

A Simple Robust and Versatile MATLAB Formulation of the Dynamic Mem-diode Model for Bipolar-Type Resistive Random-Access Memory Devices

VLSI

E Meghana¹,ECE ,Institute of Aeronautical Engineering, Hyderabad, India

22951A0496@iare.ac.in

GUDIE: DR.S China Venkateshwarlu, Professor,ECE&IARE

c.venkateshwarlu@iare.ac.in

DR.V Siva Nagaraju, Professor,ECE,IARE

v.sivanagaraju@iare.ac.in

MS. Ms P.Ganga Bhavani,Assistant Professor,ECE,IARE

p.gangabhavani@iare.ac.in

ABSTRACT:

Modeling in an emerging technology like RRAM devices is one of the pivotal concerns for its development. In the current bibliography, most of the models face difficulties in implementing or simulating unconventional scenarios, particularly when dealing with complex input signals. In addition, circuit simulators like Spice require long running times for high-resolution results because of their internal mathematical implementation. In this work, a fast, simple, robust, and versatile model for RRAM devices built in MATLAB is presented. The proposed model is a recursive and discretized version of the dynamic memdiode model (DMM) for bipolar-type resistive switching devices originally implemented in LTspice. The DMM model basically consists of two coupled equations: one for the current (non-linear current generator) and a second one for the memory state of the device (time-dependent differential equation). This work presents an easy-to-use tool for researchers to reproduce the experimental behavior of their devices and predict the outcome from nontrivial experiments. Three study cases are reported, aimed at capturing different phenomenologies: a frequency effect study, a cycle-to-cycle variability fit, and a stochastic resonance impact analysis.

INTRODUCTION:

In recent years, the scientific community in the microelectronics field has expressed huge interest in RRAM (Resistive Random Access Memory) devices due to their unique properties, such as low power consumption, high scalability, endurance, and CMOS compatibility ,among others As the switching material, these devices can be based on metal oxides like TiO₂ or HfO₂, chalcogenide glasses, lower dimensional materials like nanowires or nano-dots, as well as other novel materials like metal–oxide frameworks [2–4]. Their application range includes logic circuits, neuro-morphic systems, cryptology, and information storage. Since RRAM devices are still an emerging technology, device level characterization, and powerful modeling tools are key for enhancing its evolution to massive commercialization. To date, a lot of models have been implemented in simulators like Spice or Verilog. These simulators are useful for complex architectures involving several devices but also lead to long waiting times for running the simulations and post-processing the data. In this work, a simple and effective tool is presented for reproducing the behavior of single RRAM devices in MATLAB, a common and well-known programming language. The advantages of the tool presented consist of easy implementation(since the script is openly included in the manuscript), straightforward data analysis,short

simulating running times, and versatility in terms of parameters definition and input using MATLAB for RRAM modeling are discussed

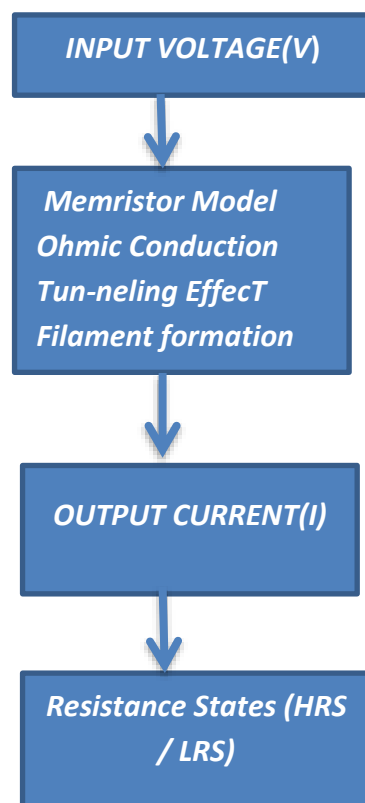
1.1:LITERATURE SURVEY

YEAR	AUOTHER	ALGORITHM/TECHNIQUE	SUMMARY	PROBLEM	REMARKS
2024	Emili Salvador, Rosana Rodriguez, Enrique Miranda	MATLAB-based Dynamic Memdiode Model	Developed a recursive and discretized version of the Dynamic Memdiode Model (DMM) in MATLAB for simulating bipolar-type resistive switching devices. The model captures nonlinear and hysteretic behaviors of memdiodes.	Existing models faced challenges in simulating complex input signals and required long simulation times.	The MATLAB implementation offers a user-friendly and efficient tool for researchers to replicate experimental behaviors and predict outcomes in non-trivial scenarios. Link
2023	Enrique Miranda, Jordi Suñé	SPICE Implementation of Dynamic Memdiode Model	Presented a SPICE-compatible implementation of the DMM for bipolar resistive switching devices, focusing on a recursive approach that aligns with the model's original foundations	Need for a simplified and SPICE-compatible model to accurately simulate conduction characteristics of resistive switching devices.	The SPICE implementation facilitates integration into existing circuit simulation workflows, enhancing accessibility for circuit designers. Link
2023	Yunki Kim, Kyu-Jin Jo, Jin-Su Oh, Cheol-Woong Yang	Ta/TaxMnyOz/Pt Structure for ReRAM	Investigated a Ta/TaxMnyOz/Pt metal-insulator-metal structure for RRAM applications, demonstrating stable bipolar resistive	Challenges in achieving stable and reproducible resistive switching characteristics in RRAM devices.	The study provides insights into material engineering for improved RRAM performance. Link
2022	Fernando Leonel	SPICE Implementation of	Detailed the fundamentals	Difficulty in	The approach

