IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

Data-Driven Parameter Estimation of Lumped-Element Models Via Automatic Differentiation

ALESSANDRO ILIC MEZZA, (Graduate Student Member, IEEE), RICCARDO GIAMPICCOLO, (Member, IEEE), and ALBERTO BERNARDINI, (Member, IEEE)

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milan, Italy
(e-mail: alessandroilic.mezza@polimi.it, riccardo.giampiccolo@polimi.it, alberto.bernardini@polimi.it)

Corresponding author: Alessandro Ilic Mezza (e-mail: alessandroilic.mezza@polimi.it).

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 — program "RESTART") and received funding support as part of the JRC STEAM STM–Politecnico di Milano agreement.

ABSTRACT Lumped-element models (LEMs) provide a compact characterization of numerous real-world physical systems, including electrical, acoustic, and mechanical systems. However, even when the target topology is known, deriving model parameters that approximate a possibly distributed system often requires educated guesses or dedicated optimization routines. This article presents a general framework for the data-driven estimation of lumped parameters using automatic differentiation. Inspired by recent work on physical neural networks, we propose to explicitly embed a differentiable LEM in the forward pass of a learning algorithm and discover its parameters via backpropagation. The same approach could also be applied to blindly parameterize an approximating model that shares no isomorphism with the target system, for which it would be thus challenging to exploit prior knowledge of the underlying physics. We evaluate our framework on various linear and nonlinear systems, including time- and frequency-domain learning objectives, and consider real- and complex-valued differentiation strategies. In all our experiments, we were able to achieve a near-perfect match of the system state measurements and retrieve the true model parameters whenever possible. Besides its practical interest, the present approach provides a fully interpretable input-output mapping by exposing the topological structure of the underlying physical model, and it may therefore constitute an explainable ad-hoc alternative to otherwise black-box methods.

INDEX TERMS Automatic differentiation, backpropagation, lumped-element models, parameter estimation.

I. INTRODUCTION

ACCURATE equivalent models of physical systems are highly sought after across all engineering sectors thanks to their efficient implementation and mathematical tractability. In particular, lumped-element models (LEMs) offer a compact and interpretable representation of countless spatially distributed physical systems and, by reducing a possibly infinite-dimensional state space to a finite number of idealized components interconnected by a known topology, they enable the application of network analysis methods not only to electrical circuits but also, e.g., to the mechanical, biological, thermodynamic, fluid, and acoustic domain.

Moreover, as far as discrete-time simulations are concerned, LEMs tend to be more robust than data-driven and black-box methods such as Wiener-Hammerstein models [1], Volterra series [2], and neural networks [3], which risk producing meaningless and physically inconsistent outputs when presented with novel and possibly out-of-distribution inputs [4]. This is due to a desirable characteristic of white-box models sometimes referred to as *inductive bias* in the machine learning community [5]. The topology of the LEM, indeed, establishes a set of relations, or *biases*, between the model parameters that, in turn, ensure compliance with the underlying physics laws governing the system.

On the downside, one of the main challenges of LEMs is the characterization of the lumped elements. Parameter estimation theory aims to find the parameters of a given model that best fit a (partial) set of observational data, possibly by solving a global least squares or maximum likelihood estimation problem [6]. This task belongs to the broader family of

so-called inverse problems, which are notoriously difficult to solve when the number of unknown parameters is large [7].

At the same time, in the past two decades, the sheer availability of observational data and the increasing ease of collecting them have led to the development of a vast array of machine learning techniques. In particular, thanks to the technological advancements in hardware acceleration, automatic differentiation (AD) has become a viable and powerful strategy for large-scale optimization [8]–[10]. Ultimately, one could think of the supervised training of a neural network via backpropagation as the solution to a classic inverse problem where the hidden layer parameters (weights and biases) are estimated from a set of observations. Yet, the promising applications of modern AD tools have remained mainly confined to the deep learning field, and backpropagation is arguably most known as an off-the-shelf method for fitting black-box neural network models.

This is not to say that cross-contamination between physically consistent models and deep-learning-inspired optimization never occurred. Analog neural networks have long explored the possibility of realizing a target input-output mapping by means of the interconnection of a finite set of electronic hardware components [11]. More recently, replacing the multilayered affine transformations paired with nonlinear activation functions typical of feed-forward neural networks for full-fledged physical systems has been investigated by Wright et al. [12]. The inputs to the resulting models, named physical neural networks, can be trained via physics-aware backpropagation using the derivatives estimated by applying AD on a parallel differentiable digital model. Notably, AD had been previously applied to electrical circuit modeling. In the early nineties, despite the limited computational resources available at the time, Feldmann et al. [13] advocated for the use of AD to evaluate the Jacobian matrices needed for circuit simulations. In [14], AD was adopted for optimizing artificial port resistances in multi-dimensional Wave Digital Filter (WD) structures with topology-related delay-free-loops; the same approach was then extended in [15] for the case of WD structures containing multiple nonlinearities. Recently, Shintani et al. [16], building upon the results of [17], showed that AD-based parameter extraction of a power metal-oxide-semiconductor field-effect transistor (MOSFET) can be up to 3.5 times faster than a corresponding numerical differentiation method. Furthermore, inspired by prior work on differentiable digital signal processing [18], Esqueda et al. [19] proposed to optimize virtual analog models using backpropagation to fit measurements of reference audio circuits.

Building on this promising trend, in this article, we present a general framework for achieving an explicit and fully interpretable mapping by incorporating a LEM of the target physical system into the forward pass of a learning algorithm. This allows us to train the LEM parameters with standard gradient-based methods to minimize an arbitrary cost function between the model outputs and some measurable quantities of the target system. In fact, akin to traditional neural networks, the LEM can be optimized via backpropagation exploiting AD to compute the gradient of one or multiple loss functions with respect to the trainable model parameters. Thanks to the inherent inductive biases, the present approach enforces physical constraints by means of the lumped model topology and constitutive relations rather than via auxiliary loss functions or regularizers. Since no bias is imposed on the learning objective, such a data-driven optimization framework accounts for both time- and frequency-domain models and loss functions, interchangeably accommodating real- and complex-valued formulations.

The learned model has all the desirable characteristics of the underlying lumped system, i.e., it can be implemented efficiently, the simulation is typically lightweight, the inference is stable and does not require hardware graphic acceleration, and the model discrepancy can be assessed a priori. Furthermore, learned parameters may characterize a target real-life device and can thus be employed both for simulation purposes, e.g., the design of digital twins [20], or downstream physics-based digital signal processing tasks, e.g., transducer virtualization [21].

Whereas the method inherently requires little to no prior information other than the LEM constitutive equations, the convergence rate can be controlled by an educated choice of the hyperparameters driven by the knowledge of the target physical system. In this regard, practical applications include learning the device-specific divergence between nominal and actual component values due to manufacturing tolerance, and compensating for both short-term (e.g., changes in operating temperature) and long-term effects (e.g., device aging and material degradation) if applied repeatedly over time.

Existing AD engines enable general-purpose tensor manipulation and constitute a straightforward replacement for more widespread programming languages and scientific computing libraries. Hence, the application of the proposed framework entails minimal modification to existing LEM implementations, thus making it readily accessible to researchers and practitioners.

In this work, we investigate a series of illustrative case studies and offer practical solutions inspired by the best practices of modern neural network training to problems specific to the use case. Our experiments show that all models considered in the present work can achieve a near-perfect match of the target behavior while consistently retrieving the underlying physical parameters.

The remainder of the manuscript is organized as follows. In Section II, we provide an overview of AD. In Section III, we outline the methodology and the implementation scheme of the LEMs under scrutiny. In Section IV-A, we discuss the use of AD to estimate the parameters of two linear time-domain systems: a simple Bridged-T network and a Thiele-Small multiphysics model of an electrodynamic loudspeaker. In Section IV-C, we match a target transfer function by means of a lowpass Sallen-Key filter optimized via complex differentiation. In Section IV-E, by estimating the circuit parameters of a diode-based dynamic ring modulator, we show that the

present approach is also capable of dealing with nonlinear time-domain systems. Finally, Section V concludes this work.

II. BACKGROUND ON AUTOMATIC DIFFERENTIATION

Automatic differentiation (AD) [8]–[10] refers to a set of techniques for algorithmically evaluating the derivatives of a differentiable function expressed as a computer program. It differs from, e.g., numerical and symbolic differentiation methods in that AD involves a non-standard interpretation of computer programs where each algebraic operation is augmented with the calculation of the corresponding derivative. Oftentimes, all computations in an algorithm can be expressed as a composition of a finite set of J elementary operations for which derivatives are known. Hence, having stored the results $w_1, \ldots, w_J$ of all intermediate operations in a data structure known as Wengert list [22], the derivatives of an arbitrary function $y(u)$ with respect to the input variable u can be obtained by repeatedly applying the chain rule

$$\frac{\partial y}{\partial u} = \frac{\partial y}{\partial w_J}\frac{\partial w_J}{\partial w_{J-1}} \cdots \frac{\partial w_2}{\partial w_1}\frac{\partial w_1}{\partial u}. \tag{1}$$

The way in which the chain rule is traversed determines the main mode of operation of the AD algorithm. In the literature, two flavors are typically reported: forward accumulation mode and reverse accumulation mode. In forward accumulation mode, the chain rule is evaluated from the inside out, starting from the input variables and computing the local derivative of the first expression in the program. Each operation is therefore augmented by extra code returning the derivative with respect to its input.

Conversely, AD in reverse accumulation mode [23] corresponds to a generalized backpropagation algorithm [24], in that derivatives are propagated backward from a given output. This is done by complementing each intermediate variable w_i with an adjoint $\partial y_i/\partial w_i$. In practice, reverse-mode AD relies on building a directed acyclic graph (DAG) where all data, executed operations, and resulting tensors are topologically arranged according to the order of their execution in the program. In this DAG, leaf vertices correspond to the input tensors, and root vertices to the output tensors. Hence, the gradient is computed by evaluating the chain rule by traversing the computational graph from roots to leaves. This way, reverse-mode AD greatly reduces the number of operations required for differentiating functions with many inputs compared to the forward mode [10].

In modern AD engines such as JAX [25], PyTorch [26] and TensorFlow [27], computational graphs are dynamically determined at run time. Therefore, a differentiable program may contain control statements such as `if`, `for`, `while`, and the gradient can be seamlessly propagated through a possibly different DAG at every iteration.

