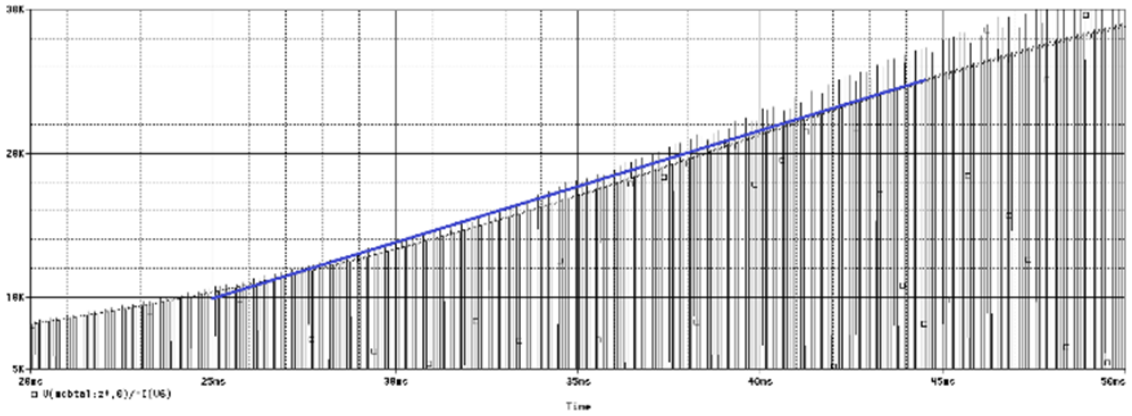

Figure 5.11 Zoomed-in memristance change with square wave

According to the graph given in Figure 5.11, approximately 50 Ω of memristance change is obtained with 1 pulse applied. By backpropagation, the required weight changes are calculated. It can be seen that approximately 100 Ω $\Delta \mathrm{R}_{ij}$ will be obtained for each 0.024 ΔW by using the coefficient in Equation 3.5. Using the obtained ΔW value, the pulse duration required to be applied is calculated according to Equation 5.1.

$$t = 0.3\frac{\Delta \mathrm{R}_{ij}}{100} ms = 0.3\frac{\Delta W}{0.024} ms \tag{5.1}$$

6
RESULTS AND DISCUSSION

The memristor started to take place in the literature with the proof of the existence of the missing circuit element. First of all, emulator circuits that give memristive results have been designed because they do not exist physically. With the increase in these circuits, the application areas of the memristor began to be explored. The memristor, which is used in many analog and digital applications, started to be used in neuromorphic applications with the realization of its similarity to synapses.

In this study, a memristor based classifier circuit is proposed. With this circuit, a multilayer network designed in software is implemented in hardware and its compatibility with the algorithm is checked. In the circuit design, the weighted summation process is carried out using a crossbar array. In addition, since the backpropagation learning algorithm is used to train the network, the circuit includes feed forward and backpropagation blocks. These blocks are designed as analog circuits. During the learning process, memristors were used for weight update processes.

For this purpose, simulations were carried out with the Joglekar and Wolf memristor model, which was obtained by adding the window function to the HP memristor model. In order to perform feed forward operations, the linearity of the memristor at high frequencies is utilized by applying a 3kHz sinusoidal signal. For the learning process, weight updates were made with the 20 ms period pulse train. With the designed network, weight changes with 0.014 precision can be realized. Necessary weights can be realized by adjusting the memristor values in the 6-10.5 kΩ range, which is considered approximately linear.

In another application, the learning process was carried out with a memristor emulator circuit in the literature. In the same way as the previous application, a sinusoidal signal with a frequency of 3kHz was applied for the feed forward operation and weight changes were performed with a 0.15 ms period pulse train. The designed network is suitable for weight change adaptation with a precision

of 0.024. Necessary weights can be realized by adjusting the memristor values in the range of approximately 10-25 kΩ. The inputs applied to the designed network were obtained with MATLAB and the operability of the network was verified with PSPICE simulation using the iris flower dataset.

In the studies carried out so far, a design has been proposed depending on the network. In further studies, it is aimed to achieve a more sensitive weight change by controlling the memristor with DC instead of pulse. In addition, no negative results were observed at the outputs of the designed network. Therefore, the outputs were directly rectified and passed through the activation function. However, if the training is performed with a different data set, it is necessary to perform phase control at the network output, since the negative output will appear positive during the rectification process. The studies were carried out with mathematical and simulation programs, but with the developing technology, the physical production of memristors was also required. For this reason, the realization of the proposed design with a commercial memristor has become one of the goals of future studies.

