How to Use the IEEEtran LATEX Templates

IEEE Publication Technology Department

***Abstract*—This document describes the most common article elements and how to use the IEEEtran class with LATEX to produce files that are suitable for submission to the Institute of Electrical and Electronics Engineers (IEEE). IEEEtran can produce conference, journal and technical note (correspondence) papers with a suitable choice of class options.**

***Index Terms*—Class, IEEEtran, LATEX, paper, style, template, typesetting.**

I. Introduction

NUMEROUS memristor models have been proposed in the scientific literature following the discovery of memristive behavior at the nanoscale. The original model, introduced in [1], conceptualizes the memristor as a series connection of two variable resistances corresponding to the conductive and insulating regions of the thin film. To more accurately model Self-Directed Channel (SDC) memristors, M. Nugent and T. Molter proposed the generalized *Mean Metastable Switch* (MMS) model, which is a semi-empirical formulation. In this approach, the time derivative of the internal state variable is defined as a function of both the transition probabilities between metastable states and the current value of the state variable [2]. In [3], the authors attempted to model memristive behavior using the *Physics-Informed Neural Networks* (PINNs) framework. Although the reported results indicate the potential of this method, the study is not grounded in experimental measurements of physical memristor devices. Instead, the analysis is limited to comparisons with the outcomes of existing simulation models. Moreover, the training data employed during the neural network learning process were generated based on previously developed theoretical models, thereby limiting the ability to assess the method's accuracy in the context of real-world physical systems.

In [4], the authors introduce the concept of deep neural models in which the dynamics of the hidden state are governed by an *ordinary differential equation* (ODE). The training process is performed in an end-to-end manner, meaning that all parameters are optimized simultaneously within a single training routine. A key innovation of the proposed approach is a novel backpropagation technique, which relies on solving the corresponding adjoint ODE backward in time using *adjoint sensitivity methods*. This formulation enables efficient gradient computation and facilitates the application of neural ODEs in various architectures, including continuous-depth residual networks and generative flow-based models.

Manuscript created October, 2020; This work was developed by the IEEE Publication Technology Department.

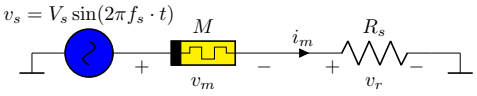

Fig. 1. Schematic diagram of the measurement setup used for the SDC memristor. The device is connected in series with a resistor, and the input voltage is applied via an arbitrary waveform generator. Voltage measurements are acquired using a data acquisition (DAQ) system.

In [5], the authors extend the classical Neural ODE framework to enable the modeling of discontinuous events in continuous time—without requiring prior knowledge of the number or timing of such events. The proposed differentiable event functions allow for efficient simulation of hybrid systems, such as systems with collisions, state-switching mechanisms, or point processes. This development opens up new possibilities for modeling and training systems with discrete control inputs and non-smooth dynamics.

The present study aims to investigate the feasibility of modeling Self-Directed Channel (SDC) memristors using artificial neural networks, and to compare the effectiveness of this approach with that of existing theoretical models. Specifically, we focus on the application of Neural Ordinary Differential Equation (Neural ODE) models to simulate the behavior of SDC memristors, utilizing experimental data obtained from real physical systems.

Our objective is to assess whether neural network-based models can accurately reproduce the dynamic characteristics of SDC memristors, and to determine whether they offer any advantages over traditional theoretical approaches—particularly in terms of modeling flexibility, accuracy, and applicability to real-world, measurement-driven scenarios.

II. Materials and Methods

This study utilizes experimental data obtained from physical Self-Directed Channel (SDC) memristors doped with tungsten. The structure and properties of these devices have been described in detail in the literature, e.g., in [6], [7].

The experimental setup, illustrated in Fig. 1, consisted of an SDC memristor connected in series with a resistor. The sinusoidal input voltage was supplied by an arbitrary waveform generator, while the voltage signals were measured using a data acquisition (DAQ) system.

Measurements were carried out for various combinations of supply voltage amplitudes and frequencies. Specifically, the amplitude of the applied voltage was varied as $V_s \in \{0.5,\ 1.0,\ 1.5\}$ [V], and the frequency was selected from the set $f \in \{1,\ 5,\ 10,\ 20,\ 50,\ 100\}$ [Hz].

A. Modeling with Neural Networks

The analyzed model is grounded in the general theoretical framework of memristive devices, originally introduced in [8], and is expressed through equations (1).

$$\begin{aligned} v(t) &= \mathcal{M}\big(\mathbf{x}(t), v(t)\big)\, i(t), \\ \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} &= f\big(\mathbf{x}(t), v(t)\big) \end{aligned} \tag{1}$$

where $\mathbf{x}$ denotes the vector of internal state variables, whose temporal evolution is governed by the dynamic function f, and $\mathcal{M}$ represents the memristance of the device. A fundamental challenge in memristor modeling lies in accurately identifying the functional forms of $\mathcal{M}(\mathbf{x}, v)$ and $f(\mathbf{x}, v)$.

In this study, two neural network–based modeling approaches are investigated. The first, denoted as **Det-MemODE**, employs a deterministic formulation of the memristor conductance. The second, referred to as **Dual-NN-MemODE**, adopts a data-driven strategy in which the conductance is directly modeled using neural networks. Although both approaches utilize neural networks to approximate the dynamic function $f(\mathbf{x}(t), v(t))$, they differ fundamentally in the representation of the memristor conductance.

In the first approach (Det-MemODE), the memristor conductance is defined on the basis of the physical mechanisms underlying self-directed channel (SDC) devices. In such memristors, the resistance evolves as a result of Ag^+ ion migration and the dynamic formation of conductive filaments. Consequently, the device conductance exhibits a continuous transition between distinct resistance states. This transition can be described as a weighted combination of the limiting resistance values, modulated by the internal state variable. Formally, the memristor conductance is expressed as $G_m(x) = \mathcal{M}(x)^{-1}$, where $\mathcal{M}(x)$ denotes the state-dependent resistance.

$$G_m(x) = \frac{x}{R_{\mathrm{ON}}} + \frac{1-x}{R_{\mathrm{OFF}}}, \tag{2}$$

where R_{ON} and R_{OFF} denote the resistances in the low-conductance and high-conductance states, respectively. For all of these approaches also the function $f(\mathbf{x}(t), v(t))$ was implemented using an artificial neural network, whose conceptual architecture is illustrated in Fig. 2 for the Det-MemODE and in Fig. 3 for the Dual-NN-MemODE.

The neural network consists of interconnected nodes (neurons) that process information through weighted connections. Each neuron in the network computes its output according to:

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3}$$

where w_i are the weights that determine the strength and importance of each input connection x_i, b is the bias term that provides an adjustable threshold for neuron activation, and σ is the activation function (e.g., sigmoid, ReLU, or tanh) that introduces non-linearity into the model.

