Neural ODE Model of Power Electronic Converters with Accelerated Computation and High Fidelity

Hanchen Ge, Yaofeng Liang, Jinpeng Lei, Canjun Yuan, Zhicong Huang, *Senior Member, IEEE*

***Abstract*—Nowadays, conducting detailed numerical simulation of power electronic (PE) converters is time-consuming due to the iterative solution of nonlinear equations. While simplified numerical models can reduce the computational burden, they are unable to guarantee fidelity in highly nonlinear systems. To accelerate the detailed numerical simulation while maintaining high fidelity, this paper proposes a novel data-driven approach to modeling PE converters based on neural ordinary differential equations (ODE). This model eliminates the iterative solution of nonlinear equations by being data-driven and thus accelerates the simulation. In addition, to enhance the model fidelity, this approach incorporates prior knowledge about numerical ODE solvers into the model structures and distinguishes multi-scale characteristics from the dataset using a specific filtered MSE loss function. Experiment results demonstrate significant improvement of calculation speed and reduction of computational overheads compared to the detailed numerical methods. In addition, the proposed model has shown excellent modeling fidelity across various input frequencies using variable step solvers.**

***Index Terms*—Data-driven, Machine Learning, Neural ODE, Power Electronics Converter, Accelerated Computation, High Fidelity**

I. Introduction

THE speed and fidelity of modeling and simulation of Power Electronics (PE) systems have always been in hot pursuit. At present, the dramatic increase of renewable energy plants, distributed generators, VSC-HVDC, and FACTS devices in the power grid has put forward higher requirements for rapid simulation of PE systems [1]. Nevertheless, the inherent complexity and nonlinearity of PE systems require careful consideration of factors such as determining switching times, modeling switching transitions and loss mechanisms, and analyzing instantaneous voltage and current stresses. Moreover, with fast switching power semiconductors, simulation time steps of a few nanoseconds or less may be required, particularly during on/off switching events, leading to a substantial computational burden [2]. While physics-based models are the most common approach for simulating PE systems, modeling fidelity had to be compromised to reduce computational costs [3]. Data-driven models have shown promise in reducing computational costs while maintaining fidelity, but they encounter challenges ensuring high generalization performance and seamless integration with other systems. This paper aims to address these issues by proposing a new data-driven modeling approach for PE converters based on neural ordinary differential equations (ODE).

Illustrated in Fig. 1, years of research have yielded various types of physics-based models at 4 fidelity levels, revealing that model fidelity is inevitably compromised to reduce computations [4]. Detailed models such as SPICE and SABER capture all physical behaviors, including switching transients, power losses, and thermal characteristics [2], but are time-consuming due to complexity and iterative nonlinear equation solutions. Ideal models simplify by replacing nonlinear components with linear equivalents, such as the approximate discrete-time switching model [5], [6], switch-state prediction method [7] and event-driven methods [8]. Alternatively, the switching-function approach [9] replaces switching converter circuit with controlled sources and switching functions, bypassing the need for event handling. Circuit averaging and linearization further accelerate simulation by focusing on macroscopic characteristics while disregarding periodical switching behaviors [10], [11], providing analytical insights but lacking in modeling small ripples.

In contrast with the physics-based methods, data-driven methods present a promising solution that minimizes computational costs while maintaining fidelity. An intuitive explanation is that these methods can expedite simulation by replacing the computationally expensive equation solving with an instantaneous query of data [12]. However, the vital barrier for data-driven models of PE converters lies within the high standards of generalization performances and integration with numerical systems.

Three distinct types of data-driven models are discussed. System identification methods posit that the system is a linear combination of nonlinear functions, with the parameters of these functions being estimated using input-output data. Examples include the Instrumental-Variable method [13], Kalman-Filter, Hammerstein models, Recursive Least Squares, etc [14]–[16]. However, these approaches are typically restricted to low-order systems. The pure data-driven models uses common generative neural network (NN) models in common structures like fully-connected NN, sequential NN, convolutional NN (CNN), etc [17]–[21]. For example, long short-term

Manuscript received 9 February, 2024; revised 11 May, 2024; accepted 12 September, 2024. This work is supported by the National Natural Science Foundation of China (52007067), the Science and Technology Planning Project of Guangdong Province (2023A0505050124), and the Natural Science Foundation of Guangdong Province (2023A1515011623). (Corresponding author: Zhicong Huang)

Hanchen Ge, Yaofeng Liang, Jinpeng Lei, Canjun Yuan and Zhicong Huang are with the Shien-Ming Wu School of Intelligent Engineering, South China University of Technology, Guangzhou, 510006, China (e-mail: zhiconghuang@scut.edu.cn).

Digital Object Identifier TXXXxxxx.0000.0000.0000

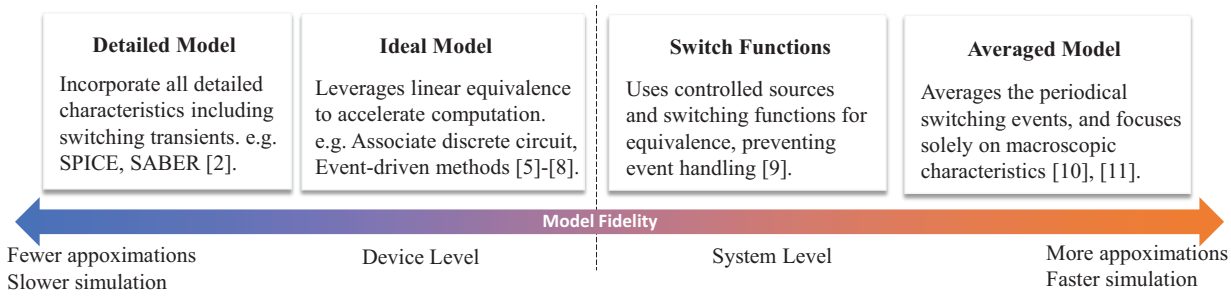


Fig. 1. An overview of numerical models for switching devices in PE converters.

memory (LSTM) are used to model the transient behaviors of the Buck converters with fixed step sizes [22]; CNN with unsupervised techniques are used to model the microscopic signals in the Buck converters [23]; attention-based NN structures are used to model the DC-DC converters [24]. However, without enough prior knowledge of the physical system, pure data-driven models often suffers from a lack of generalization. Especially for over-parameterized deep learning models, a large amount of data is required to reinforce these biases and generate predictions that comply with the basic laws of the circuit [25]. Moreover, conventional NN structures are difficult to integrate with other models into a larger system, which limits their application in controller or topology design. Physics-informed machine learning (PiML) overcome these challenges by integrating physical laws into the NN structures, substantially improving model generalization and reducing the data requirements. PiML has demonstrated impressive results in modeling a variety of systems, including electromagnetism, hydromechanics, and thermodynamics [12], [26]–[28].

In this paper, we propose a data-driven model based on neural ordinary differential equations (ODE) that boosts the detailed simulation speed of PE converters while maintaining fidelity. Neural ODE is a physics-informed NN model that effectively integrates into the numerical ODE solvers [29]–[31]. The structure of Neural ODE is illustrated in Fig. 2, along with a comparison with the detailed numerical model. In the detailed numerical model, the nonlinear system ODEs are iteratively solved in an implicit and discretized integrated form over time. In contrast, the neural ODE model, though sharing a similar time-stepped format, replaces the kernel function of the numerical integration with an explicit NN. Such structure also makes it easily integrated with other numerical models to form a closed-loop system.

First, the neural ODE shows excellent potential in accelerating simulations. On the one hand, in each simulation step of the neural ODE model, the computationally intensive solution of nonlinear equations is replaced by the forward propagation of an NN. As a result, the computational cost does not increase with the system scale; it is solely determined by the network structure, including fidelity requirements and the number of input-output (IO) channels. On the other hand, the neural ODE model directly captures the overall system characteristics without delving into microscopic temporal behaviors. This reduction in detail allows for a decrease in the total number of simulation steps by eliminating the necessity to adhere to the stability of microscopic processes, such as the switching behaviors.

Second, the neural ODE model makes high-fidelity predictions that align closely with the detailed numerical model in addition to the speed advancements. This is achieved by incorporating prior knowledge of numerical ODE solvers into the NN structures. Also, a specific filtered Mean Squared Error (MSE) Loss function is proposed to distinguish multi-scale characteristics from the dataset and improve model fidelity.

Our contributions are summarized as follows:

1) A neural ODE model enhanced with external inputs and temporal encoding has been proposed, together with a novel filtered MSE Loss to accelerate the detailed simulation and improve the model fidelity.
2) The computational cost and speed of the proposed Neural ODE model are analyzed and compared with the detailed numerical models, revealing a significant speed improvement and reduction of computational overheads.
3) The fidelity of the proposed Neural ODE model has been validated on the buck, boost and buck-boost converter, revealing an excellent consistency with the detailed numerical models.

The remainder of this paper is organized as follows. Section II presents the proposed Neural ODE model and the training process. Section III compares the computational cost and speed of Neural ODE with the detailed numerical models. Section IV validates the model fidelity of the Neural ODE model. Finally, Section V concludes the paper.

II. Neural ODE Model for PE Converters

In this section, the ODE of PE converters and the numerical ODE solvers are first introduced as they form the basis of the proposed neural ODE model. Secondly, the neural ODE models are presented, followed by the introduction of Filtered Mean Squared Error (FMSE) Loss and an its back propagation process. Thirdly, the neural ODE model of Buck and Boost converters is trained as an example. Finally, the closed-loop deployment of the neural ODE model under Simulink platform is discussed.