	Aguirre, Jordi Suñé, Enrique Miranda	Dynamic Memdiode Model	and SPICE implementation of the DMM for bipolar resistive switching devices, using an equivalent circuit approach with behavioral components.	implementing accurate models for bipolar resistive switching devices in SPICE simulators.	enhances the capability to simulate and design circuits incorporating resistive switching devices. Link
2022	Enrique Miranda, Jordi Suñé	Fundamentals and SPICE Implementation of DMM	Provided a comprehensive overview of the fundamentals and SPICE implementation of the DMM for bipolar resistive switching devices	Need for a robust model to simulate the conduction characteristics of bipolar resistive switching devices.	The paper offers a foundational understanding and practical implementation guide for the DMM in SPICE. Link

1.2 BLOCK DIAGRAMS:

EXISTING ReRAM Model Block Diagram:



1.2.1 DEFINE PROBLEM:

The development of high-performance Resistive Random Access Memory (RRAM) devices requires accurate and efficient modeling techniques. Traditional models often lack robustness, versatility, and computational efficiency when simulating the dynamic behavior of memristive devices, especially for bipolar-type switching mechanisms. This work aims to formulate a simple, robust, and versatile MATLAB-based dynamic memdiode model for bipolar-type RRAM devices. The model should accurately capture the resistance switching dynamics, state variable evolution, and voltage-current characteristics while ensuring computational efficiency. The formulated MATLAB model should be capable of handling various input signals, state-dependent resistance changes, and nonlinear switching dynamics, making it suitable for circuit simulations and memory device analysis.

1.3 PROBLEM IDENTIFICATION:

The primary challenge in modeling bipolar-type Resistive Random Access Memory (RRAM) devices lies in accurately capturing their nonlinear and dynamic switching behavior. Existing models often struggle to represent the physical mechanisms governing resistive switching and hysteresis effects, leading to inaccuracies in simulations. Additionally, most available models are either device-specific or computationally inefficient, making them unsuitable for large-scale or versatile applications. The absence of a standardized and adaptable simulation framework further complicates the development of reliable RRAM models. Given these challenges, a MATLAB-based formulation of the dynamic memdiode model offers a promising solution by providing a simple, robust, and computationally efficient approach to simulate bipolar switching characteristics effectively.

1.3.1 DESCRIPTION:

Resistive Random Access Memory (Re-RAM) has emerged as a promising non-volatile memory (NVM) technology due to its high-speed operation, low power consumption, and scalability. Unlike conventional memory technologies, Re-RAM leverages resistive switching to store data, making it an attractive option for next-generation computing systems. One of the key elements of Re-RAM devices is the memdiode, which exhibits dynamic resistance changes under applied voltage, forming the foundation of Re-RAM functionality.

Bipolar-type Re-RAM devices operate by alternating the polarity of the applied voltage to induce switching between high and low resistance states. This behavior is governed by complex nonlinear dynamics, which must be accurately captured to facilitate device modeling and simulation. Existing Re-RAM models, however, are often computationally intensive, lack adaptability, or are difficult to integrate into circuit simulation environments like MATLAB.

This work presents a simple, robust, and versatile MATLAB-based formulation for the dynamic memdiode model tailored to bipolar-type Re-RAM devices. The model captures key electrical characteristics such as resistive switching, hysteresis, and state-dependent resistance variations. It aims to provide an efficient and user-friendly tool for researchers and engineers working on Re-RAM-based memory applications.

Background and Challenges in Re-RAM Modelling

Re-RAM operates on the principle of resistive switching, where the application of an external voltage leads to the formation and dissolution of conductive filaments within a dielectric material. This process is characterized by nonlinearity, memory effects, and dynamic state evolution, making it essential to develop accurate mathematical models to describe Re-RAM behavior.

Several modelling approaches have been proposed for Re-RAM devices, including physical, empirical, and compact models. While physical models offer deep insight into the underlying mechanisms, they often require complex differential equations and computationally expensive simulations. Empirical models, on the other hand, provide simplified representations but may lack generalizability across different Re-RAM architectures. Compact models aim to balance accuracy and computational efficiency, making them suitable for circuit-level simulations.

CHAPTER-2: Proposed Dynamic Memdiode Model Block Diagram:

2.1 Technical specifications and methodology:

The proposed block diagram of the MATLAB-based dynamic mem-diode model for bipolar-type RRAM consists of several key components that work together to simulate the device's electrical characteristics.

1. Input Voltage Source

Provides an external voltage stimulus to the RRAM model.

- Used to analyze the response of the mem-diode under different operating conditions (e.g., DC sweep, pulse response).

2. Memristor State Equation Block

- Defines the mathematical formulation governing the internal state variable of the mem-diode.
- Captures the dynamic switching behavior based on the applied voltage and current.
- Incorporates nonlinear characteristics to accurately represent resistive switching mechanisms.

3. Current-Voltage (I-V) Characteristics Module

- Computes the memristor's current response based on the state equation and applied voltage.
- Implements a mathematical function that models the hysteresis behavior and nonlinearity of the device.