In the case of a simple single-input single-output feed-forward network, the DAG consists of one linear branch obtained by subsequently storing the gradient functions and outputs of all hidden layers. In the case of sample-by-sample inference, instead, the DAG can be thought of as a stack of

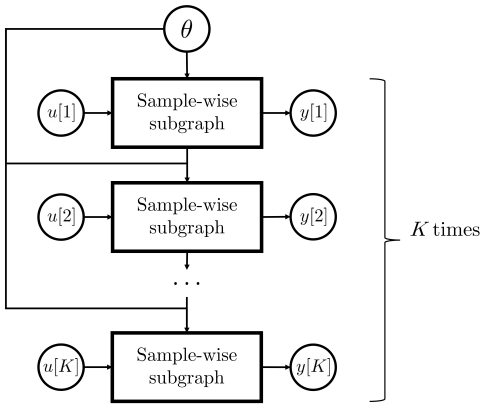

FIGURE 1. Computational graph for the automatic differentiation of $y[k] = \varphi(u[k]; \theta)$ with respect to the parameter θ for $k = 1, \ldots, K$.

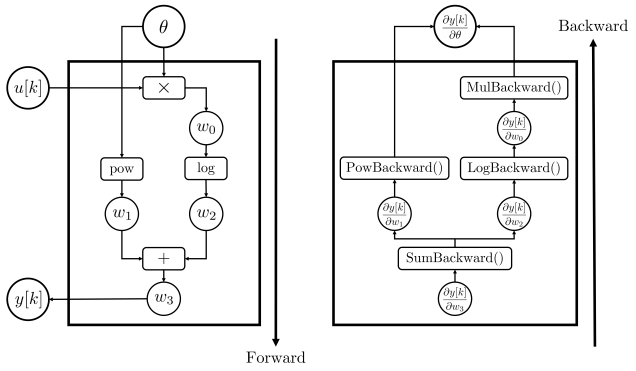

FIGURE 2. Sample-wise subgraph for the automatic differentiation of $y[k] = \log(\theta \cdot u[k]) + \theta^2$ with respect to the parameter θ.

identical subgraphs, as shown in Fig. 1. Each subgraph is responsible for the instantaneous mapping between the input $u[k]$ and the corresponding output $y[k]$. When the computation of the kth subgraph relies on a set of previous state variables, the computational graph is dynamically updated, and the gradient is let flow through the DAG obtained by unfolding all the $k-1$ previous subgraphs. In turn, this yields short computational branches associated with the first few samples and progressively longer branches as k increases.

An example of a sample-wise subgraph is given in Fig. 2. In the forward pass, all elementary operations are executed, and the intermediate variables w_i are stored. Simultaneously, the DAG is populated with the gradient functions associated with every primitive. Then, in a backward pass, the gradients are computed locally and iteratively aggregated all the way to the leaves. Therefore, unlike numerical differentiation methods, AD does not incur truncation errors and, for what concerns real arithmetic, yields exact results accurate to machine precision.

Reverse-mode AD constitutes the backbone of the classic backpropagation algorithm, the leading method for training

deep neural networks. In the next section, we describe how it can be employed for the parameter estimation of LEMs.

III. PARAMETER ESTIMATION VIA AUTOMATIC DIFFERENTIATION

The proposed framework for LEM parameter estimation hinges on the idea of recasting the problem into a classic deep learning (DL) formulation. Feed-forward neural networks can be described as a cascade of interleaved parametric affine transformations and static point-wise nonlinearities. The overall function composition can be thus written as

$$\hat{\mathbf{y}}_k = \varphi(\mathbf{u}_k; \boldsymbol{\theta}), \tag{2}$$

where $\mathbf{u}_k \in \mathbb{C}^N$ is the kth (possibly) complex-valued input vector taken from a dataset of K samples, and $\hat{\mathbf{y}}_k \in \mathbb{C}^M$ is the corresponding output vector, which is conditioned on the model parameters $\boldsymbol{\theta} = [\theta_1, ..., \theta_P]^{\mathrm{T}}$. The main task of DL is that of learning $\boldsymbol{\theta}$ such that the resulting output follows some predetermined criteria. In a typical supervised scenario, this is achieved by minimizing a target loss function defined on $\hat{\mathbf{y}}_k$ and some ground truth data $\mathbf{y}_k$ for $k = 1, ..., K$.

Likewise, the input-output relationships characterizing a lumped physical system can also be expressed as in (2), where $\mathbf{y}_k$ denotes a snapshot of M observable physical quantities sampled at time k, and $\boldsymbol{\theta}$ indicates the LEM parameters. Just as in the previous case, our goal is to determine $\boldsymbol{\theta}$ so that the model outputs $\hat{\mathbf{y}}_k$ match the desired behavior for $k = 1, ..., K$.

Following this analogy, it is therefore natural to think of a LEM as a parametric mapping $\varphi(\mathbf{u}_k; \boldsymbol{\theta})$ whose trainable parameters can be optimized drawing from a large corpus of well-established DL techniques. For instance, we can optimize the LEM using the classic backpropagation algorithm and avail ourselves of the best practices in neural network training.

Namely, given a series of K exogenous variables $\mathbf{u} = [\mathbf{u}_1, ..., \mathbf{u}_K]^{\mathrm{T}}$, we start by setting the LEM parameters according to some initial guess, compute the model outputs $\hat{\mathbf{y}} = [\hat{\mathbf{y}}_1, ..., \hat{\mathbf{y}}_K]^{\mathrm{T}}$ given the current set of parameters $\boldsymbol{\theta}$, and evaluate an objective function $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$, where $\mathbf{y} = [\mathbf{y}_1, ..., \mathbf{y}_K]^{\mathrm{T}}$. Then, we use reverse-mode AD to compute $\nabla_\theta \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$ and solve the following minimization problem using either stochastic gradient descent or one of its more recent adaptive extensions [28]–[31]:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}). \tag{3}$$

The iterative process is repeated multiple times until convergence; borrowing from DL naming conventions, we refer to each iteration encompassing all K input snapshots in $\mathbf{u}$ as an epoch.

Despite the overarching parallelism, however, the proposed AD-based estimation technique and traditional neural networks trained with backpropagation are fundamentally different. Whereas the former yields a physically-consistent set of lumped parameters thanks to the inductive bias granted by imposing the LEM topology and constitutive equations as priors, the latter learn a large number of weights and biases. Although biologically inspired, weights and biases are non-semantic and bear no direct physical interpretation for what concerns the system being modeled unless additional learning constraints are imposed.

In this work, we assume no learning bias, and loss functions can be thus defined freely and tailored to the specific learning task. In fact, various losses can be employed depending on the use case. These include classic regression objectives, such as the mean squared error between predicted and measured time-domain signals, as well as loss functions defined on complex-valued transfer functions in the frequency domain.

In the literature, regularization is typically enforced by means of auxiliary losses constraining the ℓ_p-norm of the trainable parameters [32]. Due to the gigantic size of modern neural networks, this is typically performed in a layer-wise fashion. However, in our case study, one often deals with a tractable number of lumped elements so dedicated regularization can be applied to each parameter instead.

Similarly, multiple optimizers can be run sequentially or alternately. In particular, we found it beneficial to apply several adaptive gradient-based methods acting on different subsets of the model parameters. The learning rate can be independently tuned for each optimizer, and it might be helpful to inform the decision based on some prior information about the underlying model. Indeed, governing laws or expert knowledge often suggest a reasonable range of values a parameter may take. We observed that setting the optimizer learning rates in the range of the expected order of magnitude of the target parameters leads to a faster and more consistent convergence rate.

Alternatively, such information can also be injected directly into the LEM implementation. When the expected order of magnitude is known a priori, one could, in fact, multiply a certain parameter by that quantity as it enters the computational flow of the program, effectively incorporating the operation in the AD graph. As a result, the method learns a scaled version of the true parameter. This could be exploited to bring all parameters in a similar range, hence aiding joint optimization. Whereas the former approach grants more flexibility in terms of model implementation, the latter allows one to perform hyperparameter tuning more freely, e.g., by choosing the learning rates via a simple grid search.

A similar approach can be used to place hard constraints on the LEM parameters. For instance, many physical quantities, such as mass and electrical resistances, are typically required to be non-negative. This may be enforced by inserting a differentiable nonlinearity, such as an exponential or softplus function, at the program entry point for a given trainable parameter. This way, one would effectively learn the arguments of such functions, and the parameter values can be retrieved by applying the same nonlinearities to the output of the optimization process. In practice, the requirements can also be relaxed to include functions that are differentiable almost everywhere, such as the absolute value and the well-known rectified linear unit function.

Ultimately, we found that the proposed model-based learning framework can rely on just a tiny amount of data. In the following, we use no more than a few milliseconds of a single set of system-state measurements. This is another key difference with respect to deep neural networks, whose training would require extensive datasets collected for many different types of inputs. This unique characteristic of our methodology opens the possibility for, e.g., online learning throughout the lifecycle of a device or otherwise impractically expensive finite element analyses aimed at simulating just enough data to fit a lightweight digital twin.

Contrary to expectations, increasing the training data size beyond a certain limit appears to worsen the model performance. Feeding the algorithm with many input samples would cause the DAG to dynamically grow in size. In turn, this might lead to critical gradient flow problems. Indeed, as multiple branches with length proportional to the sample index k converge at each DAG leaf, one could expect the aggregated gradients to either explode or start vanishing for longer input sequences. In principle, one could batch the input signals and perform multiple gradient updates within one epoch. In our experiments, however, we did not find it to be helpful as convergence was always achieved in a small-data regime, so we opted for a single-batch strategy instead.

AD-based lumped parameter estimation is general and places no constraint on the LEM topology as long as it can be characterized in terms of input-output mappings described by the composition of differentiable tensor primitives. Hence, the present framework could be seamlessly applied to countless multiphysics scenarios, including mechanical, magnetic, acoustic, thermal, and fluid systems. In general, this holds true also for approximating models for which there exists no isomorphism with the target physical system. In the following, however, we focus on electrical equivalent models since they allow for precise and efficient numerical simulation and provide exact ground truth for the model parameters.