A. The ODE of PE Converters

The PE converters are intrinsically nonlinear time-variant systems with multiple input and output signals, which can be modeled using differential algebraic equations (DAE) that are

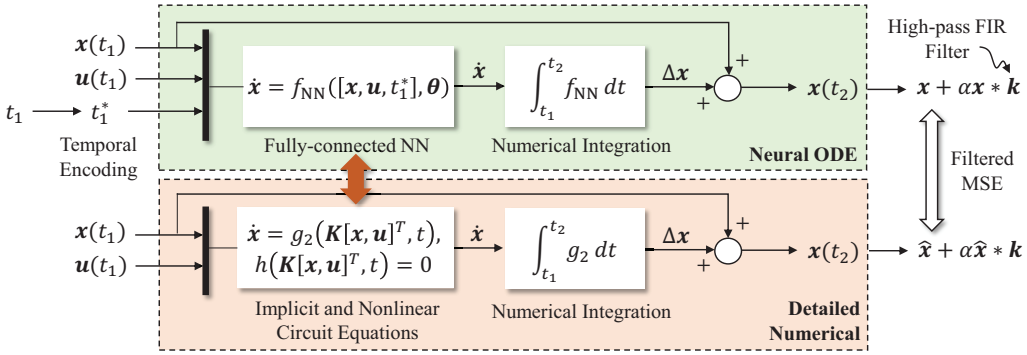


Fig. 2. The proposed Neural ODE model with inputs and temporal encoding, together with the detailed numerical model as a comparison.

composed of Kirchhoff's current law (KCL), Kirchhoff's voltage law (KVL), and the branch equations, given as follows:

$$\begin{bmatrix} 0 & 0 & \boldsymbol{A} \\ -\boldsymbol{A}^T & \boldsymbol{I} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{i} \\ \boldsymbol{v}_n \end{bmatrix} = 0, \qquad \text{KCL\&KVL}, \tag{1}$$

$$g_1([\boldsymbol{v}, \boldsymbol{i}, \boldsymbol{v}_n]^T, t) = 0, \qquad \text{static branches}, \tag{2}$$

$$g_2([\boldsymbol{v}, \boldsymbol{i}, \boldsymbol{v}_n]^T, t) = \frac{d\boldsymbol{x}}{dt}, \qquad \text{dynamic branches}, \tag{3}$$

$$[\boldsymbol{v}, \boldsymbol{i}, \boldsymbol{v}_n]^T = \boldsymbol{K}[\boldsymbol{x}, \boldsymbol{u}]^T, \qquad \text{states and inputs}. \tag{4}$$

where $\boldsymbol{i}$ and $\boldsymbol{v}$ are the branch currents and voltages, $\boldsymbol{v}_n$ is the nodal voltages, $\boldsymbol{A}$ is the incidence matrix that reflects circuit topology. $\boldsymbol{x}$ and $\boldsymbol{u}$ are the state variables and inputs of the system. $\boldsymbol{K}$ is a nonsquare matrix that projects the states and inputs into branch currents and voltages. g_1 are the branch equations of passive components, and g_2 are the branch equations of dynamic components. Time t is considered in these branch equations because the system is time-variant.

By inserting (4) into (1)-(3), the branch currents and voltages can be eliminated, resulting in a semi-explicit DAE:

$$\frac{d\boldsymbol{x}}{dt} = g_2(\boldsymbol{K}[\boldsymbol{x}, \boldsymbol{u}]^T, t), \tag{5}$$

$$h(\boldsymbol{K}[\boldsymbol{x}, \boldsymbol{u}]^T, t) = \begin{bmatrix} g_1(\boldsymbol{K}[\boldsymbol{x}, \boldsymbol{u}]^T, t) \\ \begin{bmatrix} 0 & 0 & \boldsymbol{A} \\ -\boldsymbol{A}^T & \boldsymbol{I} & 0 \end{bmatrix} \boldsymbol{K}[\boldsymbol{x}, \boldsymbol{u}]^T \end{bmatrix} = 0. \tag{6}$$

In detailed models, the intricate and nonlinear equation (6) represents an implicit mapping between $\boldsymbol{x}$ and $\boldsymbol{u}$, which cannot be directly incorporated into (5). Therefore, when solving such a system using a DAE solver, this implicit nonlinear equation must be solved iteratively at each time step before numerical integration, which has been the most time-intensive aspect of the simulation.

Given the time-intensive nature of solving nonlinear equations in detailed numerical models, one might consider utilizing neural networks to explicitly represent the mapping between $\boldsymbol{x}$ and $\boldsymbol{u}$. This approach would eliminate the need for iterative solution of (6), transforming the DAE system (5) into a straightforward explicit ODE:

$$\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t). \tag{7}$$

This is exactly the idea of the Neural ODE model, which is introduced in the next section.

B. Neural ODE Model Structures

The original Neural ODE model proposed by Chen et al [30] is a type of NN approximation of ODEs without inputs, where the kernel function f in the numerical integration is replaced by a fully-connected NN $\boldsymbol{f}_{\text{NN}}$, given as:

$$\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}(t)) \rightarrow \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}(t), \boldsymbol{\theta}). \tag{8}$$

where $\boldsymbol{\theta}$ is the parameters of the NN.

However, as was mentioned in (7), the ODEs of PE systems are inhomogenous with input terms $\boldsymbol{u}$, and both time-invariant and time-variant system should be considered. Therefore, the Neural ODE model is enhanced with external inputs and temporal encoding technique, as is shown in Fig. 2.

$$\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \rightarrow \boldsymbol{f}_{\text{NN}}([\boldsymbol{x}(t), \boldsymbol{u}(t), t^*], \boldsymbol{\theta}), \tag{9}$$

where t^* is the temporal encoding term, which introduces the observational bias of time into the NN.

The concept of temporal encoding technique draws inspiration from the positional encoding technique introduced in Transformer networks [32]. Since time t spans a continuous range with multiple scales, it needs to be transformed into a finite and periodic representation t^* before being inputted into the NN. Various encoding methods can be employed, such as sinusoidal encoding, one-hot encoding, etc. In applications like DC-DC converters, the switching behavior plays a crucial role in determining the system dynamics at microscopic time scales. Therefore, the temporal encoding term can be defined as the PWM signal of the switching devices, represented as:

$$t^* = \begin{cases} 1, t \bmod T > d(t) \\ 0, t \bmod T \leq d(t) \end{cases}, \tag{10}$$

where T is the switching period.

To solve these ODE with NN (9) and (10), numerical ODE solvers are used, which approach it as an initial value problem. The ODEs are solved iteratively in the discretized integrated form as follows:

$$\begin{aligned} \boldsymbol{x}(t_2) &= \boldsymbol{x}(t_1) + \int_{t_1}^{t_2} \boldsymbol{f}_{\text{NN}}([\boldsymbol{x}(t), \boldsymbol{u}(t)], \boldsymbol{\theta}) dt, \\ &= \boldsymbol{x}(t_1) + \text{int}(\boldsymbol{f}_{\text{NN}}, \boldsymbol{x}, \boldsymbol{u}, t_1, t_2), \end{aligned} \tag{11}$$

where int is the numerical integration that depends on the ODE solver. Taking the variable step Domand-Prince method as an

Algorithm 1 Dormand-Prince solver: $\text{int}(f, \boldsymbol{x}(t_0), \boldsymbol{u}, t_0, t_1)$

Input: The kernel function f, input signal $\boldsymbol{u}$, initial states $\boldsymbol{x}(t_0)$, the start and stop time t_0, t_1, initial step size h_0.
Output: $\int_{t_0}^{t_1} f(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau))d\tau$.

$t \leftarrow t_0$, $h \leftarrow h_0$, $\boldsymbol{x} \leftarrow \boldsymbol{x}(t_0)$.
while $t < t_1$ **do**
  Compute $x^{(h)}(t+h)$ using (12).
  Compute two steps into $x^{(h/2)}(t+h)$ using (12).
  if $\frac{1}{15}|\boldsymbol{x}^{(h/2)}(t+h) - \boldsymbol{x}^{(h)}(t+h)| \leq \varepsilon$ **then**
    $t \leftarrow t+h$, $\boldsymbol{x} \leftarrow \boldsymbol{x}^{(h)}(t+h)$, $h \leftarrow 2h$.
  else
    $h \leftarrow h/2$.
  end if
end while
return $\boldsymbol{x}(t_1)$.

example, this process is depicted in **Algorithm 1**, and the step can be expressed as follows:

$$\begin{aligned}
\text{int}(f, \boldsymbol{x}(t_0), \boldsymbol{u}, t_0, t_0+h) &= \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\
k_1 &= f(\boldsymbol{x}(t_0), \boldsymbol{u}(t_0)), \\
k_2 &= f(\boldsymbol{x}(t_0) + \frac{h}{2}k_1, \boldsymbol{u}(t_0 + \frac{h}{2})), \\
k_3 &= f(\boldsymbol{x}(t_0) + \frac{h}{2}k_2, \boldsymbol{u}(t_0 + \frac{h}{2})), \\
k_4 &= f(\boldsymbol{x}(t_0) + hk_3, \boldsymbol{u}(t_0 + h)).
\end{aligned} \tag{12}$$

In variable step solvers, an error estimation method is used to adaptively modify the step size of the numerical solver to speed up the simulation. This is achieved by a comparison of results obtained with two different step sizes. If the error falls within the tolerance threshold ε, the step size is doubled. Conversely, if the error exceeds this threshold, the step size is halved. This process is described as follows [8]:

$$\frac{1}{15}|\boldsymbol{x}^{(h/2)}(t+h) - \boldsymbol{x}^{(h)}(t+h)| \leq \varepsilon, \tag{13}$$

where $\boldsymbol{x}^{(h/2)}$ is computed under a step size of $h/2$.