4. Resistance Modulation Unit

- Adjusts the device resistance dynamically based on the applied voltage polarity and magnitude.
- Captures the transition between high-resistance state (HRS) and low-resistance state (LRS).

5. Output Response Block

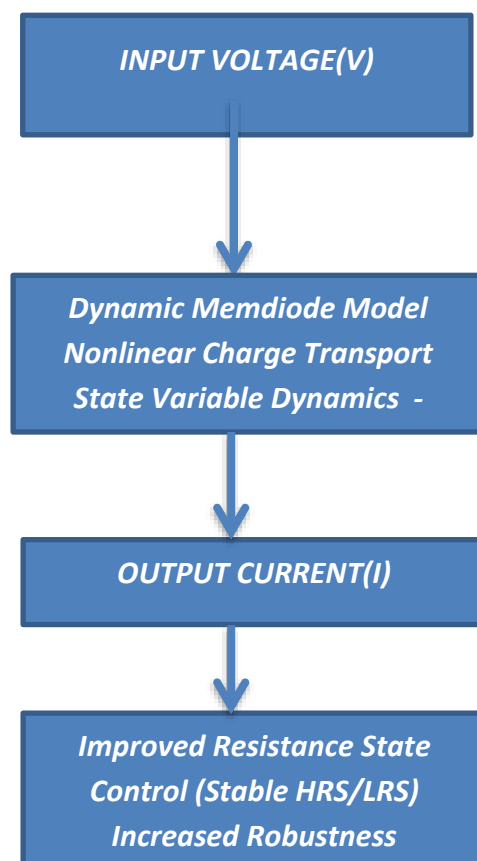
- Provides the final simulated results, including I-V characteristics, switching dynamics, and hysteresis behavior.
- Displays graphical plots for better visualization of the mem-diode performance.

6. MATLAB Simulation Environment

- Acts as the computational framework where all the mathematical models are implemented.
- Uses MATLAB scripts and Simulink (if applicable) to execute simulations efficiently.

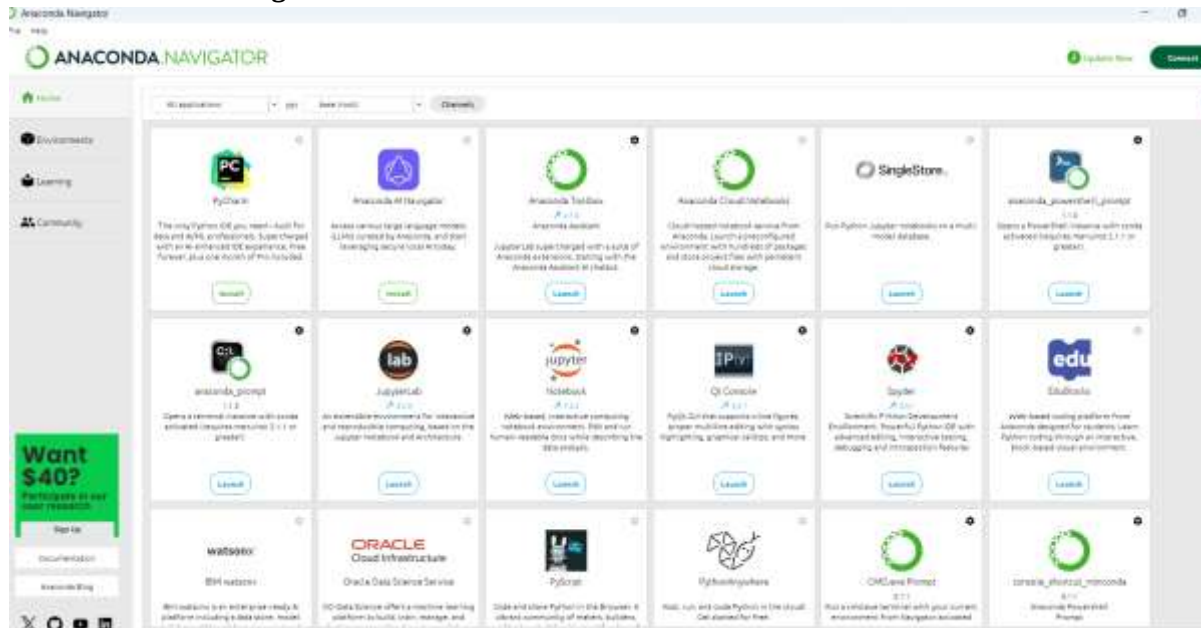
This structured approach ensures that the proposed model is simple, robust, and versatile, enabling accurate and efficient simulation of bipolar-type RRAM devices.

PROPOSED BLOCK DIAGRAM:



2.3 SOFTWARE USED:

2.3.1. Anaconda navigator



Anaconda Navigator is a desktop graphical user interface (GUI) that comes with the Anaconda distribution, which is widely used for data science, machine learning, and scientific computing. It allows users to manage packages, environments, and launch applications without needing to use command-line tools. With Anaconda Navigator, users can easily start tools like Jupyter Notebook, Spyder, VS Code, and others. It simplifies the workflow by providing a user-friendly interface to install, update, and remove packages, and manage multiple Python environments efficiently.