The weights represent the learnable parameters that encode the relationship between inputs and outputs, while biases allow neurons to shift their activation threshold, enabling the network to better fit the training data even when all inputs are zero. During the optimization process, the network parameters (weights and biases) are tuned jointly with the values of R_{ON} and R_{OFF}.

The learning process involves iteratively adjusting the network parameters to minimize a loss function that quantifies the difference between predicted and target outputs. This is achieved through backpropagation algorithm combined with gradient descent optimization, where gradients of the loss function with respect to each parameter are computed and used

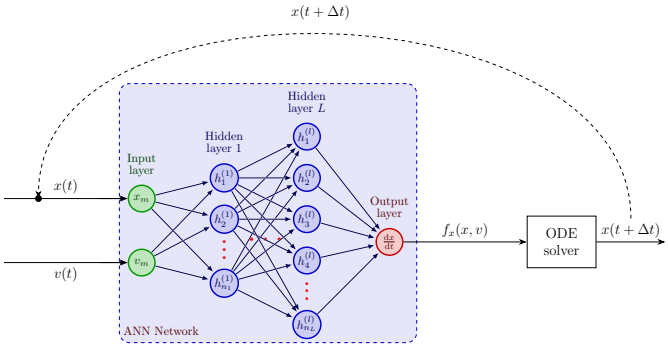

Fig. 2. Conceptual diagram illustrating the use of a neural network to simulate the dynamics of the memristor, for the Det-MemODE, with schematic representation of the Neural ODE framework, where an artificial neural network (ANN) computes the derivative function $f_x(x, v)$, which is then integrated by an ODE solver to predict the state at the next time step $x(t+\Delta t)$. The dashed feedback loop indicates the recurrent nature of the process, where the output state is fed back as input for subsequent predictions.

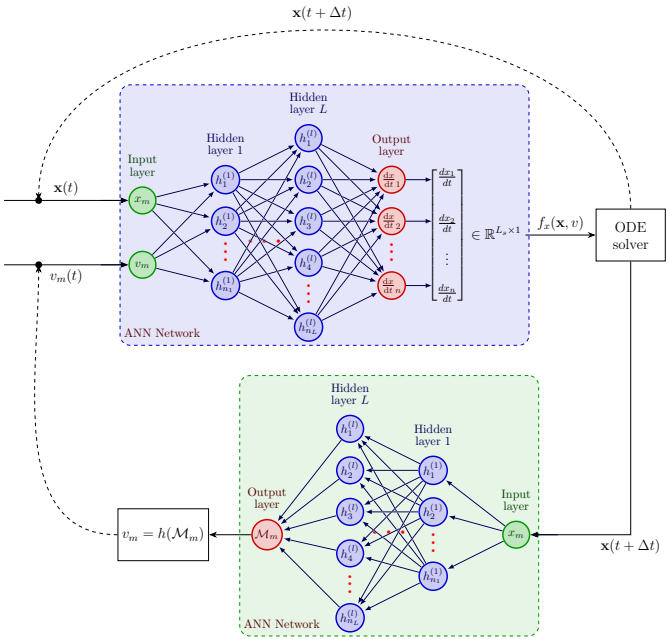

Fig. 3. Conceptual diagram illustrating the use of a neural network to simulate the dynamics of the memristor, for the Dual-NN-MemODE, with schematic representation of the Neural ODE framework, where an artificial neural network (ANN) computes the derivative function $f_x(x, v)$, which is then integrated by an ODE solver to predict the state at the next time step $x(t + \Delta t)$. The dashed feedback loop indicates the recurrent nature of the process, where the output state is fed back as input for subsequent predictions.

to update the weights and biases in the direction that reduces the overall error.

B. Neural Network Architecture Design

The neural network model development and training pipeline was implemented using the `PyTorch` deep learning framework, described in [9], within the `Python` programming environment, leveraging its automatic differentiation capabilities. To enable seamless integration of ordinary differential equations (ODEs) within the neural network training paradigm, the specialized `torchdiffeq` library was employed. This library provides differentiable ODE solvers that facilitate efficient gradient computation with respect to solution trajectories through the adjoint sensitivity method, well described in [4], [5], [10]. This approach enables end-to-end training of neural differential equation models by maintaining gradient flow through the numerical integration process, which is essential for learning dynamic systems governed by differential equations.

Specifically, the `dopri5` (Dormand-Prince 5th order) adaptive Runge-Kutta solver was utilized for its superior balance between computational efficiency and numerical accuracy. This solver employs embedded error estimation for adaptive step-size control, ensuring stable integration of the memristor dynamics while maintaining computational tractability during training.

The neural network architecture is illustrated in Figure 4 and incorporates a strategically designed bottleneck configuration with dimensional two time reduction in the initial and final hidden layers. This architectural choice addresses several critical aspects of neural ODE training and memristor system modeling, like:

- Dimensional reduction at the start and end of the network significantly decreases the total parameter count,
- Parameter count reduction contributes to more stable optimization dynamics by limiting the search space complexity
- Dimensional reduction helps mitigate gradient vanishing and explosion problems that commonly arise in deep networks processing long temporal sequences
- The bottleneck structure implicitly enforces a low-dimensional manifold assumption consistent with the underlying physics of memristor dynamics

The network weights were optimized using several optimization techniques, including the `Adam` (Adaptive Moment Estimation) optimizer. The choice of optimizer and its parameters was guided by hyperparameter tuning conducted with the `Optuna` framework, as described in [11]. The hyperparameter search space comprised the learning rate, weight decay, and batch size, while the optimal configuration was selected based on validation loss minimization. To enable adaptive control of the learning rate throughout training, the `ReduceLROnPlateau` scheduling strategy was employed. This performance-based scheduler dynamically adjusts the learning rate in response to plateaus in the monitored validation loss, thereby improving convergence stability. Specifically, the scheduler reduces the learning rate once the monitored loss stagnates, applies a predefined reduction factor, incorporates a patience interval before making adjustments, and enforces lower and upper bounds to prevent instability. This implementation strategy ensures robust and efficient training of neural differential equation models for memristor circuit simulation, integrating state-of-the-art deep learning practices with domain-specific techniques for dynamic system modeling.

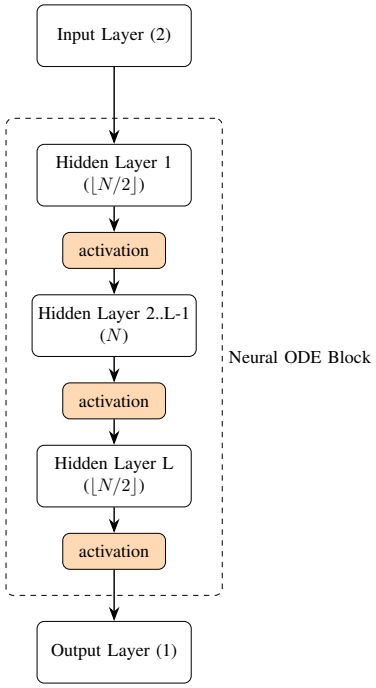

Fig. 4. Architecture of the feedforward neural network used within the Neural ODE framework for memristor dynamics modeling. The network consists of an input layer receiving two inputs, multiple hidden layers with decreasing-increasing-decreasing neuron counts ($\lfloor N/2 \rfloor, N, \lfloor N/2 \rfloor$), activation functions between layers, and a single output. The Neural ODE block encompasses the core computational layers that approximate the derivative function $f(x(t), v(t))$ in the differential equation system.