C. Filtered MSE Loss Functions

In many PE systems, the output signals consist of both low-frequency large signals and high-frequency ripples. This multi-fidelity data across different amplitude and frequency scales has proved to be difficult to model using the common MSE loss function. In this section, a novel filtered MSE loss function $\mathscr{L}_f$ is proposed to separate the low-frequency and high-frequency components of the output signals and to improve the fidelity. The loss function is defined as follows:

$$\mathscr{L}_f(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \alpha \mathscr{L}_{\text{mse}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \mathscr{L}_{\text{mse}}(\boldsymbol{x} * \boldsymbol{k}, \hat{\boldsymbol{x}} * \boldsymbol{k}), \tag{14}$$

where the first and second term depicts the loss of large signals and small ripples respectively, and α depicts the weight hyperparameter. $\mathscr{L}_{\text{mse}}$ is the MSE loss function, described as follows:

$$\mathscr{L}_{\text{mse}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}(t_i) - \hat{\boldsymbol{x}}(t_i)\|^2. \tag{15}$$

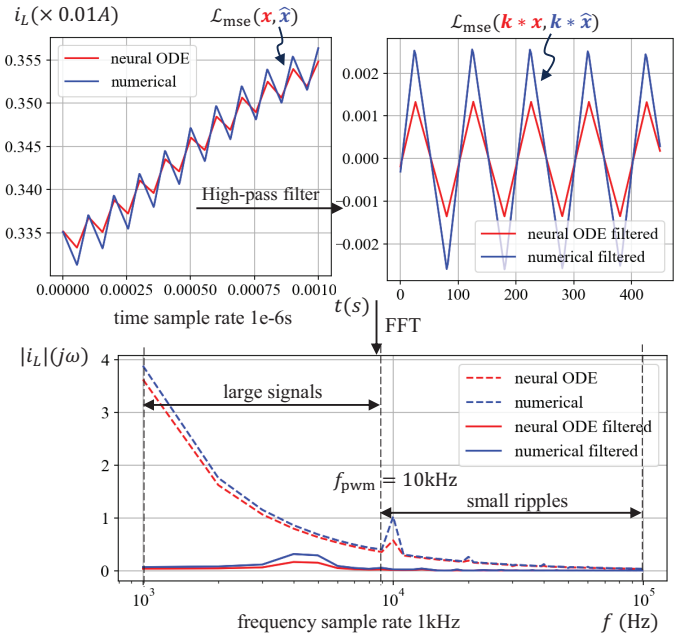


Fig. 3. An example of the Filtered MSE loss function.

In (14), a high-pass finite impulse response (FIR) filter is leveraged to separate the large signals and small ripples, as is shown in Fig. 3. $\boldsymbol{k}$ is the high-pass FIR filter kernel, and $*$ is signal convolution. The high-pass FIR filter is designed using the remez (equiripple) method [33], given as follows:

$$\boldsymbol{k} = \text{remez}(N, \boldsymbol{f}, \boldsymbol{a}, \boldsymbol{f}_s), \tag{16}$$

where N is the filter order, $\boldsymbol{f}$ is the frequency band, $\boldsymbol{a}$ is the amplitude band, and $\boldsymbol{f}_s$ is the sampling frequency. e.g. In the buck converter example, given the switching frequency as f_s, the frequency band is set as $[0, 0.5f_s]$, the amplitude band is set as $[0, 0.5]$, and the filter order N is defined using Parks-McClellan optimal FIR filter order estimation [34].

D. Backward propagation of Neural ODE Model

The forward and backward propagation procedure of the neural ODE is demonstrated in Fig. 2. The forward propagation is performed by numerical integration specified in (11), while the backpropagation through time (BPTT) is performed by adjoint sensitivity method [30], depicted as follows:

$$\begin{aligned}
\boldsymbol{\theta} &= \arg\min_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{x}(t_i), \hat{\boldsymbol{x}}(t_i)), \\
\text{s.t. } \boldsymbol{x}' &= f_{\text{NN}}(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i), t_i, \boldsymbol{\theta}), \boldsymbol{x}(0) = \hat{\boldsymbol{x}}(0).
\end{aligned} \tag{17}$$

To perform BPTT for neural ODE, the derivative of loss $\mathscr{L}$ w.r.t. $[\boldsymbol{x}, \boldsymbol{\theta}]$ has to be calculated, which is defined as the augmented adjoint state $[\boldsymbol{a}, \boldsymbol{a}_\theta]$:

$$[\boldsymbol{a}, \boldsymbol{a}_\theta] = [\frac{d\mathscr{L}}{d\boldsymbol{x}(t)}, \frac{d\mathscr{L}}{d\boldsymbol{\theta}}]. \tag{18}$$

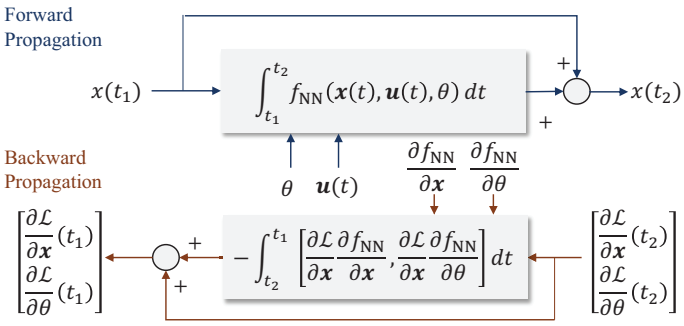


Fig. 4. The forward and backward propagation of the Neural ODE.

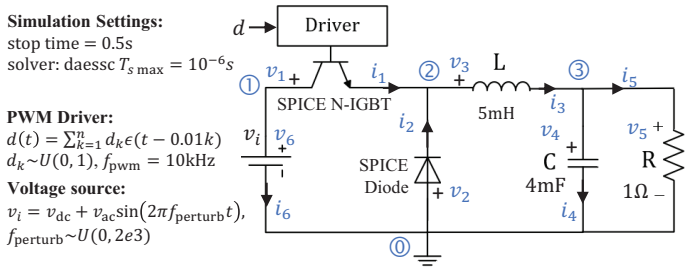


Fig. 5. The detailed SPICE model of the Buck converter based on the Simulink Simscape electrical library.

Algorithm 2 Training process of the neural ODE model

Input: The length of samples l_{sample}, the number of epochs N_{epoch}, batch size N_{batch}, the training set, including the inputs $\boldsymbol{u}(t)$ and the outputs $\boldsymbol{x}(t)$

Load dataset and define $\boldsymbol{u}(t)$ as global;
Initialize the fully connected NN with parameters $\boldsymbol{\theta}$;
for i in N_{epoch} **do**
  Get a batch of random start time t_0 and sample rate h;
  Get a batch of training samples with l_{sample}, h, and t_0;
  Propagate forward based on (11):

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \text{int}(\boldsymbol{f}_{\text{NN}}, \boldsymbol{x}(t_0), \boldsymbol{u}, t_0, t)$$

  Compute the loss function $\mathscr{L}$;
  Propagate backward based on (20):

$$\boldsymbol{a}_\theta(t_1) = \boldsymbol{a}_\theta(t_0) - \text{int}(\boldsymbol{a}^T \frac{\partial \boldsymbol{f}_{\text{NN}}}{\partial \boldsymbol{\theta}}, \boldsymbol{a}(t_1), \frac{\partial \boldsymbol{f}_{\text{NN}}}{\partial \boldsymbol{\theta}}, t, t_0).$$

  Update the parameters $\boldsymbol{\theta}$.
end for

It can be proved by the definition of the derivative that the adjoint state $\boldsymbol{a}$ satisfies the following equation:

$$\begin{aligned} \boldsymbol{a}' &= -\boldsymbol{a}^T \frac{\partial \boldsymbol{f}_{\text{NN}}}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta}), \\ \boldsymbol{a}'_\theta &= -\boldsymbol{a}^T \frac{\partial \boldsymbol{f}_{\text{NN}}}{\partial \boldsymbol{\theta}}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta}). \end{aligned} \tag{19}$$

Therefore, the gradient of the loss function $\mathscr{L}$ w.r.t. the parameters $\boldsymbol{\theta}$ can be calculated using numerical solvers, given as:

$$\begin{aligned} \boldsymbol{a}(t_1) &= \frac{\partial \mathscr{L}}{\partial \boldsymbol{x}} = \boldsymbol{a}(t_0) - \int_{t_1}^{t_0} \boldsymbol{a}^T \frac{\partial \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta})}{\partial \boldsymbol{x}} d\tau, \\ \boldsymbol{a}_\theta(t_1) &= \frac{\partial \mathscr{L}}{\partial \boldsymbol{\theta}} = \boldsymbol{a}_\theta(t_0) - \int_{t_1}^{t_0} \boldsymbol{a}^T \frac{\partial \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} d\tau, \end{aligned} \tag{20}$$

where $\frac{\partial \boldsymbol{f}_{\text{NN}}}{\partial \boldsymbol{\theta}}$ and $\frac{\partial \boldsymbol{f}_{\text{NN}}}{\partial \boldsymbol{x}}$ can be efficiently evaluated by automatic differentiation. To summarize, the overall forward and backward propagation of Neural ODE is illustrated in Fig. 4 and **Algorithm 2**.

E. Training of Neural ODE Model for Buck and Boost Converters

In this section, the Neural ODE model of two open-loop converters including Buck and Boost are trained and validated. Taking the Buck converter shown in Fig. 5 as an example, the system DAEs are given as follows:

$$\begin{aligned} &\text{KCL}, \quad f_1(i_1, v_1) = 0, \quad Li_3' = v_3, \quad v_5 = Ri_5, \\ &\text{KVL}, \quad f_2(i_2, v_2) = 0, \quad Cv_4' = i_4, \quad v_6 = v_i. \end{aligned} \tag{21}$$

in which f_1 and f_2 are the branch equations of the IGBT and diode, and the system inputs and states are specified as follows:

$$\begin{aligned} \boldsymbol{u}(t) &= [d(t), v_i(t)]^T, \\ \boldsymbol{x}(t) &= [i_3(t), v_4(t)]^T, \end{aligned} \tag{22}$$

where d is the duty cycle of IGBT driver signals. The same inputs and states are specified in the Boost converter, and the dataset generation and training settings are comprehensively summarized as follows:

1) **Data Generation:** Two open-loop converters are build to generate the dataset, including the Buck and Boost converters. The input voltage comprises an 80V DC component and a 20V perturbation signal, with the frequency of the perturbation signal f_{perturb} randomly determined according to a uniform distribution. The duty cycle is set to a random cyclic varied signal that also follows the uniform distribution. The output signals v_o and i_L are obtained by the detailed SPICE models using the Simulink Simscape Electrical library, with the data saved in the mat 7.3 format. The parameter settings and circuit diagram of the Buck converter are shown in Fig. 5. These parameter settings are also applied to the Boost converter.
The detailed parameters of the SPICE IGBT and Diode model are set as default, given as Table I. The switches are modeled using Full I-V and capacitance characteristics, which is suitable for simulating detailed switching characteristics and predicting component losses.
For each converter, 120 simulations are conducted, each lasting 0.5 seconds, using the variable step solver daessc. The result is then downsampled into a step size of 1e-6s. As was illustrated in (22), both input and output signals consisted of two channels, resulting in a dataset size of $120 \times 5e5 \times 4$. Alternatively, a single simulation lasting 60 seconds could be performed to yield the same dataset size. However, dividing the simulations into multiple runs facilitates parallelization, thereby enhancing the speed of data generation.