2.3.2 Jupyter Notebook

Jupyter Notebook is an open-source interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It is especially popular in the fields of data science, machine learning, and scientific research due to its ability to combine code execution with rich text annotations in a single document.

2.3.3 python code

Python is a powerful and flexible programming language that plays a crucial role in simulating the Dynamic Memdiode Model (DMM), which describes the behavior of resistive switching memory devices like memristors. In this context, Python is used to implement the mathematical model that governs the current-voltage relationship and memory state dynamics of the device. With scientific libraries such as numpy, scipy, and matplotlib, Python can efficiently handle complex numerical computations, solve nonlinear equations (such as those involving the Lambert W function), and visualize the results through plots. The use of Jupyter notebooks further enhances the workflow by allowing researchers to combine code, equations, and graphical outputs in an interactive environment. Python also supports the simulation of multiple switching cycles, the introduction of variability or noise, and the generation of statistical distributions such as histograms of set and reset voltages. Overall, Python provides an accessible, open-source, and highly effective platform for exploring and analyzing the dynamics of memristive systems modeled by the DMM.

LIBRARIES: 2.3.4 NumPy

NumPy, short for Numerical Python, is a foundational library in the scientific Python ecosystem. It provides highly efficient operations on arrays and matrices, making it ideal for modeling physical systems like memristive devices. In the context of the DMM, NumPy is used to define voltage waveforms, manage memory states, and perform vectorized

mathematical operations that are critical for simulation efficiency. Its ability to handle large numerical datasets with speed and precision makes it essential for running multiple cycles of device behavior and storing voltage-current characteristics.

2.3.5 Matplotlib

Matplotlib is a widely used plotting library in Python that allows for the creation of static, animated, and interactive visualizations. For DMM simulations, it is instrumental in visualizing the nonlinear current-voltage (I-V) characteristics, memory state evolution, and statistical distributions such as histograms of switching voltages. Using Matplotlib, researchers can generate clear, publication-quality plots that reveal the dynamic and stochastic behavior of memristive devices over multiple cycles. Its intuitive syntax and customization options make it a go-to tool for scientific data visualization.

2.3.6 SciPy (scipy.special)

The `scipy.special` module in the SciPy library provides advanced mathematical functions that go beyond what NumPy offers. A key feature used in the DMM model is the **Lambert W function**, which is essential for calculating the current response of the memristor. This special function solves equations of the form $x \cdot e^x = y$, which appears in the exponential current-voltage relation in the memdiode model. Without SciPy, implementing this aspect of the model would be far more complex or require external libraries. SciPy ensures the mathematical rigor needed for accurate simulation of nonlinear systems.

2.3.7 Random / NumPy.Random

Randomness and variability are inherent in real-world electronic devices, and Python offers tools to simulate these effects. The `random` module and the more powerful `numpy.random` module are used to introduce variability into model parameters such as switching thresholds, resistance states, and noise. This is especially important in stochastic resonance experiments or cycle-to-cycle (C2C) variability simulations, where distributions (normal, log-normal, etc.) are used to mimic physical imperfections. By generating reproducible random inputs, these modules enable statistical analysis of device behavior under realistic conditions.

2.4 Practical setup

Hardware setup

Component

Memristor Device

SourceMeter / Function Gen

Measurement Instruments

Control Software (LabVIEW, Python)

Analysis Software (MATLAB, Python)

Environmental Control

Algorithm: Dynamic Memdiode Model (DMM)

Step 1: Initialization

1. **Define simulation parameters:**
 - Time step (dt), total simulation time, or number of voltage steps
 - Input voltage waveform ($V(t)$) – e.g., ramp, triangle
2. **Set physical model parameters:**
 - a_{on} , a_{off} : nonlinearity factors
 - I_{on} , I_{off} : current bounds
 - R_{on} , R_{off} : resistance limits
 - τ_{set} , τ_{reset} : time constants for SET and RESET
 - V_s , V_r : threshold voltages
 - R_i : input resistance
 - A_t : time constant scaling factor
 - γ : exponent for reset behavior
3. **Initialize state variables:**
 - Memory state $l(0) = 0$
 - Current $I(0) = 0$

Step 2: Loop Over Input Voltage Steps

For each time step $t = 1$ to N :

1. Compute Internal Voltage (V_i):
2. Compute Intermediate Variables:
3. Compute Time Constant T_{sr} :
4. Compute Current $I(t+1)$ using the Lambert W function:
5. Update Memory State $l(t+1)$:
6. Store Results:
 - Append $V(t)$, $I(t+1)$, and $l(t+1)$ to arrays for plotting

Step 3: Post-processing

- Plot **Current vs Voltage (I–V)** curve
- Plot **Memory state vs Voltage** (optional)
- Compute and plot **set/reset voltages**, if doing variability analysis

INPUT:

To simulate the Dynamic Memdiode Model (DMM) in Python, several key inputs must be defined before executing the model. The primary input is a time-dependent voltage waveform, typically a triangular or sinusoidal sweep, which serves as the external excitation applied across the device. This voltage input is represented as a numerical array and dictates how the memristor responds over time. In addition to the voltage signal, the simulation requires a set of physical and model-specific parameters. These include the ON and OFF state nonlinearities (a_{on} , a_{off}), the ON and OFF currents (I_{on} , I_{off}), the minimum and maximum resistance values (R_{on} , R_{off}), and the threshold voltages for switching (V_s , V_r).