C. Deterministic Memristor Model for Comparison

To evaluate the effectiveness of the proposed model, it was compared with the two deterministic models: the Mean Metastable Switch (MMS) model introduced in [12] and the Generalized Mean Metastable Switch Memristor (GMMS) model described in [2], [13].

1) Mean Metastable Switch (MMS) Model: The Mean Metastable Switch (MMS) model is a simplified representation of memristor behavior, focusing on the average switching characteristics rather than the detailed dynamics. It captures the essential features of memristive switching by modeling the device as a two-state system, where the resistance can switch between a high-resistance state (HRS) and a low-resistance state (LRS) based on the applied voltage. This model describes the memristor dynamics using Eq. (4).

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{1}{\tau}\left[f_{\mathrm{on}}(t)(1-x) - f_{\mathrm{off}}(t)x\right] \tag{4}$$

where

$$f_{\text{on}}(t) = \frac{1}{1 + e^{-\beta(v(t)-V_{\text{ON}})}} \tag{5}$$

$$f_{\text{off}}(t) = 1 - \frac{1}{1 + e^{-\beta(v(t)-V_{\text{OFF}})}} \tag{6}$$

where $\beta = \frac{q}{kT}$ is a temperature-related parameter, V_{ON} denotes the switching voltage to the low-resistance state (LRS), V_{OFF} denotes the switching voltage to the high-resistance state (HRS), k is the Boltzmann constant, T is the absolute temperature, and q represents the elementary charge. The instantaneous conductance of the memristor is defined as in Eq. (2).

2) Generalized Mean Metastable Switch (GMMS) Model: In [2], the authors introduced the Generalized Mean Metastable Switch (GMMS) model, in which the memristor is represented as a parallel connection of two memristors and a Schottky diode. This formulation extends the original metastable switch concept by incorporating nonlinear current–voltage characteristics, thereby enabling a more accurate representation of memristor behavior across a wide range of operating conditions.

Nevertheless, the inclusion of additional parameters and nonlinearities increases the model's complexity and poses challenges for parameter optimization. In this framework, the evolution of the state variable is governed by Eq. (4), while the current–voltage relationship is described by the following set of equations:

$$\begin{aligned} i(t) &= \phi i_m(v,t) + (1-\phi) i_d(v) \\ i_d(v) &= \alpha_f e^{\beta_f v} - \alpha_r e^{-\beta_r v} \\ i_m(v,t) &= \mathcal{G}_m(x) v \end{aligned} \tag{7}$$

where $\phi \in [0,1]$ is a parameter representing the fraction of the total current flowing through the memristor relative to the entire current through the device. The parameters α_f, β_f, α_r, and β_r are positive constants characterizing the forward and reverse current behavior along the Schottky barrier.

In the case of the GMMS model, an additional challenge arises due to the Schottky effect, which introduces nonlinearity into the current–voltage characteristic. When the memristor is connected in series with a resistor, this requires solving a nonlinear equation of the form Eq. (8). As a result, the system of ordinary differential equations is transformed into a differential-algebraic equation (DAE).

$$\begin{cases} v_s(t) = v_r(t) + v_m(t) \\ \dfrac{\mathrm{d}x}{\mathrm{d}t} = \dfrac{1}{\tau}\big[f_{\text{on}}(t)(1-x) - f_{\text{off}}(t)x\big] \end{cases} \tag{8}$$

where $v_s(t)$ denotes the supply voltage, $v_r(t) = R_s i(t)$ is the voltage across the resistor, and $v_m(t)$ is the voltage across the memristor.

D. Objective Function

The loss function based on the phase portrait is designed to simultaneously account for both the signal values and its dynamic structure. To this end, it is defined as the sum of four components, each serving a distinct and complementary role in the optimization process. The total loss function is given by:

$$\begin{aligned} \mathcal{L}_{\text{total}} = \lambda_{\text{base}} \cdot \mathcal{L}_{\text{base}} + \lambda_{\text{vel}} \cdot \mathcal{L}_v \\ + \lambda_{\text{curv}} \cdot \mathcal{L}_c + \sum_{i=1}^{N_c} \lambda_{\text{con}_i} \cdot \mathcal{L}_{\text{con}_i}(x) \end{aligned} \tag{9}$$

Here, $\mathcal{L}_{\text{base}}$ denotes the primary error term, measuring the difference between predicted and reference values, in that case the mean squared error (MSE). The term $\mathcal{L}_v$ enforces accurate modeling of the velocity characteristics by considering the derivatives of trajectories in phase space. The component $\mathcal{L}_c$ promotes consistency of the trajectories in terms of their curvature, for example by analyzing changes in trajectory direction or curvature. Finally, $\mathcal{L}_{\text{con}}(x)$ introduces additional physical or structural constraints that the model is required to satisfy, including conditions of stability, nonlinearities, or conformity to the underlying physical model. The weighting coefficients λ_{base}, λ_{vel}, and λ_{curv} balance the influence of each term within the overall loss function according to the priorities of the modeling task, and each of the values were calculated during each optimization step using the Geometric Loss Strategy (GLS), described in [14], [15]. with the following algorithm presented in Algorithm 1.

Algorithm 1 Adaptive Loss Balancing with Clamping (Compact)

Require: Loss terms $\mathcal{L}_{\text{base}}, \mathcal{L}_{\text{vel}}, \mathcal{L}_{\text{curv}}$, constant $\varepsilon > 0$, bounds $[\lambda_{\min}, \lambda_{\max}]$
Ensure: Adaptive weights $\lambda_{\text{base}}, \lambda_{\text{vel}}, \lambda_{\text{curv}}$
1: Collect losses: $\mathbf{L} \leftarrow [\mathcal{L}_{\text{base}}, \mathcal{L}_{\text{vel}}, \mathcal{L}_{\text{curv}}]$
2: Compute geometric mean: $\overline{\mathcal{L}}_{\text{geo}} \leftarrow \exp\left(\frac{1}{|\mathbf{L}|}\sum_{i=1}^{|\mathbf{L}|} \ln(\mathcal{L}_i + \varepsilon)\right)$
3: **for** each loss $\mathcal{L}_i \in \mathbf{L}$ **do**
4: $\quad \lambda_i \leftarrow \frac{\overline{\mathcal{L}}_{\text{geo}}}{\mathcal{L}_i + \varepsilon}$
5: $\quad \lambda_i \leftarrow \text{clamp}(\lambda_i, \lambda_{\min}, \lambda_{\max})$
6: **end for**
7: **return** $(\lambda_{\text{base}}, \lambda_{\text{vel}}, \lambda_{\text{curv}})$