REFERENCES

[1] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[2] G. Dearnaley, A. Stoneham, D. Morgan, "Electrical phenomena in amorphous oxide films," *Reports on Progress in Physics*, vol. 33, no. 3, p. 1129, 1970.

[3] R. Waser, M. Aono, "Nanoionics-based resistive switching memories," *Nature materials*, vol. 6, no. 11, pp. 833–840, 2007.

[4] R. Oligschlaeger, R. Waser, R. Meyer, S. Karthäuser, R. Dittmann, "Resistive switching and data reliability of epitaxial (ba, sr) ti o 3 thin films," *Applied Physics Letters*, vol. 88, no. 4, p. 042 901, 2006.

[5] D. B. Strukov, G. S. Snider, D. R. Stewart, R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[6] L. Chua, "Everything you wish to know about memristors but are afraid to ask," *Handbook of Memristor Networks*, pp. 89–157, 2019.

[7] M. R. Azghadi, B. Linares-Barranco, D. Abbott, P. H. Leong, "A hybrid cmos-memristor neuromorphic synapse," *IEEE transactions on biomedical circuits and systems*, vol. 11, no. 2, pp. 434–445, 2016.

[8] H. Li, Y. Chen, "An overview of non-volatile memory technology and the implication for tools and architectures," in *2009 Design, Automation & Test in Europe Conference & Exhibition*, IEEE, 2009, pp. 731–736.

[9] C. Xu, X. Dong, N. P. Jouppi, Y. Xie, "Design implications of memristor-based rram cross-point structures," in *2011 Design, Automation & Test in Europe*, IEEE, 2011, pp. 1–6.

[10] S. P. Adhikari, C. Yang, H. Kim, L. O. Chua, "Memristor bridge synapse-based neural network and its learning," *IEEE Transactions on neural networks and learning systems*, vol. 23, no. 9, pp. 1426–1435, 2012.

[11] Q. Guo, N. Wang, G. Zhang, "A novel current-controlled memristor-based chaotic circuit," *Integration*, vol. 80, pp. 20–28, 2021.

[12] A. L. Hodgkin, A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

[13] L. Chua, "Memristor, hodgkin–huxley, and edge of chaos," *Nanotechnology*, vol. 24, no. 38, p. 383 001, 2013.

[14] S. Satyanarayana, Y. P. Tsividis, H. P. Graf, "A reconfigurable vlsi neural network," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 1, pp. 67–81, 1992.

[15] T. Morie, Y. Amemiya, "An all-analog expandable neural network lsi with on-chip backpropagation learning," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 9, pp. 1086–1093, 1994.

[16] B. Zamanlooy, M. Mirhassani, "Efficient vlsi implementation of neural networks with hyperbolic tangent activation function," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 1, pp. 39–48, 2013.

[17] C. Mead, "Analog vlsi and neutral systems," *NASA STI/Recon Technical Report A*, vol. 90, p. 16 574, 1989.

[18] H. Hikawa, "Implementation of simplified multilayer neural networks with on-chip learning," in *Proceedings of ICNN'95-International Conference on Neural Networks*, IEEE, vol. 4, 1995, pp. 1633–1637.

[19] D. Abramson, K. Smith, P. Logothetis, D. Duke, "Fpga based implementation of a hopfield neural network for solving constraint satisfaction problems," in *Proceedings. 24th EUROMICRO Conference (Cat. No. 98EX204)*, IEEE, vol. 2, 1998, pp. 688–693.

[20] S. Duan, X. Hu, Z. Dong, L. Wang, P. Mazumder, "Memristor-based cellular nonlinear/neural network: Design, analysis, and applications," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 6, pp. 1202–1213, 2014.

[21] C. Xu, C. Wang, Y. Sun, Q. Hong, Q. Deng, H. Chen, "Memristor-based neural network circuit with weighted sum simultaneous perturbation training and its applications," *Neurocomputing*, vol. 462, pp. 581–590, 2021.

[22] G. S. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *2008 IEEE international symposium on nanoscale architectures*, Ieee, 2008, pp. 85–92.