TABLE I
DETAILED PARAMETER SETTINGS FOR THE CONVERTER MODELS, DATA GENERATION, TRAINING AND EVALUATION.

Detailed Model Parameter Settings (Buck, Boost)			
Zero V_g collector current I_{ces}, IGBT	2mA	Diode Satuation current, I_s	1e-12A
V_c at which I_{ces} is defined	600V	Diode Emission coeff, N	1
$V_{\mathrm{ge(th)}}$, IGBT	6V	T	25°C
V_{ces}, IGBT	2.6V	Diode BV	200V
I_c at which V_{ces} is defined	400A	Parallel resistance of L	1e-9Ω
V_{ge} at which V_{ces} is defined, IGBT	10V	Series resistance of C	1e-6Ω
Input capacitance, IGBT	26.4nF	L	5mH
Reverse transfer capacitance of IGBT	2.7nF	f_{pwm}	10kHz
C	4mF	R	1Ω
Data Generation Parameter Settings			
Num of simulations	120	Stop time	0.5s
Solver	daessc	Resolution	1e6
Size	120x5e5x4	Max step size	1e-6s
range of f_{perturb}	(0, 2e3)	$v_{\mathrm{dc}}, v_{\mathrm{ac}}$	80V, 20V
range of D	(0, 1)	$v_C(0), i_L(0)$	0V, 0A
Training Parameter Settings			
Solver	RK4	Step size	1e-5s
Batch size	64	Sequence length	100
Sample interval	10	Optimizer	Adam
Learning rate	0.01	Loss function	FMSE, 0.1
Test batch size	128	Test length	1000
Evaluation Parameter Settings			
Solver	dopri45	Max step size	1e-5s
Batch size	48	Stop time	0.04s
Num of batches	10		

2) **Model implementation:** The Torchdiffeq repository [30] based on PyTorch platform is modified to support variable input signals, temporal encoding and mini-batch gradient descent, implementing the enhanced Neural ODE models given in Fig. 2. Multiple fully-connected NN structures are evaluated and $[4, 32, 32, 2]$ is found to be the most proficient version, with leaky-ReLU activation function at the first two layers.
   The forward propagation of the Neural ODE model is implemented using multiple solver types including the Dormand-Prince solver and the Runge-Kutta 4 (RK4) solver. The backward propagation is implemented using the adjoint sensitivity method. The filtered MSE loss function is implemented using the Scipy and PyTorch library.
3) **Training settings:** The training process is performed using Python/Pytorch on a single NVIDIA RTX 3090Ti GPU, and the Adam optimizer is used with a learning rate of 0.01. The training batch size is set to 64, and the loss function is defined as the Filtered MSE between the output of the Neural ODE model and the detailed numerical model. The training parameters settings are summarized in Table I.
   During model training, batches of input-output data were randomly sampled from the dataset at a fixed interval of 10, with a sequence length of 100. This resulted in a fixed step size of 1e-5s for the Neural ODE model, which is 10 times larger than the generated data. Runge-Kutta 4 (RK4) solver is used to train the model. Due to the random nature of data extraction and the large dataset size (potentailly 6e7 different samples), it is unlikely to repeatedly obtain the same data samples.
   Every 20 epochs of training, a batch of input-output data was randomly sampled from the dataset for testing. The testing batch maintained a sampling interval of 10, a batch size of 128, and a sequence length of 1000, which is ten times longer than the training length. Training typically continued until the Filtered MSE of multiple epochs of testing fell below 1e-3, indicating that the model's accuracy met the required threshold. This process usually necessitates training for approximately 20,000 epochs.

F. Deployment of Neural ODE model using Simulink

Since the forward propagation of Neural ODE model is based on numerical solvers, it can be seamlessly integrated into numerical simulation platforms such as Simulink, enabling closed-loop deployment and further evaluation of speed and fidelity.

In this section, the Neural ODE model in the PyTorch format is exported into Simulink using the MATLAB function block, as is illustrated in Fig. 6. In contrast with the trianing process which uses a fixed step RK4 solver, the solver setting is set to variable Dormand-Prince 45 solver with a maximum step size 1e-5s, which illustrates the model's generalization ability across different kind of solvers. As an equivalence to the detailed numerical model, the Neural ODE model is connected with numerical PI controllers and PWM generators to form a closed-loop system in the voltage-controlled mode.

In the closed-loop system of Fig. 6, there are two memory blocks that store states variables: the PI controller block and the numerical integration block. In each time step of the simulation, three steps are performed as follows:

1) The outputs of the memory blocks are computed according to the stored states variables.
2) The other blocks including the fully-connected NN are called based on the memory blocks output, therefore all the signals at current time step are known.
3) The state equations of these memory blocks are computed based on the current signals, and the simulation is advanced to the next time step.

This process is repeated until the simulation is completed.

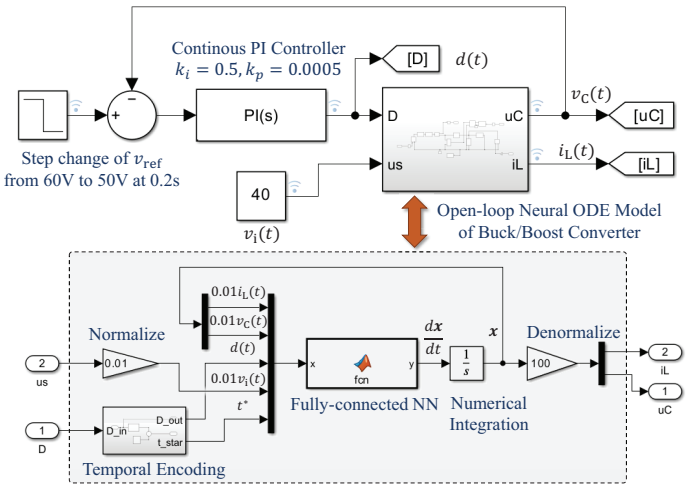


Fig. 6. The closed-loop deployment of the Neural ODE model in Simulink.

III. Comparison of Computation Speed between the Proposed Neural ODE Model and Detailed Numerical Model

To gain a more detailed understanding, a quantitative estimation and comparison of the computational costs is performed over the proposed neural ODE model and the detailed numerical model. Notably, the averaged and ideal numerical models are not compared because they are not always available or precise for complex nonlinear systems. Various settings are evaluated, including the step size, network structure, and the type of circuits. The computational cost is quantified in terms of the number of floating-point operations (FLOPs) required to simulate the PE converters within a given time unit. Table II defines the notations used and their values in three different settings which are listed as follows:

1) Setting (a): The circuit is a buck converter. The network structure is set to $[4, 32, 32, 2]$.
2) Setting (b): The circuit is a sepic converter, while the step size and the network structure remain the same.
3) Setting (c): The circuit is a full-bridge DC-DC converter, while the step size and the network structure remain the same.

A. Computational Cost of the Detailed Numerical Models

According to the key steps of the detailed numerical methods summarized in **Algorithm 3**. In addition, 3 key algorithms are evaluated, including the SVD decomposition method for system simplification, the Newton iterative method for solving nonlinear equations, and the Dormand Prince method for numerical integration.

1) The SVD decomposition method is used to simplify the linear part of the circuit equation before the simulation starts. This set of equations consists of the linear branch equations, KCL, and KVL, given as follows:

$$\begin{aligned} &\boldsymbol{F}\boldsymbol{v}+\boldsymbol{H}\boldsymbol{i}=\boldsymbol{b}, \boldsymbol{A}\boldsymbol{i}=0, \boldsymbol{v}-\boldsymbol{A}^T\boldsymbol{v}_n=0 \qquad (23)\\ &\Rightarrow \boldsymbol{B}[\boldsymbol{v}_n,\boldsymbol{v},\boldsymbol{i}]^{\mathrm{T}}=[\boldsymbol{0},\boldsymbol{0},\boldsymbol{b}]^{\mathrm{T}},\\ &\text{where } \boldsymbol{B}=\begin{bmatrix}\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}\\ -\boldsymbol{A}^{\mathrm{T}} & \boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{F} & \boldsymbol{H}\end{bmatrix}. \qquad (24)\end{aligned}$$

Suppose that the number of branches b, the number of nodes n, the number of linear and nonlinear branches $n_{\mathrm{lin}}, n_{\mathrm{nlin}}$, then, the size of $\boldsymbol{A}$ is $(n-1)\times b$, the size of the underdetermined matrix $\boldsymbol{B}$ is $(n_{\mathrm{lin}}+n-1+b)\times(n-1+2b)$. The rank of $\boldsymbol{B}$ is

$$R(\boldsymbol{B})=n_{\mathrm{lin}}+n-1+b. \qquad (25)$$

TABLE II
Notations and Values in Computational Cost Analysis

Symbol	Meaning	Estimated Values		
		(a)	(b)	(c)
N	The number of time steps.	1e5	1e5	1e6
m	The average number of iterations of Newton iterative method.	10	10	10
o	The length of state vector.	2	2	2
l	The number of input channels.	2	2	2
n	The number of nodes.	3	4	8
a	The length of node voltage, branch voltage and current.	14	19	39
$b, n_{\mathrm{lin}}, n_{\mathrm{nlin}}$	The number of overall, linear, and nonlinear branches.	6,4, 2	8,6, 2	16,8 ,8