For circuit trajectory datasets indexed by $k = 1, \ldots, N_{\text{traj}}$, a trajectory-specific standardization procedure is implemented to normalize each electrical variable $z \in \{v, i\}$ (representing voltage and current measurements, respectively). This normalization process utilizes the standard deviation computed independently for each individual trajectory sequence, ensuring statistical consistency within each temporal dataset. The trajectory-specific standard deviation is calculated using the unbiased sample estimator:

$$\sigma_{z,k} = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}\left(z_{k,t}^{\text{true}} - \bar{z}_k^{\text{true}}\right)^2} \tag{10}$$

where the trajectory-specific temporal mean is defined as:

$$\bar{z}_k^{\text{true}} = \frac{1}{T}\sum_{t=1}^{T} z_{k,t}^{\text{true}} \tag{11}$$

The parameter T represents the temporal resolution of the trajectory, corresponding to the total number of discrete time-series sampling points within each measurement sequence.

This trajectory-wise normalization methodology serves several critical analytical objectives in circuit analysis and machine learning applications. Primarily, it establishes scale invariance across electrical signals exhibiting disparate amplitude characteristics, thereby facilitating quantitative comparison of dynamic waveform morphologies independent of absolute magnitude variations. Furthermore, this standardization approach enhances the numerical stability and convergence properties of optimization algorithms employed in neural network training procedures.

By implementing trajectory-specific normalization rather than global standardization, the methodology preserves the inherent temporal dynamics and statistical properties unique to each circuit configuration while simultaneously ensuring that amplitude disparities do not introduce systematic bias during pattern recognition, feature extraction, or predictive modeling procedures. This approach is particularly advantageous in scenarios involving heterogeneous circuit topologies or varying operational conditions, where maintaining the relative temporal structure of electrical phenomena is paramount for accurate system identification and dynamic analysis. The neural network loss function comprises several specialized components designed to optimize memristor circuit modeling performance across multiple dynamical characteristics. Each component addresses specific aspects of the temporal and nonlinear behavior exhibited by memristive systems.

The fundamental component of the loss function is responsible for model fitting to experimental voltage-current data, incorporating trajectory-specific normalization procedures. This component minimizes the mean squared error (MSE) between predicted and ground-truth voltage and current waveforms, normalized by the standard deviation of each individual trajectory. This normalization ensures scale invariance across datasets with varying amplitude characteristics.The mean squared error for a sequence of n data points is defined in Eq (12):

$$\mathrm{MSE}(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{12}$$

The base loss is formally expressed as it shown in Eq. (13):

$$\begin{aligned}\mathcal{L}_{\text{base}} = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\Bigg[&\mathrm{MSE}\left(\frac{v_{\text{pred},k}}{\sigma_{v,k}}, \frac{v_{\text{true},k}}{\sigma_{v,k}}\right) \\ &+ \mathrm{MSE}\left(\frac{i_{\text{pred},k}}{\sigma_{i,k}}, \frac{i_{\text{true},k}}{\sigma_{i,k}}\right)\Bigg]\end{aligned} \tag{13}$$

where $v_{\text{pred},k}$ and $i_{\text{pred},k}$ represent the predicted voltage and current for trajectory k, while $v_{\text{true},k}$ and $i_{\text{true},k}$ denote the corresponding experimental measurements.

The first-order dynamics comparison enables evaluation of temporal derivative agreement between model predictions and experimental data, particularly critical for capturing rapid memristor state transitions. This component ensures that the neural network learns not only absolute signal values but also their instantaneous rates of change, which is fundamental for accurate modeling of dynamic systems exhibiting switching phenomena. The first-order loss component is defined as:

$$\begin{aligned}\mathcal{L}_v = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\Bigg[&\mathrm{MSE}\left(\frac{\frac{\mathrm{d}v}{\mathrm{d}t}_{\text{pred},k}}{\sigma_{\dot{v},k}}, \frac{\frac{\mathrm{d}v}{\mathrm{d}t}_{\text{true},k}}{\sigma_{\dot{v},k}}\right) \\ &+ \mathrm{MSE}\left(\frac{\frac{\mathrm{d}i}{\mathrm{d}t}_{\text{pred},k}}{\sigma_{\dot{i},k}}, \frac{\frac{\mathrm{d}i}{\mathrm{d}t}_{\text{true},k}}{\sigma_{\dot{i},k}}\right)\Bigg]\end{aligned} \tag{14}$$

This formulation captures the velocity characteristics of electrical variables, ensuring that transient behaviors and switching dynamics are accurately reproduced by the model.

The second-order comparison term addresses trajectory curvature characteristics, which are essential for capturing subtle variations in system dynamics that manifest as nonlinear memristor behaviors. This component facilitates superior representation of curvature properties within the phase-space trajectory, particularly important for modeling systems exhibiting complex nonlinear and highly dynamic responses. The curvature loss is formulated as:

$$\begin{aligned}\mathcal{L}_c = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\Bigg[&\mathrm{MSE}\left(\frac{\frac{\mathrm{d}^2v}{\mathrm{d}t^2}_{\text{pred},k}}{\sigma_{\ddot{v},k}}, \frac{\frac{\mathrm{d}^2v}{\mathrm{d}t^2}_{\text{true},k}}{\sigma_{\ddot{v},k}}\right) \\ &+ \mathrm{MSE}\left(\frac{\frac{\mathrm{d}^2i}{\mathrm{d}t^2}_{\text{pred},k}}{\sigma_{\ddot{i},k}}, \frac{\frac{\mathrm{d}^2i}{\mathrm{d}t^2}_{\text{true},k}}{\sigma_{\ddot{i},k}}\right)\Bigg]\end{aligned} \tag{15}$$

By incorporating second-order temporal derivatives, this component ensures that acceleration characteristics and higher-order dynamical features are preserved during the learning process.