A. WHY AUTOMATIC DIFFERENTIATION?

At its core, the proposed framework entails a gradient-based optimization algorithm. As such, it relies on the gradients with respect to the stochastic objectives to iteratively update an initial parameter estimate. In specific cases, it would be feasible to analytically compute such gradients. However, in many instances, this may prove hard to set up or even impossible.

First, analytical and symbolic approaches would require expressing the system's characteristics in terms of the trainable parameters, which can be a daunting task when dealing with complex, high-dimensional models. Moreover, even if such a function exists and can be derived, it might not be possible to make it explicit with respect to each parameter for computing the gradients. AD provides a solution by allowing for the automatic and efficient computation of the gradients, relieving the need to manually define such functions.

Furthermore, in the case of nonlinear systems, functions can be extremely challenging to differentiate. AD excels in handling such complexity, as it yields error-free gradients for even the most intricate nonlinear functions.

Analytical and symbolic differentiation methods are likely to become unfeasible when working with a large number of parameters, as they demand the derivation of an equally large number of gradients. This computational burden can quickly become overwhelming. AD, on the other hand, effortlessly manages high-dimensional parameter spaces, making it a more practical choice for many real-life LEMs.

When a program implementing a LEM involves iterative methods (e.g., fixed-point, Newton-Raphson, etc.), the number of iterations required to reach convergence is typically data-dependent. Iterative methods are usually needed for solving LEMs whose time-derivatives are discretized by means of implicit methods, or when nonlinear elements and/or delay-free loops are present [33]. In order to express the analytic gradients in such cases, the number of iterations must be decided in advance, degrading the accuracy of the model, which in some cases may lead to completely wrong solutions. It is important to stress that for-loops where system variables depend on the result of previous iterations can be seen as composite functions; defined I as the number of iterations, an analytical approach would entail differentiating an I-composite function, which is known to be challenging for $I \gg 1$. Thus, there is a trade-off between the number of iterations and the number of derivatives to compute. On the one hand, if we want to keep the number of iteration low for easing the differentiation step, the model could not reach convergence and lead to wrong solutions. On the other hand, if we want to ensure convergence, we may opt for a conservatively high number of iterations but this entails the computation of the gradients of complex composite functions, which may be impractical. AD circumvents this issue by seamlessly accommodating loops and the data-dependent nature of convergence, ensuring that the model reaches an accurate solution without compromising on computation efficiency.

Lastly, models incorporating conditional statements, such as `if-else` constructs, require the computation of derivatives for all possible branches in advance. AD excels in this situation, as it dynamically rebuilds the DAG at each step in a data-dependent fashion, making it feasible for modeling with conditionals and branching modes of operation.

Despite these favorable properties, it is worth pointing out that the proposed AD-based lumped parameter estimation framework present some limitations. For instance, the method may fail to converge in presence of severe signal degradation due, e.g., to additive noise (see Section IV-A2), and may exhibit model identifiability problems depending on the choice of observables (see Section III-B).

B. PARAMETER UNIDENTIFIABILITY

In general, given a LEM, there may exist infinitely many parameter configurations that produce the same outputs. Therefore, it is possible for multiple learned models to all match the same target observables while being described by widely

different sets of parameters depending on the initialization and training routine. We refer to this property as *parameter unidentifiability*.

In many use cases, especially when the goal is to simply approximate a target physical behavior, parameter unidentifiability is of no practical relevance as any of the infinitely many equivalent LEMs would fulfill the task. Besides, a principled choice of initial values and hyperparameters may still yield a set of physically-consistent parameters.

Parameter identifiability becomes critical when the goal is to estimate unknown physical quantities, e.g., when fitting a specific real-life device. For instance, one might want to quantify the deviation of the electrical elements in a given circuit from their nominal values. In that case, the focus is placed on retrieving true component values by observing, e.g., port voltages and currents, rather than realizing an equivalent network. Therefore, one must pay particular attention to which quantities are to be measured. Indeed, by simultaneously fitting multiple observables, one could typically restrict the degrees of freedom of the system and avoid parameter unidentifiability. We will address this particular scenario in Section IV-A when dealing with the case of the Bridged-T network.

C. LUMPED-ELEMENT MODEL IMPLEMENTATION

In the literature, the discrete-time implementation of circuits is addressed following different approaches [34]–[36]. In this work, all the considered LEMs make use of Wave Digital Filters (WDFs) [35] and are implemented in Python taking advantage of the PyTorch Autograd engine [26]. WDFs are a particular class of digital filters based on physical modeling principles [35]. The reference circuit (in our case, the LEM under consideration) is represented as an interconnection of input-output blocks characterized by scattering relations. In fact, the so-called Kirchhoff variables (port voltages and port currents) are substituted with a linear combination of incident and reflected *waves*, whose most spread definition is [35]

$$a = v + Zi, \quad b = v - Zi, \tag{4}$$

where a is the incident wave, b is the reflected wave, whereas Z is a free parameter called *port resistance*. Moreover, in the Wave Digital (WD) domain, the topology and element descriptions are addressed independently, enabling an efficient implementation of nonlinear elements [37], [38].

Whereas losses based on wave variables could be defined, in the following, we minimize objective functions in the Kirchhoff domain by taking the inverse mapping of (4) and incorporating such operation in the DAG. In practice, however, not all M pairs of Kirchhoff variables might be measurable at every $k = 1, ..., K$. Hence, the observability of a partial or noisy set of physical quantities can be accounted for by masking the loss functions with (sparse) binary matrices, thus zeroing out the gradients associated with missing or outlier data. Furthermore, we compute the absolute value as the first operation in the forward AD graph in order to enforce parameter non-negativity. Finally, since raw physical parameters may take values well below the machine precision granted by 32-bits floating-point numbers, all our implementations use double-precision floating-point numbers.

IV. EXPERIMENTS

In this section, we provide a series of experiments concerning the application of the proposed framework to different scenarios (linear and nonlinear LEMs, time and frequency domains, etc.), pointing out its features and wide generality. In Section IV-A, we test the proposed methodology on a Bridge-T network (Fig. 3), in Section IV-B on a Thiele-Small network modeling an electrodynamic loudspeaker (Fig. 4), in Section IV-D on a lowpass Sallen-Key filter (Fig. 5), while in Section IV-F on a typical ring modulator circuit (Fig. 6).

In the following, we will use the Normalized Mean Squared Error (NMSE) between the true and estimated parameters as a metric to assess the performance of the method, i.e.,

$$\epsilon(\boldsymbol{\theta}^\star) = \frac{1}{P}\sum_{p=1}^{P}\frac{|\theta_p - \hat{\theta}_p|^2}{|\theta_p|^2}, \tag{5}$$

where P is the total number of model parameters.

A. LINEAR TIME-DOMAIN MODELS: BRIDGED-T NETWORK

As a first case study, let us consider the simple Bridged-T network shown in Fig. 3 and implemented in the WD domain as in [39]. Our goal is to estimate all model parameters starting from some initial guess regarding their values so to match time-domain system measurements. The ground truth for the target parameters is as follows: $R_1 = 2$ kΩ, $R_2 = 20$ kΩ, $R_3 = 5$ kΩ, $R_4 = 3$ kΩ, $C_1 = 3.2$ nF. The input signal $u[k]$ at a sampling rate of 96 kHz determines the voltage of the ideal generator V_{in} for $k = 1, ..., K$. We express $u[k]$ in tensor notation as $\mathbf{u} \in \mathbb{R}^{K\times 1}$. We also define the network observables $\mathbf{y}$ as a tensor of M port voltages and currents. Namely, $\mathbf{y} \in \mathbb{R}^{K\times 2M}$ can be written as $\mathbf{y} = [\mathbf{y}_1, ..., \mathbf{y}_K]^{\mathrm{T}}$, where

$$\mathbf{y}_k := \left[v_1[k], ..., v_M[k], i_1[k], ..., i_M[k]\right]^{\mathrm{T}}. \tag{6}$$

We optimize the Bridged-T network by minimizing the following NMSE loss function for a total of 3000 epochs:

$$\mathcal{L}_{\text{BT}}(\mathbf{y}, \hat{\mathbf{y}}) := \frac{1}{2MK}\sum_{m=1}^{2M}\sum_{k=1}^{K}\frac{(y_m[k] - \hat{y}_m[k])^2}{y_m[k]^2}. \tag{7}$$

The objective function in (7) can be thought of as the sum of the NMSE relative to voltages $v_1[k], ..., v_M[k]$ and the NMSE relative to currents $i_1[k], ..., i_M[k]$, i.e.,

$$\mathcal{L}_{\text{BT}}(\mathbf{y}, \hat{\mathbf{y}}) := \text{NMSE}_{\text{v}} + \text{NMSE}_{\text{i}}, \tag{8}$$

where

$$\text{NMSE}_{\text{v}} := \frac{1}{MK}\sum_{m=1}^{M}\sum_{k=1}^{K}\frac{(v_m[k] - \hat{v}_m[k])^2}{v_m[k]^2}, \tag{9a}$$

$$\text{NMSE}_{\text{i}} := \frac{1}{MK}\sum_{m=1}^{M}\sum_{k=1}^{K}\frac{(i_m[k] - \hat{i}_m[k])^2}{i_m[k]^2}. \tag{9b}$$

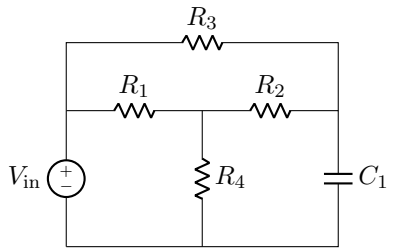

FIGURE 3. Bridged-T network.

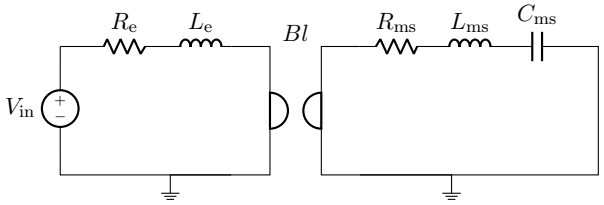

FIGURE 4. Thiele-Small model of an electrodynamic loudspeaker with linear electromechanical coupling *Bl* (gyrator).