Parameters governing the dynamic memory behavior such as τ_{set} , τ_{reset} (time constants for SET/RESET processes), A (a scaling factor for time evolution), γ (nonlinearity exponent), and R_i (series resistance) must also be specified. For more advanced simulations, random variability can be introduced in these parameters to emulate device-to-device or cycle-to-cycle variations, often using normal or log-normal distributions. Additionally, Gaussian noise can be superimposed on the input voltage to investigate stochastic resonance phenomena. Together, these inputs allow the DMM to model realistic memristive behavior under various electrical and environmental conditions

3 IMPLEMENTATION:

Reproduces I–V and memory state–voltage curves

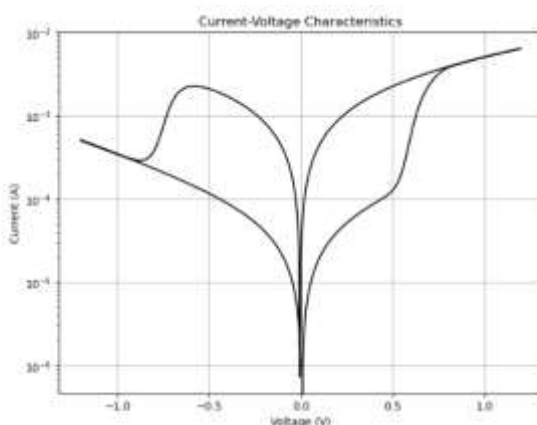
```

jupyter Untitled2 Last Checkpoint: 8 minutes ago
File Edit View Run Kernel Settings Help
JupyterLab Python [conda envbase]

In [1]: import numpy as np
import matplotlib.pyplot as plt
from scipy.special import lambertw
step = 0.1
V1 = np.arange(0, 1.2 + step, step)
V2 = np.arange(0, -1.2 - step, -step)
V = np.concatenate((V1, V2[1:-1]))
aoff, aon = 2, 2
R1 = 40
aoff, aon = 30, 30
loff, lon = 1e-4, 1e-3
etas, etar = 40, 15
gam = 0.1
AT = 1e-3
Vs, Vr = 0.5, -0.3
I = [0]
I = [0]
Vm, Im, Im = [], [], []
for t in range(len(V)):
    Vt = V[t] - R1 * I[t]
    I0 = (lon - loff) * I[t] + loff
    a = (aon - aoff) * I[t] + aoff
    r = (aon - aoff) * I[t] + aoff
    if Vt > 0:
        tau = np.exp(-etas * (Vt - Vs))
    else:
        tau = np.exp(-etar * I[t]**gam * (Vt - Vr))
    term1 = lambertw((a * R * I0 / 2) * np.exp(a * Vt)).real
    term2 = lambertw((a * R * I0 / 2) * np.exp(-a * Vt)).real
    I_next = (1 / (a * R)) * (term1 - term2)
    I.append(I_next)
    H = 1 if Vt > 0 else 0
    I_next = (I[t] - H) * np.exp(-AT / tau) + H
    I.append(I_next)
    Vm.append(Vt)
    Im.append(I[t])
    Im.append(I[t])
plt.figure(figsize=(8, 6))
plt.semilogy(Vm, np.abs(Im), color='black')
plt.xlabel("Voltage (V)")
plt.ylabel("Current (A)")
plt.title("Current-Voltage Characteristics")
plt.grid(True)
plt.show()

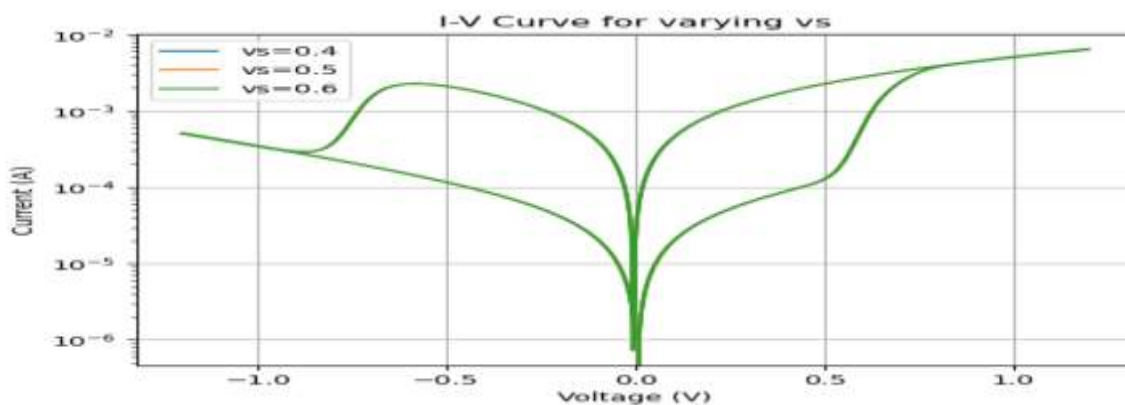
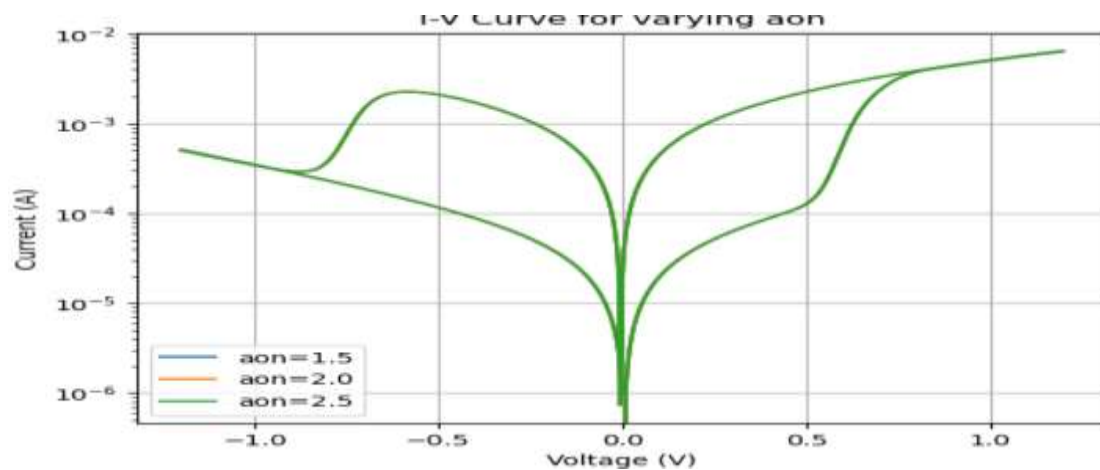
```