Algorithm 3 Detailed numerical simulation for PE converters

Input: The linear branch equations $\boldsymbol{F}\boldsymbol{v}+\boldsymbol{H}\boldsymbol{i}=\boldsymbol{b}$,
The nonlinear branch equations $g(\boldsymbol{v}_{\mathrm{nlin}}, \boldsymbol{i}_{\mathrm{nlin}})=0$,
The adjoint matrix $\boldsymbol{A}$, the initial state $\boldsymbol{x}_0$,
The start and stop time t_0, t_1, initial step size h_0.
Output: $\boldsymbol{x}(t_1)$.
$t \leftarrow t_0$, $h \leftarrow h_0$, $\boldsymbol{x} \leftarrow \boldsymbol{x}(t_0)$.
Simplify the linear part of the circuit equation set using the SVD decomposition:

$$\begin{aligned}&\boldsymbol{F}\boldsymbol{v}+\boldsymbol{H}\boldsymbol{i}=\boldsymbol{b}, \boldsymbol{A}\boldsymbol{i}=0, \boldsymbol{v}-\boldsymbol{A}^T\boldsymbol{v}_n=0\\ &\Rightarrow [\boldsymbol{v}_n,\boldsymbol{v}_{\mathrm{lin}},\boldsymbol{i}_{\mathrm{lin}}]^{\mathrm{T}}=\boldsymbol{V}_r\boldsymbol{\Sigma}_r^{-1}\boldsymbol{U}_r^{\mathrm{T}}\boldsymbol{c}.\end{aligned}$$

while $t < t_1$ **do**
Compute the nonlinear equation set using the Newton iterative method:

$$\begin{aligned}&g(\boldsymbol{v}_{\mathrm{nlin}},\boldsymbol{i}_{\mathrm{nlin}})=0,\\ &[\boldsymbol{v}_n,\boldsymbol{v}_{\mathrm{lin}},\boldsymbol{i}_{\mathrm{lin}}]^{\mathrm{T}}=\boldsymbol{V}_r\boldsymbol{\Sigma}_r^{-1}\boldsymbol{U}_r^{\mathrm{T}}\boldsymbol{c}.\end{aligned}$$

Compute the input signal $\boldsymbol{u}(t)$ by $(\boldsymbol{i},\boldsymbol{v})$ obtained.
$\boldsymbol{x}(t+h) \leftarrow \boldsymbol{x}(t)+\mathrm{int}(f,\boldsymbol{x}(t_0),\boldsymbol{u},t_0,t)$.
$t \leftarrow t+h$.
end while
return $\boldsymbol{x}(t_1)$.

To minimize the number of variables into $R(\boldsymbol{B})$, the SVD decomposition method is used, given as follows:

$$\begin{aligned}&\boldsymbol{B}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}},\\&\boldsymbol{U}=\begin{bmatrix}\boldsymbol{U}_r & \boldsymbol{U}_2\end{bmatrix},\boldsymbol{\Sigma}=\begin{bmatrix}\boldsymbol{\Sigma}_r & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\Sigma}_2\end{bmatrix},\boldsymbol{V}=\begin{bmatrix}\boldsymbol{V}_r\\ \boldsymbol{V}_2\end{bmatrix}.\end{aligned}\tag{26}$$

where $\boldsymbol{U}_r$ is the left singular matrix of $\boldsymbol{B}$, $\boldsymbol{\Sigma}_r$ is the diagonal matrix of the singular values of $\boldsymbol{B}$, $\boldsymbol{V}_r$ is the right singular matrix of $\boldsymbol{B}$. $\boldsymbol{U}_2,\boldsymbol{\Sigma}_2,\boldsymbol{V}_2$ are the remaining parts of $\boldsymbol{U},\boldsymbol{\Sigma},\boldsymbol{V}$. Therefore, the underdetermined equation set is simplified as follows:

$$\begin{aligned}&\boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^{\mathrm{T}}[\boldsymbol{v}_n,\boldsymbol{v}_{\mathrm{lin}},\boldsymbol{i}_{\mathrm{lin}}]^{\mathrm{T}}=\boldsymbol{c},\\&\Rightarrow[\boldsymbol{v}_n,\boldsymbol{v}_{\mathrm{lin}},\boldsymbol{i}_{\mathrm{lin}}]^{\mathrm{T}}=\boldsymbol{V}_r\boldsymbol{\Sigma}_r^{-1}\boldsymbol{U}_r^{\mathrm{T}}\boldsymbol{c}.\end{aligned}\tag{27}$$

The total number of FLOPs of the SVD decomposition method is as follows:

$$N_{\mathrm{FLOPs}}=a(a-n_{\mathrm{nlin}})^2.\tag{28}$$

2) The Newton iterative method is used to solve the nonlinear circuit equations at each timestep during simulation, given as follows:

$$G(\boldsymbol{v},\boldsymbol{i},\boldsymbol{v_n})=0\Leftrightarrow\begin{cases}g(\boldsymbol{v}_{\mathrm{nlin}},\boldsymbol{i}_{\mathrm{nlin}})=0,\\ [\boldsymbol{v}_n,\boldsymbol{v}_{\mathrm{lin}},\boldsymbol{i}_{\mathrm{lin}}]^{\mathrm{T}}-\boldsymbol{V}_r\boldsymbol{\Sigma}_r^{-1}\boldsymbol{U}_r^{\mathrm{T}}\boldsymbol{c}=0.\end{cases}$$

In each iteration of the Newton iterative method, the Jacobian matrix J is computed, given by:

$$J_{ij}=\frac{\partial G_i(\boldsymbol{x})}{\partial \boldsymbol{x}_j},\quad i,j=1,2,...,a,\tag{29}$$

$$\text{where } a=2b+n-1,\boldsymbol{x}=[\boldsymbol{v}_n,\boldsymbol{v},\boldsymbol{i}]^{\mathrm{T}}.\tag{30}$$

If forward difference is used to compute the partial difference, $G(\boldsymbol{x})$ is evaluated twice for each element of J, and according to the number of FLOPs of matrix multiplication, the evaluation of $G(\boldsymbol{x})$ costs $(n_{\mathrm{lin}}+n-1+b)\times a^2+n_{\mathrm{nlin}}$. While J contains $n_{\mathrm{nlin}}a$ elements, its computation costs $2n_{\mathrm{nlin}}(n_{\mathrm{lin}}+n-1+b)\times a^3+2an_{\mathrm{nlin}}^2$ in total. Considering that the total number of iterations m and the number of steps N depend on initial guess and tolerance, the total number of FLOPs of the Newton iterative method is as follows:

$$N_{\mathrm{FLOPs}}=2Nmn_{\mathrm{nlin}}((n_{\mathrm{lin}}+n-1+b)\times a^3+an_{\mathrm{nlin}}).\tag{31}$$

3) The Dormand Prince method is used to solve the branch ODEs, given as **Algorithm 1**. First is to compute the coefficients k_1,k_2,k_3,k_4 as is illustrated in (12), which evaluates f 4 times. The evaluate of f costs o^2+ol. Second is to estimate error, which requires 2 steps at half of the step size. Therefore, the total number of FLOPs of the Dormand Prince method is as follows:

$$N_{\mathrm{FLOPs}}=12(o^2+ol).\tag{32}$$

TABLE III
NUMBER OF FLOPS ESTIMATED FOR THE DETAILED NUMERICAL MODEL AND THE NEURAL ODE MODEL IN THE 3 SETTINGS.

Operation	Method	Estimated FLOPs		
Detailed Numerical Model		(a)	(b)	(c)
1. System Simplification	SVD Decomposition	2016	5491	37479
2. Nonlinear Equations Solving	Newton iterative Method	1.3e11	4.7e11	3.0e14
3. Numerical Integration	Dormand-Prince	9.6e6	9.6e6	9.6e6
Overall	-	1.3e11	4.7e11	3.0e14
Neural ODE Model		(a)	(b)	(c)
1. Forward propagation of fully-connected NN	Common Matrix Operations	1248	1248	1248
2. Numerical Integration	Dormand-Prince	1.5e9	1.5e9	1.5e10
Overall	-	1.5e9	1.5e9	1.5e10

B. Computational Cost of the Proposed Neural ODE Models

The computational cost of the fully-connected NN is analyzed as follows, where the output of the layer i is computed as:

$$\boldsymbol{x}^{[i]}=\boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{x}^{[i-1]}+\boldsymbol{b}),\tag{33}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight matrix and bias vector, and $\boldsymbol{\sigma}$ is the activation function. Therefore, the total cost is $\sum_{i=1}^{3}n_in_{i-1}$, where n_i is the number of neurons in the i-th layer. The total cost of the neural ODE model is as follows:

$$N_{\mathrm{FLOPs}}=\sum_{i=1}^{3}n_in_{i-1}+12(o^2+ol).\tag{34}$$

According to (28), (31), (32) and (34), the computational cost of the three settings is analyzed in Table III. In summary, even for simple topologies like the buck converter, the proposed neural ODE model outperforms the detailed numerical model in terms of computational efficiency. This advantage becomes more pronounced for complex topologies, as the computational cost does not significantly increase with the system's complexity. Moreover, under the same step sizes, the proposed neural ODE model consistently proves to be the most efficient option.

C. Results of Computation Speed

The speed test is conducted on the Simulink platform, wherein the proposed Neural ODE model is exported from PyTorch, as was described in Section II-F. The test case settings include the Buck converter and the Boost converter.

Due to the high stiffness of detailed numerical converter models, using a fixed step solver requires a considerably small step size which may slow down the simulation. Therefore, the implicit variable step solver daessc is employed for the detailed numerical models. To make the comparison fair enough, the Neural ODE model is also configured with variable step solver.

TABLE IV
SPEED COMPARISONS BETWEEN DETAILED NUMERICAL AND NEURAL ODE MODEL.

	Detailed Numerical Models	**Neural ODE Model**
Tested Case Settings	Buck converter f_{pwm} = 10kHz	Buck converter f_{pwm} = 10kHz
Implementation Methods	Simulink simscape electrical library	Simulink matlab function block
Stop Time	10s	10s
Solver	daessc (implicit rk45)	dorpri45
Max step size	1e-5s	1e-5s
Ave CPU Time	1833.60s	9.37s
Number of Steps	74,999,905	1,000,002
Ave Time Per Step	2.44e-5s	9.36e-6s

Considering that the Neural ODE model is explicit, Dormand-Prince 45 solver is used, which is equivalent to daessc during numerical integration.