To enforce parameter bounds within specified intervals, a sophisticated constraint loss function utilizing smooth sigmoid transitions is implemented. This component prevents parameter drift beyond physically meaningful ranges while maintaining differentiability for gradient-based optimization algorithms. The constraint loss is mathematically expressed as:

$$\begin{aligned}\mathcal{L}_{\text{con}}(x) = \mathbb{E}\Bigg[\Bigg(&\frac{1}{1+\exp\left(\frac{x-a}{f}\right)}\cdot(a-x) \\ &+ \frac{1}{1+\exp\left(\frac{-(x-b)}{f}\right)}\cdot(x-b)\Bigg)^2\Bigg]\end{aligned} \tag{16}$$

where the constituent parameters are defined as follows, $\sigma(z) = \frac{1}{1+e^{-z}}$ represents the sigmoid activation function (implicitly used), a and b denote the lower and upper bounds of the admissible parameter range, respectively, f is a smoothing (scaling) factor controlling the steepness of the transition near the bounds, the term $\frac{1}{1+\exp((x-a)/f)}(a-x)$ penalizes values of x below the lower bound a, the term $\frac{1}{1+\exp(-(x-b)/f)}(x-b)$ penalizes values of x above the upper bound b, squaring ensures the penalty is non-negative and grows quadratically with the violation magnitude, $\mathbb{E}[\cdot]$ denotes the expectation operator (ensemble average over samples or trajectories).

This document applies to version 1.8b of IEEEtran.

The IEEEtran template package contains the following example files:

bare_jrnl.tex
bare_conf.tex
bare_jrnl_compsoc.tex
bare_conf_compsoc.tex
bare_jrnl_comsoc.tex

These are "bare bones" templates to quickly understand the document structure.

It is assumed that the reader has a basic working knowledge of LaTeX. Those who are new to LaTeX are encouraged to read Tobias Oetiker's "The Not So Short Introduction to LaTeX", available at: http://tug.ctan.org/info/lshort/english/lshort.pdf which provides an overview of working with LaTeX.

III. The Design, Intent and Limitations of the Templates

The templates are intended to **approximate the final look and page length of the articles/papers**. Therefore, **they are NOT intended to be the final produced work that is displayed in print or on IEEEXplore®**. They will help to give the authors an approximation of the number of pages that will be in the final version. The structure of the LaTeXfiles, as designed, enable easy conversion to XML for the composition systems used by the IEEE's outsource vendors. The XML files are used to produce the final print/IEEEXplore® pdf and then converted to HTML for IEEEXplore®. Have you looked at your article/paper in the HTML version?

IV. LaTeX Distributions: Where to Get Them

IEEE recommends using the distribution from the TeXUser Group at http://www.tug.org. You can join TUG and obtain a DVD distribution or download for free from the links provided on their website: http://www.tug.org/texlive/. The DVD includes distributions for Windows, Mac OS X and Linux operating systems.

V. Where to get the IEEEtran Templates

The **IEEE Template Selector** will always have the most up-to-date versions of the LaTeX and MSWord templates. Please see: https://template-selector.ieee.org/ and follow the steps to find the correct template for your intended publication. Many publications use the IEEETran LaTeX templates, however, some publications have their own special templates. Many of these are based on IEEEtran, but may have special instructions that vary slightly from those in this document.

VI. Where to get LaTeX help - user groups

The following on-line groups are very helpful to beginning and experienced LaTeX users. A search through their archives can provide many answers to common questions.

http://www.latex-community.org/
https://tex.stackexchange.com/

VII. Document Class Options in IEEEtran

At the beginning of your LaTeX file you will need to establish what type of publication style you intend to use. The following list shows appropriate documentclass options for each of the types covered by IEEEtran.

Regular Journal Article
`\documentclass[journal]IEEEtran`

Conference Paper
`\documentclass[conference]IEEEtran`

Computer Society Journal Article
`\documentclass[10pt,journal,compsoc]IEEEtran`

Computer Society Conference Paper
`\documentclass[conference,compsoc]IEEEtran`

Communications Society Journal Article
`\documentclass[journal,comsoc]IEEEtran`

Brief, Correspondence or Technote
`\documentclass[9pt,technote]IEEEtran`

There are other options available for each of these when submitting for peer review or other special requirements. IEEE recommends to compose your article in the base 2-column format to make sure all your equations, tables and graphics will fit the final 2-column format. Please refer to the document "IEEEtran_HOWTO.pdf" for more information on settings for peer review submission if required by your EIC.

VIII. How to Create Common Front Matter

The following sections describe general coding for these common elements. Computer Society publications and Conferences may have their own special variations and will be noted below.

A. Paper Title

The title of your paper is coded as:

```
\title{The Title of Your Paper}
```

Please try to avoid the use of math or chemical formulas in your title if possible.

B. Author Names and Affiliations

The author section should be coded as follows:

```
\author{Masahito Hayashi
\IEEEmembership{Fellow, IEEE}, Masaki Owari
\thanks{M. Hayashi is with Graduate School
of Mathematics, Nagoya University, Nagoya,
Japan}
\thanks{M. Owari is with the Faculty of
Informatics, Shizuoka University,
Hamamatsu, Shizuoka, Japan.}
}
```

Be sure to use the \IEEEmembership command to identify IEEE membership status. Please see the "IEEEtran_HOWTO.pdf" for specific information on coding authors for Conferences and Computer Society publications. Note that the closing curly brace for the author group comes at the end of the thanks group. This will prevent you from creating a blank first page.

C. Running Heads

The running heads are declared by using the \markboth command. There are two arguments to this command: the first contains the journal name information and the second contains the author names and paper title.

```
\markboth{Journal of Quantum Electronics,
Vol. 1, No. 1, January 2021}
{Author1, Author2,
\MakeLowercase{\textit{(et al.)}:
Paper Title}
```

D. Copyright Line

For Transactions and Journals papers, this is not necessary to use at the submission stage of your paper. The IEEE production process will add the appropriate copyright line. If you are writing a conference paper, please see the "IEEEtran_HOWTO.pdf" for specific information on how to code "Publication ID Marks".

E. Abstracts

The abstract is the first element of a paper after the \maketitle macro is invoked. The coding is simply:

```
\begin{abstract}
Text of your abstract.
\end{abstract}
```

Please try to avoid mathematical and chemical formulas in the abstract.

F. Index Terms

The index terms are used to help other researchers discover your paper. Each society may have it's own keyword set. Contact the EIC of your intended publication for this list.

```
\begin{IEEEkeywords}
Broad band networks, quality of service
\end{IEEEkeywords}
```

IX. How to Create Common Body Elements

The following sections describe common body text elements and how to code them.

A. Initial Drop Cap Letter

The first text paragraph uses a "drop cap" followed by the first word in ALL CAPS. This is accomplished by using the \IEEEPARstart command as follows:

```
\IEEEPARstart{T}{his} is the first paragraph
of your paper. . .
```

B. Sections and Subsections

Section headings use standard LaTeX commands: \section, \subsection and \subsubsection. Numbering is handled automatically for you and varies according to type of publication. It is common to not indent the first paragraph following a section head by using \noindent as follows:

```
\section{Section Head}
\noindent The text of your paragraph . . .
```

C. Citations to the Bibliography

The coding for the citations are made with the LaTeX \cite command. This will produce individual bracketed reference numbers in the IEEE style. At the top of your LaTeX file you should include:

```
\usepackage{cite}
```

For a single citation code as follows:

```
see \cite{ams}
```

This will display as: see **ams**

For multiple citations code as follows:

```
\cite{ams,oxford,lacomp}
```

This will display as **ams**, **oxford**, **lacomp**

D. Figures

Figures are coded with the standard LaTeX commands as follows:

```
\begin{figure}[!t]
\centering
\includegraphics[width=2.5in]{fig1}
\caption{This is the caption for one fig.}
\label{fig1}
\end{figure}
```

The [!t] argument enables floats to the top of the page to follow IEEE style. Make sure you include:

```
\usepackage{graphicx}
```

at the top of your LaTeXfile with the other package declarations.