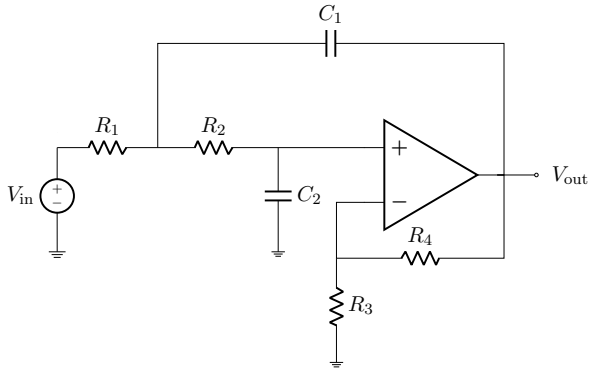

FIGURE 5. Lowpass Sallen-Key filter.

Resistances and capacitances take values in hugely different ranges spanning several orders of magnitude. Therefore, we found it beneficial to alternately apply two optimizers, the first acting only on the resistors and having a large learning rate, and the second acting on all circuit elements with a lower learning rate. Specifically, we use two Adam optimizers [30] with learning rate 10^3 and 10^{-10}, respectively. We empirically observed that the algorithm achieved convergence only if, at every iteration, the global optimizer step followed the one updating the four resistances.

We fed the Bridged-T network with three different input signals of a duration of 1.6 ms: a sine and a square wave, both with unit amplitude and frequency $f_0 = 3000$ Hz, and white Gaussian noise with zero mean and unit variance. The results obtained observing all circuit elements are reported in Table 1 and the corresponding loss functions are depicted in Fig. 7. After 3000 epochs, the losses read 4.09×10^{-21}, 9.09×10^{-9}, and 9.06×10^{-7} for sine, square, and white noise inputs, respectively. In turn, this results in $\epsilon(\boldsymbol{\theta}^\star) = 0$ for the sinusoidal input, $\epsilon(\boldsymbol{\theta}^\star) = 8.69 \times 10^{-6}$ for the square wave input, and $\epsilon(\boldsymbol{\theta}^\star) = 3.06 \times 10^{-6}$ for white noise input. Table 1 shows that the minimization procedure yields parameters matching the ground truth regardless of the input type. In the worst case, i.e., R_2 for the square wave input, the percentage estimation error falls below 0.3% of the true parameter value.

TABLE 1. Bridged-T Network (3000 Epochs).

	R_1 (Ω)	R_2 (Ω)	R_3 (Ω)	R_4 (Ω)	C_1 (nF)
Ground truth	2000	20 000	5000	3000	3.2
Initial guess	1000	1000	1000	1000	10
Sine wave	2000	20 000	5000	3000	3.2
Square wave	2000.06	19 940.91	5000.79	3000.07	3.201
White noise	1997.85	19 964.93	4999.96	2996.38	3.201

1) Example of Parameter Unidentifiability

Inspecting the network topology, we may notice that R_1 and R_4 form a voltage divider characterized by

$$v_{\text{R}_4}[k] = \frac{R_4}{R_1 + R_4} V_{\text{in}}[k]. \tag{10}$$

Therefore, there exist infinitely many combinations of R_1 and R_4 resulting in the same voltage across R_4. Without observing either component, we are bound to face parameter unidentifiability when optimizing the network (see Section III-B).

In this regard, we performed two experiments: we trained the model observing only C_1 (triggering unidentifiability) and observing both C_1 and one of the two resistors in the voltage divider (in this case R_4). In both cases, we use the sinusoidal input detailed in the previous section. The loss functions are depicted in Fig. 11 and the exploration of the parameter space is shown in Fig. 8a and 8b, respectively. As evidenced by the (numerically) zeroed training losses, the target voltages and currents are perfectly matched in both cases. However, when observing only the capacitance, the resistances promptly converge to different minima, as shown in Fig. 8b, yielding an equivalent model. Notably, C_1 is correctly estimated in both cases, as its value is key for the correct behavior of the output variables.

2) Noise Robustness

To evaluate the robustness with respect to the presence of sensor noise, we simulate a more realistic scenario in which the observations of voltages and currents are corrupted by additive white Gaussian noise (AWGN). Specifically, we assume that each measurement is characterized by a signal-to-noise ratio (SNR) of 20 dB. We repeat the experiment in Section IV-A1 observing the now corrupted measurements on C_1 and R_4. As illustrated in Fig. 9, the exploration of the parameter space seems not to be substantially affected by the presence of AWGN. After 3000 epochs, the estimated parameters are $\hat{R}_1 = 2020.87\ \Omega$, $\hat{R}_2 = 20184.77\ \Omega$, $\hat{R}3 =$

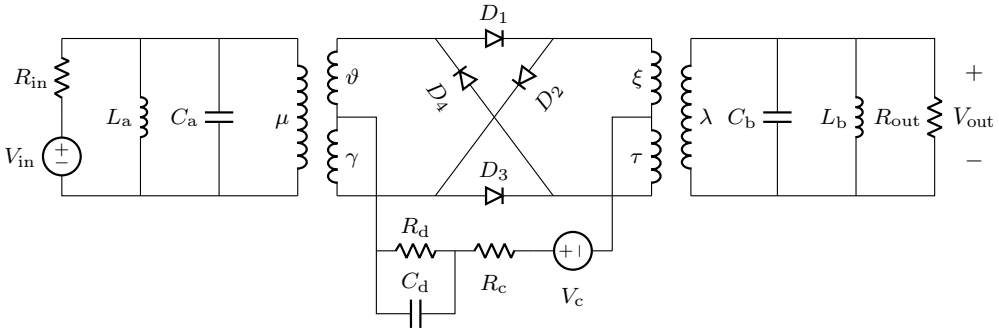

FIGURE 6. Diode-based dynamic ring modulator.

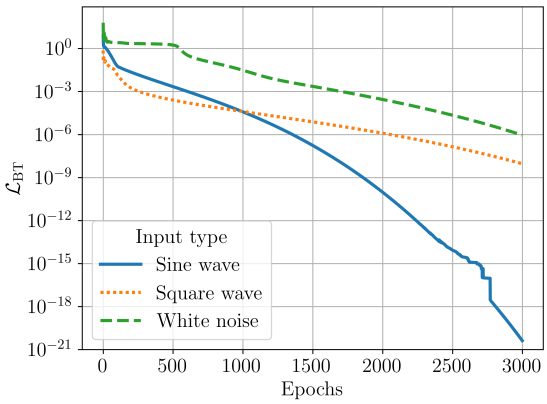

FIGURE 7. Trend of Bridged-T network loss functions using different input signals, i.e., a sine wave (solid), a square wave (dotted), and white Gaussian noise (dashed).

5045.27 Ω, $\hat{R}_4 = 3001.98\ \Omega$, $\hat{C}_1 = 3.22$ nF, all within $\pm 1\%$ of the true values (see Table 1). Remarkably, this range is compatible with the tolerance of most capacitors and carbon- or metal-film resistors commercially available. Overall, we report $\epsilon(\boldsymbol{\theta}^\star) = 8.53 \times 10^{-5}$ for an SNR of 20 dB.

In order to stress the method robustness to noise, we repeat the same experiment considering an SNR of 6 dB. The resulting exploration of the parameter space is shown in Fig. 10, where it can be seen that the estimates rapidly converge to $\hat{R}_1 = 10831\ \Omega$, $\hat{R}_2 = 3094\ \Omega$, $\hat{R}3 = 11311\ \Omega$, $\hat{R}_4 = 8050\ \Omega$, $\hat{C}_1 = 8.81$ nF, i.e., far off from the ground truth values. Indeed, this configuration yields $\epsilon(\boldsymbol{\theta}^\star) = 5.54$, i.e., orders of magnitude larger than the NMSE reported for an SNR of 20 dB. Ultimately, such an error indicates limited noise robustness in the face of a significant degradation of the observable signals.

B. LINEAR TIME-DOMAIN MODELS: THIELE-SMALL ELECTRODYNAMIC LOUDSPEAKER MODEL

Let us now consider the Thiele-Small equivalent model of a boxed dynamic loudspeaker [36], whose circuit schematic is given in Fig. 4. As opposed to the Bridged-T network discussed in the previous section, the Thiele-Small model represents a multiphysics system by means of an electrical equivalent and contains multiple dynamical elements. The electromechanical coupling is realized via a gyrator implemented in the WD domain as in [40], whereas the whole circuit is implemented by making use of linear WDFs [38]. The electrical parameters of the model are $R_{\text{e}} = 3.33\ \Omega$ and $L_{\text{e}} = 0.23$ mH, i.e., the resistance and the inductance of the coil, respectively. The mechanical parameters describing a dampened harmonic oscillator are characterized by the capacitance $C_{\text{ms}} = 490.82\ \mu$F modeling a massless ideal spring, the inductance $L_{\text{ms}} = 11.012$ mH modeling the mass of the coil and loudspeaker diaphragm, and the resistance $R_{\text{ms}} = 1.039\ \Omega$ accounting for frictions and other dissipative effects. Finally, the magnetic contribution is reduced to a constant force factor $Bl = 4.15\ \Omega$. In this scenario, we aim to apply the proposed framework to learn electrical, mechanical, and magnetic parameters jointly.

We minimize the NMSE loss function in (7) for 12 000 epochs. We assume to be able to measure only the Kirchhoff variables on R_e and R_{ms}, i.e., one pair for the electrical part, and one for the mechanical part ($M = 4$). Hence, the vector of observables at time k can be expressed as

$$\mathbf{y}_k := \left[v_{R_{\text{e}}}[k], v_{R_{\text{ms}}}[k], i_{R_{\text{e}}}[k], i_{R_{\text{ms}}}[k]\right]^{\text{T}}. \tag{11}$$

As an initial guess for the learnable circuit parameters, we choose 10 Ω for the resistances, 1 mH for the inductances, 1 μF for the capacitances, and 1 Ω for the force factor Bl. We feed the Thiele-Small circuit model with a low-frequency sinusoidal input $u[k] = A\sin(2\pi f_0 k)$ of duration 1.6 ms, where $A = 1$ V, $f_0 = 50$ Hz, and k is the discrete-time index.