OUTPUT:



3.1 CURRENT VOLTAGE CURVES FOR BOTH VS AND AON:

```
6]: params = {'vs': [0.4, 0.5, 0.6], 'aon': [1.5, 2.0, 2.5]} # Example
for param, values in params.items():
    plt.figure()
    for val in values:
        # Change parameter dynamically
        if param == 'vs':
            Vs = val
        elif param == 'aon':
            aon = val
        # Run the same simulation loop as in the core model
        # Plot each result
        plt.semilogy(Vm, np.abs(Im), label=f"{param}={val}")
    plt.title(f"I-V Curve for varying {param}")
    plt.xlabel("Voltage (V)")
    plt.ylabel("Current (A)")
    plt.legend()
    plt.grid(True)
    plt.show()
```



3.1.2 Ramp Rate I-V Simulation:

```
[20]: import numpy as np
import matplotlib.pyplot as plt

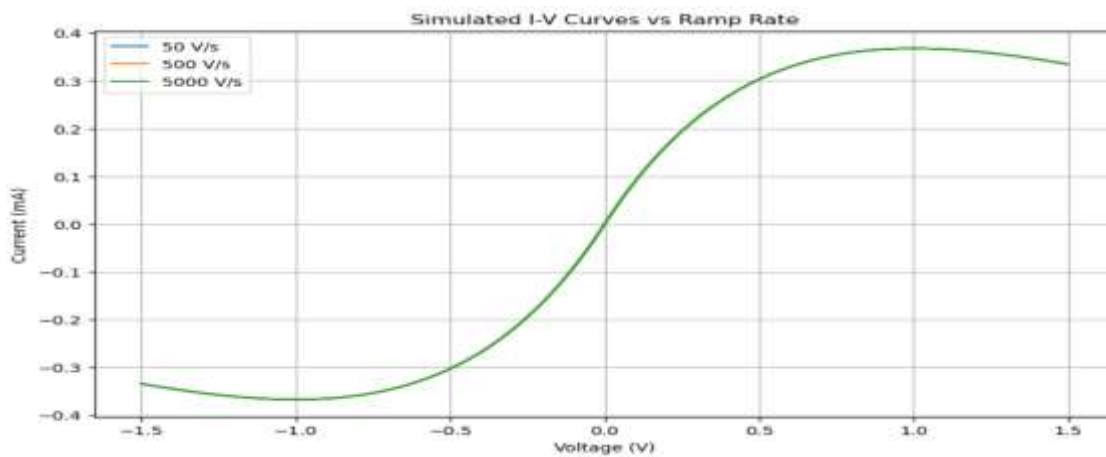
# Dummy memristor model (replace with actual model if available)
def memristor_model(V):
    # A basic non-linear approximation for current
    return 1e-3 * V * np.exp(-np.abs(V)) # I in A

# Define ramp rates in V/s
ramp_rates = [50, 500, 5000]
Vmax = 1.5 # Maximum voltage
n_points = 1000 # Points per simulation

plt.figure(figsize=(10, 6))

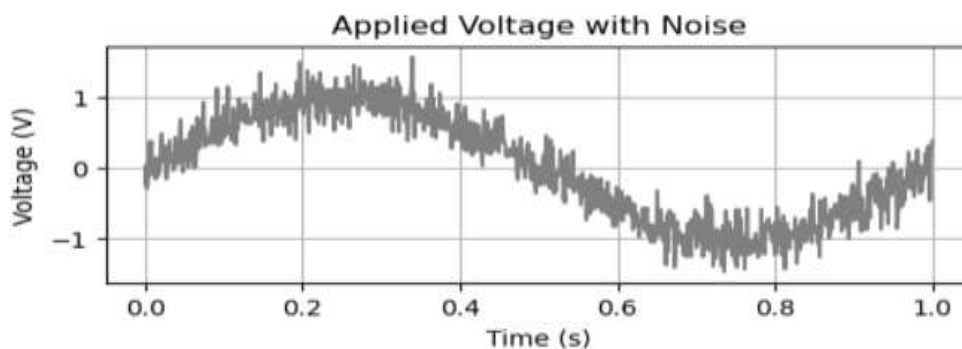
for rate in ramp_rates:
    t = np.linspace(0, 2 * Vmax / rate, n_points)
    V = Vmax * np.sin(2 * np.pi * t / t[-1]) # Simulated sinusoidal sweep
    I = memristor_model(V)
    plt.plot(V, I * 1e3, label=f'{rate} V/s') # Convert A to mA for plot

plt.xlabel("Voltage (V)")
plt.ylabel("Current (mA)")
plt.title("Simulated I-V Curves vs Ramp Rate")
plt.legend()
plt.grid(True)
plt.show()
```