To cross-reference your figures in the text use the following code example:

```
See figure \ref{fig1} ...
```

This will produce:
See figure 5 . . .

E. Tables

Tables should be coded with the standard LaTeX coding. The following example shows a simple table.

```
\begin{table}
\begin{center}
\caption{Filter design equations  ...}
\label{tab1}
```

Fig. 5. This is the caption for one fig.

TABLE I
A SIMPLE TABLE EXAMPLE.

Order of filter	Arbitrary coefficients e_m	coefficients b_{ij}
1	$b_{ij} = \hat{e}.\hat{\beta_{ij}}$,	$b_{00} = 0$
2	$\beta_{22} = (\ 1, -1, -1,\ \ 1,\ \ 1,\ \ 1)$	
3	$b_{ij} = \hat{e}.\hat{\beta_{ij}}$,	$b_{00} = 0$,

```
\begin{tabular}{| c | c | c |}
\hline
Order & Arbitrary coefficients &
coefficients\\
of filter & \(e_m\) &   \(b_{ij}\) \\
\hline
1& \(b_{ij}=\hat{e}.\hat{\beta_{ij}}\),
& \(b_{00}=0\)\\
\hline
2&\(\beta_{22}=(~1,-1,-1,~~1,~~1,~~1)\) &\\
\hline
3& \(b_{ij}=\hat{e}.\hat{\beta_{ij}}\),
& \(b_{00}=0\),\\
\hline
\end{tabular}
\end{center}
\end{table}
```

To reference the table in the text, code as follows:

```
Table~\ref{tab1} lists the closed-form...
```

to produce:

Table I lists the closed-form . . .

F. Lists

In this section, we will consider three types of lists: simple unnumbered, numbered and bulleted. There have been numerous options added to IEEEtran to enhance the creation of lists. If your lists are more complex than those shown below, please refer to the "IEEEtran_HOWTO.pdf" for additional options.

A plain unnumbered list

bare_jrnl.tex
bare_conf.tex
bare_jrnl_compsoc.tex
bare_conf_compsoc.tex
bare_jrnl_comsoc.tex

coded as:

```
\begin{list}{}{}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{list}
```

A simple numbered list

1) bare_jrnl.tex
2) bare_conf.tex
3) bare_jrnl_compsoc.tex
4) bare_conf_compsoc.tex
5) bare_jrnl_comsoc.tex

coded as:

```
\begin{enumerate}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{enumerate}
```

A simple bulleted list

- bare_jrnl.tex
- bare_conf.tex
- bare_jrnl_compsoc.tex
- bare_conf_compsoc.tex
- bare_jrnl_comsoc.tex

coded as:

```
\begin{itemize}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{itemize}
```

G. Other Elements

For other less common elements such as Algorithms, Theorems and Proofs, and Floating Structures such as page-wide tables, figures or equations, please refer to the "IEEEtran_HOWTO.pdf" section on "Double Column Floats."

X. How to Create Common Back Matter Elements

The following sections demonstrate common back matter elements such as Acknowledgments, Bibliographies, Appendicies and Author Biographies.

A. Acknowledgments

This should be a simple paragraph before the bibliography to thank those individuals and institutions who have supported your work on this article.

```
\section{Acknowledgments}
\noindent Text describing those who
supported your paper.
```

B. Bibliographies

References Simplified: A simple way of composing references is to use the `\bibitem` macro to define the beginning of a reference as in the following examples:

[6] H. Sira-Ramirez. "On the sliding mode control of nonlinear systems," *Systems & Control Letters*, vol. 19, pp. 303–312, 1992.
coded as:

```
\bibitem{Sira3}
H. Sira-Ramirez. ``On the sliding mode
control of nonlinear systems,''
\textit{Systems \& Control Letters},
vol. 19, pp. 303--312, 1992.
```

[7] A. Levant."Exact differentiation of signals with unbounded higher derivatives," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, California, USA, pp. 5585–5590, 2006.
coded as:

```
\bibitem{Levant}
A. Levant. ``Exact differentiation of
signals with unbounded higher
derivatives,''  in \textit{Proceedings
of the 45th IEEE Conference on
Decision and Control}, San Diego,
California, USA, pp. 5585--5590, 2006.
```

[8] M. Fliess, C. Join, and H. Sira-Ramirez. "Non-linear estimation is easy," *International Journal of Modelling, Identification and Control*, vol. 4, no. 1, pp. 12–27, 2008.
coded as:

```
\bibitem{Cedric}
M. Fliess, C. Join, and H. Sira-Ramirez.
``Non-linear estimation is easy,''
\textit{International Journal of Modelling,
Identification and Control}, vol. 4,
no. 1, pp. 12--27, 2008.
```

[9] R. Ortega, A. Astolfi, G. Bastin, and H. Rodriguez. "Stabilization of food-chain systems using a port-controlled Hamiltonian description," in *Proceedings of the American Control Conference*, Chicago, Illinois, USA, pp. 2245–2249, 2000.
coded as:

```
\bibitem{Ortega}
R. Ortega, A. Astolfi, G. Bastin, and H.
Rodriguez. ``Stabilization of food-chain
systems using a port-controlled Hamiltonian
description,'' in \textit{Proceedings of the
American Control Conference}, Chicago,
Illinois, USA, pp. 2245--2249, 2000.
```

C. Accented Characters in References

When using accented characters in references, please use the standard LaTeX coding for accents. **Do not use math coding for character accents**. For example:

```
\'e, \"o, \`a, \~e
```

will produce: é, ö, à, ẽ

D. Use of BibTeX

If you wish to use BibTeX, please see the documentation that accompanies the IEEEtran Bibliography package.

E. Biographies and Author Photos

Authors may have options to include their photo or not. Photos should be a bit-map graphic (.tif or .jpg) and sized to fit in the space allowed. Please see the coding samples below:

```
\begin{IEEEbiographynophoto}{Jane Doe}
Biography text here without a photo.
\end{IEEEbiographynophoto}
```

or a biography with a photo

```
\begin{IEEEbiography}[{\includegraphics
[width=1in,height=1.25in,clip,
keepaspectratio]{fig1.png}}]
{IEEE Publications Technology Team}
In this paragraph you can place
your educational, professional background
and research and other interests.
\end{IEEEbiography}
```

Please see the end of this document to see the output of these coding examples.