At each iteration, we employ three Adam optimizers subsequently invoked one at a time. The first updates the resistances with learning rate $\alpha_{\text{R}} = 10^2$. The second updates the force factor Bl with learning rate $\alpha_{Bl} = 10^{-2}$. The third updates all the parameters with learning rate $\alpha_{\text{all}} = 5 \times 10^{-5}$. We observed that this learning rate configuration causes the minimization to become unstable after a few hundred epochs. The training loss would indeed start to chaotically oscillate around a steady-state value. One choice was to select different initialization values for all learning rates. However, this would have led to an overall slower convergence rate. As

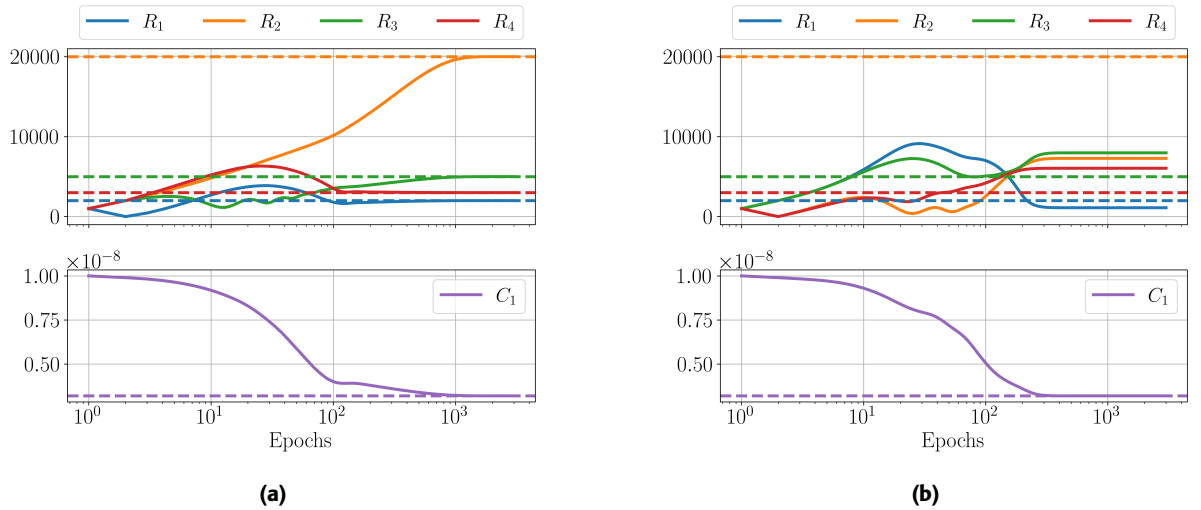

FIGURE 8. Exploration of the parameter space of the Bridged-T network (a) observing C_1 and R_4; (b) observing only C_1.

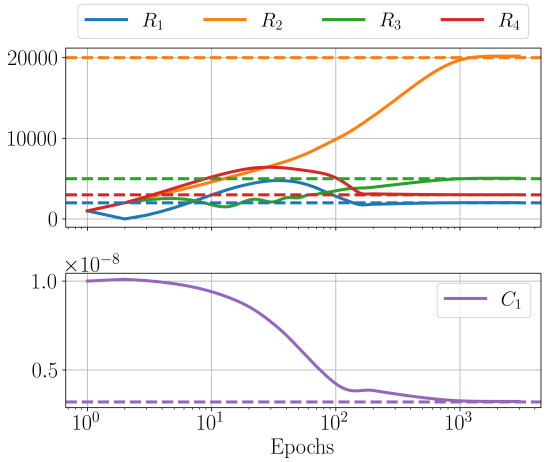

FIGURE 9. Exploration of the parameter space of the Bridged-T network observing C_1 and R_4 whose measurements were corrupted by AWGN taking into account an SNR of 20 dB.

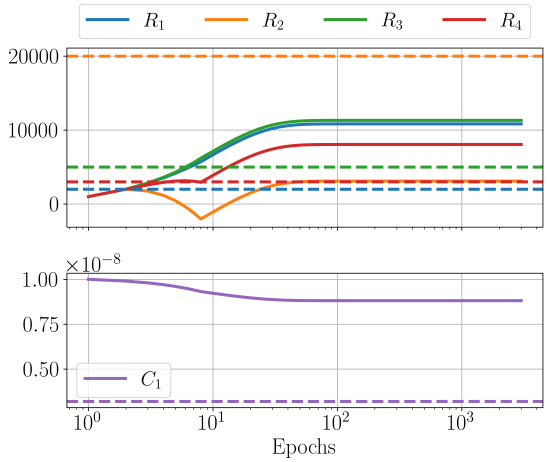

FIGURE 10. Exploration of the parameter space of the Bridged-T network observing C_1 and R_4 whose measurements were corrupted by AWGN taking into account an SNR of 6 dB.

an alternative, we opted for reducing the learning rates by a factor of ten every time the training loss would not decrease for ten consecutive epochs. The resulting loss curves can be seen in Fig. 12, along with the corresponding learning rate dynamics. The spikes in the loss function occurring roughly at epochs 4000 and 9000 trigger the learning rate reduction, thus avoiding entering the aforementioned unstable regime. The optimization results are summarized in Table 2, which, overall, correspond to $\epsilon(\boldsymbol{\theta}^\star) = 2.63 \times 10^{-8}$. Among all estimates, only C_{ms} presents an appreciable deviation from the true value (by approximately 4.5%). The initialization of C_{ms}, however, was the furthest from the ground truth, requiring to span two orders of magnitude to converge at the true value. Hence, we expect that longer training and additional fine-tuning of the learning rates would result in a perfect match.

TABLE 2. Thiele-Small Model (12000 Epochs).

	R_{e} (Ω)	L_{e} (mH)	C_{ms} (μF)	L_{ms} (mH)	R_{ms} (Ω)	Bl (Ω)
Ground truth	3.33	0.23	490.82	11.012	1.039	4.15
Initial guess	10	1	1	1	10	1
Estimated	3.329	0.23	468.87	11.005	1.039	4.15

C. LINEAR FREQUENCY-DOMAIN MODELS

In Section IV-A, we explored the capability of the proposed framework to match the multiphysics characteristics of a target linear time-domain system. However, many real-world phenomena exhibit frequency-dependent behaviors, which are best described in the Laplace domain. In this section,

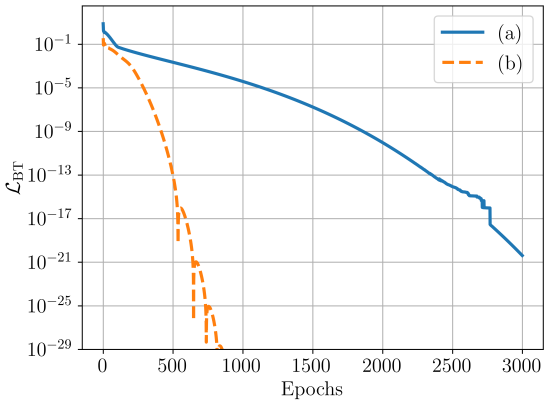

FIGURE 11. **Trend of Bridged-T network loss functions (a) observing C_1 and R_4; (b) observing only C_1.**

we investigate the use of AD to learn an equivalent LEM by fitting the complex-valued transfer function $H(s)$ of a target system. This can be achieved, e.g., by solving (3) with $\mathbf{y} = [H(\omega_1), ..., H(\omega_K)]^{\mathrm{T}}$ and $\hat{\mathbf{y}} = [\hat{H}(\omega_1), ..., \hat{H}(\omega_K)]^{\mathrm{T}}$, where $H(\omega)$ is the target transfer function evaluated on the $j\omega$ axis, and $\hat{H}(\omega)$ is that of the chosen LEM, which is determined by the model parameters $\boldsymbol{\theta}$.

In general, when the analytical expression of the frequency response is known, it can be explicitly included in the forward pass of the algorithm. Alternatively, one could excite the LEM with an impulse and compute the Fourier transform of the observable outputs. In both cases, the proposed framework entails an optimization based on complex gradients that can be performed by means of Wirtinger calculus [41]. In the following, we will consider three different loss functions, given in (12), (15), and (16), respectively.

First, let us consider

$$\mathcal{L}_{\mathrm{B}} := \mathrm{MSE}_{|\cdot|} + \mathrm{MSE}_{\phi}, \tag{12}$$

where

$$\mathrm{MSE}_{|\cdot|} := \sum_k \Big(\big|H(\omega_k)\big| - \big|\hat{H}(\omega_k)\big| \Big)^2, \tag{13}$$

$$\mathrm{MSE}_{\phi} := \sum_k \Big(\angle H(\omega_k) - \angle \hat{H}(\omega_k) \Big)^2. \tag{14}$$

In particular, (13) describes the mean squared error (MSE) between the true and estimated magnitudes, and (14) describes the MSE between true and estimated phases.

Second, let us consider the logarithmic energy function proposed in [42]:

$$\begin{aligned}\mathcal{L}_{\mathrm{ln}} &:= \sum_k \Big(\ln \hat{H}(\omega_k) - \ln H(\omega_k) \Big) \Big(\ln \hat{H}(\omega_k) - \ln H(\omega_k) \Big)^* \\ &= \sum_k \ln \left(\frac{\big|\hat{H}(\omega_k)\big|}{\big|H(\omega_k)\big|} \right)^2 + \Big(\angle \hat{H}(\omega_k) - \angle H(\omega_k) \Big)^2,\end{aligned} \tag{15}$$

where $(\cdot)^*$ denotes the complex conjugation operator.

Finally, let us consider the complex mean squared loss function described in [43]:

$$\mathcal{L}_{\mathrm{c}} := \sum_k \Big(H(\omega_k) - \hat{H}(\omega_k) \Big) \Big(H(\omega_k) - \hat{H}(\omega_k) \Big)^*. \tag{16}$$

It is important to stress that (12), (15), and (16) do not take on complex values but rather describe real-valued mappings that tend to zero as the magnitude of the complex errors decreases. However, since the AD of these losses typically entails the explicit computation of the complex-valued transfer function as part of the DAG, it can only be solved by means of complex differentiation.

By exploiting the polar form of $H(\omega)$ and $\hat{H}(\omega)$, both $\mathcal{L}_{\mathrm{B}}$ and $\mathcal{L}_{\mathrm{ln}}$ expose the magnitude and phase quantities. Therefore, (12) and (15) might not require the explicit differentiation of the complex-valued transfer function. Indeed, if the frequency response can be analytically derived, one might define parallel branches for magnitude/phase or real/imaginary parts in the control flow of the program. In such cases, $\mathcal{L}_{\mathrm{B}}$ and $\mathcal{L}_{\mathrm{ln}}$ may be preferred over $\mathcal{L}_{\mathrm{c}}$ as they would enable standard real-valued AD. In the next section, however, we focus on complex gradient-based optimization as it encompasses the most general use case.