3.1.3 Voltage vs Time with noise:

```
[36]: plt.figure(figsize=(6, 2))
plt.plot(t, V_noisy, color='gray')
plt.title("Applied Voltage with Noise")
plt.xlabel("Time (s)")
plt.ylabel("Voltage (V)")
plt.grid(True)
plt.show()
```



3.1.4 CURRENT VOLTAGE WITHOUT NOISE:

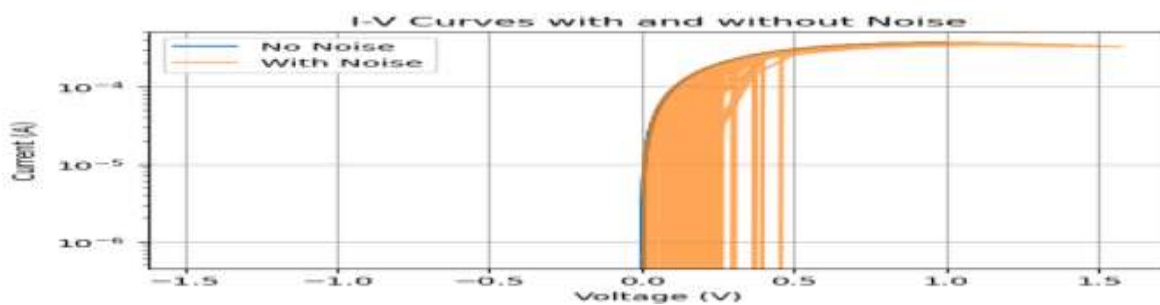
```
[34]: import numpy as np
import matplotlib.pyplot as plt

# Time vector
t = np.linspace(0, 1, 1000)
V_clean = np.sin(2 * np.pi * t)
noise = np.random.normal(0, 0.2, size=t.shape) # 200 mV noise
V_noisy = V_clean + noise

# Dummy memristor model (nonlinear I-V)
def memristor_model(V):
    return 1e-3 * V * np.exp(-np.abs(V)) # Simplified

I_clean = memristor_model(V_clean)
I_noisy = memristor_model(V_noisy)

# (a) Plot I-V curves
plt.figure(figsize=(6, 4))
plt.plot(V_clean, I_clean, label='No Noise')
plt.plot(V_noisy, I_noisy, label='With Noise', alpha=0.7)
plt.yscale("log")
plt.xlabel("Voltage (V)")
plt.ylabel("Current (A)")
plt.title("I-V Curves with and without Noise")
plt.legend()
plt.grid(True)
plt.show()
```