XI. Mathematical Typography and Why It Matters

Typographical conventions for mathematical formulas have been developed to **provide uniformity and clarity of presentation across mathematical texts**. This enables the readers of those texts to both understand the author's ideas and to grasp new concepts quickly. While software such as LaTeX and MathType® can produce aesthetically pleasing math when used properly, it is also very easy to misuse the software, potentially resulting in incorrect math display.

IEEE aims to provide authors with the proper guidance on mathematical typesetting style and assist them in writing the best possible article.

As such, IEEE has assembled a set of examples of good and bad mathematical typesetting. You will see how various issues are dealt with. The following publications have been referenced in preparing this material:

Mathematics into Type, published by the American Mathematical Society
The Printing of Mathematics, published by Oxford University Press
The LaTeXCompanion, by F. Mittelbach and M. Goossens
More Math into LaTeX, by G. Grätzer
AMS-StyleGuide-online.pdf, published by the American Mathematical Society

Further examples can be seen at http://journals.ieeeauthorcenter.ieee.org/wp-content/uploads/sites/7/IEEE-Math-Typesetting-Guide.pdf

A. Display Equations

A simple display equation example shown below uses the "equation" environment. To number the equations, use the `\label` macro to create an identifier for the equation. LaTeX will automatically number the equation for you.

$$x = \sum_{i=0}^{n} 2iQ. \tag{17}$$

is coded as follows:

```
\begin{equation}
\label{deqn_ex1}
x = \sum_{i=0}^{n} 2{i} Q.
\end{equation}
```

To reference this equation in the text use the `\ref` macro. Please see (17)
is coded as follows:

```
Please see (\ref{deqn_ex1})
```

B. Equation Numbering

Consecutive Numbering: Equations within an article are numbered consecutively from the beginning of the article to the end, i.e., (1), (2), (3), (4), (5), etc. Do not use roman numerals or section numbers for equation numbering.

Appendix Equations: The continuation of consecutively numbered equations is best in the Appendix, but numbering as (A1), (A2), etc., is permissible.

Hyphens and Periods: Hyphens and periods should not be used in equation numbers, i.e., use (1a) rather than (1-a) and (2a) rather than (2.a) for sub-equations. This should be consistent throughout the article.

C. Multi-line equations and alignment

Here we show several examples of multi-line equations and proper alignments.

A single equation that must break over multiple lines due to length with no specific alignment.

$$\begin{multline}\text{The first line of this example}\\ \text{The second line of this example}\\ \text{The third line of this example}\end{multline} \tag{18}$$

is coded as:

```
\begin{multline}
\text{The first line of this example}\\
\text{The second line of this example}\\
\text{The third line of this example}
\end{multline}
```

A single equation with multiple lines aligned at the = signs

$$\begin{aligned} a &= c+d \\ b &= e+f \end{aligned} \tag{19, 20}$$

is coded as:

```
\begin{align}
a &= c+d \\
b &= e+f
\end{align}
```

The `align` environment can align on multiple points as shown in the following example:

$$\begin{aligned} x &= y & X &= Y & a &= bc \\ x' &= y' & X' &= Y' & a' &= bz \end{aligned} \tag{21, 22}$$

is coded as:

```
\begin{align}
x &= y & X & =Y & a &=bc\\
x' &= y' & X' &=Y' &a' &=bz
\end{align}
```

D. Subnumbering

The amsmath package provides a `subequations` environment to facilitate subnumbering. An example:

$$\begin{subequations}\label{eq:2}\begin{aligned} f &= g \label{eq:2A}\\ f' &= g' \label{eq:2B}\\ \mathcal{L}f &= \mathcal{L}g \label{eq:2c}\end{aligned}\end{subequations} \tag{23a, 23b, 23c}$$

is coded as:

```
\begin{subequations}\label{eq:2}
\begin{align}
f&=g \label{eq:2A}\\
f' &=g' \label{eq:2B}\\
\mathcal{L}f &= \mathcal{L}g \label{eq:2c}
\end{align}
\end{subequations}
```

E. Matrices

There are several useful matrix environments that can save you some keystrokes. See the example coding below and the output.

A simple matrix:

$$\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \tag{24}$$

is coded as:

```
\begin{equation}
\begin{matrix}  0 &  1 \\
1 &  0 \end{matrix}
\end{equation}
```

A matrix with parenthesis

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{25}$$

is coded as:

```
\begin{equation}
\begin{pmatrix} 0 & -i \\
 i &  0 \end{pmatrix}
\end{equation}
```

A matrix with square brackets

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{26}$$

is coded as:

```
\begin{equation}
\begin{bmatrix} 0 & -1 \\
1 &  0 \end{bmatrix}
\end{equation}
```

A matrix with curly braces

$$\begin{Bmatrix} 1 & 0 \\ 0 & -1 \end{Bmatrix} \tag{27}$$

is coded as:

```
\begin{equation}
\begin{Bmatrix} 1 &  0 \\
0 & -1 \end{Bmatrix}
\end{equation}
```

A matrix with single verticals

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \tag{28}$$

is coded as:

```
\begin{equation}
\begin{vmatrix} a &  b \\
c &  d \end{vmatrix}
\end{equation}
```

A matrix with double verticals

$$\begin{Vmatrix} i & 0 \\ 0 & -i \end{Vmatrix} \tag{29}$$

is coded as:

```
\begin{equation}
\begin{Vmatrix} i &  0 \\
0 & -i \end{Vmatrix}
\end{equation}
```

F. Arrays

The `array` environment allows you some options for matrix-like equations. You will have to manually key the fences, but you'll have options for alignment of the columns and for setting horizontal and vertical rules. The argument to `array` controls alignment and placement of vertical rules.

A simple array

$$\left(\begin{array}{cccc} a+b+c & uv & x-y & 27 \\ a+b & u+v & z & 134 \end{array} \right) \tag{30}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{cccc}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

A slight variation on this to better align the numbers in the last column

$$\left(\begin{array}{cccr} a+b+c & uv & x-y & 27 \\ a+b & u+v & z & 134 \end{array} \right) \tag{31}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{cccr}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

An array with vertical and horizontal rules

$$\left(\begin{array}{c|c|c|r} a+b+c & uv & x-y & 27 \\ \hline a+b & u+v & z & 134 \end{array} \right) \tag{32}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{c|c|c|r}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

Note the argument now has the pipe "|" included to indicate the placement of the vertical rules.

G. Cases Structures

Many times we find cases coded using the wrong environment, i.e., `array`. Using the `cases` environment will save keystrokes (from not having to type the `\left\lbrace`) and automatically provide the correct column alignment.

$$z_m(t) = \begin{cases} 1, & \text{if } \beta_m(t) \\ 0, & \text{otherwise.} \end{cases}$$

is coded as follows:

```
\begin{equation*}
{z_m(t)} =
\begin{cases}
1,&{\text{if}}\ {\beta }_m(t),\\
{0,}&{\text{otherwise.}}
\end{cases}
\end{equation*}
```

Note that the "&" is used to mark the tabular alignment. This is important to get proper column alignment. Do not use `\quad` or other fixed spaces to try and align the columns. Also, note the use of the `\text` macro for text elements such as "if" and "otherwise".