D. LINEAR FREQUENCY-DOMAIN MODELS: LOWPASS SALLEN-KEY FILTER

Let us consider the Bode plots in Fig. 13 describing the transfer function $H(s)$ of a target linear time-invariant system. $H(s)$ is characterized by a lowpass frequency response with a -40 dB/decade roll-off rate in the audio bandwidth. Therefore, we could think of approximating it with a second-order active filter. In this example, we consider the lowpass Sallen-Key topology [44] depicted in Fig. 5, i.e., a simple two-pole voltage-controlled voltage source filter. The transfer function of the Sallen-Key filter is given by

$$\hat{H}(s) = \frac{1 + \frac{R_4}{R_3}}{s^2 \left(R_1 R_2 C_1 C_2\right) + s \left((R_1 + R_2) C_2 - \frac{R_1 R_4 C_1}{R_3} \right) + 1}. \tag{17}$$

Our objective is to learn the parameters R_1, R_2, R_3, R_4, C_1, C_2 so that the estimated $\hat{H}(s)$ matches the observable $H(s)$. To this end, we initialize the resistances to 1 kΩ and the capacitances to 1 nF. We train for 3000 epochs using two Adam optimizers with learning rates 10^3 and 10^{-10}, respectively. The first optimizer acts on the four resistors, whereas the second updates all circuit elements at the same time. Throughout the epochs, both learning rates are decreased according to a cosine annealing schedule (without restarts) [45], as it was observed to provide increased stability during late training. As a loss function, we choose either $\mathcal{L}_{\mathrm{B}}$ (12), $\mathcal{L}_{\mathrm{ln}}$ (15), or $\mathcal{L}_{\mathrm{c}}$ (16). The comparison between the three losses is shown in Fig. 14: $\mathcal{L}_{\mathrm{ln}}$ plateaus at around 4×10^{-12}, whereas both $\mathcal{L}_{\mathrm{B}}$ and $\mathcal{L}_{\mathrm{c}}$ reach values well below practical machine precision. Among the three, $\mathcal{L}_{\mathrm{c}}$ appears to achieve faster convergence. Ultimately, these results suggest that complex differentiation may be regarded as a particularly

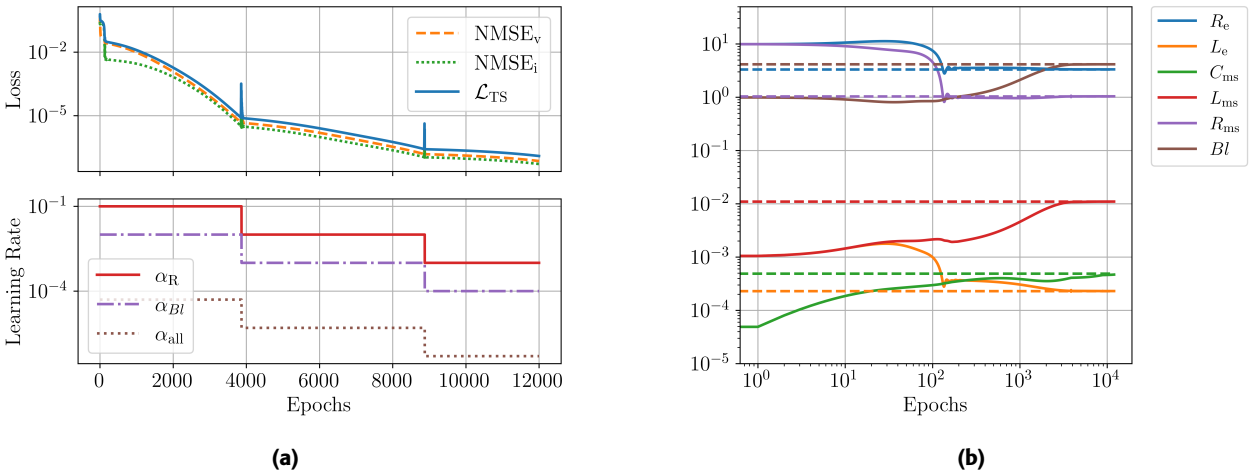

FIGURE 12. **Thiele-Small model observing R_e and R_{ms}. (a) Loss and learning rate dynamics; (b) exploration of the parameter space of the model.**

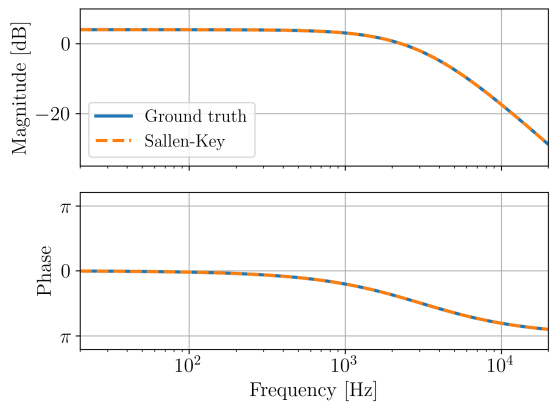

FIGURE 13. **Target (in blue) and estimated (in orange) Bode plots of the complex-valued transfer function $H(s)$.**

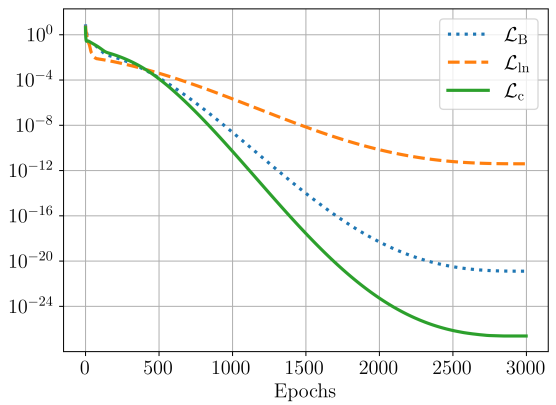

FIGURE 14. **Comparison among different loss functions for the parameter estimation of the lowpass Sallen-Key filter.**

suitable strategy for frequency-domain learning objectives. Notably, there exist infinitely many parameter configurations of a Sallen-Key topology yielding a perfect match of the target transfer function. Therefore, the method would converge to a different equivalent circuit depending on the loss function and the chosen parameter initialization. The Bode plots of the filter learned using $\mathcal{L}_c$ as the loss function are displayed in Fig. 13 against the ground truth, pointing out the accuracy of the estimated response.

E. NONLINEAR TIME-DOMAIN MODELS

In the previous sections, we focused on linear systems. However, many interesting physical systems exhibit nonlinear characteristics. In this section, we show that the proposed framework can be seamlessly applied to the estimation of both the electrical and constructive parameters of the elements of a target nonlinear circuit.

In the WD domain, circuit elements are modeled by separate input/output blocks. This makes WDFs a suitable framework for dealing with nonlinearities as they can be typically handled locally. In particular, the waves reflected from all individual elements can be computed through parallel one-dimensional nonlinear solvers, such as the Newton-Raphson (NR) algorithm, as done in [37]. In our case study, however, this has the unwanted side effect of enlarging the AD graph as the number of iterations increases. In fact, the gradient with respect to the nonlinear element parameters would have to be backpropagated through the unfolding of every iteration cycle of the NR algorithm. In turn, this may lead to a dramatic increase in training time and result in well-known gradient flow pathologies.

An alternative approach is to find explicit wave mappings for the target nonlinearities such as, e.g., Canonical Piece-Wise Linear (CPWL) representations [46]. In fact, despite not circumventing the use of global iterative methods, CPWL functions eliminate the need for a dedicated solver for each nonlinear element. In the following, we base our CPWL implementation on [47], [48].

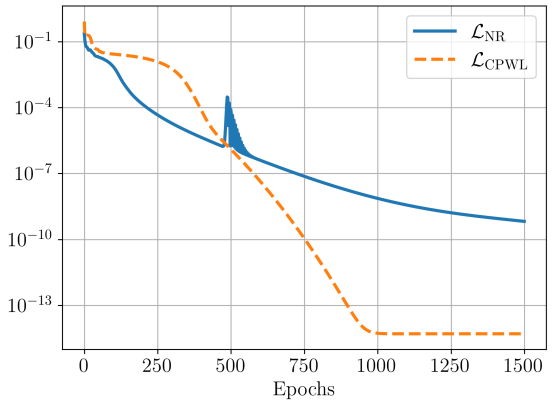

FIGURE 15. Comparison between $\mathcal{L}_{\mathrm{CPWL}}$ and $\mathcal{L}_{\mathrm{NR}}$ having fixed $V_{\mathrm{t}} = 26$ mV.

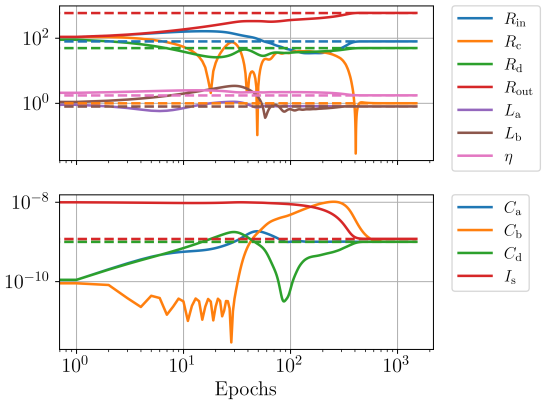

FIGURE 16. Exploration of the parameter space of the CPWL-based implementation of the dynamic ring modulator circuit with $V_{\mathrm{t}} = 26$ mV.