Employing the same hardware and software environment, the results are summarized in Table IV. Each test case has been evaluated 100 times with the inputs as random. The proposed model exhibits a speed advantage of over 200 times compared to the detailed numerical model, featuring fewer calculated points and shorter time per step. These results highlight that the proposed model can substantially reduce computational costs while maintaining high precision.

IV. EVALUATION OF MODELING FIDELITY

The modeling fidelity of the proposed Neural ODE models are evaluated and compared with the detailed numerical model on both the open-loop and closed-loop Buck and Boost converter. This evaluation is conducted within the Simulink platform, utilizing the solver configuration identical to that described in Section III-C. Specifically, the daessc solver is employed for the detailed numerical models, while the Dormand-Prince 45 solver is utilized for the Neural ODE model. The maximum step size is defined as 1e-5s. As the simulation results are obtained at irregular time intervals, they are subsequently resampled to a fixed time step of 1e-5s using linear interpolation to calculate errors.

In open-loop systems, the input and output of this evaluation has been shown in Fig. 7, which involves randomly generate the initial value of the state variables and a pair of input signal, including voltage v_i and the duty cycle d, and comparing the output of both models v_C and i_L. In order to quantitatively evaluate the prediction error of the proposed Neural ODE models with regard to different amplitude and frequency scales, the input signals are defined as follows:

$$\begin{aligned} v_i(t) &= v_{dc} + v_{ac}\sin(2\pi f_{\text{perturb}} t), \\ d(t) &= \sum_{k=1}^{n} d_k \varepsilon(t-0.01k), \quad t \in [0, 0.04], \end{aligned} \tag{35}$$

where f_{perturb} is the perturbation frequency, $\varepsilon(t)$ is the step signal. The variable d_k, v_{dc}, v_{ac}, $v_C(0)$, $i_L(0)$ are randomly generated according to the uniform distribution, given as:

$$\begin{aligned} &d_k \sim U(0,1), && v_{dc} \pm v_{ac} \sim U(V_{\text{low}}, V_{\text{high}}), \\ &v_C(0) \sim U(V_{\text{Clow}}, V_{\text{Chigh}}), && i_L(0) \sim U(I_{\text{Llow}}, I_{\text{Lhigh}}). \end{aligned} \tag{36}$$

Similarly, in closed-loop systems, the input and output channels have been shown in Fig. 7. The input signals are defined as follows:

$$\begin{aligned} v_i(t) &= v_{dc}, \\ v_{\text{ref}}(t) &= v_k \varepsilon(t-0.25k), \quad t \in [0, 0.5], \\ v_k &\sim U(v_{dc}, 2v_{dc}). \end{aligned} \tag{37}$$

Similar to the training procedure, the evaluation runs are also organized into batches, with the batch size set to 48. In a batch of test samples, the parameter f_{perturb}, V_{low}, V_{high}, V_{Clow}, V_{Chigh}, I_{Llow}, I_{Lhigh} is predefined according to the training dataset, representing the setting of amplitude and frequencies scales.

For the Buck and Boost converter model, 10 test batches (480 runs) are conducted respectively, with f_{perturb} increases from 200 Hz to 2000 Hz. This range is chosen based on the distribution of f_{perturb} in the training dataset. The errors in each test batch are averaged, and the results are summarized in Table V. For open-loop systems, the error of each samples are computed using the MSE, filtered MSE, Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) metrics. The definition of MSE and filtered MSE has been introduced in (15) and (14). The MAE and MAPE are defined as follows:

$$\mathscr{L}_{\text{MAE}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{N}\sum_{i=1}^{N} |\boldsymbol{x}(t_i) - \hat{\boldsymbol{x}}(t_i)|, \tag{38}$$

$$\mathscr{L}_{\text{MAPE}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{N}\sum_{i=1}^{N} \frac{|\boldsymbol{x}(t_i) - \hat{\boldsymbol{x}}(t_i)|}{\boldsymbol{x}(t_i)}. \tag{39}$$

The error of closed-loop systems are determined by the error of the open-loop system, as a result, the error of the closed-loop system is not calculated. Our result in Table V indicates that the proposed Neural ODE model can achieve a high fidelity with a MAPE less than 0.1% for the Buck converter and less than 0.15% for the Boost converter. The correlation between MSE and f_{perturb} is calculated using the Pearson correlation coefficient, given as follows:

$$\rho = \frac{\sum_{i=1}^{N}(\boldsymbol{x}(t_i) - \hat{\boldsymbol{x}}(t_i))}{\sqrt{\sum_{i=1}^{N}\boldsymbol{x}(t_i)^2}\sqrt{\sum_{i=1}^{N}\hat{\boldsymbol{x}}(t_i)^2}}. \tag{40}$$

which is -0.57 for the Buck converter and -0.50 for the Boost converter. The results demonstrate that the MSE does not correlates with the f_{perturb} within the test range.

A few test samples of open-loop and closed-loop Buck and Boost has been visualized in Fig. 8, Fig. 9 and Fig. 10 respectively. Within these figures, the absolute error curves are computed to illustrate the relationship between error and signal magnitude. It was observed that the prediction error is more closely linked to the temporal difference of the output signals $d\boldsymbol{x}/dt$ rather than the amplitude itself. This observation

TABLE V
PREDICTION ERROR OF THE OPEN-LOOP NEURAL ODE MODEL W.R.T PERTURBATION FREQUENCY

Open-loop Buck Converter, Neural ODE Model

Test batch size = 48, Stop time = 0.04s, $T_{smax} = 10^{-5}$s, $[V_{low}, V_{high}] = [60,100]$, $[V_{Clow}, V_{Chigh}] = [0,120]$, $[I_{Llow}, I_{Lhigh}] = [0,120]$.

$f_{perturb}$	200Hz	400Hz	600Hz	800Hz	1000Hz	1200Hz	1400Hz	1600Hz	1800Hz	2000Hz	Average
MSE	1.27E-03	1.49E-03	6.52E-04	8.91E-04	7.62E-04	1.03E-03	8.76E-04	7.70E-04	4.58E-04	9.52E-04	9.15E-04
MAE	2.49E-02	2.97E-02	1.69E-02	1.90E-02	1.82E-02	2.20E-02	2.01E-02	1.90E-02	1.41E-02	2.04E-02	2.04E-02
MAPE	0.100%	0.110%	0.064%	0.075%	0.066%	0.082%	0.075%	0.071%	0.051%	0.077%	0.077%
Filtered MSE	2.28E-03	2.67E-03	1.17E-03	1.60E-03	1.37E-03	1.85E-03	1.57E-03	1.38E-03	8.23E-04	1.71E-03	1.64E-03

Open-loop Boost Converter, Neural ODE Model

Test batch size = 48, Stop time = 0.04s, $T_{smax} = 10^{-5}$s, $[V_{low}, V_{high}] = [10,70]$, $[V_{Clow}, V_{Chigh}] = [20,50]$, $[I_{Llow}, I_{Lhigh}] = [60,80]$.

$f_{perturb}$	200Hz	400Hz	600Hz	800Hz	1000Hz	1200Hz	1400Hz	1600Hz	1800Hz	2000Hz	Average
MSE	1.78E-03	1.79E-03	1.77E-03	1.66E-03	1.85E-03	1.66E-03	1.72E-03	1.77E-03	1.63E-03	1.69E-03	1.73E-03
MAE	4.18E-02	4.19E-02	4.17E-02	4.04E-02	4.26E-02	4.03E-02	4.12E-02	4.16E-02	3.99E-02	4.07E-02	4.12E-02
MAPE	0.132%	0.135%	0.138%	0.128%	0.142%	0.126%	0.123%	0.130%	0.126%	0.128%	0.131%
Filtered MSE	3.19E-03	3.21E-03	3.19E-03	2.98E-03	3.31E-03	2.97E-03	3.09E-03	3.17E-03	2.92E-03	3.03E-03	3.11E-03

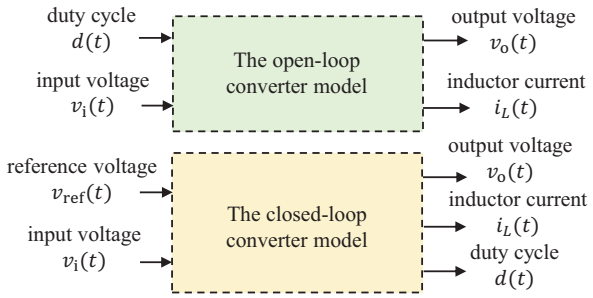


Fig. 7. The input and output signals of the open-loop and closed-loop Buck and Boost converter.

is attributed to the fact that the output of the fully-connected NN within the Neural ODE model represents the temporal difference, and its distribution within the dataset is constrained.

For a clearer demonstration of the small ripples caused by the switching behaviors, the output signals are zoomed in and displayed in the small window in Fig. 8 and Fig. 9. The results indicate that the proposed Neural ODE model can accurately predict the output signals of the Buck and Boost converters under different time scales, even under large signal perturbations.

V. CONCLUSION

This paper introduced a novel data-driven modeling approach for PE converters utilizing neural ODE, which incorporates prior knowledge of numerical ODE solvers into the NN structures. By eliminating the need for iterative solutions of nonlinear equations, the proposed model accelerates simulations. Moreover, it achieves high fidelity by discerning multi-scale characteristics within the dataset, employing a specific filtered MSE loss function.

The findings showcase a noteworthy acceleration in simulation speed compared to traditional detailed numerical methods. Furthermore, this innovative approach ensures a higher level of fidelity, rendering it a valuable tool for the efficient and reliable analysis of PE converters.

In conclusion, this paper addresses the time-consuming nature of detailed numerical simulations for PE converters, stemming from the iterative solution of nonlinear equations. The demonstrated acceleration in simulation speed and improved fidelity highlight the potential impact of this approach on the efficient and reliable analysis of PE converters. The integration of neural ODEs signifies a valuable step towards achieving a balance between computational efficiency and precision in the simulation of complex systems.