H. Function Formatting in Equations

In many cases there is an easy way to properly format most common functions. Use of the `\` in front of the function name will in most cases, provide the correct formatting. When this does not work, the following example provides a solution using the `\text` macro.

$$d_R^{KM} = \arg\min_{d_i^{KM}} \{d_1^{KM}, \ldots, d_6^{KM}\}.$$

is coded as follows:

```
\begin{equation*}
 d_{R}^{KM} = \underset {d_{l}^{KM}}
 {\text{arg min}} \{ d_{1}^{KM},
 \ldots,d_{6}^{KM}\}.
\end{equation*}
```

I. Text Acronyms inside equations

This example shows where the acronym "MSE" is coded using `\text{}` to match how it appears in the text.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

```
\begin{equation*}
 \text{MSE} = \frac {1}{n}\sum _{i=1}^{n}
(Y_{i} - \hat {Y_{i}})^{2}
\end{equation*}
```

J. Obsolete Coding

Avoid the use of outdated environments, such as `eqnarray` and
(
) math delimiters, for display equations. The
(
) display math delimiters are left over from PlainTeX and should not be used in LaTeX, ever. Poor vertical spacing will result.

K. Use Appropriate Delimiters for Display Equations

Some improper mathematical coding advice has been given in various YouTube™ videos on how to write scholarly articles, so please follow these good examples:

For **single-line unnumbered display equations**, please use the following delimiters:

```
\[ . . . \] or
\begin{equation*} . . . \end{equation*}
```

Note that the * in the environment name turns off equation numbering.

For **multiline unnumbered display equations** that have alignment requirements, please use the following delimiters:

```
\begin{align*} . . . \end{align*}
```

For **single-line numbered display equations**, please use the following delimiters:

```
\begin{equation} . . . \end{equation}
```

For **multiline numbered display equations**, please use the following delimiters:

```
\begin{align} . . . \end{align}
```

XII. LaTeX Package Suggestions

Immediately after your documenttype declaration at the top of your LaTeX file is the place where you should declare any packages that are being used. The following packages were used in the production of this document.

```
\usepackage{amsmath,amsfonts}
\usepackage{algorithmic}
\usepackage{array}
\usepackage[caption=false,font=normalsize,
   labelfont=sf,textfont=sf]{subfig}
\u00sepackage{textcomp}
\usepackage{stfloats}
\usepackage{url}
\usepackage{verbatim}
\usepackage{graphicx}
\usepackage{balance}
```

XIII. Additional Advice

Please use "soft" (e.g., `\eqref{Eq}`) or `(\ref{Eq})` cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BIBTEX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BIBTEX to produce a bibliography you must send the .bib files.

LATEX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LATEX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Please do not use `\nonumber` or `\notag` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

XIV. A Final Checklist

1) Make sure that your equations are numbered sequentially and there are no equation numbers missing or duplicated. Avoid hyphens and periods in your equation numbering. Stay with IEEE style, i.e., (1), (2), (3) or for subequations (1a), (1b). For equations in the appendix (A1), (A2), etc..
2) Are your equations properly formatted? Text, functions, alignment points in cases and arrays, etc.
3) Make sure all graphics are included.
4) Make sure your references are included either in your main LaTeX file or a separate .bib file if calling the external file.

References

[1] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008, ISSN: 1476-4687. DOI: 10.1038/nature06932. [Online]. Available: http://dx.doi.org/10.1038/nature06932.

[2] T. W. Molter and M. A. Nugent, "The generalized metastable switch memristor model," in *CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications*, 2016, pp. 1–2.

[3] Y. Lee, K. Kim, and J. Lee, "A compact memristor model based on physics-informed neural networks," *Micromachines*, vol. 15, no. 2, 2024, ISSN: 2072-666X. DOI: 10.3390/mi15020253. [Online]. Available: https://www.mdpi.com/2072-666X/15/2/253.

[4] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," 2018.

[5] R. T. Q. Chen, B. Amos, and M. Nickel, "Learning neural event functions for ordinary differential equations," *International Conference on Learning Representations*, 2021.

[6] B. Garda and K. Bednarz, "Comprehensive study of sdc memristors for resistive ram applications," *Energies*, vol. 17, no. 2, p. 467, Jan. 2024, ISSN: 1996-1073. DOI: 10.3390/en17020467. [Online]. Available: http://dx.doi.org/10.3390/en17020467.

[7] K. A. Campbell, "Self-directed channel memristor for high temperature operation," *Microelectronics Journal*, vol. 59, pp. 10–14, Jan. 2017, ISSN: 1879-2391. DOI: 10.1016/j.mejo.2016.11.006. [Online]. Available: http://dx.doi.org/10.1016/j.mejo.2016.11.006.

[8] L. Chua and S. M. Kang, "Memristive devices and systems," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976, ISSN: 0018-9219. DOI: 10.1109/proc.1976.10092. [Online]. Available: http://dx.doi.org/10.1109/PROC.1976.10092.

[9] A. Paszke *et al.*, *Pytorch: An imperative style, high-performance deep learning library*, 2019. arXiv: 1912.01703 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1912.01703.

[10] R. T. Q. Chen, *Torchdiffeq*, 2018. [Online]. Available: https://github.com/rtqichen/torchdiffeq.

[11] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, ACM, Jul. 2019, pp. 2623–2631. DOI: 10.1145/3292500.3330701. [Online]. Available: http://dx.doi.org/10.1145/3292500.3330701.

[12] L. Minati, L. Gambuzza, W. Thio, J. Sprott, and M. Frasca, "A chaotic circuit based on a physical memristor," *Chaos, Solitons & Fractals*, vol. 138, p. 109 990, Sep. 2020, ISSN: 0960-0779. DOI: 10.1016/j.chaos.2020.109990. [Online]. Available: http://dx.doi.org/10.1016/j.chaos.2020.109990.

[13] V. Ostrovskii, P. Fedoseev, Y. Bobrova, and D. Butusov, "Structural and parametric identification of knowm memristors," *Nanomaterials*, vol. 12, no. 1, p. 63, Dec. 2021, ISSN: 2079-4991. DOI: 10.3390/nano12010063. [Online]. Available: http://dx.doi.org/10.3390/nano12010063.

[14] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 7482–7491. DOI: 10.1109/cvpr.2018.00781. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2018.00781.

[15] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Oct. 2018, pp. 794–803. [Online]. Available: https://proceedings.mlr.press/v80/chen18a.html.

Jane Doe Biography text here without a photo.

IEEE Publications Technology Team In this paragraph you can place your educational, professional background and research and other interests.