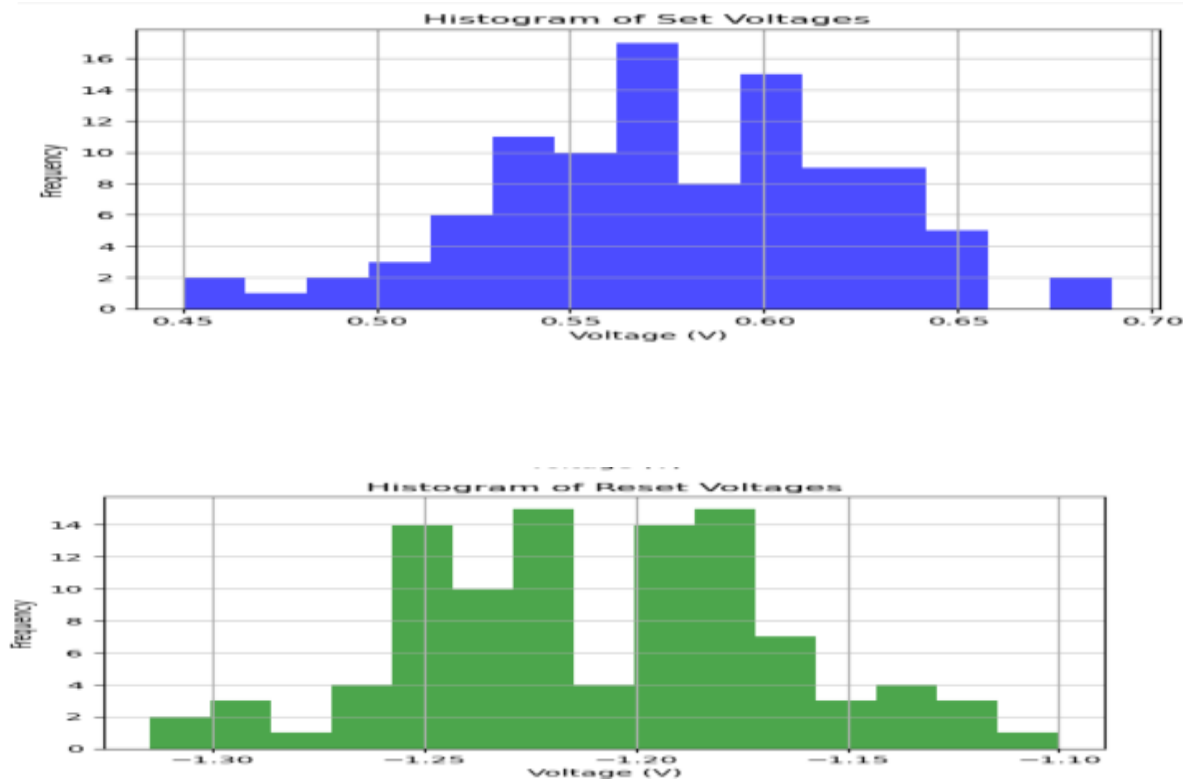
HISTOGRAMS OF SET AND RESET VOLTAGES:

```
[20]: set_voltages = []
reset_voltages = []

for i in range(ncyc):
    pos = np.where(np.array(lm_all[i]) >= 0.5)[0]
    if len(pos) > 1:
        set_voltages.append(Vm_all[i][pos[0]])
        reset_voltages.append(Vm_all[i][pos[-1]])

# Plot histogram for Set voltages
plt.hist(set_voltages, bins=15, color='blue', alpha=0.7)
plt.title("Histogram of Set Voltages")
plt.xlabel("Voltage (V)")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()

# Plot histogram for Reset voltages
plt.hist(reset_voltages, bins=15, color='green', alpha=0.7)
plt.title("Histogram of Reset Voltages")
plt.xlabel("Voltage (V)")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```



4 ACKNOWLEDGEMENT :

The author sincerely acknowledges the invaluable guidance, continuous support, and constructive feedback provided by Dr. S. China Venkateswarlu Dr . V. Siva Nagaraju faculty and Ms P.Ganga Bhavani and members of the Department of Electronics and Communication Engineering at the Institute of Aeronautical Engineering (IARE). Their expert advice and encouragement have been instrumental throughout the entire course of this research.

Special thanks are also extended to the faculty and staff of the Institute for providing a conducive academic environment and essential resources that greatly facilitated the successful completion of this work. The author appreciates the support and collaboration of peers and colleagues who contributed their time and expertise

REFERENCES:

Core Memristor Modeling and Simulation:

1. Strukov, D. B., Snider, G. S., Stewart, D. R., & Williams, R. S.
The missing memristor found
Nature, 2008.
<https://doi.org/10.1038/nature06932>
2. Biolek, D., Biolek, Z., & Biolková, V.
SPICE Model of Memristor with Nonlinear Dopant Drift
Radioengineering, 2009.
https://www.radioeng.cz/fulltexts/2009/09_03_01_04.pdf
3. Yakopcic, C., Taha, T. M., Subramanyam, G., & Pino, R. E.
Generalized Memristive Device SPICE Model and Its Application in Circuit Design
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2013.
<https://doi.org/10.1109/TCAD.2012.2234784>

4. Kvatinsky, S., Friedman, E. G., Kolodny, A., & Weiser, U.

TEAM: Threshold Adaptive Memristor Model

IEEE Transactions on Circuits and Systems I, 2013.

<https://doi.org/10.1109/TCSI.2013.2264950>

5. Wang, Z., Joshi, S., Savel'ev, S. E., Song, W., Midya, R., Li, Y., ... & Yang, J. J.

Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing

Nature Materials, 2017.

<https://doi.org/10.1038/nmat4756>



Variability and Noise in Memristor Modeling

6. Ambrogio, S., Balatti, S., Nardi, F., Ielmini, D.

Impact of variability in resistive switching memory—Part I: Experimental analysis of variability

IEEE Transactions on Electron Devices, 2014.

<https://doi.org/10.1109/TED.2014.2322632>

7. Zhang, M., Wang, Z., Rao, M., Midya, R., Li, C., Joshi, S., ... & Xia, Q.

Experimental demonstration of time-domain variability and stochasticity in RRAM devices

IEEE Electron Device Letters, 2019.

<https://doi.org/10.1109/LED.2019.2905112>

8. Chabi, D., Molas, G., Bichler, O., & Gamrat, C.

Robust learning with noisy memristor synapses

IEEE Transactions on Nanotechnology, 2015.

<https://doi.org/10.1109/TNANO.2015.2400462>



Simulation Tools and Techniques

9. Mehonic, A., & Kenyon, A. J.

Emulating synaptic plasticity with memristors

Philosophical Transactions of the Royal Society A, 2020.

<https://doi.org/10.1098/rsta.2019.0556>

10. Menzel, S., Böttger, U., & Wuttig, M.

Physics of the resistive switching in transition metal oxides

Advanced Functional Materials, 2015.

<https://doi.org/10.1002/adfm.201403100>