F. NONLINEAR TIME-DOMAIN MODELS: DYNAMIC RING MODULATOR

As an illustrative example, let us consider the dynamic ring modulator depicted in Fig. 6 containing multiple dynamic elements and nonlinear diodes [38]. We assume all four diodes D_1, D_2, D_3, D_4 to be characterized by the *extended Shockley diode model* described by the nonlinear implicit equation [38]

$$g(v,i) = I_{\mathrm{s}}\left(\exp\left(\frac{v - R_{\mathrm{s}}i}{\eta V_{\mathrm{t}}}\right) - 1\right) + \frac{v - R_{\mathrm{s}}i}{R_{\mathrm{p}}} - i = 0, \quad (18)$$

where I_s is the saturation current, η is the ideality factor, V_t is the thermal voltage, and R_s and R_p are the series resistance and the shunt resistance of the p-n junction, respectively. Our implementation refers to the RFN5BM6S silicon diode. Namely, we set $I_s = 1.186$ nA, $\eta = 1.7549$, $V_{\mathrm{t}} = 26$ mV, $R_{\mathrm{s}} = 1$ mΩ, and $R_{\mathrm{p}} = 100$ kΩ. The other parameters of the circuit are $R_{\mathrm{in}} = 80\ \Omega$, $R_{\mathrm{out}} = 600\ \Omega$, $R_{\mathrm{c}} = 1\ \Omega$, $R_{\mathrm{d}} = 50\ \Omega$, $L_{\mathrm{a}} = L_{\mathrm{b}} = 0.8$ H, and $C_{\mathrm{a}} = C_{\mathrm{b}} = C_{\mathrm{d}} = 1$ nF. The turn ratios of all ideal transformers are set to $1/2$. The given circuit is fed with two inputs: a carrier signal, modeled by the voltage source V_{c}, and a modulating signal, modeled by the generator V_{in}. Here, we use two sine waves with a duration of 1 ms at a sampling rate of 44.1 kHz and frequencies $f_{\mathrm{c}} = 50$ Hz and $f_{\mathrm{in}} = 150$ Hz, respectively.

Considering the extended Shockley equation in (18), we can readily notice that the product of the ideality factor η and the thermal current V_{t} appears in the argument of the exponential function. Therefore, it is impossible to unambiguously determine both parameters at the same time, as any scalar multiplier in one term can be canceled out by multiplying the other by the reciprocal. A principled choice for the initial values of η and V_{t} may lead to discovering physically meaningful values, even if the problem remains undeterminable. Alternatively, one might assume to know the operating temperature T of the device. In this case, the thermal voltage can be set to $V_{\mathrm{t}} = k_{\mathrm{B}}T/q$, where k_{B} is the Boltzmann constant and q is the elementary charge, thus making the problem converge to a unique solution. In the following, we consider this latter case by fixing $V_{\mathrm{t}} = 26$ mV (corresponding to a temperature of approximately 28.6 °C) and observing port voltages and currents as done in Section IV-A. Specifically, the loss function is formulated as in (7) with $M = 13$. The initial guess for the circuit parameters is as follows: 100 Ω for all resistances, 1 H for all inductances, 10 pF for all capacitances, 10 nA for the saturation current, and 2 for the ideality factor. We implement the circuit in the WD domain considering, as far as the modeling of the four diodes is concerned, both the approach based on CPWL functions [47] and NR solvers [38]. We refer to the corresponding loss functions as $\mathcal{L}_{\mathrm{CPWL}}$ and $\mathcal{L}_{\mathrm{NR}}$, respectively.

In this example, instead of tuning the learning rates according to prior assumptions on the orders of magnitude of the circuit parameters, we incorporate this knowledge directly into the model. We assume that every capacitance will be in the range of a few nanofarads and that the saturation current will be in the range of a few nanoamperes. Therefore, C_{a}, C_{b}, C_{d}, and I_{s} are multiplied by 10^{-9} as they enter the execution flow of the program. This means that we are, in fact, learning a scaled version of those parameters that takes value in a range comparable to that of the other circuit parameters. Ultimately, this gave us more freedom in choosing the hyperparameters. We train both models by means of two alternating Adam optimizers, one updating the resistances with learning rate $\alpha_{\mathrm{R}} = 10$, and the other updating all circuit elements with a learning rate $\alpha_{\mathrm{all}} = 0.1$. We empirically observed that the optimization of the NR-based model enters an unstable regime after a few hundred epochs. Instead of reducing the learning rates globally, we introduce a policy for dynamically decreasing the learning rates when hitting a plateau as described in Section IV-B. Conversely, no learning rate schedule was applied to the CPWL-based model as it proved to needlessly slow down the learning process.

Fig. 15 shows the ensuing behavior of $\mathcal{L}_{\mathrm{CPWL}}$ and $\mathcal{L}_{\mathrm{NR}}$ over 1500 epochs. On the one hand, $\mathcal{L}_{\mathrm{NR}}$ rapidly decreases in the first few epochs. Then, the chaotic oscillation noticeable around epoch 500 triggers the learning rate reduction by a

TABLE 3. Dynamic Ring Modulator (1500 Epochs); NR (Newton-Raphson) and CPWL (Canonical PieceWise Linear) indicate two different implementations of nonlinear elements in the Wave Digital domain.

	R_{in} (Ω)	R_{c} (Ω)	R_{d} (Ω)	R_{out} (Ω)	L_{a} (H)	L_{b} (H)	C_{a} (nF)	C_{b} (nF)	C_{d} (nF)	I_{s} (nA)	η	V_{t} (mV)
Ground truth	80	1	50	600	0.8	0.8	1	1	1	1.186	1.7549	26
Initial guess	100	100	100	100	1	1	0.01	0.01	0.01	10	2	25
	Trainable V_{t}											
NR	80.00088	1.00042	49.9999	600.009	0.7999	0.8000025	1.0000019	1.0214578	1.0000067	1.1862	1.8429	24.7587
CPWL	80.00001	1.0000076	50	600	0.8	0.8	0.9999993	1.0000005	0.999999	1.1859	1.8465	24.7099
	Fixed V_{t}											
NR	80.00118	1.00056	49.9999	600.012	0.7999	0.8000035	1.0000025	1.0287621	1.000009	1.1862	1.7549	—
CPWL	80.00001	1.0000076	50	600	0.8	0.8	0.9999993	1.0000004	0.999999	1.1859	1.7549	—

TABLE 4. NMSE between true and estimated parameters for the diode-based Dynamic Ring Modulator model.

	Trainable V_{t}	Fixed V_{t}
NR	2.25×10^{-10}	3.99×10^{-10}
CPWL	1.99×10^{-13}	4.93×10^{-18}

TABLE 5. Average time-per-epoch ($\pm$ standard deviation) of every model considered in the present study.

	Average time-per-epoch [ms]	Epochs
Bridged-T Network	68.2 ± 4.2	3000
Thiele-Small Model	155.1 ± 39.7	12 000
Sallen-Key Filter ($\mathcal{L}_{\mathrm{B}}$)	2.6 ± 0.7	3000
Sallen-Key Filter ($\mathcal{L}_{\mathrm{ln}}$)	2.2 ± 0.7	3000
Sallen-Key Filter ($\mathcal{L}_{\mathrm{c}}$)	1.9 ± 0.7	3000
Dynamic Ring Modulator (NR)	1413.9 ± 169.3	1500
Dynamic Ring Modulator (CPWL)	1050.7 ± 40.3	1500

factor ten. Afterward, the loss decreases at a steady pace reaching 6.61×10^{-10}. On the other hand, $\mathcal{L}_{\mathrm{CPWL}}$ appears to decrease slower than $\mathcal{L}_{\mathrm{NR}}$ during early training. However, after such an initial phase, the loss of the CPWL-based model exponentially drops before settling at around 5.08×10^{-15}. Table 3 summarizes the estimated parameters in the two cases. Arguably, both the NR- and CPWL-based implementations allow one to fit the true circuit with negligible deviation from the ground truth values, yielding remarkably small values of $\epsilon(\boldsymbol{\theta}^{\star})$ in both cases, as shown in Table 4. The exploration of the parameter space of the CPWL-based model is depicted in Fig. 16, wherein, after 1000 epochs, all the parameters have reached the respective ground-truth values.

For completeness, let us now include V_{t} among the trainable parameters and choose 25 mV as an initial guess. Similar to what was done for the capacitances and the saturation current, the thermal voltage estimate is multiplied by 10^{-3} to scale it in the same range of the other parameters. Thus, the learning algorithm described above is run again for 1500 epochs. The results in Table 3 show that the only indeterminate predictions are those relative to η and V_{t}, as expected. Notably, the values of all the remaining passive and dynamic elements are almost perfectly retrieved.

G. COMPUTATIONAL TIME

In Table 5, we report the time-per-epoch measured for every LEM considered in the present study. These figures are obtained by averaging the elapsed time of 1000 single-batch epochs run on a Intel Xeon E5-2687W v4. The time-per-epoch of each LEM depends on several factors including the number of input samples, the number of trainable parameters, and the program efficiency in terms of differentiable operations. Unsurprisingly, the slowest model to fit is the diode-based dynamic ring modulator, whose solution requires the application of a global iterative method (Scattering Iterative Method [37], in our case) due to presence of nonlinear elements. As a result, the number of differentiable operations involved in the AD backward pass increases with every iteration. In addition, the computational time increases further when nonlinearities are implemented using dedicated NR solvers, whose iterations adds even more operations to the DAG. Indeed, the NR-based implementation of the dynamic ring modulator takes, on average, 1.41 s per epoch, whereas the CPWL-based implementation only takes 1.05 s. Conversely, that of the lowpass Sallen-Key filter stands out as the quickest optimization routine, with training epochs clocking in at under 2 milliseconds when employing the complex mean squared loss function $\mathcal{L}_{\mathrm{c}}$. This efficiency is a result of the filter operating in the frequency domain, where it can obtain the transfer function estimate in a single forward pass, unlike time-domain models that rely on a sample-by-sample simulation mechanism.

V. CONCLUSIONS AND FUTURE WORK

In this article, we presented a general data-driven framework for estimating the parameters of physical systems that can be formulated as lumped-element models. Our approach relies on automatic differentiation, a well-understood technique for the error-free gradient evaluation of functions expressed as non-standard computer programs, which, to date, has received relatively little attention outside of off-the-shelf deep learning applications. We argued that the input-output mappings characterizing LEMs could be learned via the backpropagation of some error metric between target and observable physical quantities. Unlike neural networks, though, the proposed

model-based paradigm is explicit and fully interpretable as it exposes the underlying topology of the system and learns a physically-consistent set of parameters. Moreover, since the forward architecture incorporates the model in full, inference is unlikely to suffer from out-of-distribution and generalization issues. In all the experiments conducted on several linear and nonlinear systems, both in the time and frequency domain, we reported a near-perfect match of the desired behavior, and, when considering the right set of observables, we were able to retrieve the true lumped-parameter values with a remarkable degree of accuracy. We showed that a principled choice of the hyperparameters informed by the underlying physical constraints and best practices borrowed from the deep learning field ensure reliable and fast convergence rates.