REFERENCES

[1] Z. Zhao, D. Tan, B. Shi, Y. Zhu, and H. Jin, "A breakthrough in design verification of megawatt power electronic systems," *IEEE Power Electron. Mag.*, vol. 7, no. 3, pp. 36-43, Sep. 2020.

[2] Han Xu, Jialin Zheng, Yangbin Zeng, Weicheng Liu, Fuhai Zhao, Chunhui Qu, Zhengming Zhao, "Topology-aware matrix partitioning method for FPGA real-time simulation of power electronics systems", *IEEE Trans. Ind. Electron.*, vol. 71, no. 7, pp. 7158-7168, July 2024.

[3] H. Jin, "Behavior-mode simulation of power electronic circuits," *IEEE Trans. Power Electron.*, vol. 12, no. 3, pp. 443-452, May 1997.

[4] Y. Liu, Y. Gao and J. Liang, "Simulation of switched-mode power conversion circuits with extended impedance method," in *IEEE Trans. Circuits Syst. I, Regul. Pap.*, vol. 69, no. 9, pp. 3851-3860, Sept. 2022.

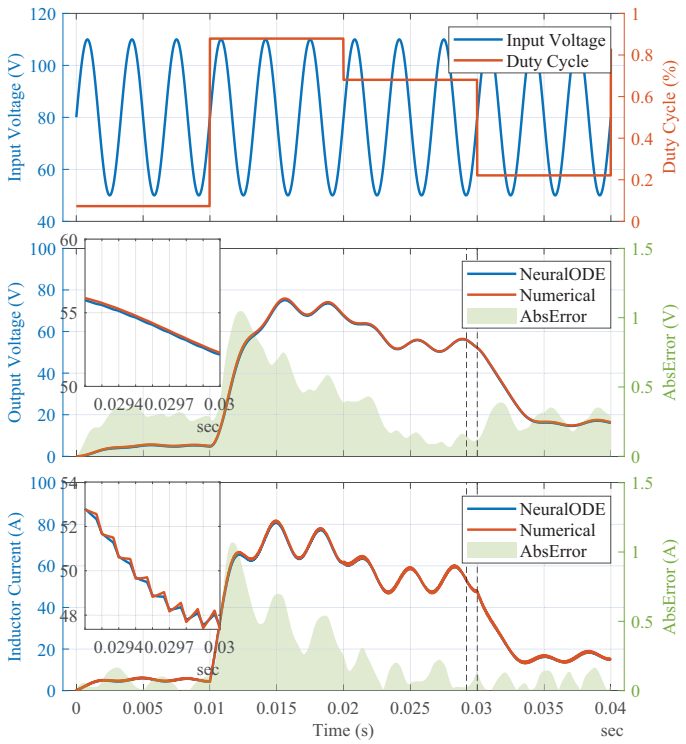
[5] K. Sano, S. Horiuchi, and T. Noda, "Comparison and selection of grid-tied inverter models for accurate and efficient EMT simulations," *IEEE Trans. Power Electron.*, vol. 37, no. 3, pp. 3462-3472, Mar. 2022.

[6] K. Wang, J. Xu, G. Li, N. Tai, A. Tong, and J. Hou, "A generalized associated discrete circuit model of power converters in real-time simulation," *IEEE Trans. Power Electron.*, vol. 34, no. 3, pp. 2220-2233, Mar. 2019.

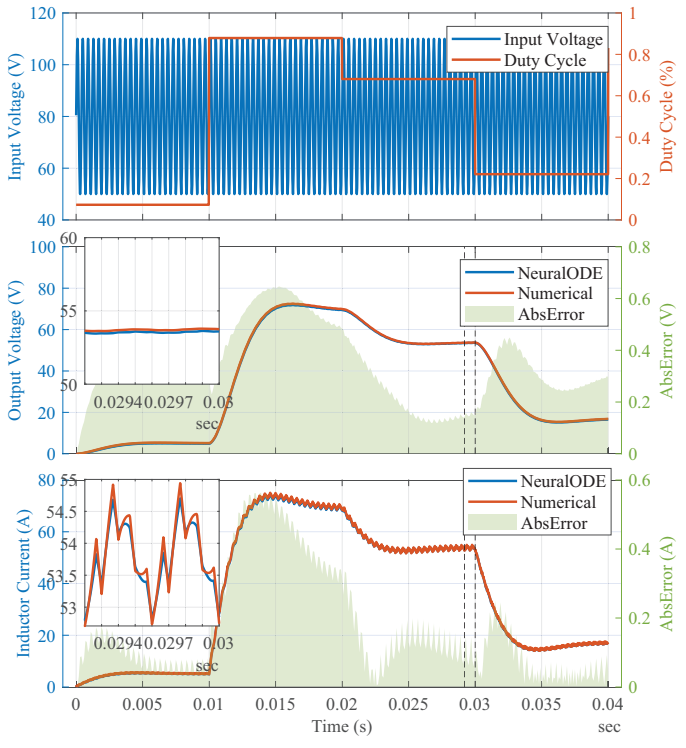
[7] S. Gao, Y. Song, Y. Chen, Z. Yu, and R. Zhang, "Fast simulation model of voltage source converters with arbitrary topology ssing switch-state prediction," *IEEE Trans. Power Electron.*, vol. 37, no. 10, pp. 12167-12181, May 2022.

[8] Y. Zhu, Z. Zhao, B. Shi, and Z. Yu, "Discrete state event-driven framework with a flexible adaptive algorithm for simulation of power electronic systems," *IEEE Trans. Power Electron.*, vol. 34, no. 12, pp. 11692-11705, Dec. 2019.

[9] Q. An, L. Sun, K. Zhao, and L. Sun, "Switching function model-based fast-diagnostic method of open-switch faults in inverters without sensors," *IEEE Trans. Power Electron.*, vol. 26, no. 1, pp. 119-126, Jan. 2011.

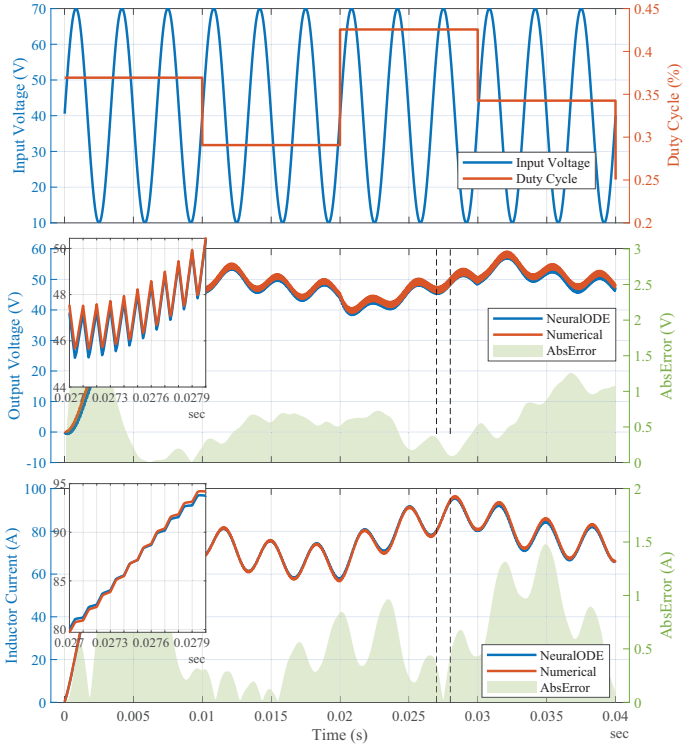


(a) f_{perturb} = 300Hz

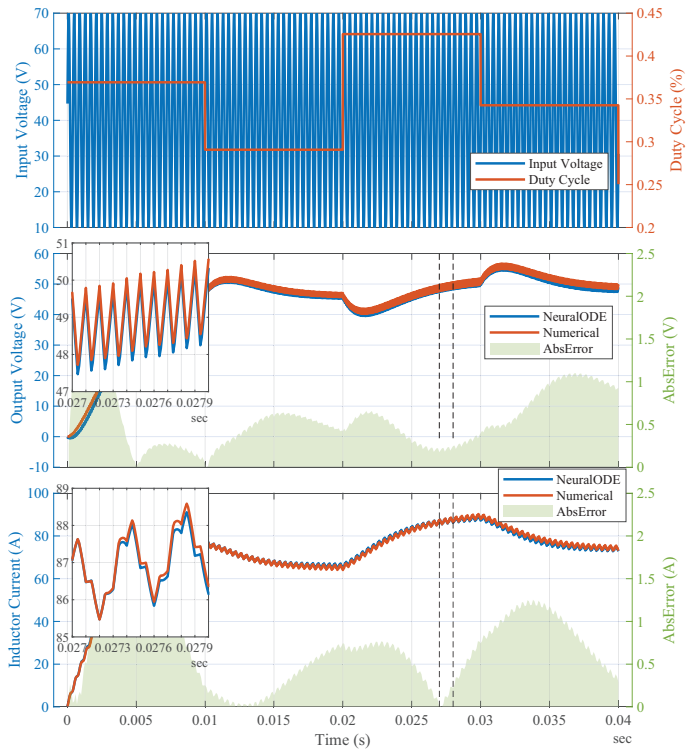


(b) f_{perturb} = 2500Hz

Fig. 8. Modeling of the Buck converter using Neural ODE model.