[23] J. A. Pérez-Carrasco, C. Zamarreno-Ramos, T. Serrano-Gotarredona, B. Linares-Barranco, "On neuromorphic spiking architectures for asynchronous stdp memristive systems," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, IEEE, 2010, pp. 1659–1662.

[24] G. S. Rose, R. Pino, Q. Wu, "A low-power memristive neuromorphic circuit utilizing a global/local training mechanism," in *The 2011 international joint conference on neural networks*, IEEE, 2011, pp. 2080–2086.

[25] D. Querlioz, O. Bichler, P. Dollfus, C. Gamrat, "Immunity to device variations in a spiking neural network with memristive nanodevices," *IEEE transactions on nanotechnology*, vol. 12, no. 3, pp. 288–295, 2013.

[26] E. Rosenthal, S. Greshnikov, D. Soudry, S. Kvatinsky, "A fully analog memristor-based neural network with online gradient training," in *2016 IEEE international symposium on circuits and systems (ISCAS)*, IEEE, 2016, pp. 1394–1397.

[27] R. Hasan, T. M. Taha, C. Yakopcic, "On-chip training of memristor crossbar based multi-layer neural networks," *Microelectronics journal*, vol. 66, pp. 31–40, 2017.

[28] Y. Zhang, X. Wang, E. G. Friedman, "Memristor-based circuit design for multilayer neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 2, pp. 677–686, 2017.

[29] H. Ran *et al.*, "Memristor-based edge computing of blaze block for image recognition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 2121–2131, 2020.

[30] J. Sun, X. Xiao, Q. Yang, P. Liu, Y. Wang, "Memristor-based hopfield network circuit for recognition and sequencing application," *AEU-International Journal of Electronics and Communications*, vol. 134, p. 153 698, 2021.

[31] X. Liu, Z. Zeng, D. C. Wunsch II, "Memristor-based lstm network with in situ training and its applications," *Neural Networks*, vol. 131, pp. 300–311, 2020.

[32] S. P. Adhikari, M. P. Sah, H. Kim, L. O. Chua, "Three fingerprints of memristor," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 11, pp. 3008–3021, 2013.

[33] A. Adamatzky, G. Chen, *Chaos, Cnn, Memristors and Beyond: A Festschrift For Leon Chua (With Dvd-rom, Composed By Eleonora Bilotta)*. World Scientific, 2013.

[34] Y. N. Joglekar, S. J. Wolf, "The elusive memristor: Properties of basic electrical circuits," *European Journal of physics*, vol. 30, no. 4, p. 661, 2009.

[35] Y. V. Pershin, M. Di Ventra, "Memristive circuits simulate memcapacitors and meminductors," *arXiv preprint arXiv:0910.1583*, 2009.

[36] Z. G. Cam Taskiran, U. E. Ayten, H. Sedef, "Dual-output operational transconductance amplifier-based electronically controllable memristance simulator circuit," *Circuits, Systems, and Signal Processing*, vol. 38, pp. 26–40, 2019.

A
JOGLEKAR AND WOLF PSPICE MODEL

```
.SUBCKT memristor_joglekar plus minus PARAMS:
+ Ron=100 Roff=16K Rinit=5K D=10N uv=10F p=10
**********************************************
* DIFFERENTIAL EQUATION MODELING *
**********************************************
Gx 0 x value= I(Emem)*uv*Ron/D**2*f(V(x),p)
Cx x 0 1 IC=(Roff-Rinit)/(Roff-Ron)
Raux x 0 1T
* RESISTIVE PORT OF THE MEMRISTOR *
*******************************
Emem plus aux value=-I(Emem)*V(x)*(Roff-Ron)
Roff aux minus Roff
**********************************************
*Flux computation*
**********************************************
Eflux flux 0 value=SDT(V(plus,minus))
**********************************************
*Charge computation*
**********************************************
Echarge charge 0 value=SDT(I(Emem))
```

```
**********************************************
* WINDOW FUNCTIONS
* FOR NONLINEAR DRIFT MODELING *
**********************************************
*window function, according to Joglekar
.func f(x,p)=1-(2*x-1)**(2*p)
.ENDS memristor_joglekar
```

PUBLICATIONS FROM THE THESIS

Conference Papers

1. S. Günakın And Z. G. Çam Taşkıran, "Memristör Tabanlı Çok Katmanlı Algılayıcı ve Ağırlık Uyarlaması," 2nd International Conference on Innovative Academic Studies , vol.2, pp.474-480, 2023.