Optimal results were achieved using just a few milliseconds of system measurements. Therefore, the present approach appears very promising in all those scenarios in which extensive domain knowledge exists, and yet data are inherently scarce or too expensive to simulate using, e.g., finite element methods (FEM). Mechanics, vibroacoustics, and fluid dynamics, to name a few, often rely on accurate but lengthy multiphysics solvers that are unsuitable for real-time simulation. Researchers and practitioners in those fields may thus benefit from the adoption of the present framework by using existing FEM models to generate the minimal amount of training data needed to learn an equivalent LEM via AD. Furthermore, such a data-driven optimization scheme may allow one to compensate for the long- and short-term deviations in operating conditions that may naturally occur during the lifecycle of a device by correcting the initial parameter estimates according to online measurements.

Ultimately, the strength of the proposed method resides in the a-priori choice of the approximating lumped model. However, handcrafted LEMs of possibly distributed systems might prove too simplistic to accurately capture real-world physical behaviors. To address this potential issue, future developments will include learning the model structure alongside its parameters, e.g., by including network topology matrices among the trainable parameters. Moreover, the present framework could be easily applied to the discovery of nonlinear inverse systems by enforcing an identity mapping between the input and the output of a series of direct and learnable models. Finally, the proposed learning framework could be readily adapted to hybrid designs in which an explicit arrangement of lumped elements interfaces with implicit neural network models accounting for those physical aspects that cannot be fully modeled otherwise.

REFERENCES

[1] S. A. Billings and S. Fakhouri, "Identification of systems containing linear dynamic and static nonlinear elements," *Automatica*, vol. 18, no. 1, pp. 15–26, 1982.

[2] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. R.E. Krieger Publishing Company, 1989.

[3] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, 1990.

[4] X. Jia, J. Willard, A. Karpatne, J. S. Read, J. A. Zwart, M. Steinbach, and V. Kumar, "Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles," *ACM/IMS Trans. Data Science*, vol. 2, no. 3, pp. 1–26, 2021.

[5] 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, 2021.

[6] L. Ljung, *System identification*. Springer, 1998.

[7] R. C. Aster, B. Borchers, and C. H. Thurber, *Parameter Estimation and Inverse Problems*. Elsevier, 2019.

[8] A. Verma, "An introduction to automatic differentiation," *Curr. Sci.*, pp. 804–807, 2000.

[9] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[10] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *J. Mach. Learn. Res.*, vol. 18, pp. 1–43, 2018.

[11] R. Frye, E. Rietman, and C. Wong, "Back-propagation learning and nonidealities in analog neural network hardware," *IEEE Trans. Neural Netw.*, vol. 2, no. 1, pp. 110–117, 1991.

[12] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon, "Deep physical neural networks trained with backpropagation," *Nature*, vol. 601, no. 7894, pp. 549–555, 2022.

[13] P. Feldmann, R. Melville, and S. Moinian, "Automatic differentiation in circuit simulation and device modeling," in *Proc. of the 1992 IEEE/ACM Int. Conf. on Computer-Aided Design*, 1992, pp. 248–253.

[14] L. Kolonko, J. Velten, and A. Kummert, "An improved multi-dimensional approach to wave digital filters with topology-related delay-free loops using automatic differentiation," in *Proc. of the 62nd Int. Midwest Symp. on Circuits and Syst.*, 2019, pp. 1163–1166.

[15] ——, "Automatic differentiating wave digital filters with multiple nonlinearities," in *Proc. of the 28th Eur. Signal Process. Conf.*, 2021, pp. 146–150.

[16] M. Shintani, A. Ueda, and T. Sato, "Accelerating parameter extraction of power mosfet models using automatic differentiation," *IEEE Trans. Power Electron.*, vol. 37, no. 3, pp. 2970–2982, 2022.

[17] M. Shintani, M. Hiromoto, and T. Sato, "Efficient parameter-extraction of spice compact model through automatic differentiation," in *Proc. of the 2018 IEEE Int. Conf. on Microelectron. Test Struct.*, 2018, pp. 35–40.

[18] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Proc. Int. Conf. on Learning Representations*, 2020.

[19] F. Esqueda, B. Kuznetsov, and J. D. Parker, "Differentiable white-box virtual analog modeling," in *Proc. of the 23rd Int. Conf. on Digital Audio Effects*, 2021, pp. 41–48.

[20] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020.

[21] R. Giampiccolo, A. Bernardini, O. Massi, and A. Sarti, "On the virtualization of audio transducers," *Sensors*, vol. 23, no. 11, 2023.

[22] R. E. Wengert, "A simple automatic derivative evaluation program," *Commun. ACM*, vol. 7, no. 8, pp. 463–464, 1964.

[23] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT Numer. Math.*, vol. 16, no. 2, pp. 146–160, 1976.

[24] D. E. Rumelhart, J. L. McClelland, P. R. Group *et al.*, *Parallel distributed processing*. IEEE New York, 1988, vol. 1.

[25] R. Frostig, M. J. Johnson, and C. Leary, "Compiling machine learning programs via high-level tracing," *Syst. Mach. Learn.*, vol. 4, no. 9, 2018.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 8024–8035.

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

[28] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *J. Mach. Learn. Res.*, vol. 12, no. 7, 2011.
[29] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv:1212.5701*, 2012.
[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of the 3rd Int. Conf. on Learning Representations*, Y. Bengio and Y. LeCun, Eds., 2015.
[31] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *Proc. of the 6th Int. Conf. on Learning Representations*, 2018.
[32] A. Krogh and J. Hertz, "A simple weight decay can improve generalization," in *Adv. Neural Inf. Process. Syst.*, vol. 4, 1991.
[33] G. Borin, G. De Poli, and D. Rocchesso, "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 5, pp. 597–605, 2000.
[34] C. W. Ho, A. E. Ruehli, and P. A. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits Syst.*, vol. 22, pp. 504–509, 1975.
[35] A. Fettweis, "Wave digital filters: Theory and practice," *Proc. of the IEEE*, vol. 74, pp. 270–327, 1986.
[36] A. Falaize and T. Hélie, "Passive modelling of the electrodynamic loudspeaker: From the Thiele–Small model to nonlinear port-Hamiltonian systems," *Acta Acustica*, vol. 4, p. 1, 2020.
[37] A. Bernardini, P. Maffezzoni, L. Daniel, and A. Sarti, "Wave-based analysis of large nonlinear photovoltaic arrays," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, pp. 1363–1376, 2018.
[38] A. Bernardini, P. Maffezzoni, and A. Sarti, "Linear multistep discretization methods with variable step-size in nonlinear wave digital structures for virtual analog modeling," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 27, pp. 1763–1776, 2019.
[39] A. Bernardini, K. J. Werner, J. O. Smith, and A. Sarti, "Generalized wave digital filter realizations of arbitrary reciprocal connection networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, pp. 694–707, 2019.
[40] K. J. Werner, A. Bernardini, J. O. Smith, and A. Sarti, "Modeling circuits with arbitrary topologies and active linear multiports using wave digital filters," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, pp. 4233–4246, 2018.
[41] R. Remmert, *Theory of complex functions*. Springer, 1991, vol. 122.
[42] R. Savitha, S. Suresh, and N. Sundararajan, "Projection-based fast learning fully complex-valued relaxation neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 529–541, 2013.
[43] J. Bassey, L. Qian, and X. Li, "A survey of complex-valued neural networks," *arXiv:2101.12249*, 2021.
[44] R. P. Sallen and E. L. Key, "A practical method of designing RC active filters," *IRE Trans. Circuit Theory*, vol. 2, no. 1, pp. 74–85, 1955.
[45] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with warm restarts," in *Proc. of the 5th Int. Conf. on Learning Representations*, 2017.
[46] L. O. Chua and S. M. Kang, "Section-wise piecewise-linear functions: Canonical representation, properties, and applications," *Proc. of the IEEE*, vol. 65, pp. 915–929, 1977.
[47] A. Bernardini and A. Sarti, "Canonical piecewise-linear representation of curves in the wave digital domain," in *Proc. of the 25th Eur. Signal Process. Conf.*, 2017, pp. 1125–1129.
[48] R. Giampiccolo, A. Bernardini, G. Gruosso, P. Maffezzoni, and A. Sarti, "Multiphysics modeling of audio circuits with nonlinear transformers," *J. Audio Eng. Soc.*, vol. 69, pp. 374–388, 2021.

ALESSANDRO ILIC MEZZA (Graduate Student Member, IEEE) received the B.S. degree and the M.S. degree (cum laude) in Computer Science and Engineering from the Politecnico di Milano, Italy, in 2015 and 2019, respectively. He is currently a Ph.D. Candidate in Information Technology at the Department of Electronics, Information, and Bioengineering at Politecnico di Milano. His main research interests include deep learning and data-driven methods for audio signal processing, anomaly detection, domain adaptation, and nonlinear systems identification and modeling.

RICCARDO GIAMPICCOLO (Member, IEEE) received both the B.S. and the M.S. degrees in Electronics Engineering from the Politecnico di Milano, Italy, in 2017 and 2020, respectively. In 2023, he received his Ph.D. degree (cum laude) in Information Technology from the Politecnico di Milano, Italy, where he is currently a postdoctoral researcher. His main research interests include audio signal processing, virtual analog modeling, and nonlinear systems.

ALBERTO BERNARDINI (Member, IEEE) received the B.S. degree from the University of Bologna, Italy, in 2012 and the M.S. degree (cum laude) from the Politecnico di Milano, Italy, in 2015, both in computer engineering. In 2019, he received the Ph.D. degree (cum laude) in information engineering from the Politecnico di Milano, where he worked as a postdoctoral researcher from 2019 to 2021. He is currently an Assistant Professor at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) of the Politecnico di Milano. His main research interests include audio signal processing, computational acoustics, and modeling of nonlinear systems. He authored more than 40 publications in international journals and proceedings of international conferences. He is the first author of two international patents. In 2019 he was a recipient of the Dimitris N. Chorafas Award. He serves as an Associate Editor for the "IEEE Transactions on Circuits and Systems I: Regular Papers" and for the "EURASIP Journal on Audio, Speech, and Music Processing."