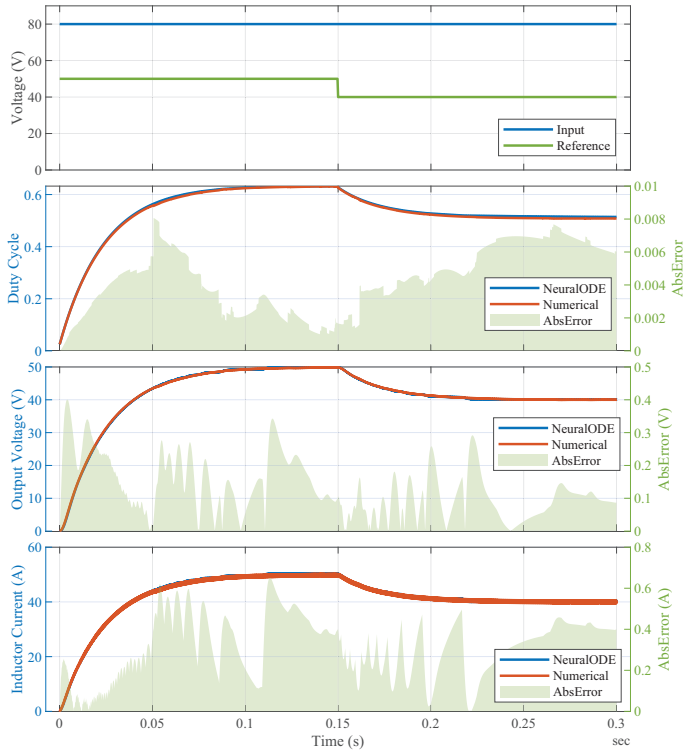


(a) f_{perturb} = 300Hz

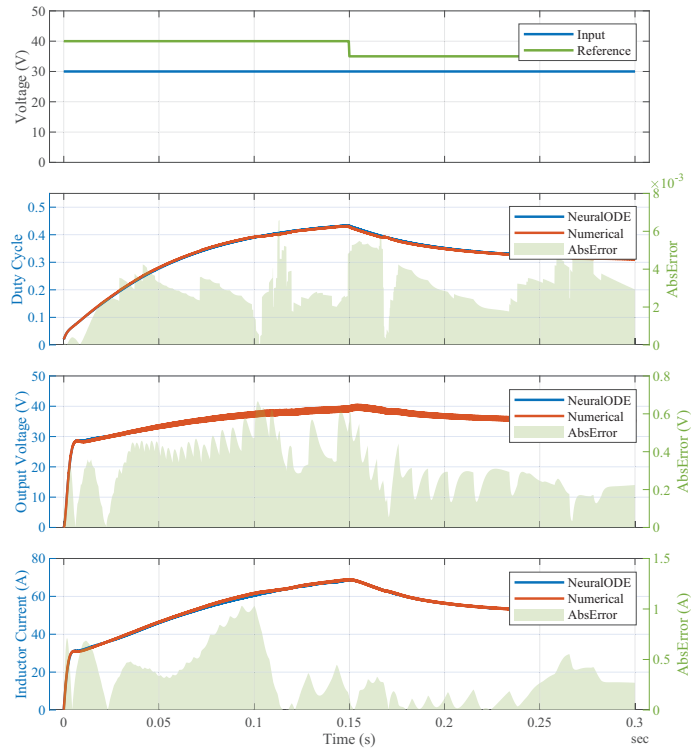


(b) f_{perturb} = 2500Hz

Fig. 9. Modeling of the Boost converter using Neural ODE model.



(a) The closed-loop Buck converter system.



(b) The closed-loop Boost converter system.

Fig. 10. Modeling of the closed-loop Buck and Boost converter using Neural ODE model.

[10] R. W. Erickson and D. Maksimovic, "Circuit Averaging, Averaged Switch Modeling, and Simulation," in *Fundamentals of power Electron.,* 3rd ed. Switzerland AG: Springer Nature, 2020, ch. 14, sec.3, pp. 566-567.

[11] J. Xu, A. M. Gole, and C. Zhao, "The use of averaged-value model of modular multi-level converter in DC grid," *IEEE Trans. Power Del.,* vol. 30, no. 2, pp. 519-528, Apr. 2015.

[12] J. Brandstetter, D. Worrall, and M. Welling, "Message passing neural pde solvers," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Jan. 2022, pp. 1-27.

[13] F. Chen, A. Padilla, P. C. Young, and H. Garnier, "Data-driven modeling of wireless power transfer systems with slowly time-varying parameters," *IEEE Trans. Power Electron.*, vol. 35, no. 11, pp. 12442-12456, Nov. 2020.

[14] M. Al-Greer, M. Armstrong, M. Ahmeid, and D. Giaouris, "Advances on system identification techniques for DC-DC switch mode power converter applications," *IEEE Trans. Power Electron.*, vol. 34, no. 7, pp. 6973-6990, Jul. 2019.

[15] F. Alonge, R. Rabbeni, M. Pucci, and G. Vitale, "Identification and robust control of a quadratic dc/dc boost converter by Hammerstein model," in *Proc. IEEE Energy Convers. Congr. Expo.*, 2014, pp. 3355-3362.

[16] N. Beohar, V. N. K. Malladi, D. Mandal, S. Ozev, and B. Bakkaloglu, "Online built-in self-test of high switching frequency dc-dc converters using model reference based system identification techniques," *IEEE Trans. Circuits Syst. I, Regul. Pap.*, vol. 65, no. 2, pp. 818-831, Feb. 2018.

[17] A. Wunderlich and E. Santi, "Digital twin models of power electronic converters using dynamic neural networks," in *proc. IEEE Appl. Power Electron. Conf. and Expo. (APEC)*, Phoenix, AZ, USA, Jun. 2021, pp. 2369-2376.

[18] G. Rojas-Dueñas, J.-R. Riba, and M. Moreno-Eguilaz, "Black-box modeling of DC-DC converters based on wavelet convolutional neural networks," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1-9, 2021.

[19] G. Rojas-Dueñas, J.-R. Riba, and M. Moreno-Eguilaz, "A deep learning-based modeling of a 270 V-to-28 V DC-DC converter used in more electric aircrafts," *IEEE Trans. Power Electron.*, vol. 37, no. 1, pp. 509-518, Jan. 2022.

[20] M. Moradi, S. A. Sadrossadat, and V. Derhami, "Long short-term memory neural networks for modeling nonlinear electronic components," *IEEE Trans. Compon Packaging. Manuf Technol.*, vol. 11, no. 5, pp. 840-847, May 2021.

[21] H. S. Krishnamoorthy and T. Narayanan Aayer, "Machine learning based modeling of power electronic converters," in *Proc. IEEE Energy Convers. Congr. Expo. (ECCE)*, Baltimore, MD, USA, Sep. 2019, pp. 666-672.

[22] P. Qashqai, K. Al-Haddad, and R. Zgheib, "Modeling power electronic converters using a method based on long-short term memory (LSTM) networks," in *Proc. 46th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Singapore, Oct. 2020, pp. 4697-4702.

[23] H. Ge, Z. Huang, and Z. Huang, "Modeling methodology based on fast and refined neural networks for non-isolated PE converters with configurable parameter settings," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 13, no. 2, pp. 617-628, Jun. 2023.

[24] Q. Shang, F. Xiao, Y. Fan, W. Kang, H. Qin and R. Wang, "Effinformer: A Deep-Learning-Based Data-Driven Modeling of DC-DC Bidirectional Converters," *IEEE Trans. Instru. Meas.*, vol. 72, pp. 1-13, Oct. 2023.

[25] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nat. Rev. Phys.*, vol. 3, no. 6, pp. 422-440, May 2021.

[26] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686-707, Feb. 2019.

[27] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," *Nat. Mach. Intell.* , vol. 3, no. 3, pp. 218-229, Mar. 2021.

[28] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, and A. Anandkumar, "Multipole graph neural operator for parametric partial differential equations," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Virtual, Dec. 2020, pp. 6755-6766.

[29] T. Xiao, Y. Chen, S. Huang, T. He, and H. Guan, "Feasibility study of neural ODE and DAE modules for power system dynamic component modeling," *IEEE Trans. Power Syst.*, vol. 38, no. 3, pp. 2666-2678, May 2023.

[30] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Montréal, Canada, Dec. 2018, pp. 6572-6583.
[31] P. Kidger, "On neural differential equations," Ph.D. dissertation, Math. Inst., Univ. Oxford, Oxford, England, 2021.
[32] A. Vaswani, N. Shazeer, and N. Parmar, "Attention is all you need," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 1-11.
[33] A. E. Cetin, O. N. Gerek, and Y. Yardimci, "Equiripple FIR filter design by the FFT algorithm," *IEEE Signal Process. Mag.*, vol. 14, no. 2, pp. 60-64, Mar. 1997.
[34] J. H. McClellan and T. W. Parks, "A personal history of the Parks-McClellan algorithm," *IEEE Signal Process. Mag.*, vol. 22, no. 2, pp. 82-86, Mar. 2005.

Hanchen Ge received the B.Eng. degree in electrical engineering and automation from North China Electric Power University, Beijing, China, in 2021. He received the M.Eng. degree from South China University of Technology, Guangzhou, China, in 2024. He is now pursuing Ph.D. degree in power system at the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden. His research interest is artificial intelligence in power electronics and power systems.

Yaofeng Liang received the B.Eng. degree in vehicle engineering from South China University of Technology, Guangzhou, China, in 2023. He is now pursuing M.Eng. degree in control science and engineering at South China University of Technology, Guangzhou, China. His research interest is artificial intelligence in power electronics and power systems.

Jinpeng Lei received the B.Eng. degree in vehicle engineering from South China University of Technology, Guangzhou, China, in 2023. He is now pursuing M.Eng. Degree in control science and engineering at South China University of Technology, Guangzhou, China. His research interest is artificial intelligence in power electronics and power systems.

Canjun Yuan received the B.Eng. degree in automation from Wuhan University of Science and Technology, Wuhan, China, in 2022. He is currently pursuing the M.Eng. degree in electronic information at South China University of Technology, Guangzhou, China. His current research interest is machine learning in power electronics.

Zhicong Huang (Senior Member, IEEE) received the B.Eng. degree in electrical engineering and automation and the M.Eng. degree in mechanical and electronic engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2010 and 2013, respectively, and the Ph.D. degree in power electronics from The Hong Kong Polytechnic University, Hong Kong, in 2018.

Since 2020, he has been an Associate Professor with the Shien-Ming Wu School of Intelligent Engineering, South China University of Technology, Guangzhou, China. In 2019, he was a Postdoctoral Fellow under the UM Macao Talent Program with the State Key Laboratory of Analog and Mixed-Signal VLSI, Macau, China. He is interested in power electronics techniques in power systems and electric vehicles.

Dr. Huang has received reviewer awards from a variety of IEEE journals including JETCAS, TPE and JESTPE.